Microsoft

Machine Learning, AI, and Data Science Conference

November 12–14
Redmond

2018

WiFi Information

Network: MLADS2018

Password: MLADS2018!

Microsoft

DevOps for AI with Azure ML

Development of Retraining & Conditional Deployment Pipelines on Azure

Praneet Singh Solanki
Dmitry Pechyoni
Richin Jain
Vivek Gupta

Session Goals

- Target Audience:
  - Data Scientist, AI Software Engineer
- Understanding DevOps for AI
- Implementation using Azure DevOps and Azure ML Service
- Real Customer Implementations
- Access to the Code/Pipeline sample

Agenda

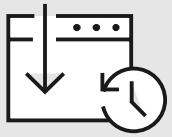
- DevOps
  - What is it? Who is it designed for?
- Why DevOps for AI?
  - Why should data scientists care about DevOps?
- How to implement with Azure DevOps?
  - Azure DevOps Walkthrough, AML SDK Walkthrough, AI DevOps Demo using AML SDK
- Customer Implementations
- Future Enhancements
- How to get started now?

Who We Are?

- AI CAT Team, part of Azure CAT
- Our Mission: Make AI real for our customers
- Teams working on DevOps for AI

What is DevOps?

DevOps brings together people, processes, and technology automating software delivery to provide continuous value to our users.



Continuous Integration (CI)

- Focuses on blending the work of individual developers together into a repository.
- Each time you commit code, it's automatically built and tested, and bugs are detected faster.

Continuous Deployment (CD)

- Automate the entire process from code commit to production if your CI/CD tests are successful.

Continuous Learning & Monitoring

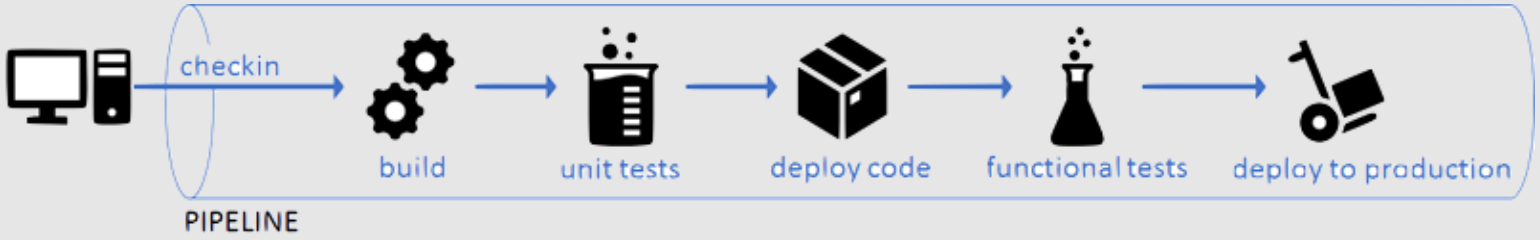
- Using CI/CD practices, paired with monitoring tools, safely deliver features to your customers as soon as they're ready.

Who uses DevOps & Why?

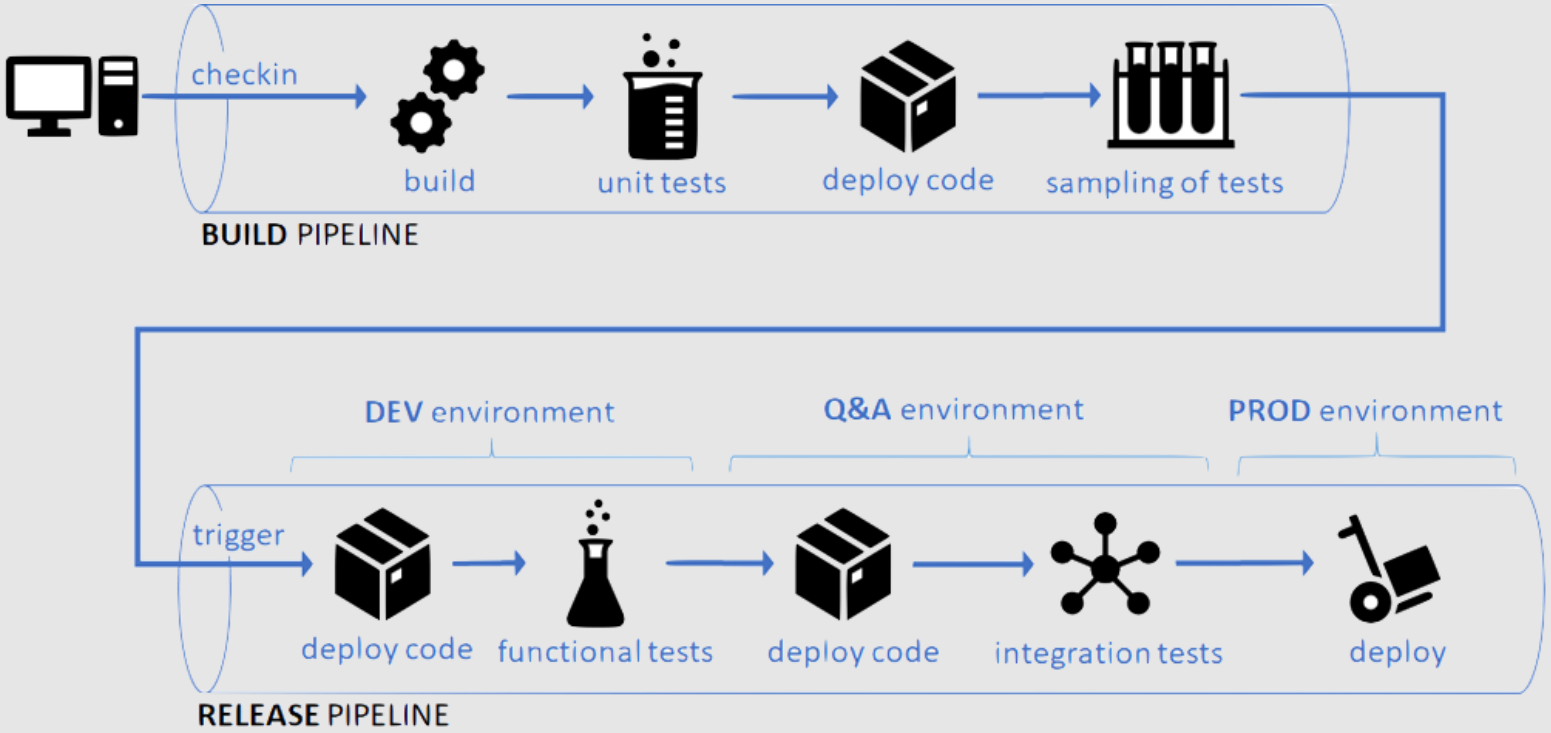
- Mostly used in developing apps and now also for AI app
- Used by Software/AI/Data Engineer and Data Scientists to:
  - record the steps that they do repeatedly and make sure they occur the same way every time
  - make sure their code quality is consistently checked amongst all contributors
  - manage deployments to multiple environments
  - need to integrate with the work of others and test that the dependencies work
  - monitor their deployments
  - automate all the above steps

