MLOps, training e inference pipelines con Azure ML

Gianni Rosa Galina

Sr. R&D Engineer @Deltatre

Vito Flavio Lorusso

PM @Microsoft

Clemente Giorio

Sr. R&D Engineer @Deltatre

Platinum Sponsor

Microsoft

avanade

REPLY CLUSTER

ellycode
WHAT DO YOU NEED?

Gold Sponsor

LOBRA

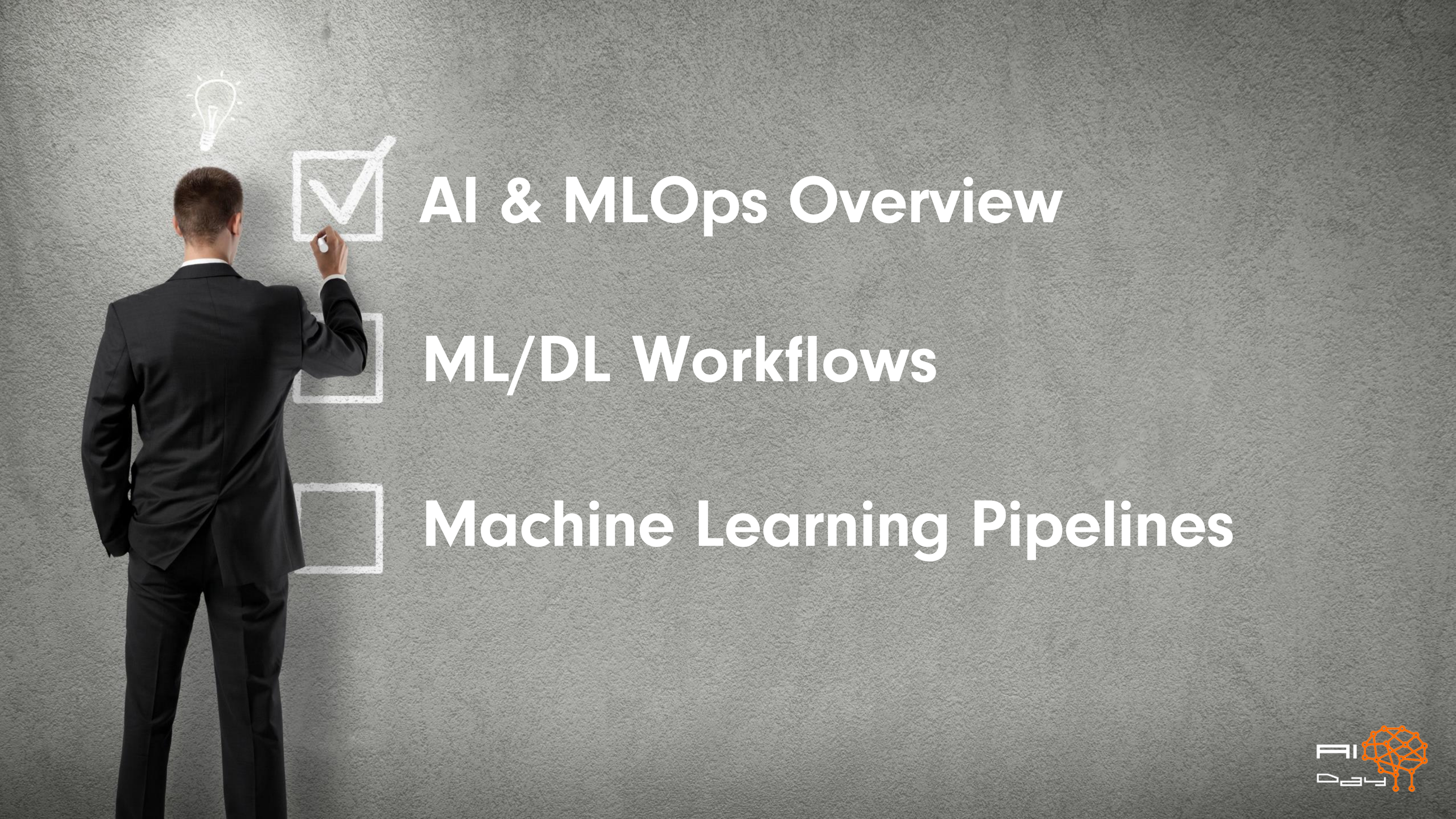
OVH

Technical Sponsor

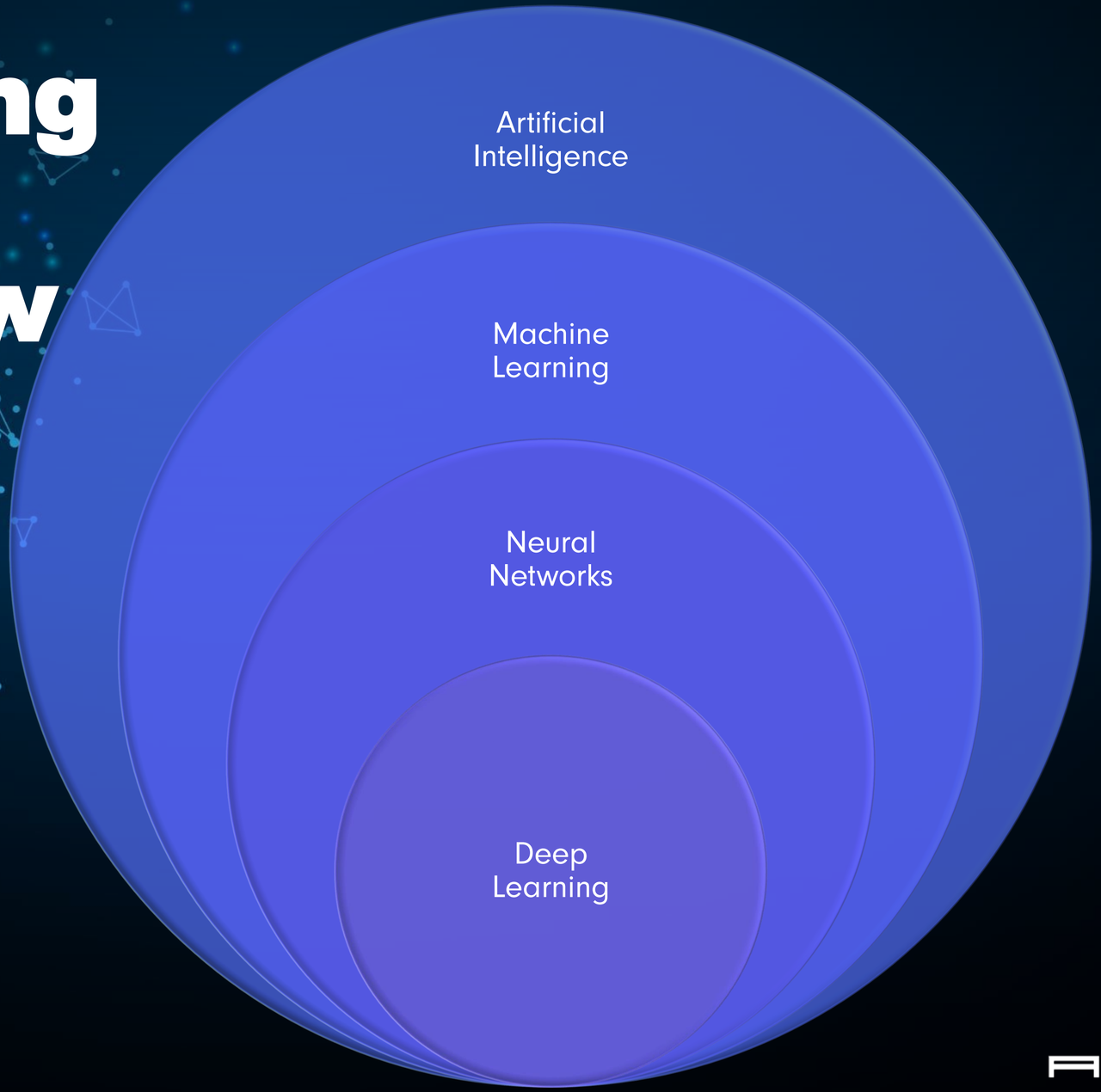
NET CODE

JET BRAINS

Packt

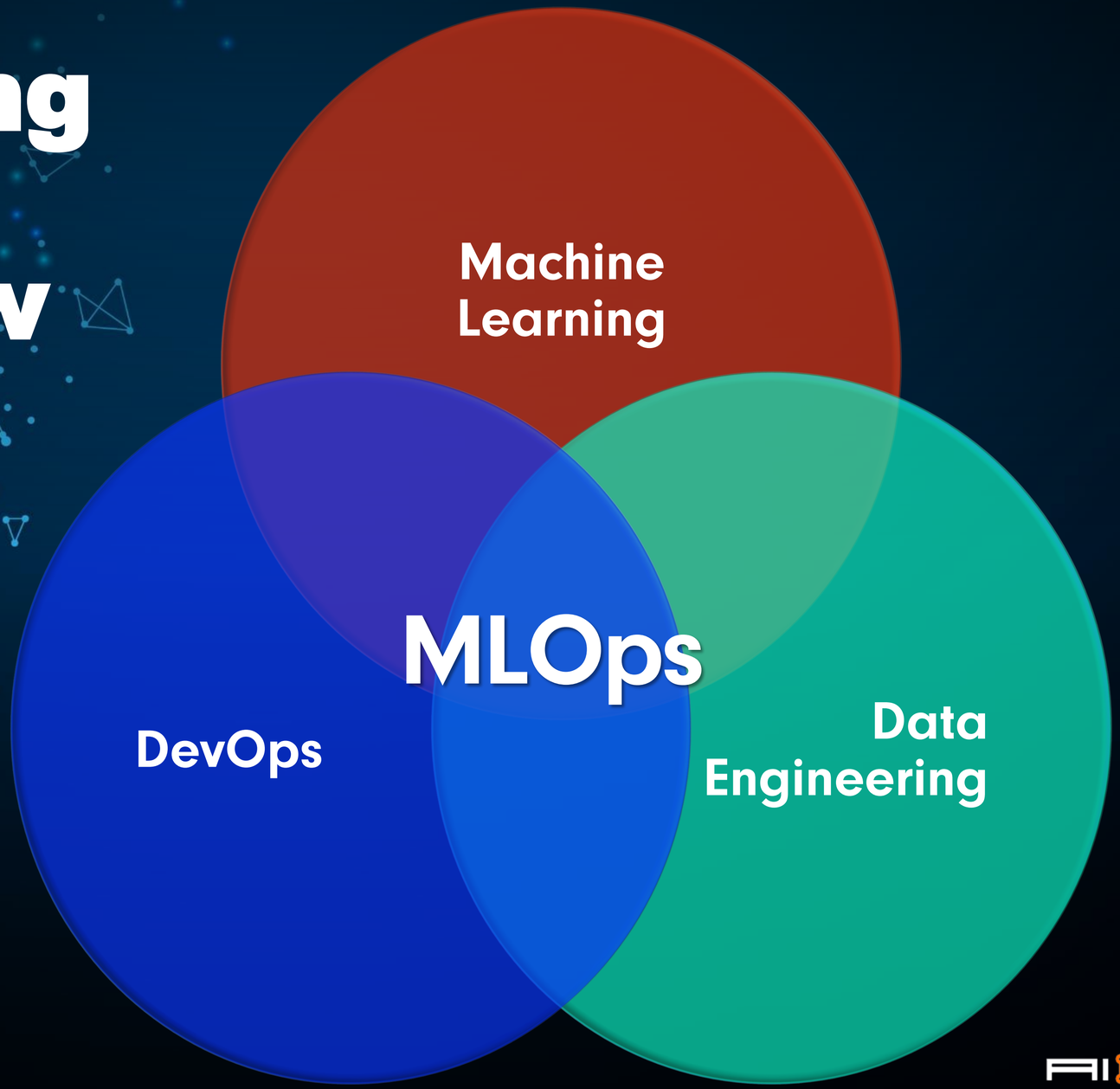

AI & MLOps Overview
ML/DL Workflows
Machine Learning Pipelines

MachineLearning and MLOps overview

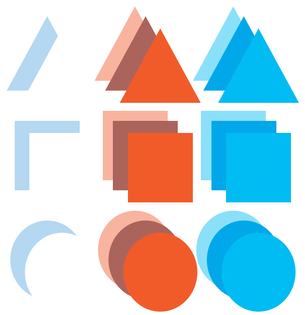


AI Day

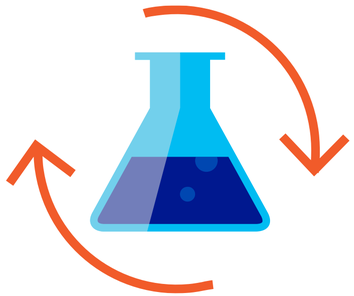
MachineLearning and MLOps overview



Typical ML Workflow

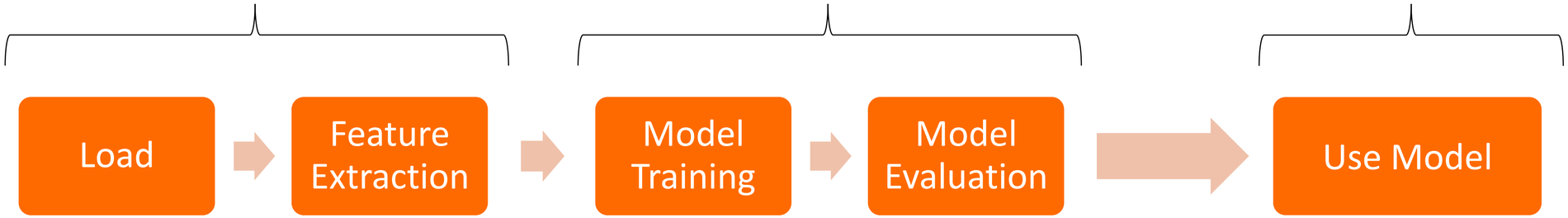


Data preparation

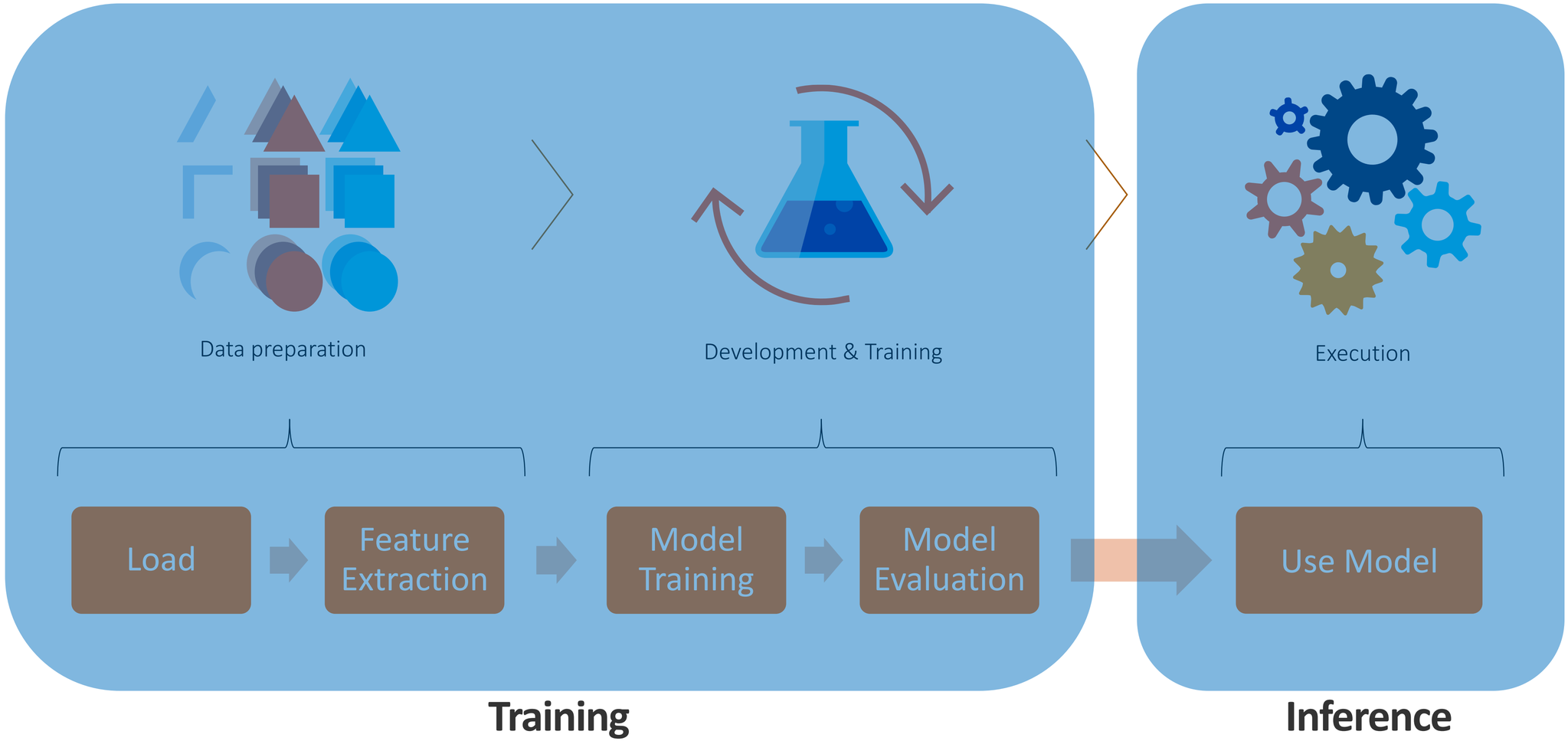


Development & Training

Execution



Training vs Inference/Scoring



Training

“Development” phase of a **Machine Learning project**

Usually **lead** and executed by **Data Scientists**
from start to end, **through experiments** and **trial & error**

Iterative process of **variable duration and results**
(until **specified target metrics are achieved**)

Typically, **resource & time intensive**
usually done on (lots of) CPUs and GPUs or AI-accelerators (TPU, FPGA)

Scalability ➔ reduce training time or **improve quality**

size of datasets and/or models + parallel processing = hardware/storage/bandwidth

