AI & DevOps Workshop

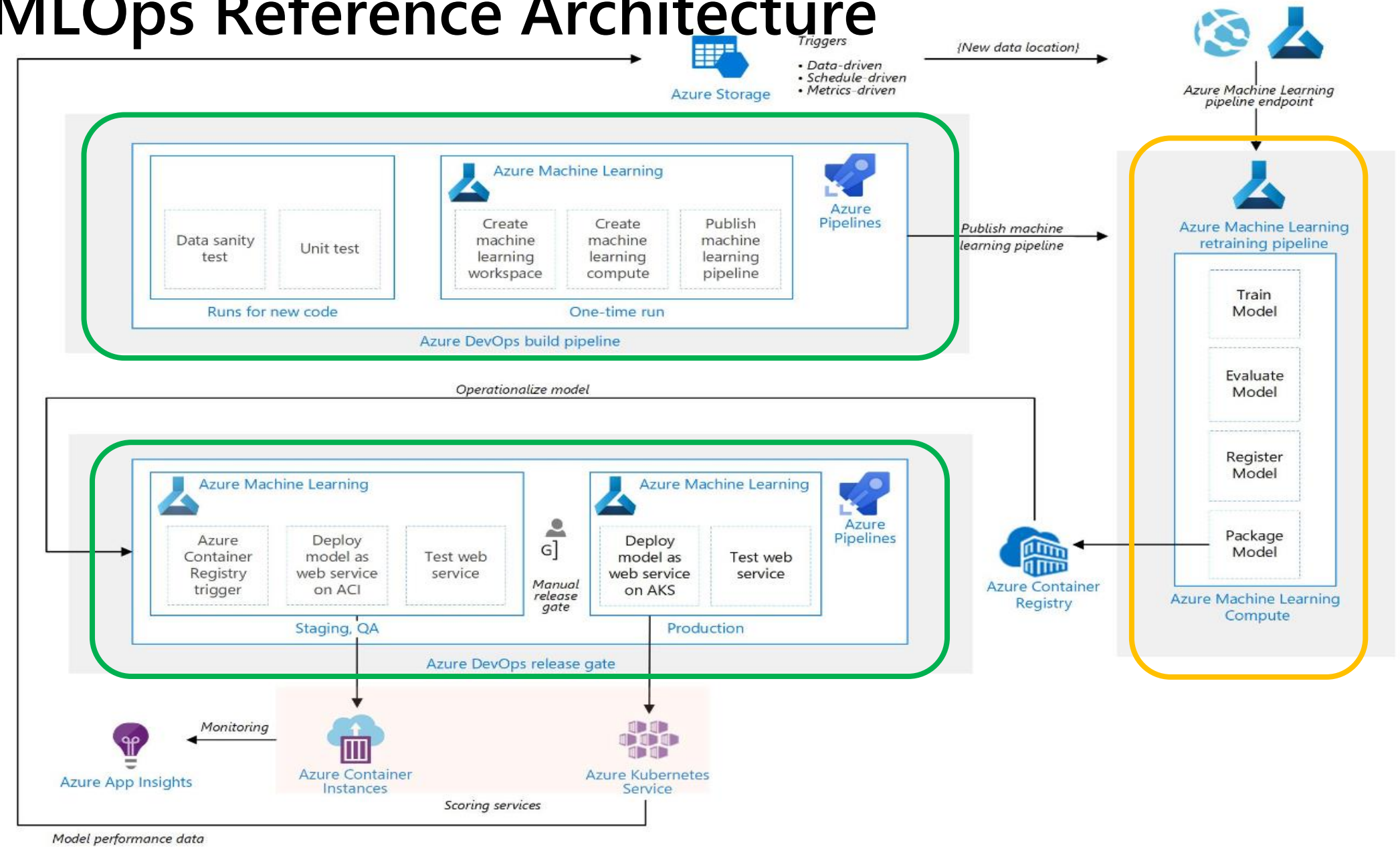
June 24, 2019

Agenda

Time	Topic	Presenter
8:30 – 9:00	Breakfast & Registration	
9:00 – 9:10	Welcome	Hyun
9:10 – 10:00	Session: Overview of AML Pipelines	Giovanni
10:00 – 11:00	Session: Intro to Lifecycle Management with Azure DevOps/YAML	Prashant
11:00 – 12:00	Session: YAML and AML Pipeline Integration	Sanjeev & Kate
12:00 – 1:00	Lunch	
1:00 – 3:30	Lab: Implement Build, Retrain, Deployment Pipelines	Kate
3:30 – 4:00	Wrap-up	

Session: Overview of Azure ML Services and Pipelines

MLOps Reference Architecture



What is Azure Machine Learning service?

Set of Azure Cloud Services

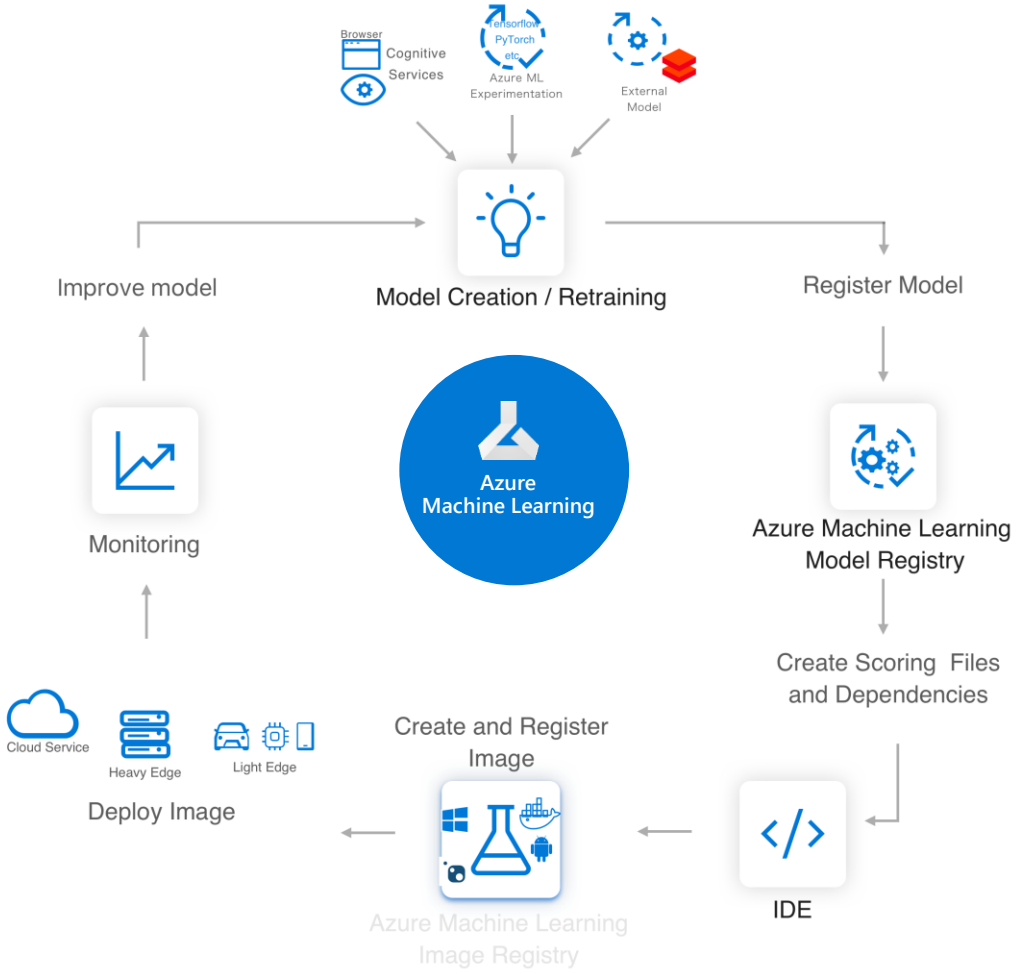
Python SDK

That enables you to:

- ✓ Prepare Data
- ✓ Build Models
- ✓ Train Models
- ✓ Manage Models
- ✓ Track Experiments
- ✓ Deploy Models

Azure ML service

Lets you easily implement this AI/ML Lifecycle



Workflow Steps

Develop machine learning training scripts in Python.

Create and configure a compute target.

Submit the scripts to the configured compute target to run in that environment. During training, the compute target stores run records to a datastore. There the records are saved to an experiment.

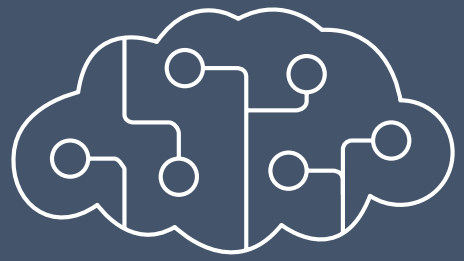
Query the experiment for logged metrics from the current and past runs. If the metrics do not indicate a desired outcome, loop back to step 1 and iterate on your scripts.

Once a satisfactory run is found, register the persisted model in the model registry.

Develop a scoring script.

Create an Image and register it in the image registry.

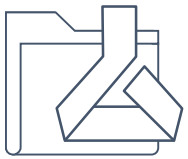
Deploy the image as a web service in Azure.



Azure Machine Learning: Technical Details

Azure ML service

Key Artifacts

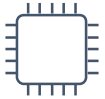


Workspace

Models

Experiments

Pipelines



Compute Target

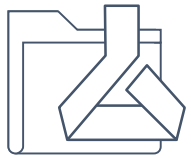
Images

Deployment

Data Stores

Azure ML service Artifact

Workspace



The workspace is the **top-level resource** for the Azure Machine Learning service. It provides a centralized place to work with all the artifacts you create when using Azure Machine Learning service.

The workspace keeps a list of compute targets that can be used to train your model. It also keeps a history of the training runs, including logs, metrics, output, and a snapshot of your scripts.

