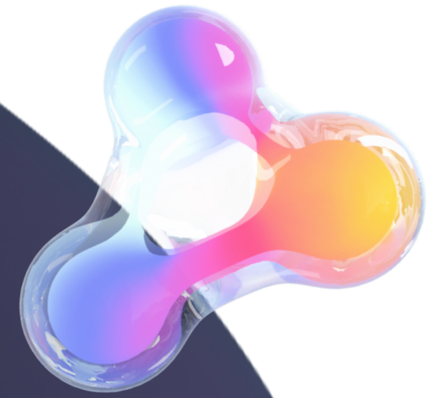
iguazio

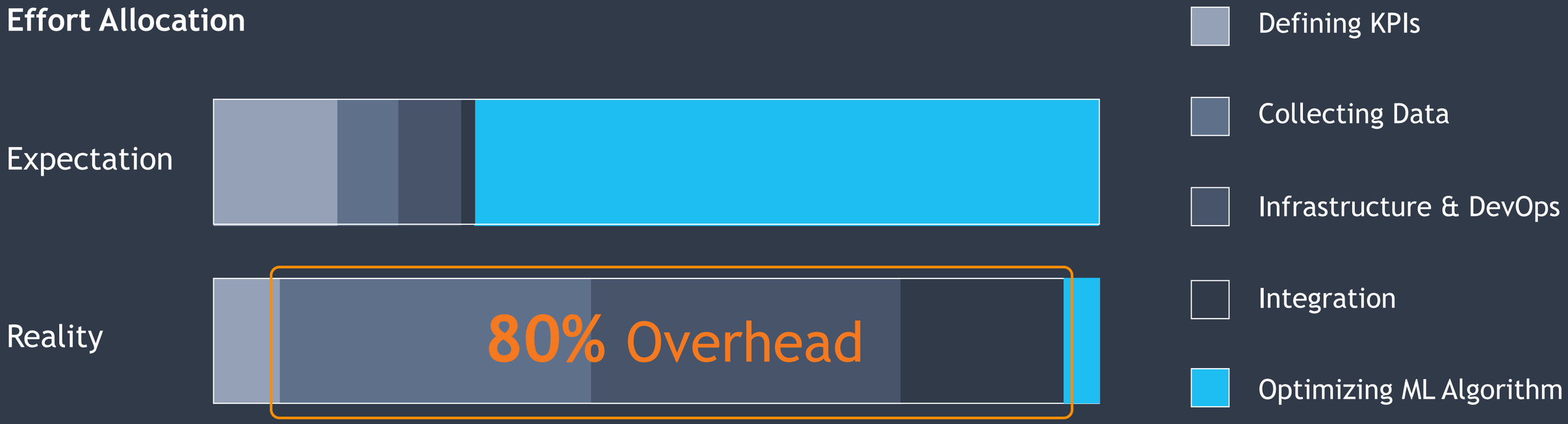
Data Science from Research to Production



Or Zilberman, Data Scientist, Iguazio Jul 2020

Data-Scientists Don’t Do Data-Science !

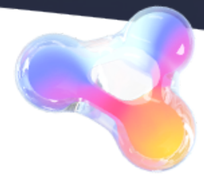
Effort Allocation



The need: Simpler Solutions, Better Data Integration

Source: Google Developers Launchpad

Goal



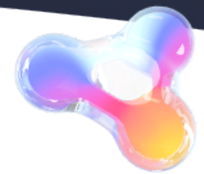
Help you Maximize your time focusing on the

Business Goal

instead of boilerplate related tasks

*focus on your business rather then Devops / Engineering

Goal



Help you Maximize your time focusing on the

Business Goal

instead of boilerplate related tasks

How?

Kubernetes
Cloud OS

Jupyter
DS IDE

Nuclio
Serverless Framework

Kubeflow
Orchestration

MLRun
Orchestration

Or Zilberman

Data Scientist @ Iguazio

Data Scientist @ Glispa Global Group:

Managed Glispa's Data Management Platform (User Personalization)
Owned Glispa's SSP Platform's Data Science efforts

RTB Process & Revenue Optimization
Ad serving network optimizations

Data Scientist @ Teddy Sagi's Group:

Supplied solutions to multiple companies in the group as part of a specialized AI team

E-Commerce: Recommendation engines based on Image, Text and user actions
Forex: Risk management,
Customer Life Time Value, Customer Churn
Automated lead generation