Inference

“Production” phase of a **Machine Learning project**

Usually **lead** by **Software/AI Engineers** and **DevOps**

Data scientists’ role is mainly to **monitor** model **behavior** in the field

Different requirements**,** tools and frameworks

compared to training, more **similar** to **traditional development**

Deploy on **edge device** or **on-prem/cloud datacenter**

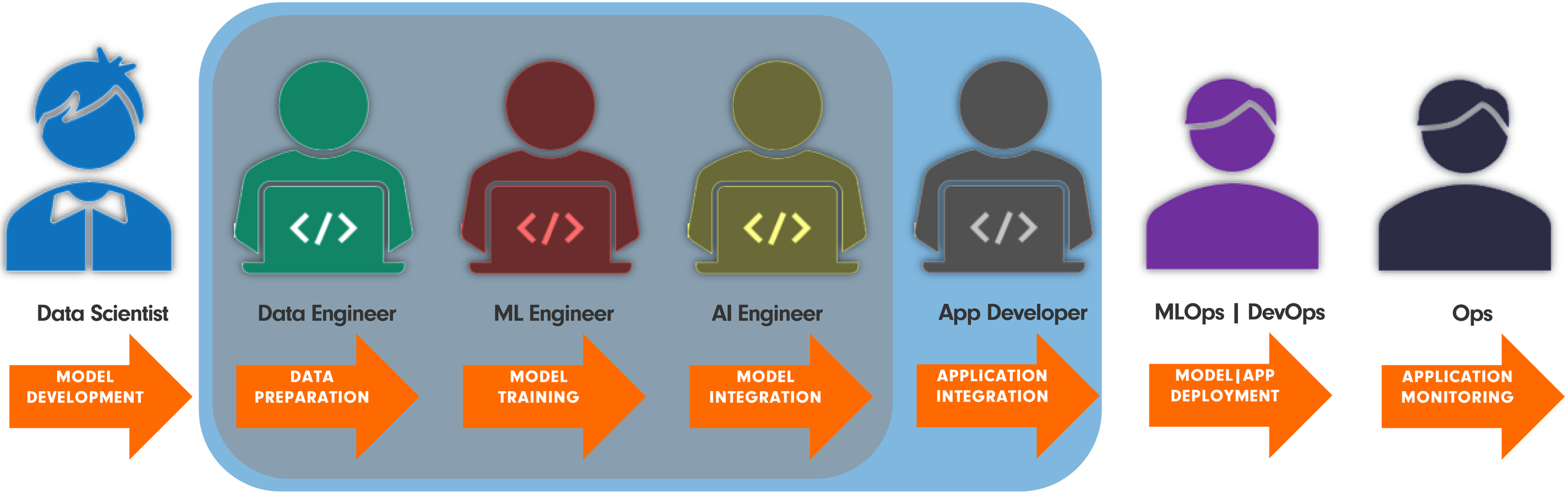
Scalability ➔ increase # of requests/sec

optimize models, parallel processing, increase/scale scoring instances

New data **Inference app/service** **Predictions/Scores**

THE AI-TEAM

Typical ML Project Team



MLOps: DevOps on ML components

Model reproducibility & versioning

- Track, snapshot & manage assets used to create the model
- Enable collaboration and sharing of ML pipelines

Model auditability & explainability

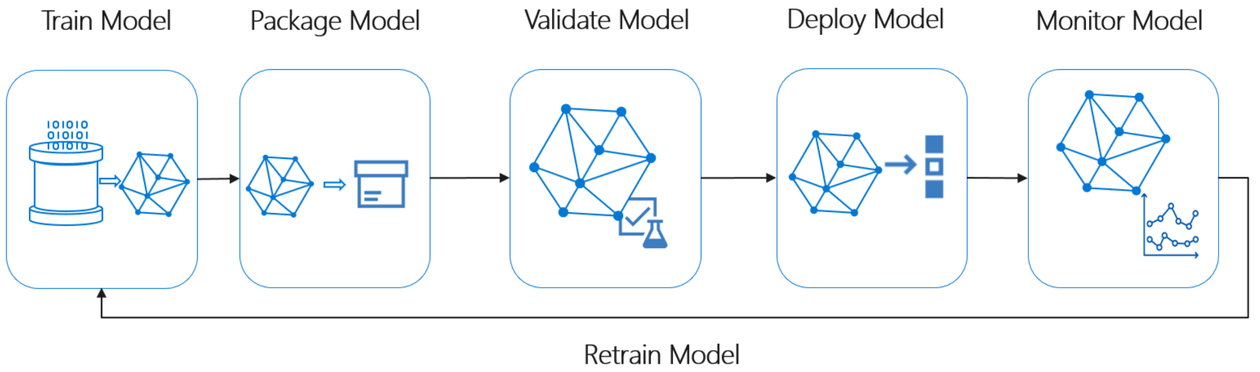
- Maintain asset integrity & persist access control logs
- Certify model behavior meets regulatory & adversarial standards

Model packaging & validation

- Support model portability across a variety of platforms
- Certify model performance meets functional and latency requirements

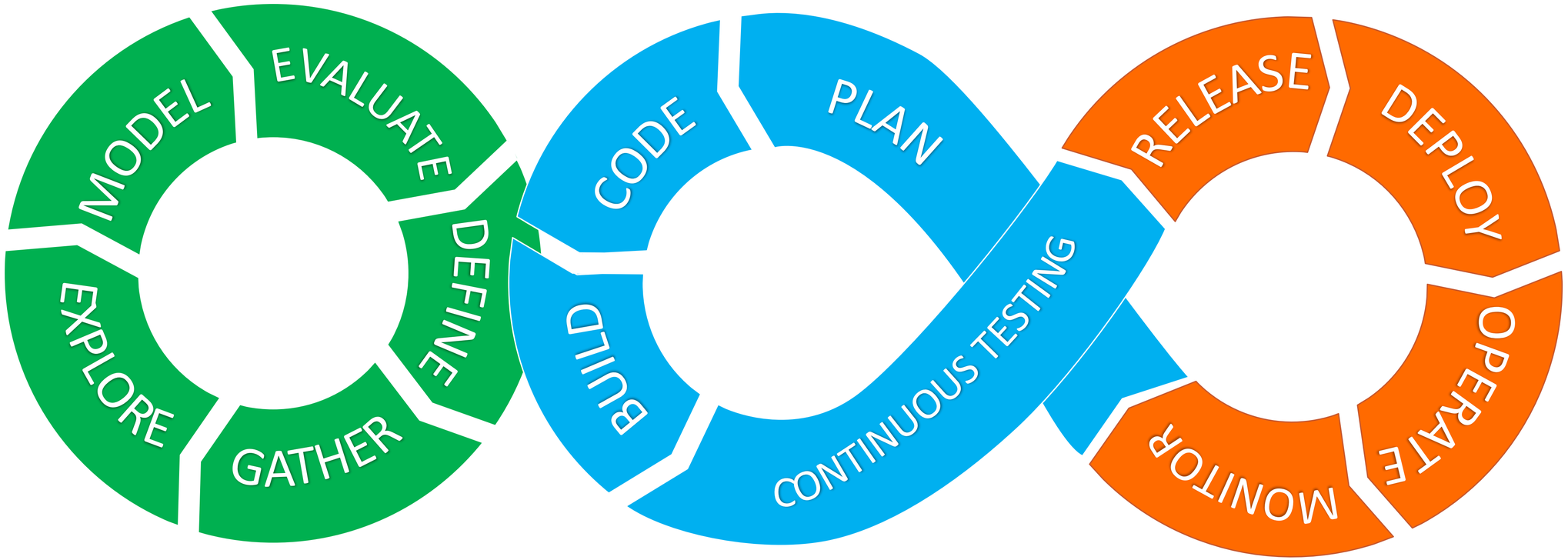
Model deployment & monitoring

- Release models with confidence
- Monitor & know when to retrain by analyzing signals such as data drift




