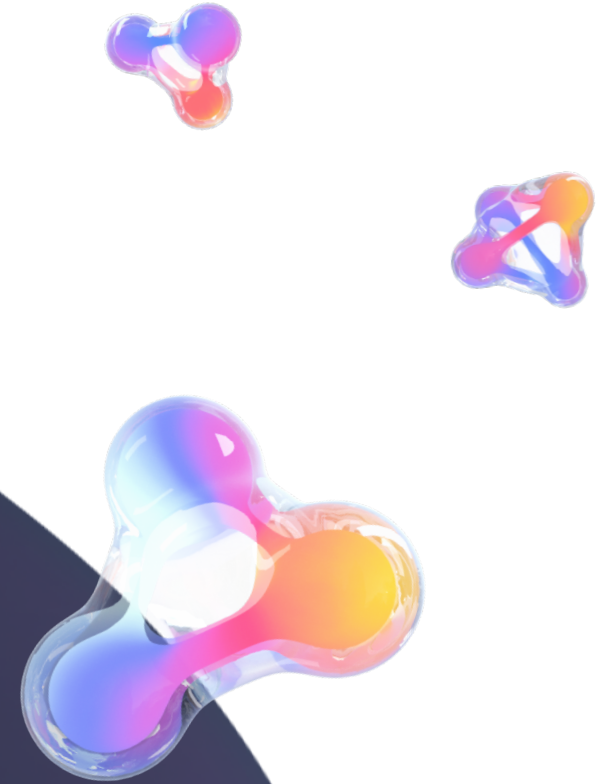




# MLOps Automation with Git Based CI/CD for ML

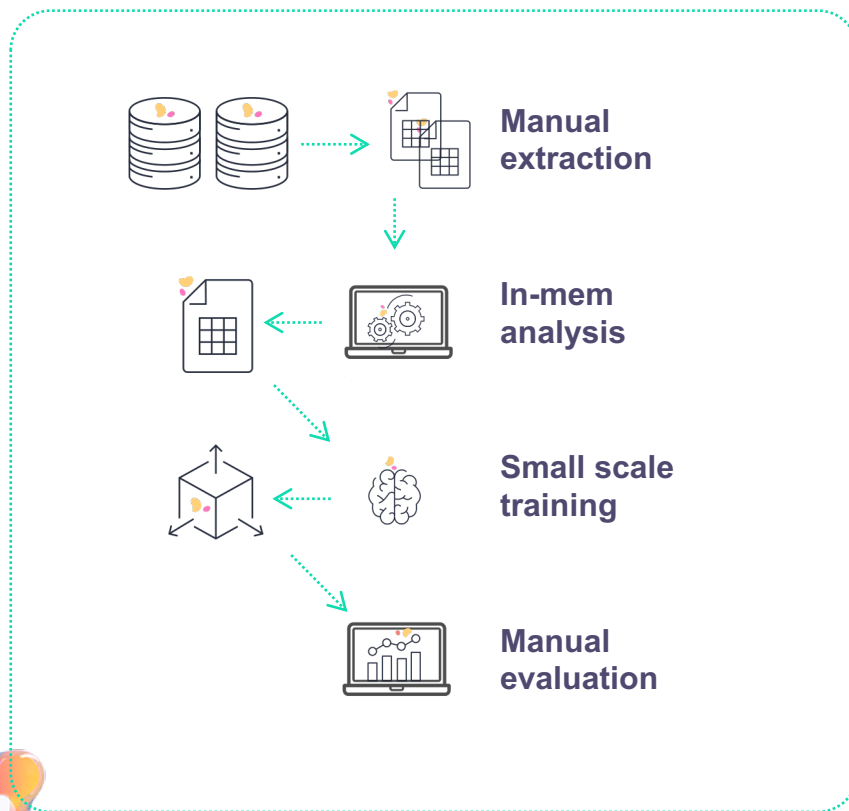
---

Yaron Haviv  
CTO, Iguazio  
[@yaronhaviv](#)



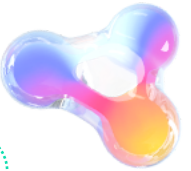
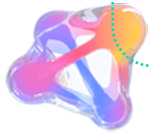
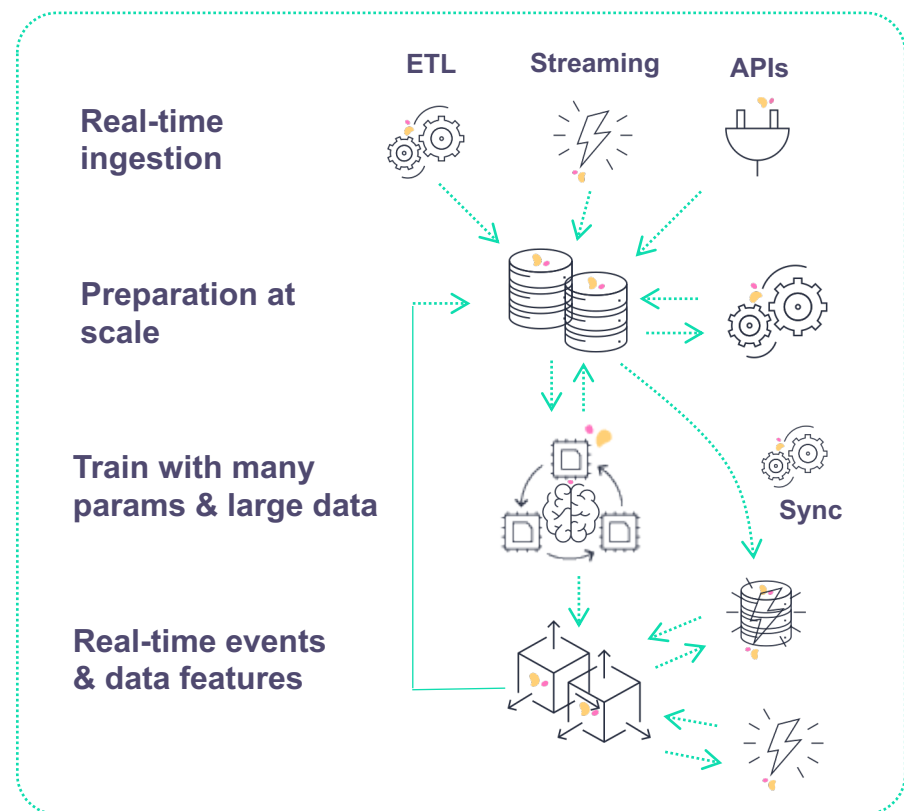
# 80% of AI Projects Never Make it to Production