DevOps for a Typical Application

Single environment



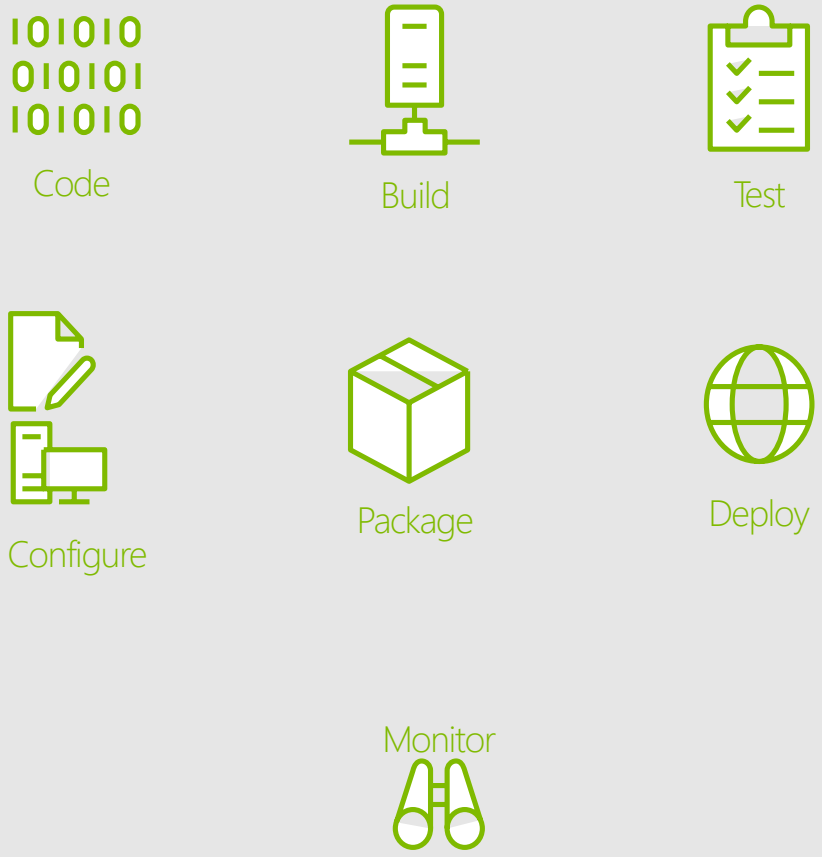
Multiple environments



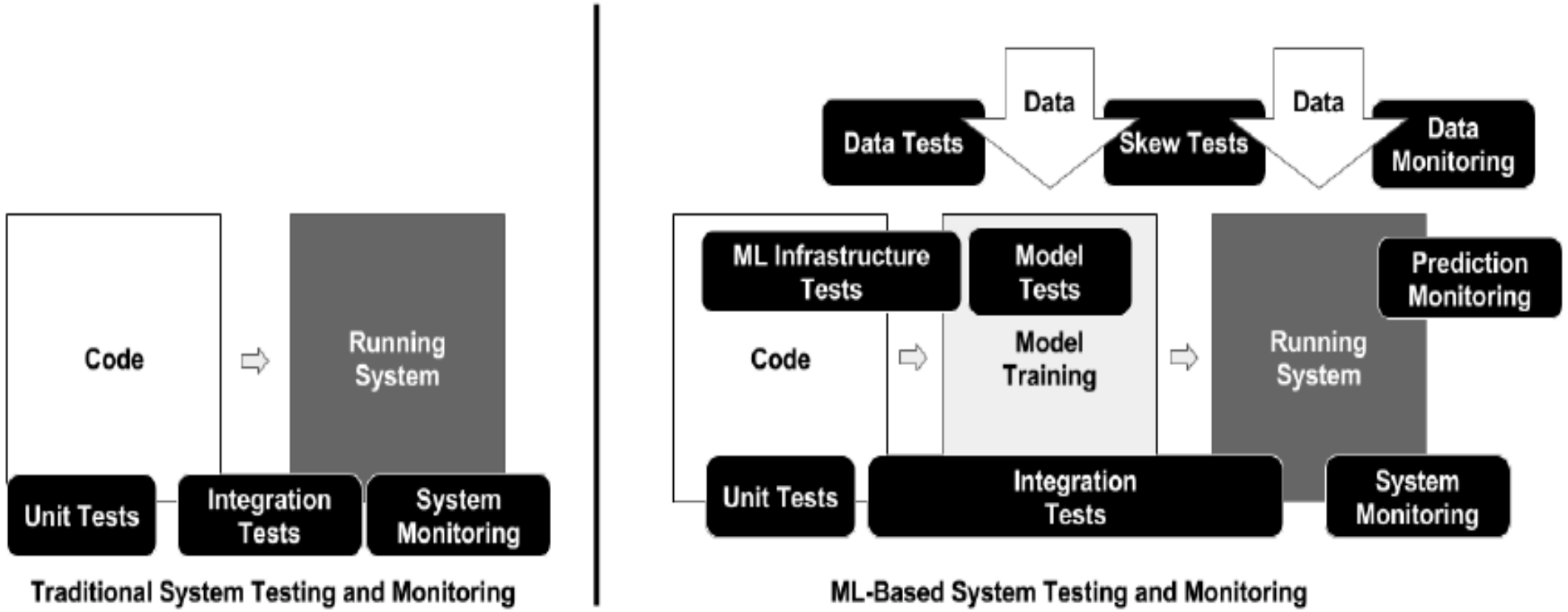
DevOps for AI

DevOps Tasks in Enterprise Software

- Continuous Integration
- Automated Testing
- Continuous Deployment
- Infrastructure as Code (IaC)
- Release Management
- App Performance Monitoring
- Load Testing & Auto-Scale



Traditional Software Engineer vs AI/ML DevOps



https://ai.google/research/pubs/pub46555

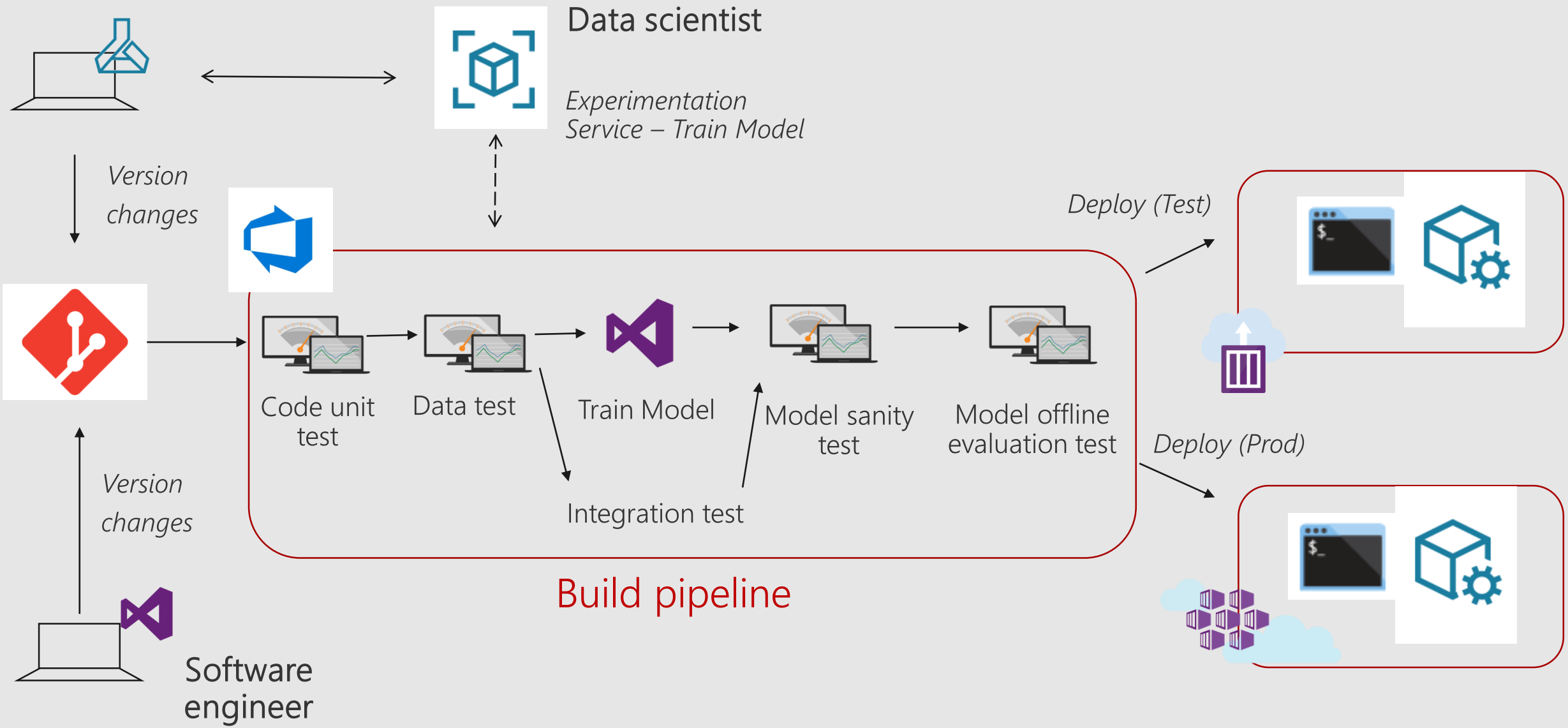
Why DevOps for AI Development?

- Can be used to manage pain points
  - Integration between data scientists and software engineers
  - Scripts stored on one person's machine and process are not repeatable by anyone else
  - Model versioning based on names, hard to know where model is deployed
  - Accuracy of model is determined during training and might change over time because of the data drift
  - Determine the value of the trained model, decide if it should be deployed

Key Goals of DevOps for AI

- Repeatability of model performance
- Evaluation of model predictions
- Managing different model versions and files
- Integration testing
- Operationalization of the model
- Monitoring of training and scoring pipelines

Integrated Pipeline for Data Scientists and Software Engineers



Data test

- Test that the new and the old training/scoring data have the same
  - schema
  - distribution of feature values and labels
- Missing values test
- Detection of anomalous values

https://ai.google/research/pubs/pub46555
http://www.vldb.org/pvldb/vol11/p1781-schelter.pdf

Unit Test

- Data preprocessing code unit test
- Feature generation code unit test
- Check that training code generates model
- Check that scoring code generates predictions

Model Test

- Model quality sanity test
  - run training code over a small fixed training set
  - score a small fixed test set with the newly create model
  - compute offline metric (e.g. precision@k) and check that it passes some threshold
- Measure training code latency
- Measure scoring code latency

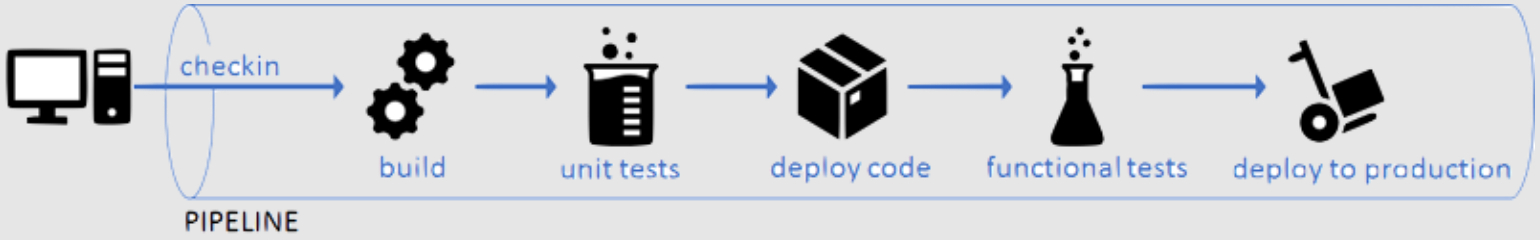
Model Offline Evaluation Test

- Train the new model over full training set
- Score validation data with the new model, compute offline evaluation metric
- Score the same validation data with production model, compute offline evaluation metric
- Deploy the new model if its offline evaluation metric is better than the one of production model