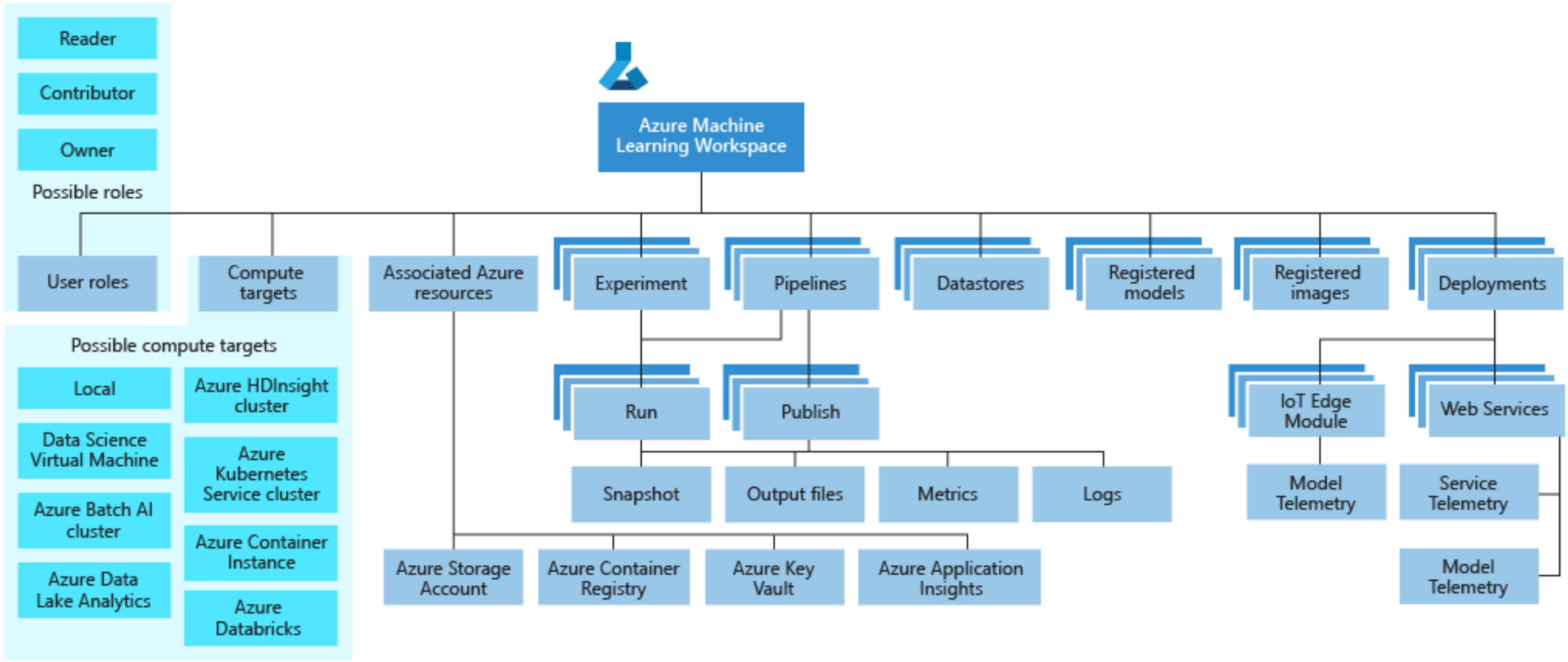
Models are registered with the workspace.

You can create multiple workspaces, and each workspace can be shared by multiple people.

When you create a new workspace, it automatically creates these Azure resources:

- Azure Container Registry - Registers docker containers that are used during training and when deploying a model.
- Azure Storage - Used as the default datastore for the workspace.
- Azure Application Insights - Stores monitoring information about your models.
- Azure Key Vault - Stores secrets used by compute targets and other sensitive information needed by the workspace.

Azure ML service Workspace Taxonomy



Azure ML service Artifacts

Models and Model Registry

Model

A machine learning model is an artifact that is created by your training process. You use a model to get predictions on new data.

A model is produced by a **run** in Azure Machine Learning.

> Note: You can also use a model trained outside of Azure Machine Learning.

Azure Machine Learning service is framework agnostic — you can use any popular machine learning framework when creating a model.

A model can be registered under an Azure Machine Learning service workspace

Model Registry

Keeps track of all the models in your Azure Machine Learning service workspace.

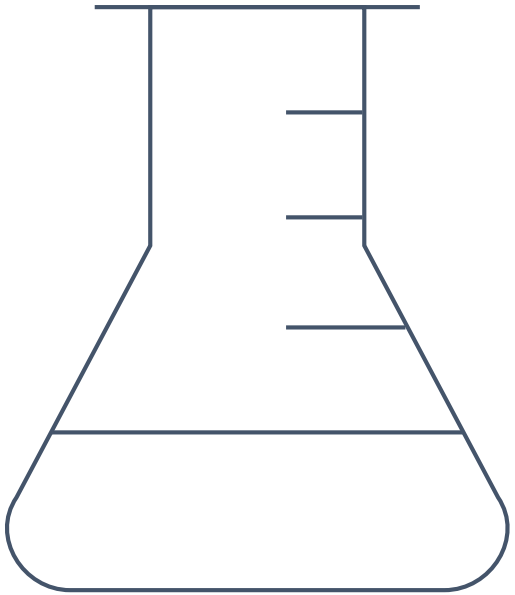
Models are identified by name and version.

You can provide additional metadata tags when you register the model, and then use these tags when searching for models.

You cannot delete models that are being used by an image.

Azure ML Artifacts

Runs and Experiments



Experiment

Grouping of many runs from a given script.

- Always belongs to a workspace.
- Stores information about runs

Run

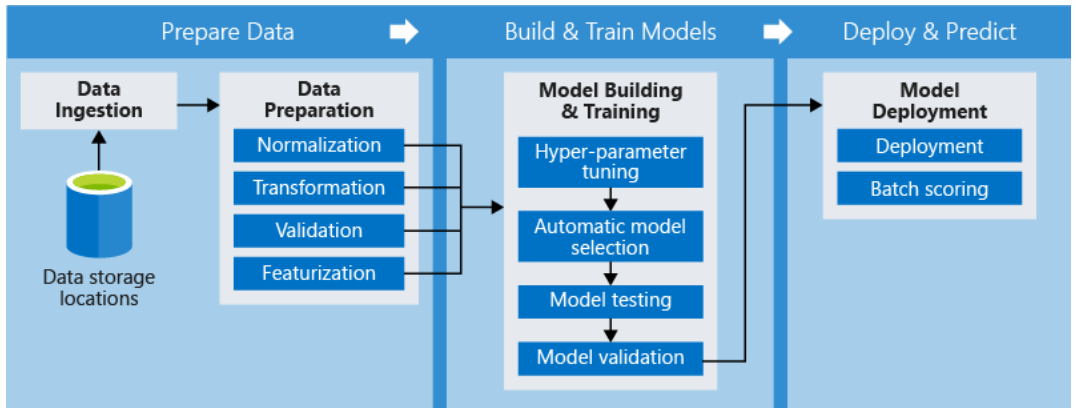
Produced when you submit a script to train a model. Contains:

- Metadata about the run (timestamp, duration etc.)
- Metrics logged by your script.
- Output files autocollected by the experiment, or explicitly uploaded by you.
- A snapshot of the directory that contains your scripts, prior to the run.

Run configuration

A set of instructions that defines how a script should be run in a given compute target.

Azure ML Artifacts
Pipelines



- Pipelines consist of steps
  - Steps can be independent
  - Or in sequence, part of a deployment

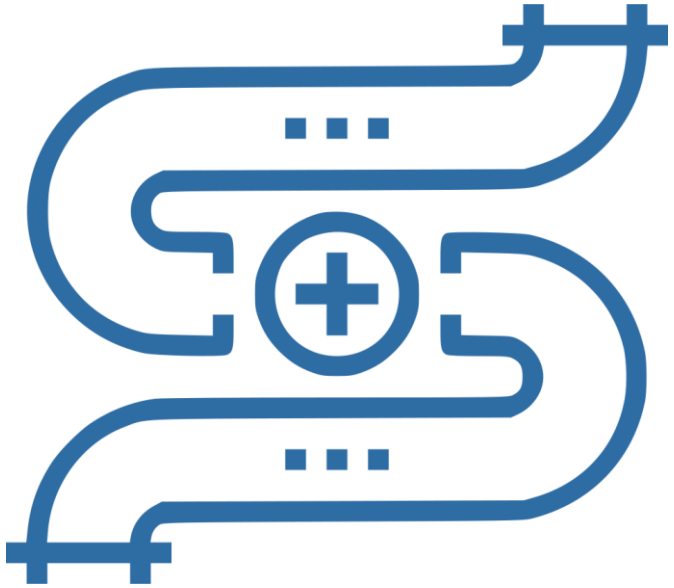
Azure ML Pipelines

Advantages

Advantage	Description
Unattended runs	Schedule a few steps to run in parallel or in sequence in a reliable and unattended manner. Since data prep and modeling can last days or weeks, you can now focus on other tasks while your pipeline is running.
Mixed and diverse compute	Use multiple pipelines that are reliably coordinated across heterogeneous and scalable computes and storages. Individual pipeline steps can be run on different compute targets, such as HDInsight, GPU Data Science VMs, and Databricks.
Reusability	Pipelines can be templatized for specific scenarios such as retraining and batch scoring. They can be triggered from external systems via simple REST calls.
Tracking and versioning	Instead of manually tracking data and result paths as you iterate, use the pipelines SDK to explicitly name and version your data sources, inputs, and outputs as well as manage scripts and data separately for increased productivity

Azure ML Pipeline

Python SDK



- Imperative constructs for sequencing or parallelizing the steps
- Declarative data dependencies
- Pre-built modules for common tasks
- Extensible
- Compute targets and storage resources can be managed directly from the SDK.
- Pipelines can be saved as templates and deployed to a REST endpoint

Azure ML Artifact

Compute Target

Compute Targets are the compute resources used to run training scripts or host your model when deployed as a web service.

They can be created and managed using the Azure Machine Learning SDK or CLI.

You can attach to existing resources.

You can start with local runs on your machine, and then scale up and out to other environments.

Currently supported compute targets

Compute Target	Training	Deployment
Local Computer	✓	
A Linux VM in Azure (such as the Data Science Virtual Machine)	✓	
Azure ML Compute	✓	
Azure Databricks	✓	
Azure Data Lake Analytics	✓	
Apache Spark for HDInsight	✓	
Azure Container Instance		✓
Azure Kubernetes Service		✓
Azure IoT Edge		✓
Field-programmable gate array (FPGA)		✓

Azure ML

Currently Supported Compute Targets

Compute target	GPU acceleration	Hyperdrive	Automated model selection	Can be used in pipelines
Local computer	Maybe		✓	
Data Science Virtual Machine (DSVM)	✓	✓	✓	✓
Azure ML compute	✓	✓	✓	✓
Azure Databricks	✓		✓	✓
Azure Data Lake Analytics				✓
Azure HDInsight				✓

https://docs.microsoft.com/en-us/azure/machine-learning/service/how-to-set-up-training-targets#supported-compute-targets

Azure ML service Artifacts

Image and Registry

Image contains

1. A model.
2. A scoring script used to pass input to the model and return the output of the model.
3. Dependencies needed by the model or scoring script/application.

Two types of images

1. **FPGA image**: Used when deploying to a field-programmable gate array in the Azure cloud.
2. **Docker image**: Used when deploying to compute targets such as Azure Container Instances and Azure Kubernetes Service.

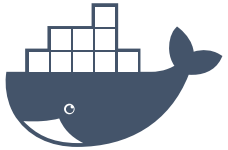


Image Registry

Keeps track of images created from models.

