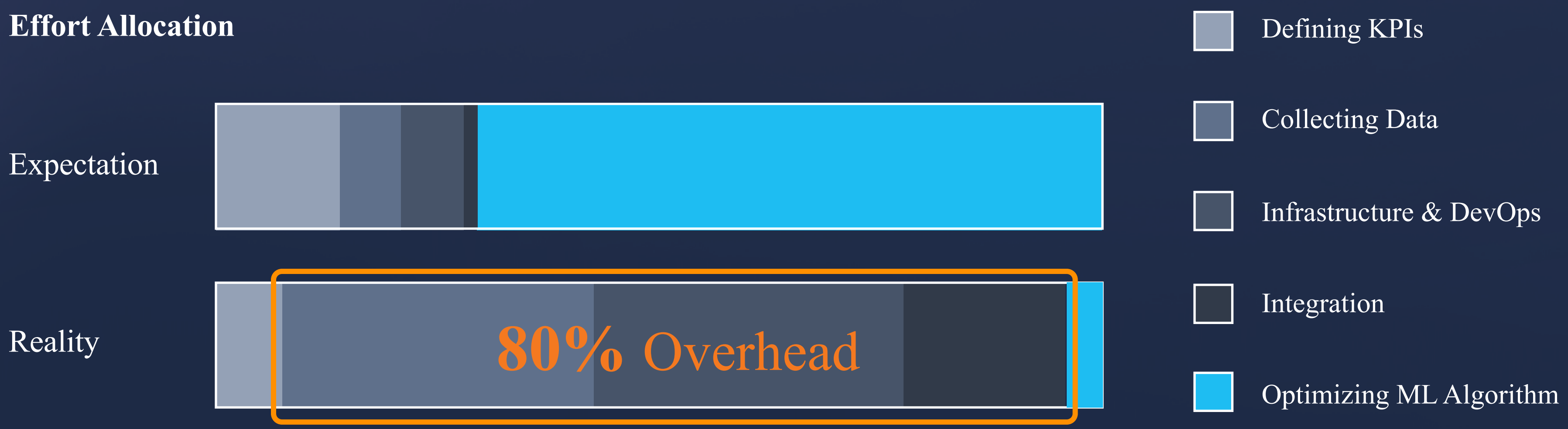
iguazio

Nuclio : KubeFlow Serverless's component

Orit Nissan-Messing, VP R&D, Iguazio

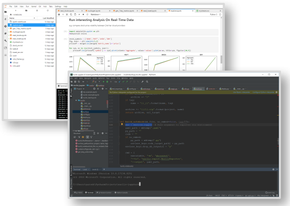
Data Science Teams Don't Do Data Science