## Research Environment



Build from Scratch  
with a Large Team

## Production Pipeline



# Did you Try Running Notebooks in Production?



```
get the name from the base folder, currently the Blu SDK does not have an option for finding a folder by name so if you are looking for a specific folder then you would need to loop thru all the items in the list below and do a name match. Once you find the folder and retrieve the id, you can save that id for subsequent runs. I have not found a way to get the id of the folder from the Blu web client.

In [ ]: import os
import shutil

print ('current working directory: ', os.getcwd())
os.chdir('/global/scratch/user_name_here/here')

# find folder contents
items = client.folder(folder_id='').get_items(limit=10, offset=0)
if type(items) is list:
    print ('number of files in top folder: ', len(items))

for item in items:
    if item['type'] == 'folder':
        print('folder name: ', item['name'])
        # download all image files
        if not item['type'] == 'folder' and item['name'].endswith('.jpg'):
            imagecontent = client.file(file_id=item['id']).download()
            newfile = open('/global/scratch/user_name_here/' + item['name'], 'wb')
            newfile.write(imagecontent)
            newfile.close()

Create a new folder in the base directory and upload image files in the current folder.

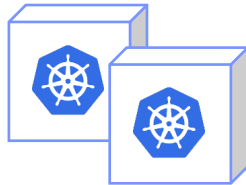
In [ ]: newfolder = client.folder(folder_id='').create_subfolder('thisisthats')
newfolderid = newfolder['id']
newsubfolder = '/global/scratch/user_name_here/'
print ('new folder id: ', newfolderid)

upload_folder = client.folder(folder_id=newfolderid).get()

# upload all the files in the current folder if os.path.isfile()
files = []
for f in os.listdir(newsubfolder):
    print ('files: ', files)

for filename in files:
    print ('file name: ', filename)
    if filename.endswith('.jpg'):
        upload_folder.upload(newsubfolder + filename)
```