Metadata tags can be attached to images. Metadata tags are stored by the image registry and can be used in image searches

Azure ML Concept
Model Management

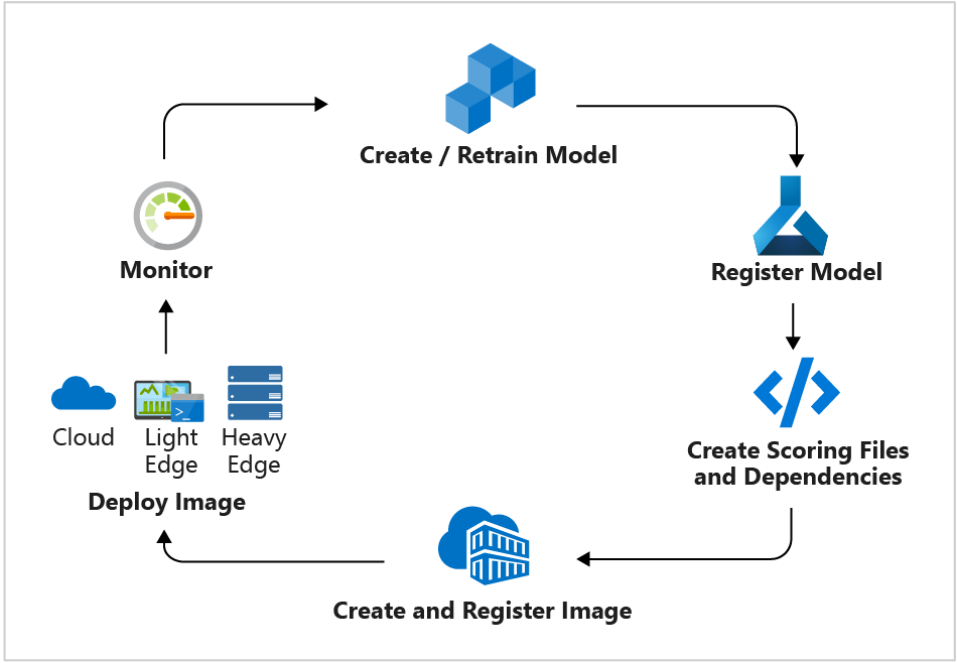
Model Management in Azure ML usually involves these four steps

Step 1: Register Model using the Model Registry

Step 2: Register Image using the Image Registry (the Azure Container Registry)

Step 3: Deploy the Image to cloud or to edge devices

Step 4: Monitor models—you can monitor input, output, and other relevant data from your model.



Azure ML Artifact

Deployment

Deployment is an instantiation of an image. Two options:

Web service

A deployed web service can run on Azure Container Instances, Azure Kubernetes Service, or field-programmable gate arrays (FPGA).

Can receive scoring requests via an exposed a load-balanced, HTTP endpoint.

Can be monitored by collecting Application Insight telemetry and/or model telemetry.

Azure can automatically scale deployments.

IoT Module

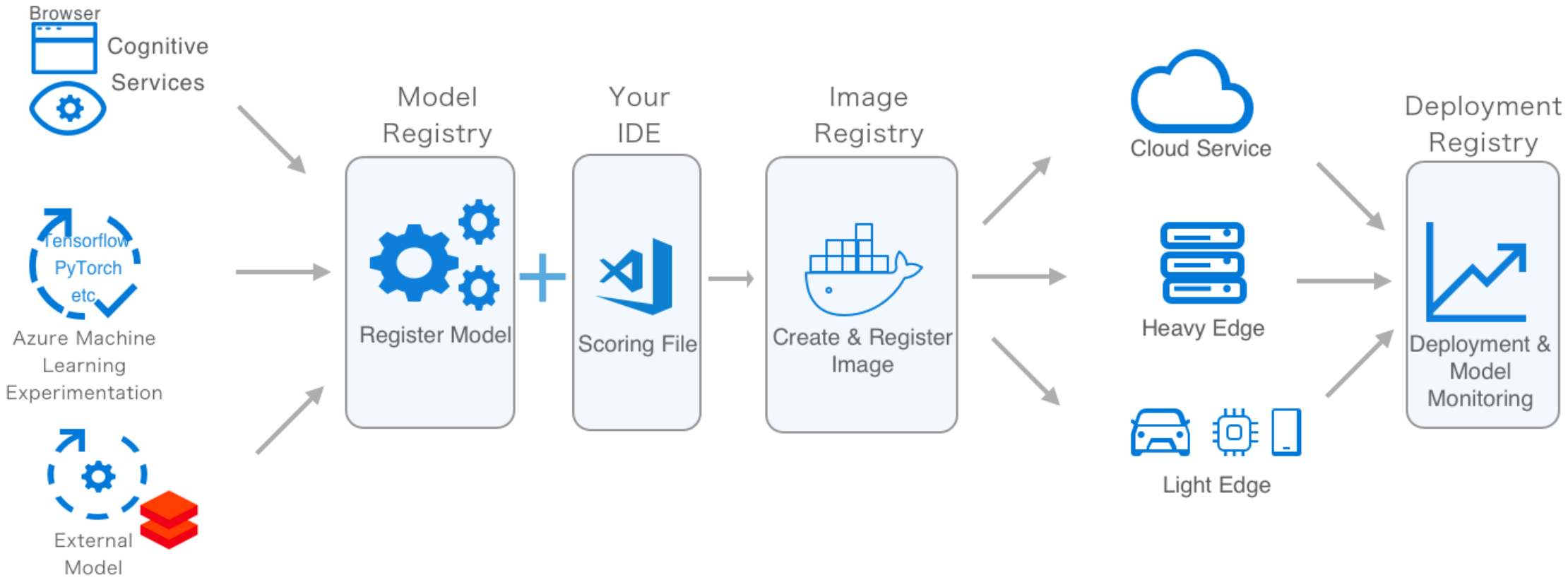
A deployed IoT Module is a Docker container that includes the model, associated script and additional dependencies.

Is deployed using **Azure IoT Edge** on edge devices.

Can be monitored by collecting Application Insight telemetry and/or model telemetry.

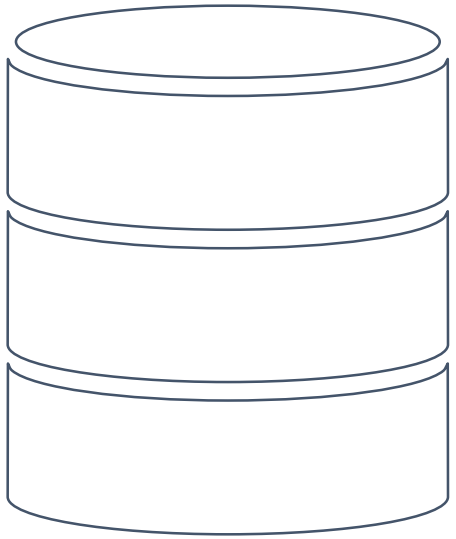
Azure IoT Edge will ensure that your module is running and monitor the device that is hosting it.

Azure ML: How to deploy models at scale



Azure ML Artifact

Datastore



A datastore is a storage abstraction over an Azure Storage Account.

The datastore can use either an Azure blob container or an Azure file share as the backend storage.

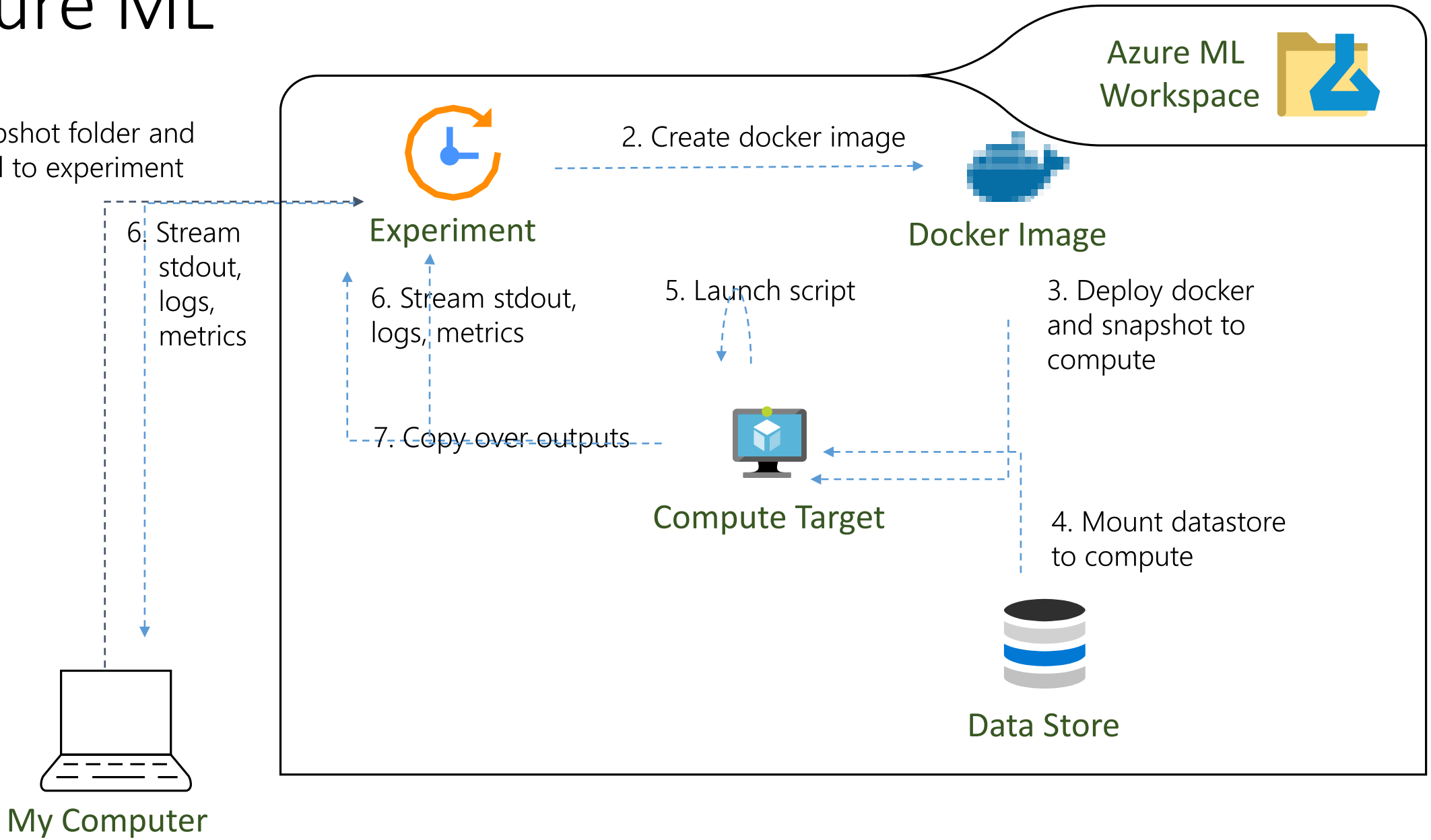
Each workspace has a default datastore, and you may register additional datastores.

Use the Python SDK API or Azure Machine Learning CLI to store and retrieve files from the datastore.

