

# Project Batcomputer

Making DevOps work for  
Machine Learning

**Ben Coleman**  
**Phil Harvey**

**@BenCodeGeek**  
**@CodeBeard**



[batcomputer.benco.io](http://batcomputer.benco.io)

# Background

## Motivation

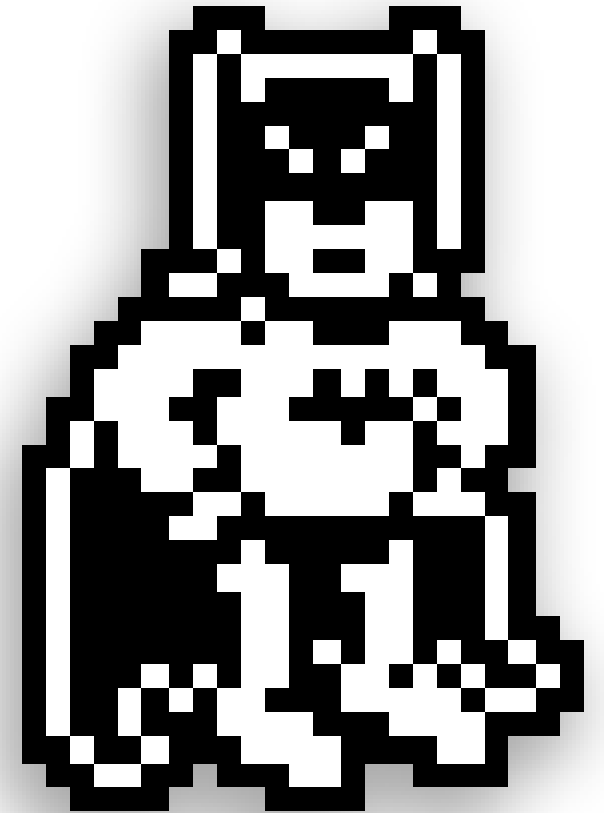
- Understand challenges in operationalisation of ML models
- “DevOps for AI”
- Integration of “all in in one” processes with real DevOps approach

## Why Batcomputer?

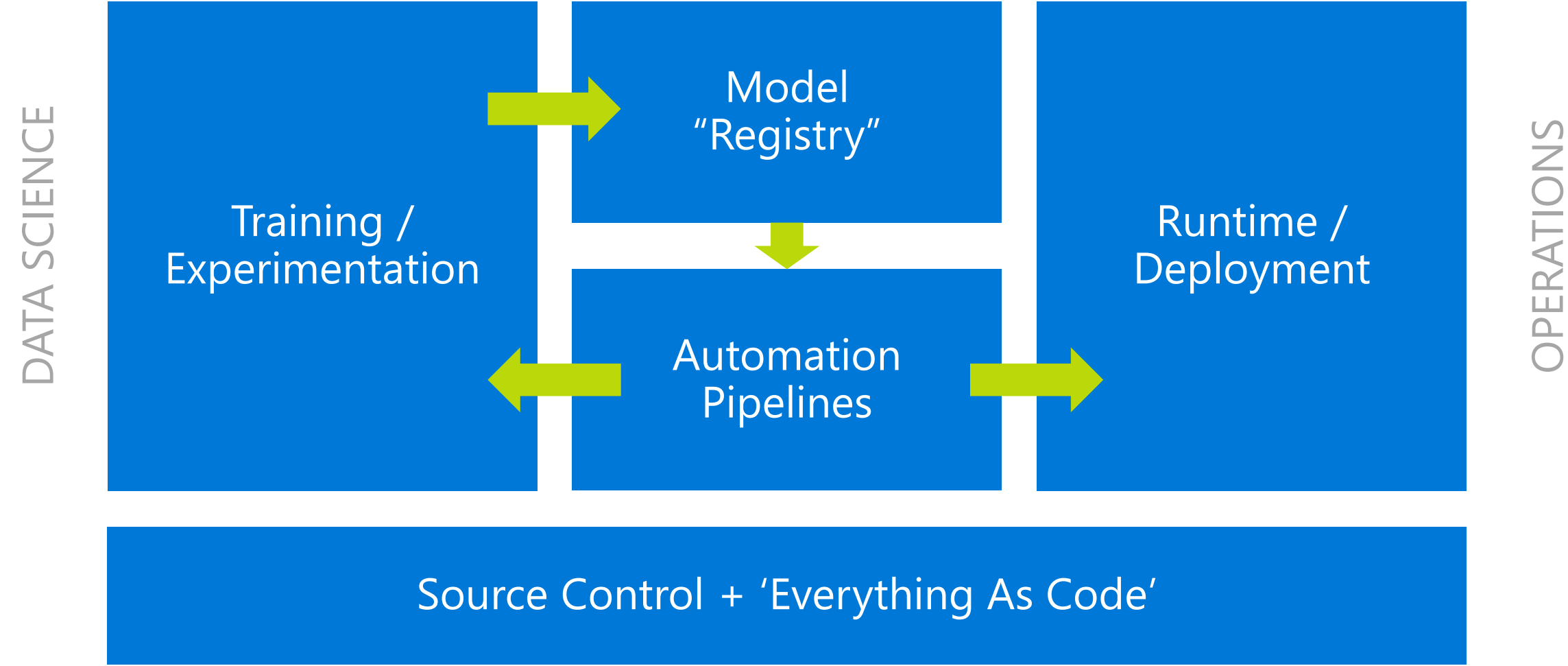
- Police recorded crime and outcomes data
- Source data as CSV - <https://data.police.uk/data>
- Build model of a given crime and region to predict – “Would you get caught?”