Refactor and operationalize



Micro-services



# Model and Code Development are Just the First Step



Develop and Test Locally

**Package**

- Dependencies
- Parameters
- Run scripts
- Build

**Scale-out**

- Load-balance
- Data partitions
- Model distribution
- AutoML

**Tune**

- Parallelism
- GPU support
- Query tuning
- Caching

**Instrument**

- Monitoring
- Logging
- Versioning
- Security

**Automate**

- CI/CD
- Workflows
- Rolling upgrades
- A/B testing

Production



**Weeks**

with **one** data scientist or developer

**Months**

with a **large team** of developers, scientists, data engineers and DevOps



~~DevOps~~

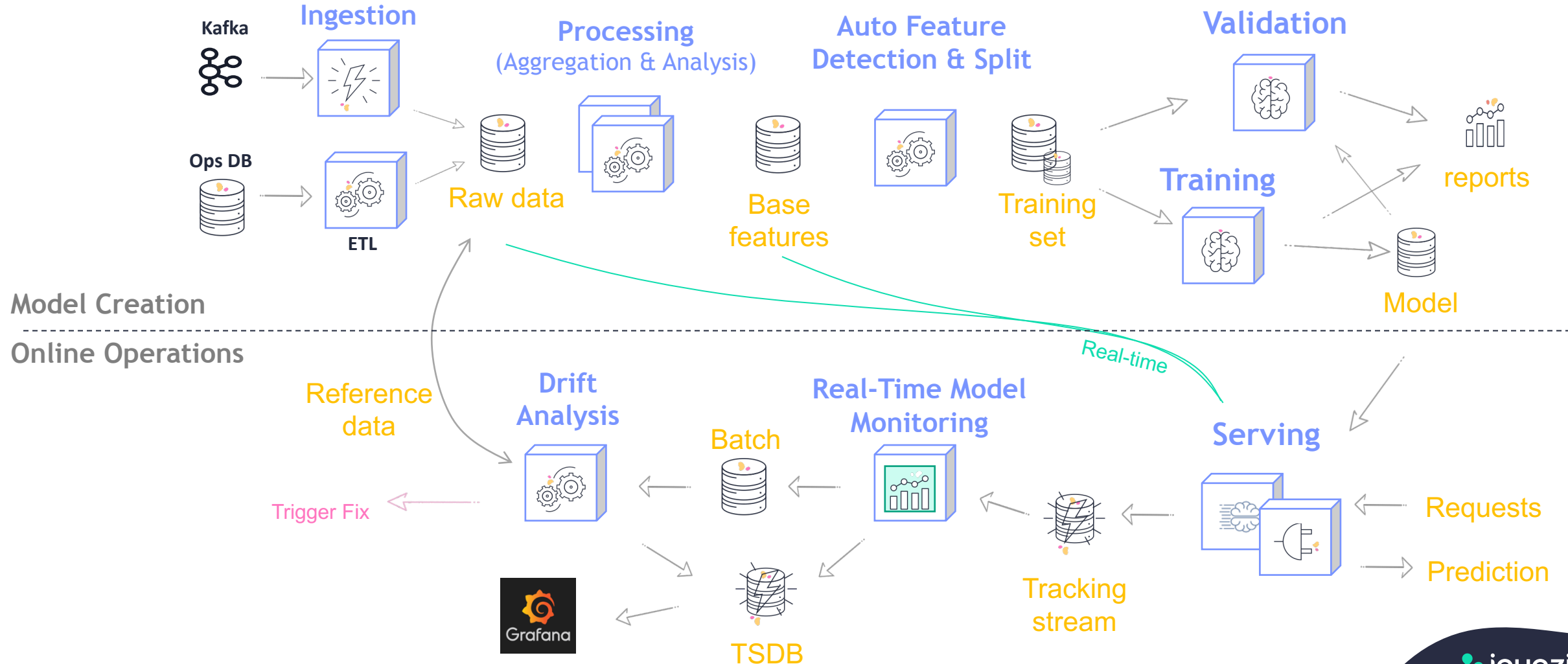
MLOps

*And data-science*

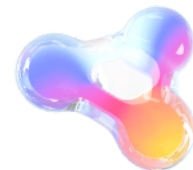
“Combine Dev and Ops to shorten the systems-development life cycle while delivering features, fixes, and updates frequently in close **alignment with business objectives.**”