HELPFUL
TIPS!
AI
Day

The forgotten exploration phase



Dataset management

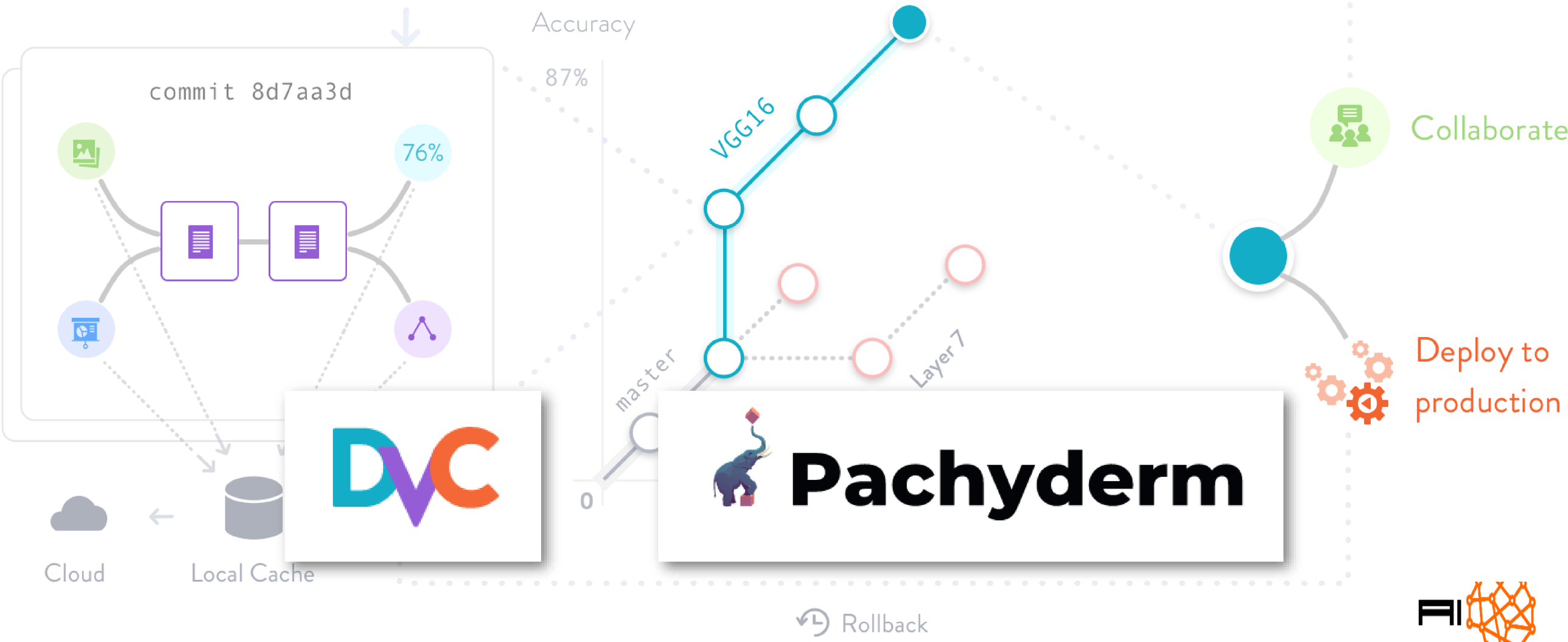


Image taken from DVC.org

Train, Validation and Test Split

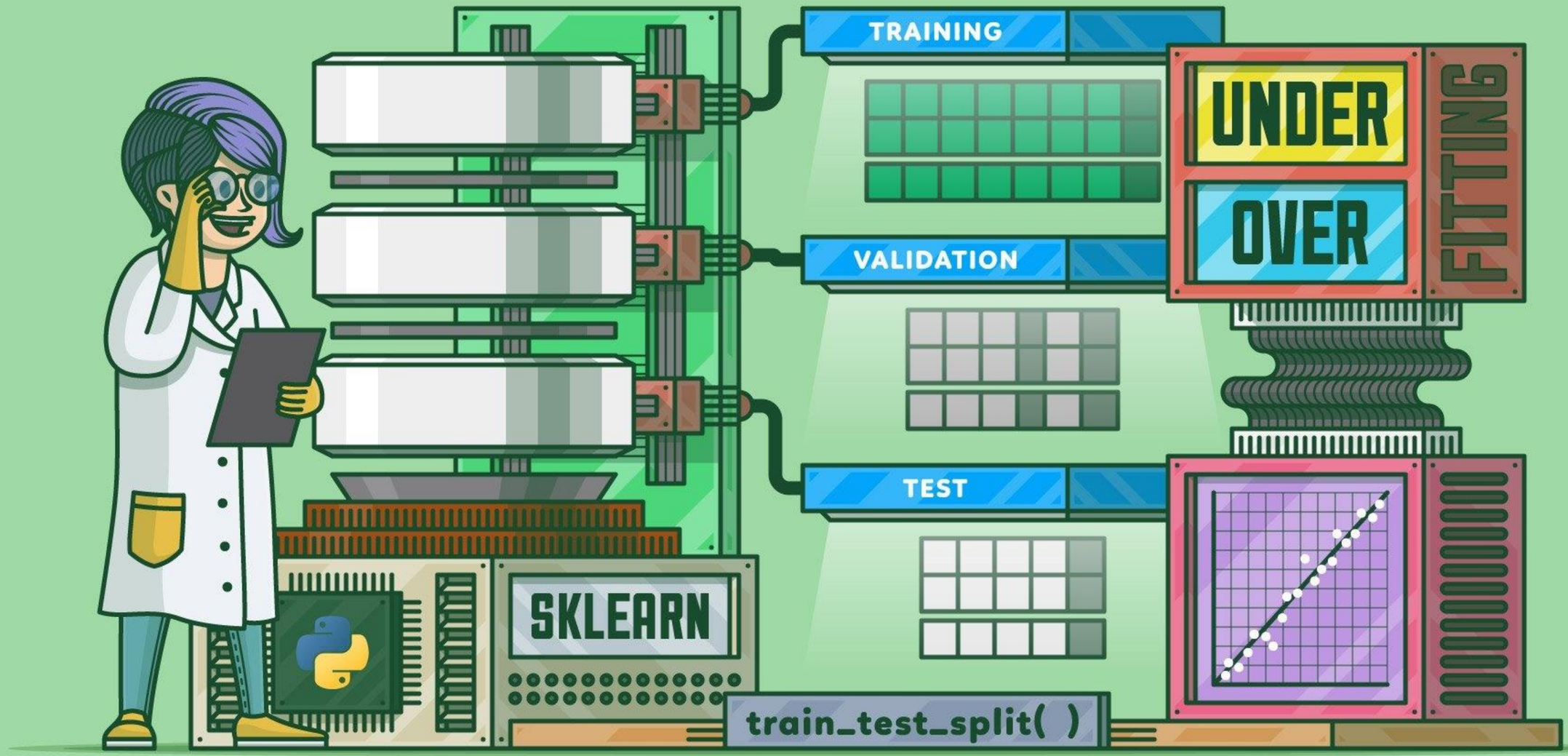
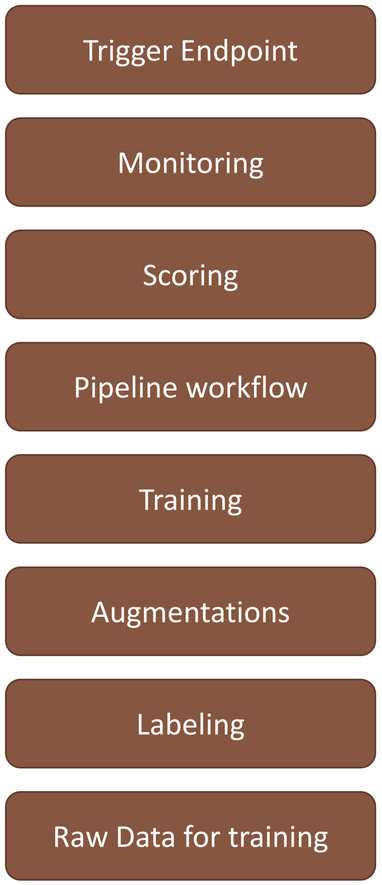


Image taken from Real Python

ML/DL Workflows

The building blocks of “usable” Machine Learning



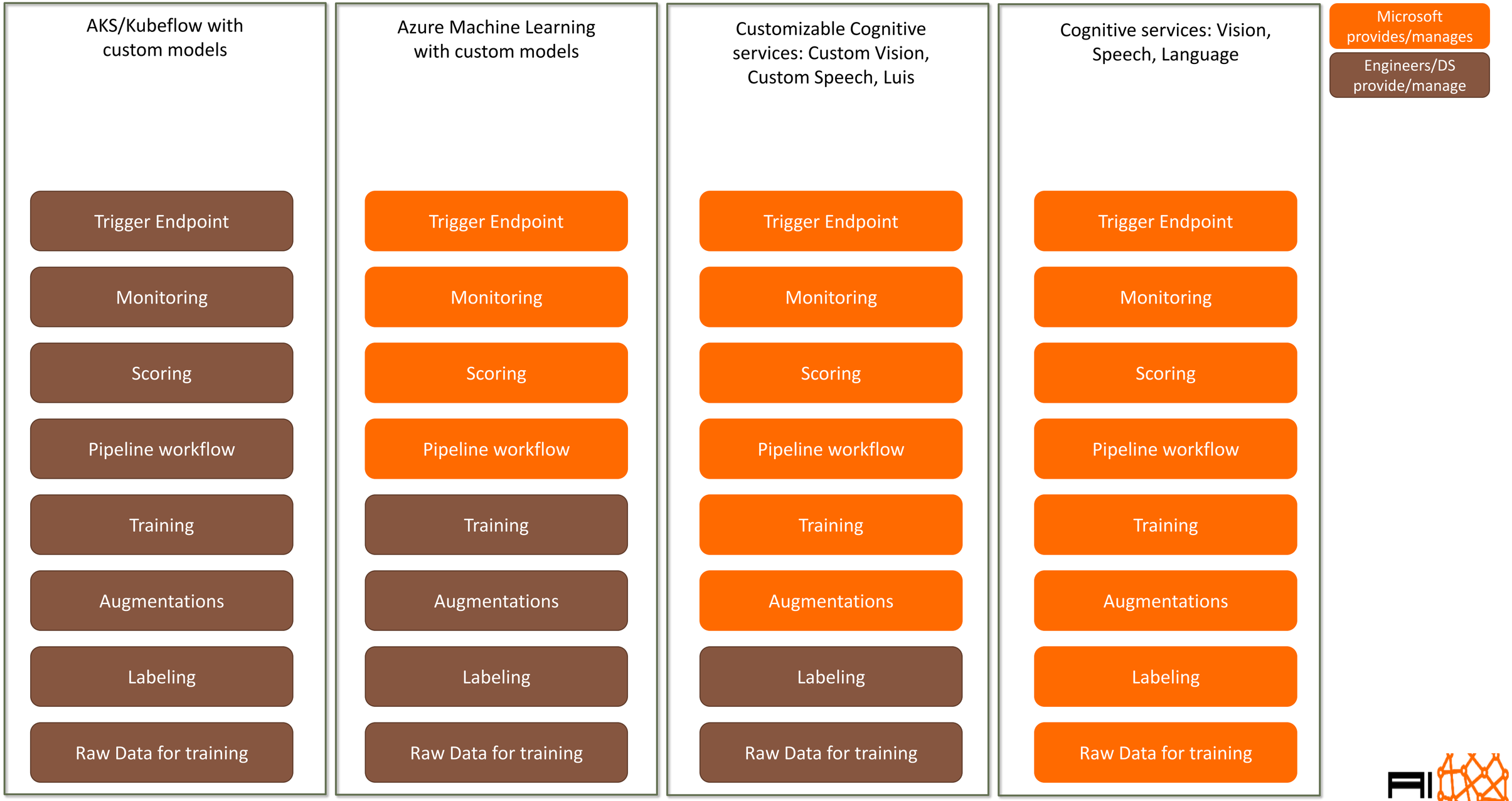
First of all, you need **DATA** (possibly good data)

Most ML problems can’t be solved without **Labeling**

Augmentations or processing make data usable across different sources for our chosen algorithm

Most ML problems can’t be solved with just one ML model. **Pipelines** and **Workflows** are key to chain **transformations** and **model evaluations and results**

In order to use and maintain a ML pipeline proper **MLOps** and results **Monitoring** must be in place


AKS/Kubeflow with custom models
Azure Machine Learning with custom models
Customizable Cognitive services: Custom Vision, Custom Speech, Luis
Cognitive services: Vision, Speech, Language
Microsoft provides/manages
Engineers/DS provide/manage
Trigger Endpoint
Monitoring
Scoring
Pipeline workflow
Training
Augmentations
Labeling
Raw Data for training

Trigger Endpoint
Monitoring
Scoring
Pipeline workflow
Training
Augmentations
Labeling
Raw Data for training

Azure Cognitive Services

Vision
recognize, identify, caption, index, and understand what is in your pictures or videos.