Source: Google Developers Launchpad

The need: Simpler Solutions, Better Data Integration

Develop/Experiment



Package

- Dependencies
- Parameters
- Run scripts
- Build

Scale-out

- Load-balance
- Data partitions
- Model distribution
- Hyper params

Tune

- Parallelism
- GPU support
- Query tuning
- Caching

Instrument

- Monitoring
- Logging
- Versioning
- Security

Automate

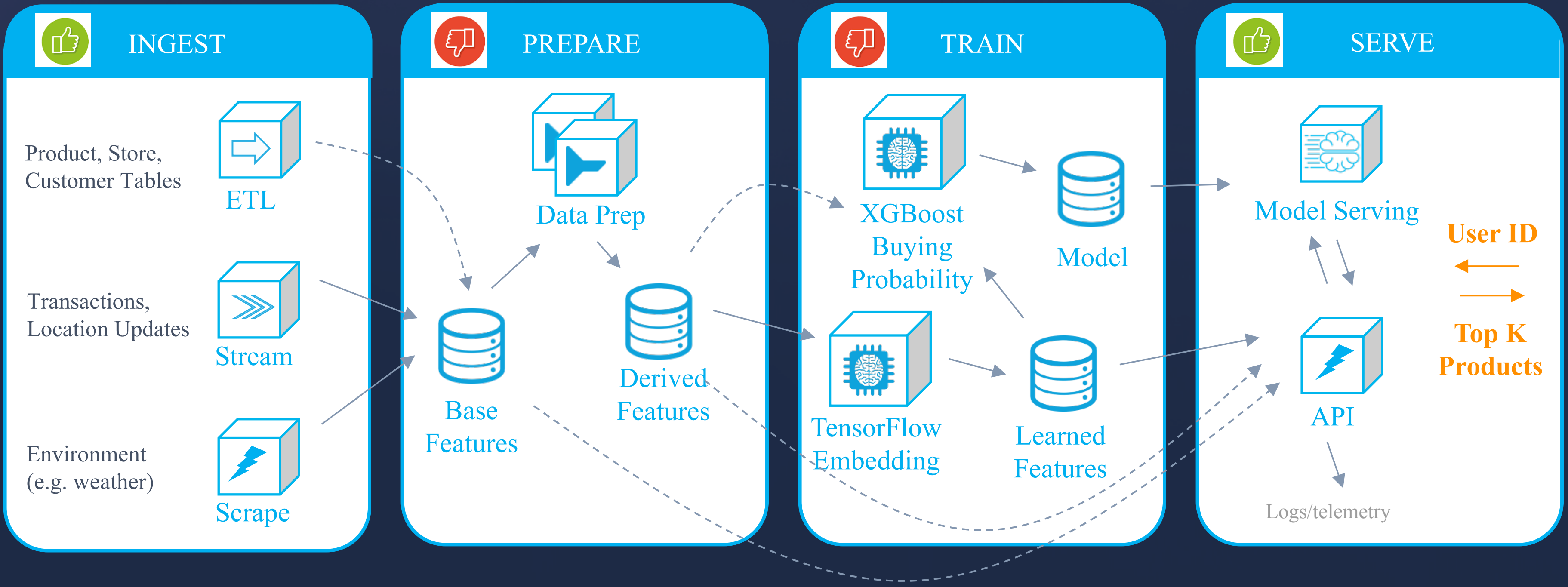
- CI/CD
- Workflows
- Rolling upgrades
- A/B testing

Weeks with one data scientist

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

Automate DevOps to Deploy Projects in One Week as Opposed to Months!

Example: Real-time Product Recommendations



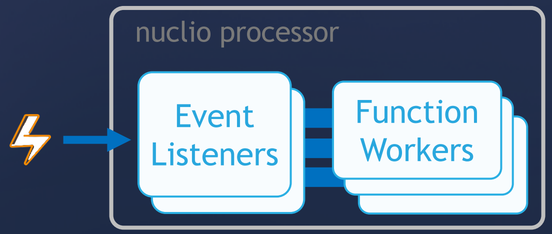
ML Function

Feature/Model Store

iguazio

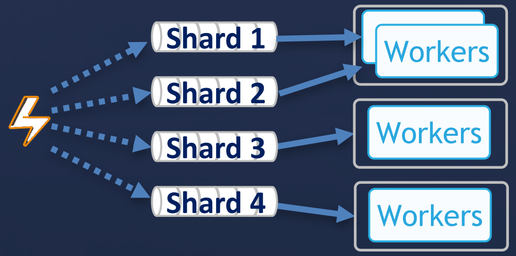
Nuclio: Taking Serverless to Data Intensive Apps

Extreme Performance



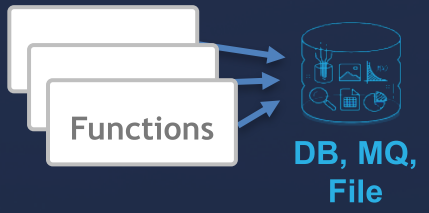
- Non-blocking, parallel
- Zero copy, buffer reuse
- Up to 400K events/sec/proc
- **GPU** optimizations

Advanced Data & AI Features



- Auto-rebalance, checkpoints
- Any source: Kafka, NATS, Kinesis, event-hub, iguazio, pub/sub, RabbitMQ, Cron, ..
- NVIDIA Rapids integration

