layline.io

event data processing. simplified.

What exactly is layline.io?

layline.io is an event-data-processing platform which combines high-configurability and operations with the power of reactive stream management.

It can handle massive amounts of real-time and batch event-data for a myriad of use cases, either locally or in a fully distributed and resilient fashion.

Define individual workflows, business logic, and deal with any data format, source or sink, simply by configuration.

No custom object coding. Instead let a product do the job.

Motivation

What motivated us to create layline.io were a number of real-life problems in the area of event-data-processing, which were largely unaddressed by existing solutions. We deemed these issues so important, that we thought it is worthwhile addressing them with a new product.

- Real-time volume to increase tenfold in the next few years.
- Companies struggling to reap transactional value of real-time data (do something useful with it in the first 0-3 seconds).
- Companies' system architectures solidly migrating to cloud architectures, without proper solutions to support this in all aspects.
- Lack of actual products to support distributed real-time event processing, instead of having to create custom programmed solution based on development toolkits.
- Availability of awesome streaming technology like akka and flink, which unfortunately have a steep learning curve and do not allow for complete setup and operations by configuration.
- lots more.

Can't I solve these problems by other means?

Yes, but it's tough, lengthy and will cost you dearly. In the past years, and in the context of the Big Data evolution we have witnessed a myriad of new concepts and truly awesome technologies emerge. Cloud, process distribution, asynchronous processing and many more, just to name a few.

You have seen many cool examples and show cases, for sure. None of them, however, were simple to create and required lots of time and specialized staff and resources.

layline.io wants to make this easy by enabling users to

1. create individual event workflow processing logic
2. which can be deployed across a distributed processing network (physically, logically and geographically)
3. which asks to configure and deploy everything without custom coding, and
4. supports any event-processing scenario from small to super extra large in a highly scalable and resilient environment (or simply on your laptop!)

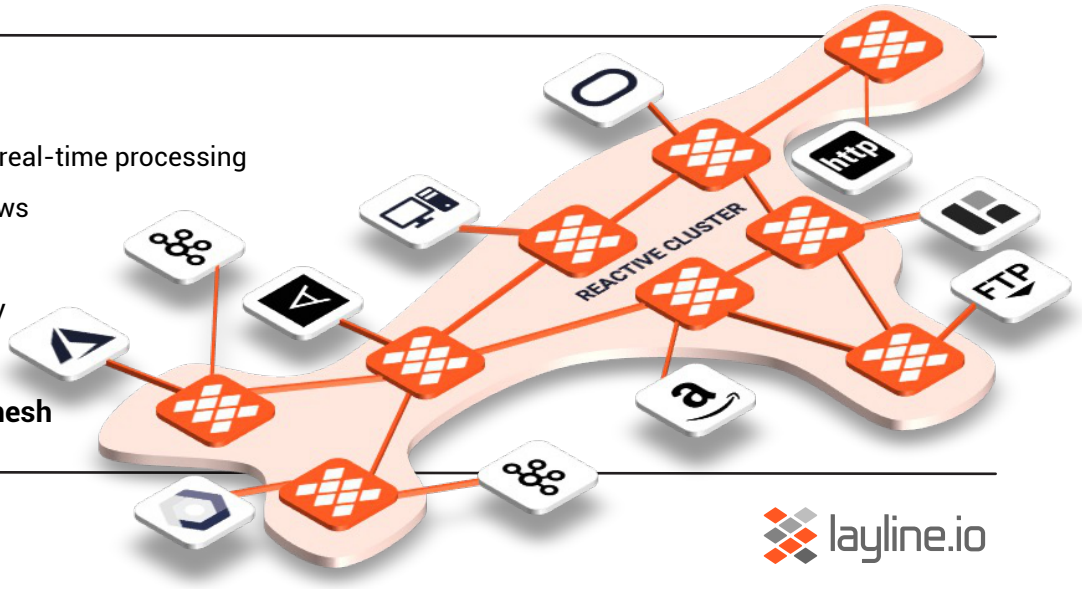
What about microservices?