How to use the Azure Machine Learning service:
E2E coding example using the SDK

Azure ML

Steps



Session: Introduction to Lifecycle Management with Azure DevOps/YAML

Why DevOps

Your Company

Build

Test

Release

Your Customers

DevOps

- Improved Deliveries
  - Speed
  - Reliability
- Improved Collaboration
  - Security
  - Scale

DevOps Main Principles

- Develop and test in an environment identical or at least similar to production
- Build and Deploy Frequently
- Validate Quality Continuously

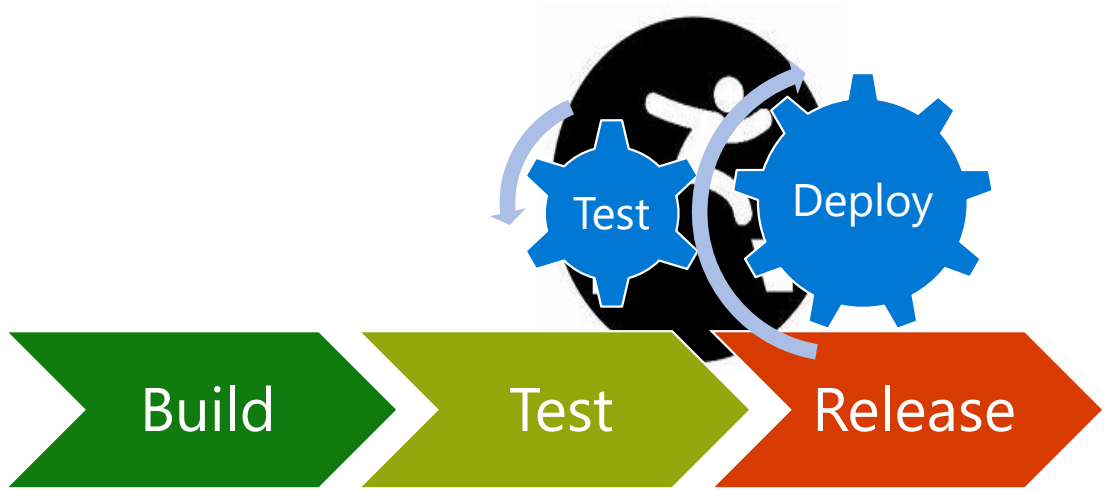
DevOps

DevOps best practices:

- **Continuous Integration (CI):**
  - Merges followed by automated builds and tests.
  - Improve scale and quality
- **Continuous Delivery (CD):**
  - Changes are automatically built, tested, and prepared for a release to production.
  - Changes are deployed to testing/staging and production.
  - Deployment ready product built from the latest code changes, that has passed standardized testing, is always available
- **Infrastructure as Code:**
  - Infrastructure provisioned and managed using code.
  - Controlled with version control and CI.
  - Elimination of manual steps improves scale.
  - Deployment is faster and repeatable. Just in time creation also improves security as it comes with the latest updates and patches.
- Microservices
- Monitoring
- Collaboration

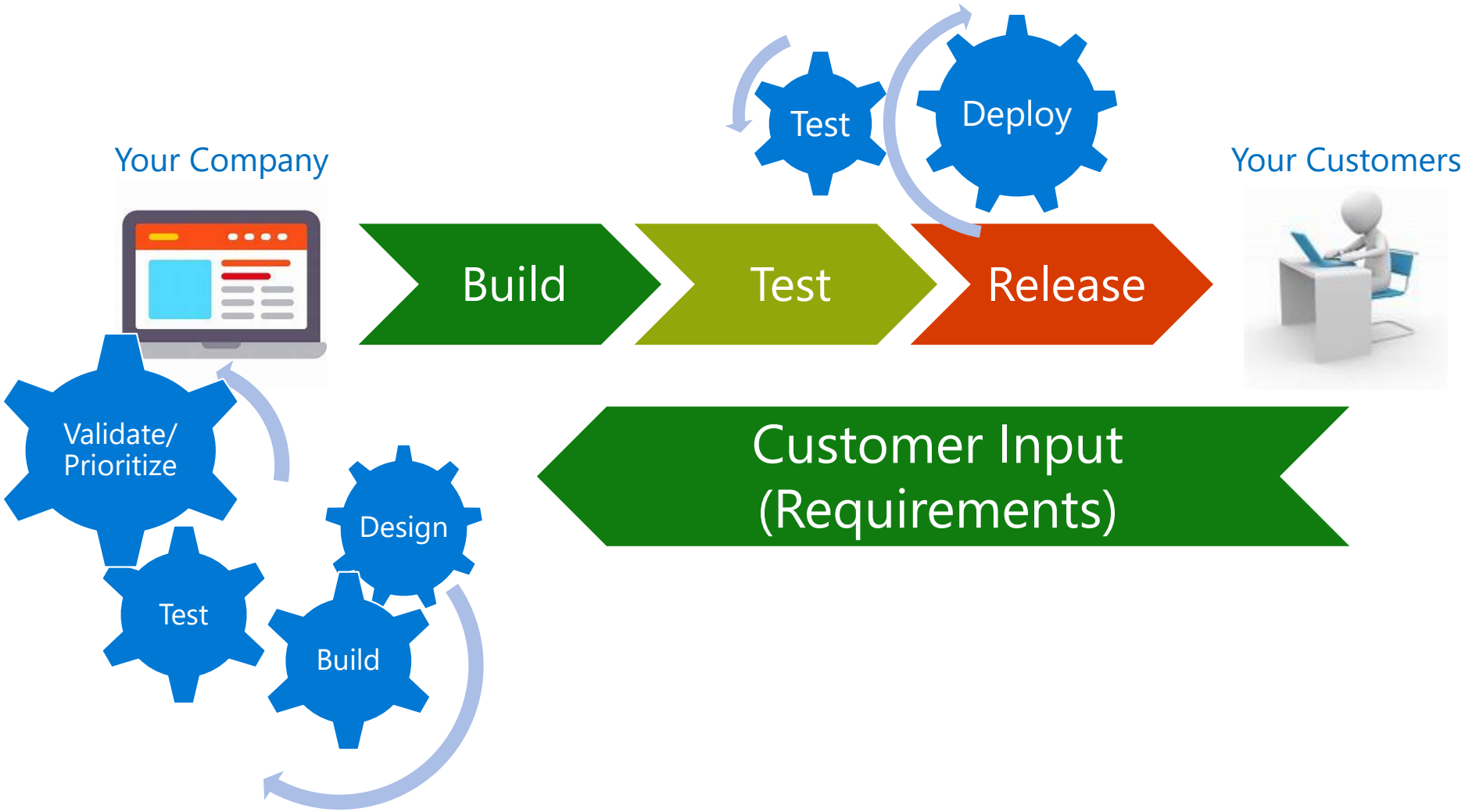
How DevOps

Your Company

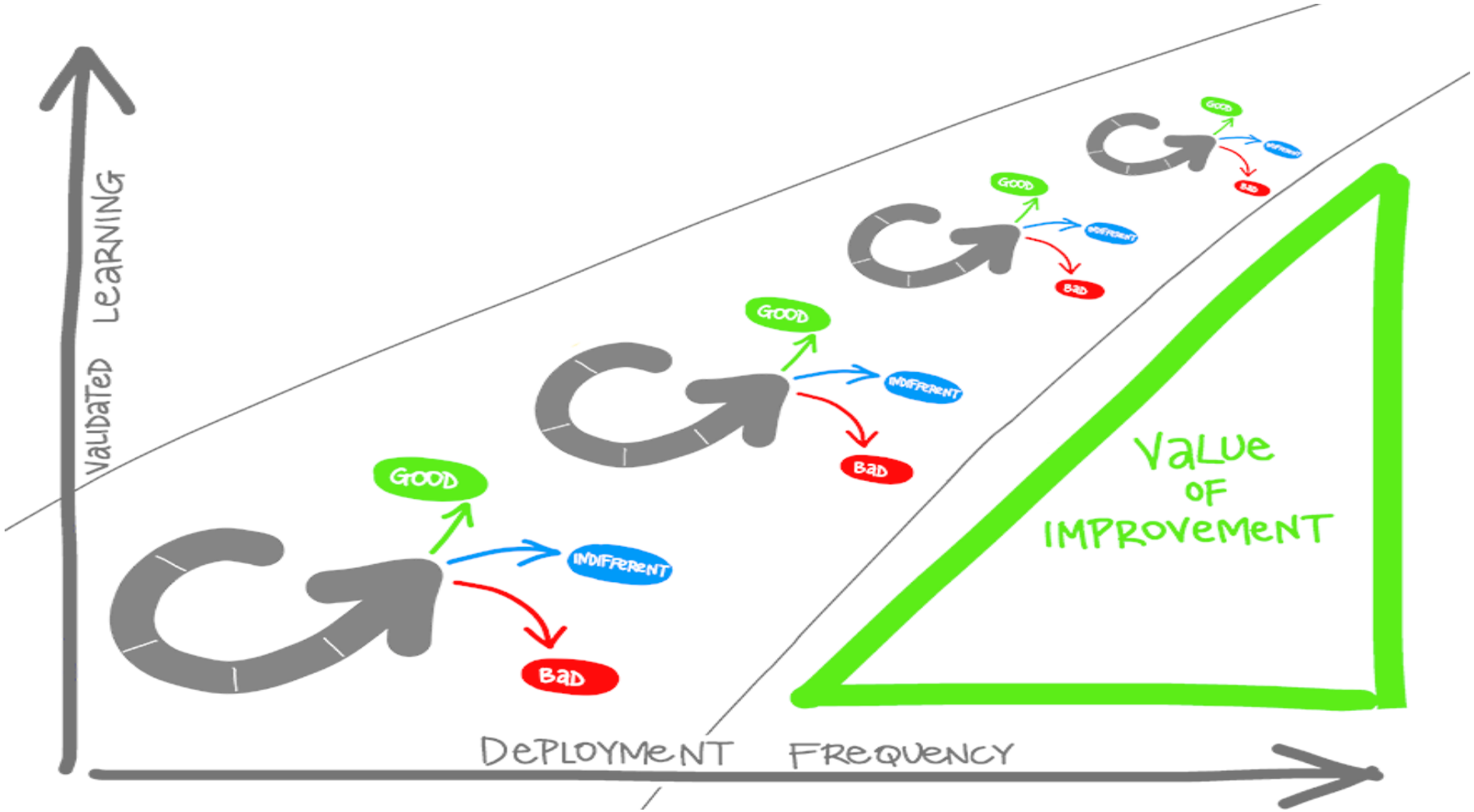


Your Customers

Agile?


Your Company
Test
Deploy
Your Customers
Build
Test
Release
Validate/
Prioritize
Design
Test
Build
Customer Input
(Requirements)

Value of Improvement



Understand CI/CD

Continuous integration (CI)	Continuous delivery (CD)
Increase code coverage.	Automatically deploy code to production.
Build faster by splitting test and build runs.	Ensure deployment targets have latest code.
Automatically ensure you don't ship broken code.	Use tested code from CI process.
Run tests continually.	

