kestra.

OPEN SOURCE DECLARATIVE DATA ORCHESTRATION

Hello ! 👋

Loïc Mathieu

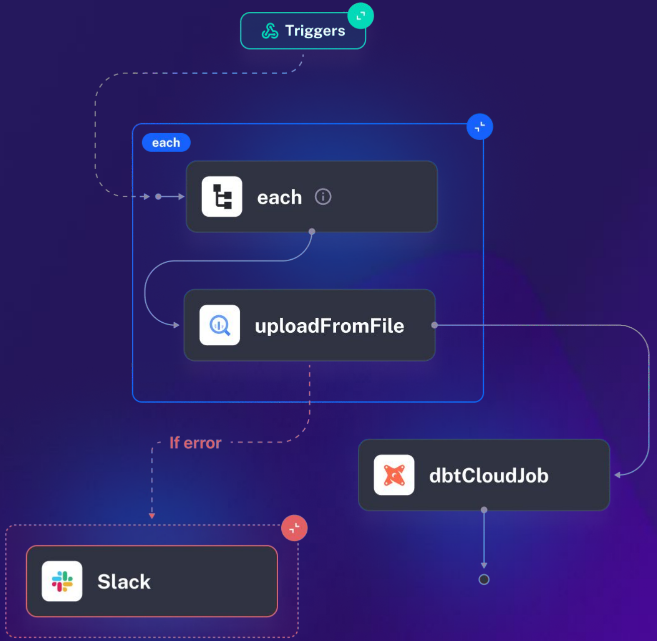
Lead Software Engineer at Kestra | GCP GDE | Quarkus contributor | Book Author **@loicmathieu**

What’s Kestra?

Powerful Orchestration Logic

Workflow can start simple and then evolve with complex branching, parallel or dynamic tasks, error management, retry, timeout, ...

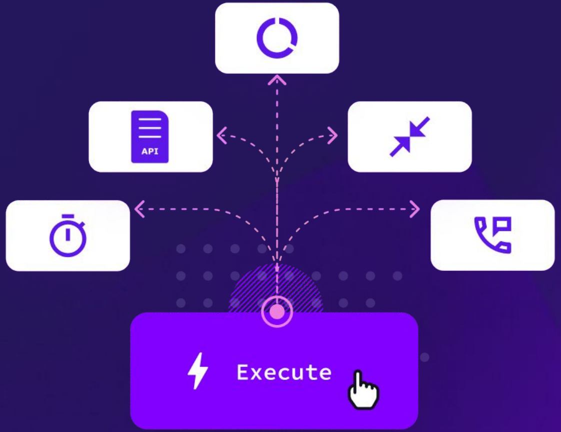
- Sequential
- Parallel
- Loop
- Pause
- Switch & if
- Subflow



Scheduling at Scale

Keep complete control over how you want to automate your data pipelines, making integrating with your existing systems and streamlining your workflow easy

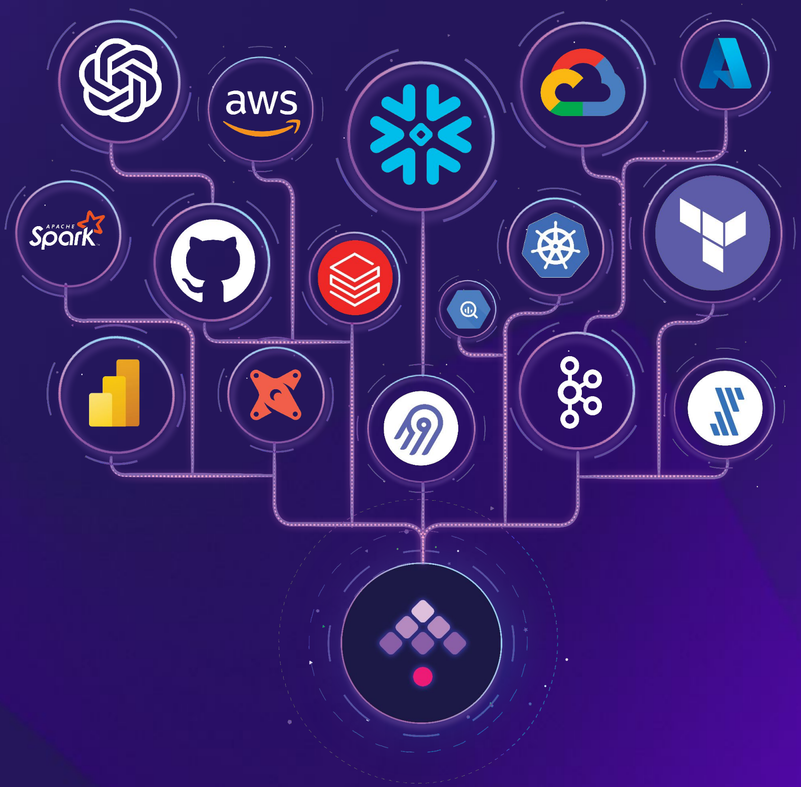
- Schedule
- File System Detection
- Database Query
- Messaging
- Flow Dependencies
- API Webhook