# Core Principals & Benefits

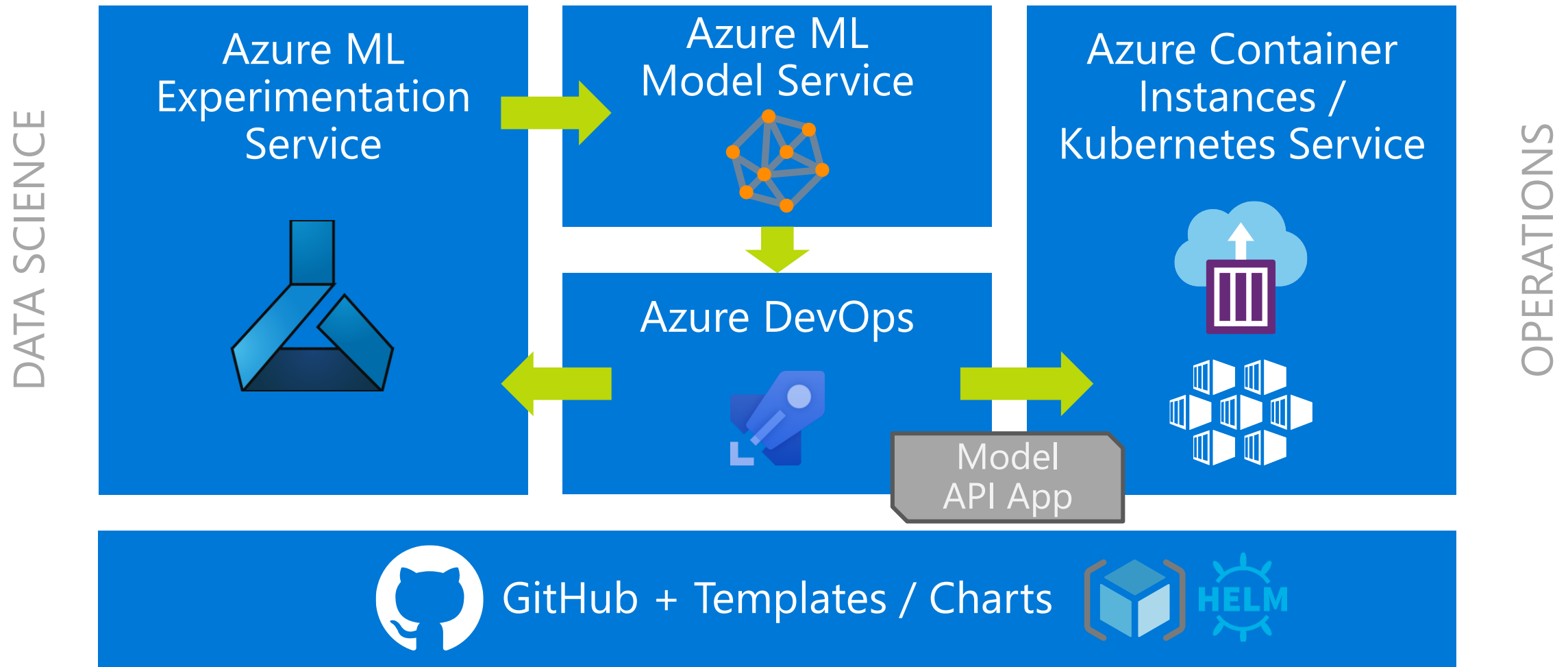
- **Decouple** model training experiments from operations/runtime
- **Continuous Integration**  
Automated training, API builds & deployment
- **Versioned** models and APIs
- A real **RESTful API**, not a thin HTTP wrapper
- Configuration as code, infrastructure as code
- **Traceability**



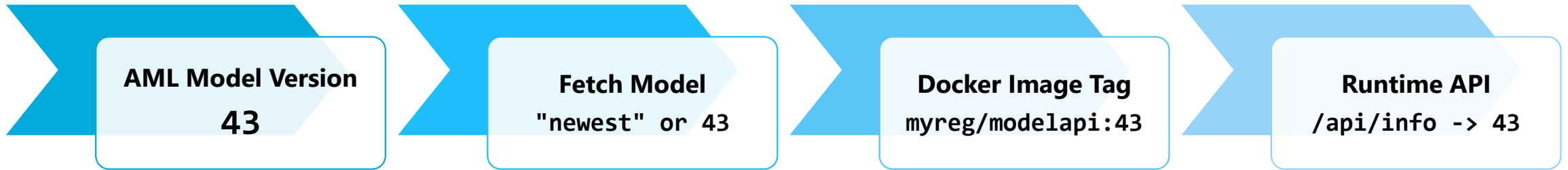
# Conceptual Building Blocks



# Conceptual Building Blocks – Project Batcomputer



# Versioning – Many Touch Points



batcomputer-model

[← Back to Models](#) [Refresh](#) [Create Image](#) [Delete](#) [Get Link](#)

[Details](#) [Deployments](#)

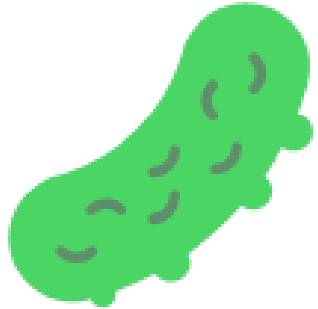
## ATTRIBUTES

Version	43
ID	batcomputer-model:43
Date Registered	08/03/2019, 13:18:30 UTC
Location	<a href="#">aml://asset/18173acf8a684:</a>
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

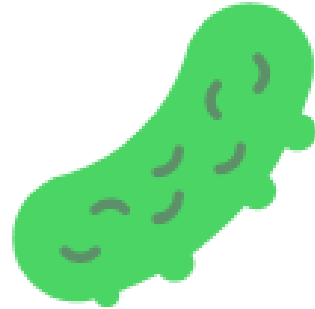
# The 'Model Registry' – Not Just The Model