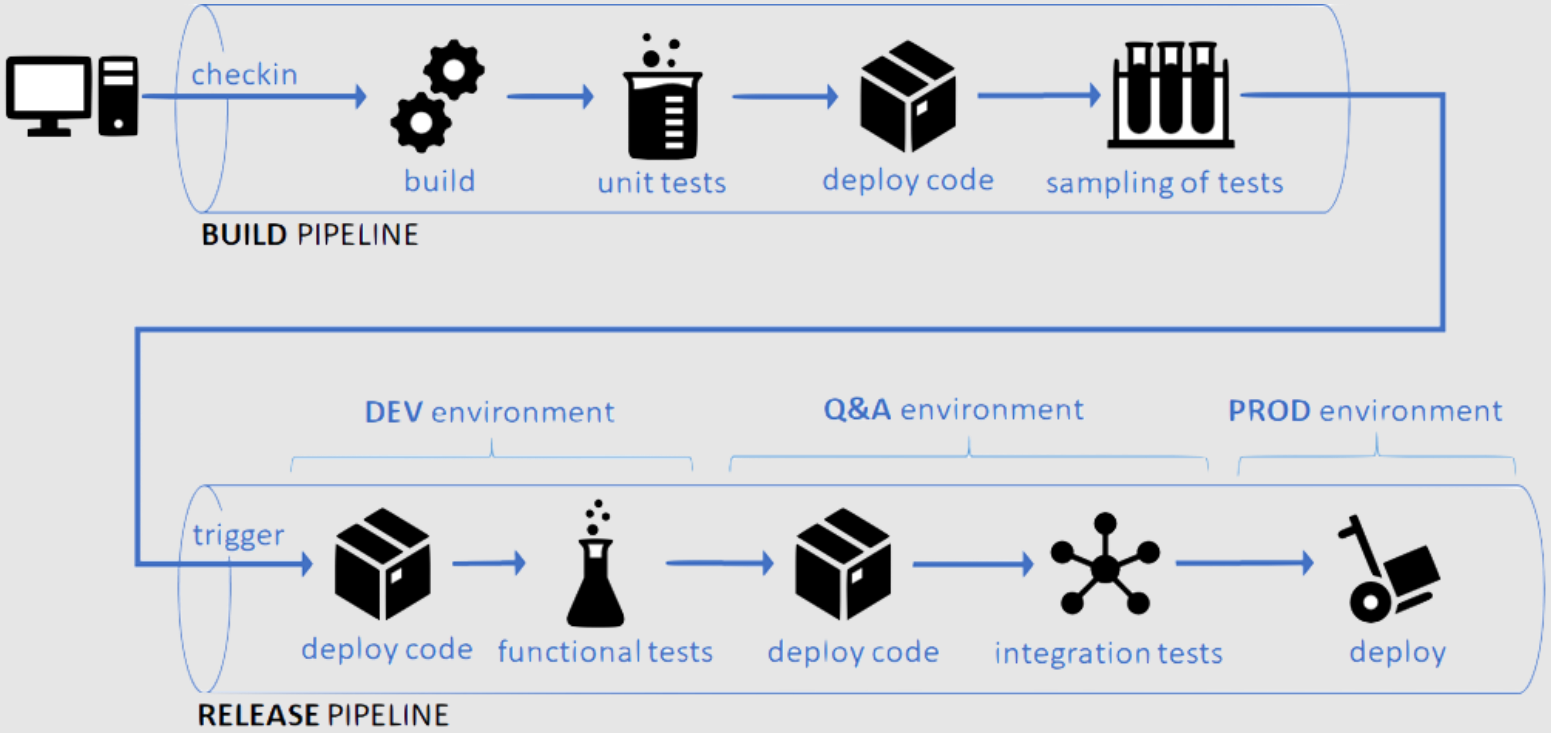
Azure DevOps Walkthrough

DevOps for a Typical Application

Single environment



Multiple environments

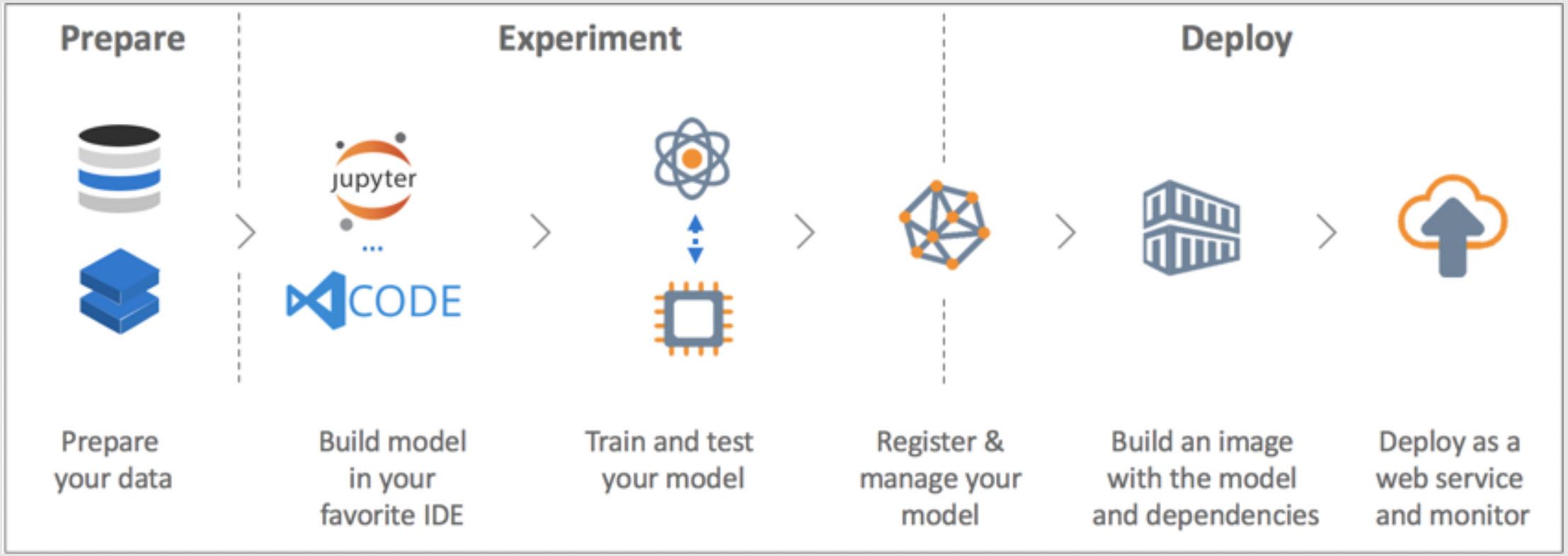


Summary of Azure DevOps Walkthrough

- Create DevOps Organization/Project
- Board: Work Items and Backlogs
- Repos: Get the source code from existing repo
- Pipelines: Agents, Build Pipeline, Release Pipeline, Tasks, Triggers, Release Environment

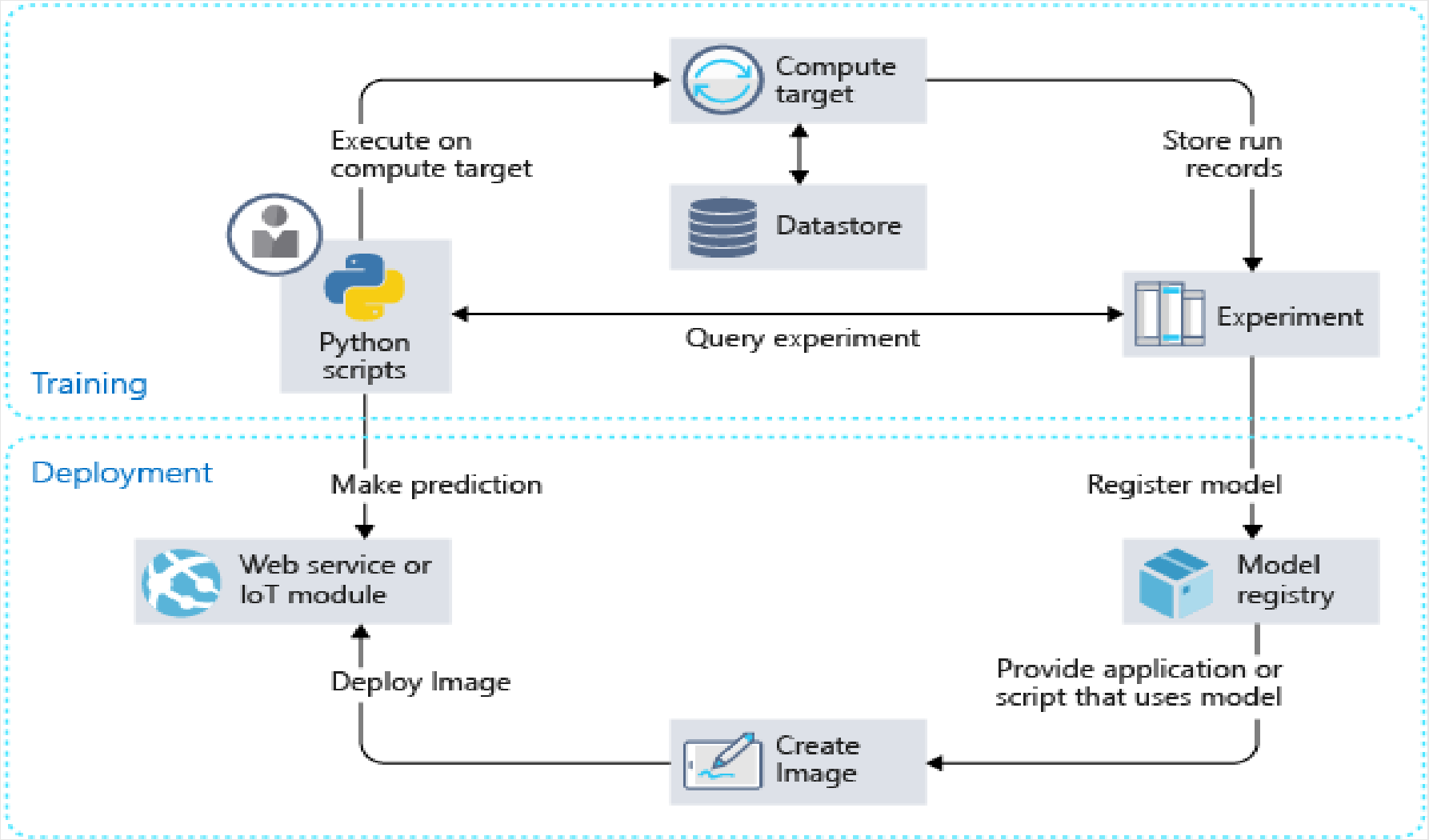
AML SDK Demo

Typical AI/ML Development Flow



Source: https://docs.microsoft.com/en-us/azure/machine-learning/service/overview-what-is-azure-ml