# Example: Predictive Maintenance Pipeline

<https://github.com/mlrun/demo-network-operations>



# You can use Separate Tools & Services, Or you can Use Kubernetes as the Baseline

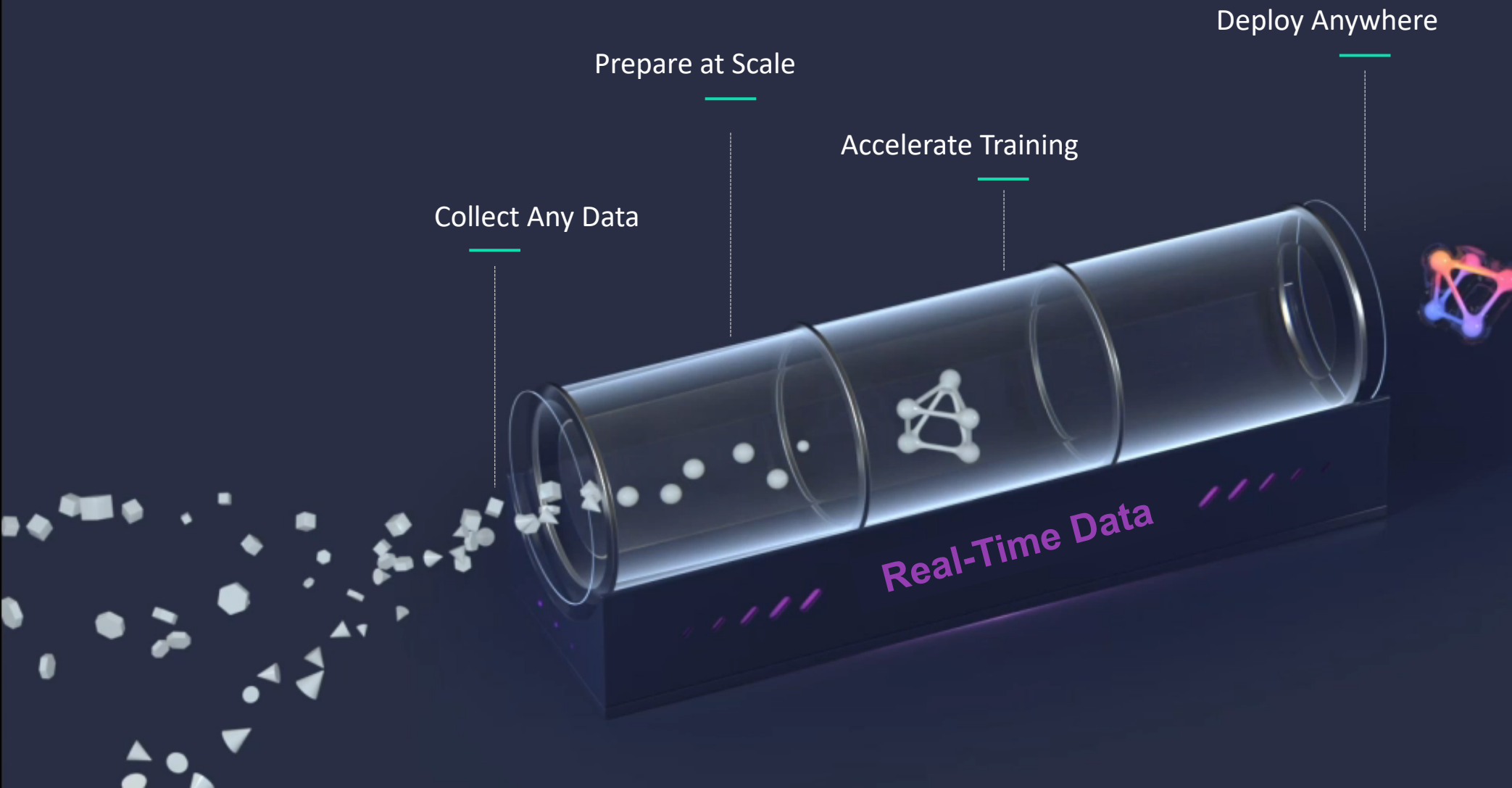


 Versioning	 Labeling
 Workflow	 Database
 Storage	
 Data	

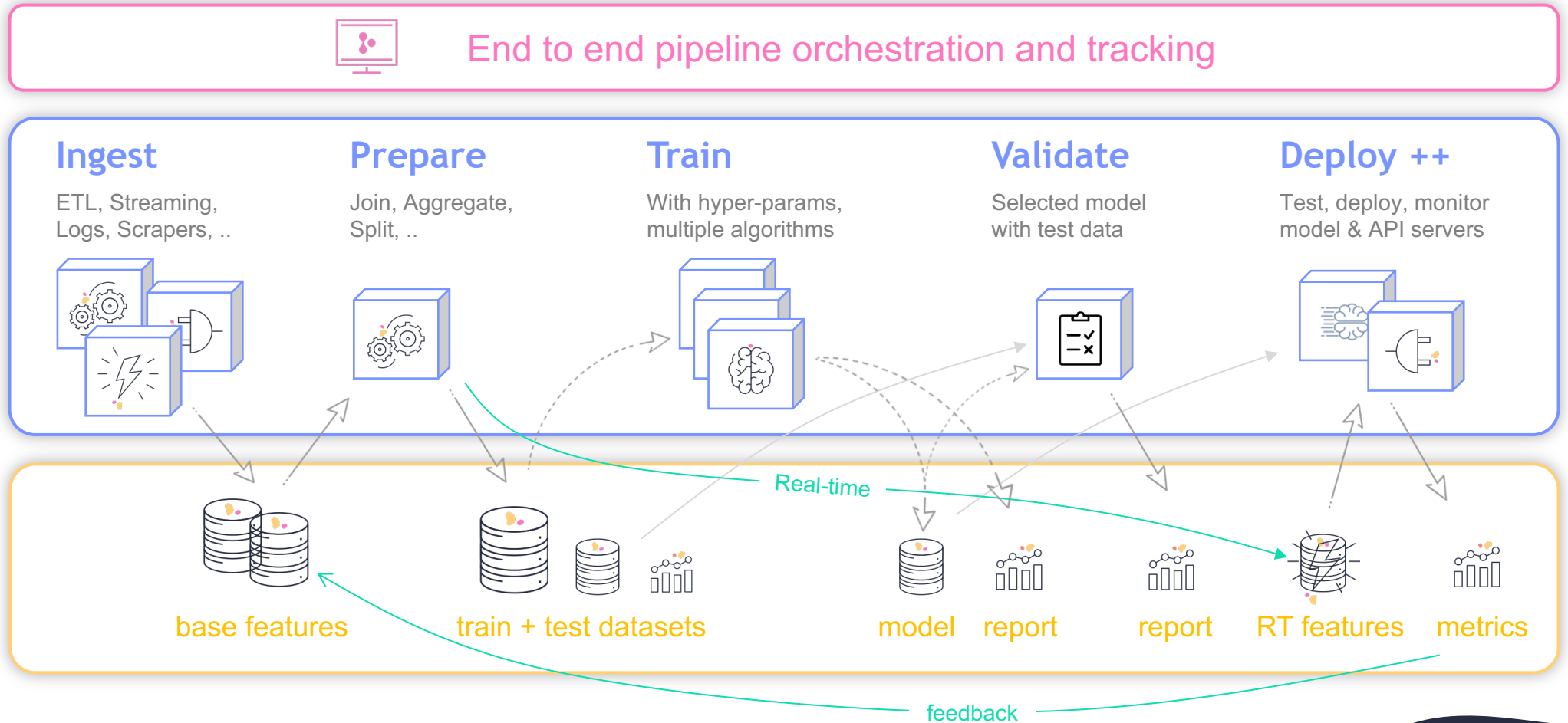
 Distributed Ten Distribu	 ization
 Frameworks	 Experiment Management
 Software Engineering	 Resource Management
 Development	 Training/Evaluation