Language
NLP tasks such as evaluate sentiment and learn how to recognize what users want.

Speech
convert speech into text vice versa, translate between languages, speaker verification and recognition

Decision
system that helps to give best recommendations for informed and efficient decision-making.

Search
search-APIs that gives you the ability to comb billions of contents with a single API call

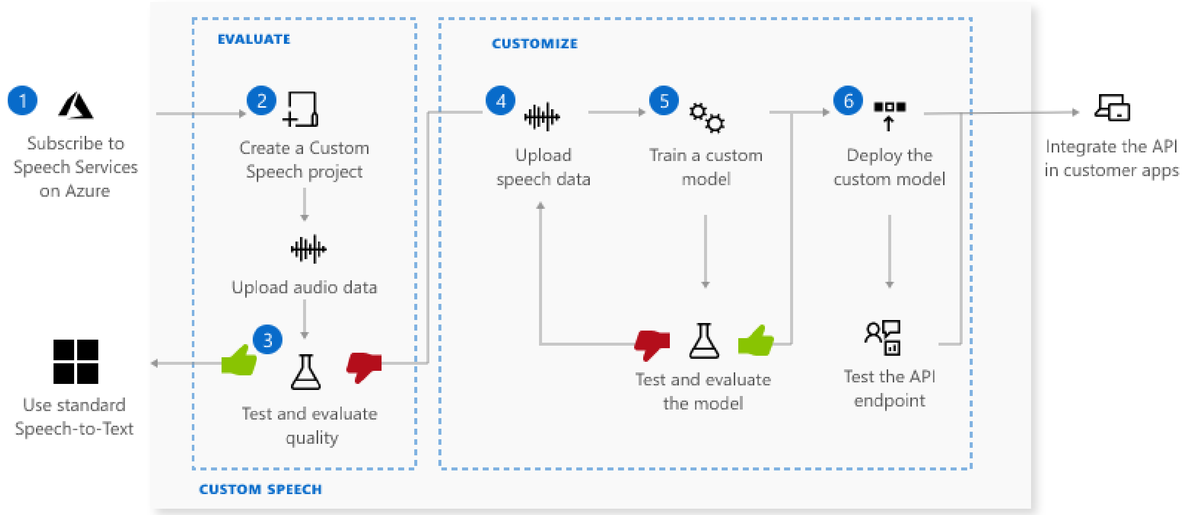
Cloud-based services to help you build cognitive intelligence into applications with very little knowledge of AI/ML or data science:

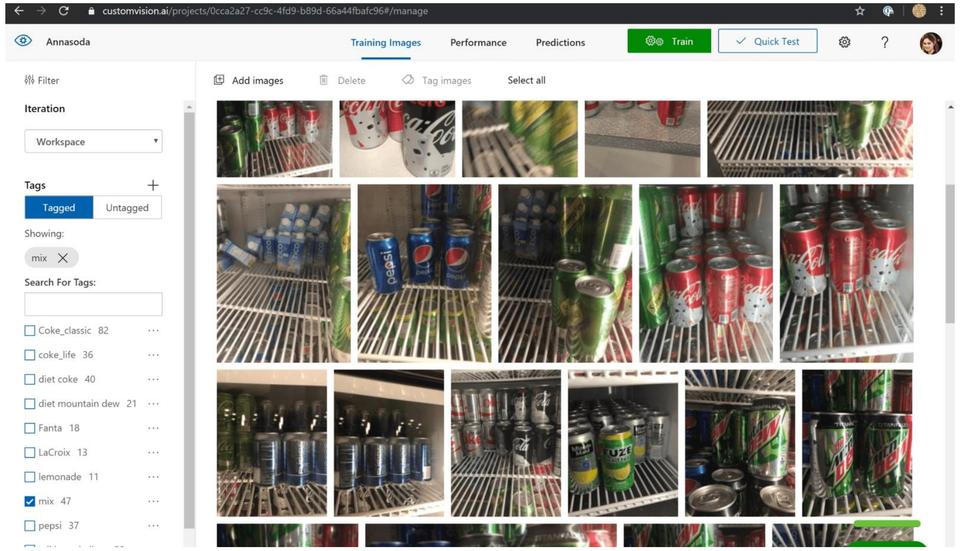
Available with REST APIs and client library SDKs available

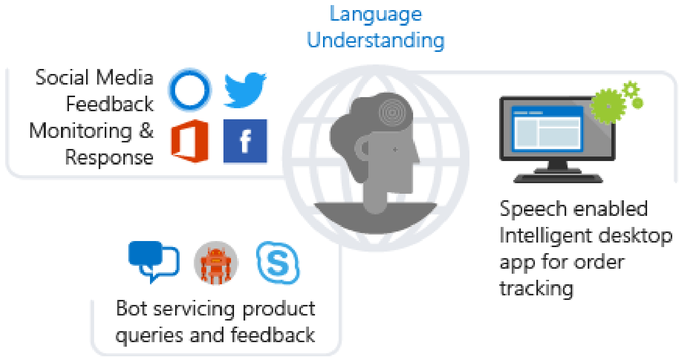
They comprise various **AI services** such as computer vision, audio processing and speech processing and understanding

Trigger Endpoint
Monitoring
Scoring
Pipeline workflow
Training
Augmentations
Labeling
Raw Data for training

Customizable Vision, Speech and Language







Still **developer** "friendly" ML services with ways to customize the training dataset to create *domain specific* ML models

Trigger Endpoint
Monitoring
Scoring
Pipeline workflow
Training
Augmentations
Labeling
Raw Data for training

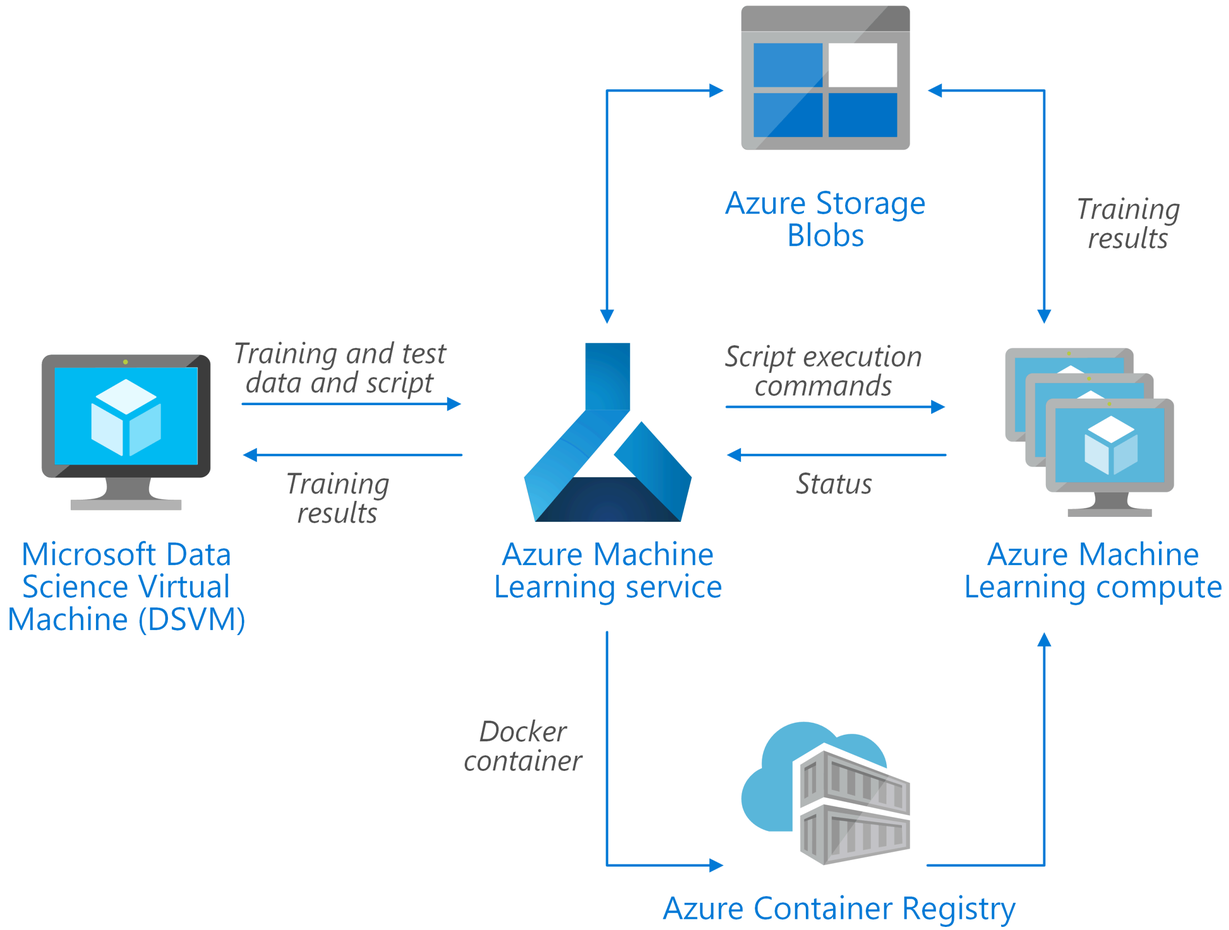
Azure Machine Learning

(managed workflow, pipeline and streamlined MLOPS)

AML is a cloud-based environment that accelerates the **end-to-end machine learning lifecycle**.

You can use AML to:

- Train
- Deploy
- Automate
- Manage
- Track ML models



https://azure.microsoft.com/services/machine-learning/

Doing Data Science with AML

Develop on your **local machine** using the AML **Python SDK** or R SDK

Open-source frameworks such as PyTorch, TensorFlow, scikit-learn, and others

Tools for ML workflows, including:

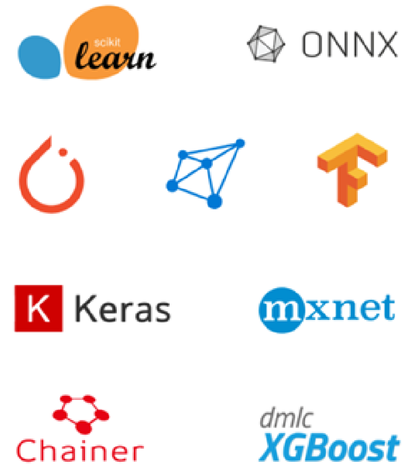
- **Jupyter notebooks**: widely used in the **exploratory phase** by data scientists because they allow to easily create, view, and share code
- **Azure Machine Learning CLI**: extension to automate ML activities in AML such as run experiments or deploy models

AML Studio: web portal combining no-code and code-first experiences for data science

Languages

Development tools

Frameworks



Cognitive Services vs AML

Azure cognitive services

Cognitive Services are for developers without machine-learning experience

- A Cognitive Service provides a (general-purpose) trained model, made available using a SaaS REST API or an SDK.
- Services can be used and integrated within minutes, depending on the scenario.
- Cognitive Service readiness is ideal for who has **no AI knowledge and deals with general problems**.