When and When Not

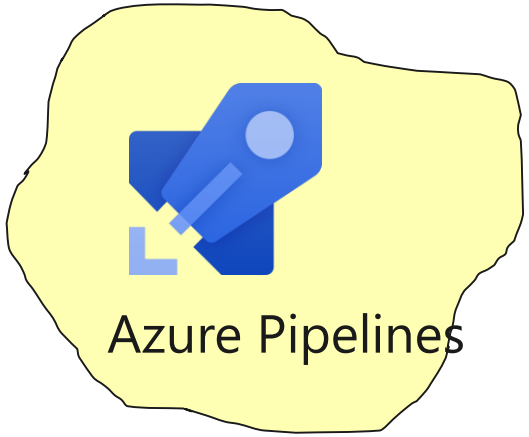
- DevOPs, CI/ CD makes sense in most cases
- May need modification/ gates/ additional loops and controls
  - High Risk
    - Financial
      - Banking
      - Markets
    - Life
      - Medical
      - Emergency Services
    - Operational
      - Power, Nuclear, Chemical
      - Autonomous Systems (Aircrafts, Automobiles, Trains, Industrial Robots)
  - High Value/ Low Frequency
    - Space Exploration
    - Elections
  - Regulatory

Azure DevOps (ADO)

GitHub Integration

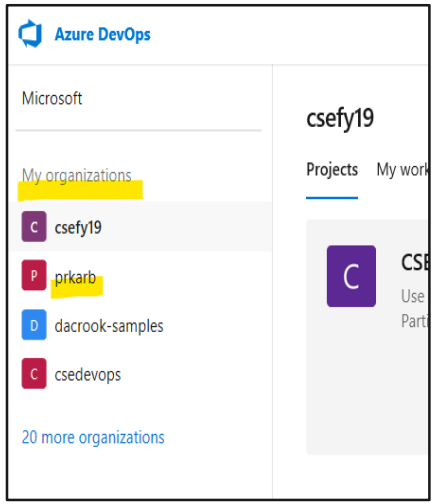
Azure Repos

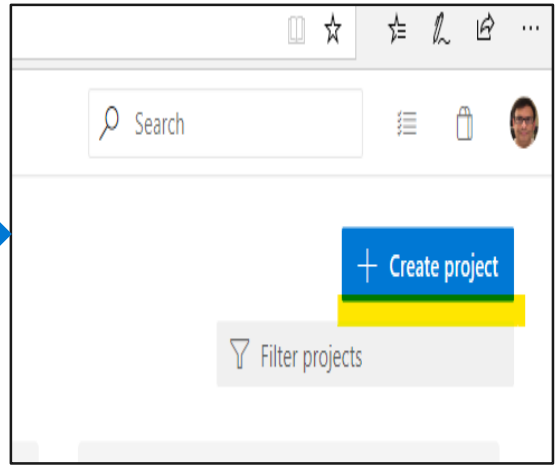
Azure Boards

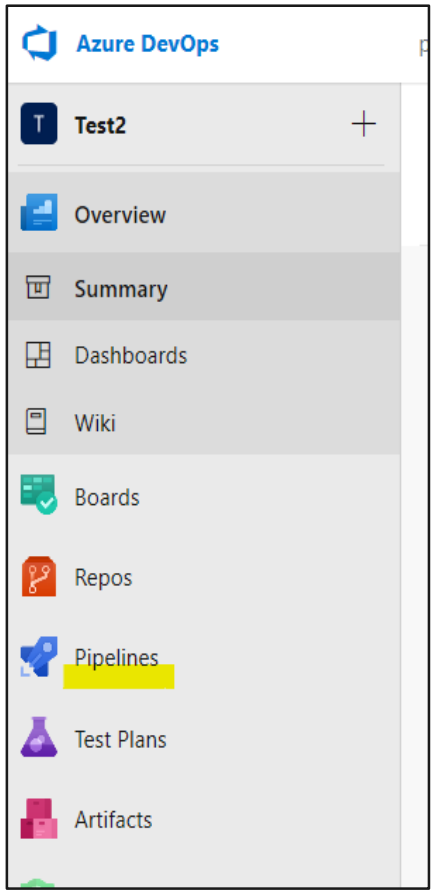


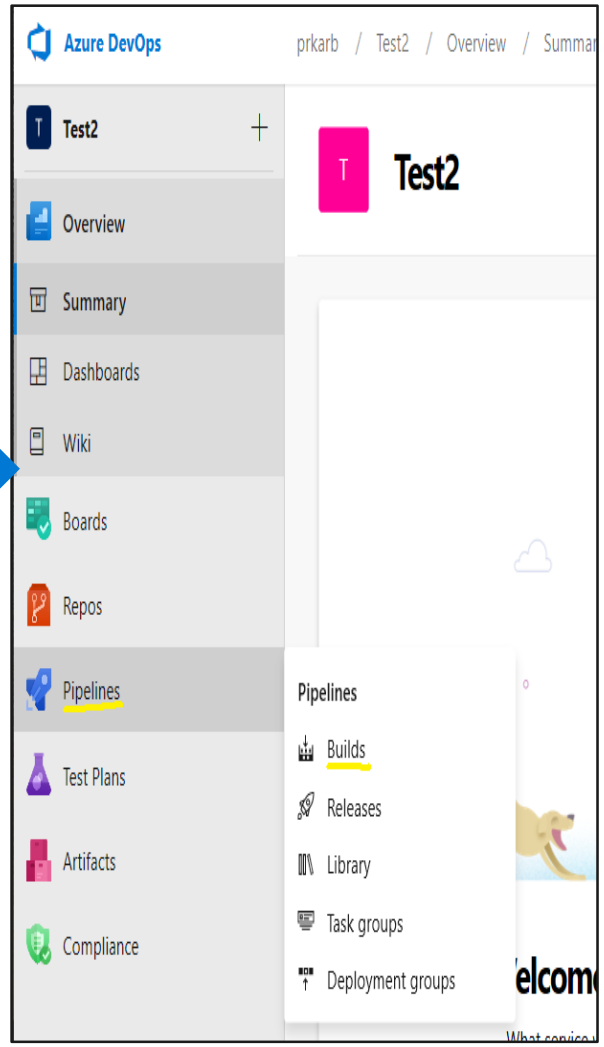
- Work with any language or platform.
- Deploy to different types of targets at the same time.
- Integrate with Azure deployments.
- Build on Windows, Linux, or Mac machines.
- Integrate with GitHub.
- Work with open-source projects.

Creating a pipeline in ADO

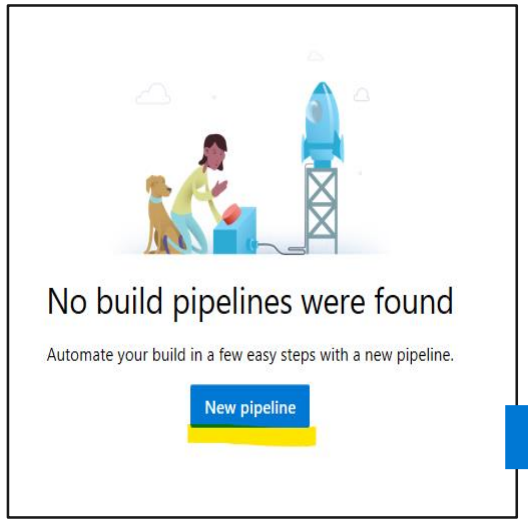


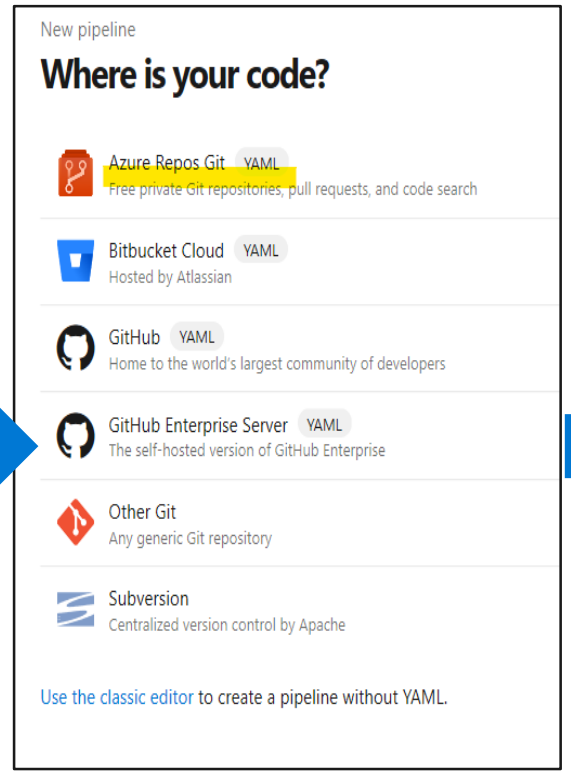


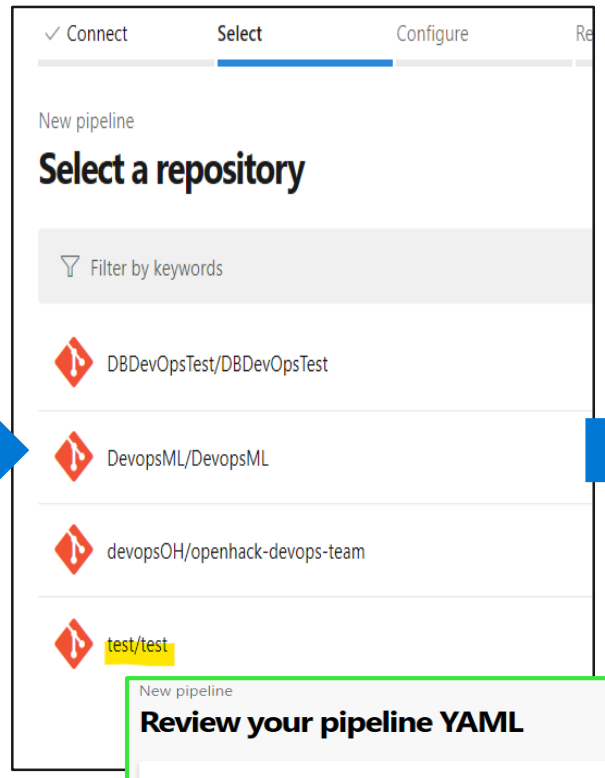


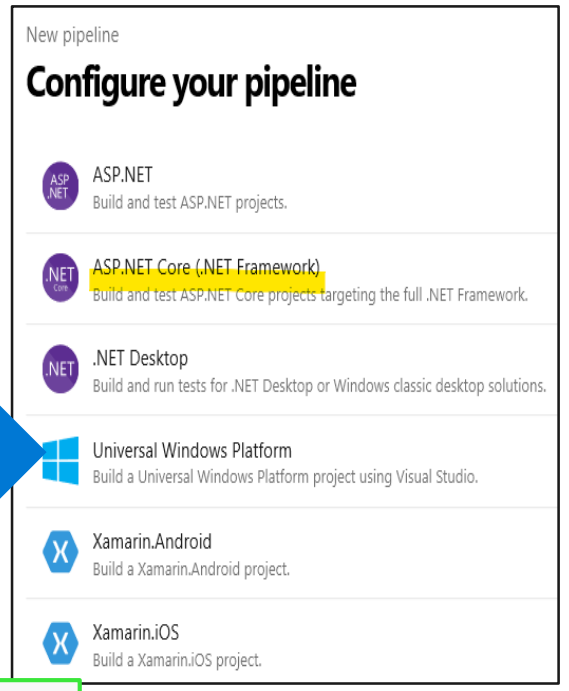


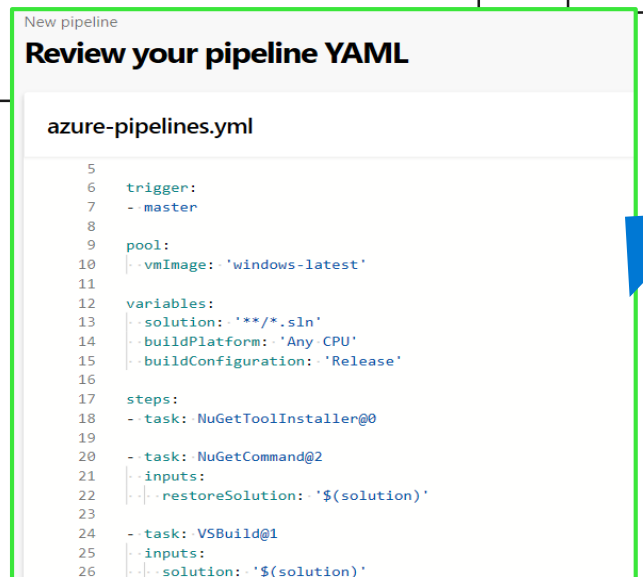
Creating a pipeline in ADO











YAML?