microservices	vs	layline.io
individual custom coding	-	configurable workflows
distribution unaware	-	distribution built-in
not resilient OOTB	-	resilient by design
scalability external	-	scalability built-in
no load-balancing	-	load-balancing built-in
not data aware	-	data aware by design
different code on each node	-	same engine on all nodes

key points

+ resilient high-performance real-time processing
+ configurable event-workflows
+ data awareness
+ out-of-the-box connectivity
+ cloud-native architecture

= distributed reactivedata mesh



layline.io

platform

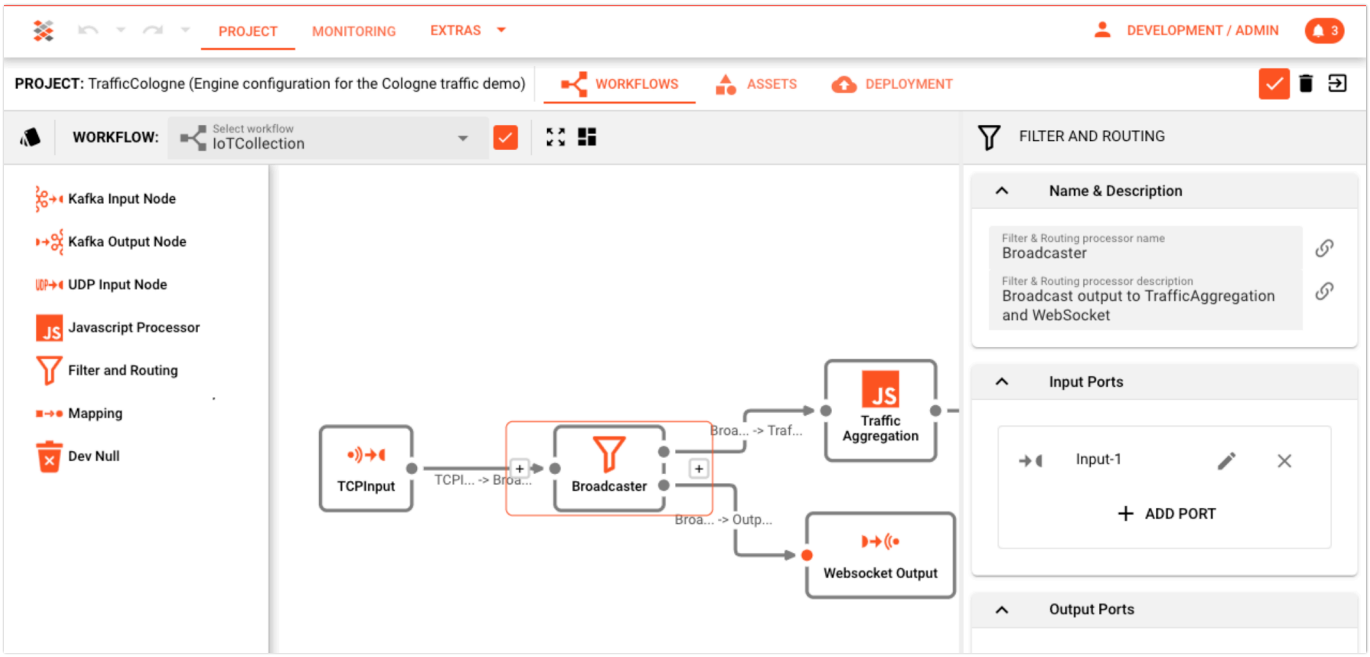
Comprehensive configuration environment

layline.io implements **reactive stream management** under the hood. It's awesome. To make things easy we have added a configurability framework, flexible data parsing technology, automated deployment, operations and monitoring tools and interfaces, connectors and much more.

Workflows

Workflows are a core part of layline.io. Everything revolves around workflows which are configured through the web-based configuration center. Each workflow manifests an event processing logic to run on one or more nodes.

Within a workflow, you can pick from a number of pre-defined, but configurable processors. Examples are filtering, data mapping, custom javascript logic, and more.



