Microsoft Azure

Project Batcomputer

Making DevOps work for Machine Learning

Ben Coleman
@BenCodeGeek

batcomputer.benco.io

v5

Background

Motivation

- Understand challenges in operationalisation of ML models
- “DevOps for AI”
- Integration of “all in in one” processes with real world DevOps approach
- Learning exercise & something fun to do ;)

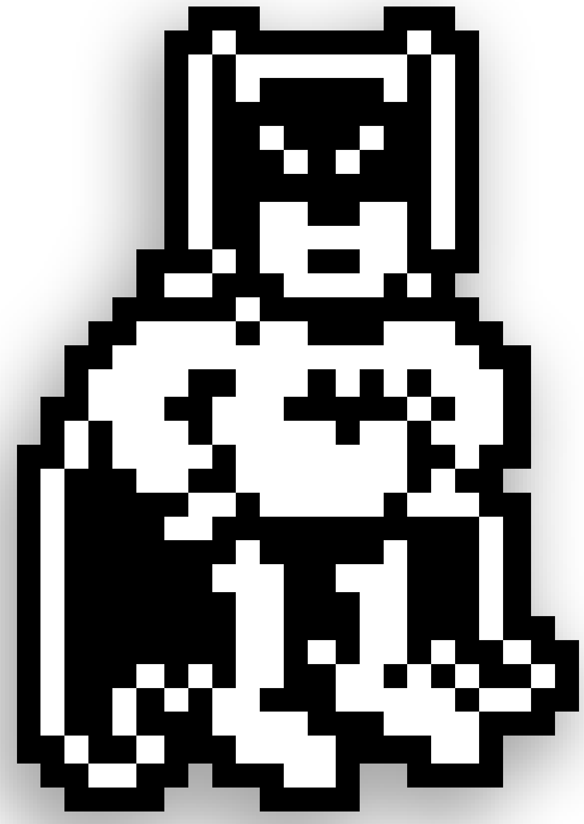
Background

Why Batcomputer?

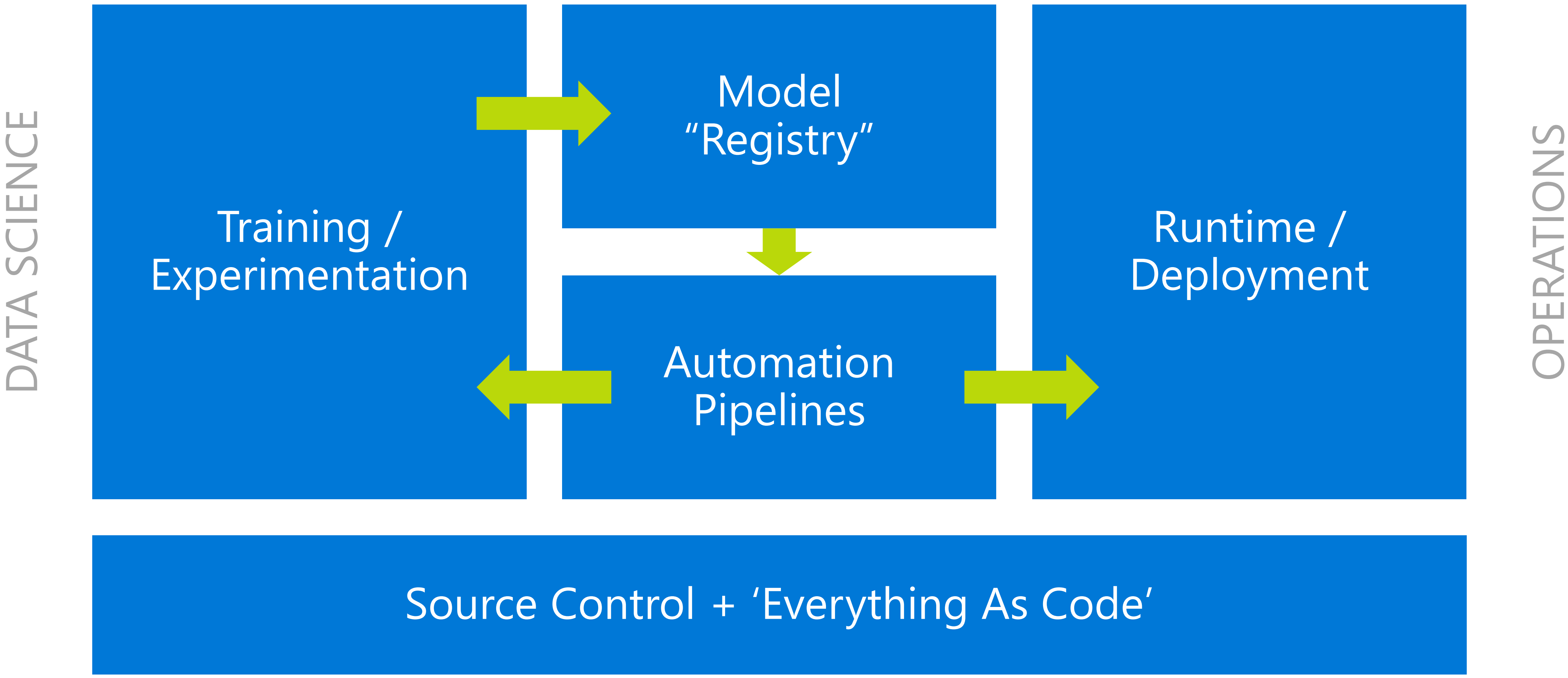
- Police recorded crime and outcomes data
- Source data as CSV - https://data.police.uk/data
- Build model of a given crime and region to predict – Would you get “caught” ?
- “That sounds a bit like something Batcomputer would try to answer”