AI Development Flow Implementation using AML



Source: https://docs.microsoft.com/en-us/azure/machine-learning/service/concept-azure-machine-learning-architecture

Actual Implementation with AML SDK

jupyter 02.train-on-local Last Checkpoint: a few seconds ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

Train locally

- Create or load AML workspace.
- Configure & execute a local run in a user-managed Python environment.
- Query run metrics to find the best model
- Register model for operationalization.
- Create Scoring image
- Deploy scoring image on ACI
- Test Scoring image

Prerequisites

Make sure you go through the 00. Installation and Configuration Notebook first if you haven't.

In [5]:

```
# Check core SDK version number
import azureml.core
print("SDK version:", azureml.core.VERSION)
!az account set  -s "ff18d7a8-962a-406c-858f-49acd23d6c01"
```

```
SDK version: 0.1.68
```

1. Create or load AML workspace ¶

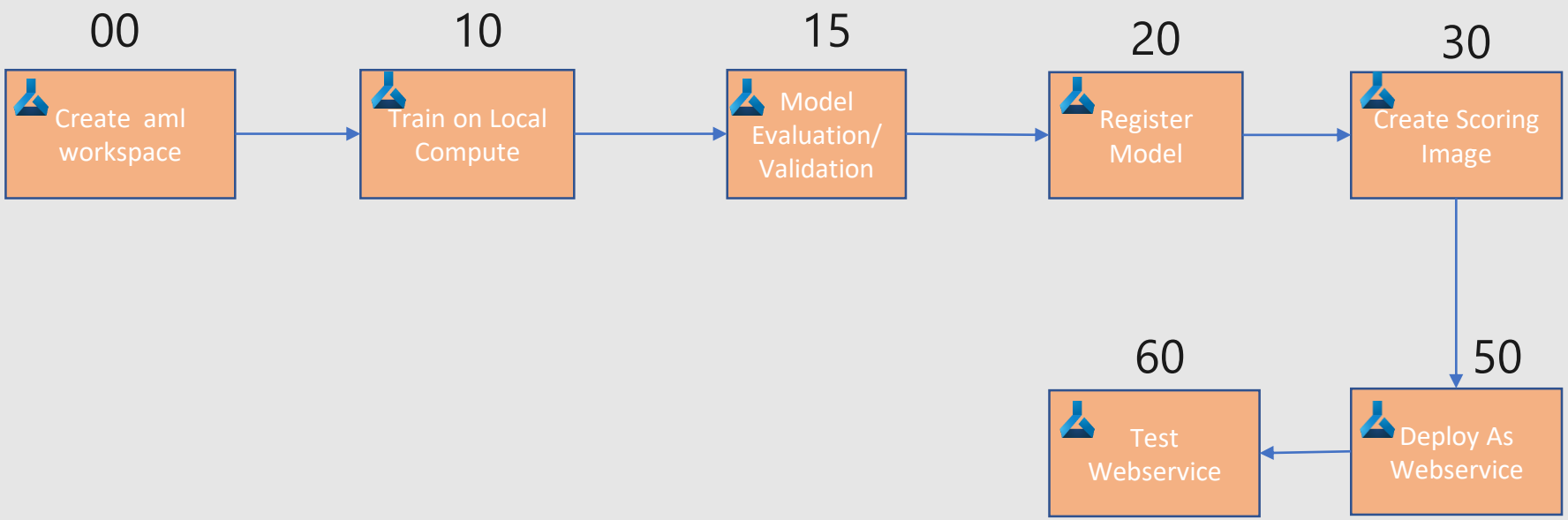
Create new workspace or Initialize a workspace object from persisted configuration.

In [15]:

```
from azureml.core.workspace import Workspace

subscription_id ='ff18d7a8-962a-406c-858f-49acd23d6c01'
resource_group ='Prsol_MLADS_Demo'
workspace_name = 'MLADS_Demo_ws'
workspace_region = 'southcentralus'
try:
    ws = Workspace(subscription_id = subscription_id, resource_group = resource_group, workspace_name = workspace_name)
    #ws.write_config()
    print('Workspace configuration succeeded. You are all set!')
except:
    print('Workspace not found. Creating new workspace.')
    ws = Workspace.create(name = workspace_name,
                          subscription_id = subscription_id,
                          resource_group = resource_group,
                          location = workspace_region,
                          create_resource_group = True,
```

AML Demo Summary



Demo AI App in Azure DevOps with AML

Recommended Code Structure

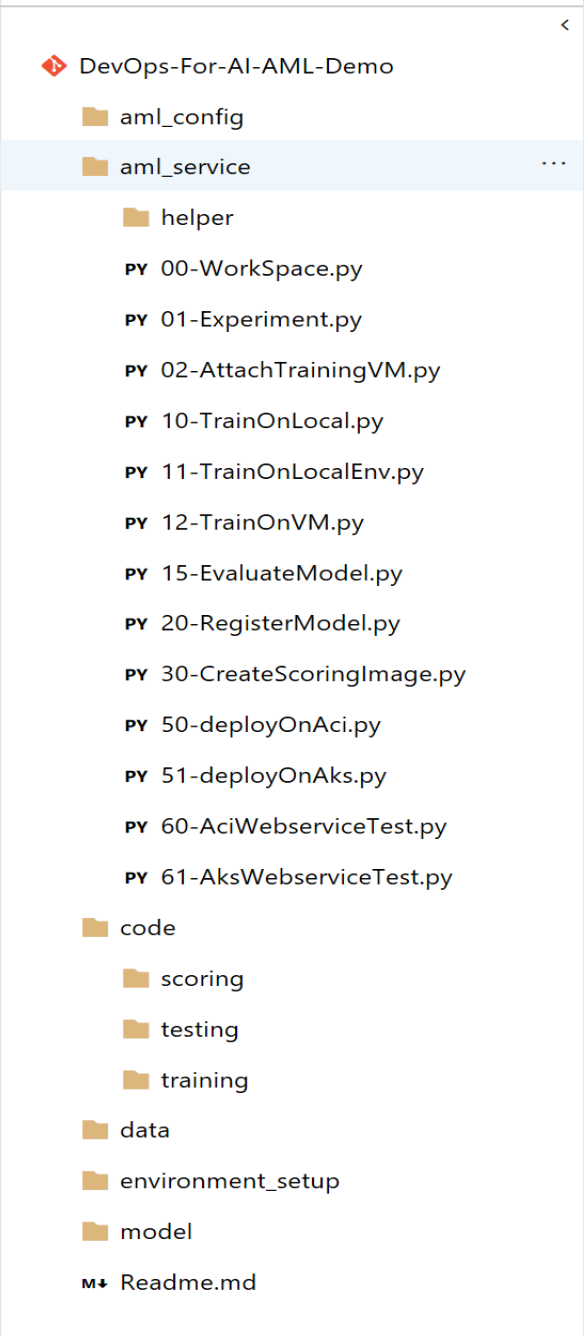
- Config:
  Keep all the configuration files inside aml_config

- AML Functionalities:
  All the python scripts under aml_service uses AML services through SDK. These scripts are run as tasks in DevOps pipeline.