Processors

Each workflow consists of a number of processors which are wired up to form the actual event-flow. Processors can be picked from a list of pre-made processors (which is continuously extended). A future version will support creation of custom processors via SDK. Generally you can distinguish between source, logic, and sink processors.

Data awareness

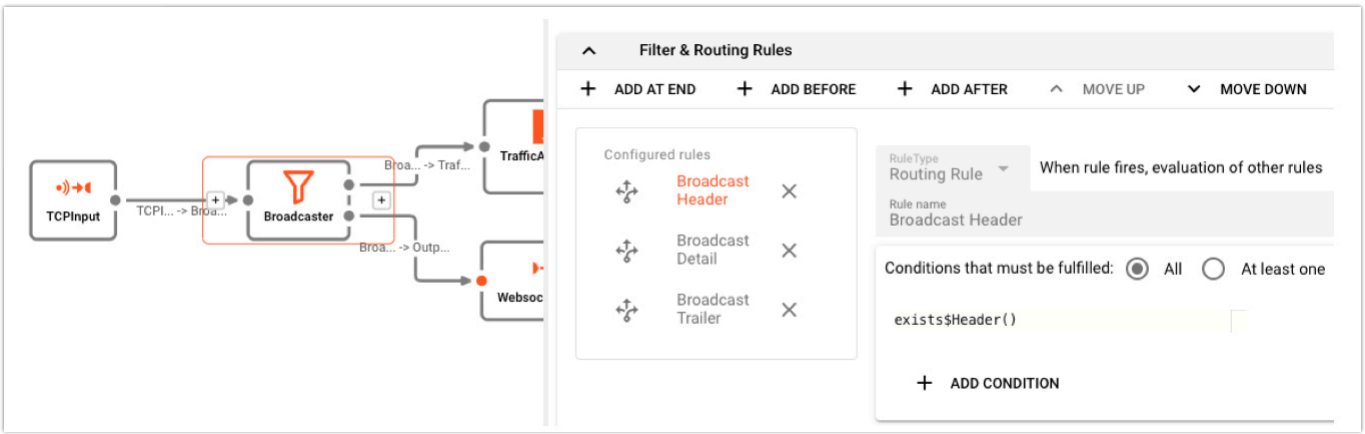
By data awareness, we mean that layline.io is able to understand any form of structured data by way of configuration. You simply define the data format using layline.io's own grammer language and can instantly see the result with a sample data file. No custom coding necessary.

Assets

An asset is a pre-made template for a processor, workflow or project. This is useful to components which can be reused across projects and workflows. For example when a processor is added to a workflow, you can decide on

1. whether you want to create a completely new asset that this processor is based on, or
2. whether you want to reuse an existing asset and inherit all its settings for this processor, or
3. not use an asset at all but simply enter the configuration for this one processor and never reuse any of it.