Core Principals & Benefits

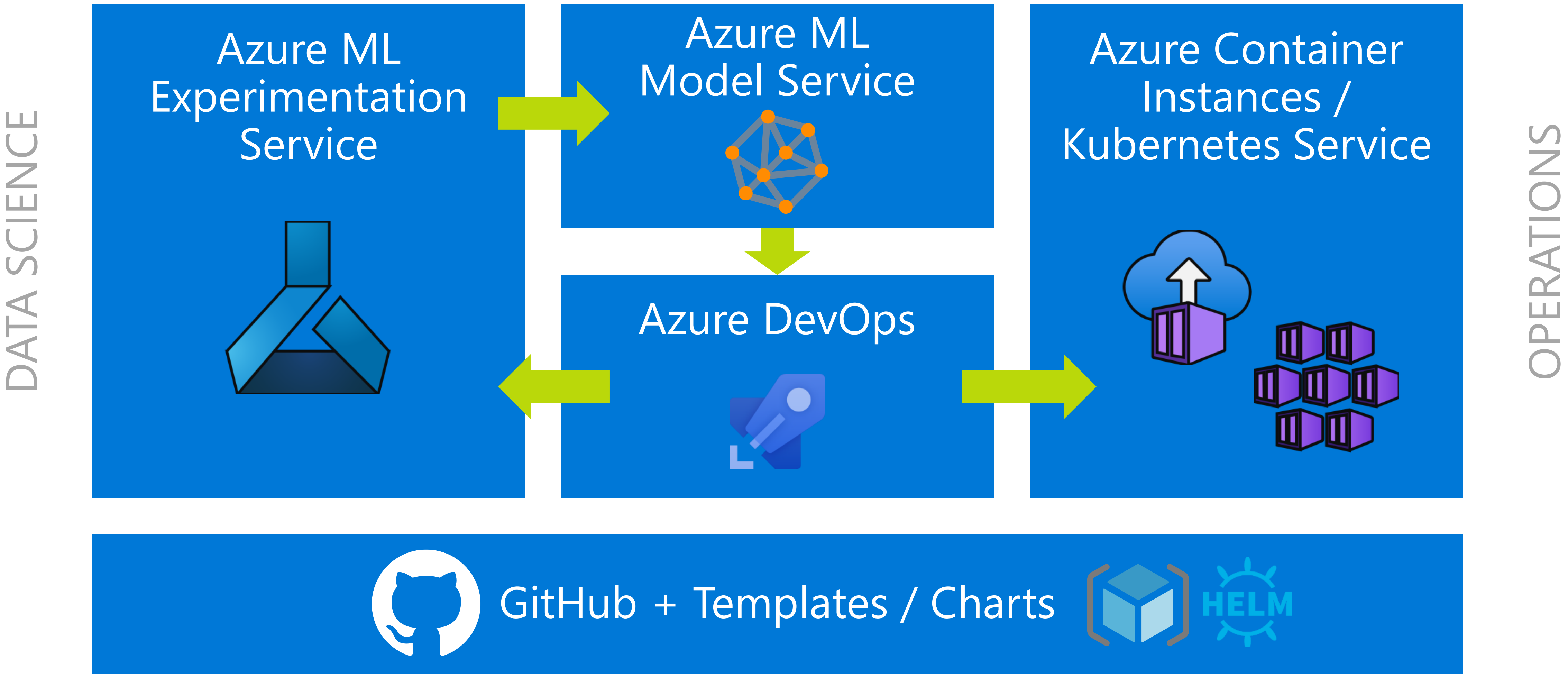
- **Continuous Integration**
  Automated training, API builds
- **Continuous Deployment**
  Automated releases, testing
- **Versioned** models and APIs
- A real **RESTful API**, not a thin HTTP wrapper
- Configuration as code, infrastructure as code
- **Traceability**



Conceptual Building Blocks



Conceptual Building Blocks – Project Batcomputer



Introduction to The Azure Services

Azure Machine Learning Service

- Streamline the building, training and deployment of machine learning models
- Python SDK
- Use standard frameworks: PyTorch, scikit-learn, TensorFlow
- UI for building experiments
- Lots more…

Azure DevOps Pipelines

- Build, test, and deploy with CI/CD
- Works with any language, platform, and cloud.
- Connect to GitHub or any other Git to deploy continuously
- Highly extensible
- Range of automation scenarios

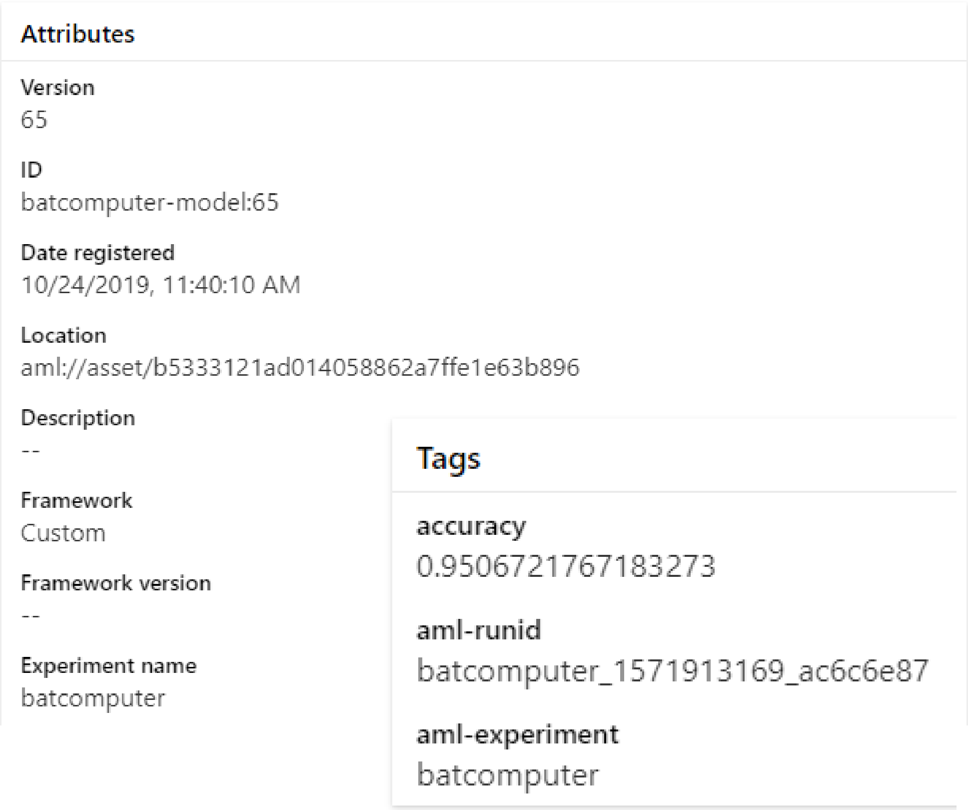
Azure Kubernetes Service

- Fully managed Kubernetes service in Azure – AKS

Kubernetes:

- Orchestrate and run containers
- Robust & scalable
- Simplify the running of complex applications

Versioning – Many Touch Points


Attributes

Version
65

ID
batcomputer-model:65

Date registered
10/24/2019, 11:40:10 AM

Location
aml://asset/b5333121ad014058862a7ffe1e63b896

Description
--

Framework
Custom

Framework version
--

Experiment name
batcomputer

Tags

accuracy
0.9506721767183273

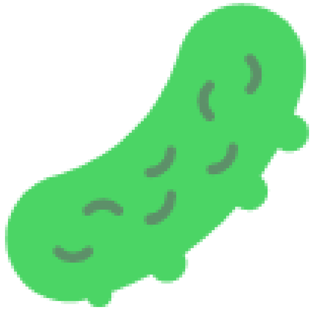
aml-runid
batcomputer_1571913169_ac6c6e87

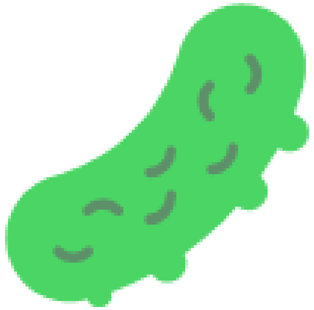
aml-experiment
batcomputer

Also…

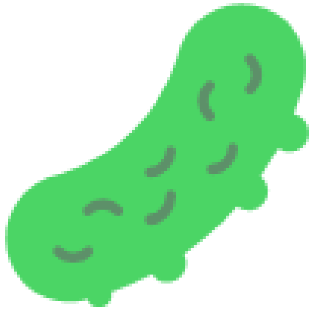
- Resource names in Azure controlled via ARM templates
- DNS names & prefixes,
  e.g. `batcomputer-`**`65`**`.westeurope.azurecontainer.io`
  `batcomputer.kube.benco.io/test-`**`65`**
- Object names in Kubernetes (pods, services), via Helm chart