Data at Heart

Kestra is built with data at heart and integrates directly with your sources & destinations

- Abstract File System
- Strong Typing
- ETL & ELT
- Polyglot
- Docker Support



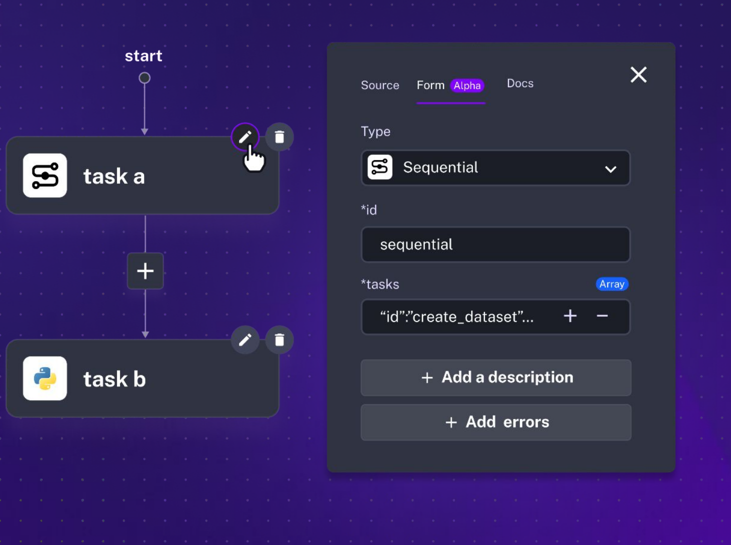
Declarative Orchestration

Avoid getting a data-engineering PhD

Easy setup

YAML based flow definition

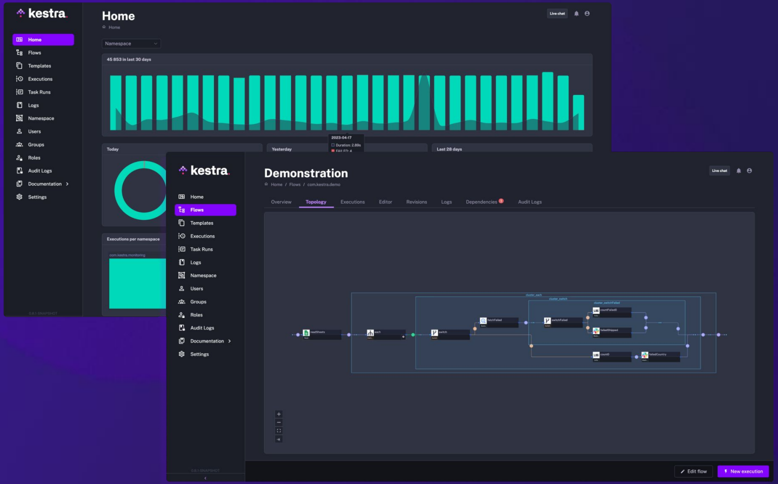
Low-code features



A Complete Platform

Do it your way

Rich User Interface



Data Ops & CI/CD

VS Code extension

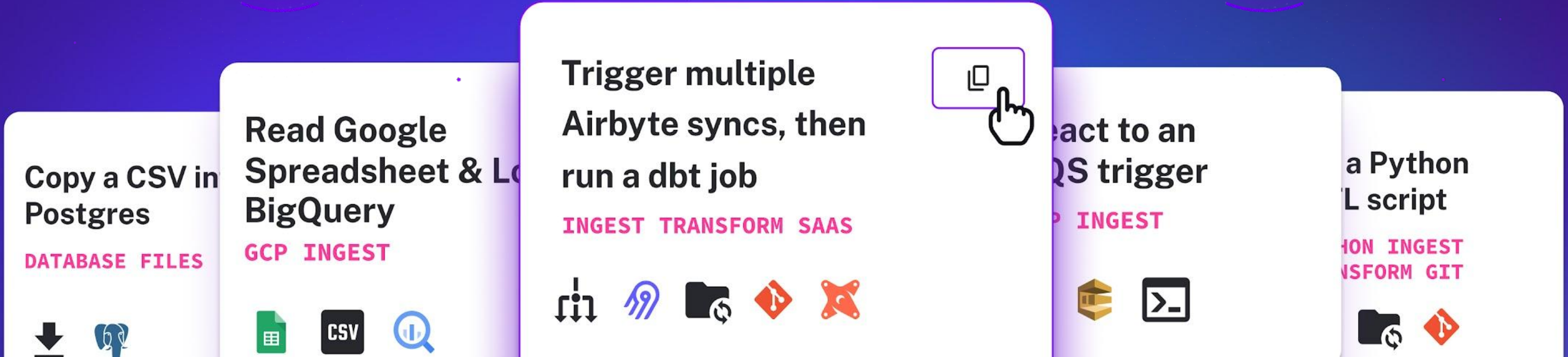
Github Action & Gitlab CI/CD support

Terraform provider

Endless Possibilities

400+ plugins available or bring your own.

Get started with the Blueprints library



Copy a CSV in
Postgres
DATABASE FILES

