Microsoft Azure

Project Batcomputer

Making DevOps work for Machine Learning

batcomputer.benco.io

Ben Coleman **@BenCodeGeek**
Phil Harvey **@CodeBeard**

v4

Background

Motivation

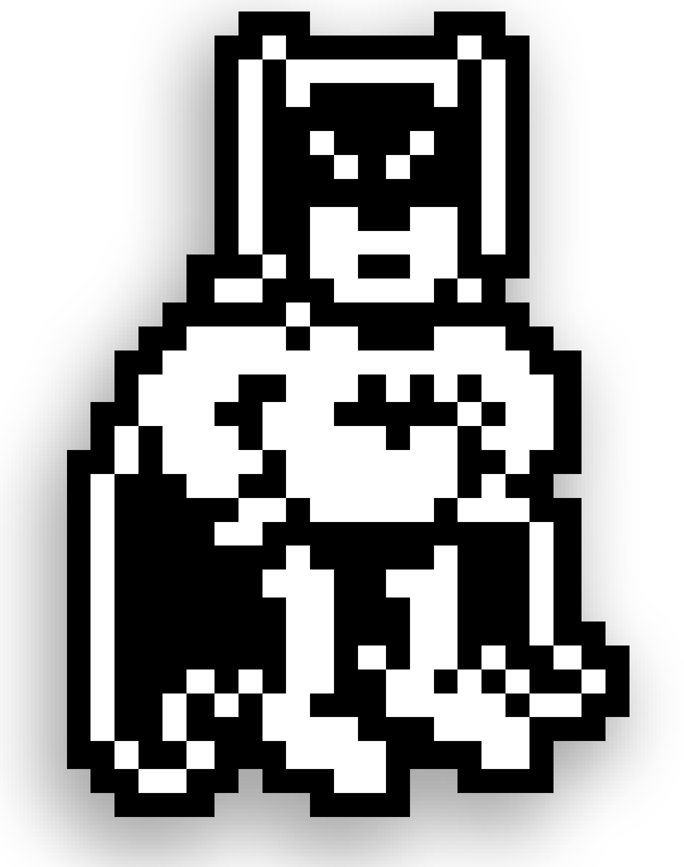
- Understand challenges in operationalisation of ML models
- “DevOps for AI”
- Integration of “all in in one” processes with real DevOps approach

Why Batcomputer?

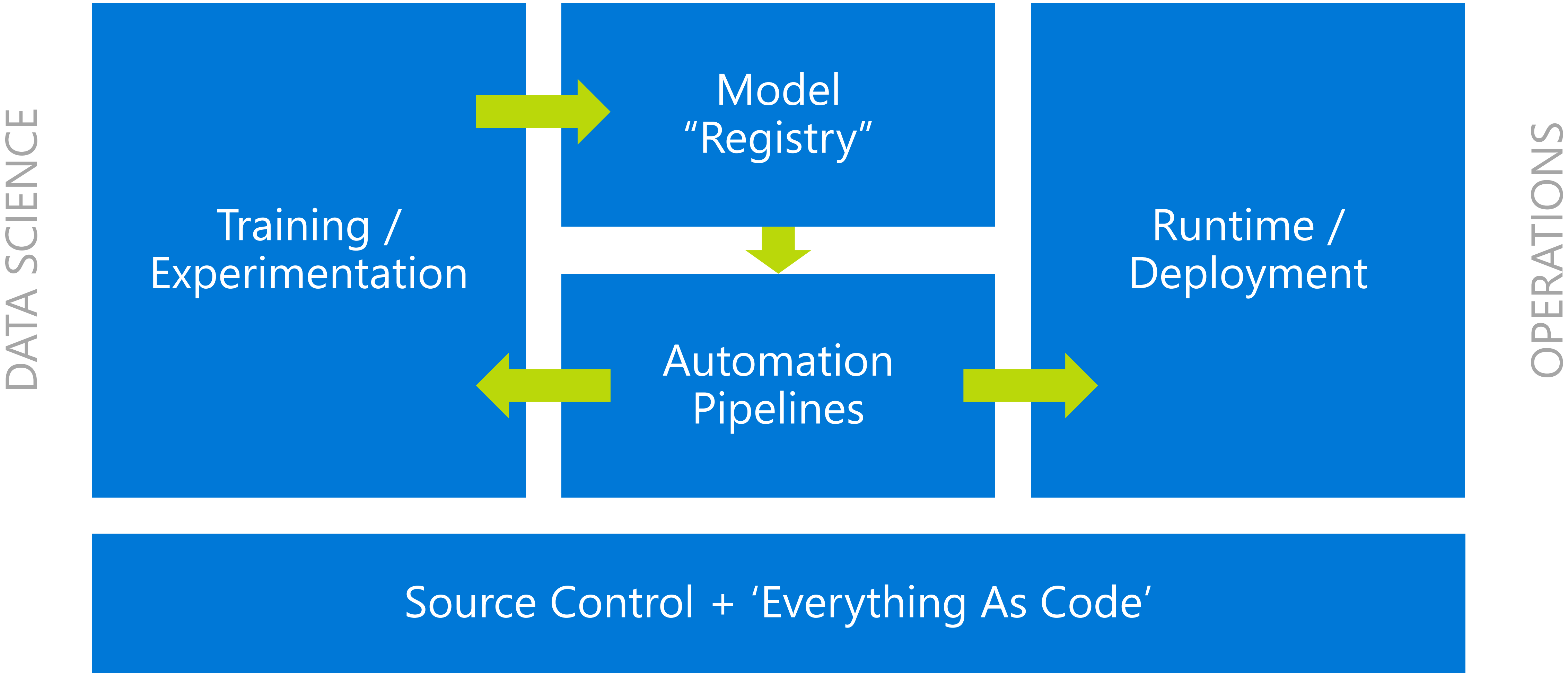
- Police recorded crime and outcomes data
- Source data as CSV - https://data.police.uk/data
- Build model of a given crime and region to predict – “Would you get caught?”

Core Principals & Benefits

- **Decouple** model training experiments from operations/runtime
- **Continuous Integration**
  Automated training, API builds & deployment
- **Versioned** models and APIs
- A real **RESTful API**, not a thin HTTP wrapper
- Configuration as code, infrastructure as code
- **Traceability**



Conceptual Building Blocks



DATA SCIENCE

Training / Experimentation

Model "Registry"