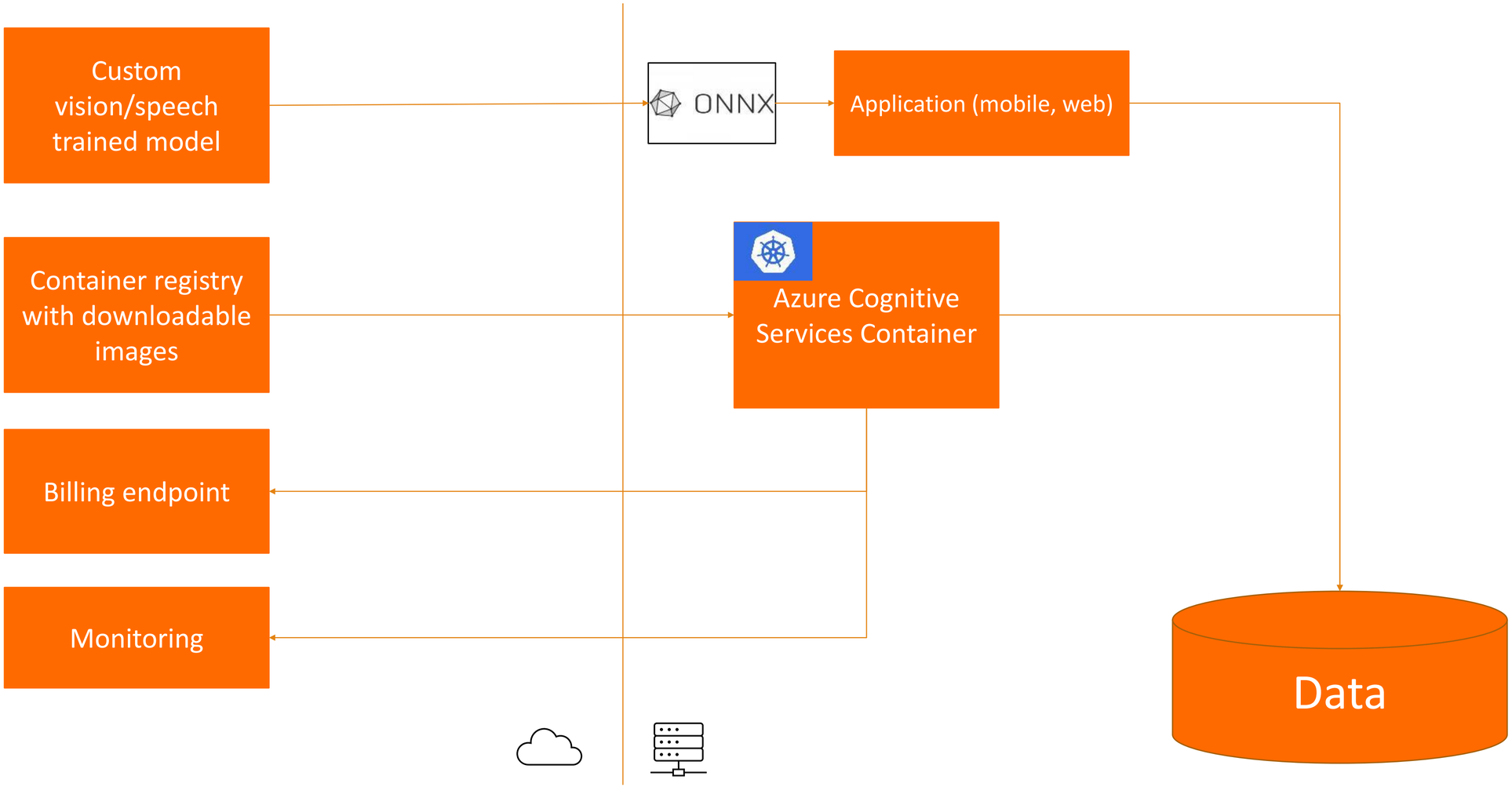
Azure machine learning

Azure Machine Learning is tailored for data scientists

- AML works for highly specialized or specific problems, often requires suiting all ML operations (data collections, cleaning, training, evaluating, …).
- Model implementation may require weeks, if not months, and engineering, maintaining and serving them requires infrastructure + software engineering + data science skills.

Familiarity and expertise with data science are required

Does it all need to be in the public Cloud?



DIY Machine Learning

Kubeflow

FRAMEWORKS/LIBRARIES

a collection of GPU-accelerated Data Science Libraries

JUPYTER

provides an interactive interface to cluster resources

KUBEFLOW

interfaces with Kubernetes simplifying the administration of Kubernetes services

KUBERNETES

acts as the cluster's operating system, keeping track of hardware resources and scheduling as needed

LINUX

tested and validated Linux operating system with NVIDIA drivers and CUDA libraries for optimal performance

GPUs

available to users from the cluster

Image taken from Nvidia.com

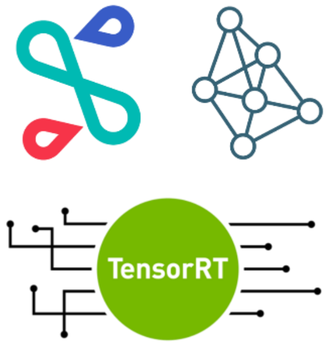
Kubeflow components

Kubeflow

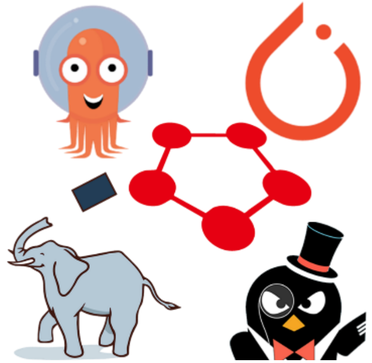
Notebooks

Tensorflow
Modelling Training

Kubeflow Pipelines

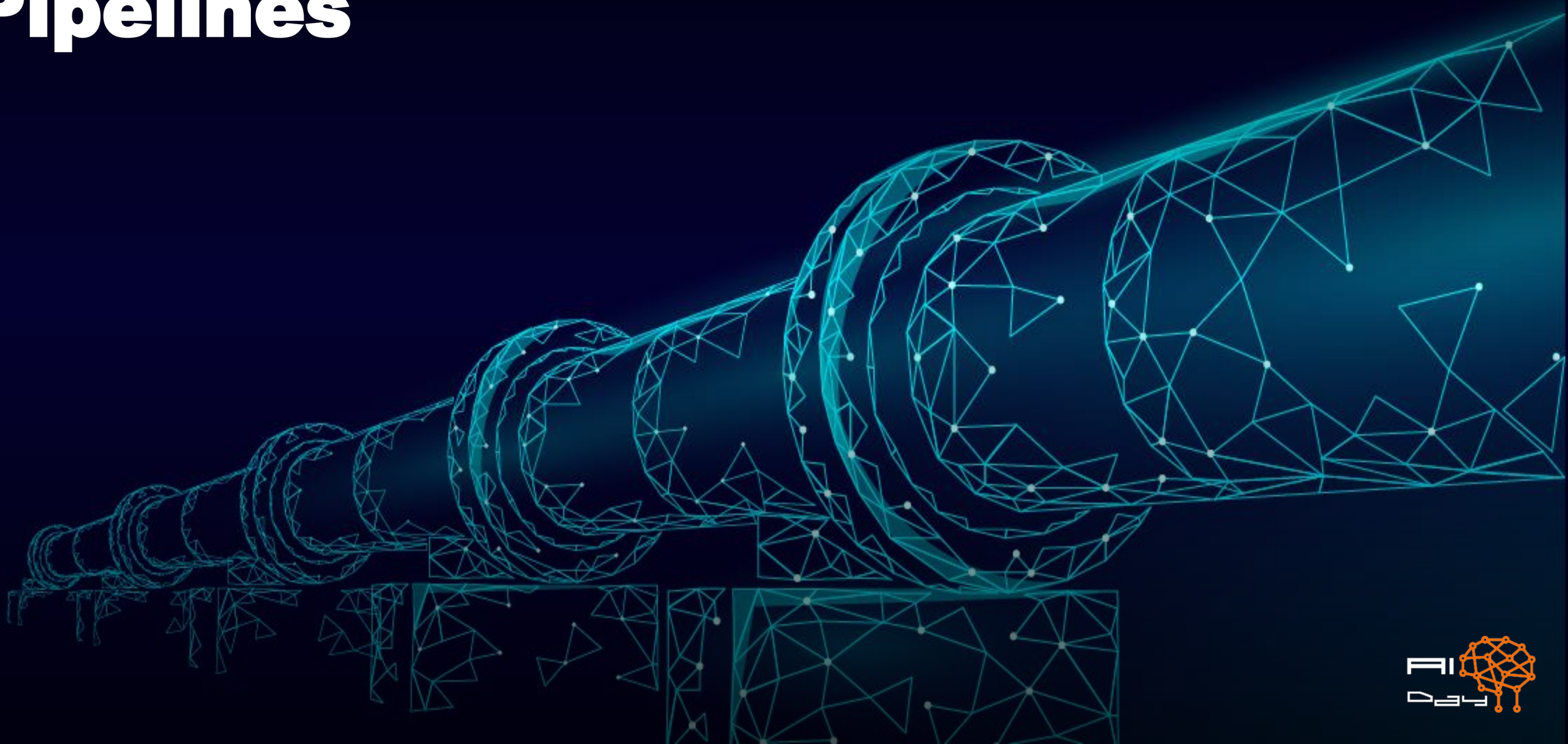


Model Serving



Multi-Framework

Machine Learning Pipelines



AI Day

Experiment's Python Production-level

Making Python Notebooks become production-level code
How to (unit) test Machine Learning models and pipelines

From C# to Python – by Tess Ferrandez

https://www.youtube.com/watch?v=LDVVvwoVtLY

What are AML pipelines?

AML Pipeline is an independently executable workflow of a **complete ML task**. Subtasks are encapsulated as a **series of steps** within the pipeline.

Step: discrete processing action designed for a specific task with its own resources and environment

AML **automatically orchestrates** all the dependencies between pipeline steps. This may include:

- Spinning up and down Docker images
- Attaching and detaching compute resources
- Moving data between steps consistently and automatically

Pipeline benefits

Simplicity

Speed

Repeatability

Flexibility

Modularity

Versioning and tracking

Quality assurance

Cost control

Controlled Runtime Environment

What type of pipelines?

DevOps CI/CD Pipelines

Related to implement CI/CD processes. They check:

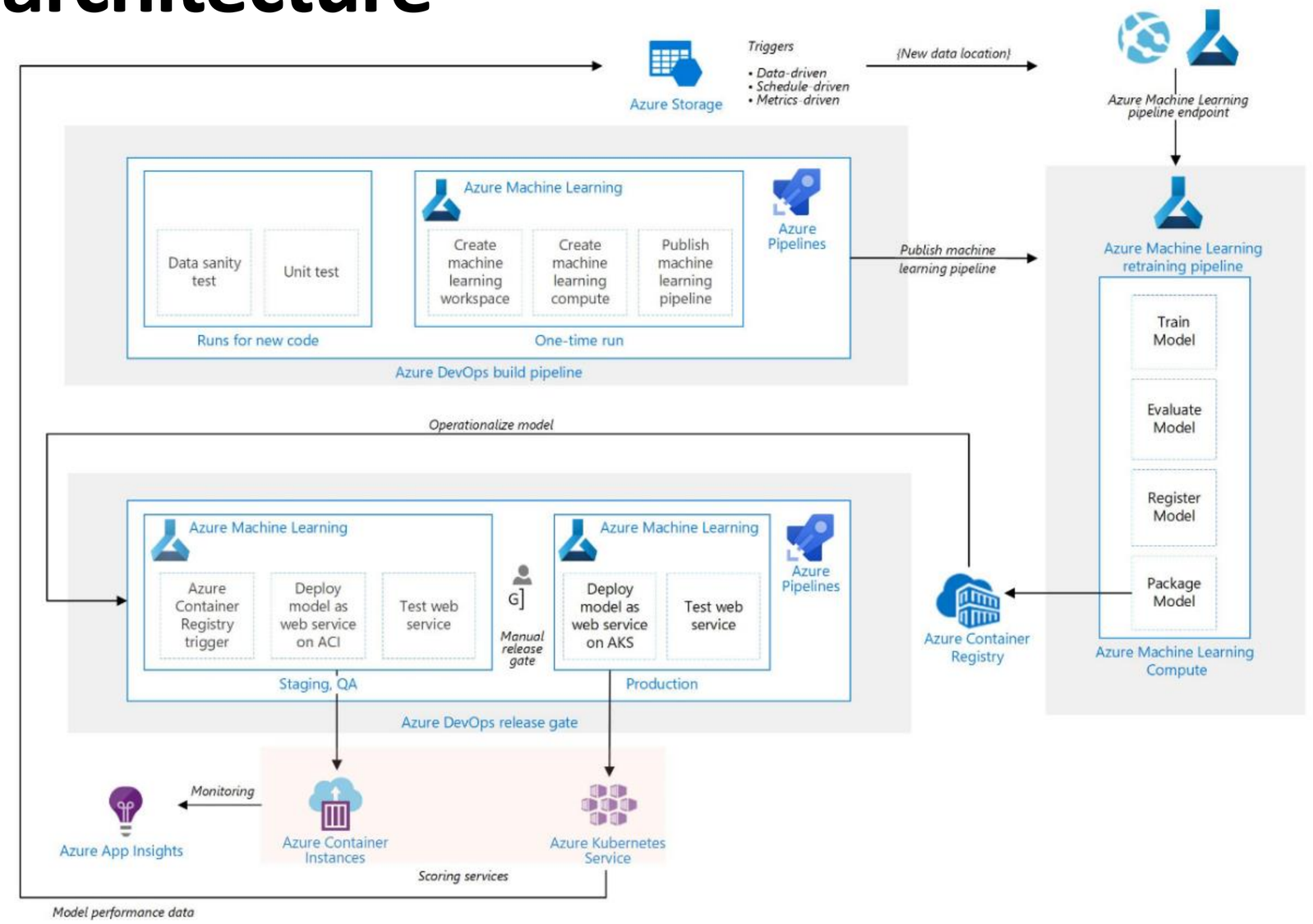
- Code quality
- Makes sure that any changes in the repository are deployed to the AML Service once they are committed/tested/approved, etc.