Read Google
Spreadsheet & L
BigQuery
GCP INGEST

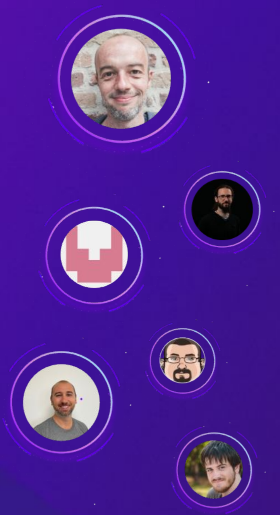
Trigger multiple
Airbyte syncs, then
run a dbt job
INGEST TRANSFORM SAAS

eact to an
QS trigger
INGEST

a Python
L script
HON INGEST
NSFORM GIT

Open Source Project

The core of Kestra and all its plugins are open source



5300+ ⭐

1200+
Members

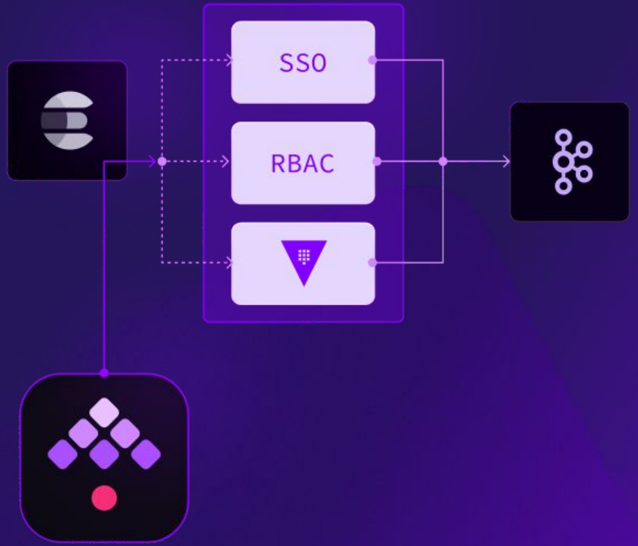
- 👉Without any limitation
- 👉Without vendor lock-in
- 👉Used by large corporations all over the world

🔗 github.com/kestra-io/kestra

Enterprise Edition ✨

Security & High Availability

- High availability with no single point of failure
- Single sign-on & Service account
- Full role-based management
- Integration with secret managers
- Audit logs
- Worker Group