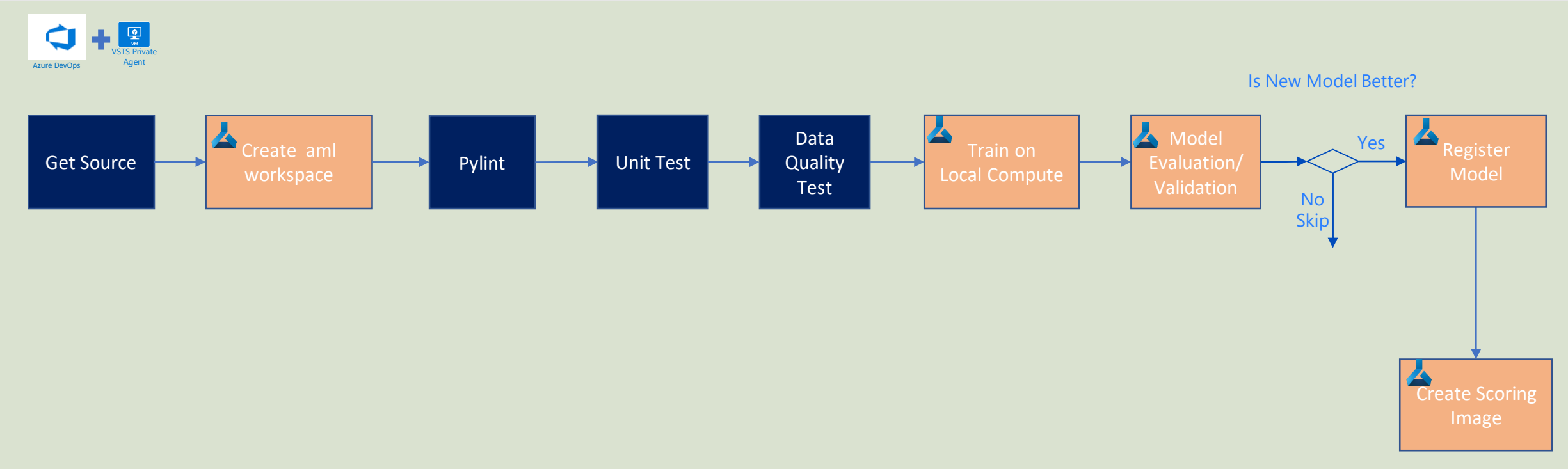
- Code:
  It consist of model training, scoring and the testing scripts.

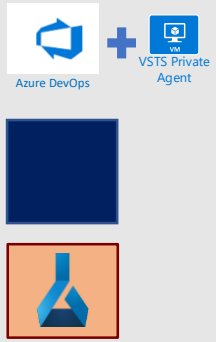
- Data:
  This is where the sample data set is kept which is used during experimentation.

- Environment Setup
  This consist of shell script which prepares the environment on the Azure DevOps Hosted agent



Continuous Integration – Build Pipeline



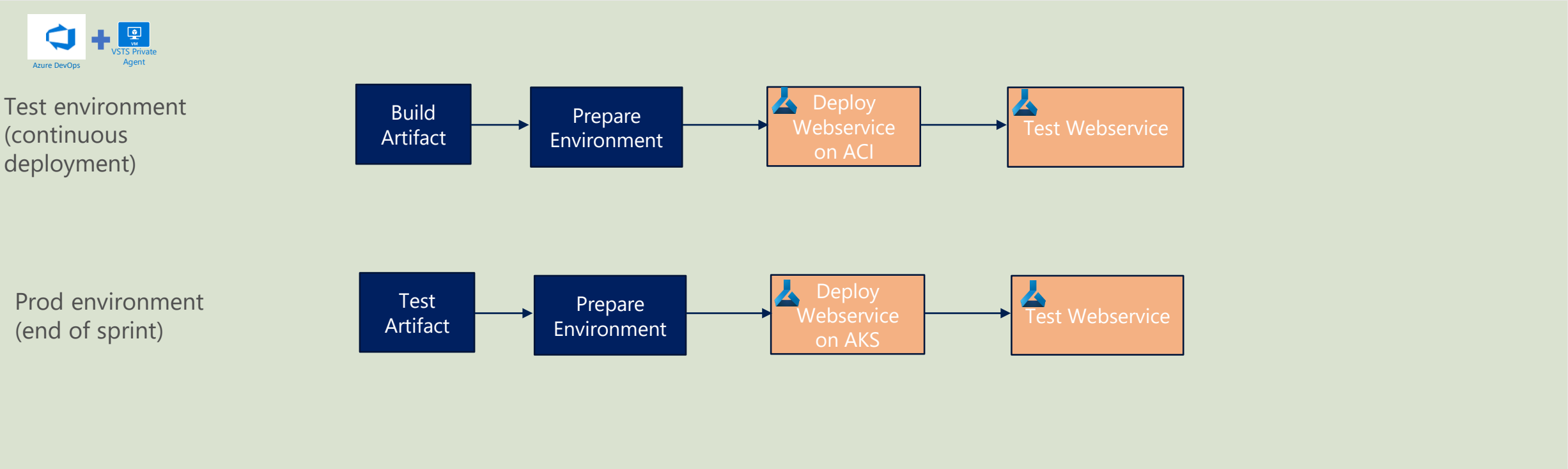


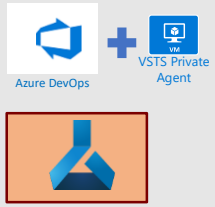
Azure DevOps Pipeline running on VSTS Agent

Task common to traditional swd apps

Pipeline task running AML SDK

Deployment to Different Environment – Release Pipeline





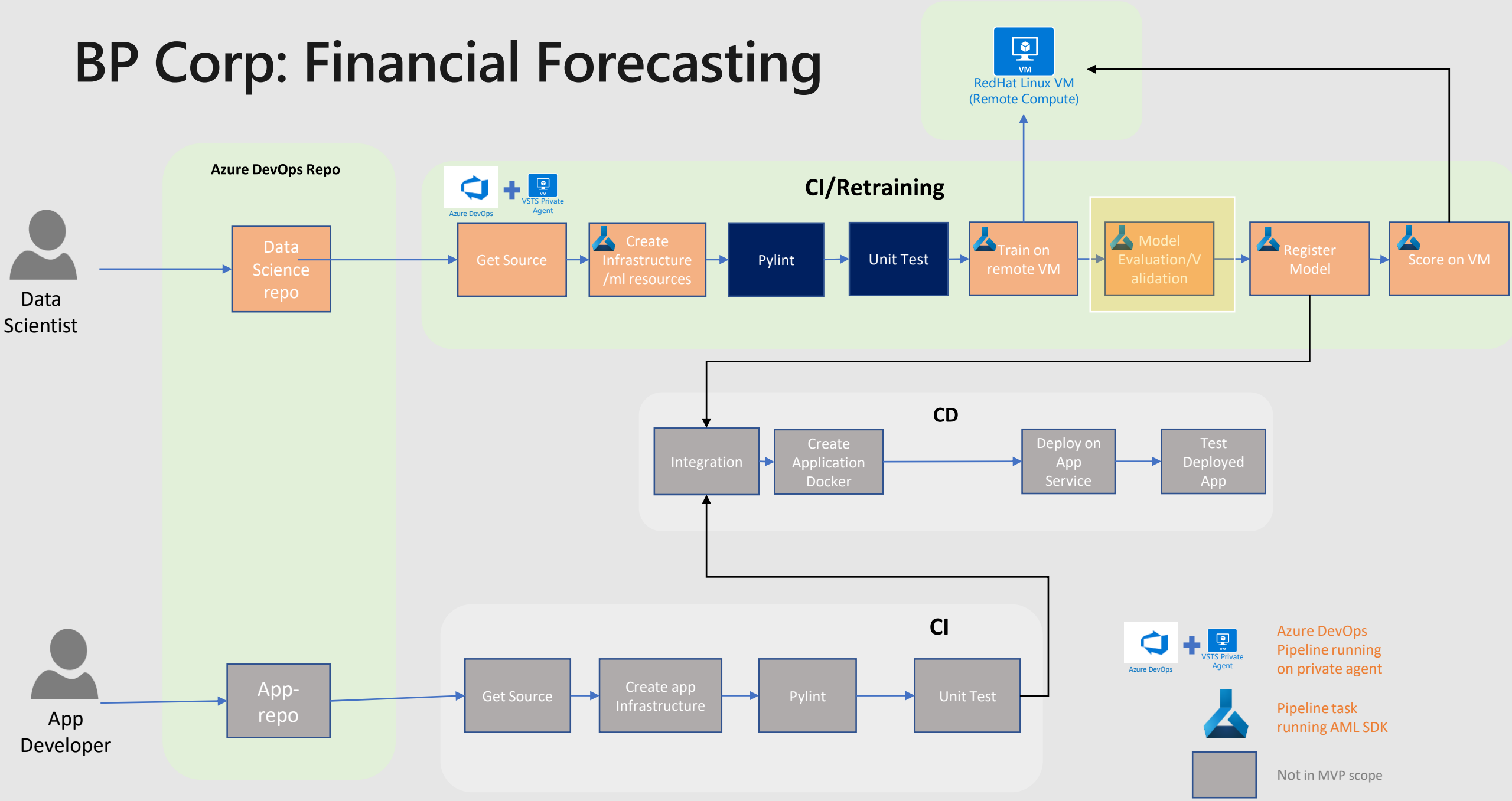
Azure DevOps Pipeline running on VSTS Agent