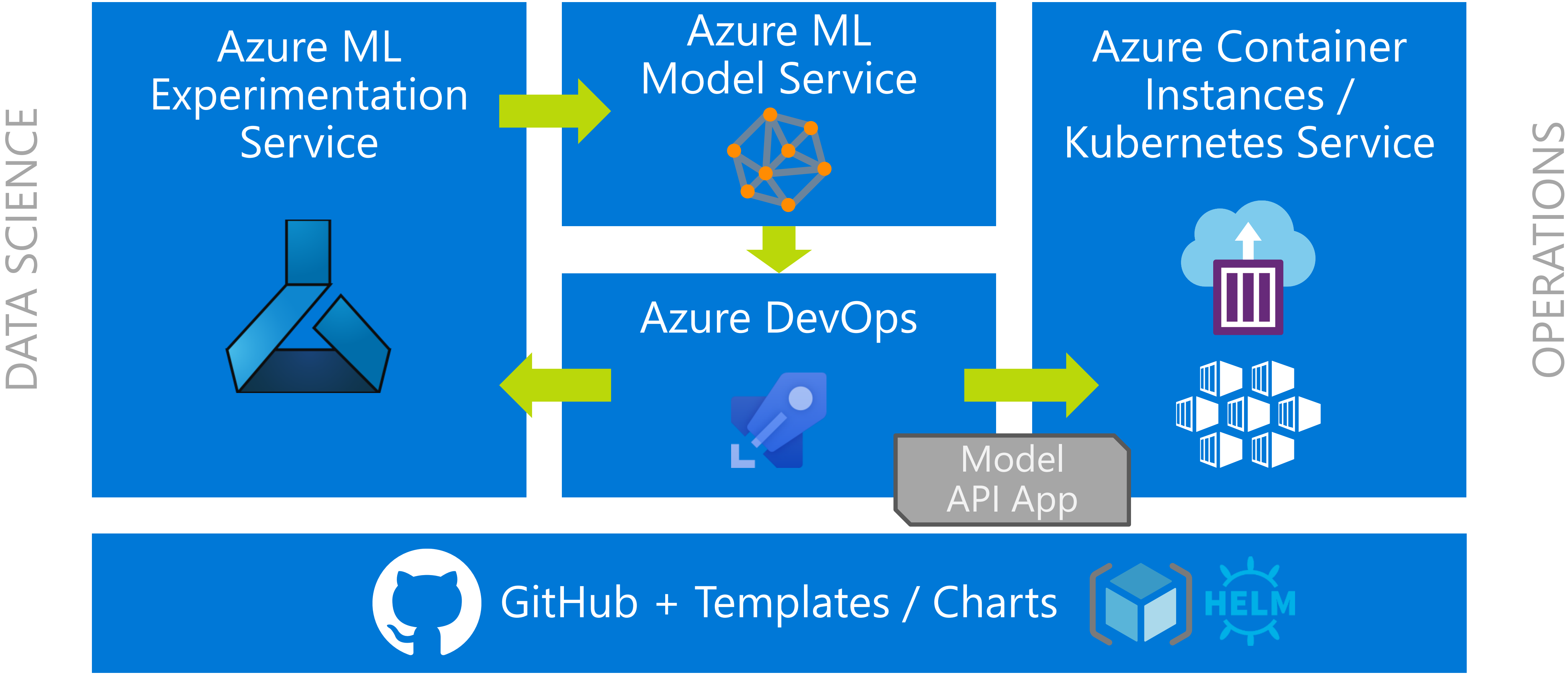
Automation Pipelines

Runtime / Deployment

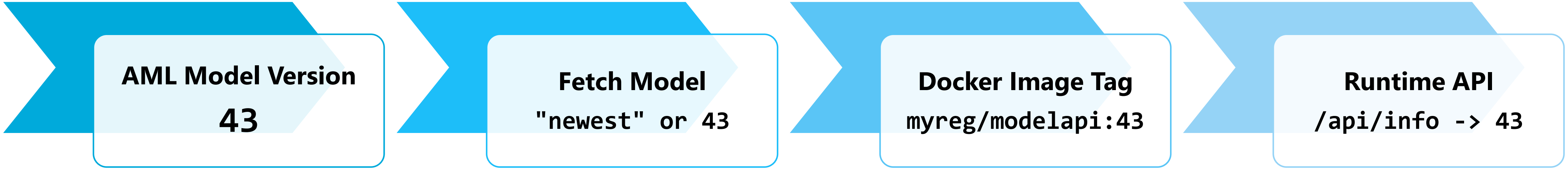
OPERATIONS

Source Control + 'Everything As Code'

Conceptual Building Blocks – Project Batcomputer


DATA SCIENCE
Azure ML Experimentation Service
Azure ML Model Service
Azure DevOps
Azure Container Instances / Kubernetes Service
Model API App
GitHub + Templates / Charts
HELM
OPERATIONS

Versioning – Many Touch Points

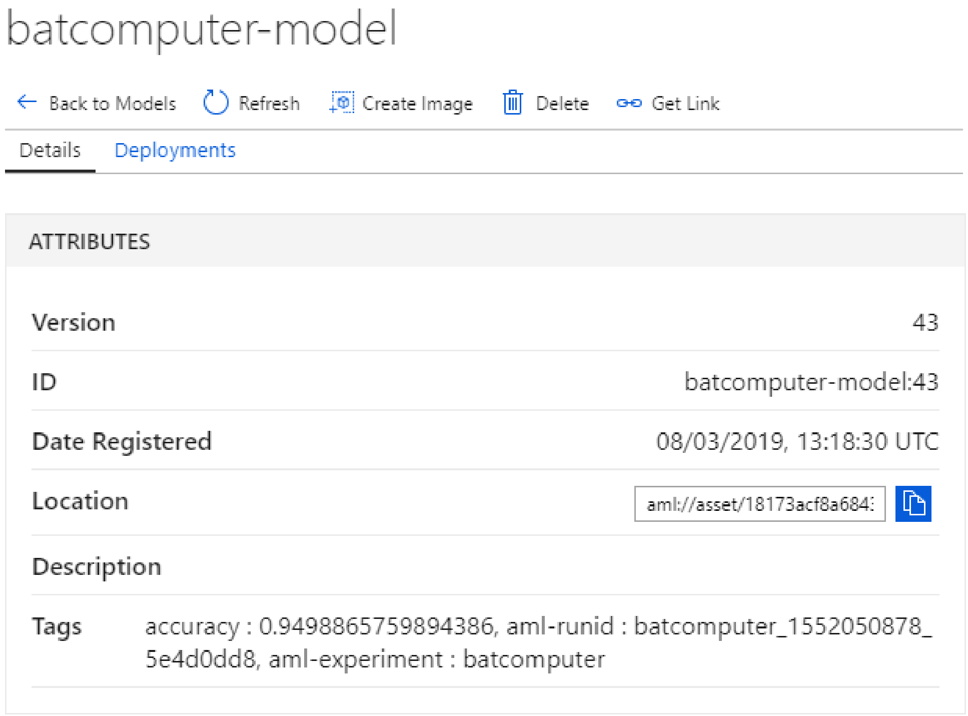


AML Model Version
43

Fetch Model
`"newest" or 43`

Docker Image Tag
`myreg/modelapi:43`

Runtime API
`/api/info -> 43`



batcomputer-model

ATTRIBUTES

Version	43
ID	batcomputer-model:43
Date Registered	08/03/2019, 13:18:30 UTC
Location	aml://asset/18173acf8a684
Description	
Tags	accuracy : 0.9498865759894386, aml-runid : batcomputer_1552050878_5e4d0dd8, aml-experiment : batcomputer

Also…

- Resource names in Azure controlled via ARM templates
- ACI DNS names & prefixes, e.g. `batcomputer-43.westeurope.azurecontainer.io`
- Object names in Kubernetes (pods, services), controlled via Helm chart

The ‘Model Registry’ – Not Just The Model

scikit learn

```
{
"gender": {
  "male": 0,
  "female": 1
}
```

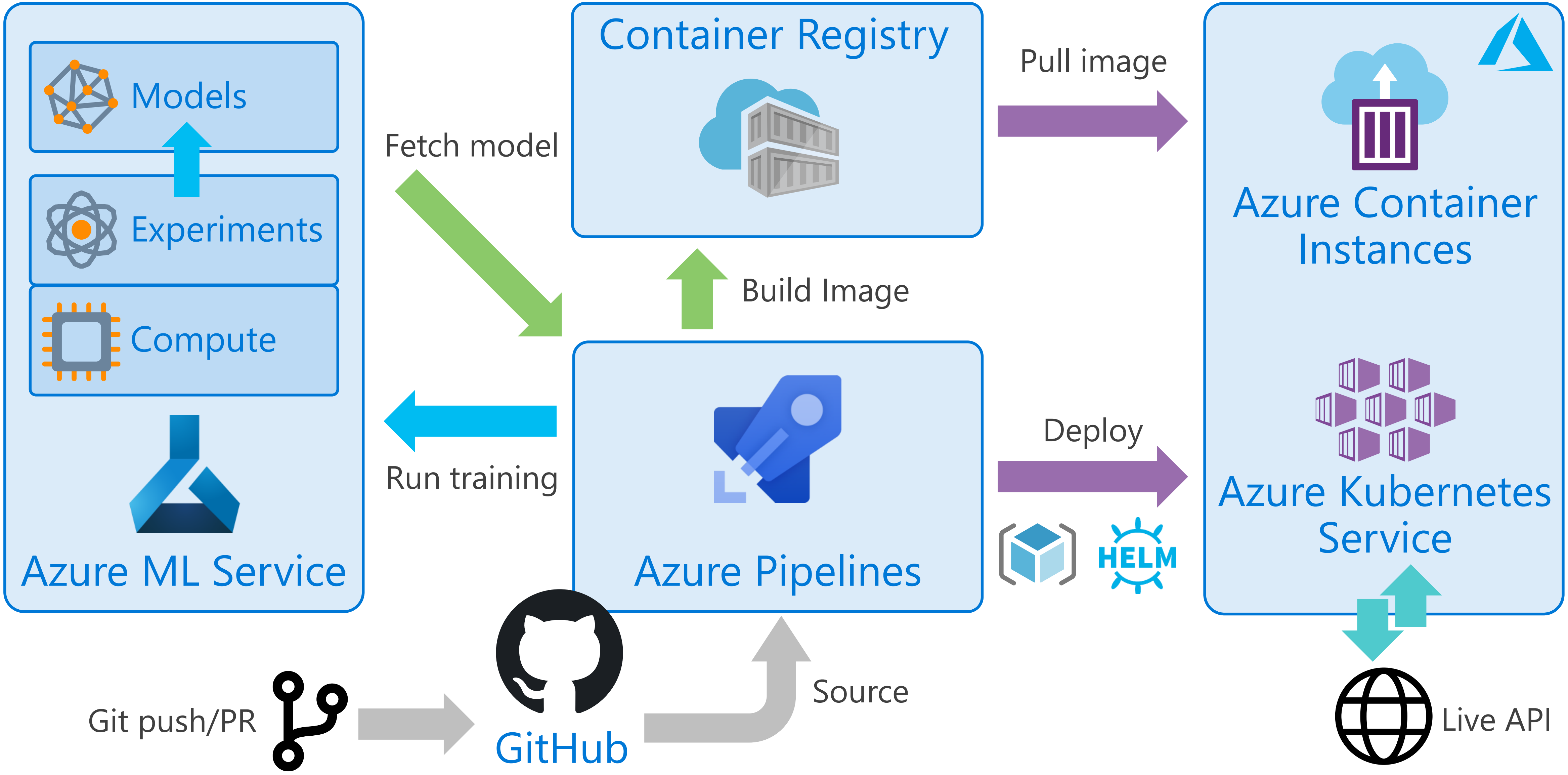
```
[
  "conviction",
  "dropped",
]
```

model.pkl	lookup.pkl	flags.pkl
Scikit-learn model/classifier	Python dictionary of dictionaries	Python array
Standard object rehydration, version sensitive	Mapping parameters/strings to num for predict function	Maps output of prediction function to human readable labels