## model.pkl

Scikit-learn model/classifier

Standard object rehydration,  
version sensitive



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

## lookup.pkl

Python dictionary of  
dictionaries

Mapping  
parameters/strings to num  
for predict function



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

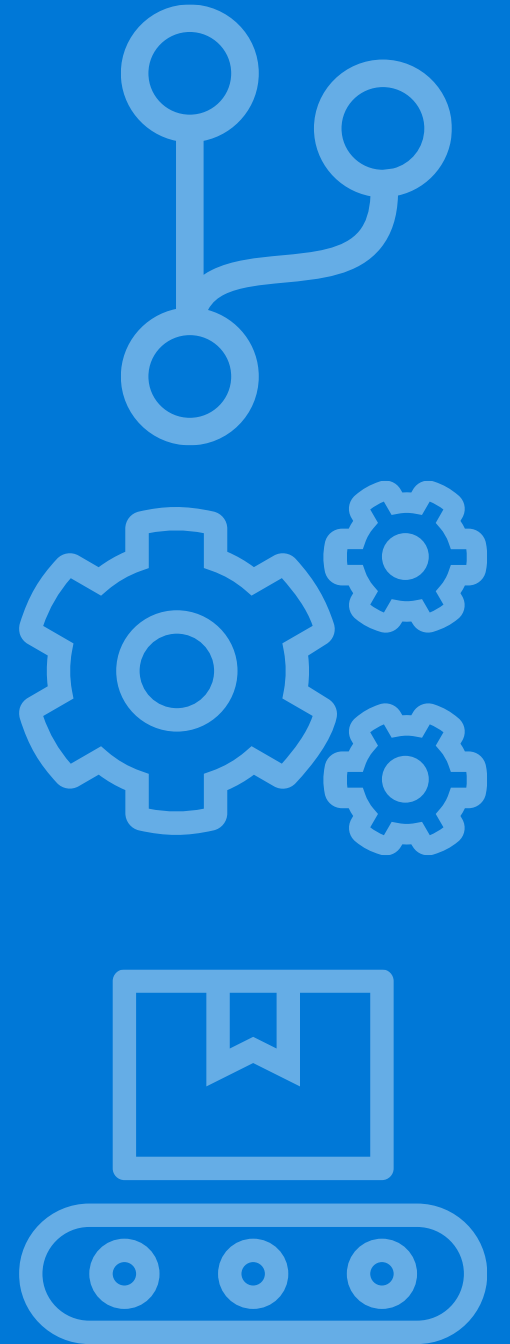
## flags.pkl

Python array

Maps output of prediction  
function to human readable  
labels

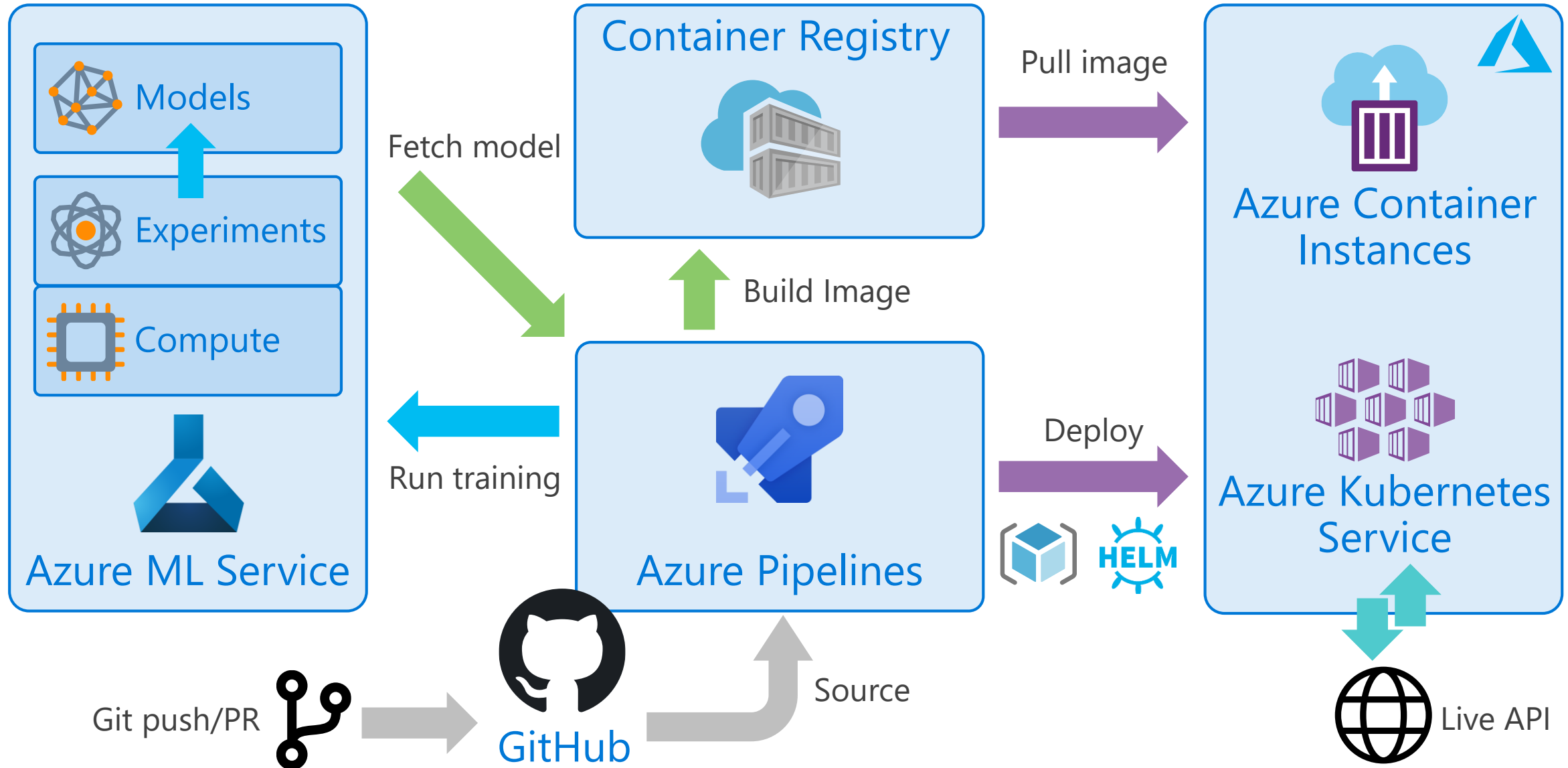
# DevOps

## Continuous Integration / Continuous Delivery



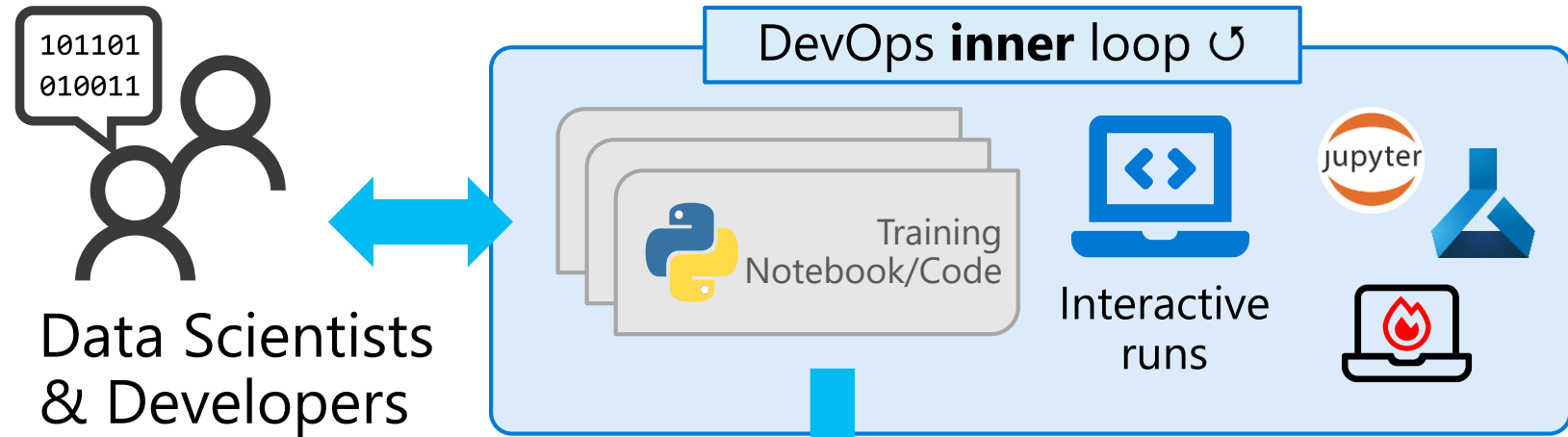


# Model Training & Deployment – End To End Flow



# Core DevOps Practice - Continuous Integration

Development & experimentation



Commit into git