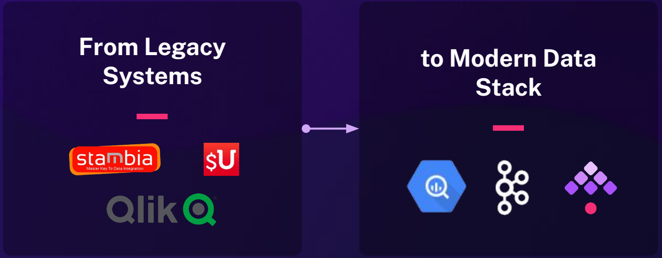


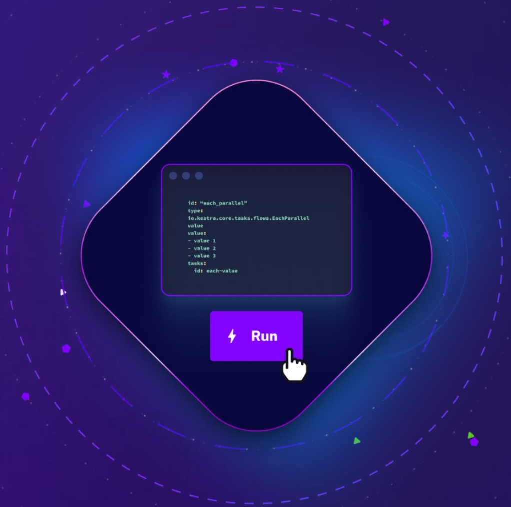
Leading companies orchestrate their Data with Kestra.

👉250 users
👉+4.000 flows created
👉+5.000.000 Tasks executed/month

From Legacy Systems

to Modern Data Stack

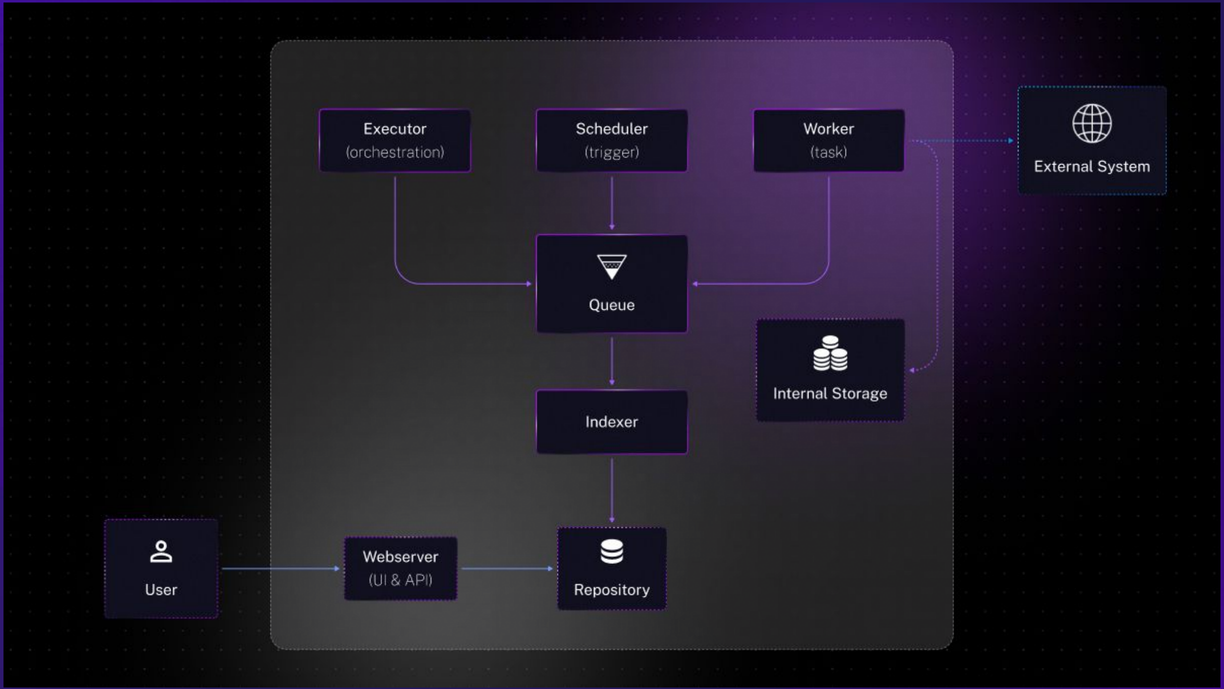




```
id: “each_parallel”
type:
io.kestra.core.tasks.flows.EachParallel
value
value:
- value 1
- value 2
- value 3
tasks:
  id: each-value
```

Demo Time!