Statefulness



- Data bindings
- Shared volumes
- Context cache

Natively integrated with Kubeflow and Jupyter Notebooks

Ingest: Using Nuclio to Accelerate ETL and Streaming

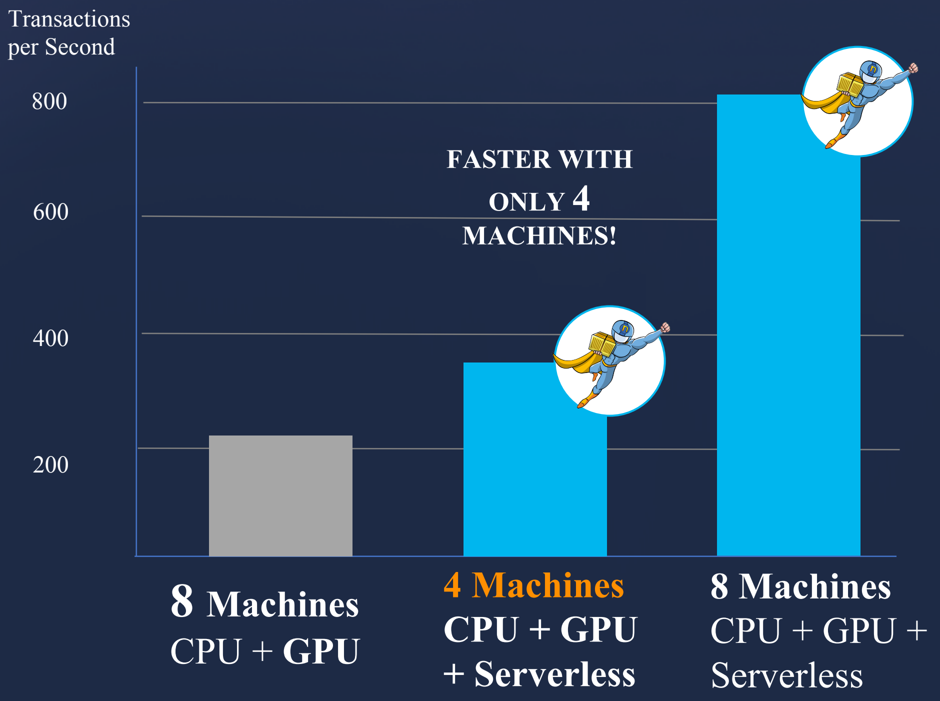
Simple code! Automated DevOps ! Any Source!

(e.g. read JSON Stream + aggregate + dump to Parquet)

```python
def init_context(context):
    os.makedirs(sink, exist_ok=True)

def handler(context, event):
    add_log_to_batch(context, event.body)

    if len(batch) > batch_len:
        df = _batch_to_df(context)
        if not df.empty:
            df = df.groupby(['log_ip']).agg({'feconn':'mean',
                                             'beconn':'mean',
                                             'time_backend_response':'max',
                                             'time_backend_response':'mean',
                                             'time_queue':'mean',
                                             'time_duration': 'mean',
                                             'time_request': 'mean',
                                             'time_backend_connect':'mean'
                                            })
        df_to_parquet(df)
        reset_batch()
```

Serving: Using Nuclio for Real-time Model Serving

4X Faster model serving on GPU system



Single command from notebook to function

iguazio

Why Not Use Serverless for Training and Data Prep?

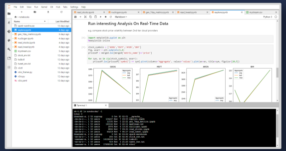
What about Training and data prep ?

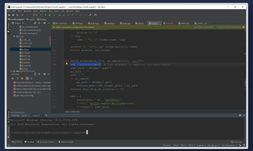
	Serverless Today	Data Prep and Training
Task lifespan	Millisecs to mins	Secs to hours
Scaling	Load-balancer	Partition, shuffle, reduce, Hyper-params, ring allreduce
State	Stateless	Stateful
Input	Event	Params, Datasets

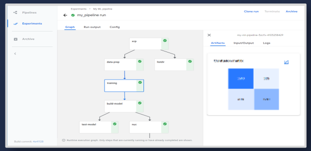
Serverless: resource elasticity and automated deployment and operations

