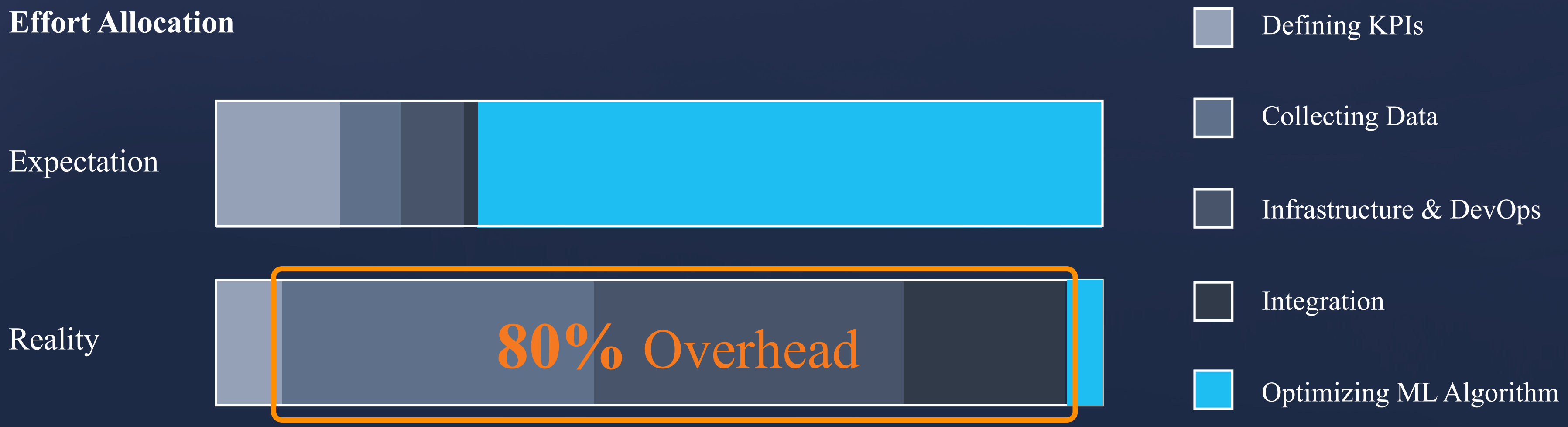
iguazio

Nuclio : KubeFlow Serverless's component

Orit Nissan-Messing, VP R&D, Iguazio

Data Science Teams Don’t Do Data Science



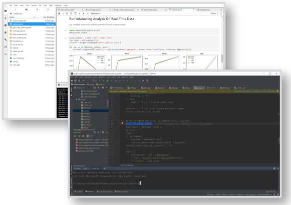
Effort Allocation

80% Overhead

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

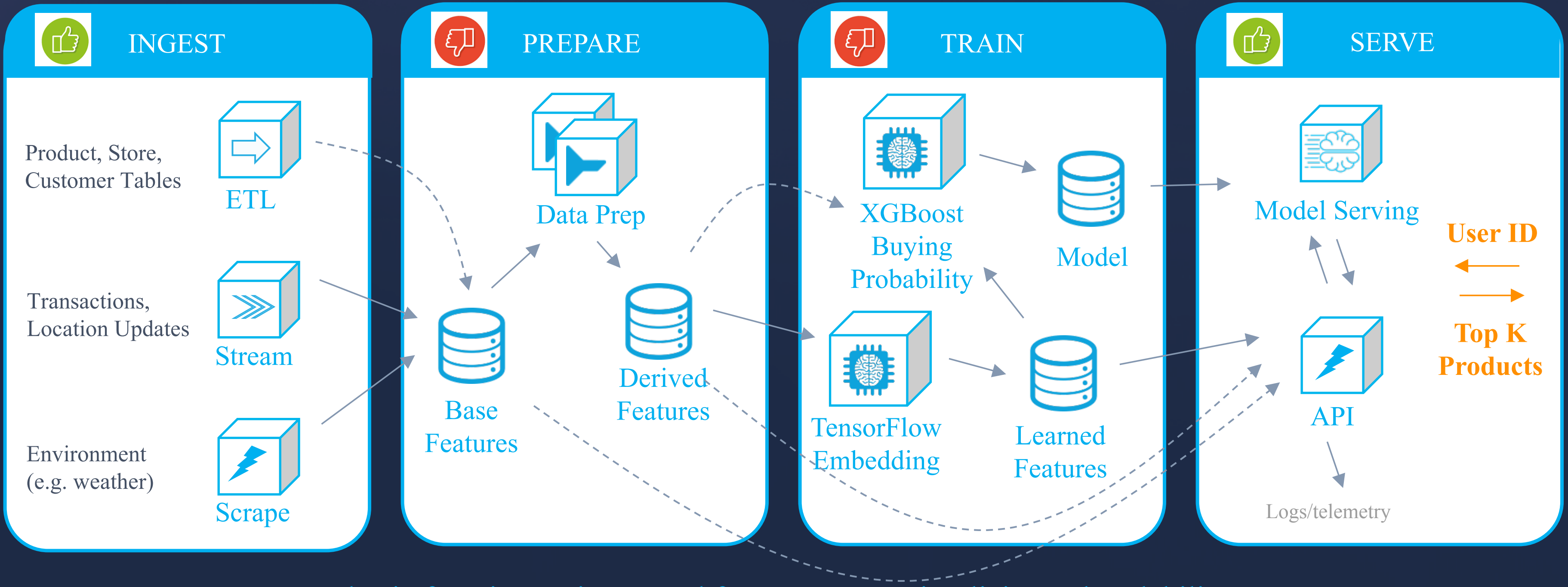
- CI/CD
- Workflows
- Rolling upgrades
- A/B testing

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!