Pipeline task running AML SDK

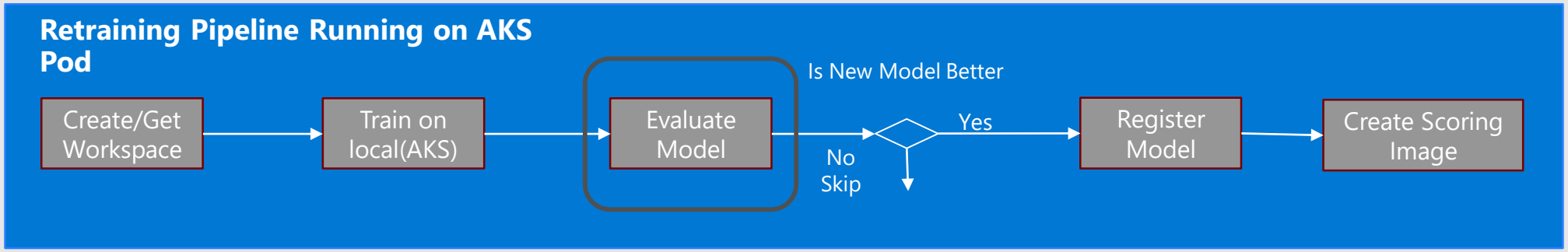
Customer Implementations

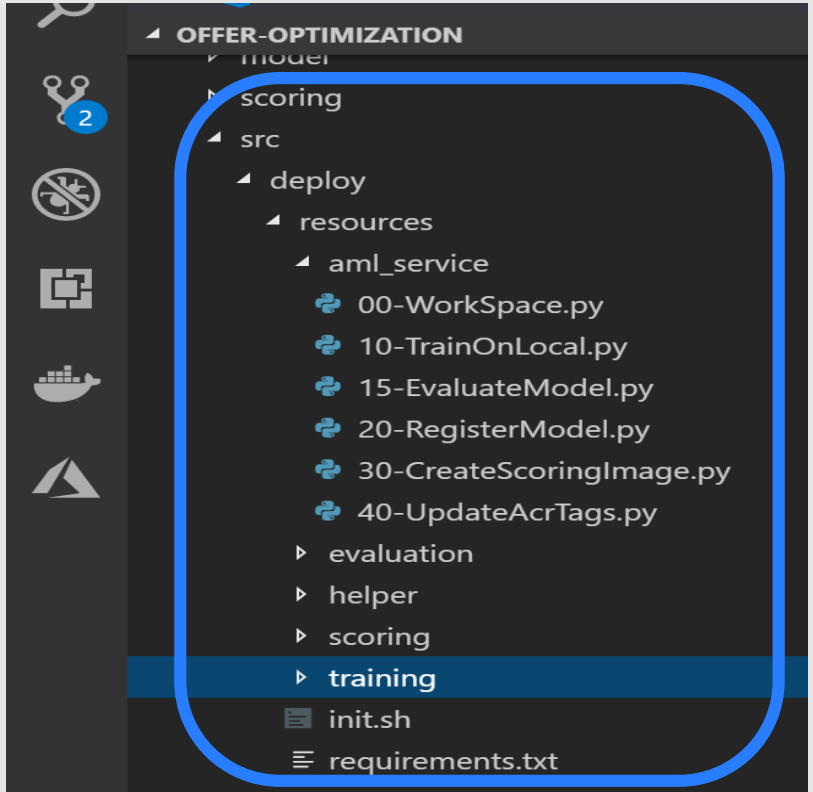
- BP Corp
- PROS

BP Corp: Financial Forecasting



PROS: Airline Ancillary Recommendation





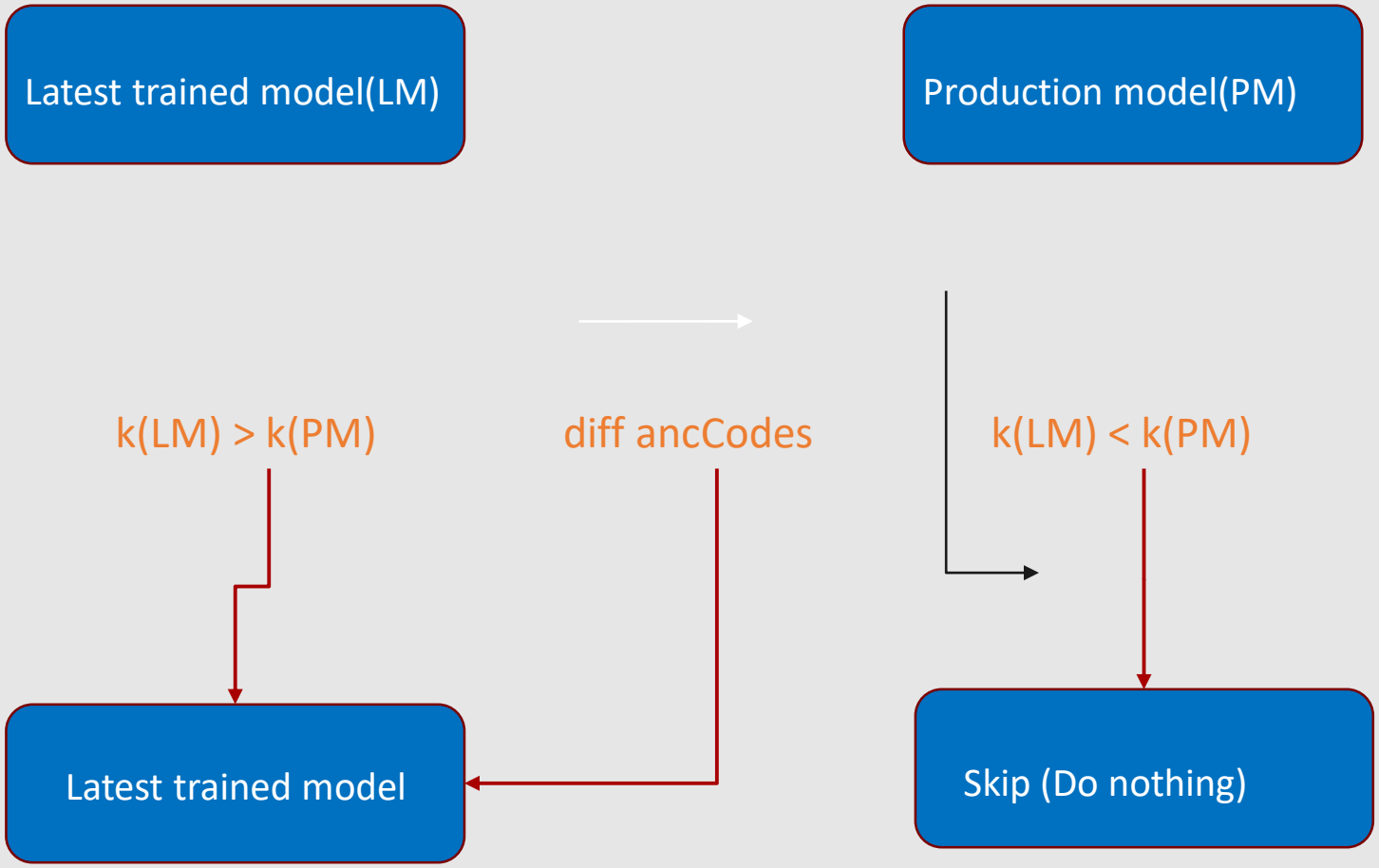
Retraining Pipeline Code

PROS: Model Evaluation

Get Model

Evaluate both models on latest data and compare Precision at k

Register Model



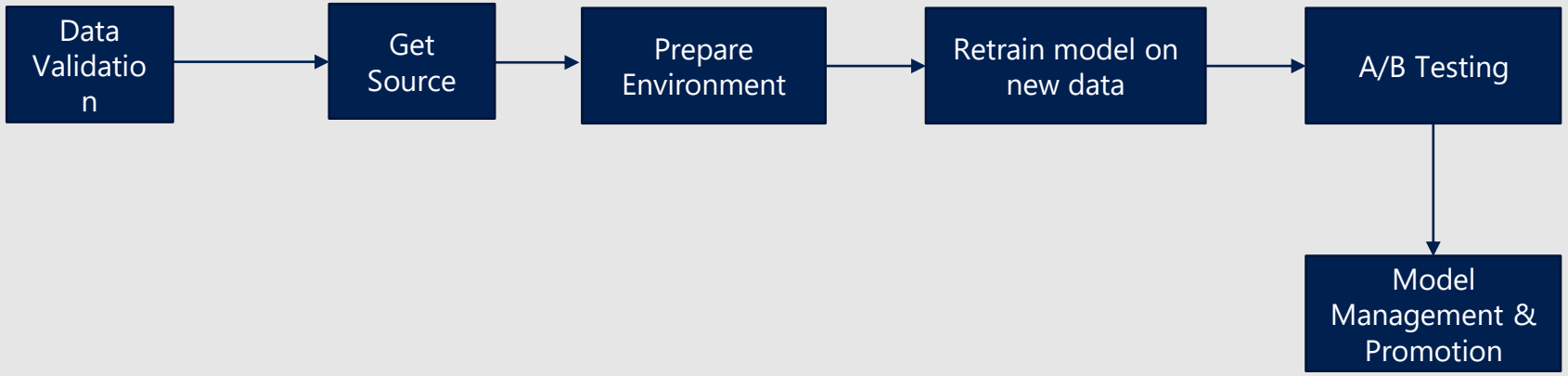
Future Enhancements

- Other Pipelines
- AML Pipelines
- DevOps Template for AI Apps
- Performance Monitoring
- AB Testing

Other Pipelines

ONNX, CoreML, WinML (end of sprint, every time there is a new model)

Retraining Pipeline (every night, or triggered on new data uploading to blob)
It will run in a pre-prod environment, so it has access to production data, and wouldn't be promoted unless it passes A/B tests against prod data.



Azure Machine Learning Pipelines

- No need to prepare environment on build agent
- Publish pipelines and trigger run with API call
- Unattended runs
- Reusability
- Tracking and versioning

More

- Performance Monitoring
- Containerized pipeline
- DevOps for AI using Databricks and Azure DevOps
- DevOps Template for AI Apps

A/B Testing

- Comparison of performance of production (A) and new (B) models over live traffic
- Commonly used in Internet use cases (advertising, recommendations)
- Successful model offline evaluation test will trigger an automatic setup of A/B test

Summary

Summary

- What is DevOps
- Traditional DevOps for Software Development
- Challenges in applying DevOps for AI App
- Build end to end automated pipeline with Azure DevOps and AML SDK

What Next?