 Monitoring	 Hardware / Mobile
 Web	 Interchange
 CI / Testing	
 Deployment	

# Bring your AI Applications to Life with Data Science Automation and MLOps



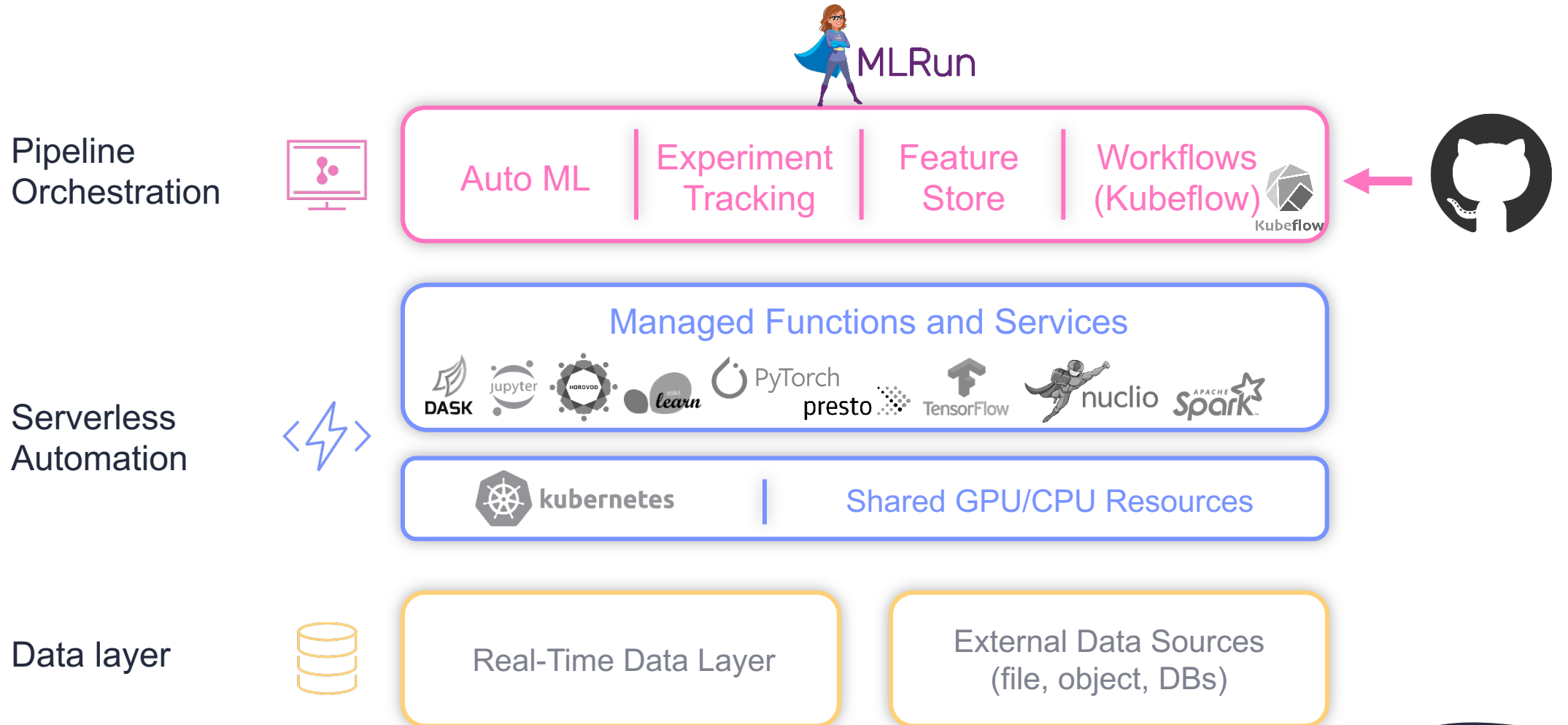
# What is an Automated ML Pipeline ?



**Serverless:**  
ML & Analytics  
Functions

**Features/Data:**  
Fast, Secure,  
Versioned

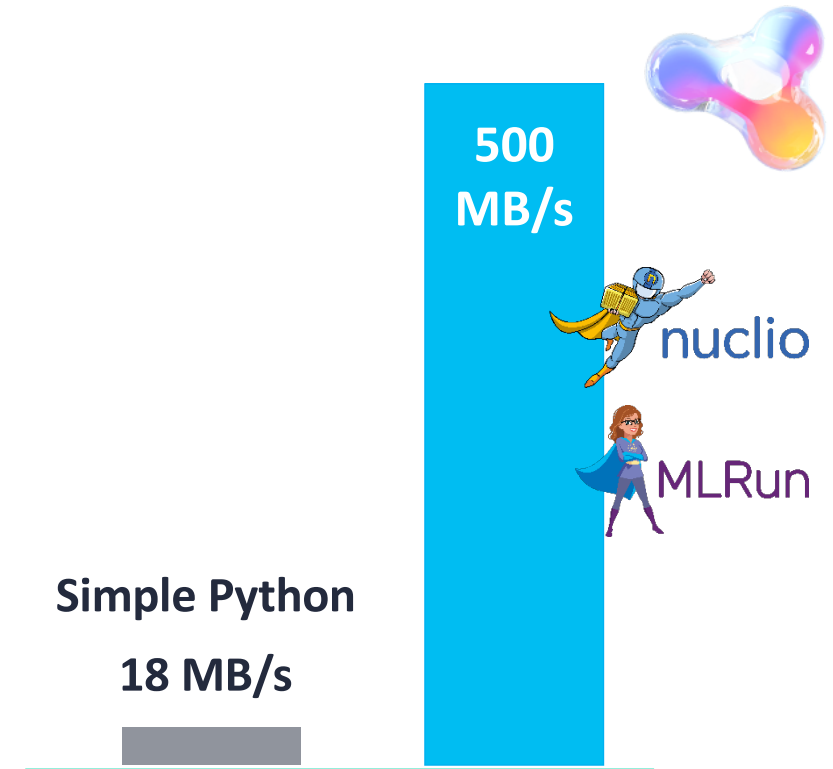
# Under The Hood: Open, Scalable, Production Ready





# Serverless Simplicity, Maximum Performance

- **Automated** code to production
- **Elastic** resource scaling (zero to N)
- **Effortless** logging, monitoring, and versioning
- **High-performance** runtimes + fast data access
- **Glue-less** pipeline and tracking integration
- **Reusable** internal/public function marketplace



”Moving from Hadoop/Java to Serverless reduce 90% of our code footprint and got us much better performance”



# Serverless: Resource Elasticity, Automated Deployment and Operations

*So why not use Serverless for training and data prep?*

	Serverless Today	Data Prep and Training
<b>Task lifespan</b>	Millisecs to mins	Secs to hours
<b>Scaling</b>	Load-balancer	Partition, shuffle, reduce, Hyper-params
<b>State</b>	Stateless	Stateful
<b>Input</b>	Event	Params, Datasets