A Fault-tolerant & Performant Architecture



A Fault-tolerant & Performant Architecture

Server components communicate via asynchronous queues.

The server access to the database directly via repositories.

The internal storage stores flow data of arbitrary size out of the database.

A **Flexible** Architecture

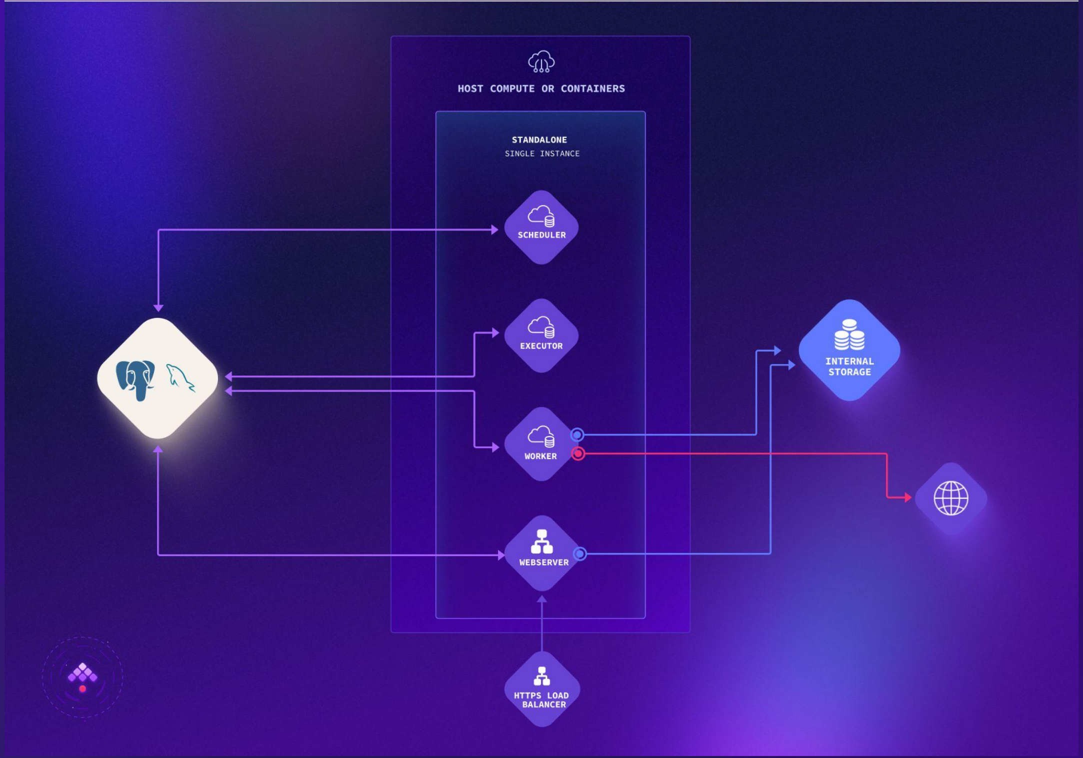
Two deployment modes:

- **Standalone** (all-in-one)
- **Microservices** (one server component per process)