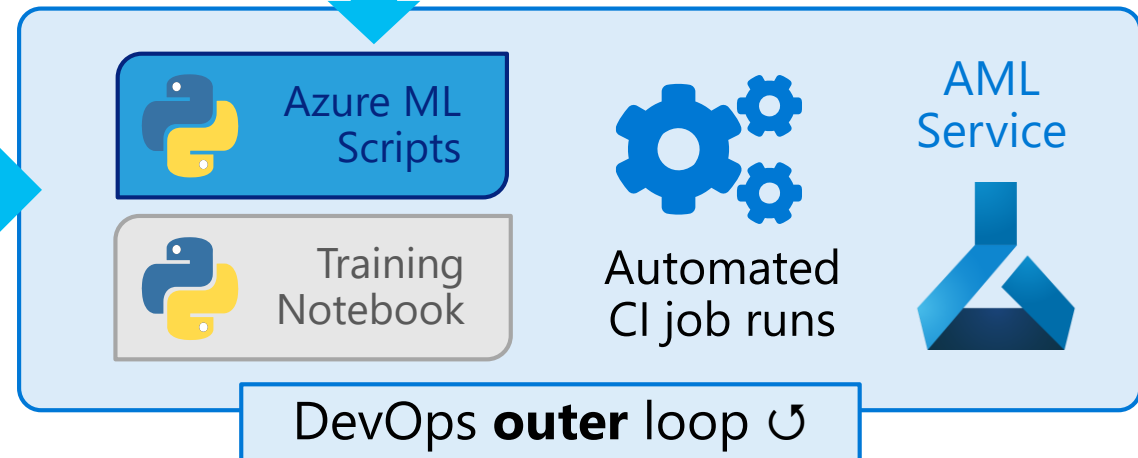


Git Repo

Checkout branch

CI Trigger

CI triggered training & testing job runs

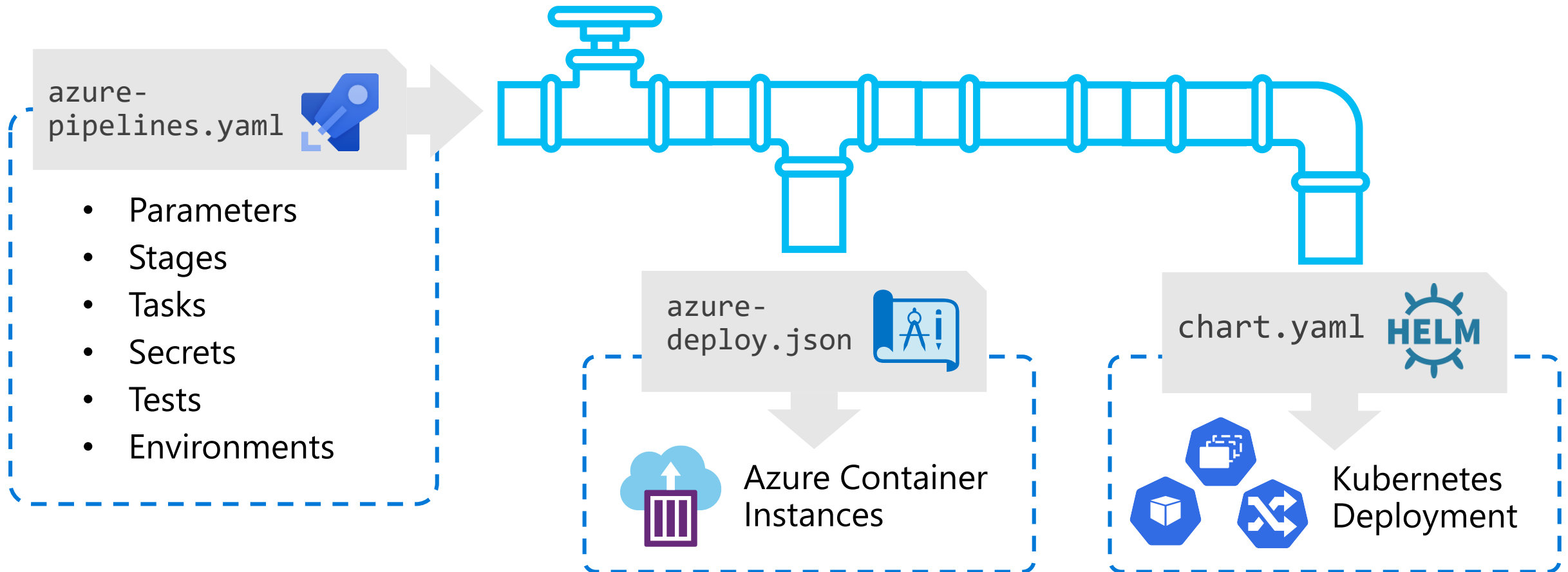


# Infrastructure As Code

Standard DevOps working practice

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

Store with your application under source control / Git



# Testing

## Integration tests against the real API using Postman & Newman



Filter

History

Collections

Trash

Batcomputer  
5 requests

POST Predict Force

GET Get Swagger JSON

GET Get Info Metadata

POST Predict Crime

GET Get Params

Dark Sky API  
1 request

Functions Demo  
2 requests

Graph Text Adve...  
5 requests

POST Predict Crime

Predict Crime

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

Send

Save

Params

Authorization

Headers (1)

Body

Pre-request Script

Tests

Cookies

Code

Comments (0)

```
1 pm.test("Predict Crime: Successful POST request", function () {
2   pm.response.to.be.ok;
3 });
4
5 pm.test("Predict Crime: Response valid & JSON body", function () {
6   pm.response.to.be.ok;
7   pm.response.to.be.withBody;
8   pm.response.to.be.json;
9 });
10
11 pm.test("Predict Crime: Validate results", function () {
12   var jsonData = pm.response.json();
13   pm.expect(jsonData.caughtProb).to.exist;
14   pm.expect(jsonData.notCaughtProb).to.exist;
15 });
16
17 pm.test("Predict Crime: Validate prediction values", function () {
18   var jsonData = pm.response.json();
19   pm.expect(jsonData.caughtProb).gt(0.001);
20   pm.expect(jsonData.caughtProb).lt(0.999);
21 });
```