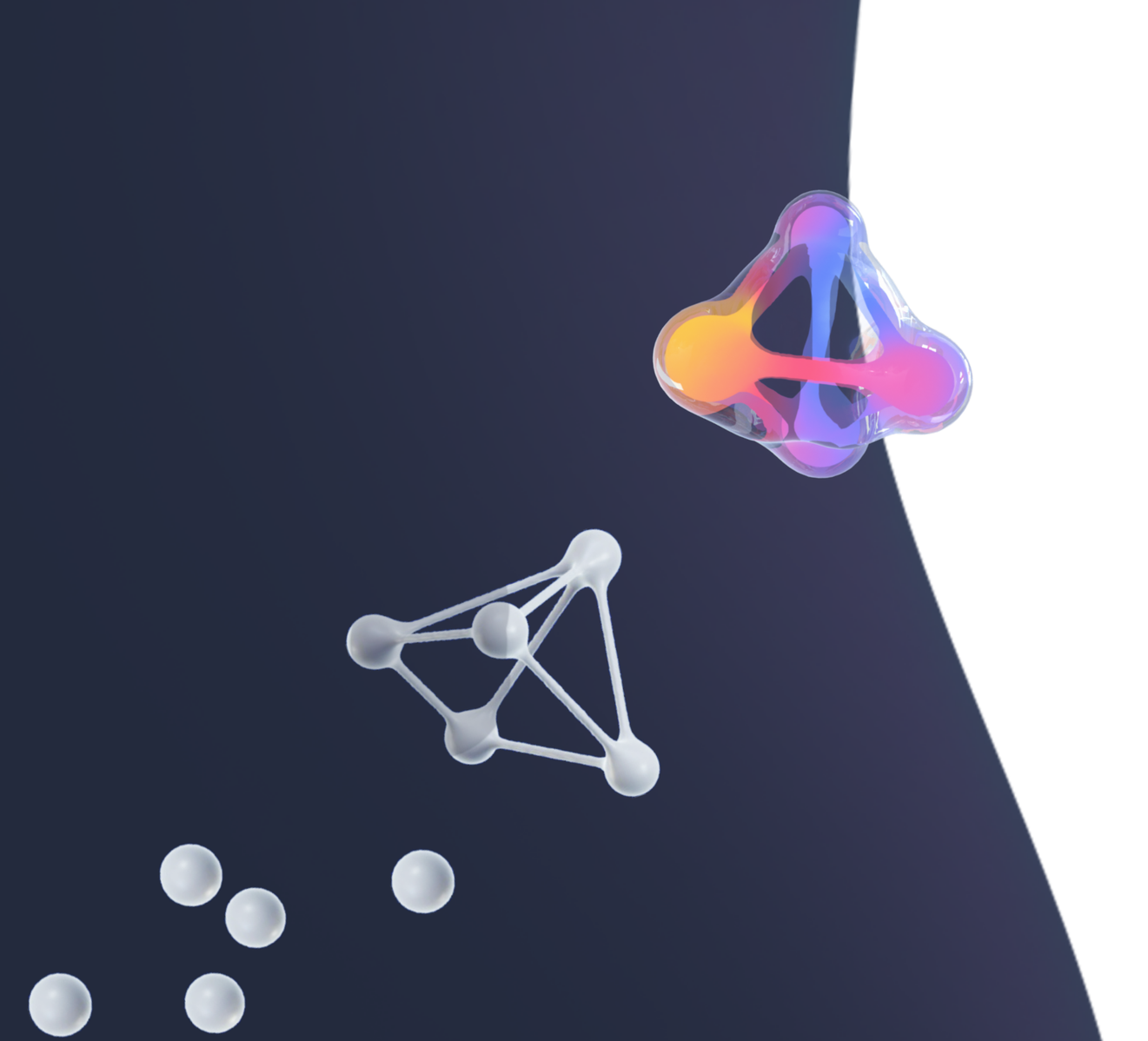
Patents: 3

Latest publication: An FPGA Scalable Parallel Viterbi Decoder @ IEEE MSOC 2018 Vietnam

Additional Experience:
- Head of SEO & CRO @ Dataworks
- VP Product @ NegevJobs
- Director @ Streamono/Gameono

Certifications:
- Google Analytics Certified Individual Qualification (01684861)
- iCDE (18012817THDUUZHU)

Education:
- University of Haifa

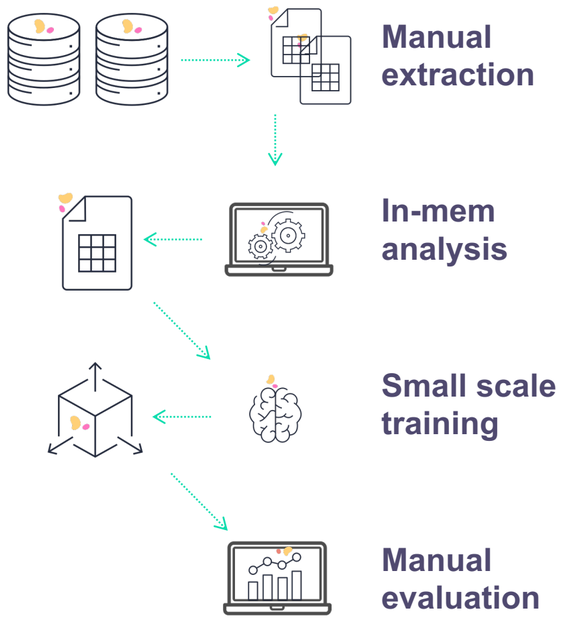


What will we do today?

1. Review the Data Science process in production
2. Technological review - base & architecture
    1. Docker
    2. Kubernetes
    3. Serverless
3. Nuclio
    1. What is nuclio
    2. How to deploy serverless functions from our Jupyter notebook
    3. Training: Deploy a function
4. Kubeflow & MLRun
    1. What is kubeflow
    2. What is MLRun
    3. Training: Deploy functions
    4. Training: Deploy pipeline