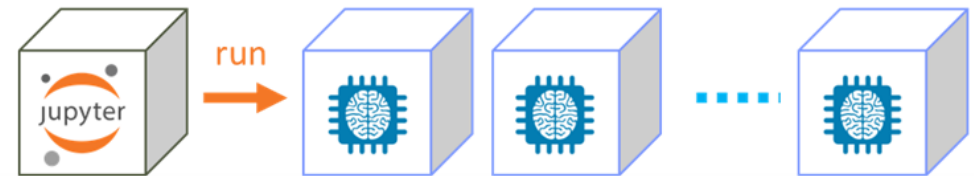
*It's time we extend Serverless to data-science!*

# Dynamic Scaling for Intensive Workloads

- Scale + Performance for intensive ML & Data processing tasks
- Seamless transition from user code to elastic, auto tracked jobs + data
- AutoML & Hyper-params are built-in
- Make frameworks “Serverless”
  - Spark, Dask
  - MPI/Horovod
  - SQL (via Presto)
  - Nuclio



Dynamically Scaled Containers + Distributed Tracking



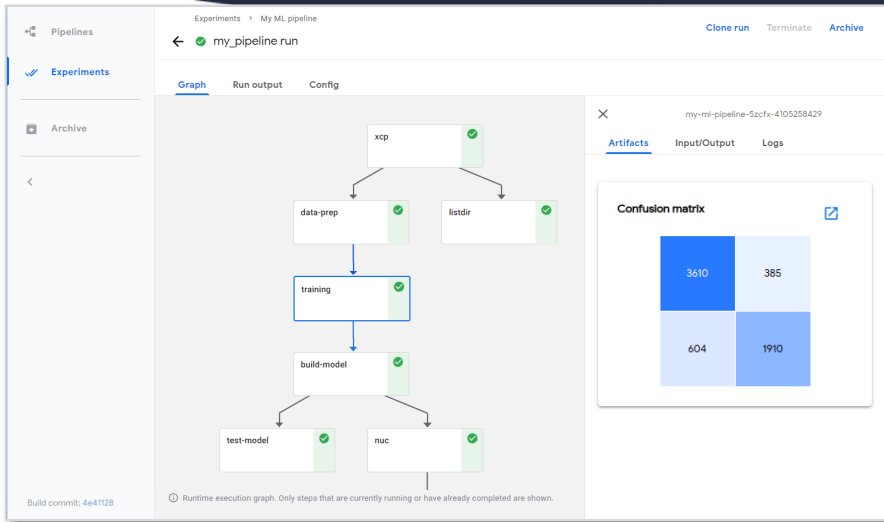
Fast inter cluster messaging (MPI, Dask, Spark, ..)

Low-latency data layer (shared code, files, dataframes)

<https://github.com/mlrun/mlrun>



# KubeFlow: Automated ML Pipelines & Tracking



## Integrating and Extending KubeFlow Pipelines



Manage experiments, runs, and artifacts

Build workflows using code or reusable components  
(across many cloud/3<sup>rd</sup> party ML and data framework)

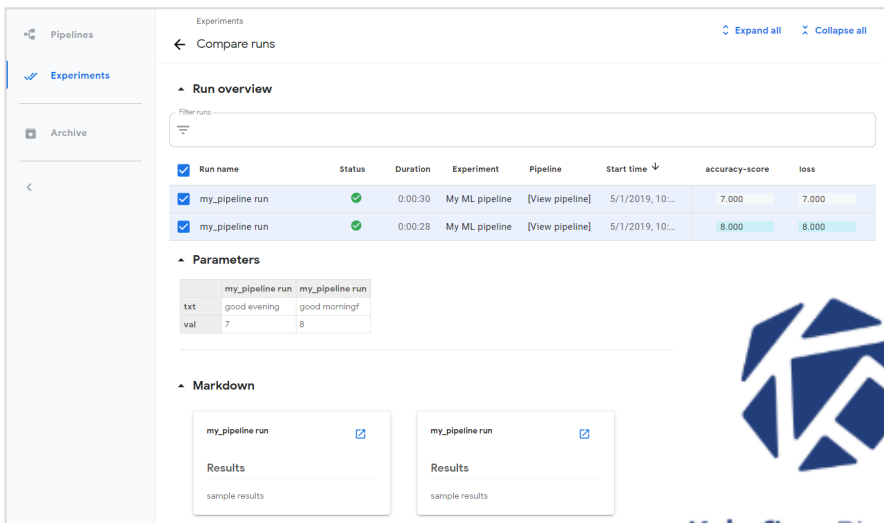
## With MLRun & Nuclio:

Automated code to serverless function

Glueless data access and parallelism

Distributed training and GPU Acceleration

Code + execution + data tracking and versioning



# Simple, Production-Ready Development Process

## Data-Scientist / Developer

1. Write and test functions locally



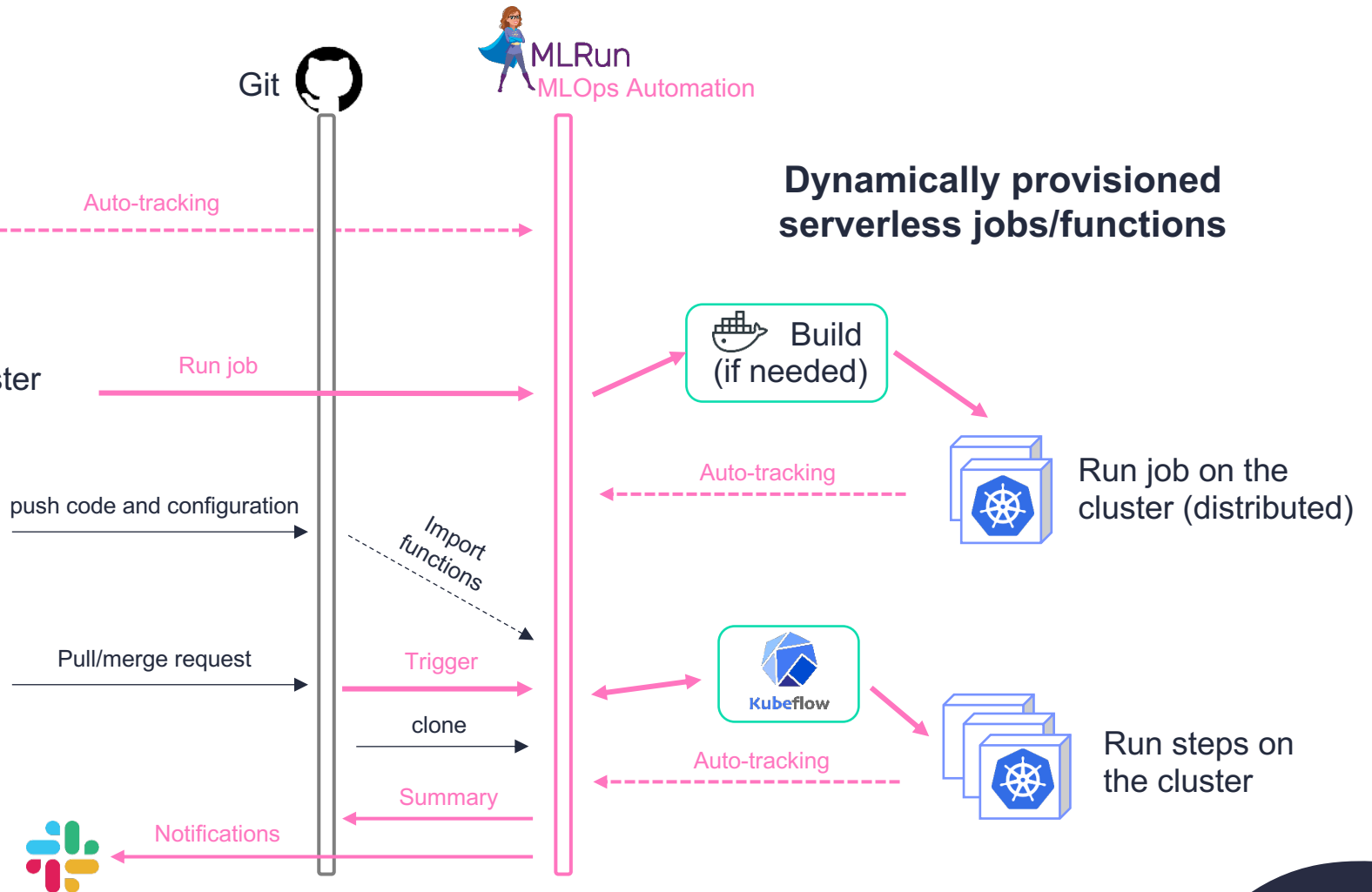
2. Add requirements, run on the cluster

Specify image, packages, cpu/gpu/mem, data, ..  
Requirements via annotation or function spec

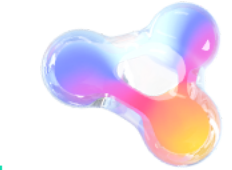
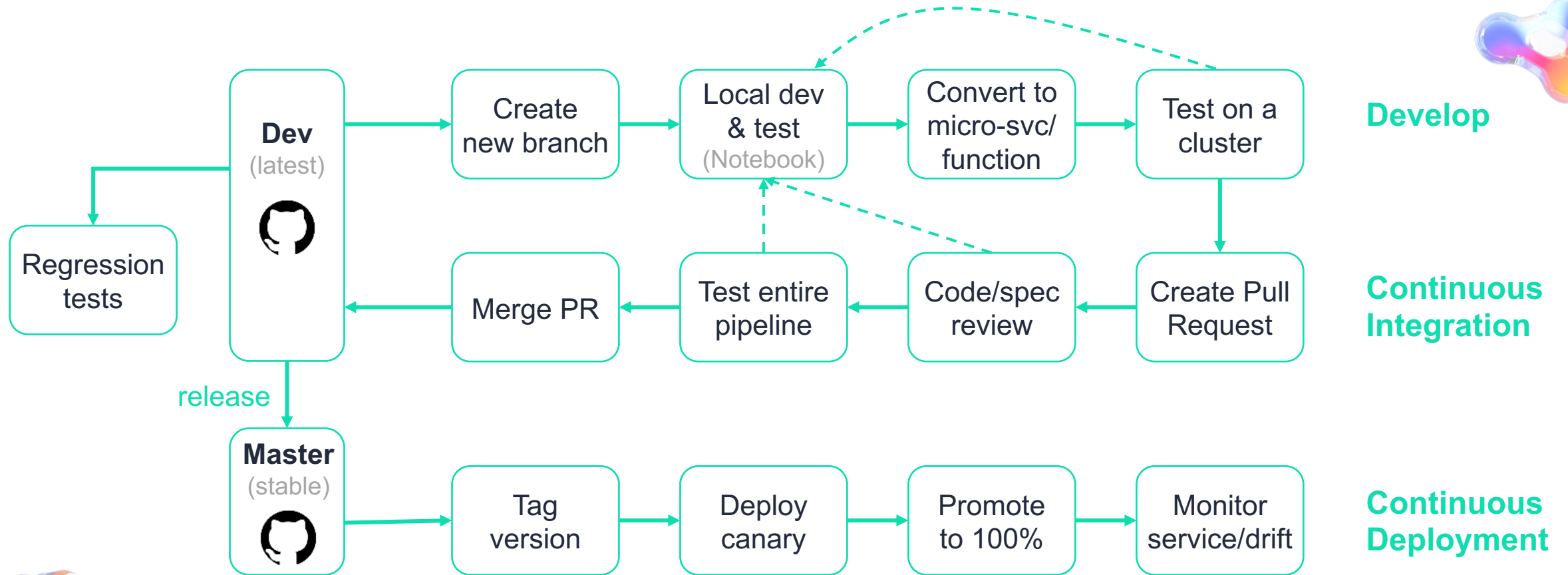
```
%nuclio cmd -c pip install pandas  
%nuclio config spec.build.baseImage = "mlrun/mlrun"
```