Azure Machine Learning Pipelines

Related to entities in the AML service such as

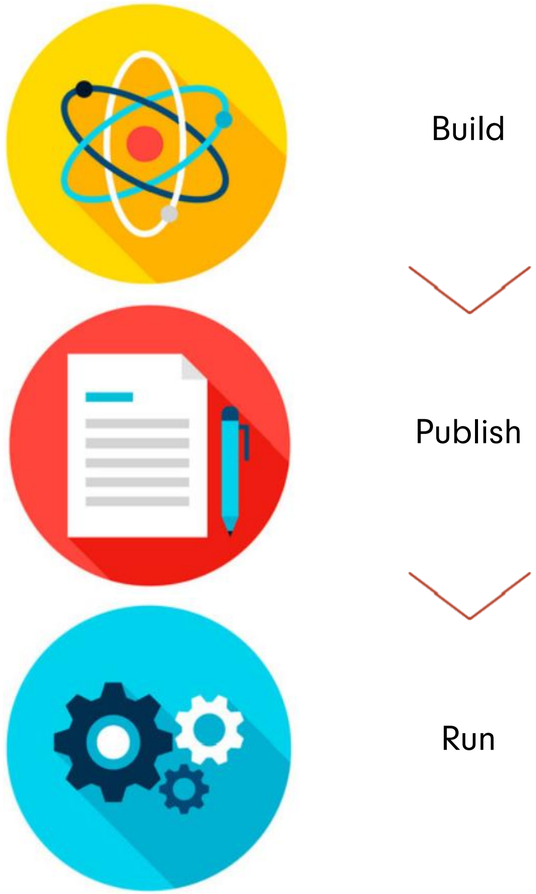
- Training models
- Scoring models
- Data augmentation pipelines

DevOps architecture



https://docs.microsoft.com/azure/architecture/reference-architectures/ai/mlops-python

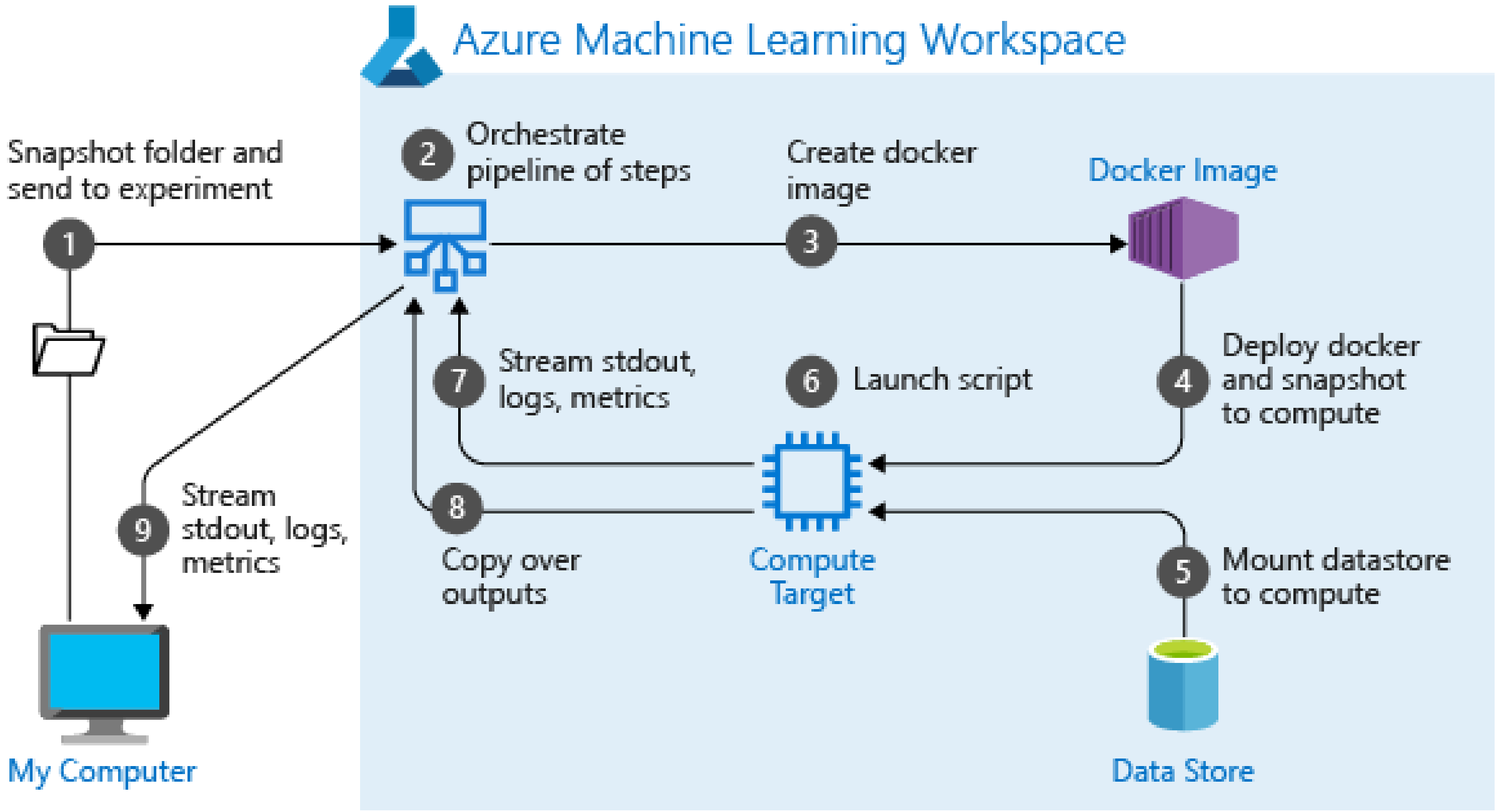
AML Pipeline phases



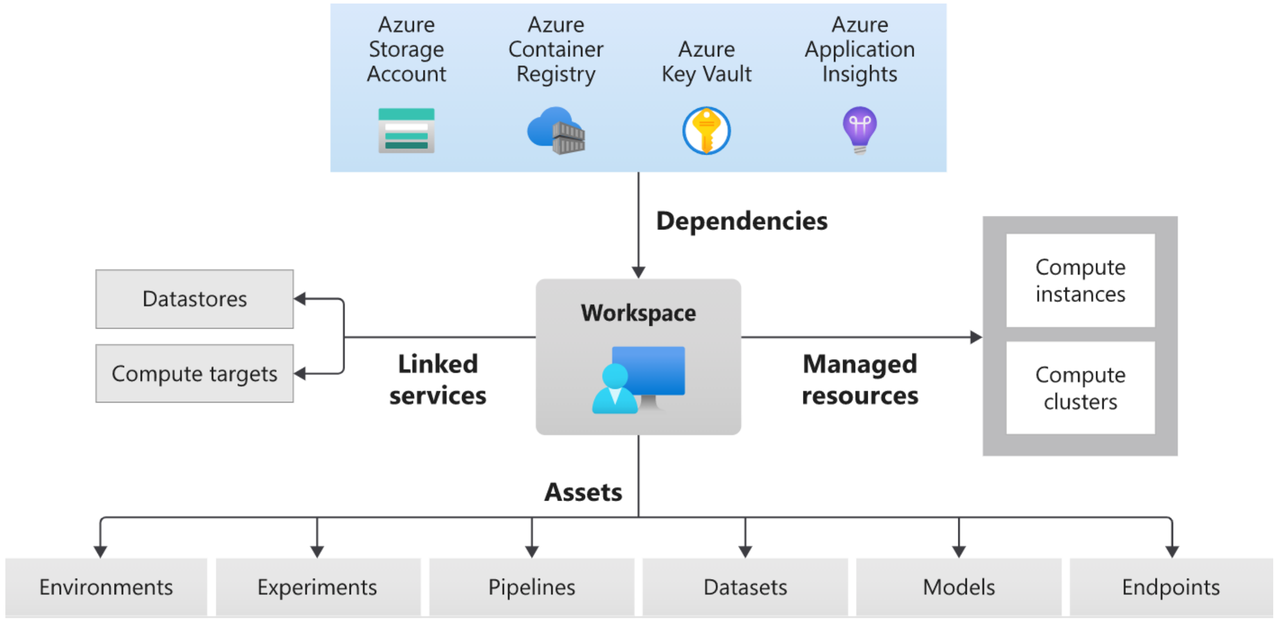
Build

Publish

Run



AML pipeline key elements



- **Workspace**: top-level resource for AML. Contains all the artifacts created when the user utilizes AML
- **Environment**: specify the Python packages, environment variables, and software settings (Python, Spark, Docker)
- **Compute targets**: machines or clusters that perform the computational steps in ML pipeline
- **Storage**: reference to datastore. Many service types (i.e., Azure blob container, Azure file share, ...)

AML pipeline phases

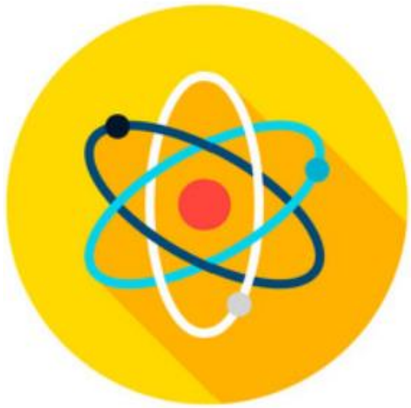
Building a pipeline

Build an AML pipeline implies to configure and set up the proper local environment. Therefore, you must:

- **Create and configure an Azure ML workspace** (WS)
- **Set up the compute target**: that means also to define the Python environment and package dependencies needed by your script.
- **Set up the machine learning resources**: where data is stored and how to access to it
- **Register a Model with the WS** to use it inside a step
- **Authenticate to Azure services** and call your environment for local developing
- **Define a pipeline** as sequence of `PipelineStep` objects and use `DataReference` to point the data and `PipelineData` to introduce a data dependency between steps and for creating an implicit execution order in the pipeline.

Publishing a pipeline

Publishing a pipeline at the end of the building script, allows you to **run the pipeline more times with different inputs**.



Building a pipeline

Construction and publishing of the pipeline

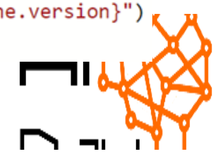
Authenticate to Azure services and call your environment

```
import sys; sys.path.append('.')
import requests
from azureml.core import Workspace, Datastore
from azureml.core.compute import AmlCompute, ComputeTarget
from azureml.core.runconfig import RunConfiguration, CondaDependencies
from azureml.exceptions import ComputeTargetException, RunConfigurationException
from azureml.pipeline.core import PublishedPipeline
from azureml.core.authentication import ServicePrincipalAuthentication
from ml_service.util.env_variables import Env

def get_aml_workspace():
    e = Env()
    svc_pr = ServicePrincipalAuthentication(
        tenant_id=e.tenant_id,
        service_principal_id=e.service_principal_id,
        service_principal_password=e.svc_pr_password
    )
    aml_workspace = Workspace.get(
        name=e.workspace_name,
        subscription_id=e.subscription_id,
        resource_group=e.resource_group_aml,
        auth=svc_pr
    )
    return aml_workspace
```

Define a pipeline as sequence of `PipelineStep` objects and use `DataReference` to point the data and `PipelineData` to introduce a data dependency between steps and for creating an implicit execution order in the pipeline.