16  
Total tests

16  
0  
0

Passed  
Failed  
Others

100%  
Pass percentage

5s 983ms  
Run duration

Release

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

Artifacts

batcomputer-src  
6e2078709  
master

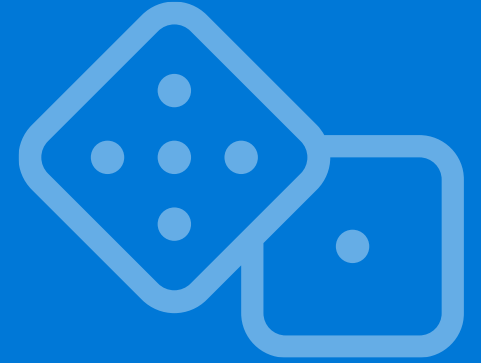
acr-image  
45

Stages

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

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

# Machine Learning & Training



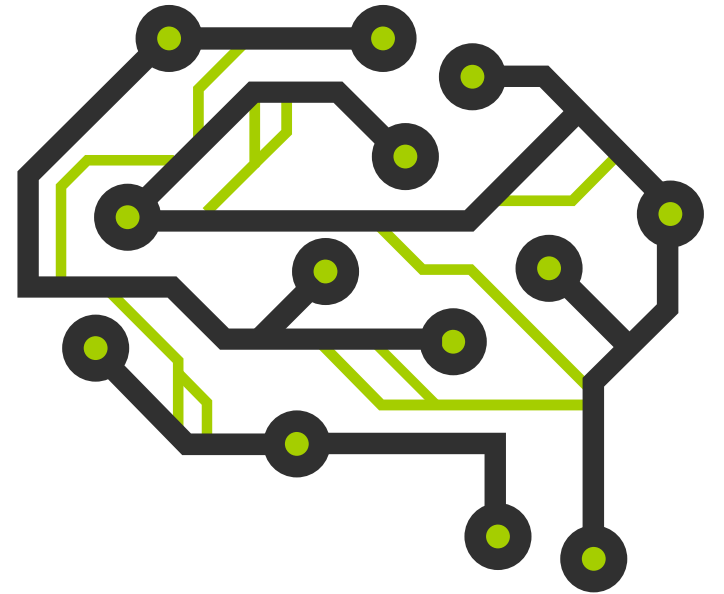
# Machine Learning – Training Scripts

**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

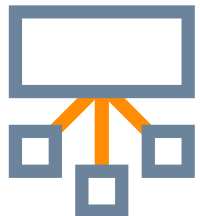
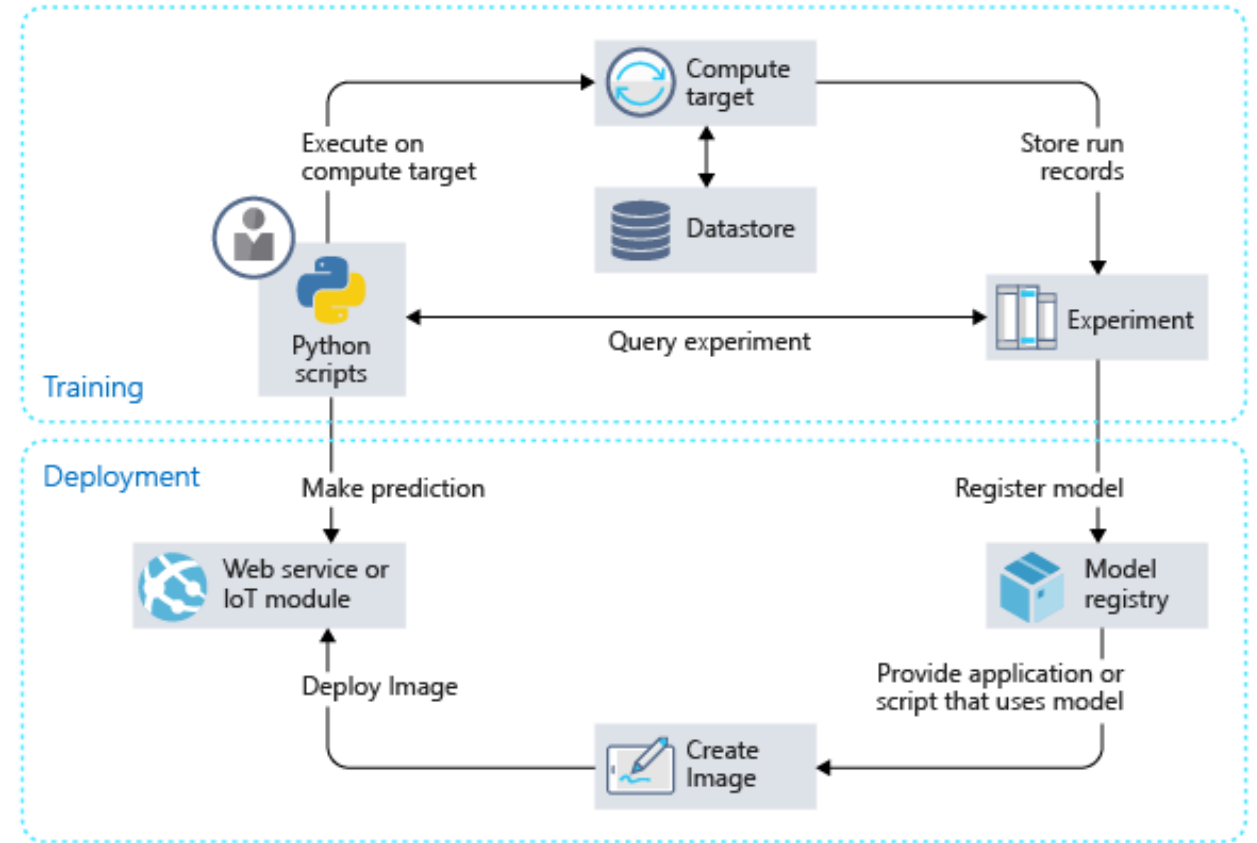
Azure Machine Learning service provides SDKs and services to prep data, train, and deploy machine learning models

Driven by Python SDK

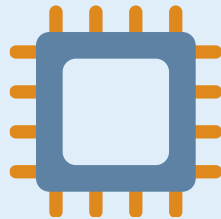
Range of training & experimentation compute targets

Model management

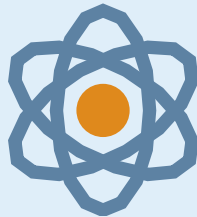
“Project Batcomputer Operationalisation Process”



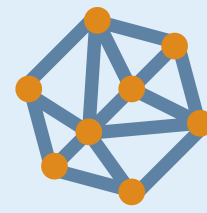
Pipelines



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)