## ML Engineer

3. Build/run ML pipeline  
(interactive or via triggers)



# Building CI/CD Process for ML(Ops)



Develop

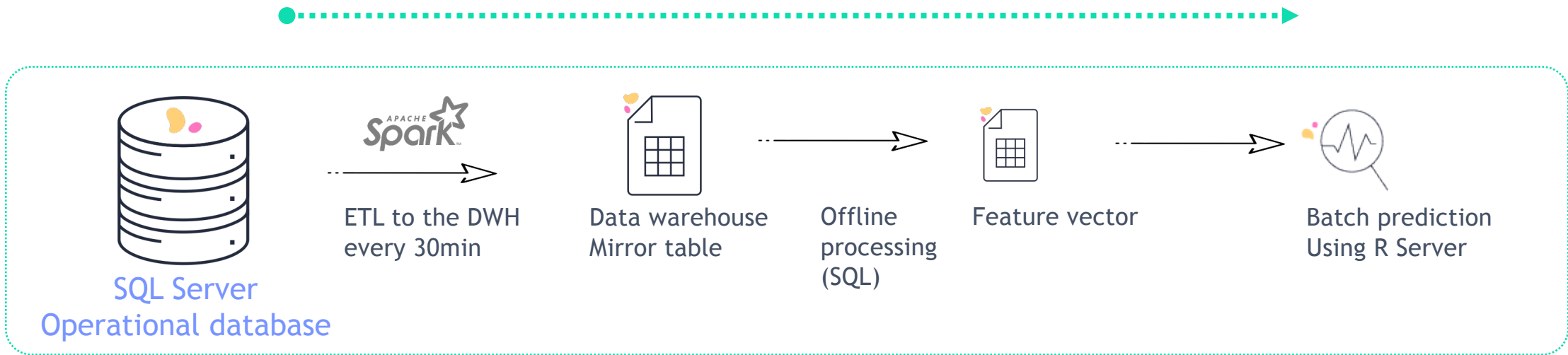
Continuous Integration

Continuous Deployment



# Traditional Fraud-Detection Architecture (Hadoop)

40 Precious Minutes (detect fraud after the fact)

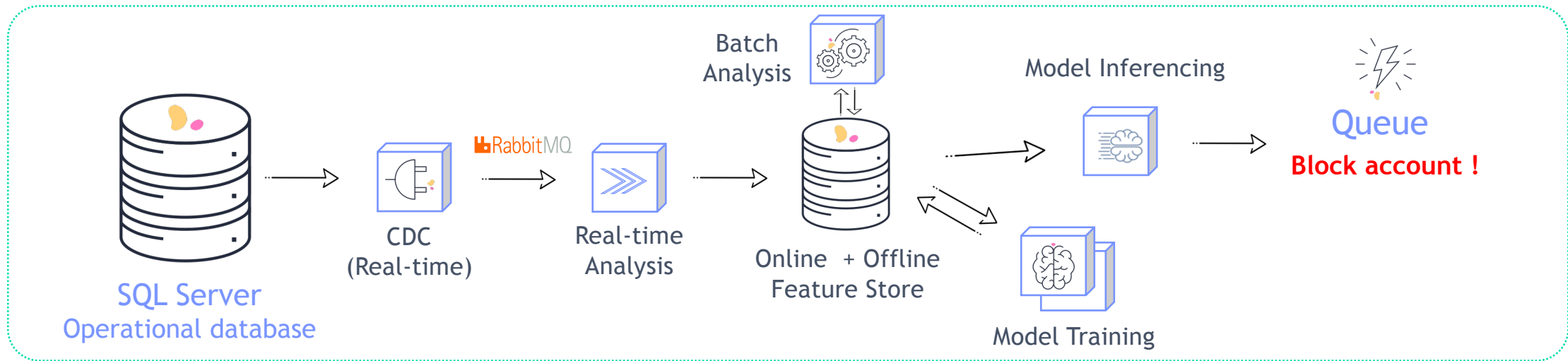


40 minutes to identify suspicious money laundering account

Long and complex process to production

# Real-Time Fraud Prediction & Prevention

12 Seconds (prevent fraud)



**12 Seconds to detect and prevent fraud !**

**Automated dev to production using a serverless approach**

# DEMO



Start creating real-world business  
impact with AI, today

---

[www.iguazio.com](http://www.iguazio.com)