87% of AI Projects Never Make it to Production

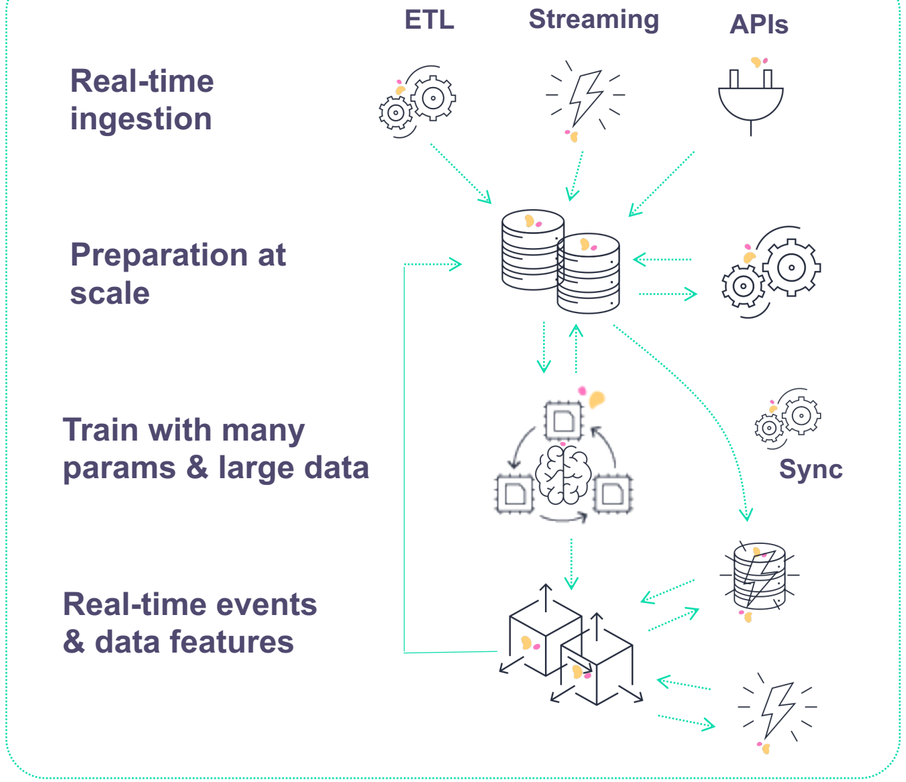
Research Environment



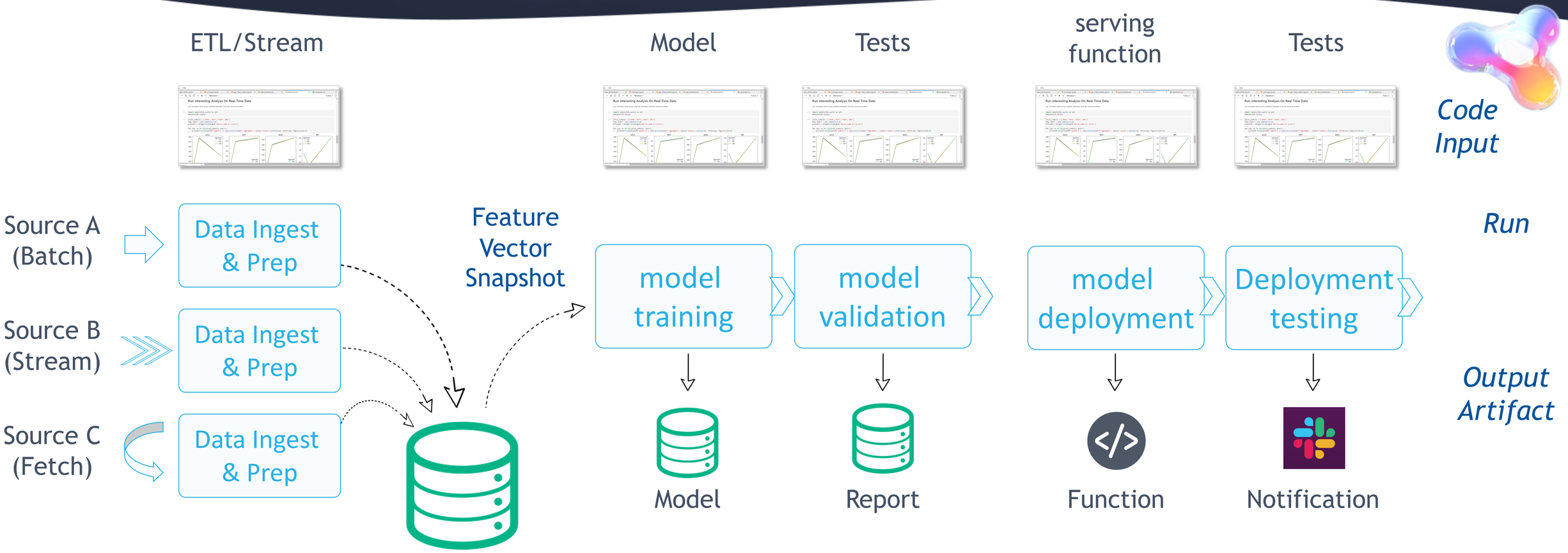
Build from Scratch

with a Large Team

Production Pipeline

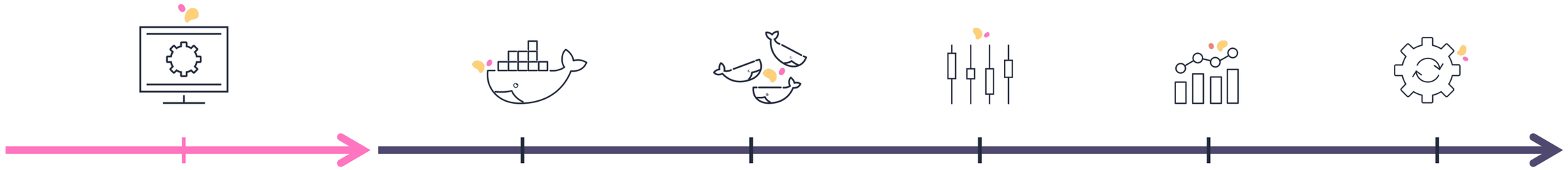


Typical Data Science Pipeline



Pipeline must be automated !

But 87% of AI Models Never Make it to Production



Data Experiment & Model Develop

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

Research - Weeks
with one/few data scientist/s

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

iguazio

Docker

Docker is a platform for developers and sysadmins to **build, share, and run** applications with containers. The use of containers to deploy applications is called *containerization*. Containers are not new, but their use for easily deploying applications is.
Containerization is increasingly popular because containers are:

Flexible: Even the most complex applications can be containerized.
Lightweight: Containers leverage and share the host kernel, making them much more efficient in terms of system resources than virtual machines.

Portable: You can build locally, deploy to the cloud, and run anywhere.
Loosely coupled: Containers are highly self sufficient and encapsulated, allowing you to replace or upgrade one without disrupting others.

Scalable: You can increase and automatically distribute container replicas across a datacenter.
Secure: Containers apply aggressive constraints and isolations to processes without any configuration required on the part of the user.