YAML

- "YAML Ain't Markup Language" (started as Yet Another Markup Language)
- Is a human-readable data-serialization language
- Commonly used for configuration files
- Uses both Python-style indentation to indicate nesting
- Uses `[]` for lists and `{}` for map

Blocks

```
--- # Seattle Suburbs
- Bellevue
- Kirkland
- Redmond
```

```
--- # Pet Animals
[dog, cat, hamster]
```

Key-Value Pairs

```
--- # Indented Block
        name: John Smith
        age: 33
--- # Inline Block
{name: John Smith, age: 33}
```

YAML Pipeline

Structure

- Pipeline Stage A
  - Job 1
    - Step 1.1
    - Step 1.2
    - ...
  - Job 2
    - Step 2.1
    - Step 2.2
    - ...
- Stage B
  - ...

Example

```
name: $(Date:yyyyMMdd)$(Rev:.r)
variables:
        var1: value1
jobs:
-  job: One
        steps:
        - script: echo First step!
```

YAML

Stage

- Collection of related jobs.
- By default, stages run sequentially

```
stages:
- stage: Build
  jobs:
  - job: BuildJob
    steps:
    - script: echo Building!
- stage: Test
  jobs:
  - job: TestOnWindows
    steps:
    - script: echo Testing on Windows!
  - job: TestOnLinux
    steps:
    - script: echo Testing on Linux!
```

Job

- Collection of steps to be run by an agent
- Can be run conditionally & May depend on earlier jobs

```
jobs:
- job: MyJob
  displayName: My First Job
  continueOnError: true
  steps:
  - script: echo My first job
  - script: echo It's a YAML job
```

Job Strategies (Matrix, Parallel)

```
jobs:
- job: Build
  strategy:
    matrix:
      Python35:
        PYTHON_VERSION: '3.5'
      Python36:
        PYTHON_VERSION: '3.6'
      Python37:
        PYTHON_VERSION: '3.7'
    maxParallel: 2
```

```
jobs:
- job: SliceItFourWays
  strategy:
    parallel: 4
```

YAML

Variables

- Hardcoded values can be added directly, or variable groups can be referenced.
- May be specified at the pipeline, stage, or job level

```
variables:      # pipeline-level
  MY_VAR: 'my value'
  ANOTHER_VAR: 'another value'

stages:
- stage: Build
  variables:    # stage-level
    STAGE_VAR: 'that happened'

  jobs:
  - job: FirstJob
    variables:  # job-level
      JOB_VAR: 'a job var'
    steps:
    - script: echo $(MY_VAR) $(STAGE_VAR) $(JOB_VAR)
```

YAML

Templates

- Reusable sections of a pipeline in a separate file.
- Four kinds of templates: [Stage, Job, Step, Variable]

```
# File: stages/test.yml

parameters:
  name: ''
  testFile: ''

stages:
- stage: Test_${{ parameters.name }}
  jobs:
  -
```

```
# File: azure-pipelines.yml

stages:
- template: stages/test.yml  # Template reference
  parameters:
    name: Mini
    testFile: tests/miniSuite.js

- template: stages/test.yml  # Template reference
  parameters:
    name: Full
    testFile: tests/fullSuite.js
```

YAML

Pools

Specifies which Agent pool to use for a job of the pipeline

```
pool:
  name: string  # name of the pool to run this job in
  demands: string | [ string ]  ## see below
  vmImage: string # name of the vm image you want to use, only valid in the Microsoft-hosted pool
```

Azure

```
pool:
  vmImage: ubuntu-16.04
```

Private Pool

```
pool:
  name: MyPool
  demands:
  - myCustomCapability   # check for existence of capability
  - agent.os -equals Darwin  # check for specific string in capability
```

YAML

Triggers

- Trigger to start a Pipeline
- Trigger Types: [Push, PR (pull request), Scheduled]

```
trigger:
  batch: true
  branches:
    include:
    - features/*
    exclude:
    - features/experimental/*
  paths:
    exclude:
    - README.md
```

```
pr:
  branches:
    include:
    - features/*
    exclude:
    - features/experimental/*
  paths:
    exclude:
    - README.md
```

```
schedules:
- cron: "0 0 * * *"
  displayName: Daily midnight build
  branches:
    include:
    - master
    - releases/*
    exclude:
    - releases/ancient/*
- cron: "0 12 * * 0"
  displayName: Weekly Sunday build
  branches:
    include:
    - releases/*
  always: true
```

YAML

Publish

- Shortcut for the Publish Pipeline Artifact task
- Publish (upload) a file or folder as a pipeline artifact that can be consumed by other jobs and pipelines

```
steps:
- publish: $(Build.SourcesDirectory)/build
  artifact: WebApp
```

Download

- Shortcut for the Download Pipeline Artifact task
- Download one or more artifacts associated with the current run to $(Pipeline.Workspace)

```
steps:
- download: current
  artifact: WebApp
  patterns: '**/.js'
```

YAML

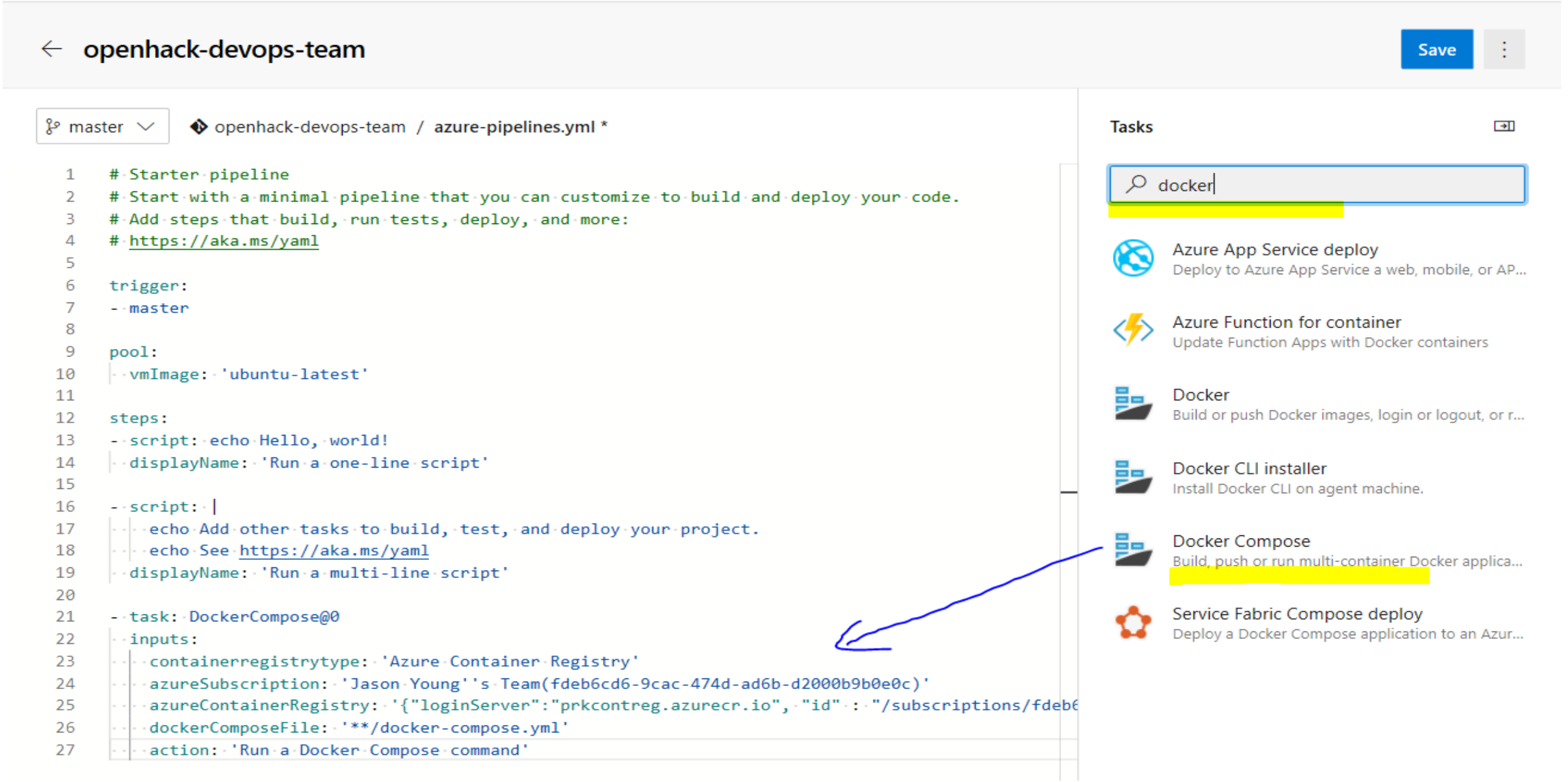
Task

- Tasks are the building blocks of a pipeline.
- There is a catalog of tasks available to choose from.

```
steps:
- task: string  # reference to a task and version, e.g. "VSBuild@1"
  displayName: string  # friendly name displayed in the UI
  name: string  # identifier for this step (A-Z, a-z, 0-9, and underscore)
  condition: string
  continueOnError: boolean  # should future steps run if this step fails? Default: 'false'
  enabled: boolean  # whether or not to run this step; defaults to 'true'
  timeoutInMinutes: number
  inputs: { string: string }  # task-specific inputs
  env: { string: string }  # list of environment variables to add
```

```
steps:
- task: VSBuild@1
  displayName: Build
  timeoutInMinutes: 120
  inputs:
    solution: '**\*.sln'
```