DevOps

Continuous Integration / Continuous Delivery

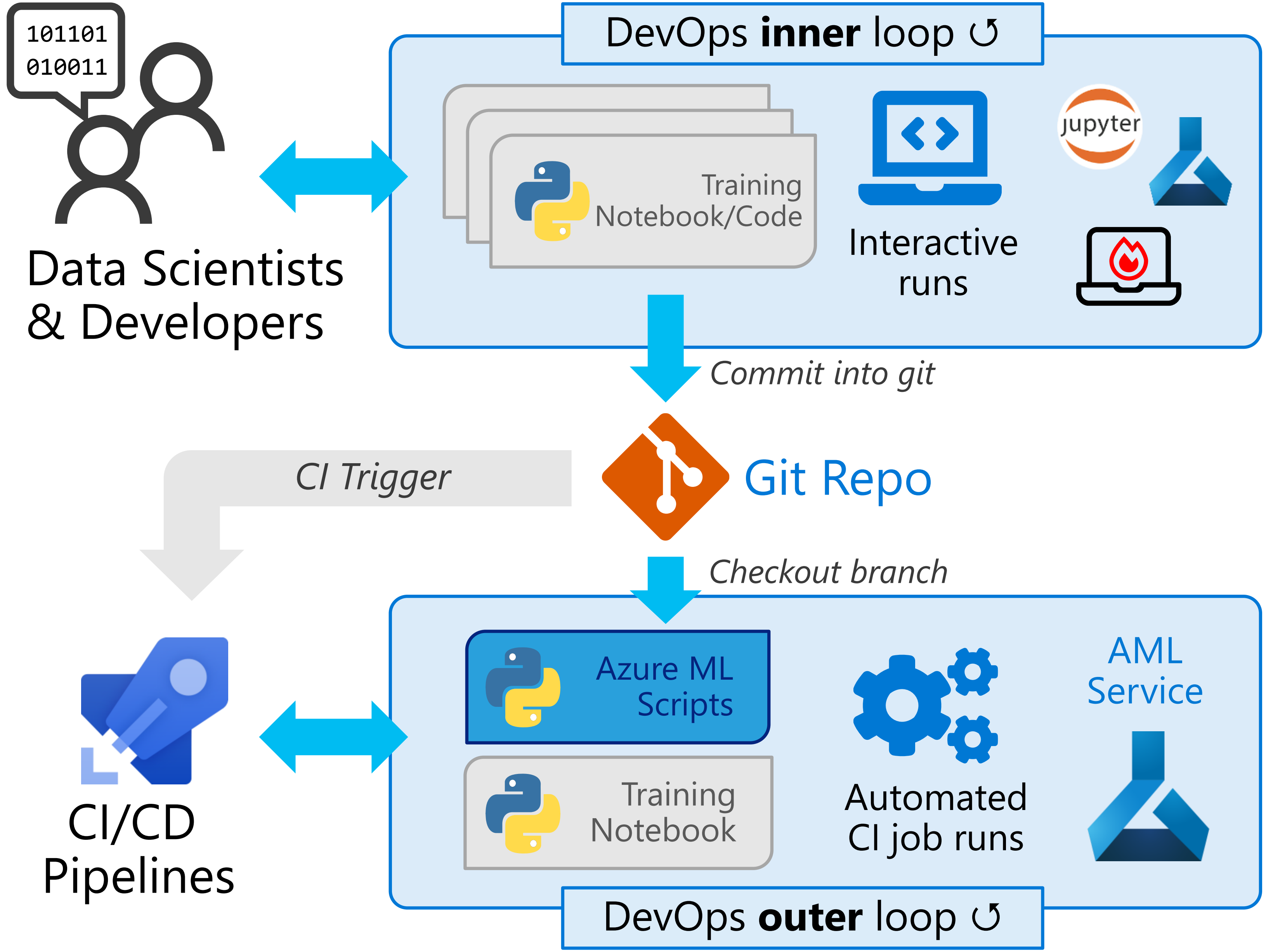
Model Training & Deployment – End To End Flow



Core DevOps Practice - Continuous Integration

Development & experimentation

CI triggered training & testing job runs

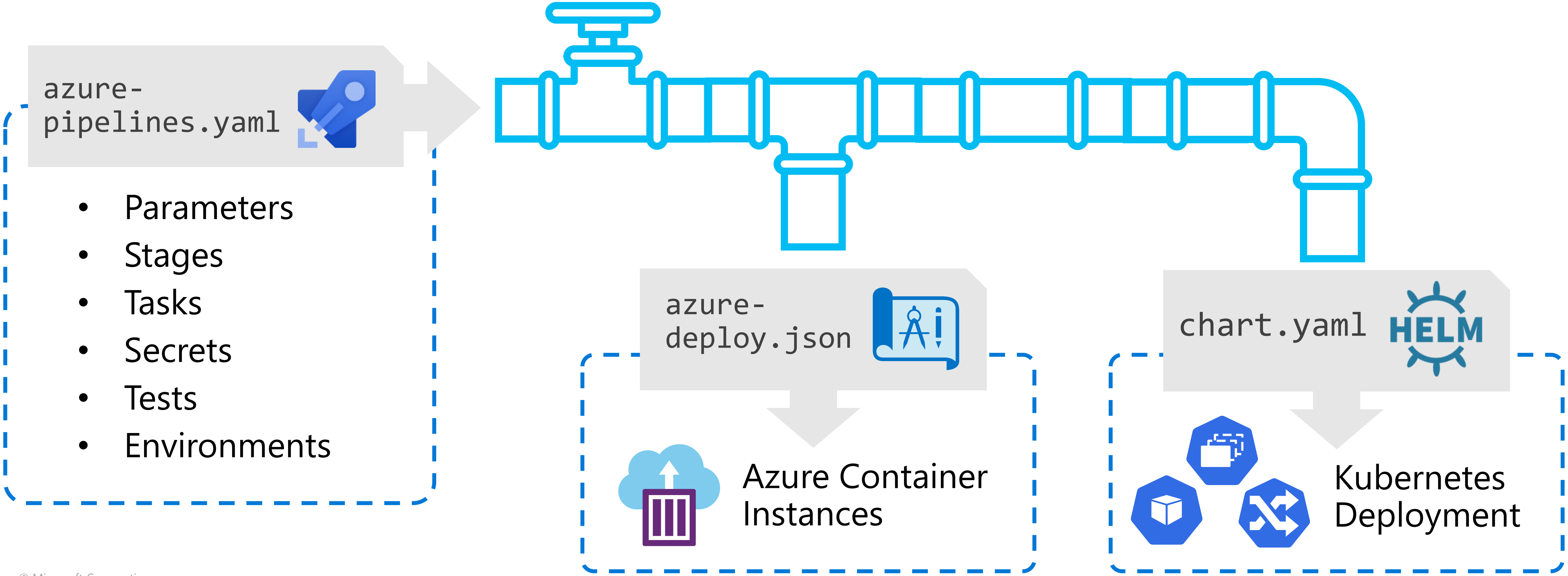


Infrastructure As Code

Standard DevOps working practice

Define everything about your environment as “code” (YAML, JSON, etc)

Store with your application under source control / Git



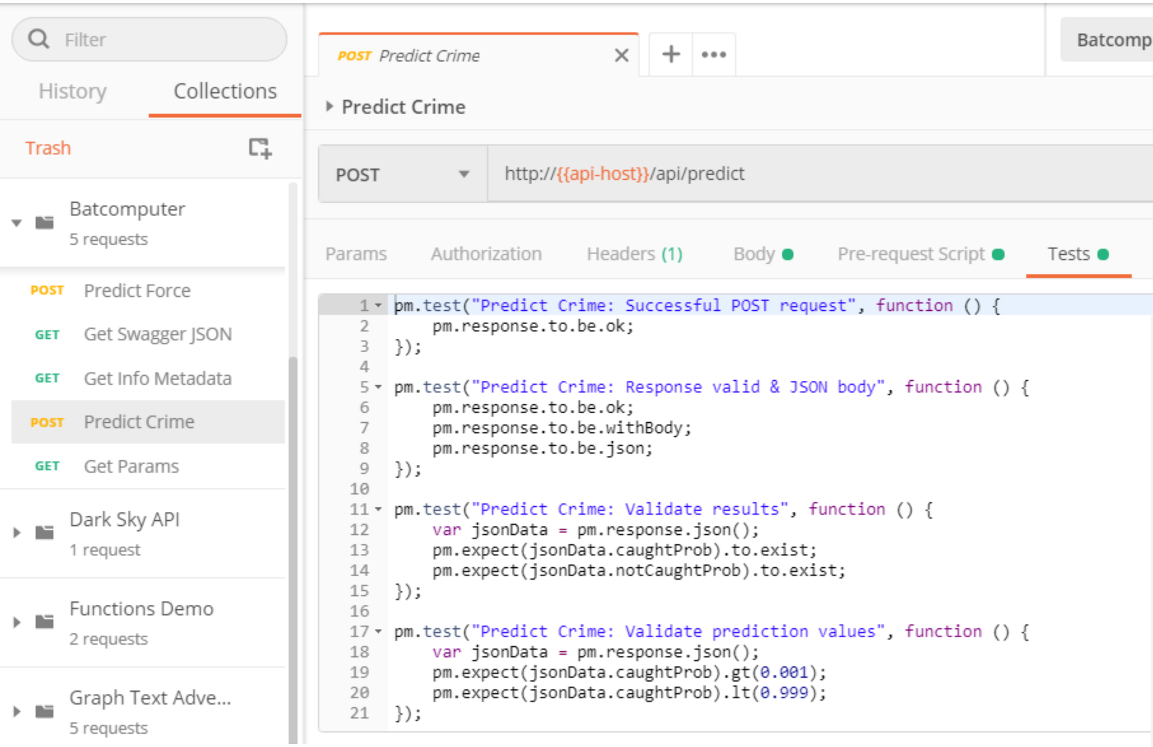
- Parameters
- Stages
- Tasks
- Secrets
- Tests
- Environments