Introducing Nuclio ML Functions

Access from your notebook, IDE, or KubeFlow







Kubeflow

Common APIs & Automation

Multiple Engines

Application & Notebook Templates

Builder | Controller | Runs & Artifacts DB

Real-time | Analytics | Training | Serving

Spark

+ KFS

Fast Persistent Data Layer (Files, DataFrames, K/V)

Built-in Artifacts & Runs Tracking

Elastic Scaling

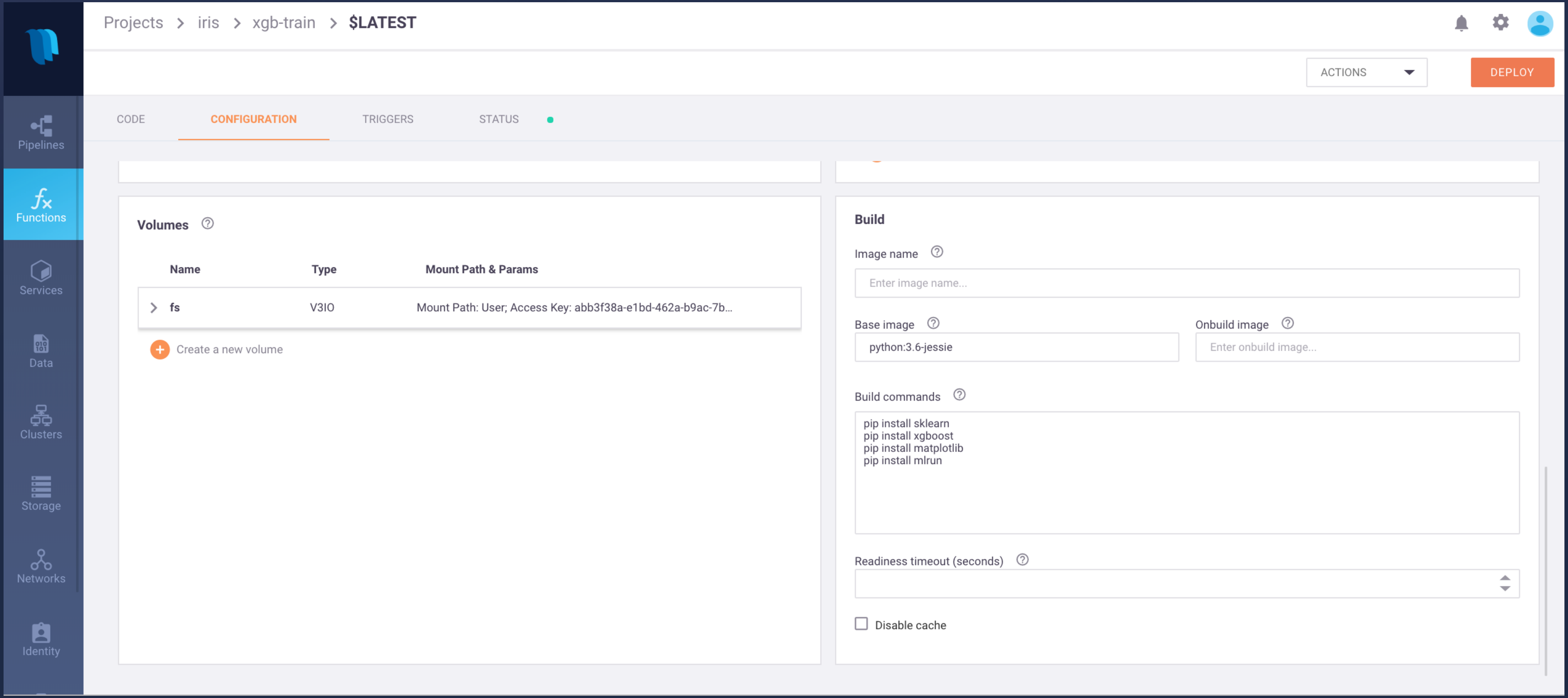
Demo: Fast and Serverless KubeFlow Pipeline