- Explore the aidemos DevOps account :aidemos.visualstudio.com
- Implement the end to end automated AI DevOps pipeline
- Play with AML services sample notebooks
- Contact us @
  - Praneet Solanki: prsol
  - Dmitry Pechyoni dmpechyo

Resources

DevOps AI Code Repo:	aka.ms/devops-for-ai/code
DevOps AI Pipelines:	aka.ms/devops-for-ai/pipelines
Azure DevOps Account:	aidemos.visualstudio.com
Azure ML Notebooks:	github.com/Azure/MachineLearningNotebooks

Papers

- The ML Test Score: A Rubric for ML Production Readiness and Technical Dept Reduction

- Automating Large-Scale Data Quality Verification

Questions

Microsoft

Thank you for attending MLADS and continuing to build our strong community

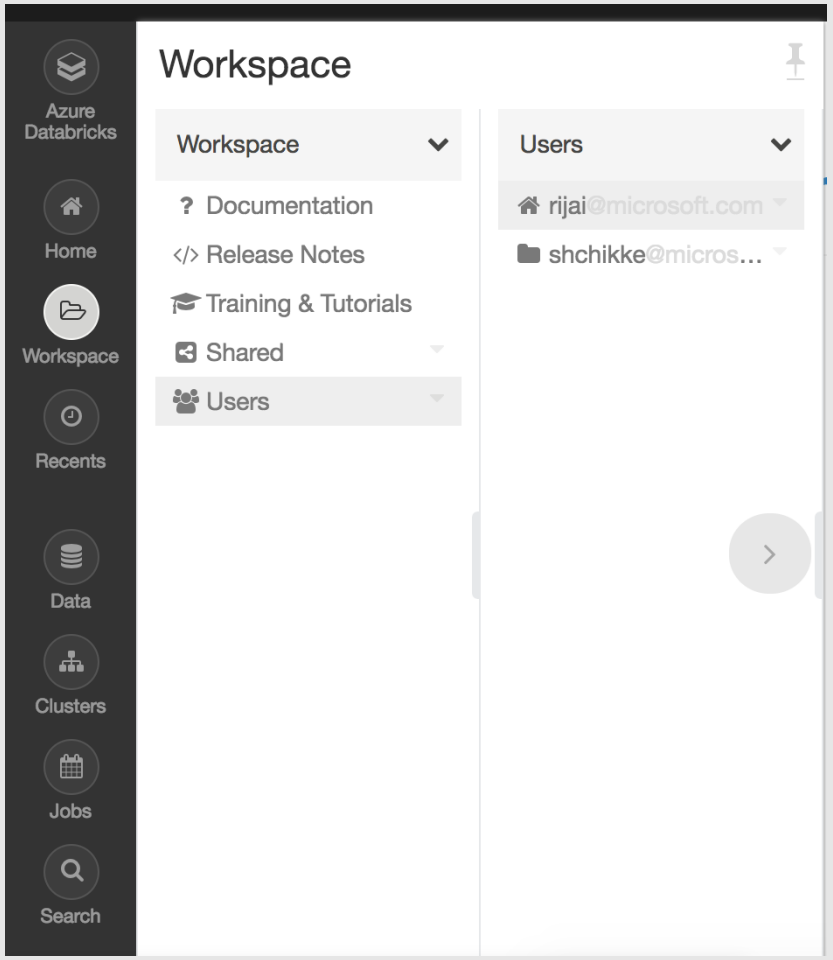
If you're interested in accessing a recorded version of content from the conference, it can be found here: http://aka.ms/fall2018mlads

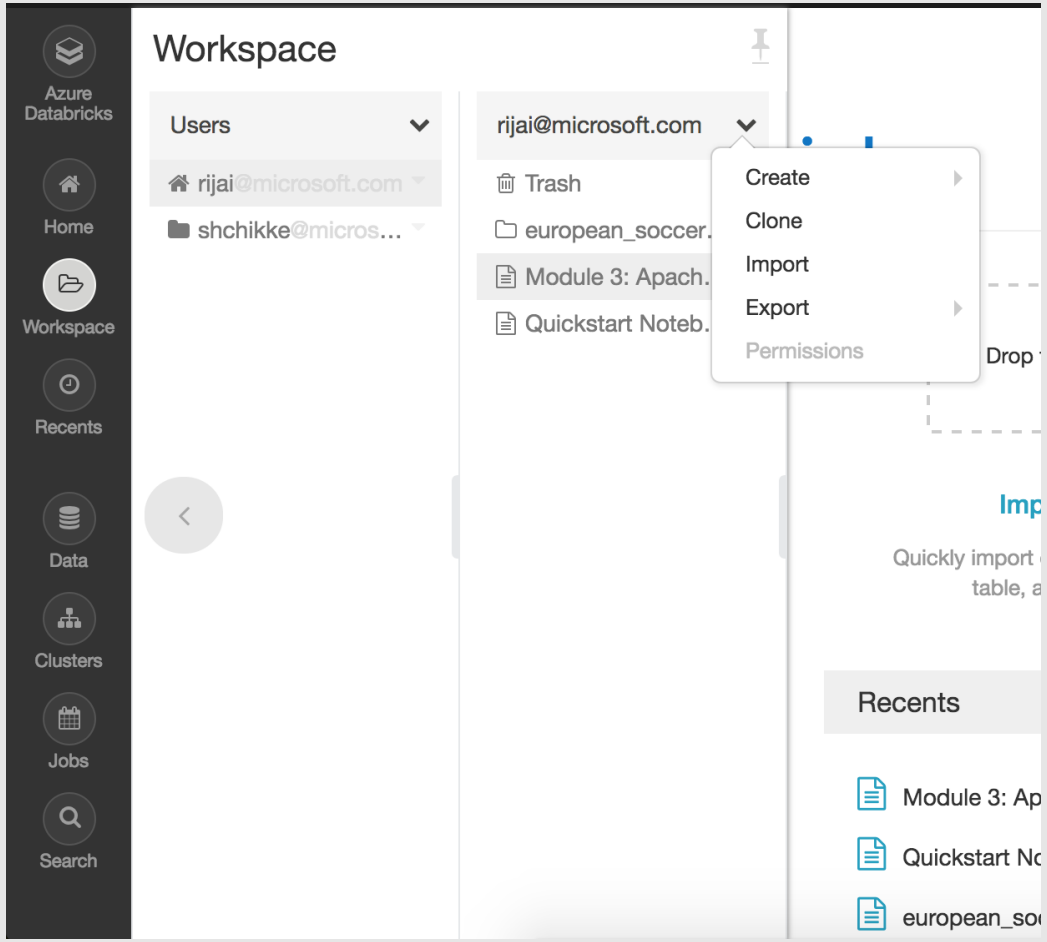
Microsoft

CI/CD with Databricks

Databricks concepts

- Workspace
- Notebooks
- Libraries
- Cluster
- Job





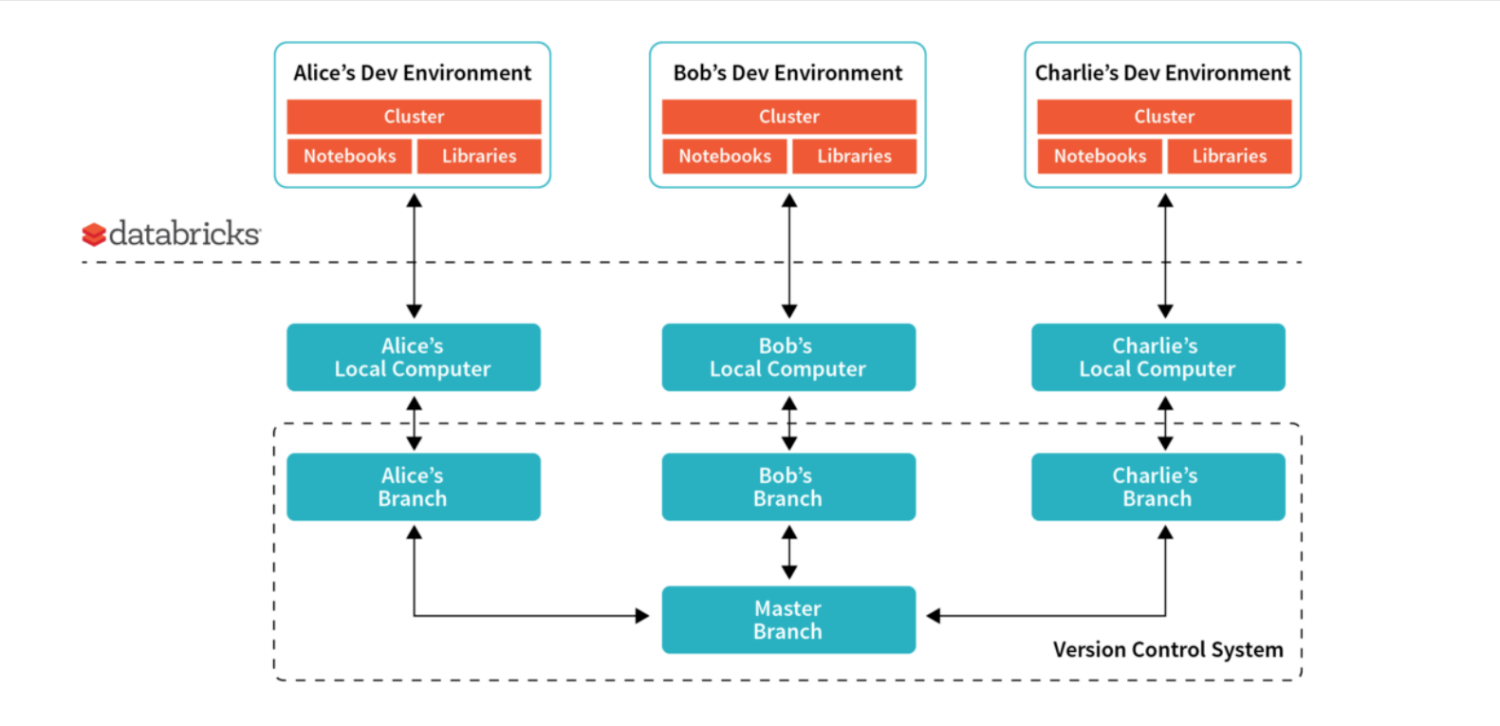
Recommended Code Structure

- Libraries
  - Put the core logic in them & upload to DBFS, attach it to clusters so it could be used by a notebook.
  - Easy to unit test the core logic.