Azure Container Instances

Kubernetes Deployment

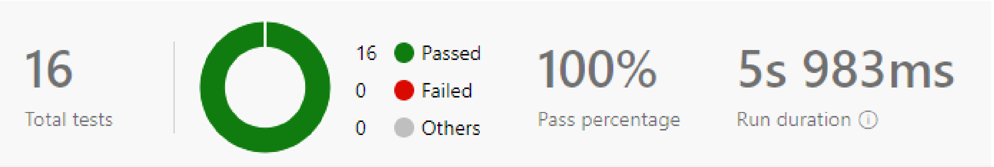
Testing

Integration tests against the real API using Postman & Newman



POST http://{{api-host}}/api/predict

```
pm.test("Predict Crime: Successful POST request", function () {
    pm.response.to.be.ok;
});

pm.test("Predict Crime: Response valid & JSON body", function () {
    pm.response.to.be.ok;
    pm.response.to.be.withBody;
    pm.response.to.be.json;
});

pm.test("Predict Crime: Validate results", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.caughtProb).to.exist;
    pm.expect(jsonData.notCaughtProb).to.exist;
});

pm.test("Predict Crime: Validate prediction values", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.caughtProb).gt(0.001);
    pm.expect(jsonData.caughtProb).lt(0.999);
});
```

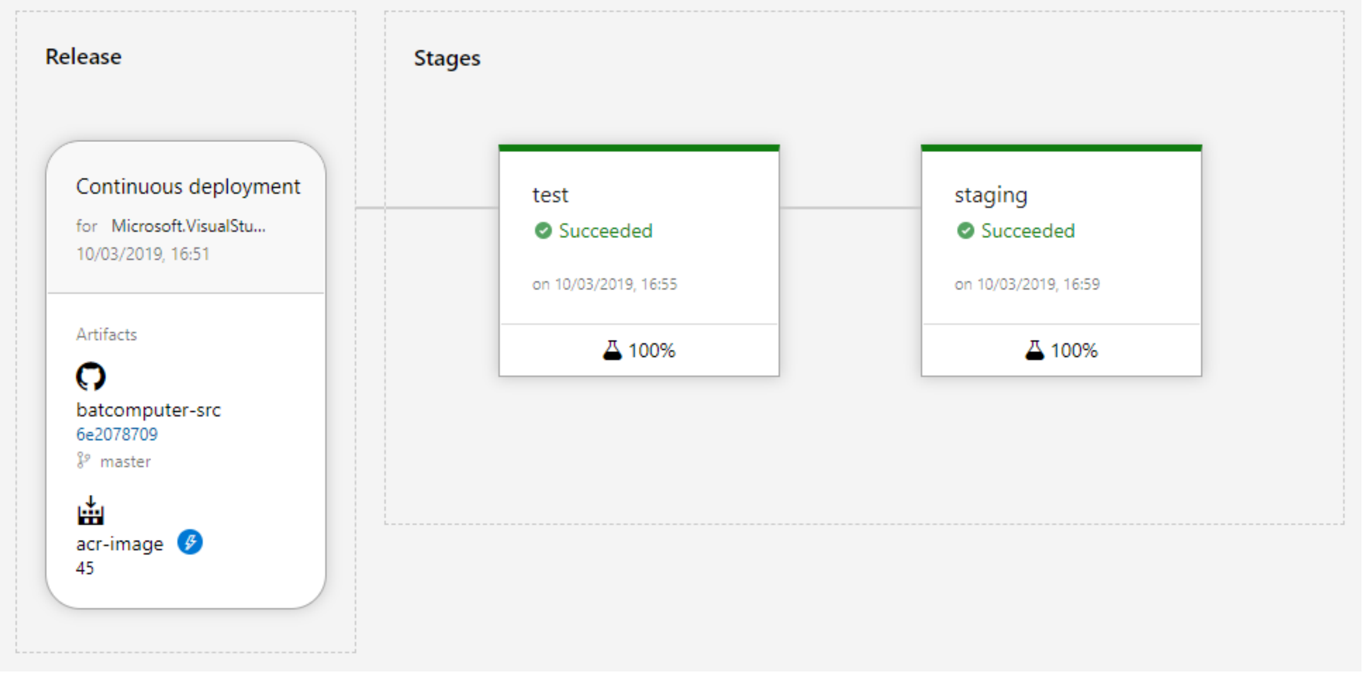


16 Total tests

16	● Passed
0	● Failed
0	● Others

100% Pass percentage

5s 983ms Run duration



Release

Continuous deployment
for Microsoft.VisualStu...
10/03/2019, 16:51

Artifacts
batcomputer-src
6e2078709
master
acr-image
45

Stages

test
Succeeded
on 10/03/2019, 16:55
100%

staging
Succeeded
on 10/03/2019, 16:59
100%

- Newman is a command-line collection runner for Postman
- It allows running a test suite using Postman collection

Machine Learning & Training

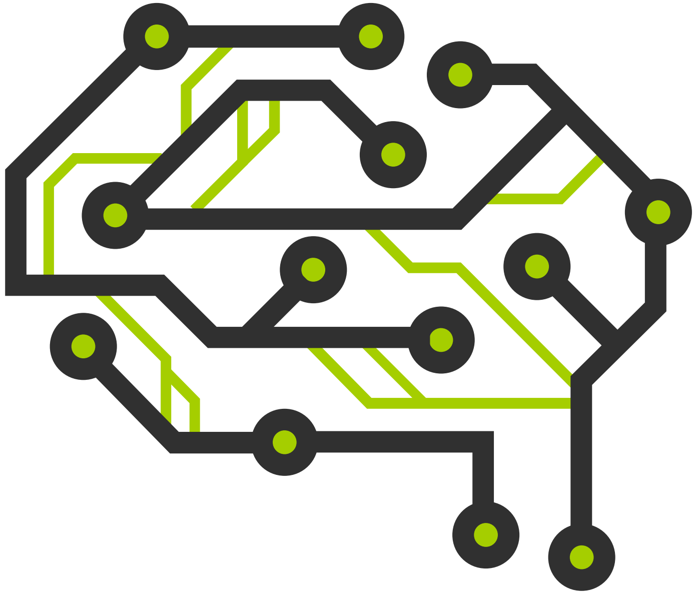
Machine Learning – Training Scripts

The focus of Batcomputer project is not best practice machine learning or rigorous data science

Well known libraries: Scikit Learn + Pandas

Build a simple classification model using labelled data (supervised learning)

Small-ish data set (1.5GB)



Azure Machine Learning Service - AML

Azure Machine Learning service provides SDKs and services to prep data, train, and deploy machine learning models