All demos can be found in github: https://github.com/mlrun/demos

Thank You

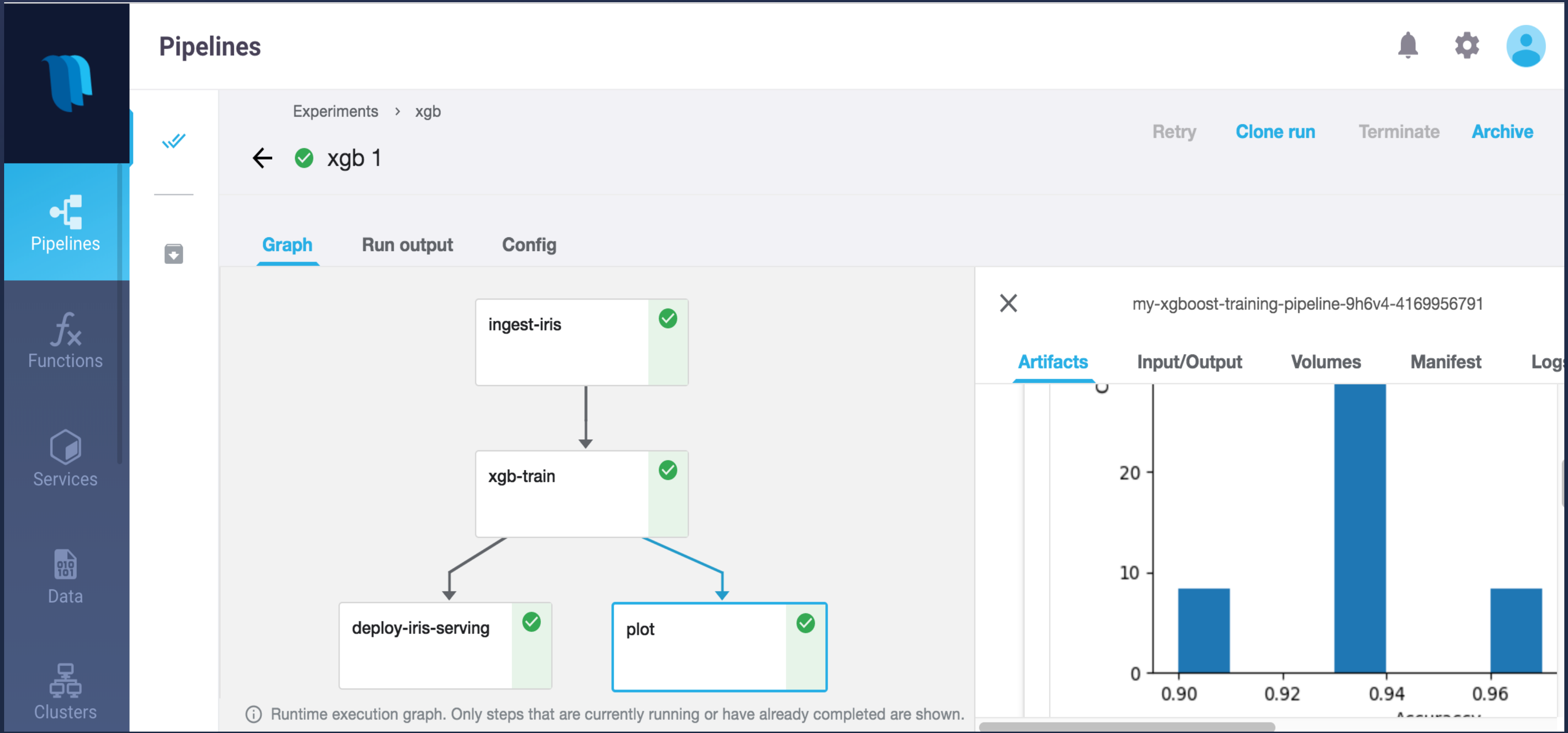
oritn@iguazio.com, www.iguazio.com

Iris Model Nuclio Function



iguazio

Iris Model Pipeline



Iris Model Serving

Projects > iris > iris-srv > $LATEST

CODE CONFIGURATION TRIGGERS STATUS

Code entry type: Source code (edit online)