Docker

Docker Basics

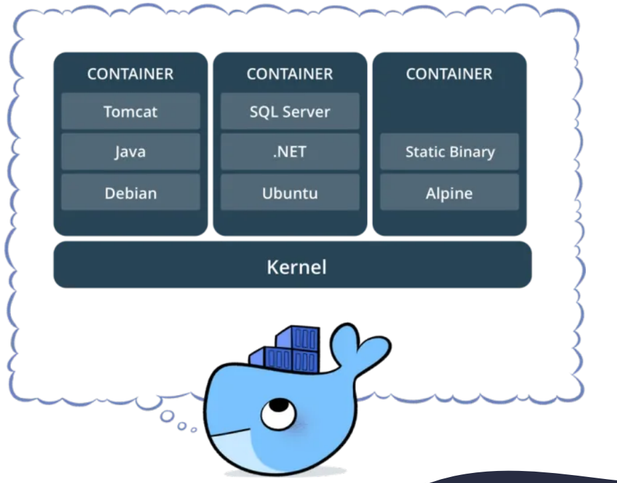
Image
The basis of a Docker container. The content at rest.

Container
The image when it is 'running.' The standard unit for app service

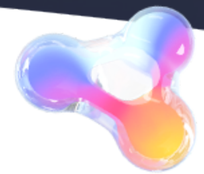
Engine
The software that executes commands for containers. Networking and volumes are part of Engine. Can be clustered together.

Registry
Stores, distributes and manages Docker images

Control Plane
Management plane for container and cluster orchestration



What is Kubernetes?



“Kubernetes” = Greek for governor, helmsman, captain

- open-source container orchestration system
- originally designed by Google, maintained by CNCF
- aim to provide "platform for automating deployment, scaling and operations of application containers across clusters of hosts"

Kubernetes - Pod

The basic, **atomically deployable unit** in Kubernetes.

A Pod consists of one or many *co-located containers*.
A Pod represents *a **single instance of an application***.

The containers in a Pod share the loopback interface (localhost) and can share mounted directories.

Each Pod has it's own, uniquely assigned and internal IP.

Pods are **mortal**, which means that if the node the Pod runs on becomes unavailable, the workload also goes unavailable.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  namespace: default
  labels:
    app: nginx
spec:
  containers:
  - image: nginx:1.13.9
    name: nginx
    ports:
    - name: http
      containerPort: 80
```

Kubernetes - Deployment

With a Deployment, you can manage Pods in a ***declarative and upgradable*** manner.

Note the *replicas* field. Kubernetes will make sure that amount of Pods created from the template always are available.

When the Deployment is updated, Kubernetes will perform a **rolling update** of the Pods running in the cluster. Kubernetes will create one new Pod and remove an old until all Pods are new.

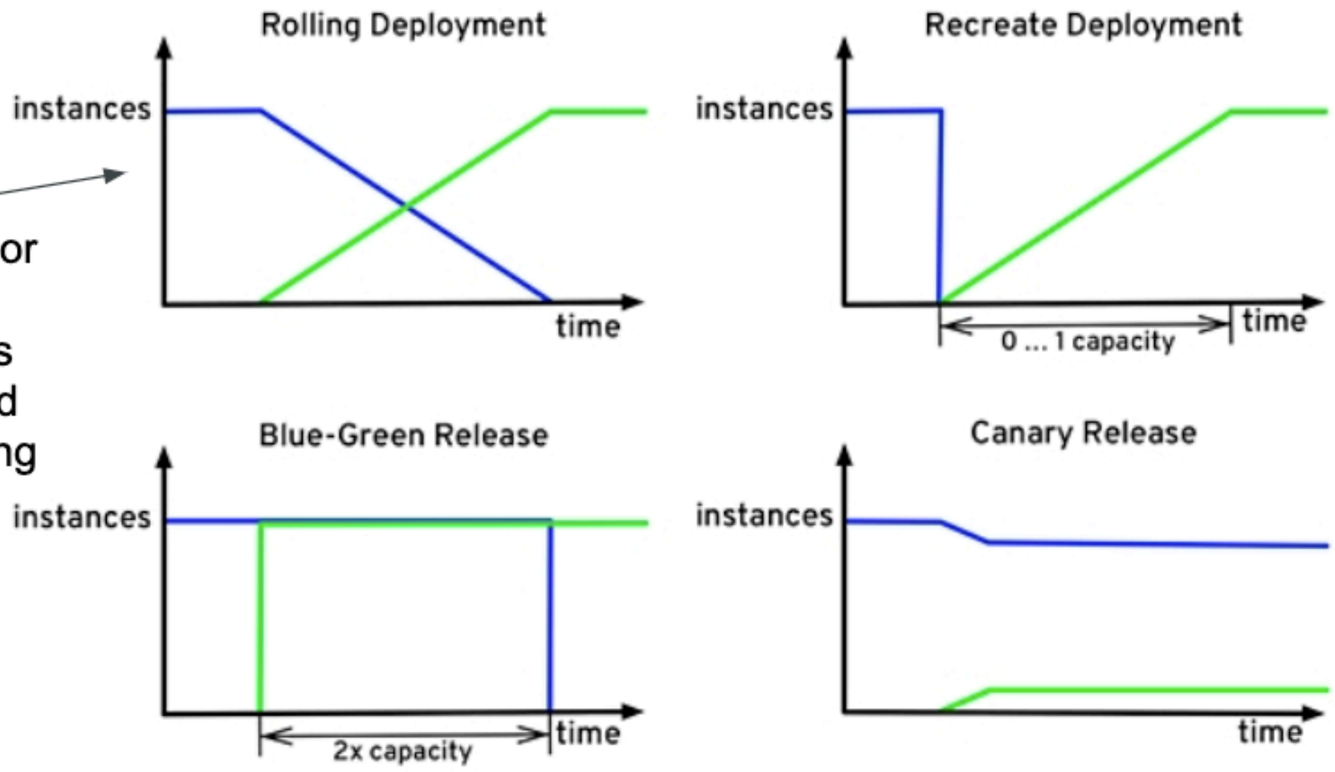
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: nginx
  name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:1.13.9-alpine
        name: nginx
        ports:
        - name: http
          containerPort: 80
```

The Pod Template

Kubernetes – Deployment strategies

The built-in Deployment behavior

The other strategies can be implemented fairly easily by talking to the API.