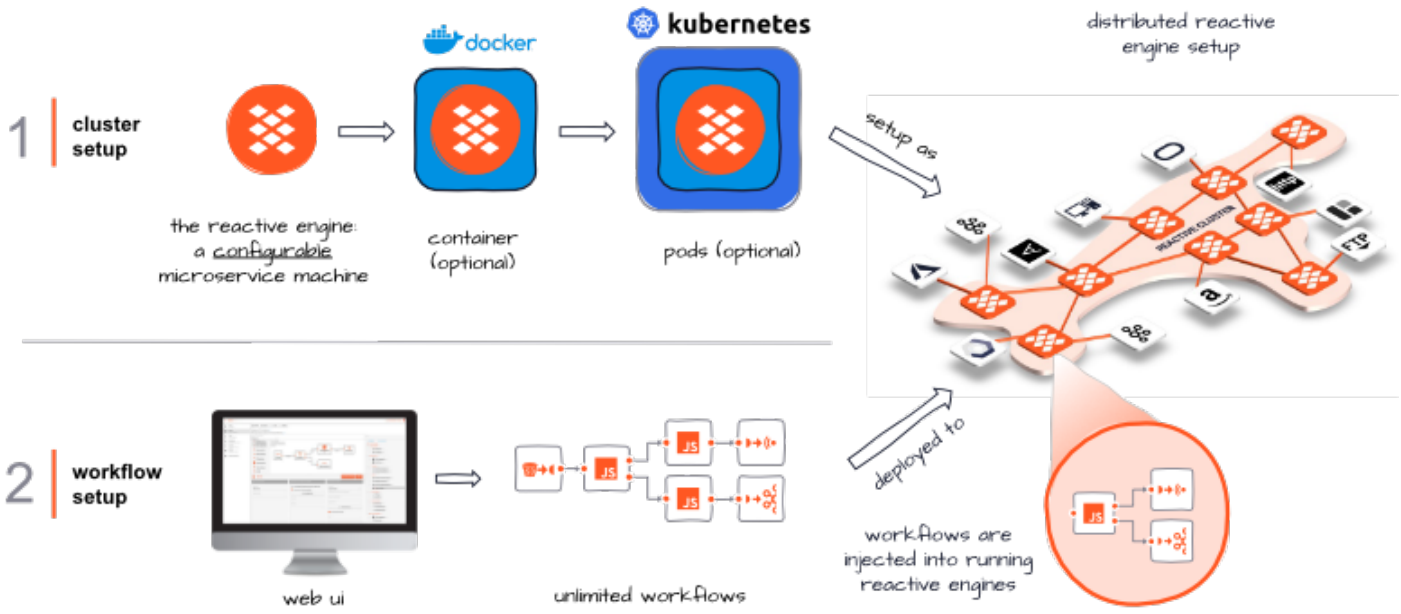
Once a processor is based on an asset, all its settings are inherited. they can be individually overwritten to deviate from the asset's standard settings.



setup

Simple two step setup process

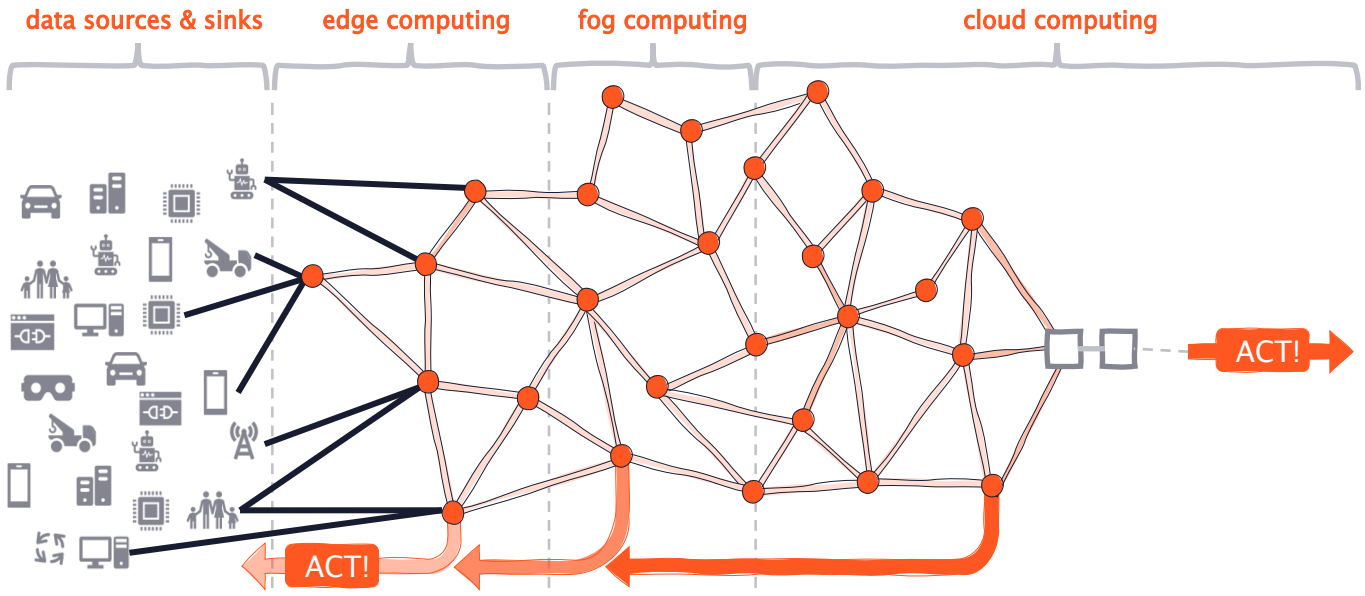
1. **Cluster setup:** Setting up a layline.io cluster involves setting up a number of nodes which each run the layline.io **reactive engine**. This can be done either by native installation, or through containerization (docker) and container management (kubernetes). The resulting layline.io cluster can be dynamically expanded or shrunk any time and at runtime.
2. **Workflow setup:** Independent of the cluster setup, the workflows can be defined and configured using the layline.io web-based configuration center. When complete the workflows can be automatically deployed to the layline.io cluster. This process can be repeated on an ongoing basis to accommodate changes.