The 'Model Registry' – Not Just The Model





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



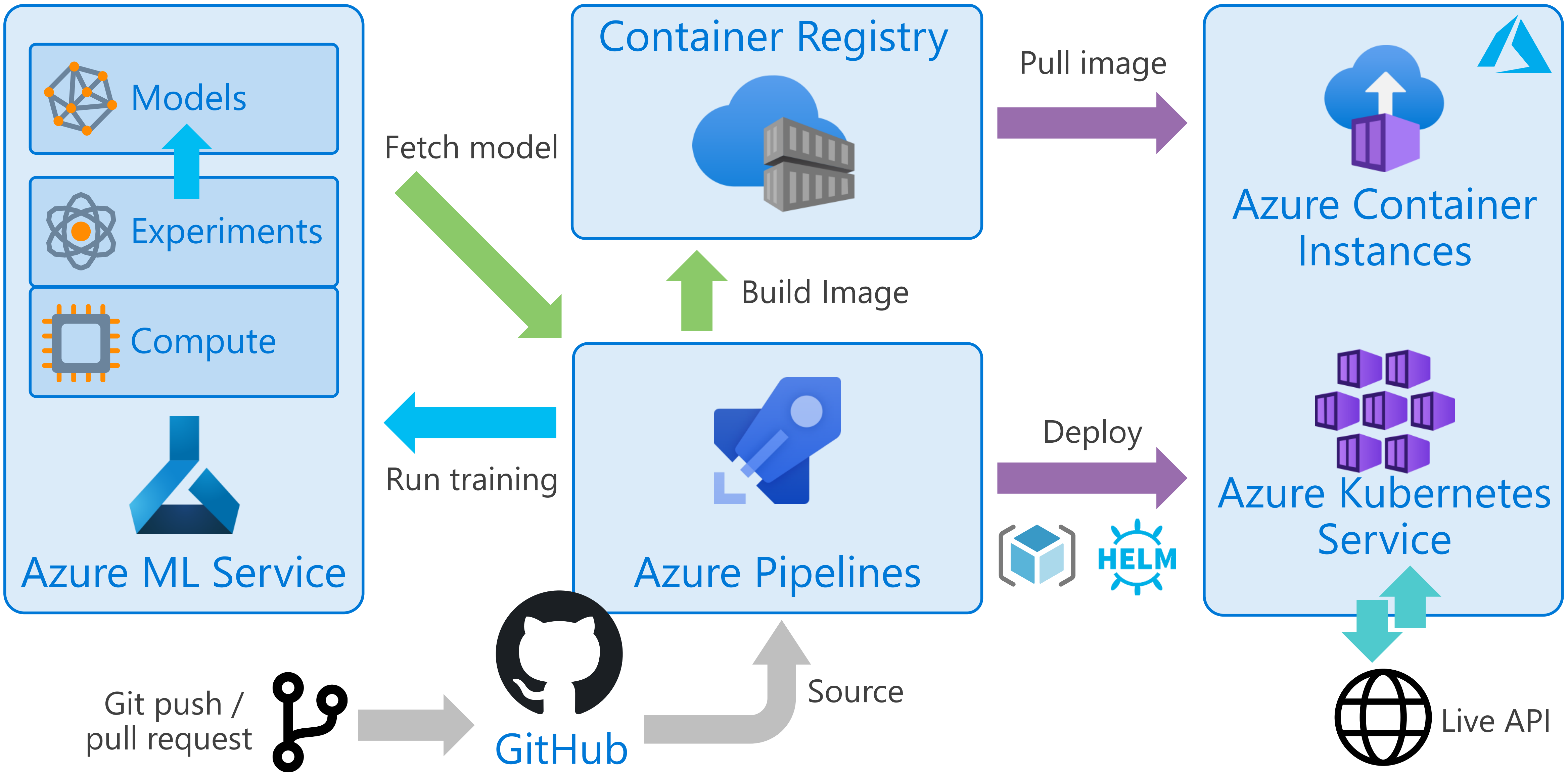
```
[
  "conviction",
  "dropped"
]
```

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

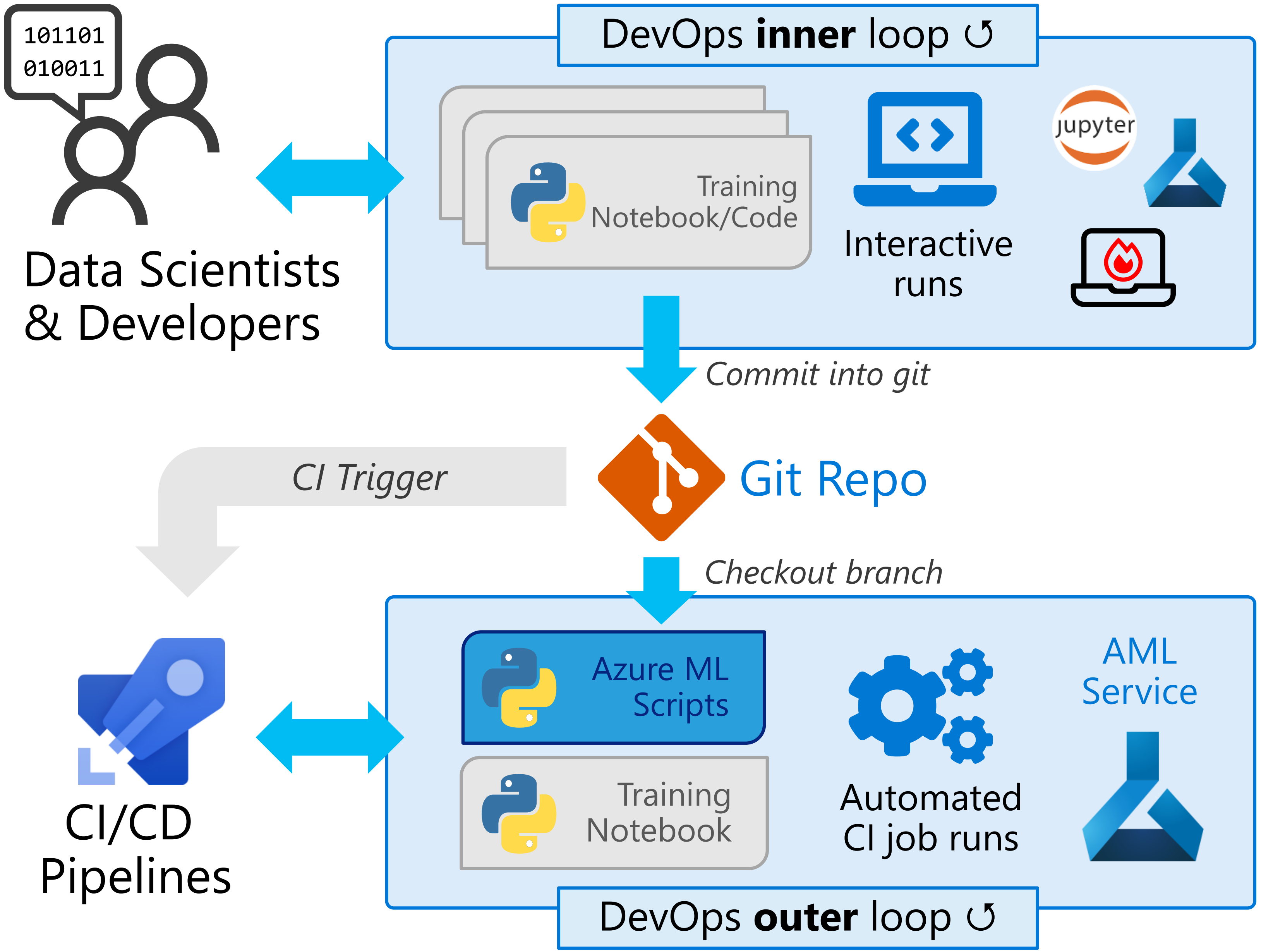
Model Training & Deployment – End To End Flow