Kubernetes - Service

A Service exposes one or many Pods via a ***stable, immortal, internal IP address***.

It's also accessible via cluster-internal DNS:
{service}.{namespace}.svc.cluster.local, e.g.
nginx.default.svc.cluster.local

The Service selects Pods based on the **label key-value selectors** (here app=nginx)

A Service may expose multiple ports. This ClusterIP can be declaratively specified, or dynamically allocated.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx
  namespace: default
  labels:
    app: nginx
spec:
  type: ClusterIP
  ports:
  - name: http
    port: 80
    targetPort: 80
  selector:
    app: nginx
```

Serverless - Moving to Micro-Services

Application Modernization

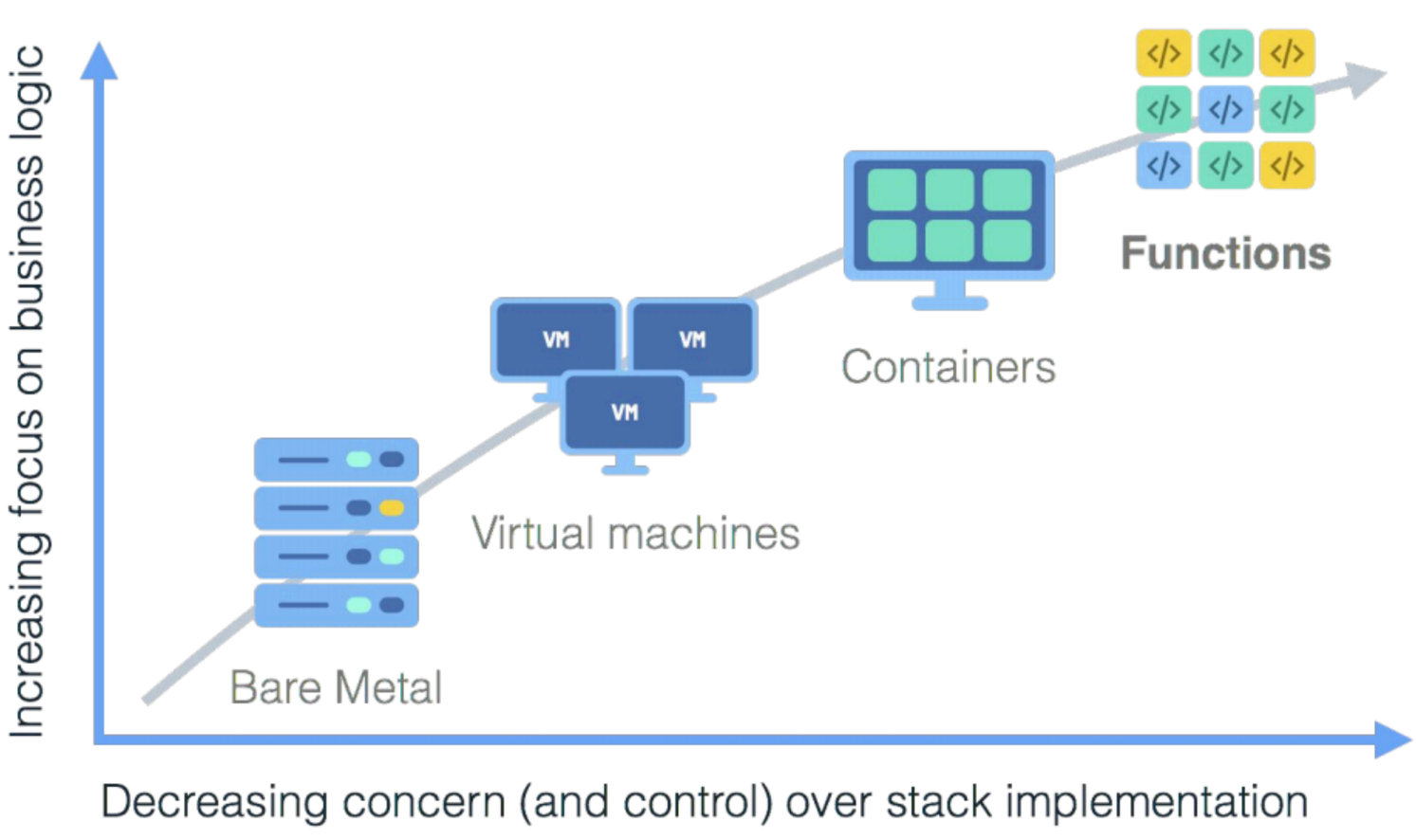
Developer Issues:

- Minor code changes require full re-compile and re-test
- Application becomes single point of failure
- Application is difficult to scale