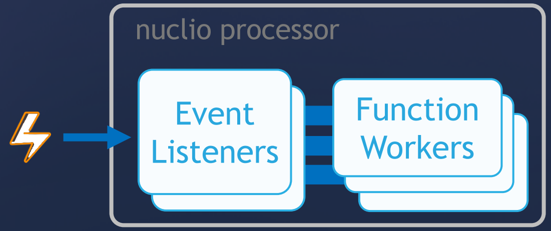
Example: Real-time Product Recommendations



Elastic functions + integrated feature store = simplicity and scalability

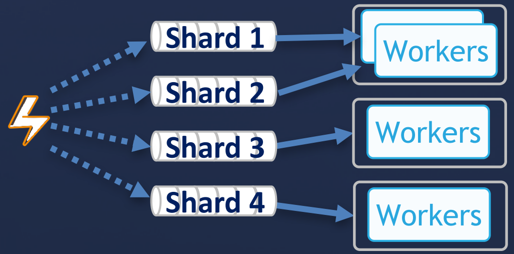
Nuclio: Taking Serverless to Data Intensive Apps

Extreme Performance



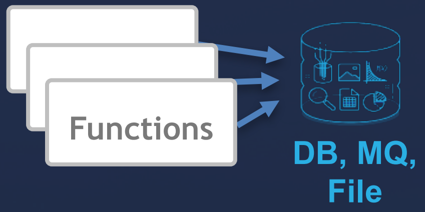
- Non-blocking, parallel
- Zero copy, buffer reuse
- Up to 400K events/sec/proc
- **GPU** optimizations