Core DevOps Practice - Continuous Integration

Development & experimentation

CI triggered training & testing job runs

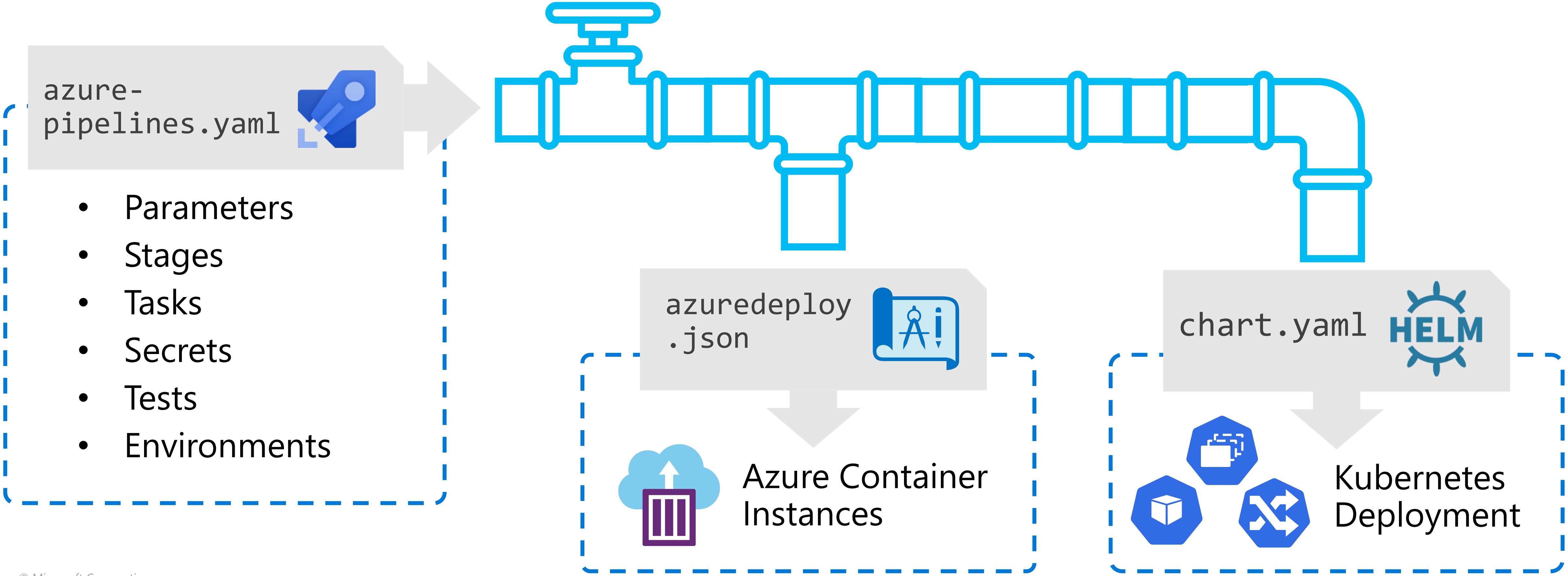


Infrastructure As Code

Standard DevOps working practice

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

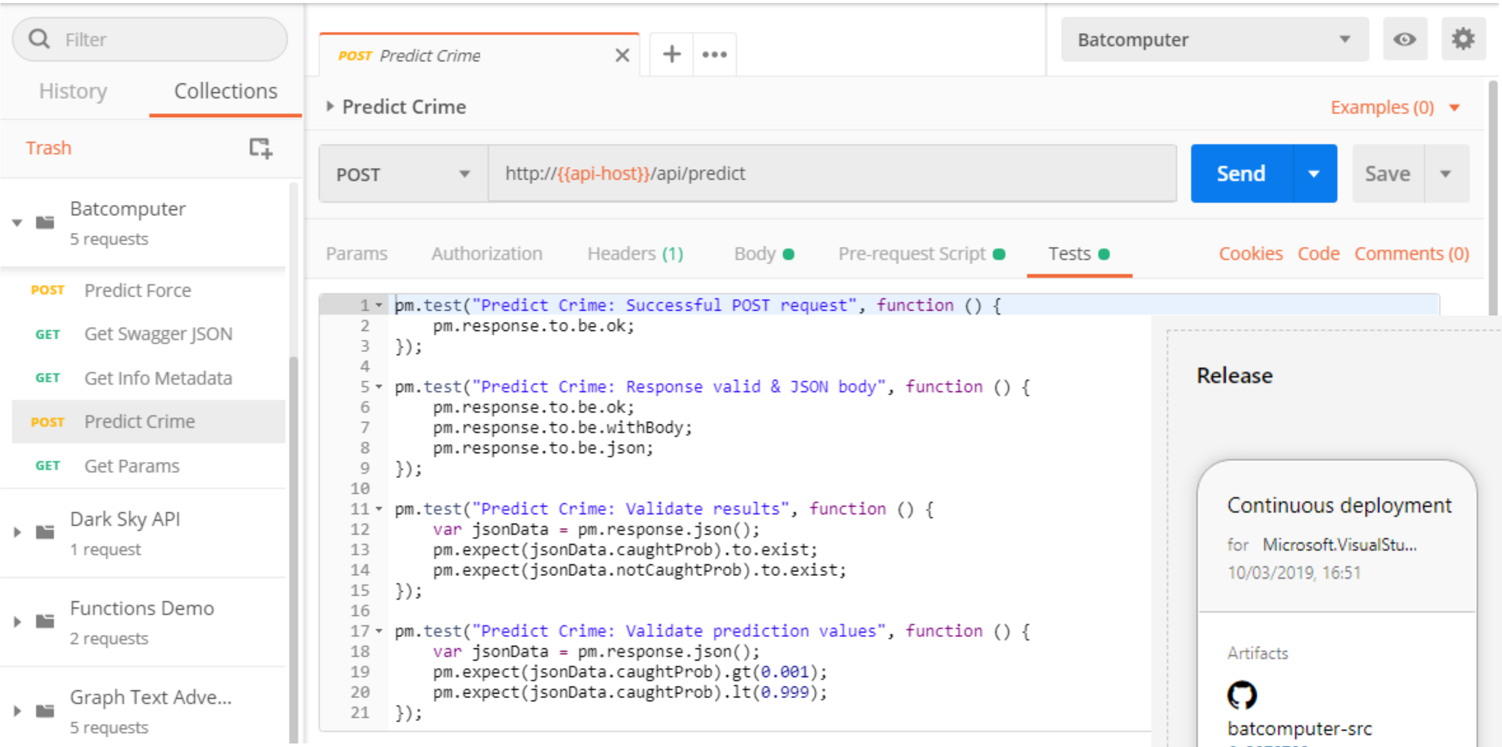
Store with your application under source control / Git