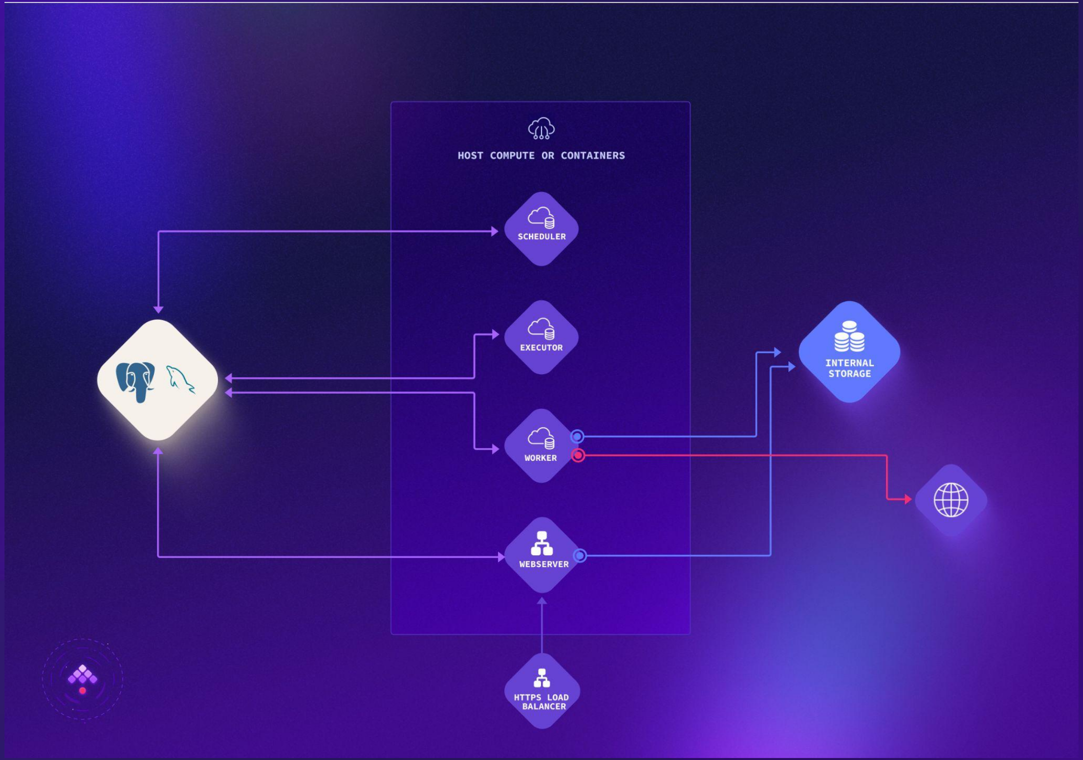
Two runners:

- **Database** (H2, MySQL, PostgreSQL)
- **Kafka & Elasticsearch** (EE only)