Runtime: Python 3.6

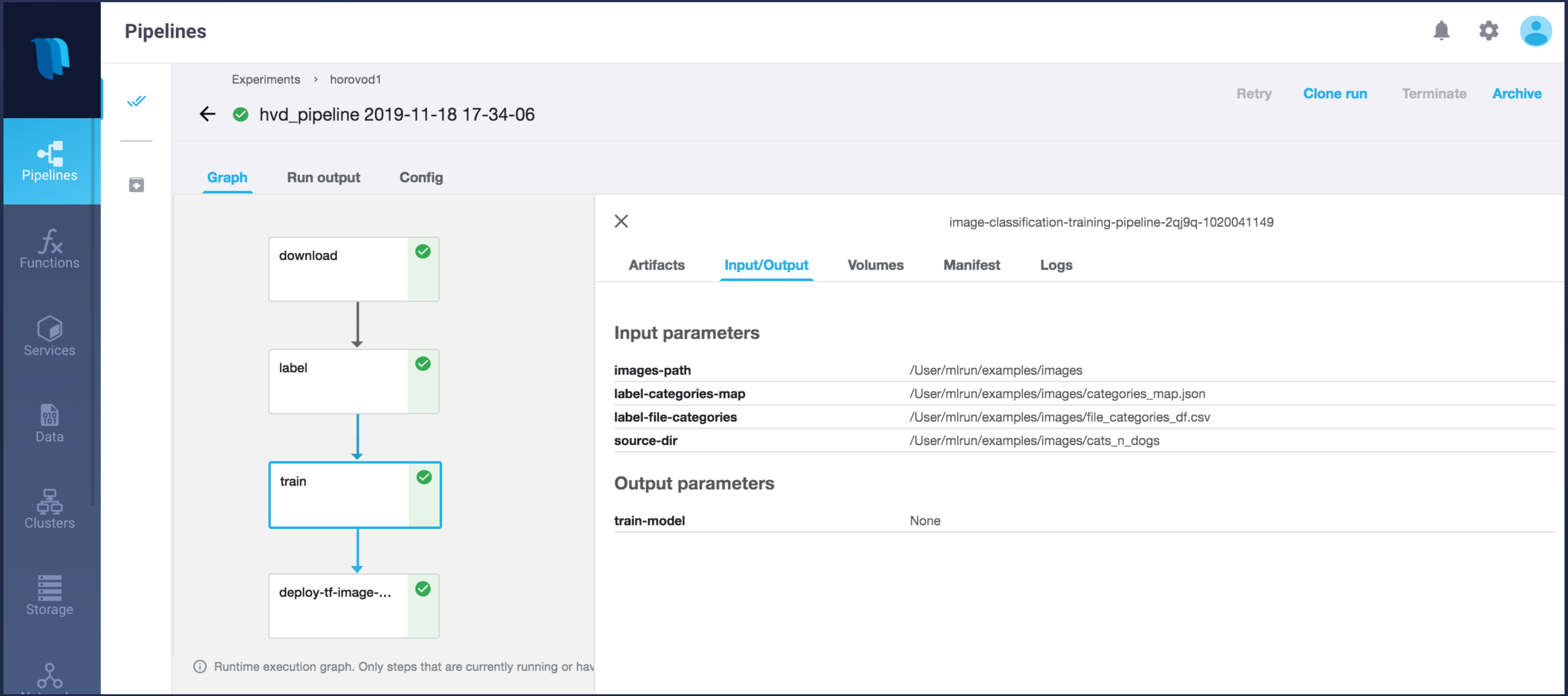
Handler: nuclio_serving:handler

Source code *

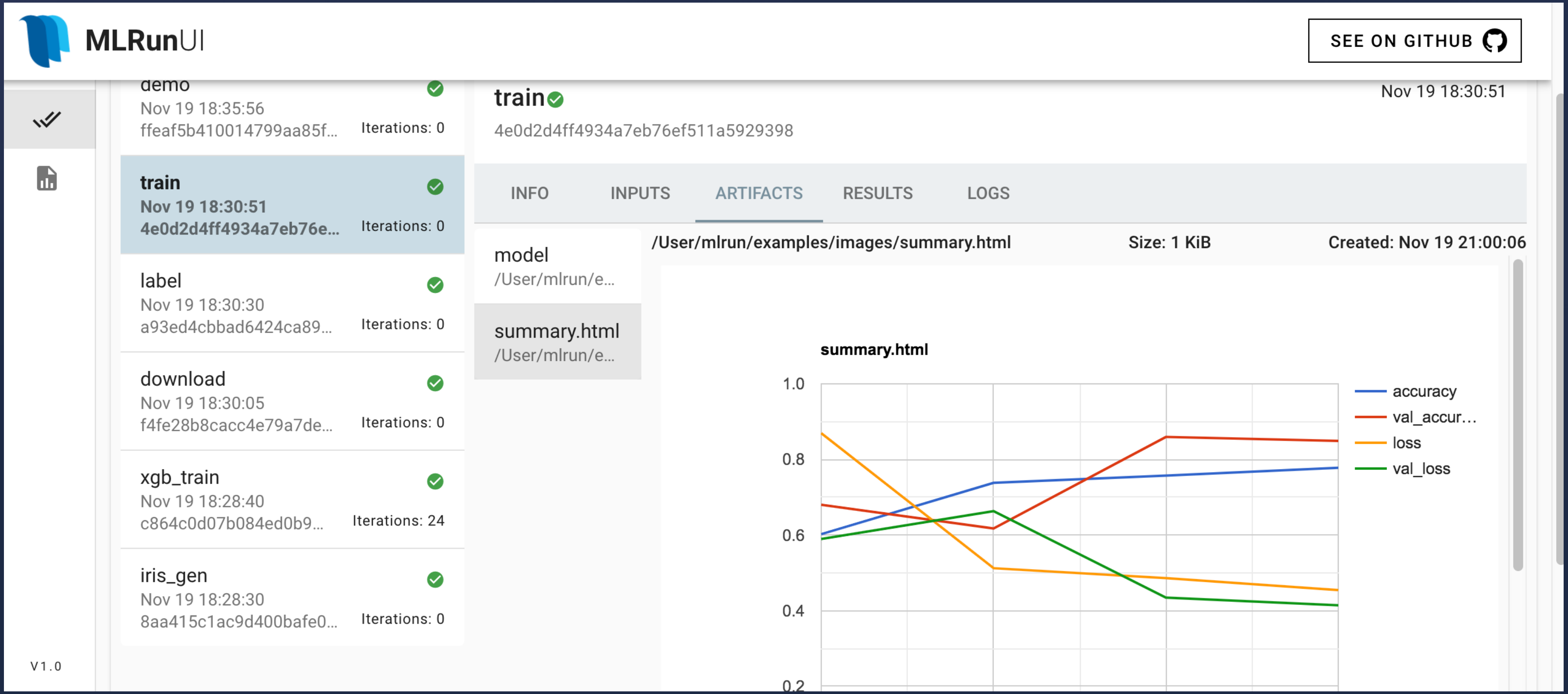
```python
BOOSTER_FILE = "model.bst"

class XGBoostModel(kfserving.KFModel):
    def __init__(self, name: str, model_dir: str, booster: xgb.XGBModel = None):
        super().__init__(name)
        self.name = name
        self.model_dir = model_dir
        if not booster is None:
            self._booster = booster
            self.ready = True

    def load(self):
        model_file = os.path.join(
            kfserving.Storage.download(self.model_dir), BOOSTER_FILE)
        self._booster = xgb.Booster(model_file=model_file)
        self.ready = True

    def predict(self, body: List) -> List:
        try:
            dmatrix = xgb.DMatrix(body)
            result: xgb.DMatrix = self._booster.predict(dmatrix)
            return result.tolist()
        except Exception as e:
            raise Exception("Failed to predict %s" % e)
```

Pipelines
Functions
Services
Data
Clusters
Storage
Networks
Identity

Distributed TenserFlow Pipeline



MLRun UI - Distributed TenserFlow Train Job



iguazio