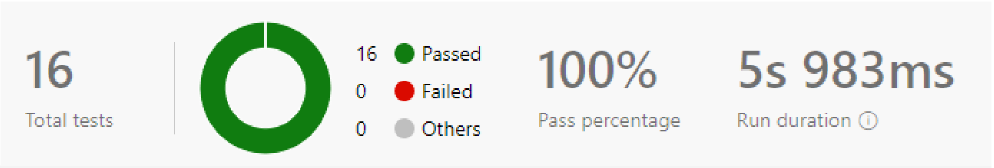


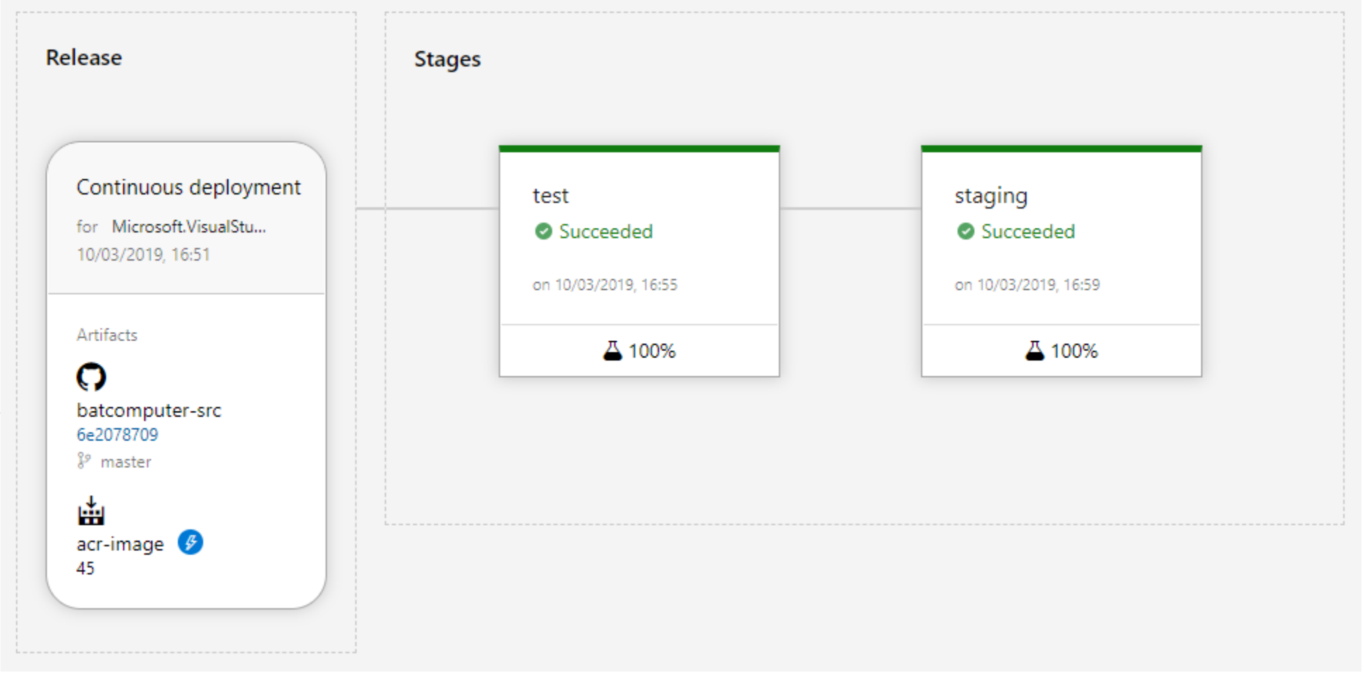
Testing

Integration tests against the real API using Postman & Newman

POSTMAN







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

Machine Learning & Training

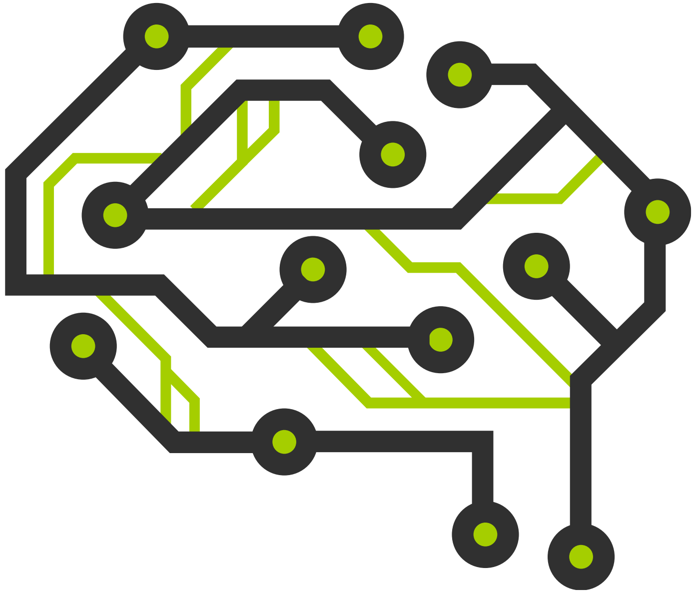
Machine Learning – Training Scripts

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

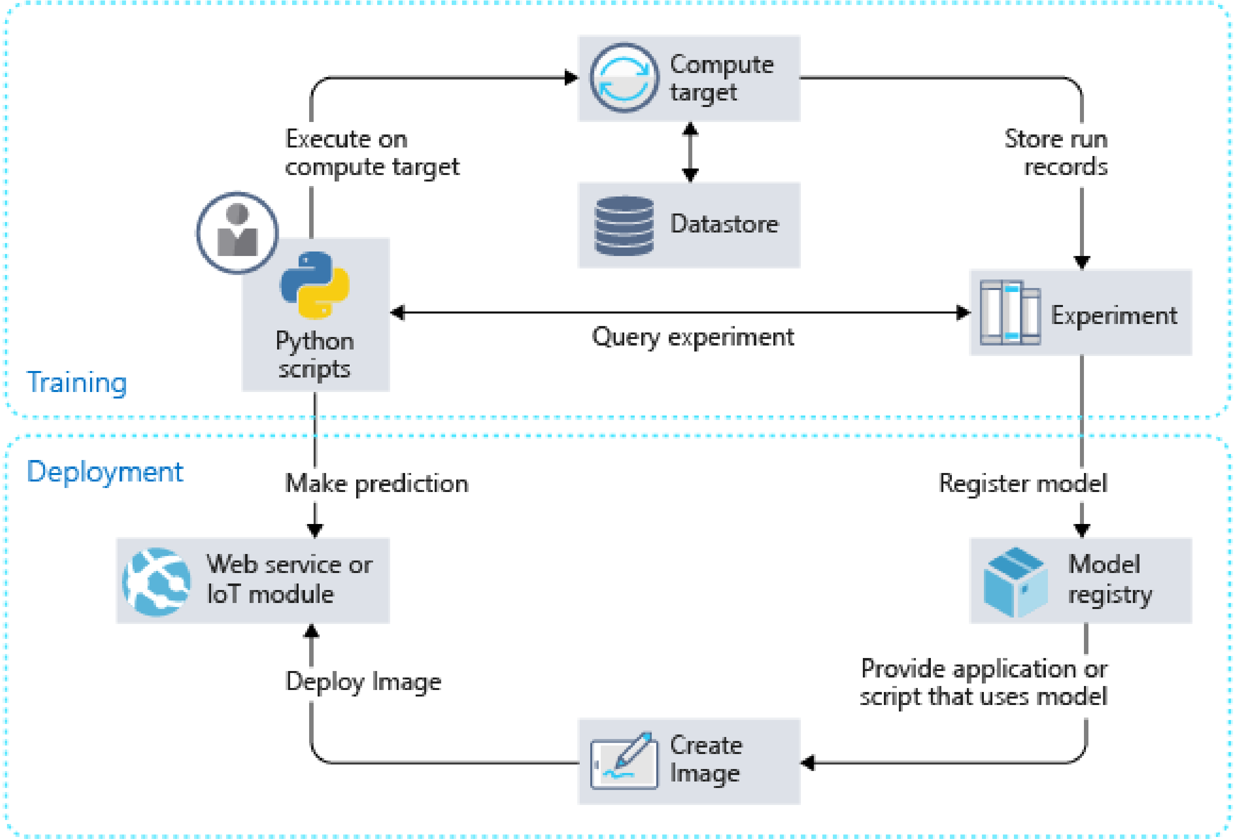
Driven by Python SDK

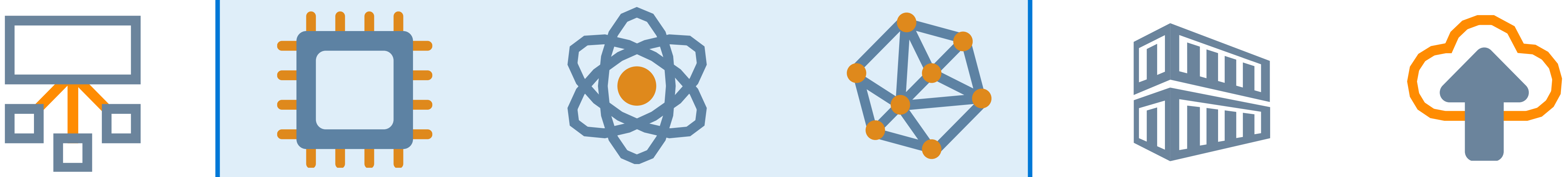
Range of training & experimentation compute targets

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`