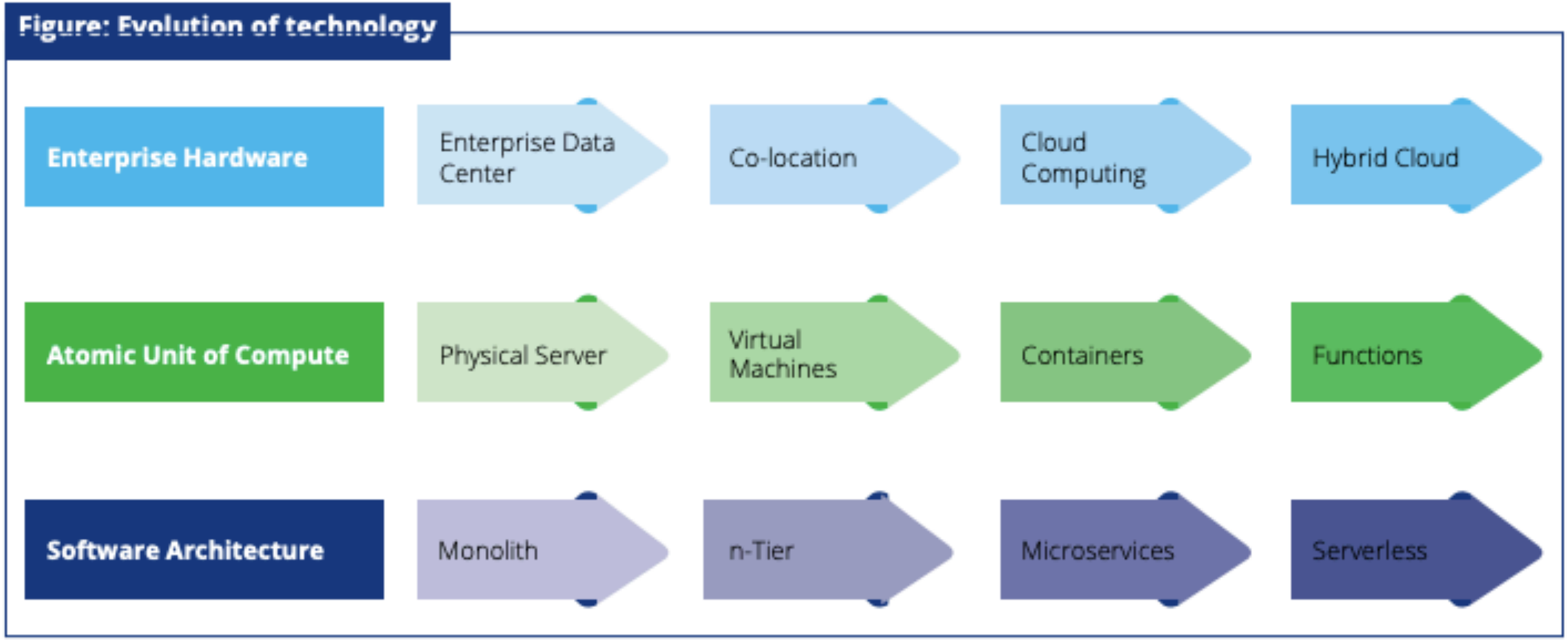
Microservices: Break application into separate operations

12-Factor Apps: Make the app independently scalable, stateless, highly available by design

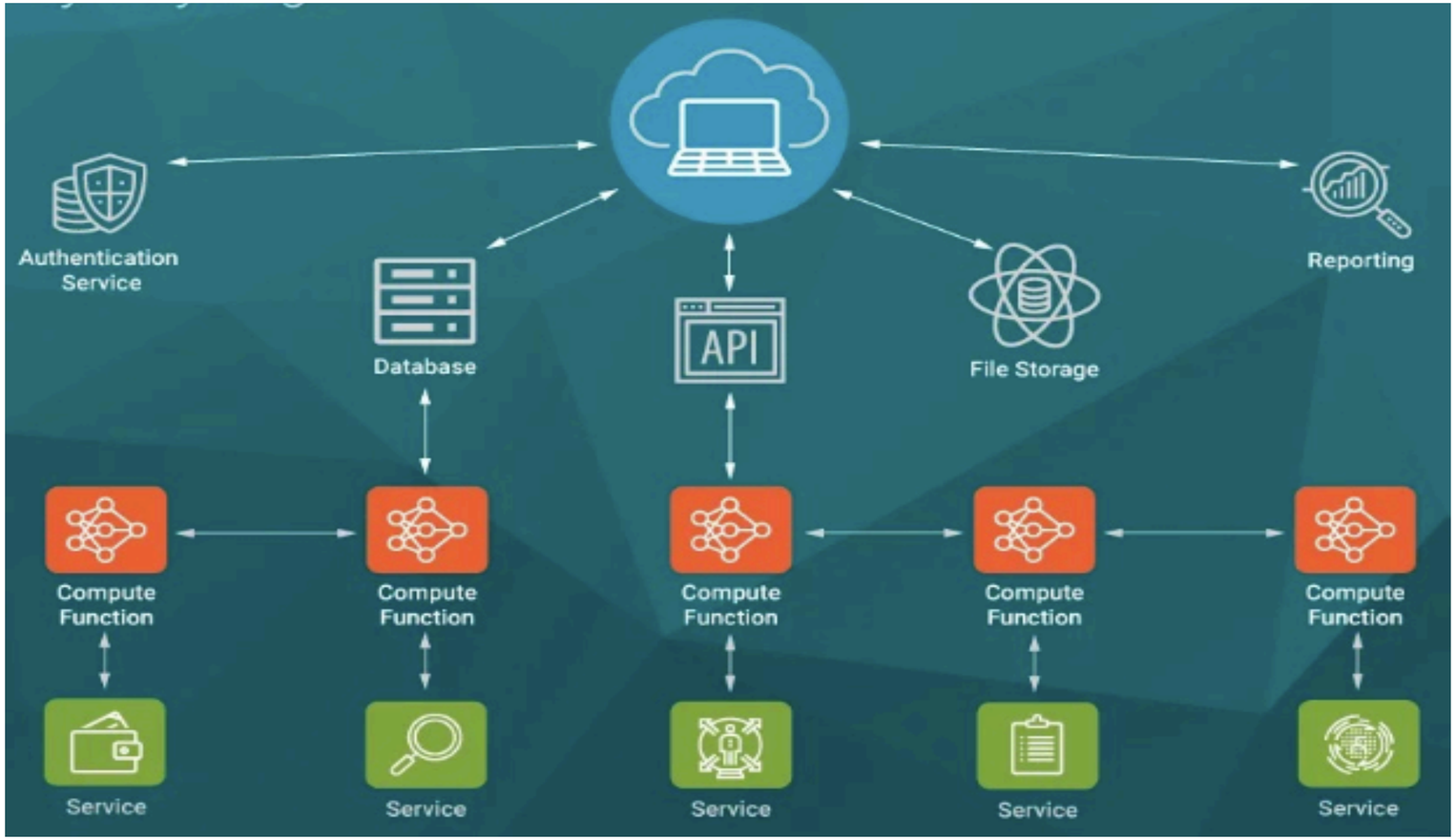
Serverless


Increasing focus on business logic
Bare Metal
VM
VM
VM
Virtual machines
Containers
Functions
Decreasing concern (and control) over stack implementation

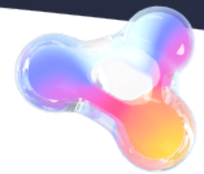
Serverless



Serverless Architecture



The Agility Challenge:
How to Innovate & Iterate Faster, Deploy Everywhere



Modern Apps Today:

Complex, Long Development

- Many moving parts, lack of skilled developers, scientists, and DevOps

Production readiness

- Error-handling, security, logging, auto-scaling, live updates, ..

Slow to respond or act

- High-latency, or fast but inaccurate
- Cloud latency/bandwidth barriers

The Need:

Serverless