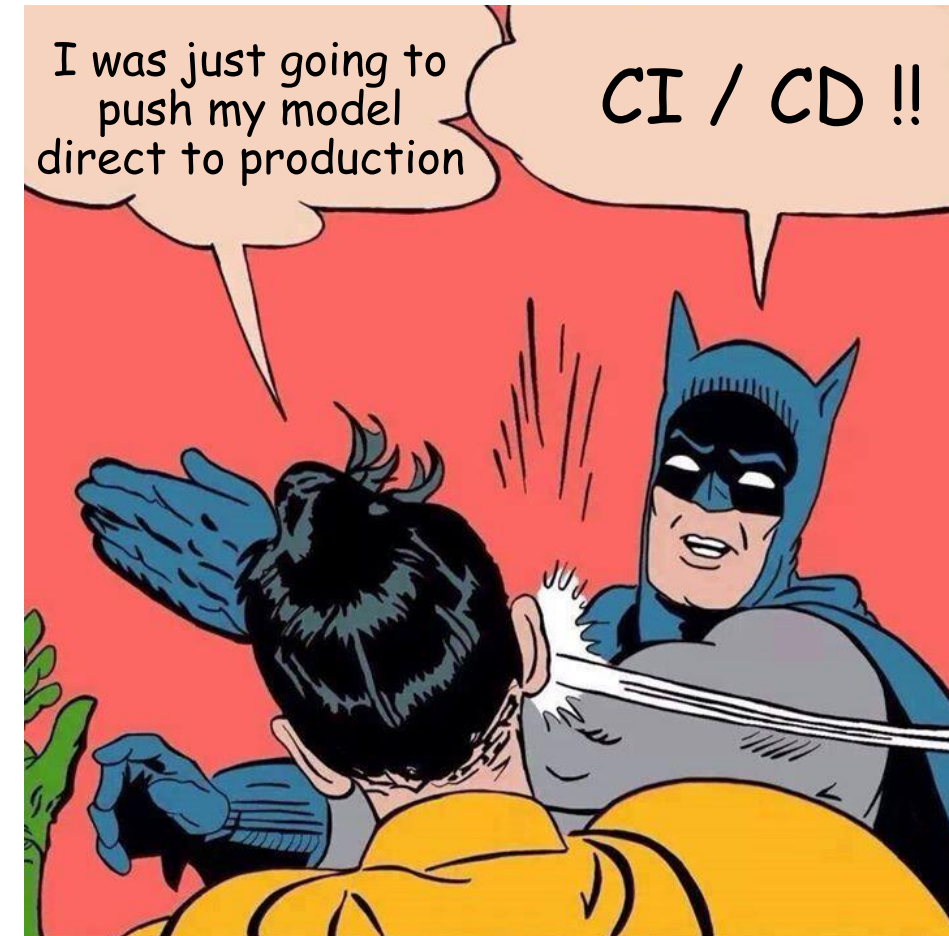
Azure Machine  
Learning SDK  
for Python



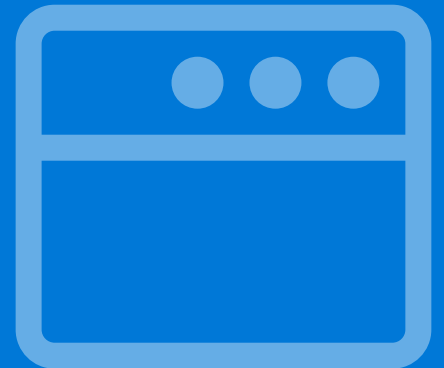
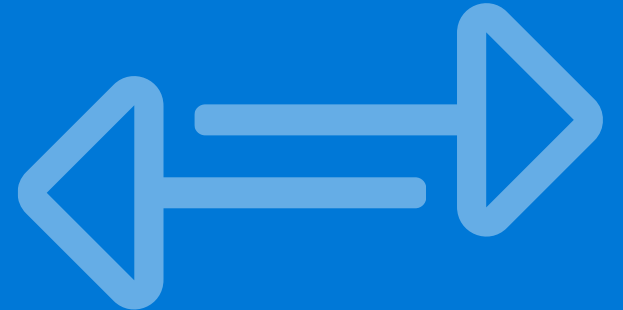
# Azure ML Deployment