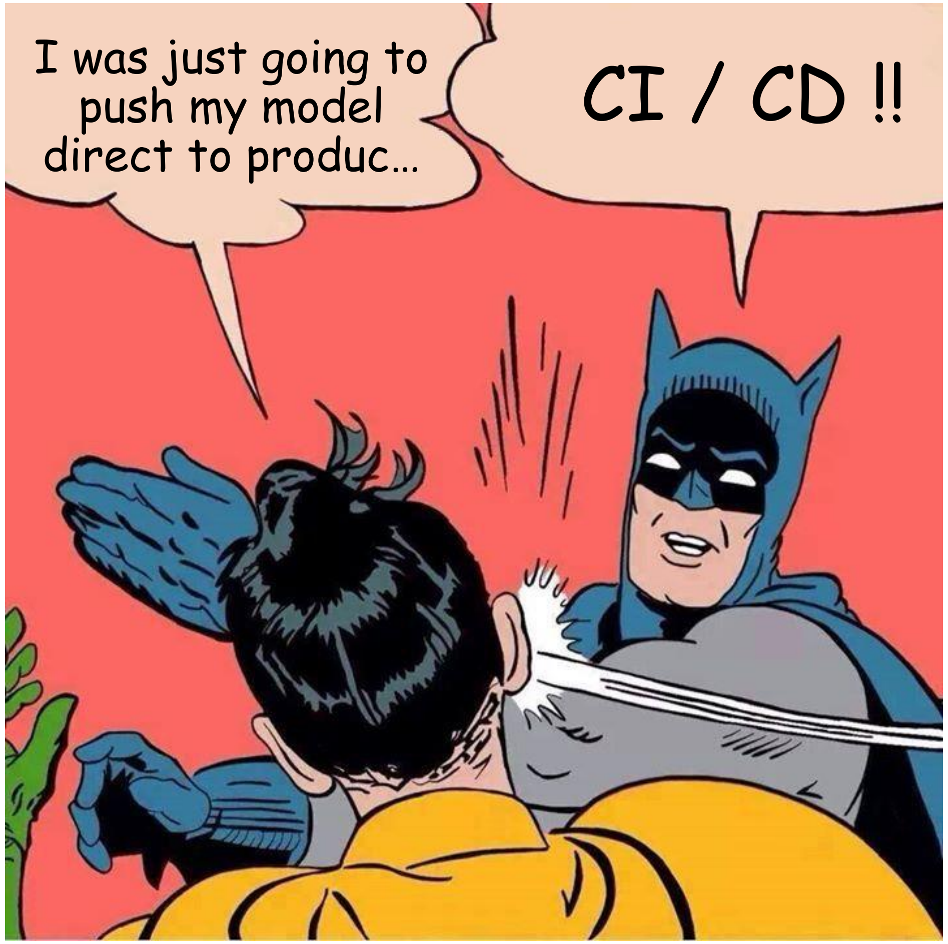
- **Downloads** serialised model from Azure ML **model service**
- In addition gets **supporting .pkl files** (more later)

Azure Machine Learning SDK for Python

Azure ML Deployment

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

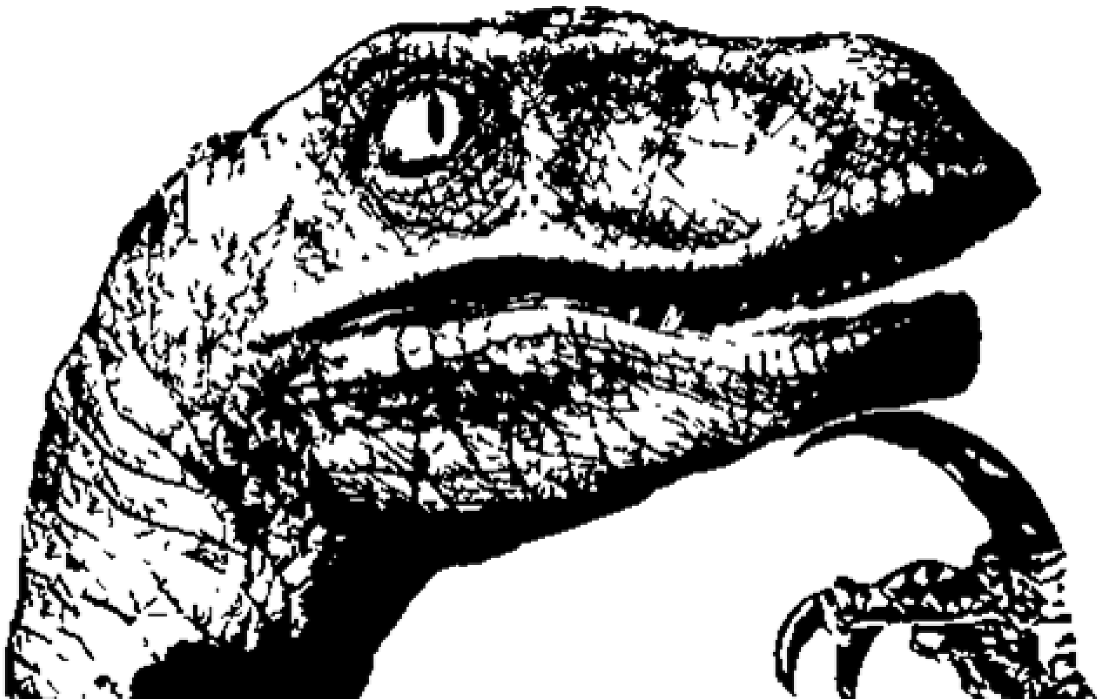
- **"Highly Opinionated"**
- **Bypasses release process**
- **No control over container build process**
- **Limited control of app structure, code or framework**
- **No infrastructure as code or release pipeline**
- **No testing!**