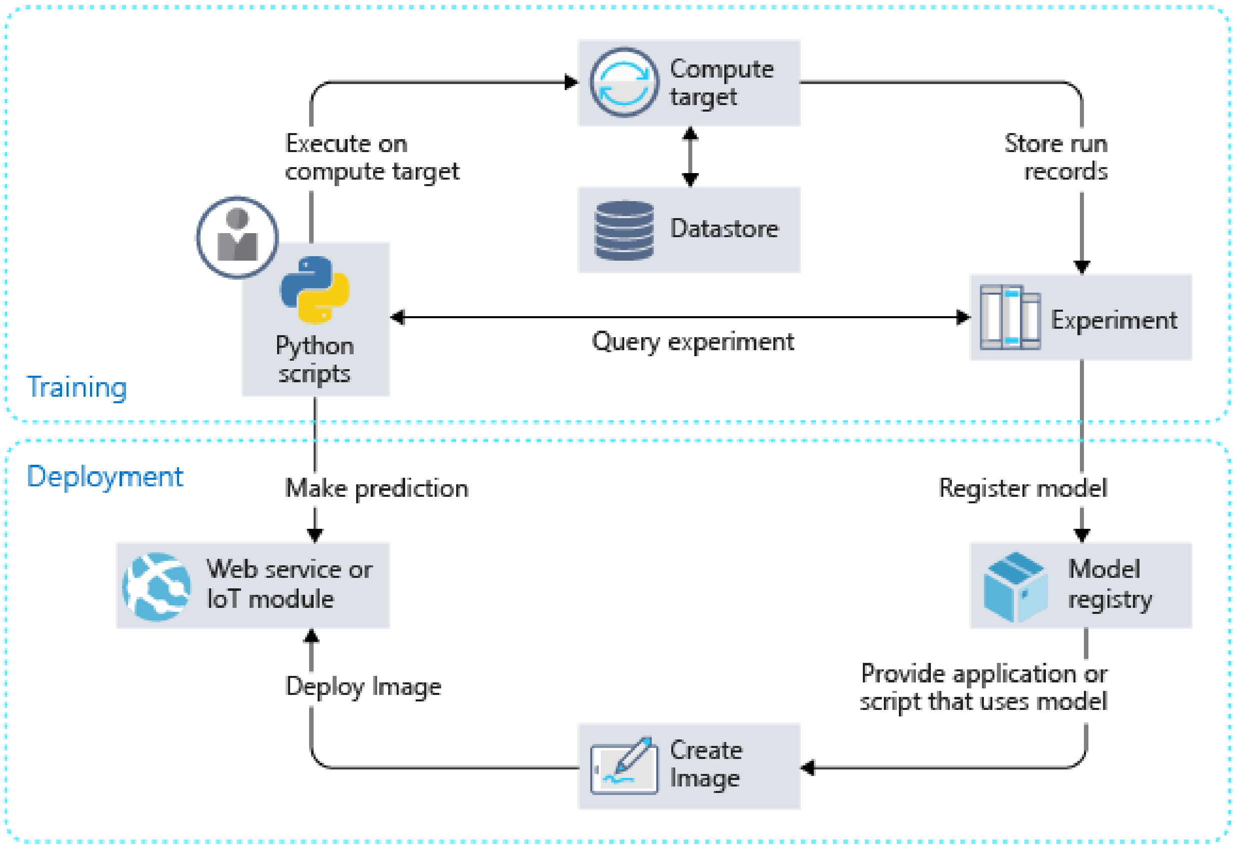
Driven by Python SDK

Range of training & experimentation compute targets

Model management

D on
i

“Project Batcomputer Operationalisation Process”



Pipelines

Compute

Experiments

Models

Images

Deployment

Azure ML Orchestration Scripts

https://docs.microsoft.com/python/api/overview/azure/ml/intro

`upload-data.py`

- Prepares environment
- **Uploads** local training data to Azure ML **datastore**

`run-training.py`

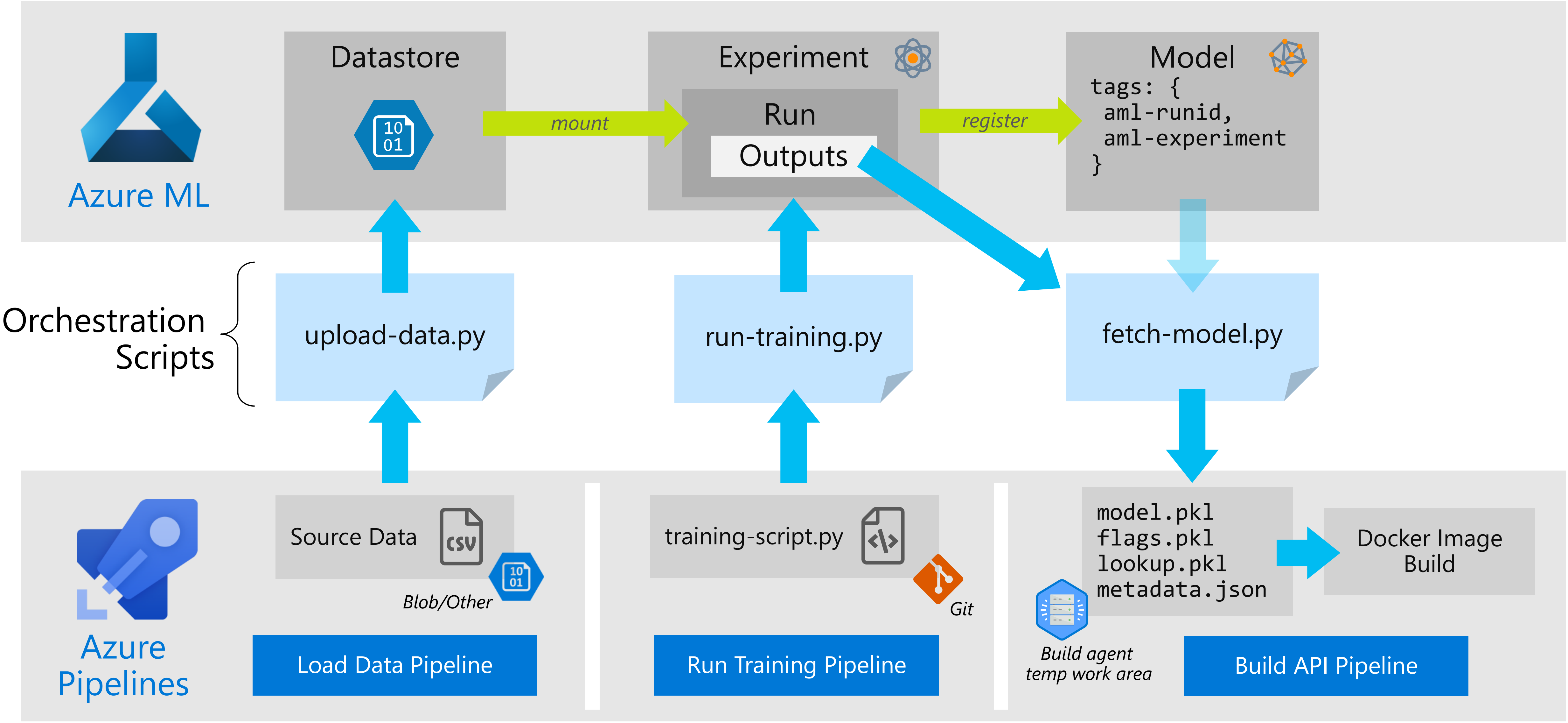
- Instructs Azure ML run an **experiment**
- Source training script is **separate** python file
- Training python is executed **remotely** in Azure ML **compute cluster**
- **Registers** resulting model in Azure ML **model service**

`fetch-model.py`

- **Downloads** serialised model from Azure ML **model service**
- In addition gets **supporting .pkl files** (more later)

Azure Machine Learning SDK for Python

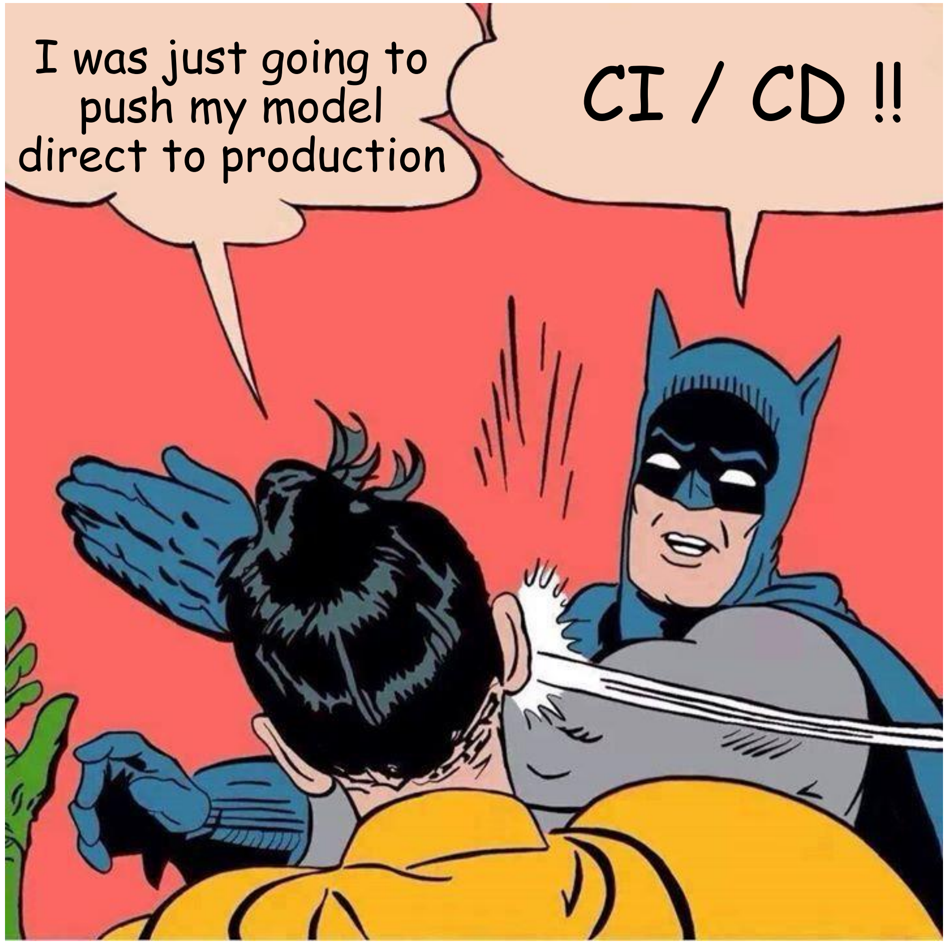
Detailed Azure DevOps Flow



Azure ML Deployment

Azure ML provides a means to deploy your models, why not use it?