Automate dev and ops

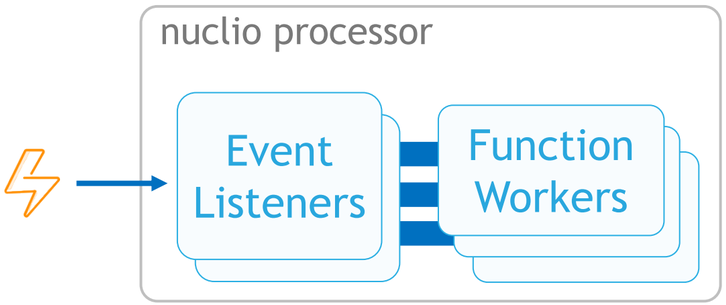
High-performance
Data, AI and app services

Federated
In the cloud, on-prem or edge

iguazio

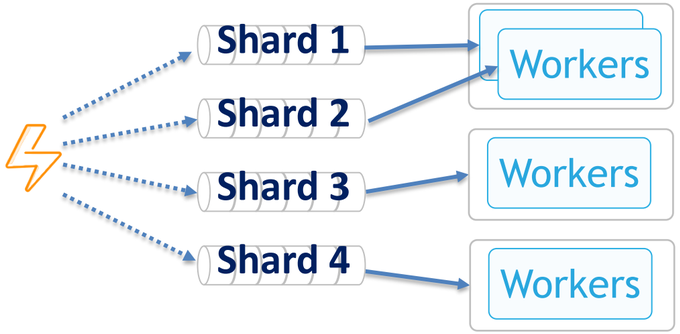
Nuclio: Taking Serverless to The Next Level

Extreme Performance



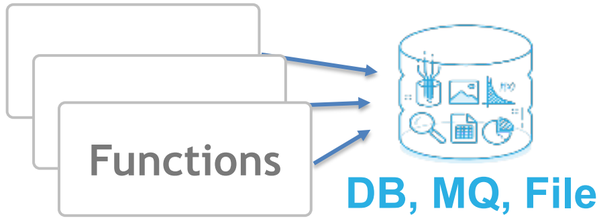
- Non-blocking, parallel
- Zero copy, buffer reuse
- Up to 400K events/sec/proc
- **GPU** Support

Advanced Data & AI Features



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

Statefulness

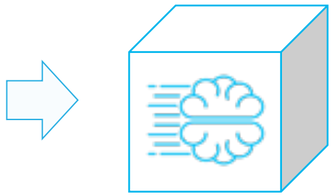


- Data bindings
- Shared volumes
- Context cache

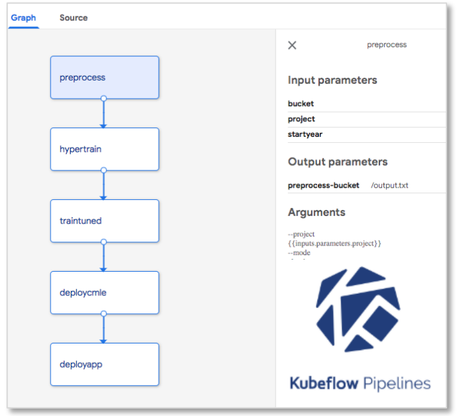
Open-source Serverless for compute and data intensive tasks, **100x faster** than AWS Lambda !

iguazio

Nuclio: Fast Serverless for Data Science & RT Analytics



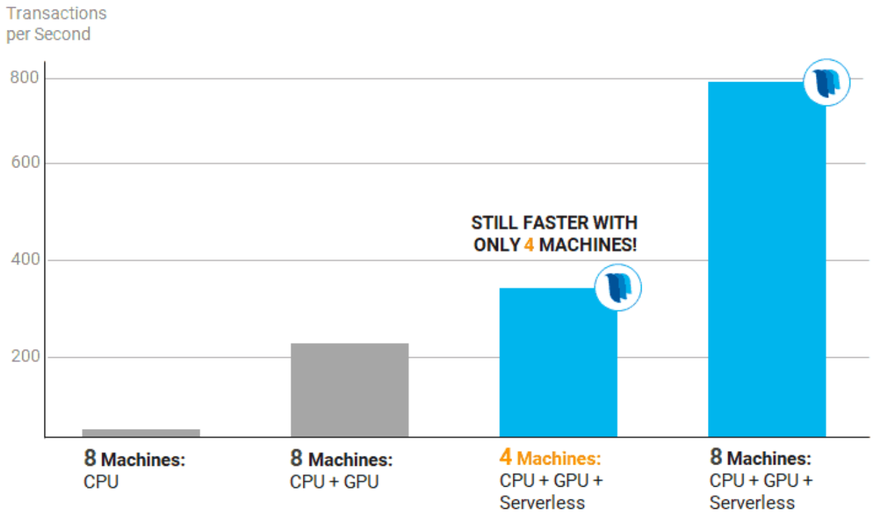
magic commands from notebook to function