Advanced Data & AI Features



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

Statefulness



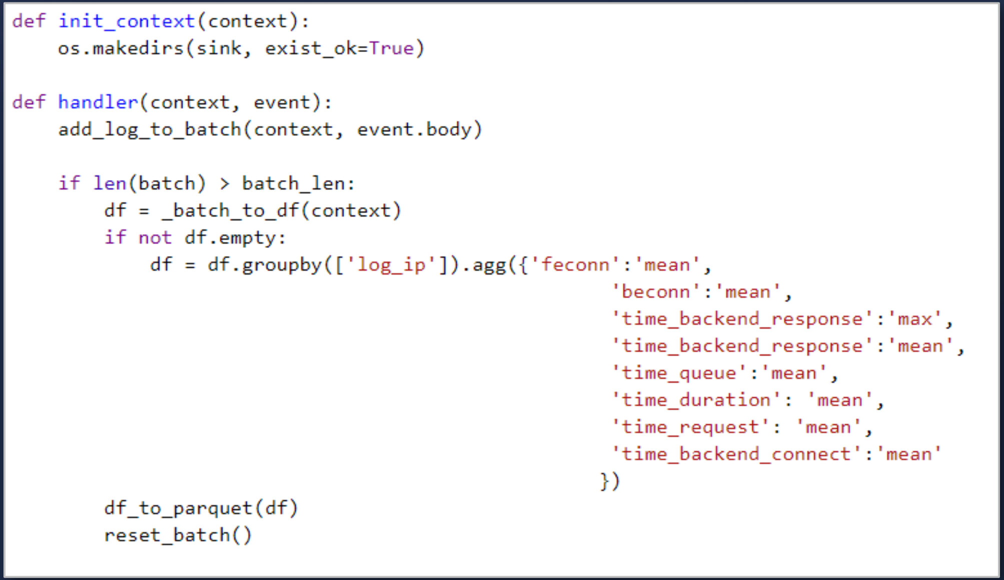
- Data bindings
- Shared volumes
- Context cache

Natively integrated with Kubeflow and Jupyter Notebooks

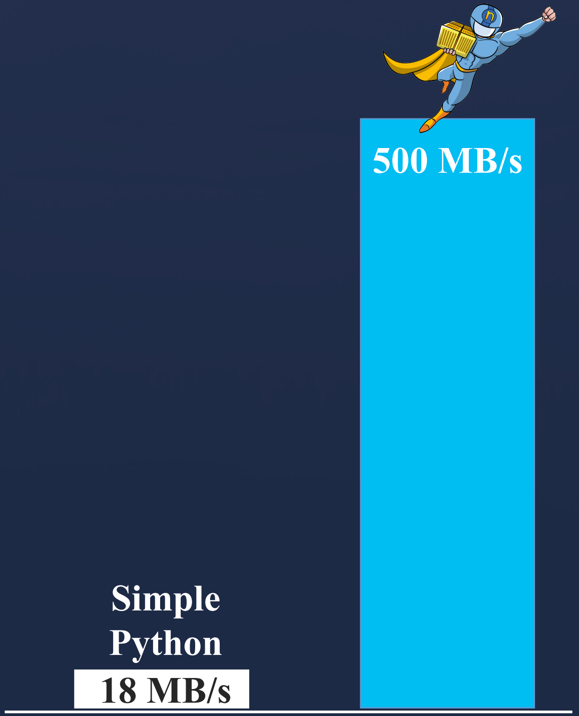
Using Nuclio to Accelerate ETL and Streaming

Simple code! Automated DevOps ! Any Source!