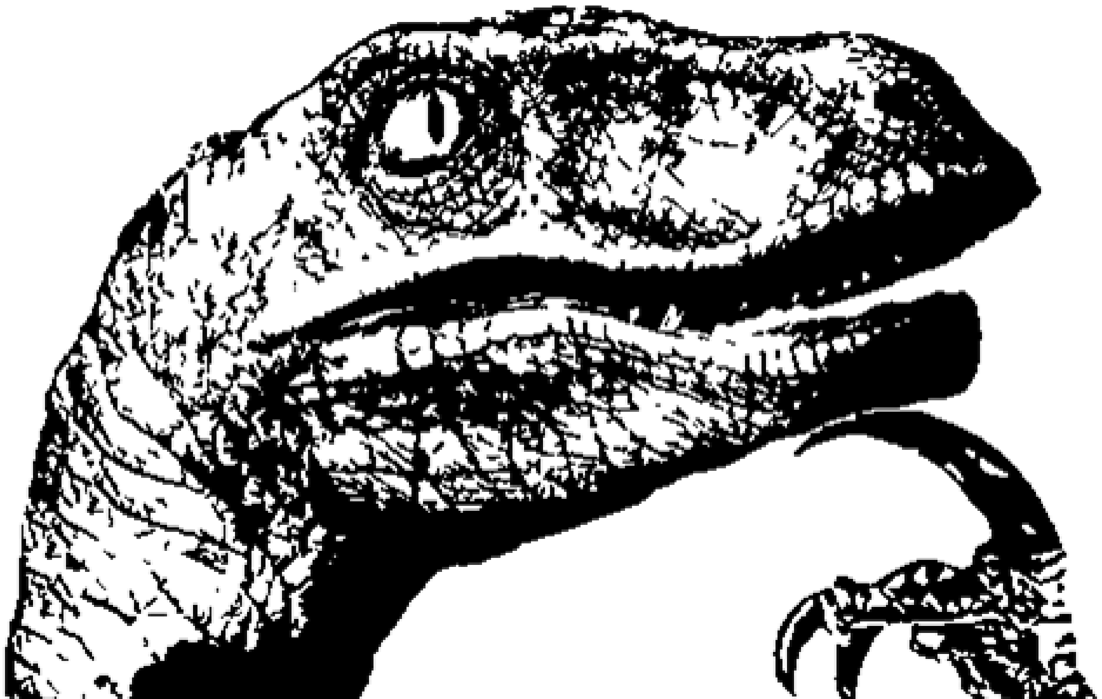
- “Highly Opinionated”
- Bypasses release process
- No control over container build process
- No choice of app structure, code or framework
- No infrastructure as code or release pipeline
- No integration or regression testing



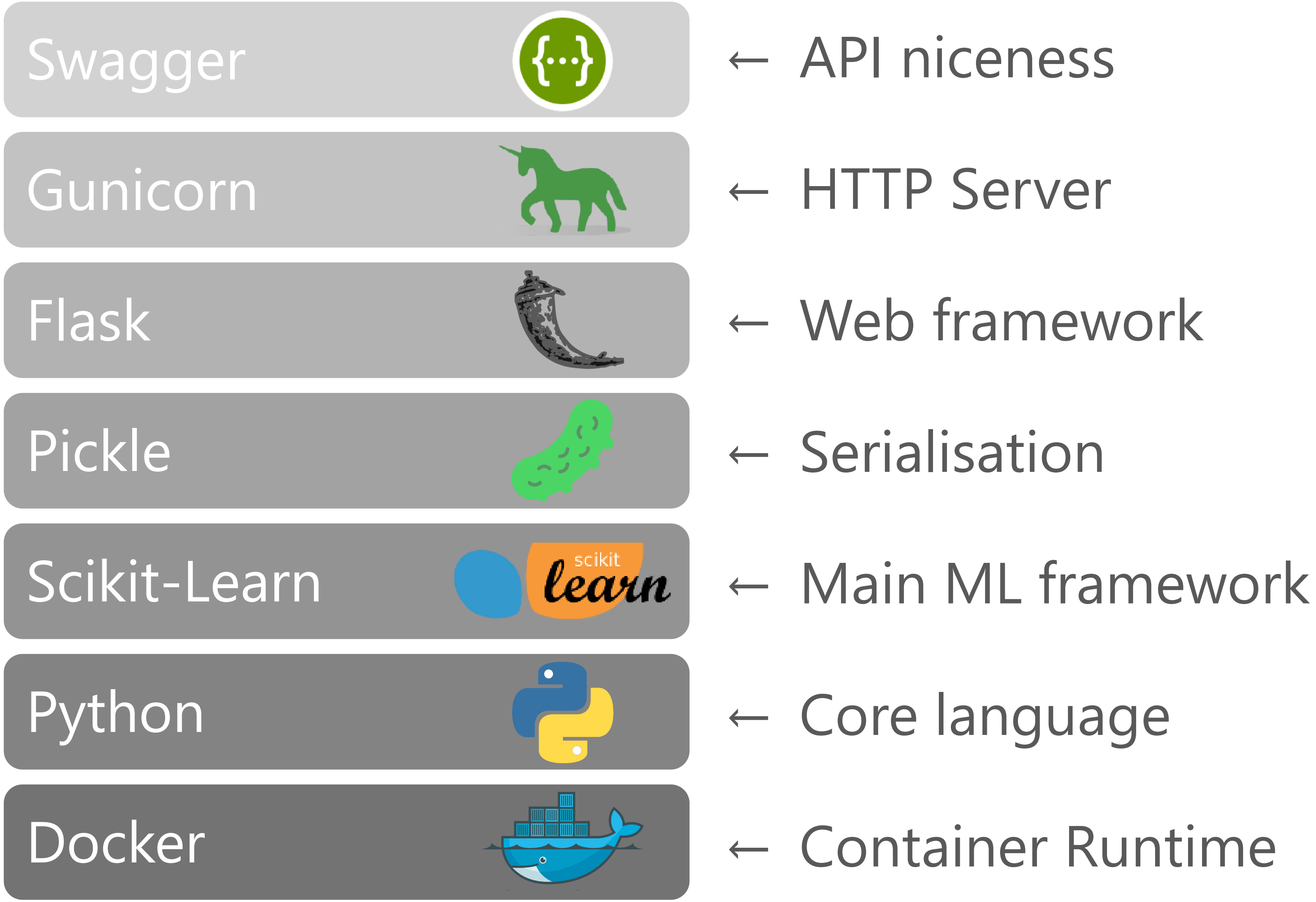
Model API Wrapper App

Some Decision Points

- Include model in container image or fetch at runtime?
- Make generic or tied to a specific model?
- What are my API parameters?
- Which web framework; Flask, Django, Gunicorn ?
- Base Python image, Alpine etc



Model API – Low Level Technology Stack



Swagger ← API niceness

Gunicorn ← HTTP Server

Flask ← Web framework

Pickle ← Serialisation

Scikit-Learn ← Main ML framework

Python ← Core language

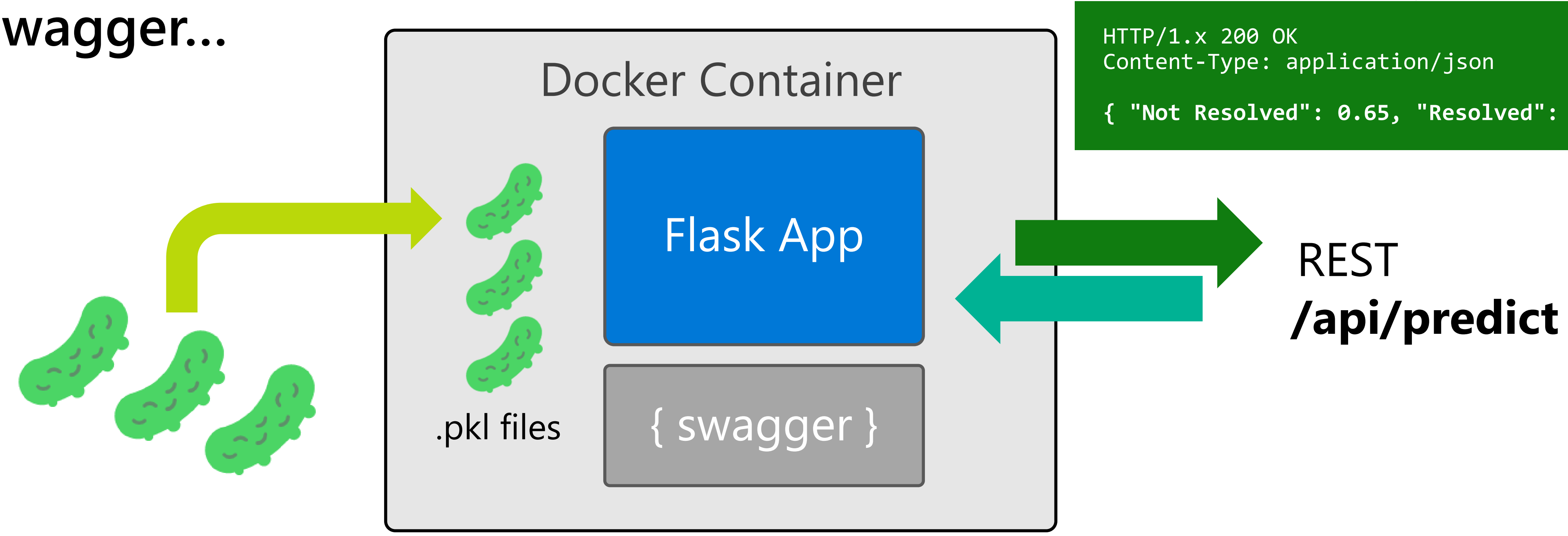
Docker ← Container Runtime

Wrapper App – Components

- Uses Flask web framework + Gunicorn
- Creates RESTful API for model parameters
- Consumes .pkl files
- Swagger…

```
POST /api/predict
{
  "force": "Thames Valley Police",
  "crime": "Bicycle theft",
  "month": 10
}
```