Extending ML Pipelines from batch:

1. **Parallel** processing
2. Code **build**/deployment
3. **Stream** processing
4. API/Model **Serving**

High-performance IO and Computation + GPU Optimizations

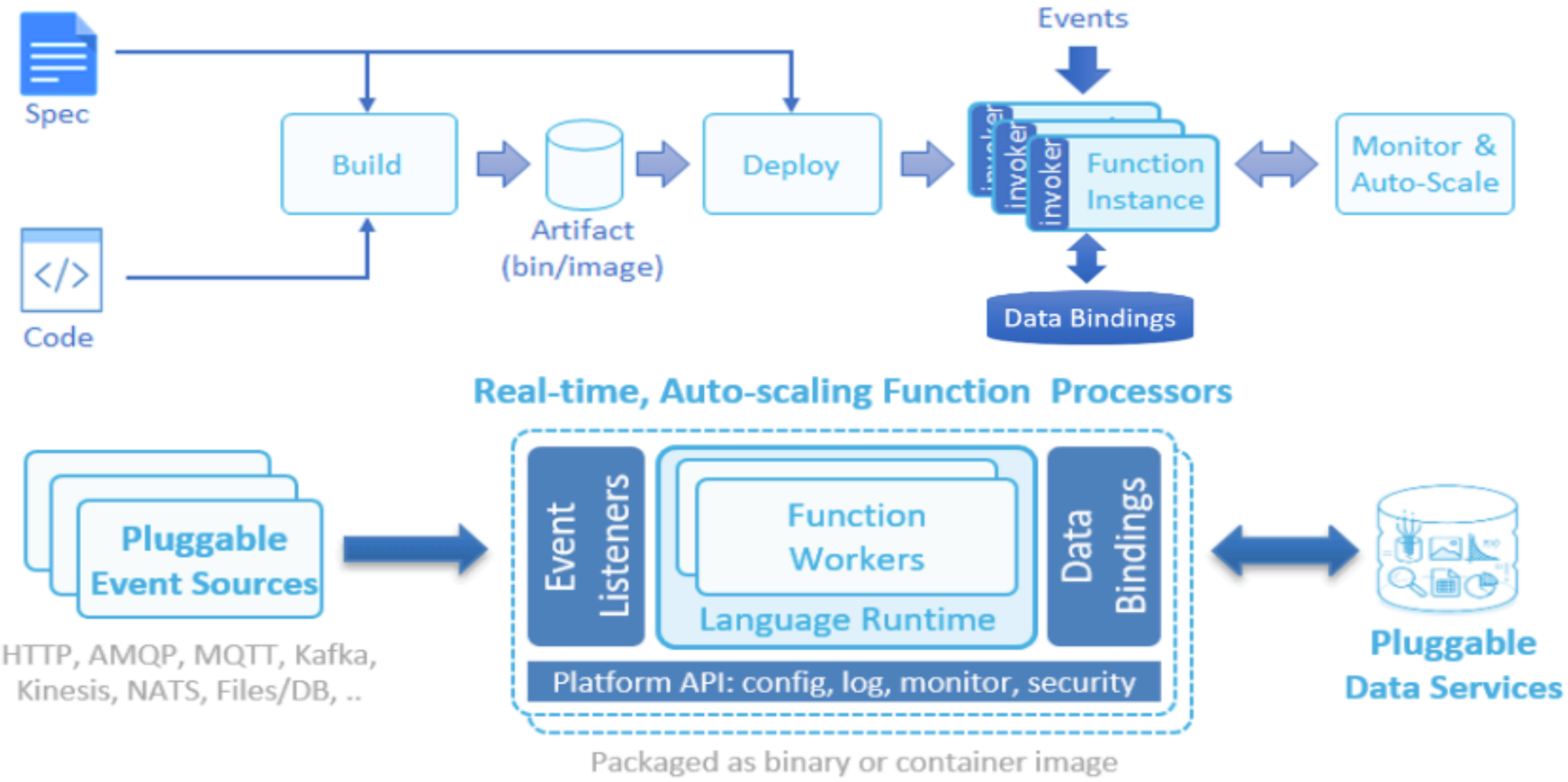


Code + DevOps Automation:

1. Auto-scaling (to zero)
2. Automated logging & monitoring
3. Security hardening
4. Auto-build and CI/CD
5. Workload mobility (cloud/edge/..)

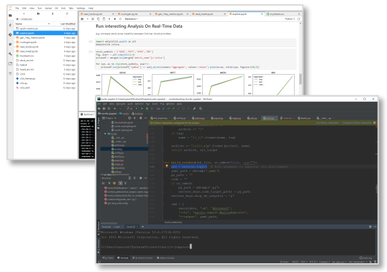
iguazio

Nuclio – Under the hood



MLRun

Vendor and Runtime Independent ML Deployment and Tracking Automation



Develop & test everywhere

Multiple data store types

Data Stores

Ship code

Add data

params

Vendor independent runtime or ML service

Runtime (+ Parallelism)

User Code

ML Pipeline

Vendor independent mechanism to collect:

- Results
- Status
- Artifacts metadata/versions
- Logs/Metrics

Check it out on github : mlrun

iguazio

Demos !