- Notebook
  - Light weight and just focused on business logic.
  - Parameters.
  - Main class that calls the libraries.
  - Easy to change and maintain.

Source: https://www.slideshare.net/databricks/continuous-integration-continuous-delivery-83585634

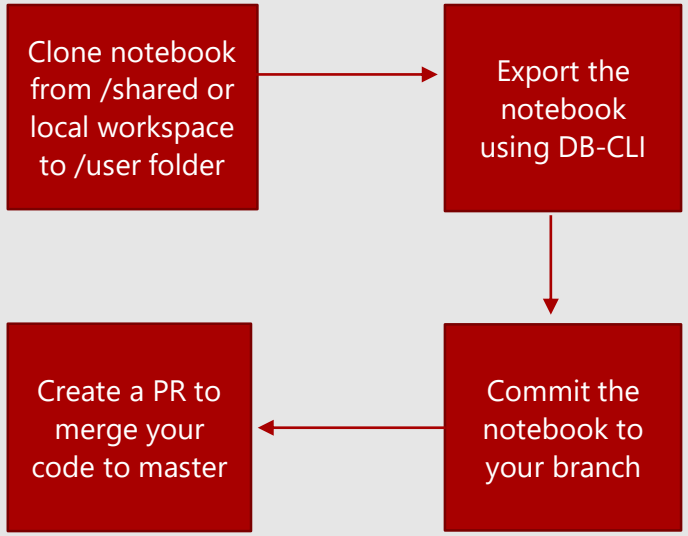
Version control of notebooks

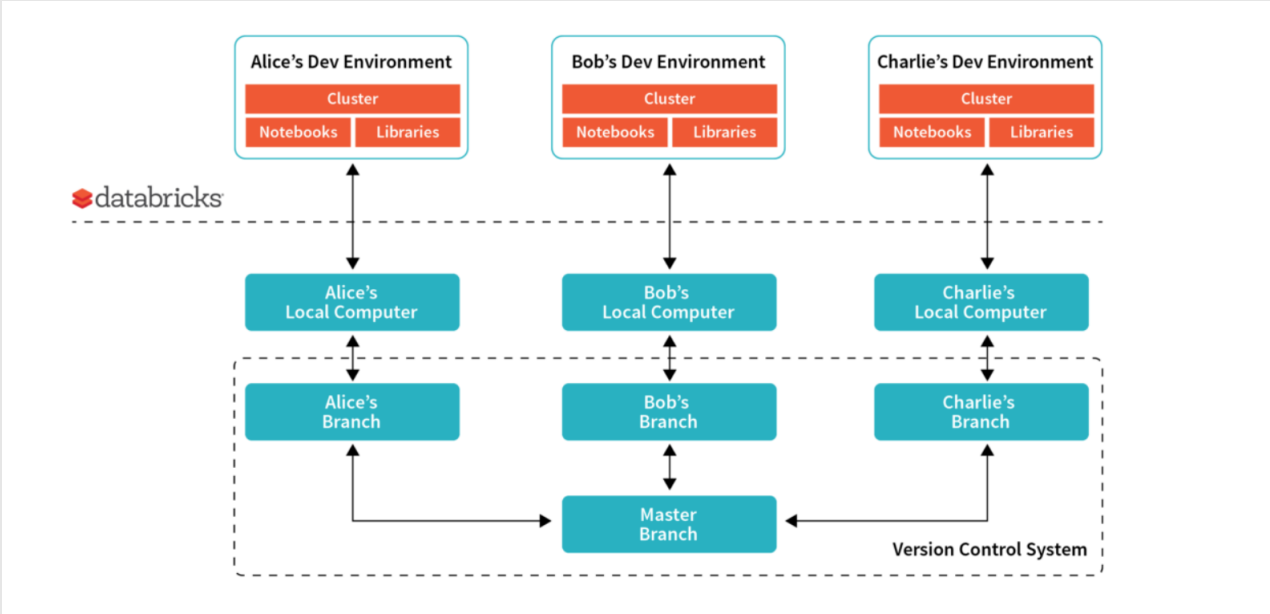


1. Upload new notebook from local workspace or clone an existing notebook from a shared folder in workspace that has restricted access.
2. Export user notebooks to local workspace (using Databricks CLI) and commit to user branch.
3. Merge user branch with master through a Pull request.

Source: https://databricks.com/blog/2017/10/30/continuous-integration-continuous-delivery-databricks.html

Version control of notebooks

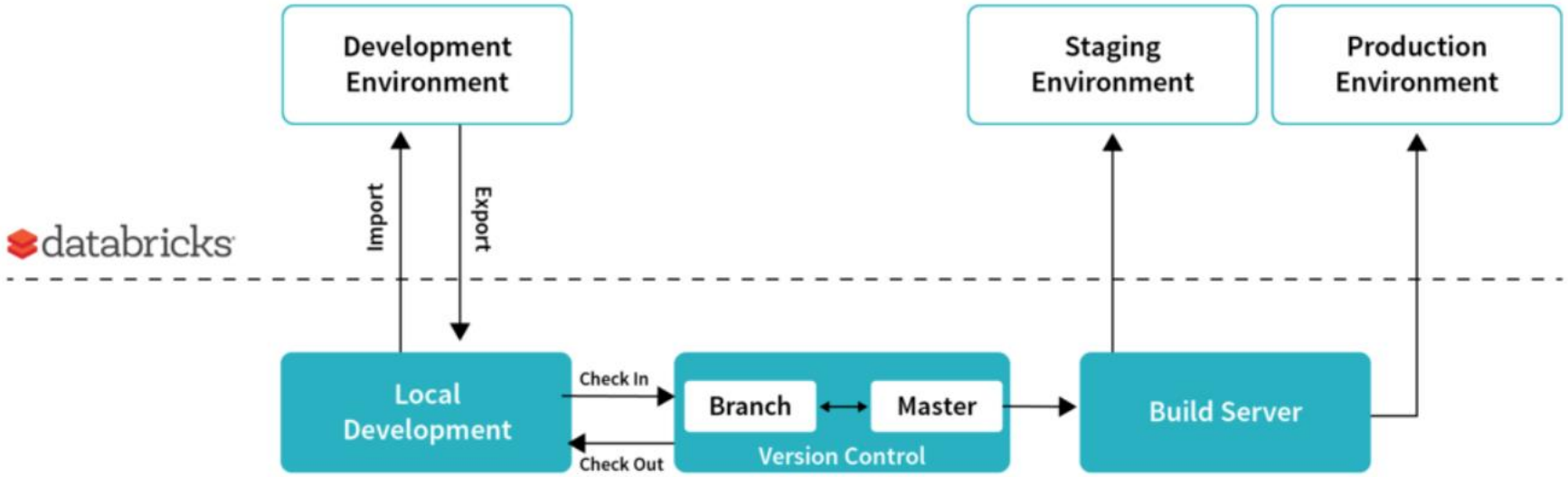




1. Version control user notebooks from user workspace to user branch, by exporting them using DB CLI.
2. Create a Pull Request from user branch against master
3. Notebooks from master is imported back to workspace under a shared folder that can be access controlled.
4. The notebooks from shared folder can be promoted to staging and prod environments.

Source: https://databricks.com/blog/2017/10/30/continuous-integration-continuous-delivery-databricks.html

CI/CD for a Data Pipeline on Databricks



- Interactive workspaces for data exploration and build pipeline on smaller dataset in development environment
- Notebooks are exported and version controlled when multiple users are working on the project.
- Once tested and vetted, data pipeline is pushed to staging environment to run on larger dataset
- From staging, pipeline is pushed to production to run on new incoming data.

Source: https://databricks.com/blog/2017/10/30/continuous-integration-continuous-delivery-databricks.html

Build Pipeline

- Notebooks and Libraries are checked into source control.
- Core logic libraries are unit tested.
- Notebook and Library artifacts are pushed to artifact server (ex. Azure Artifact, maven)

Source: https://www.slideshare.net/databricks/continuous-integration-continuous-delivery-83585634

Release Pipeline: Staging

- Push the libraries to a **staging** folder in Databricks File System (DBFS) using the DBFS API/CLI.
- Push the notebooks to a **staging** folder in the Databricks workspace through the Workspace API/CLI.
- Create a job using the Job API/CLI with **staging** configuration, provide the libraries in DBFS and point to the notebook to be executed by the job.
- Publish results from the run.

Release Pipeline: Production

- Push the new **production** ready libraries to a new DBFS location.
- Push the new **production** ready notebooks to a new folder under a restricted production folder in Databricks workspace.
- Modify the job configuration to point to the **production** notebook and library location so that the next run of the job can pick them up and run the pipeline with the new code.

Source: https://www.slideshare.net/databricks/continuous-integration-continuous-delivery-83585634