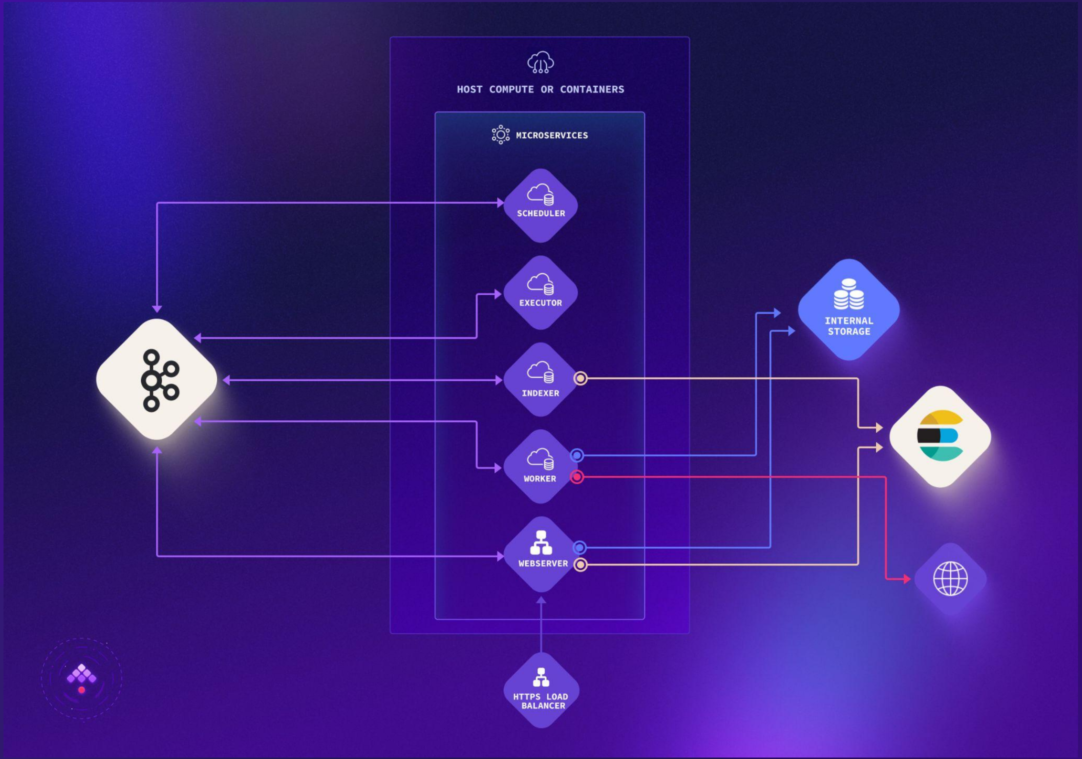
Small-sized deployment



Medium-sized deployment



HA with no SPOF deployment



An Extensible Platform

- Almost everything is a plugin
- Plugins are written in Java with Gradle.
- Small learning curve: vanilla Java
- All the monitoring features out of the box
  (errors, logging, metrics, output, ...)

Local Storage

Task

Trigger

Condition

Secrets

Runners

Even the API can be enhanced!

An Extensible Platform

Clone

Start from the Plugin template:
https://github.com/kestra-io/plugin-template

Code

Then follow the Plugin Developer Guide:
https://kestra.io/docs/plugin-developer-guide

Run

Finally, build your plugin and add it to the plugin path
using `KESTRA_PLUGIN_PATH`

Written in Java

Most Data orchestrators are written in Python.

Most Data orchestrators mandate that you write Python code.

Kestra is declarative, so you don’t need to use a programming language to use Kestra so that it can be written in a language other than Python.

Written in Java

Kestra takes advantage of the Java language:

- Inputs and outputs are strongly typed.
- Java dynamicity makes it easy to create a plugin system.
- ScriptEngine allows scripting language to be run <u>inside</u> the JVM. Useful for efficient row-to-row transformations.

Written in Java

Kestra takes advantage of the Java ecosystem:

- Huge ecosystem of libraries that support almost everything related to data.
- Java libraries and drivers are often the reference implementation.
- JDBC: so easy to support tens of databases.
- Docker, Kubernetes, Cloud libraries
- Data format: JSON, AVRO, Parquet, ...

Written in Java

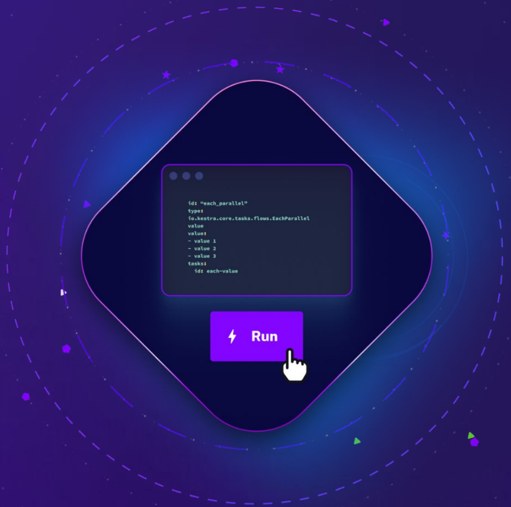
Kestra takes advantage of the JVM:

- High performance
- Leverage multi-threads
- Highly scalable
- Java Security for worker task isolation
- Robust platform, widely known by operational teams.

Written in **Java**

Kestra EE leverage Kafka Stream:

- No SPOF
- Distributed scheduling of tasks
- Blazing-fast task orchestration
- Kafka under steroid:
    - Transactional stream processing
    - Global State store
    - Punctuation (to process distributed timely events)
    - Fault tolerance



```
id: “each_parallel”
type:
io.kestra.core.tasks.flows.EachParallel
value
value:
- value 1
- value 2
- value 3
tasks:
  id: each-value
```

Demo Time!

kestra.

Thank you !