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

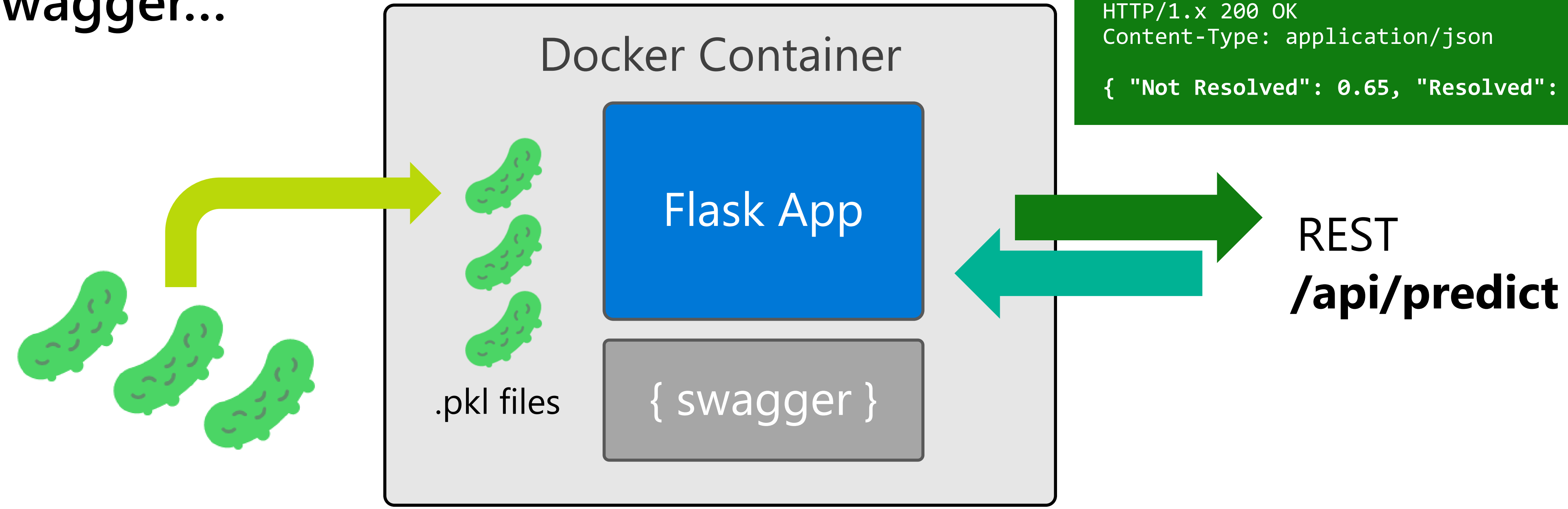
Wrapper App – Components

- 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
}
```

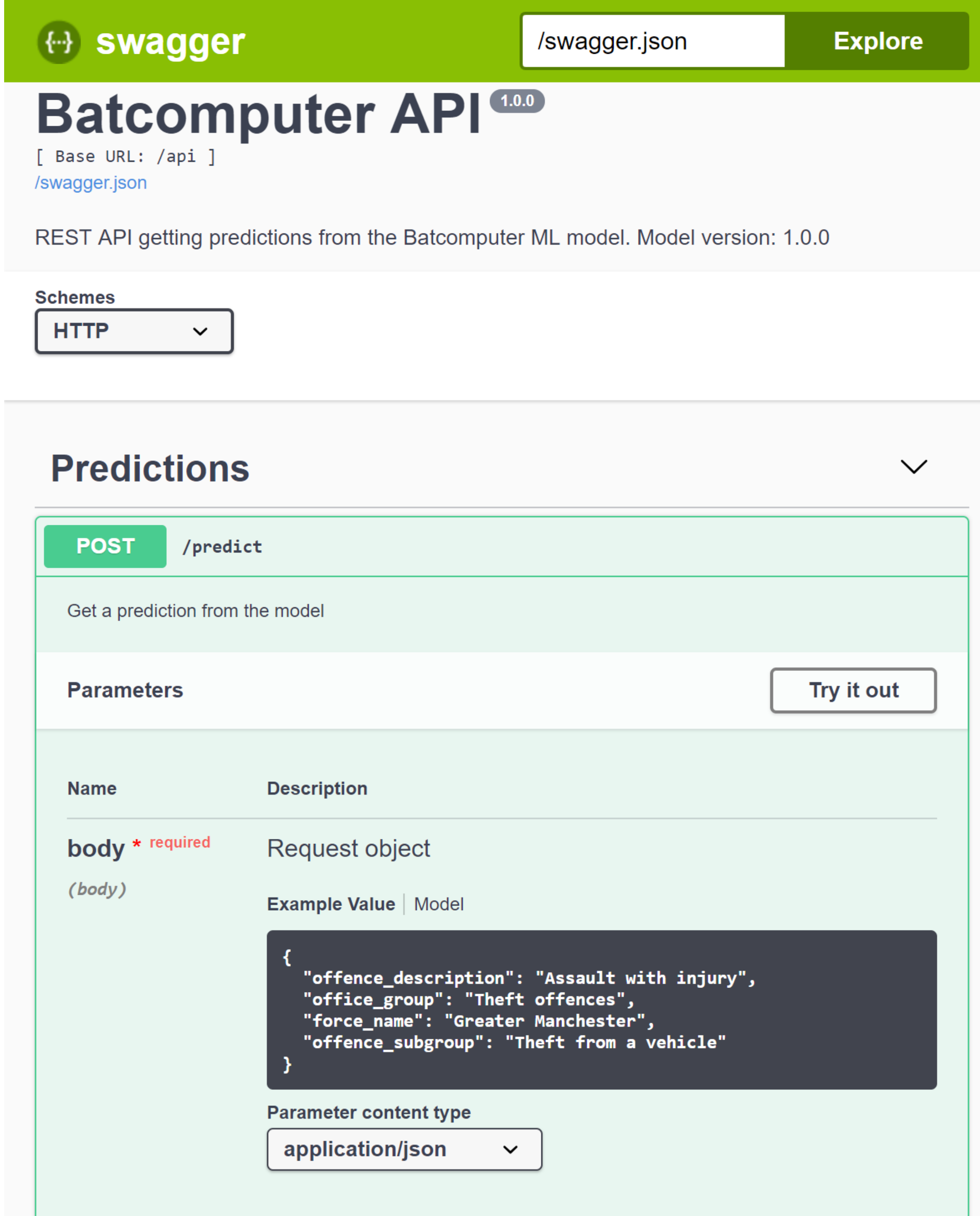
```
HTTP/1.x 200 OK
Content-Type: application/json

