Conduit Architecture

Here is an overview of the internal Conduit Architecture.

Conduit is split into the following layers:

- API layer
- exposes the public APIs used to communicate with Conduit. It exposes 2 types of APIs:* gRPC
- this is the main API provided by Conduit. The gRPC API definition can be found in api.proto
- , it can be used to generate code for the client.
- HTTP
- the HTTP API is generated using grpc-gateway
- and
- forwards the requests to the gRPC API. Conduit exposes an openapi definition that describes the HTTP API, which is
- also exposed through Swagger UI onhttp://localhost:8080/openapi/
- .
- Orchestration layer
- the orchestration layer is responsible for coordinating the flow of operations between the
- core services. It also takes care of transactions, making sure that changes made to specific entities are not visible
- to the outside until the whole operation succeeded. There are 3 orchestrators, each responsible for actions related
- to one of the 3 main entities - pipelines, connectors and processors.
- Core
- we regard the core to be the combination of the entity management layer and the pipeline engine. It
- provides functionality to the orchestrator layer and does not concern itself with where requests come from and how
- single operations are combined into more complex flows.* Entity management
- this layer is concerned with the creation, editing, deletion and storage of the main
- entities. You can think about this as a simple CRUD layer. It can be split up further using the main entities:* Pipeline
- this is the central entity managed by the Pipeline Service that ties together all other components.
- A pipeline contains the configuration that defines how pipeline nodes should be connected together in a running
- pipeline. It has references to at least one source and one destination connector and zero or multiple processors,
- a pipeline that does not meet the criteria is regarded as incomplete and can't be started. A pipeline can be
- either running, stopped or degraded (stopped because of an error). The pipeline can only be edited if it's not in
- a running state.

- Connector
- a connector takes care of receiving or forwarding records to connector plugins, depending on its
- type (source or destination). It is also responsible for tracking the connector state as records flow through it.
- The Connector Service manages the creation of connectors and permanently stores them in the Connector Store. A
- connector can be configured to reference a number of processors, which will be executed only on records that are
- received from or forwarded to that specific connector.* Connector Plugin
- interfaces with Conduit on one side, and with the standalone connector plugin on the
- other and facilitates the communication between them. A standalone connector plugin is a separate process that
- implements the interface defined inconduit-connector-protocol
- and provides the read/write
- functionality for a specific resource (e.g. a database).
- Processor
- processors are stateless components that operate on a single record and can execute arbitrary
- actions before forwarding the record to the next node in the pipeline. A processor can also choose to drop a
- record without forwarding it. They can be attached either to a connector or to a pipeline, based on that they are
- either processing only records that flow from/to a connector or all records that flow through a pipeline.* Processor Plugin
- interfaces with Conduit on one side, and with the standalone processor plugin on the

- other and facilitates the communication between them. A standalone processor plugin is a WASM binary that
- implements the interface defined in conduit-processor-sdk
- and provides the logic for processing
- a record (e.g. transforming its content).

- Pipeline Engine
- the pipeline engine consists of nodes that can be connected together with Go channels to form
- a data pipeline.* Node
- a node is a lightweight component that runs in its own goroutine and runs as long as the incoming channel
- is open. As soon as the previous node stops forwarding records and closes its out channel, the current node also
- stops running and closes its out channel. This continues down the pipeline until all records are drained and the
- pipeline gracefully stops. In case a node experiences an error all other nodes will be notified and stop running
- as soon as possible without draining the pipeline.

- Persistence
- this layer is used directly by the orchestration layer and indirectly by the core layer (through
- stores) to persist data. It provides the functionality of creating transactions and storing, retrieving and deleting
- arbitrary data like configurations or state.

Edit this page Previous What is Conduit? Next Vocabulary