Publishing a pipeline at the end of the building script, allows you to run the pipeline more times with different inputs.

```
from azureml.pipeline.core import Pipeline, PipelineData
from azureml.pipeline.core.graph import PipelineParameter
from azureml.pipeline.steps import PythonScriptStep


feature_step = PythonScriptStep(
        name="Featurize Data",
        script_name="features/extract_3dcnn_features.py",
        arguments=[
            "--frames_path", frames_path_param,
            "--segment_id", segment_id_param,
        ],
        inputs=[input_mount, frames_path_param],
        outputs=[cnn_features],
        compute_target=compute_gpu,
        allow_reuse=False,
        runconfig=run_config,
        source_directory=source_dir
    )

steps = [feature_step, score_step]

# Set up the pipeline
cnn_pipeline = Pipeline(workspace=aml_workspace, steps=steps)
cnn_pipeline.validate()

# Publish the pipeline
published_pipeline = cnn_pipeline.publish(
    name=pipeline_name,
    description="3DCNN ML Pipeline",
    version=pipeline_version
)

print(f"Published pipeline: {published_pipeline.name}, version: {published_pipeline.version}")
print(f"Endpoint: {published_pipeline.endpoint}")
```



AML pipeline phases

Running an Experiment

Run a pipeline means to **execute an Experiment**. Building and running a pipeline are defined in two different scripts. Therefore, to execute an experiment you must:

- Authenticate to Azure services
- Recall the environment: from where all environment's variables are token
- Get AML workspace: the same of the published pipeline
- Find the pipeline in the WS and get its endpoint

Then you can **run an experiment** using `PipelineRun()`.

Note. `Response = request.post()` reports all **the experiment parameters used**

Results

To assess the run execution, we used the REST API through `requests.post` object, part of requests module.

- If response = 200: everything is fine.
- If response = 401: authentication not found

Performance monitoring

When you run an experiment, logs and metrics are streamed for you.

- **Logs**: diagnose errors and warnings. Use `run.wait_for_completion(show_output = True)` to show when the model training is complete. The `show_output` flag gives you verbose output.
- **Performance metrics**: track and monitor your runs. You can view the metrics of a trained model using `run.get_metrics()` but you can also **add customized metrics** develop through python scripts.

Running a published pipeline

Run

Run a pipeline means to **execute an Experiment**. Building and running a pipeline are defined in two different scripts. Therefore, to execute an experiment you must:

- Authenticate to Azure services
- Recall the environment: from where all environment's variables are token
- Get AML workspace: the same of the published pipeline
- Find the pipeline in the WS and get its endpoint

Then you can **run an experiment** using `PipelineRun()`.

Note. `Response = request.post()` reports all **the experiment parameters used**

```
import requests
from azureml.pipeline.core import PipelineRun


pipeline = find_pipeline(aml_workspace, pipeline_name, pipeline_version)
rest_endpoint = pipeline.endpoint

response = requests.post(
     rest_endpoint,
     headers=auth_header,
     json={
          "ExperimentName": experiment_name,
          "RunSource": "API",
          "DataPathAssignments": {
               "frames_path": {"DataStoreName": "3dcnn_pipeline_blob", "RelativePath": frames_path}
            },
            "ParameterAssignments": {
                "pipeline_id": pipeline.id,
                "model_name": model_name,
                "model_version": model_version,
            }
     })


run_id = response.json().get('Id')
print(f'Submitted pipeline run: {run_id}')
pipeline_run = PipelineRun(aml_workspace.experiments[experiment_name], run_id)
pipeline_run.tag('pipeline_version', pipeline_version)
print(f'Pipeline run parameters: {pipeline_run.get_properties()}')
pipeline_run.wait_for_completion(show_output=True, raise_on_error=True)
```

Findings and best-practices

Project structure

Early stages of an ML project
Jupyter notebooks used for **explorations**

Developing the project
complexity increase. CI/CD, single py scripts for each step and phase of the workflow

Cookie cutter: a cross-platform logical, simple, but flexible project structure to carry out and share work

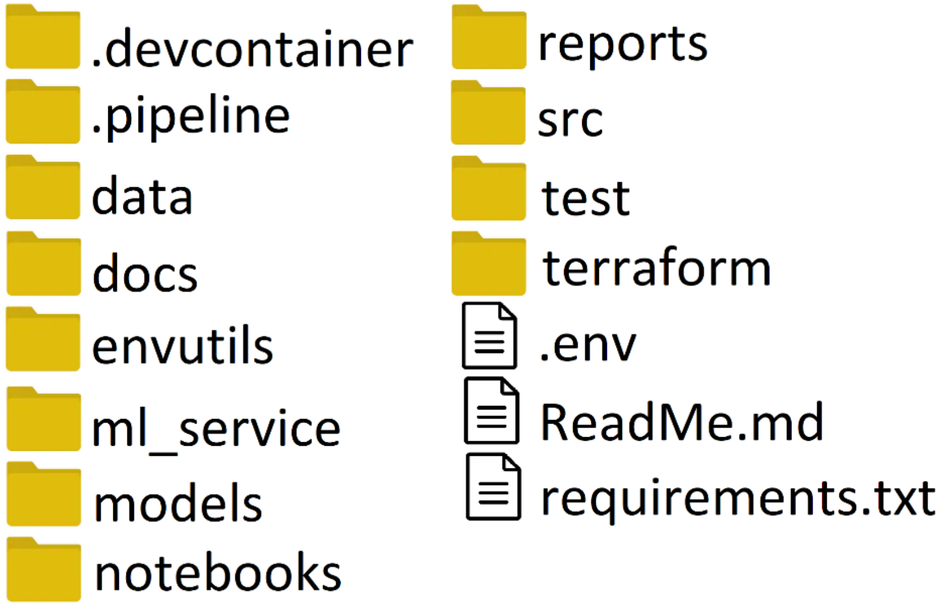
Documentation: strong, detailed and meaningful documentation for all the project phases

Python: cross-platform machine learning framework

Testing:

Unit and Integration tests are done using PyTest for Python.

When possible, acceptance criteria should be verified with acceptance tests

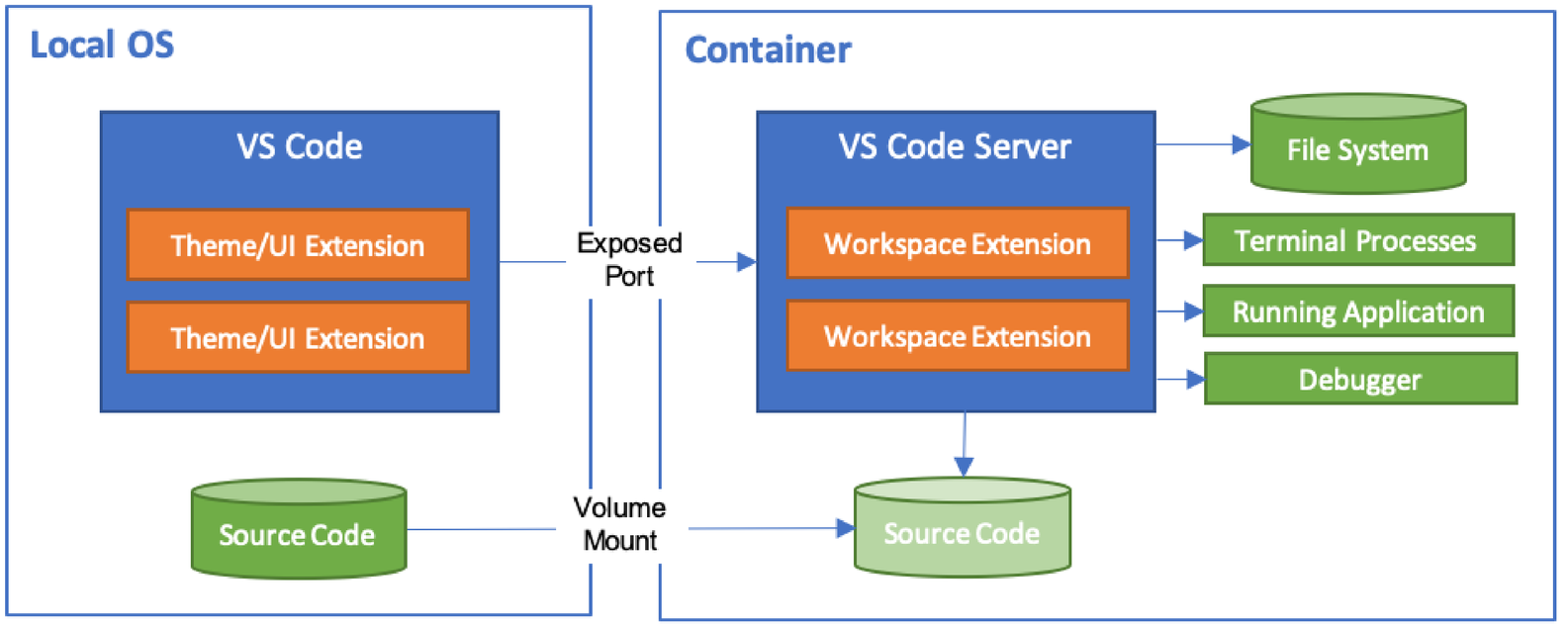


Dev Environment in Containers

Remote Development extension pack for **Visual Studio Code**

→ Use a **Docker container** as a **full-featured development environment**

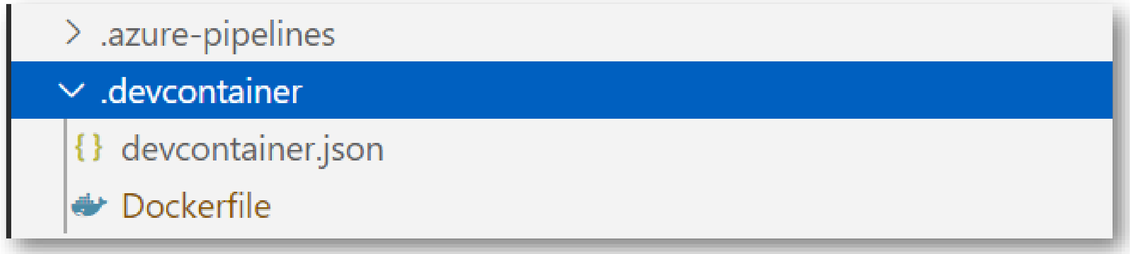
→ Use the **same Docker image** for all dev team (local machine, remote machine or cloud VMs)



https://code.visualstudio.com/docs/remote/containers
https://github.com/polatengin/project-standards/blob/master/DevContainers.md
https://channel9.msdn.com/Series/Beginners-Series-to-Dev-Containers

Dev Environment Organization

The **Dockerfile** contains **all the required tools and configuration** to work with all the frameworks and dev tools **the project** needs



Python (dev tools, linter, test framework, etc.)