{ "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



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

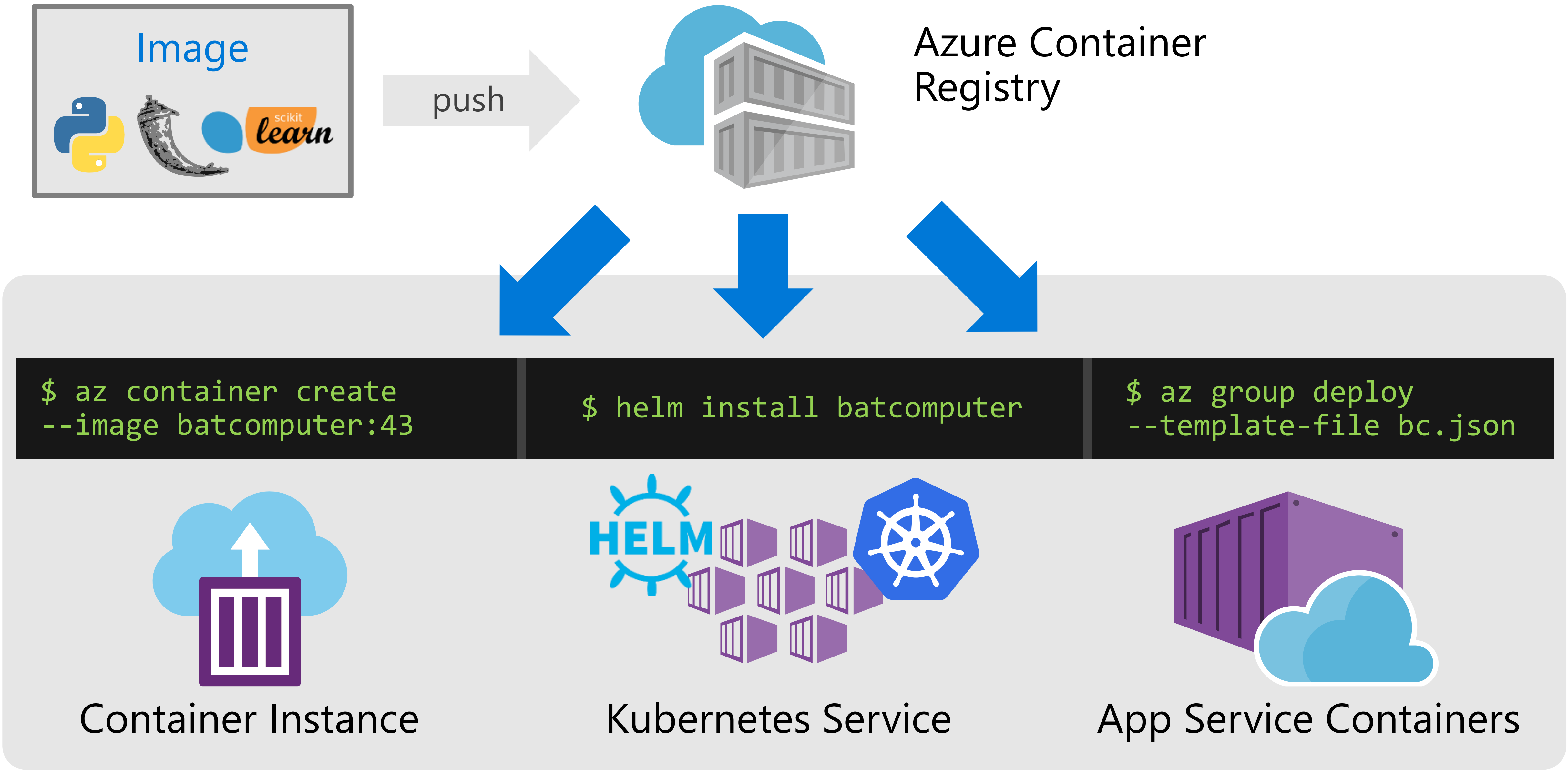
This makes installing Python packages MUCH faster

Add in app source and pickles

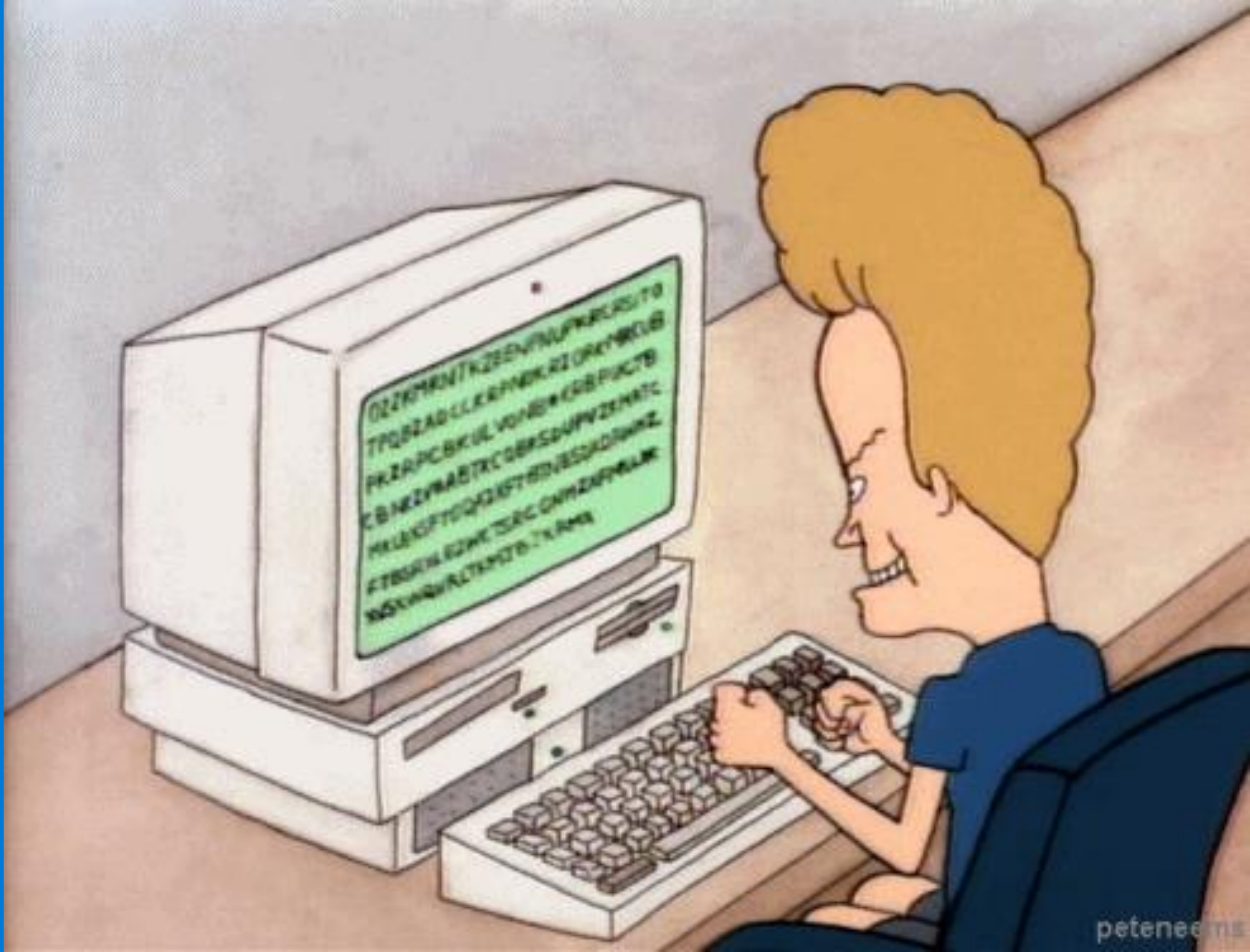
Alternative startup for Gunicorn
Requires no code changes

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

Container Deployment



Demo



Summary

Some Learnings / Gotchas

- Keep versions (Python & Scikit-learn) in sync everywhere, i.e. training vs runtime
- Writing your own wrapper isn't hard
- Azure ML is has a complex but powerful SDK
- Tracking & managing parameters & variables can get tricky
- Don't use Alpine Linux containers, when working with Python

Summary

batcomputer.benco.io

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