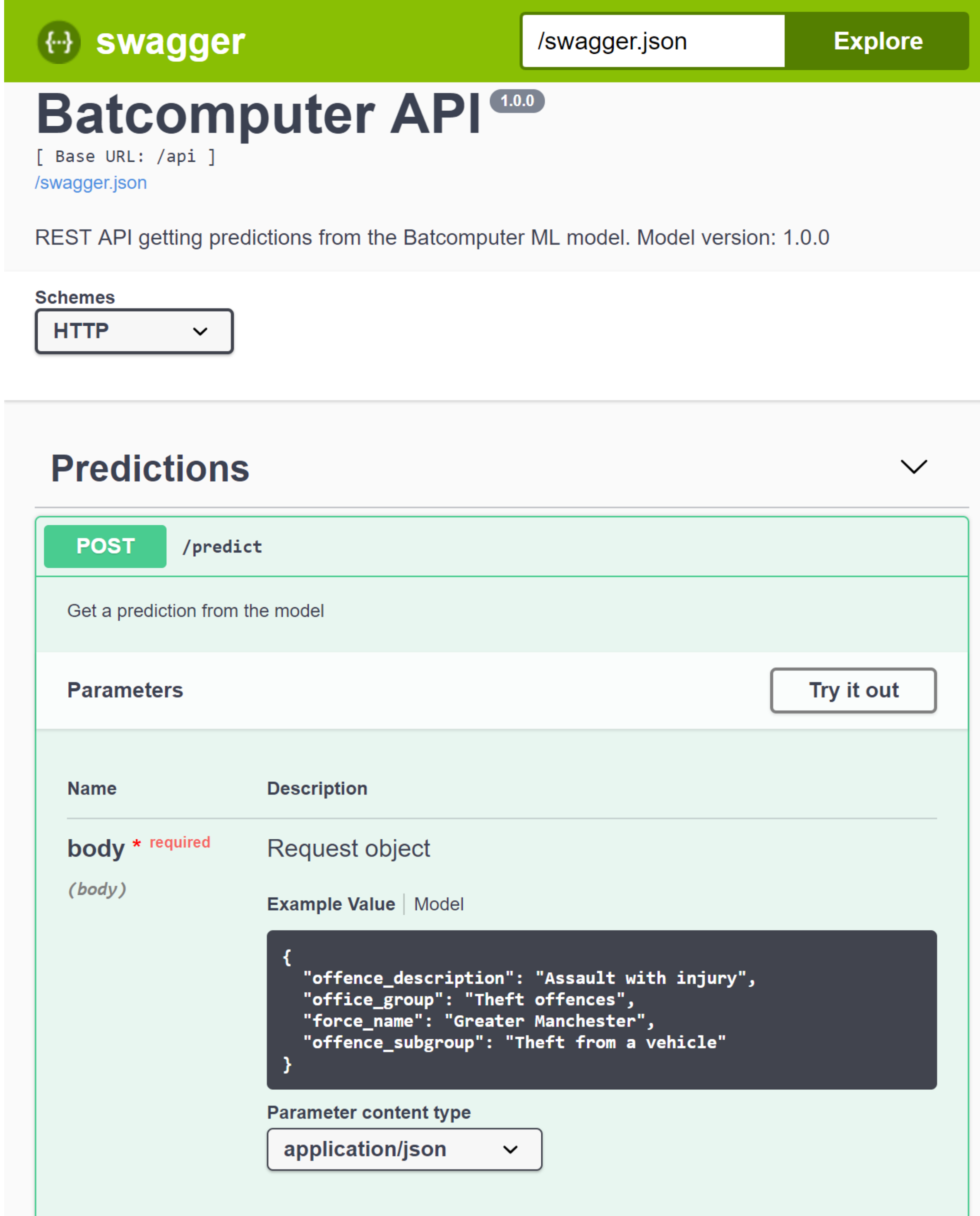
```
HTTP/1.x 200 OK
Content-Type: application/json

{ "Not Resolved": 0.65, "Resolved": 0.35 }
```



Swagger

- We want to be RESTful
- Dynamic
  - Generated from lookup & flags pickles at runtime
- Swagger UI
  - For testing & eye candy

OPENAPI INITIATIVE



Batcomputer API 1.0.0

[Base URL: /api]

/swagger.json

REST API getting predictions from the Batcomputer ML model. Model version: 1.0.0

Schemes: HTTP

Predictions

POST /predict

Get a prediction from the model

Parameters — Try it out

Name	Description
body * required *(body)*	Request object

Example Value | Model

```
{
  "offence_description": "Assault with injury",
  "office_group": "Theft offences",
  "force_name": "Greater Manchester",
  "offence_subgroup": "Theft from a vehicle"
}
```

Parameter content type: application/json

Building the Container Image

```
FROM python:3.6-slim-stretch

# Install Python requirements
ADD requirements.txt .
RUN pip3 install -r requirements.txt

# Add in our app and the pickle files
WORKDIR /app
ADD src .
ADD pickles/*.pkl ./pickles/

# Runtime configuration & settings
EXPOSE 8000

# Start the Flask server
CMD ["python3", "server.py"]
```

Base image is Debian based

This makes installing Python packages MUCH faster

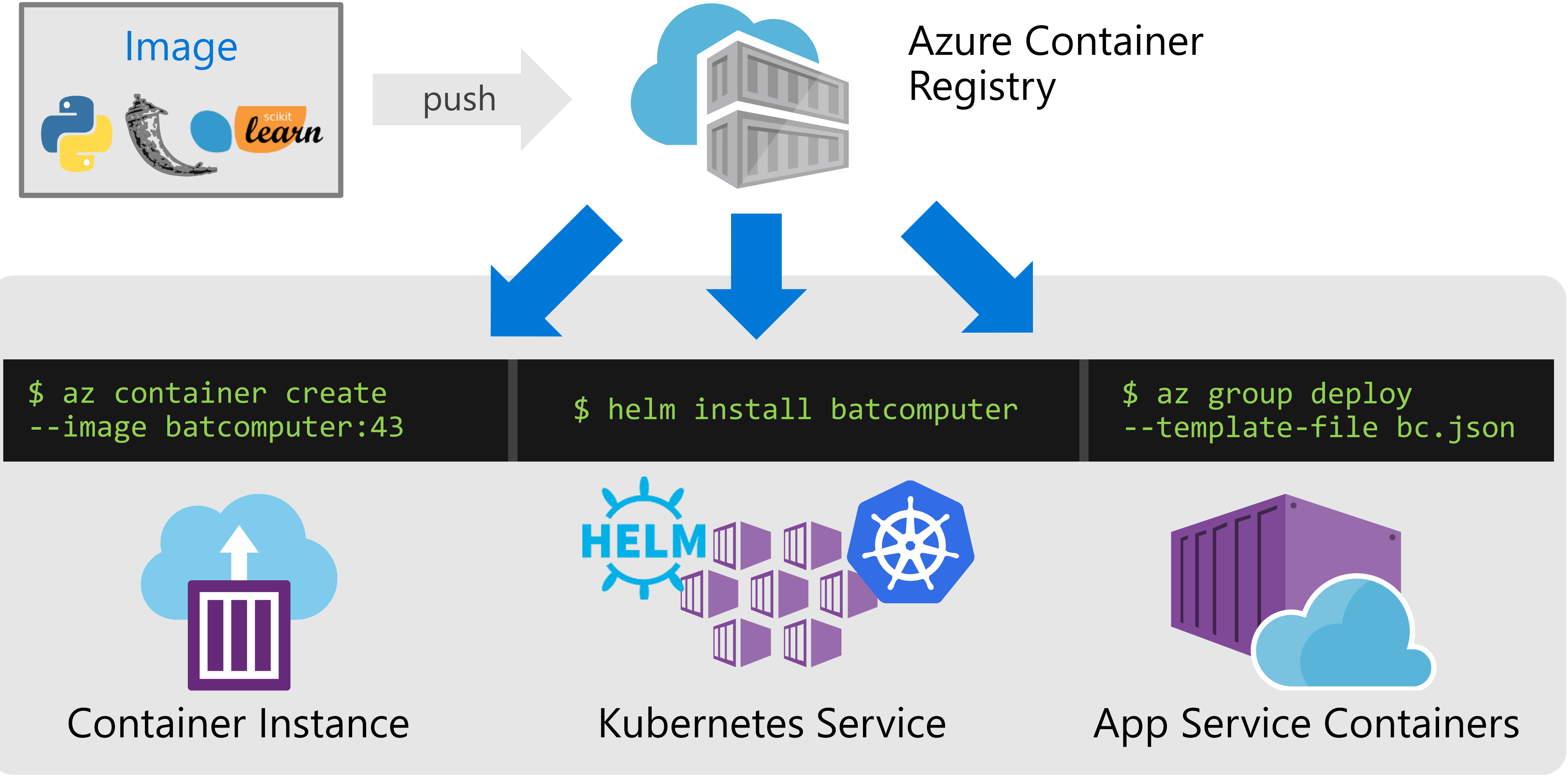
Add in app source and pickles

Alternative startup for Gunicorn

Requires no code changes

```
# Start the app via Gunicorn WSGI server
ENV GUNICORN_CMD_ARGS "--bind=0.0.0.0:8000"
CMD ["gunicorn", "--access-logfile", "-", "server"]
```