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

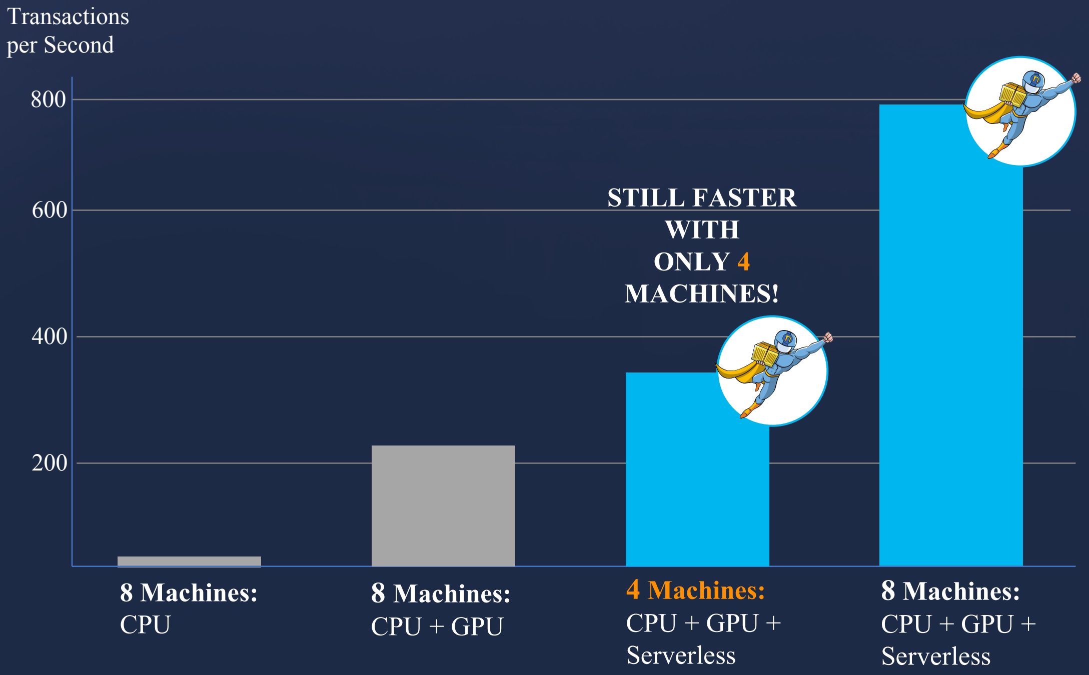


```
def init_context(context):
    os.makedirs(sink, exist_ok=True)

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()
```



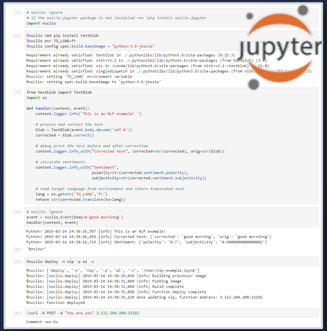
Using Nuclio for Real-time Model Serving

4X Faster model serving (Nuclio + PyTourch + GPUs)



STILL FASTER WITH ONLY 4 MACHINES!

Single command from notebook to function



```
[1]: # nuclio: ignore
     # if the nuclio-jupyter package is not installed run !pip install nuclio-jupyter
     import nuclio

[2]: %nuclio cmd pip install textblob
     %nuclio env TO_LANG=fr
     %nuclio config spec.build.baseImage = "python:3.6-jessie"

     Requirement already satisfied: textblob in ./.pythonlibs/lib/python3.6/site-packages (0.15.3)
     Requirement already satisfied: nltk>=3.1 in ./.pythonlibs/lib/python3.6/site-packages (from textblob) (3.4)
     Requirement already satisfied: six in /conda/lib/python3.6/site-packages (from nltk>=3.1->textblob) (1.12.0)
     Requirement already satisfied: singledispatch in ./.pythonlibs/lib/python3.6/site-packages (from nltk [illegible] .3)
     %nuclio: setting 'TO_LANG' environment variable
     %nuclio: setting spec.build.baseImage to 'python:3.6-jessie'

[3]: from textblob import TextBlob
     import os

     def handler(context, event):
         context.logger.info('This is an NLP example! ')

         # process and correct the text
         blob = TextBlob(event.body.decode('utf-8'))
         corrected = blob.correct()

         # debug print the text before and after correction
         context.logger.info_with("Corrected text", corrected=str(corrected), orig=str(blob))

         # calculate sentiments
         context.logger.info_with("Sentiment",
                                  polarity=str(corrected.sentiment.polarity),
                                  subjectivity=str(corrected.sentiment.subjectivity))

         # read target language from environment and return translated text
         lang = os.getenv('TO_LANG','fr')
         return str(corrected.translate(to=lang))

[4]: # nuclio: ignore
     event = nuclio.Event(body=b'good morninng')
     handler(context, event)

     Python> 2019-03-14 14:38:16,597 [info] This is an NLP example!
     Python> 2019-03-14 14:38:16,659 [info] Corrected text: {'corrected': 'good morning', 'orig': 'good morninng'}
     Python> 2019-03-14 14:38:16,715 [info] Sentiment: {'polarity': '0.7', 'subjectivity': '0.6000000000000001'}
[4]: 'Bonjour'

[5]: %nuclio deploy -n nlp -p ai -c

     %nuclio: ['deploy', '-n', 'nlp', '-p', 'ai', '-c', '/User/nlp-example.ipynb']
     %nuclio: [nuclio.deploy] 2019-03-14 14:38:29,034 (info) Building processor image
     %nuclio: [nuclio.deploy] 2019-03-14 14:38:31,060 (info) Pushing image
     %nuclio: [nuclio.deploy] 2019-03-14 14:38:31,060 (info) Build complete
     %nuclio: [nuclio.deploy] 2019-03-14 14:38:35,090 (info) Function deploy complete
     %nuclio: [nuclio.deploy] 2019-03-14 14:38:35,129 done updating nlp, function address: 3.122.204.208:32182
     %nuclio: function deployed

[6]: !curl -X POST -d "how are you" 3.122.204.208:32182

     Comment vas-tu
```