Git

Terraform

Docker CE CLI / Docker Compose

Azure CLI

Azure Functions extension

When there are a lot of layers or when building layers requires time,

use a pre-built DevContainer image and store it in a Container Registry

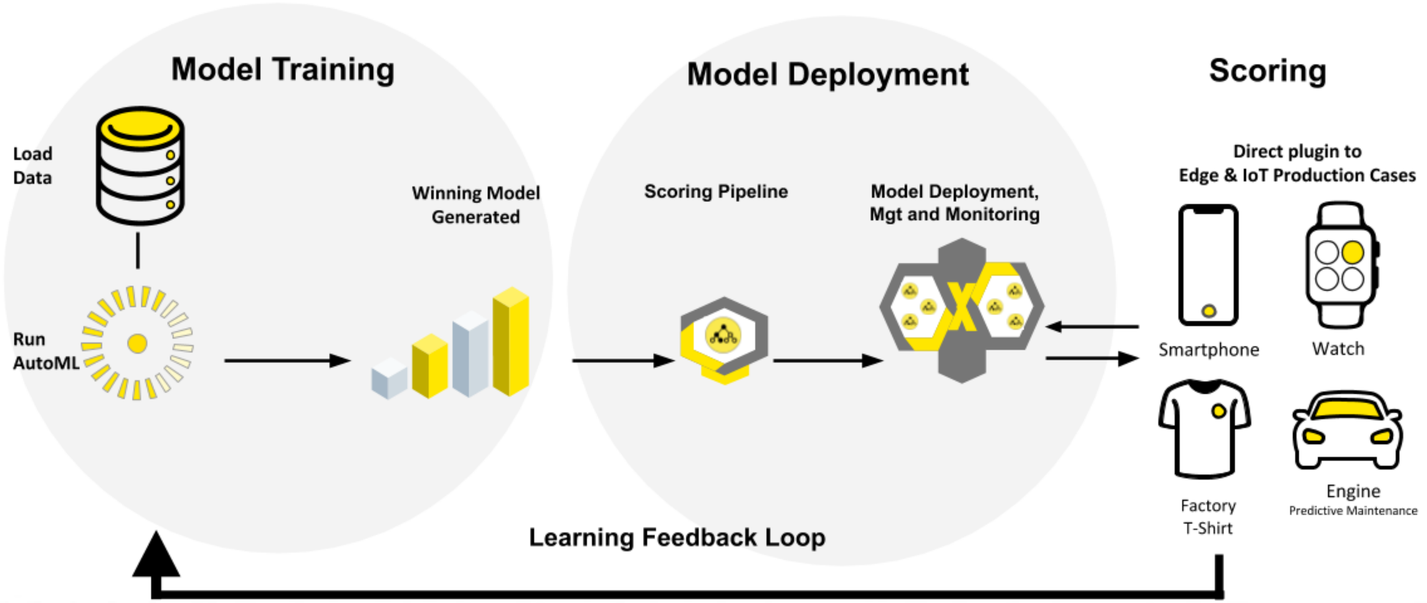
Example of Cognitive Service use

Training Pipeline

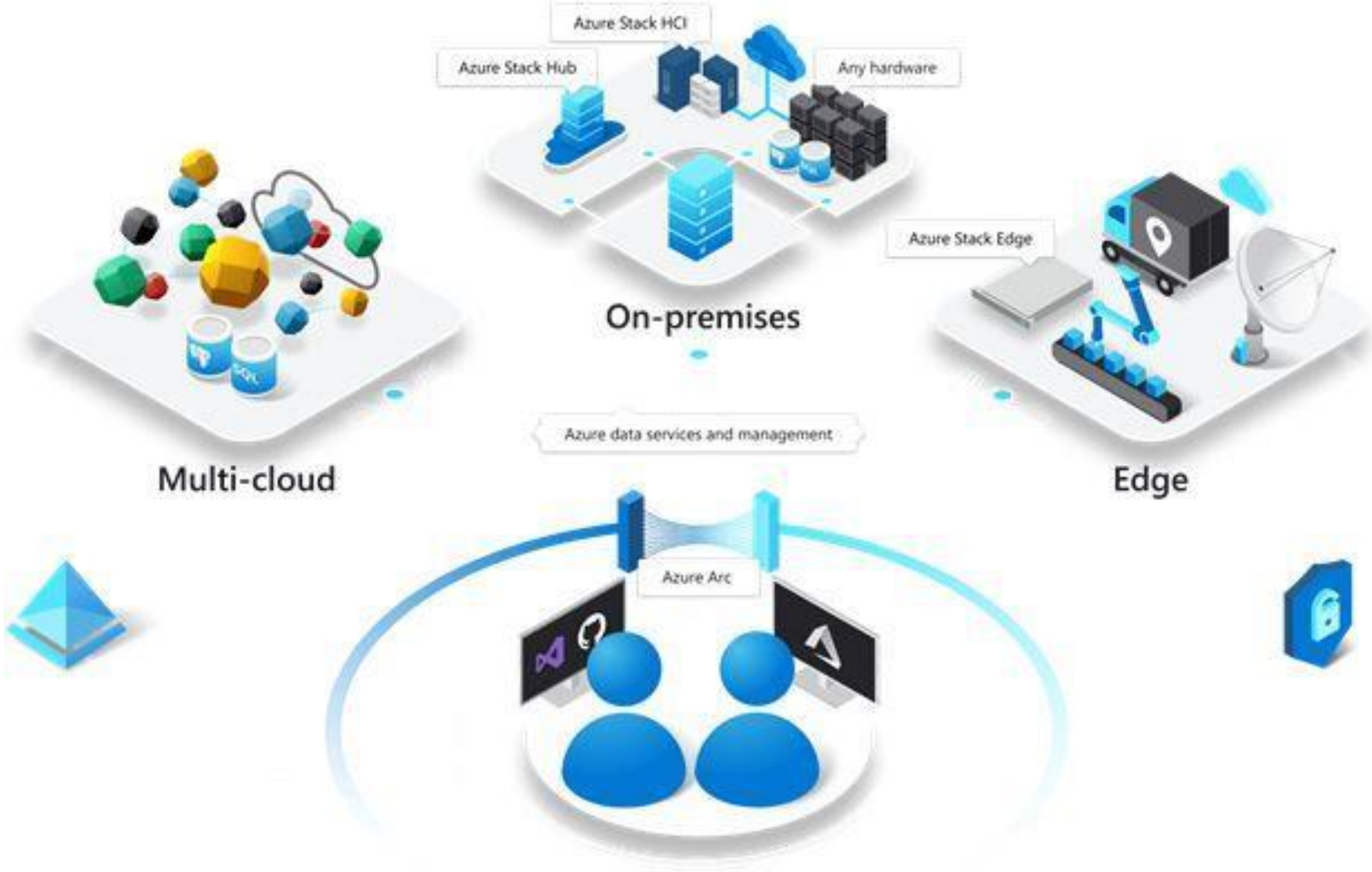
The Custom Vision Training Pipeline is responsible for training a new Custom Vision model, that is used for classification.

Scoring Pipeline

The Custom Vision Scoring Pipeline is responsible for executing the trained model, on a provided image at runtime.



Azure Arc enabled ML



Preview Access request form
https://aka.ms/amlk8s-preview-survey

Real World: in Production

Exploration

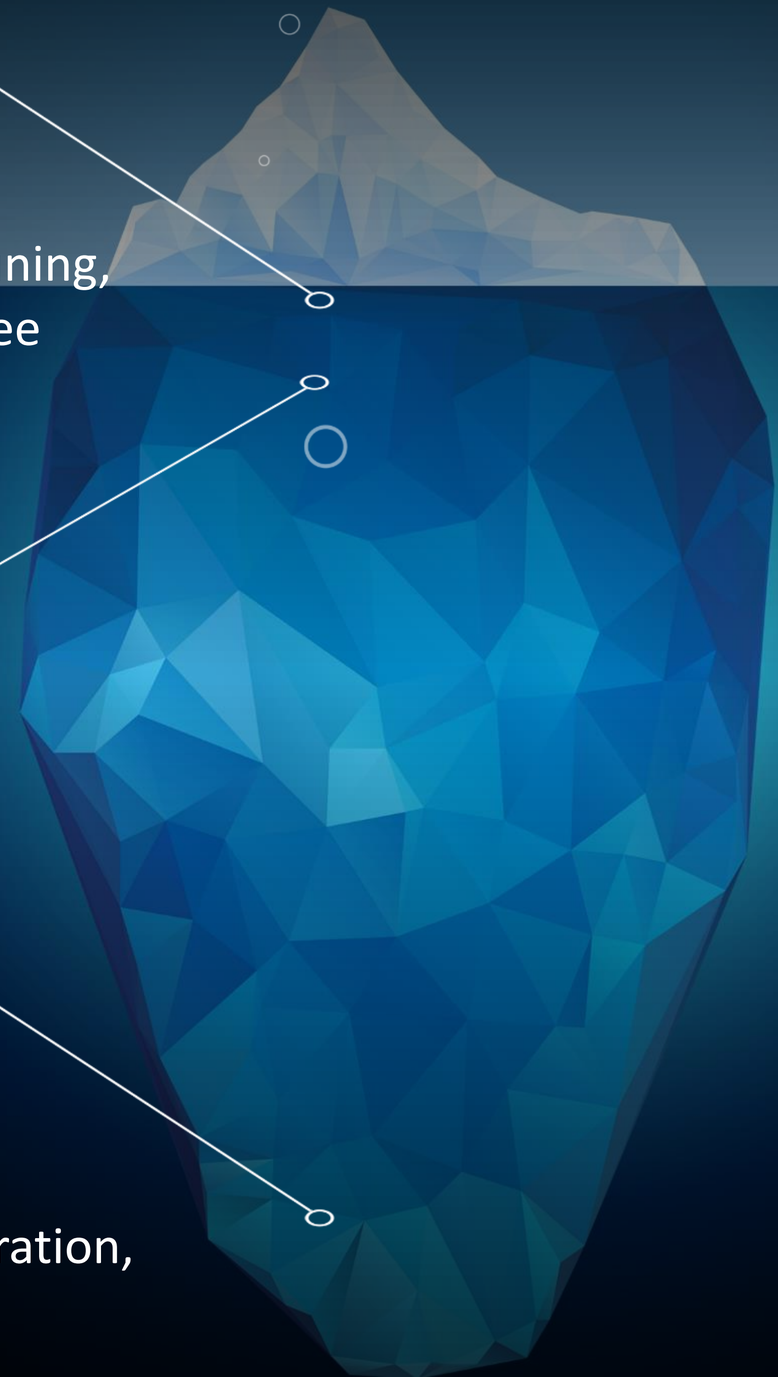
Data Preparation, Dataset Augmentation,
Jupyter Notebooks, Model Definition,
Toy Datasets, Hyperparameters, Model Tuning,
Output Metrics Evaluation, … a lot of coffee

Engineering & MLOps

Machine Learning Pipelines Definition,
Code Re-engineering, Automatic Testing,
CI/CD Pipelines, Performance Testing

Production Ready

Infrastructure as Code,
End to End Testing with real datasets,
Performance Tuning, Data Security,
Infrastructure Security, App/Service Integration,
Monitoring



Thank You!

ευχαριστώ Salamat Po متشكرم شكراً Grazie

благодаря ありがとうございます Kiitos Teşekkürler 谢谢

ขอบคุณครับ Obrigado شكريه Terima Kasih Dziękuję

Hvala Köszönöm Tak Dank u wel дякую Tack

Mulţumesc спасибо Danke Cám ơn Gracias

多謝晒 Ďakujem תודה நன்றி Děkuji 감사합니다

AI Day

Microsoft

avanade

REPLY CLUSTER

ellycode
WHAT DO YOU NEED?

LOBRA

OVH

NET CODE

JET BRAINS

Packt›

About us

Ing. Gianni ROSA GALLINA

R&D Senior Software Engineer @ Deltatre

@giannirg

- AI, Machine Learning, Deep Learning on multimedia content
- Virtual/Augmented/Mixed Reality
- Immersive video streaming & 3D graphics for sport events
- Cloud solutions, web backends, serverless, video workflows
- Mobile apps dev (Windows / Android / Xamarin)
- End-to-end solutions with Microsoft Azure

MVP Microsoft Most Valuable Professional

Microsoft Specialist
Programming in C#
Programming in HTML5 with JavaScript & CSS3

Microsoft CERTIFIED Solutions Developer
Windows Store Apps Using C#
Web Applications

Microsoft CERTIFIED AZURE AI ENGINEER ASSOCIATE

Microsoft CERTIFIED AZURE DATA SCIENTIST ASSOCIATE

unity Certified Programmer

PLURALSIGHT Author

https://gianni.rosagallina.com

About us

Vito Flavio Lorusso

Program Manager @ **Microsoft**

@vflorusso

- Former web developer
- Former data engineer and developer
- Former “doing Database cluster installations in datacenters”
- Former Solutions Architect
- Former “cloud evangelist”
- Former Distributed systems engineer
- Constantly looking for my place in the digital world to help work get done

About us

Clemente Giorio

R&D Senior Software Engineer @ Deltatre

@tinux80

- Augmented/Mixed/Virtual Reality
- Artificial Intelligence, Machine Learning, Deep Learning
- Internet of Things
- Embedded Apps
- Multimodal Tracking

dotNET{podcast}

MVP Microsoft Most Valuable Professional

intel Software INNOVATOR

[PACKT] PUBLISHING Author

FAB LAB NAPOLI

References

References (1/2)

Tools and IDE

https://www.python.org/
https://jupyter.org/
https://code.visualstudio.com/

ML/Deep Learning Services & Frameworks

https://azure.microsoft.com/services/machine-learning/
https://azure.microsoft.com/services/cognitive-services/
https://azure.microsoft.com/services/azure-arc/
https://www.tensorflow.org/
https://keras.io/
https://pytorch.org/
https://fast.ai/

References (2/2)

MLOps

https://mlflow.org/
https://www.kubeflow.org
https://dvc.org/
https://www.pachyderm.com/
https://github.com/microsoft/MLOps
https://docs.microsoft.com/azure/machine-learning/team-data-science-process
https://docs.microsoft.com/azure/architecture/reference-architectures/ai/mlops-python
https://docs.microsoft.com/azure/architecture/reference-architectures/#ai-and-machine-learning

VSCode DevContainers

https://github.com/polatengin/project-standards/blob/master/DevContainers.md
https://code.visualstudio.com/docs/remote/containers
https://channel9.msdn.com/Series/Beginners-Series-to-Dev-Containers

AI Day

Microsoft

avanade

REPLY CLUSTER

ellycode
WHAT DO YOU NEED?

LOBRA

OVH

NET CODE

JET BRAINS

Packt›