Container Deployment

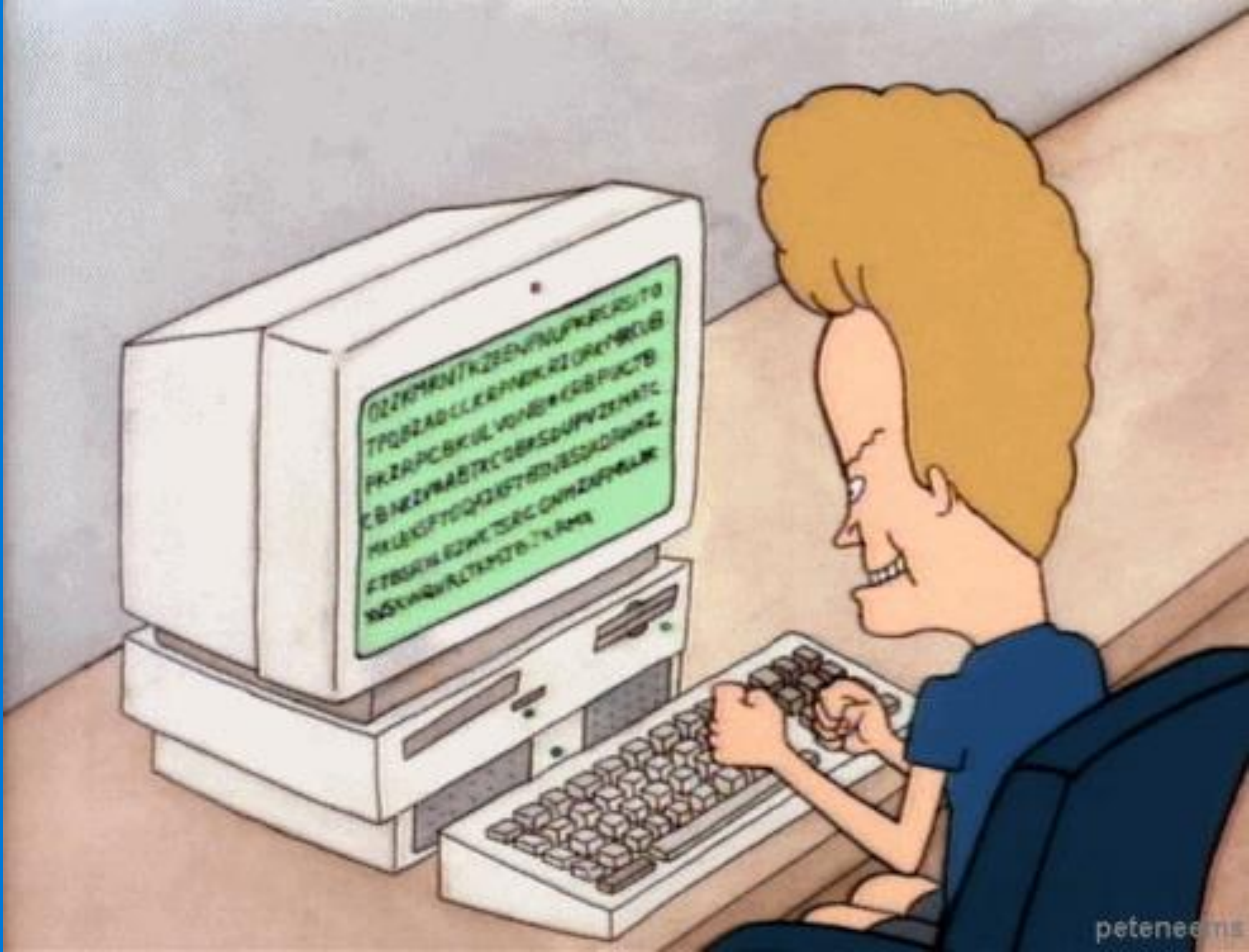


```
$ az container create
--image batcomputer:43
```

```
$ helm install batcomputer
```

```
$ az group deploy
--template-file bc.json
```

Demo

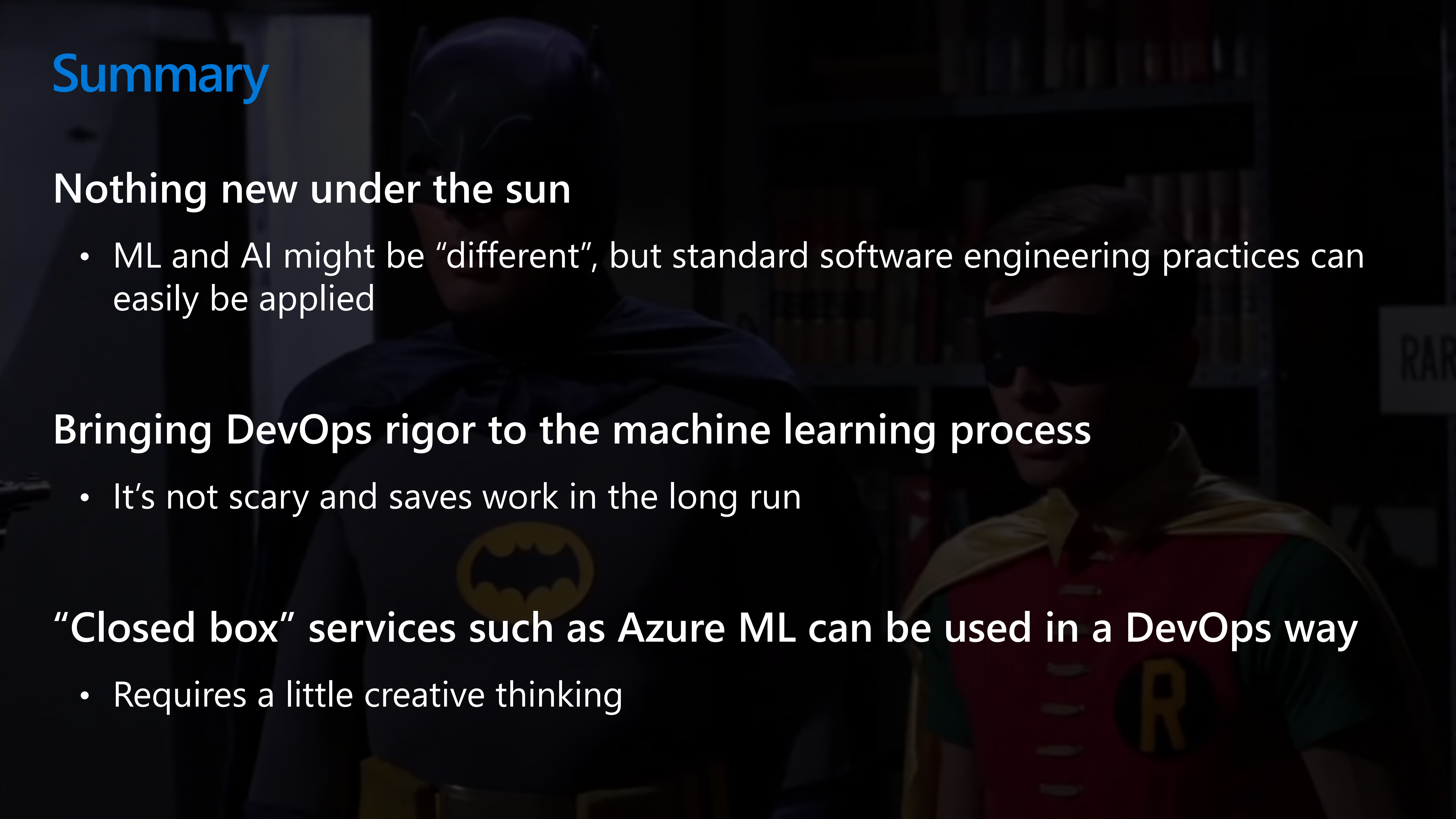


Summary

Some Learnings / Gotchas

- Pickled Scikit-learn models are version sensitive
- Keep Python version in sync everywhere
- Don’t use Alpine, use Debian Slim as container image base
- Writing your own wrapper isn’t hard
- Azure ML is has a complex but powerful SDK
- Tracking & managing parameters & variables can get tricky

Summary



Nothing new under the sun

- ML and AI might be “different”, but standard software engineering practices can easily be applied

Bringing DevOps rigor to the machine learning process

- It’s not scary and saves work in the long run

“Closed box” services such as Azure ML can be used in a DevOps way

- Requires a little creative thinking

Microsoft Azure