**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

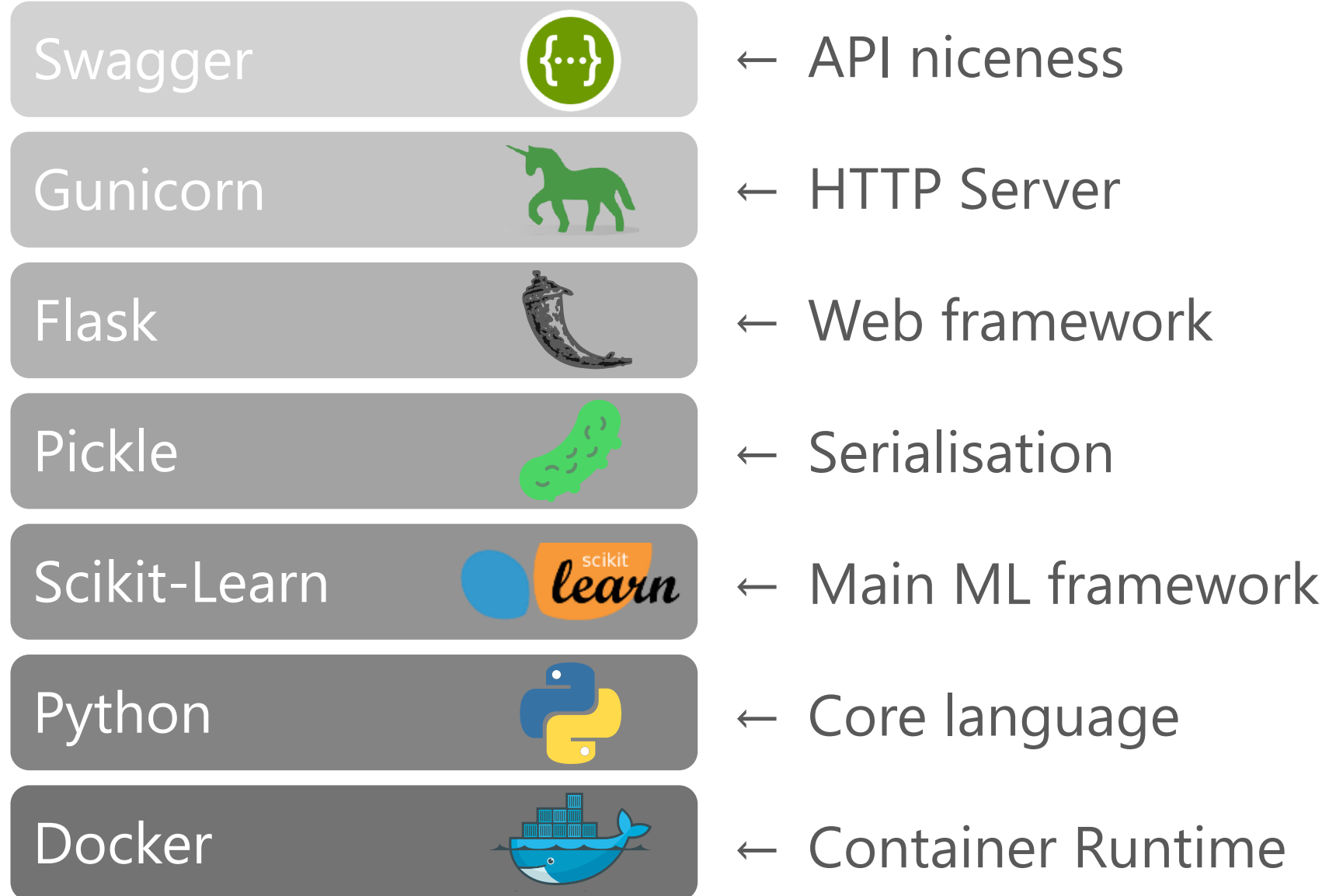


# Some Decision Points

- 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

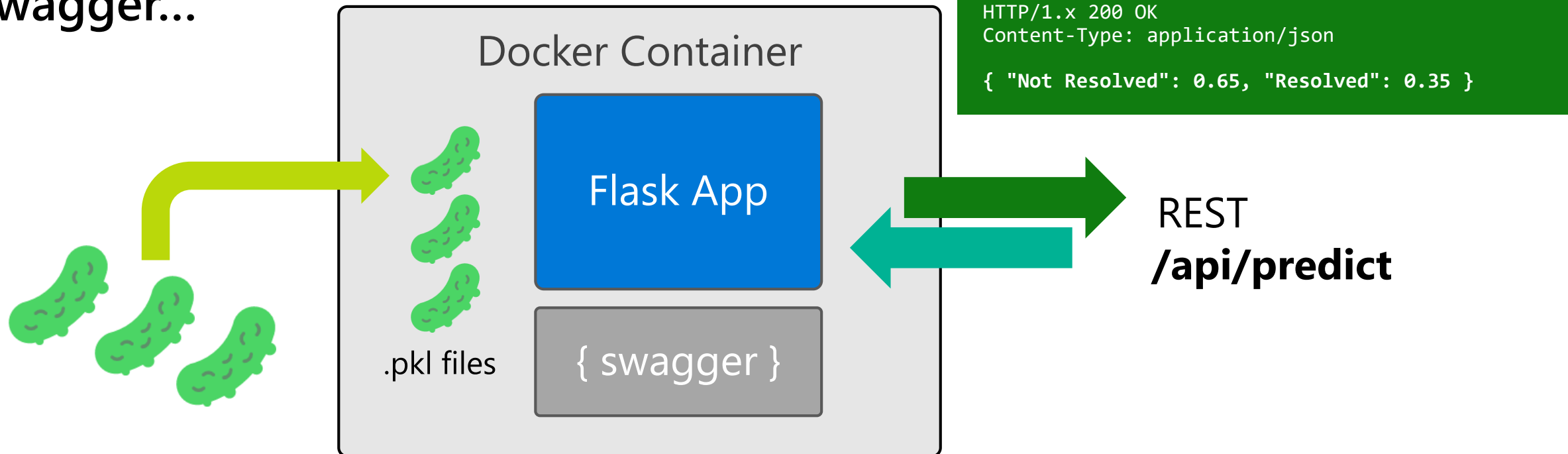


# Model API – Low Level Technology Stack



# Wrapper App – Components


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



# Swagger

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



 **swagger**

/swagger.json

Explore

## 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
<b>body</b> * required (body)	Request object

Example Value | Model

```
{  "offence_description": "Assault with injury",  "offence_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

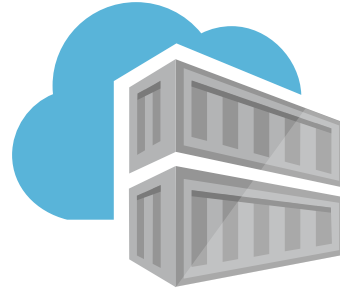
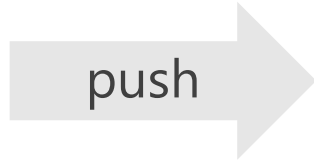
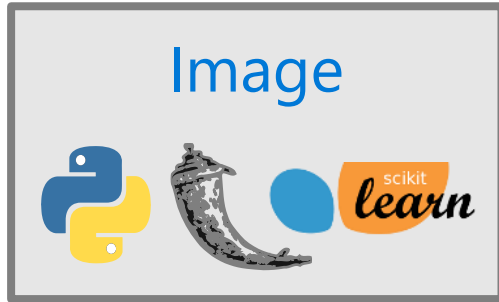
← This makes installing Python packages MUCH faster

← 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



Azure Container  
Registry



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

```
$ helm install batcomputer
```

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



Container Instance



Kubernetes Service



App Service Containers



Close

# 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



# Detailed Azure DevOps Flow