Why Not Use Serverless for Training and Data Prep?

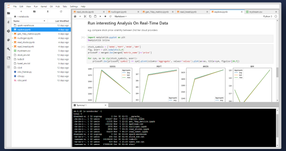
What about Training and data prep ?

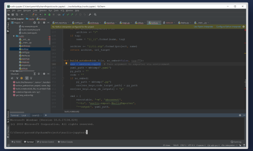
	Serverless Today	Data Prep and Training
Task lifespan	Millisecs to mins	Secs to hours
Scaling	Load-balancer	Partition, shuffle, reduce, Hyper-params
State	Stateless	Stateful
Input	Event	Params, Datasets

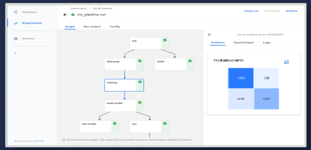
Serverless: resource elasticity (to Zero) and automated deployment and operations

Introducing Nuclio ML Functions

Access from your notebook, IDE, or KubeFlow



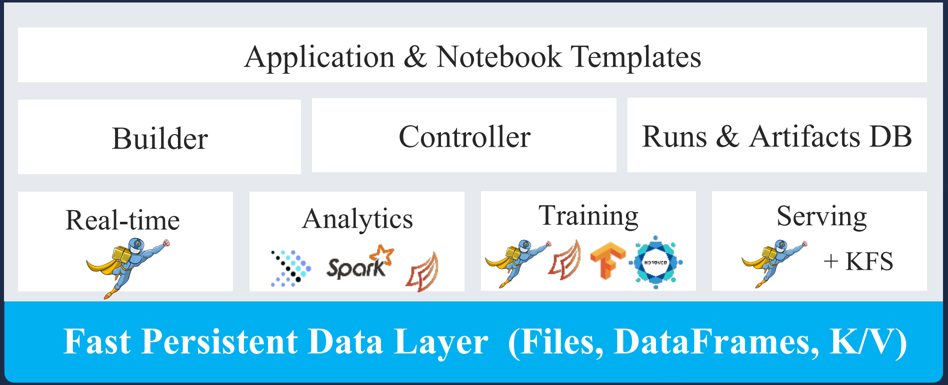




Kubeflow

Common APIs & Automation

Multiple Engines



Application & Notebook Templates

Builder

Controller

Runs & Artifacts DB

Real-time

Analytics

Training

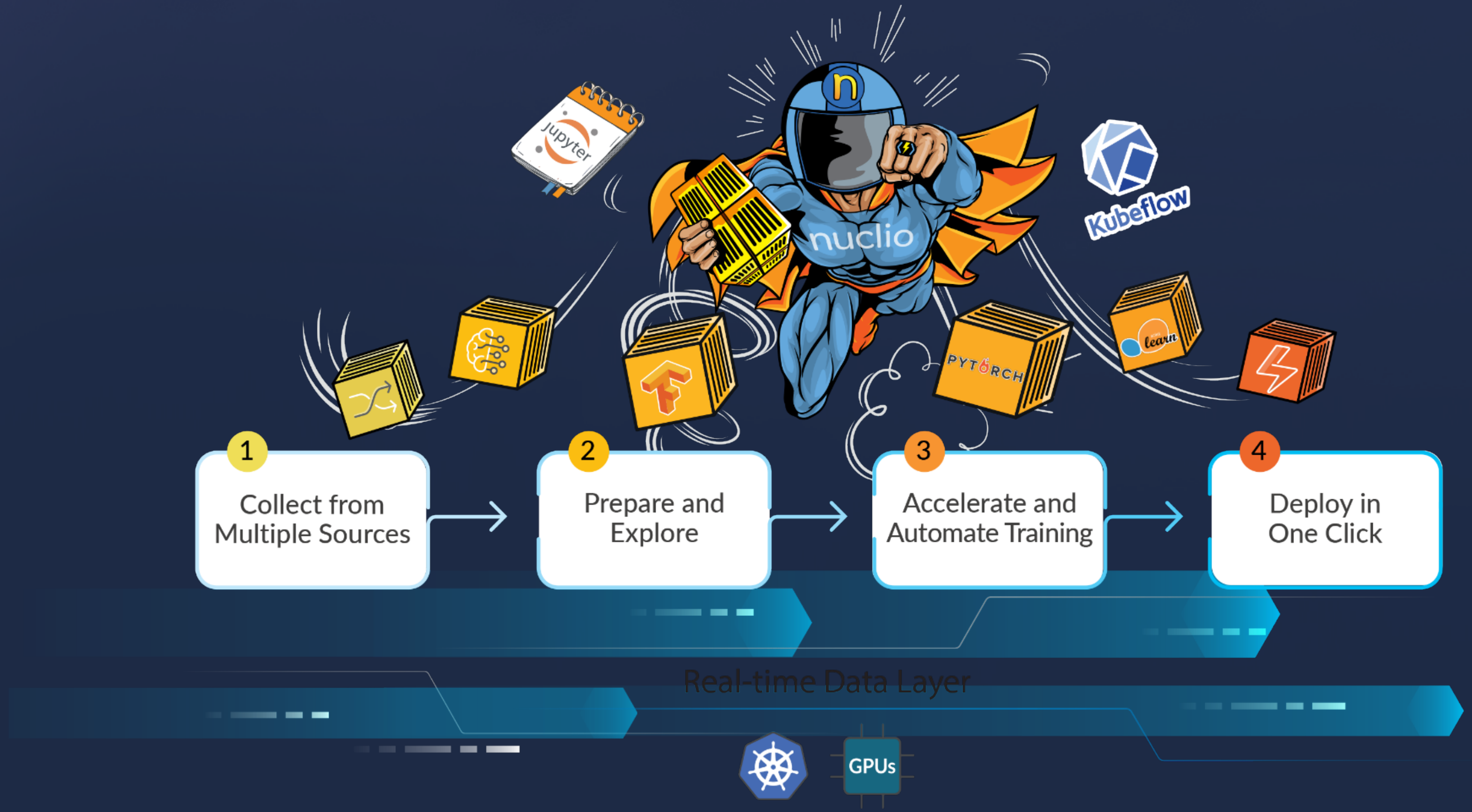
Serving

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

Built-in Artifacts & Runs Tracking

Elastic Scaling

Demo: Fast and Serverless KubeFlow Pipeline



1 Collect from Multiple Sources

2 Prepare and Explore

3 Accelerate and Automate Training

4 Deploy in One Click

Real-time Data Layer

iguazio

Thank You

oritn@iguazio.com, www.iguazio.com