Use Task Helpers to Edit a Pipeline

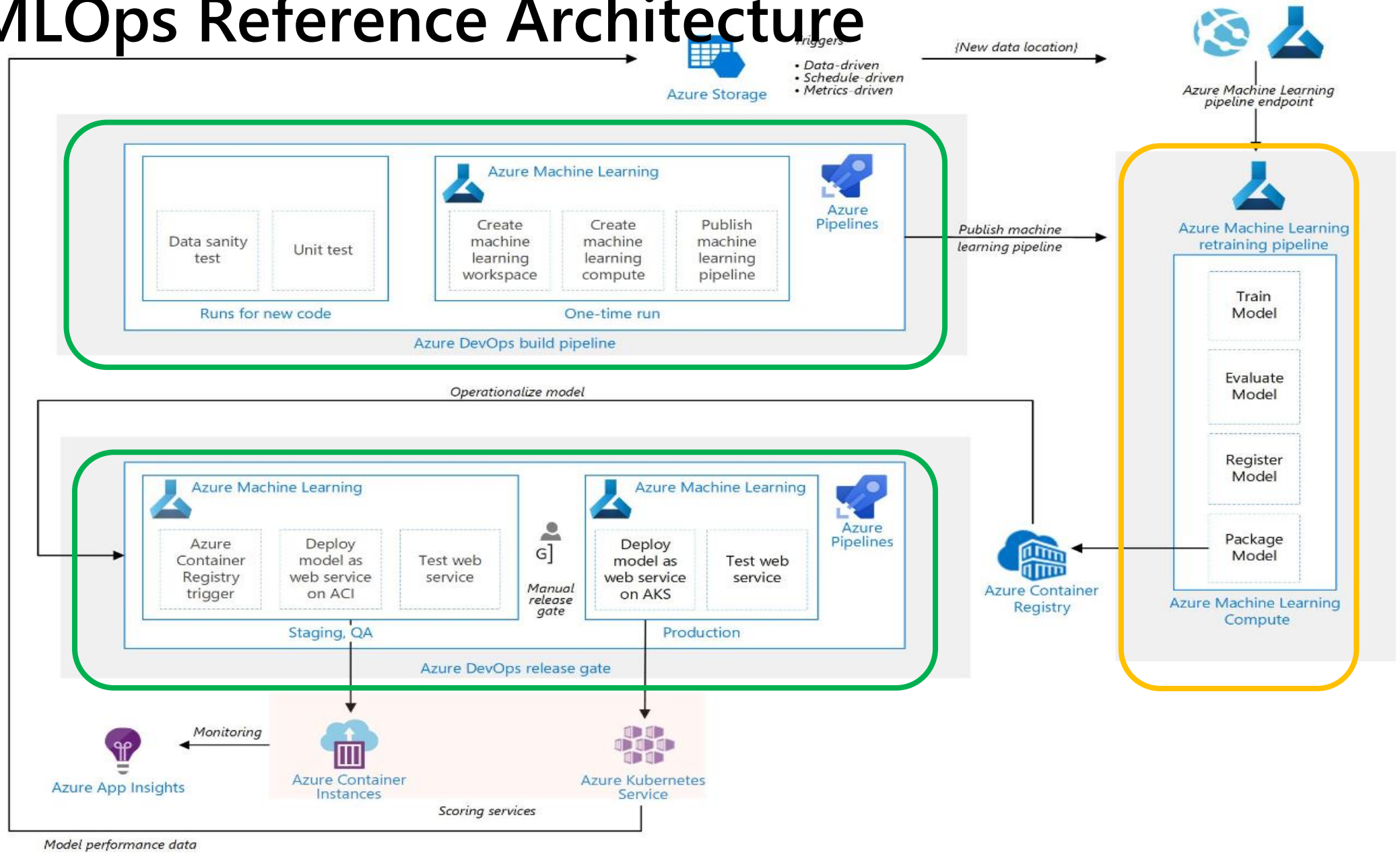


Session:YAML and AML Pipeline Integration

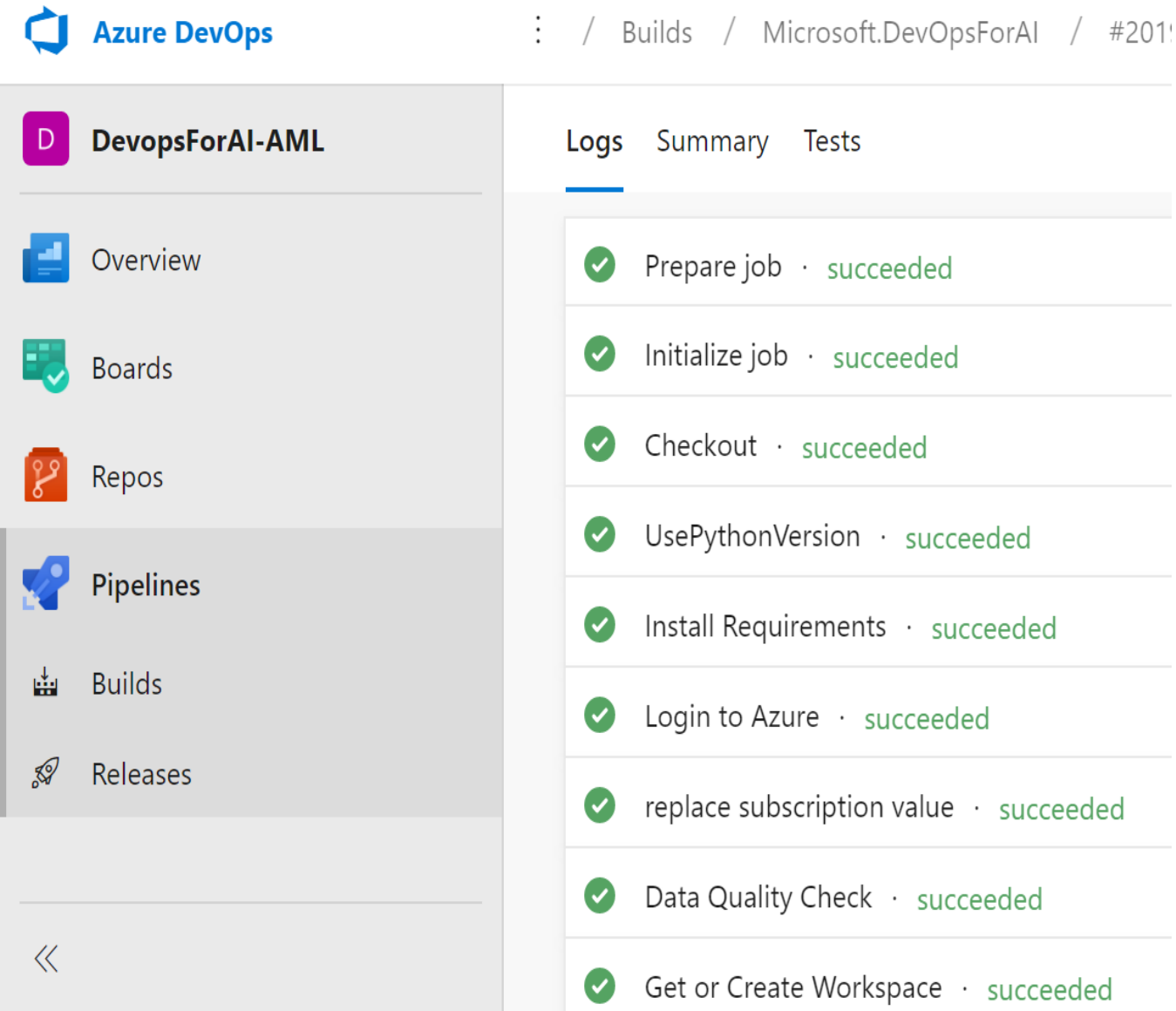
Session goals

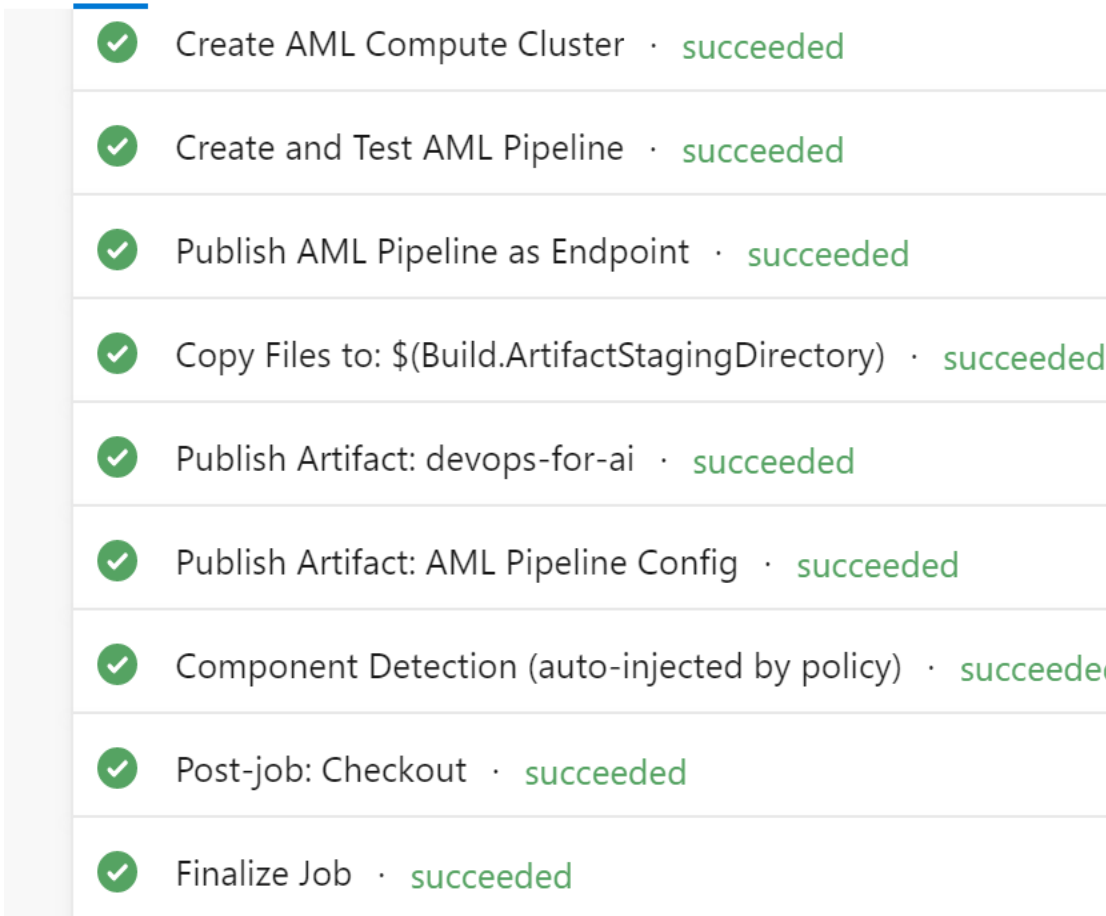
- Project architecture
- Build, Re-training, Deployment pipelines
- Lab Setup

