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:
- Manual extraction
- In-mem analysis
- Small scale training
- Manual evaluation

Production Pipeline:
- Real-time ingestion
- Preparation at scale
- Train with many params & large data
- Real-time events & data features

Build from Scratch with a Large Team
Did you Try Running Notebooks in Production?

Refactor and operationalize

WORKED FINE IN JUPYTER

NOW ITS OPS PROBLEM
Model and Code Development are Just the First Step

Develop and Test Locally
- Dependencies
- Parameters
- Run scripts
- Build

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

Scale-out
- Parallelism
- GPU support
- Query tuning
- Caching

Tune
- Monitoring
- Logging
- Versioning
- Security

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

Automate

Weeks
with one data scientist or developer

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

Production
DevOps

MLOps

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

And data-science
Example: Predictive Maintenance Pipeline

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

- **Ingestion**
  - Raw data
  - ETL
- **Processing** (Aggregation & Analysis)
  - Base features
- **Auto Feature Detection & Split**
  - Training set
- **Validation**
  - Model
- **Training**
- **Serving**
  - Requests
  - Prediction
- **Drift Analysis**
  - Reference data
- **Real-Time Model Monitoring**
  - Batch
  - Tracking stream
  - TSDB
- **Online Operations**
  - Kafka
  - Ops DB
- **Model Creation**
  - Auto Feature Detection & Split
  - Validation
  - Training
  - Serving

---

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

- Collect Any Data
- Prepare at Scale
- Accelerate Training
- Deploy Anywhere

Real-Time Data
What is an Automated ML Pipeline?

End to end pipeline orchestration and tracking

**Serverless:**
ML & Analytics Functions

**Ingest**
ETL, Streaming, Logs, Scrapers, ..

**Prepare**
Join, Aggregate, Split, ..

**Train**
With hyper-params, multiple algorithms

**Validate**
Selected model with test data

**Deploy ++**
Test, deploy, monitor model & API servers

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

- base features
- train + test datasets
- model
- report
- RT features
- metrics

Real-time feedback
Under The Hood:
Open, Scalable, Production Ready

Auto ML | Experiment Tracking | Feature Store | Workflows (Kubeflow)

Pipeline Orchestration

Managed Functions and Services
- DASK
- jupyter
- Apache
- PyTorch
- Presto
- TensorFlow
- Nuclio
- Spark

Serverless Automation

Shared GPU/CPU Resources

Data layer

Real-Time Data Layer

External Data Sources (file, object, DBs)
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?

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

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

MLRun

Fast inter cluster messaging (MPI, Dask, Spark, ..)
Low-latency data layer (shared code, files, dataframes)

https://github.com/mlrun/mlrun

13
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/3rd 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

1. Write and test functions locally
   - Write code in Jupyter Notebook
   - Test locally

2. Add requirements, run on the cluster
   - Add requirements via annotation or function spec
   - Push code and configuration
   - Build/run ML pipeline (interactive or via triggers)

3. Build/run ML pipeline (interactive or via triggers)
   - Pull/merge request
   - Build (if needed)
   - Run job on the cluster (distributed)

Data-Scientist / Developer

ML Engineer

Dynamically provisioned serverless jobs/functions
Building CI/CD Process for ML(Ops)

1. **Dev (latest)**
   - Create new branch
   - Local dev & test (Notebook)
   - Convert to micro-svc/function
   - Test on a cluster

2. **Master (stable)**
   - Merge PR
   - Test entire pipeline
   - Create Pull Request
   - Code/spec review

3. **Continuous Integration**
   - Tag version
   - Deploy canary
   - Promote to 100%
   - Monitor service/drift

4. **Continuous Deployment**
   - Regression tests
   - Release
Traditional Fraud-Detection Architecture (Hadoop)

40 Precious Minutes (detect fraud after the fact)

ETL to the DWH every 30min → Data warehouse Mirror table → Offline processing (SQL) → Feature vector → Batch prediction Using R Server

40 minutes to identify suspicious money laundering account

Long and complex process to production
Real-Time Fraud Prediction & Prevention

12 Seconds (prevent fraud)

SQL Server
Operational database

CDC
(Real-time)

Real-time Analysis

Batch Analysis

Online + Offline Feature Store

Model Training

Model Inferencing

Queue

Block account!

12 Seconds to detect and prevent fraud!
Automated dev to production using a serverless approach
Start creating real-world business impact with AI, today

www.iguazio.com