example setup

layline.io allows a sheer endless combination of setups and use cases. The image below depicts an examplary edge computing setup. It is characterized by

- Cluster spread across geographic locations (edge / fog / cloud & data center), but maintaining one coherent virtual data mesh.
- Each node running at least one reactive engine with one or more workflows assigned to it
- Ability to analyze data close to the source of origin . This allows to react quick if necessary and only forward information as necessary.



example use cases

stream data processing

real-time streaming data scenarios with multiple distributed ingestion points. sophisticated data treatment and non-stop operation.

data transformation

transform data from one format into another. apply any sort of data enrichment, filtering and routing. feed into any target.

cep – complex event processing

typical complex event processing scenarios which require utmost flexibility, scalability and adaptability.

edge computing

deployment in sophisticated distributed environments. typical for edge computing setups where computing needs to be autonomous and spread over geographies.

data mediation

typical data mediation scenarios, involving elements of data transformation, multi-connectivity, complex custom data formats, massive data volumes.

batch data processing

traditional batch processing, but in a cloud-native architecture and at much larger scale than possible with legacy systems.

data filtering

filtering data from data streams based on custom filtering rules. combine with enrichment, routing, and transformation.

etl / elt

big data loading and transformation routines for small to very large and complex scenarios.

systems monitoring

feeding systems data from any source in real-time for the purpose of systems monitoring.

connectors / data formats

The following list shows an excerpt of the most important **connectors**:

Cloud interfaces:

- AWS
  - DynamoDB
  - Kinesis
  - lambda
  - S3
  - SNS
- Azure
  - Event Hubs
  - IoT Hubs
  - Storage Queue
- Google Cloud
  - Pub/Sub
  - Pub/Sub gRPC
  - Firebase

NoSQL:

- cassandra
- couchbase
- Kudu
- HBase
- HDFS
- Mongo
- OrientDB

Protocols:

- http
- restful
- FTP/SFTP
- IronMQ
- Java JMS
- MQTT
- SSE
- Web Sockets
- TCP
- UDP
- UDS
- gRPC

Data formats are supported by configuration. Examples are:

Structured:

- XML
- JSON
- HTML
- Binary
- Any text format
- Parquet
- Avro

Unstructured:

- MS Office formats
- Open Office
- Apple iWorks
- RTF
- PDF
- ePub
- Help formats
- Mail formats

Media:

- Audio
- Video
- Images
- Syndication

We are constantly working on adding to the data parsing capability of layline.io. Ask us if you don't find what you are looking for.

key differentiators

fast data
crazy fast data processing

streamed processing
native stream processing instead of batch

scale up and out
unparalleled horizontal and vertical scalability

massive volume
ready for the real-time data tsunami

distributed
deploy distributed from tiny to XXL platforms

self-healing
nodes take on other node's work on fail until they rejoin

self-balancing
auto-balances work across available resources

configurable
no low-level coding required. define own business logic.

payload-aware
understands payload structure by configuration. know what you work on

interested?

If we were able to spawn your interest, please don't hesitate to contact us at:

layline.io GmbH
Airport Center C
Flughafenstr. 52a,
22335 Hamburg, Germany

email: hello@layline.io

web: https://layline.io