MLOps Reference Architecture

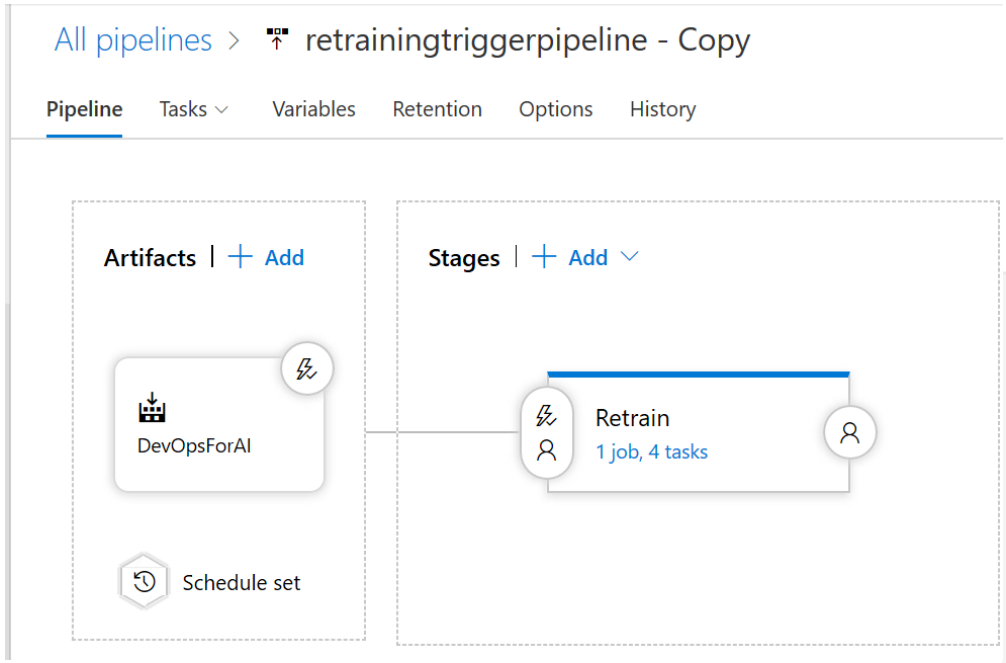


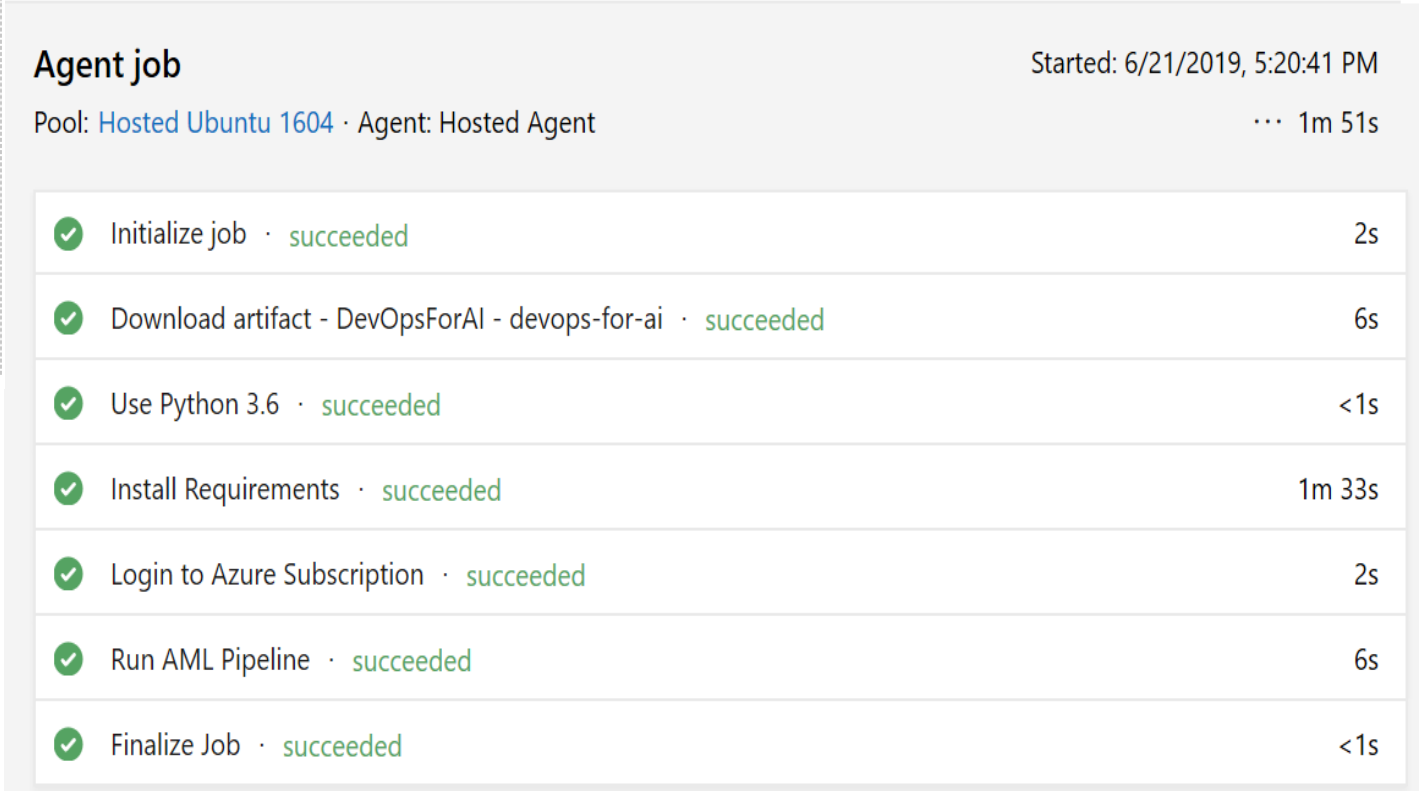
Build Pipeline



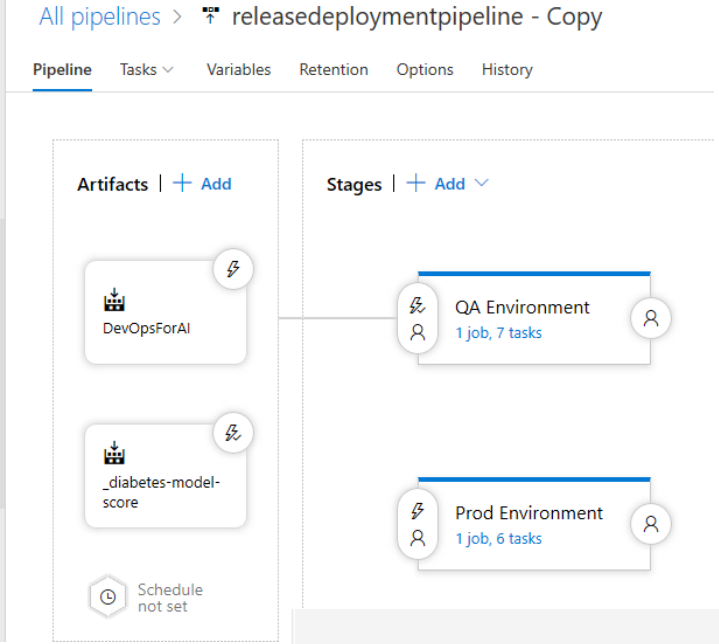


Re-training Pipeline





Deployment Pipeline



Agent job

Started: 6/22/2019, 10:26:38 AM

Pool: Hosted Ubuntu 1604 · Agent: Hosted Agent

··· 5m 47s

Initialize job · succeeded	1s
Download artifact - DevOpsForAI - devops-f... · succeeded	6s
Use Python 3.6 · succeeded	<1s
Bash Script · succeeded	1m 31s
Login to Azure Subscription · succeeded	2s
Get Latest Scoring Image Name & Version · succeeded	3s
Deploy new image to ACI · succeeded	3m 57s
Test the image on ACI · succeeded	3s
Finalize Job · succeeded	<1s

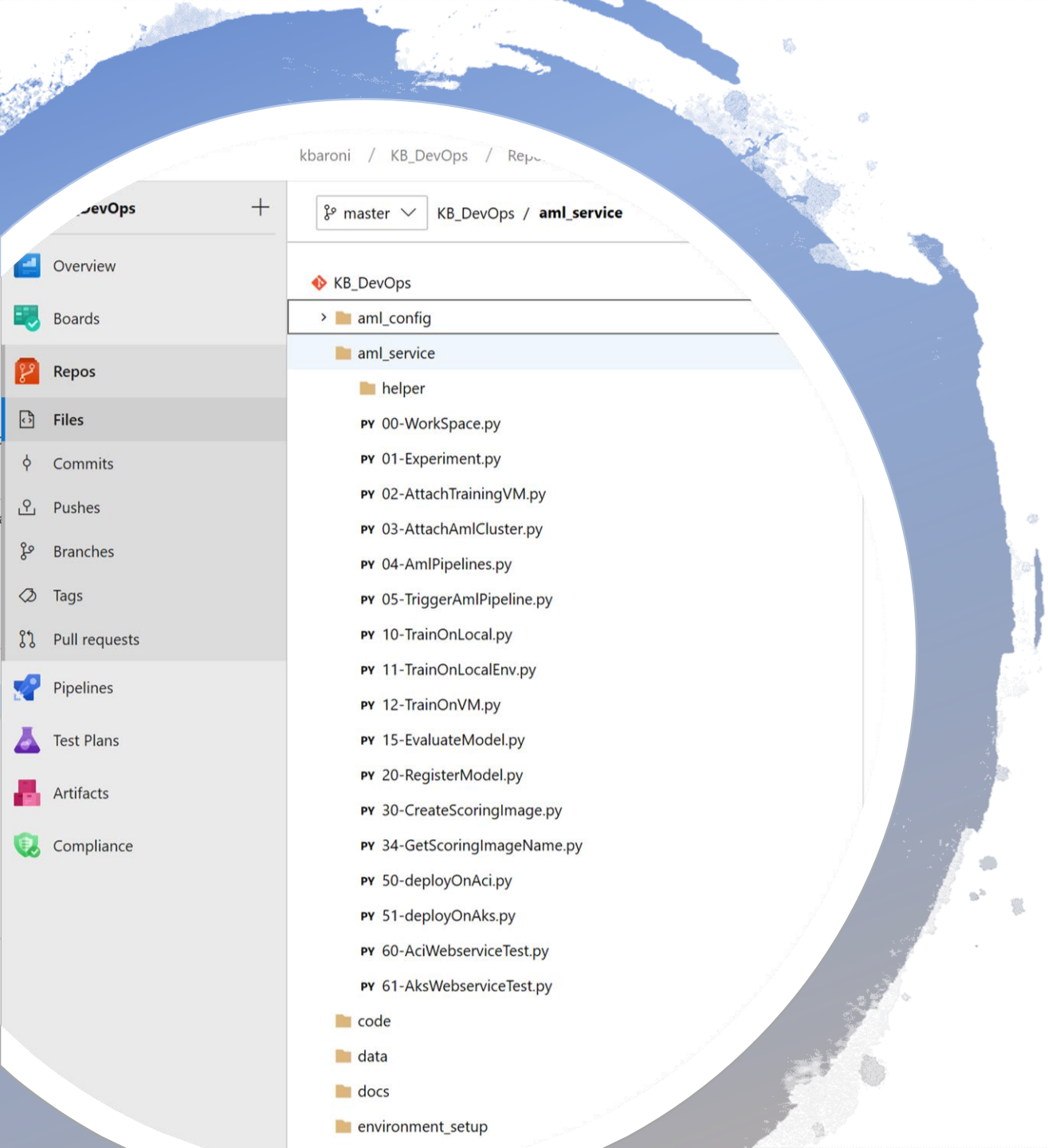
Agent job

Started: 6/22/2019, 10:37:08 AM

Pool: Hosted Ubuntu 1604 · Agent: Hosted Agent

··· 13m 59s

Initialize job · succeeded	3s
Download artifact - DevOpsForAI - devops-f... · succeeded	6s
Use Python 3.6 · succeeded	<1s
Bash Script · succeeded	1m 30s
Login to Azure Subscription · succeeded	3s
Get Latest Scoring Image Name & Version · succeeded	4s
Deploy to AKS · succeeded	10m 57s
Test AKS endpoint · succeeded	1m 13s
Finalize Job · succeeded	<1s



Project Code

Lab Setup

Create new Azure DevOps project

Create Azure DevOps service connection

Setup variable group with secrets

Import repo into Azure DevOps