

12 Factor Streaming Data Apps on Kubernetes



Presenters



Andrew Stevenson
CTO, Lenses.io



Fran Perez
Backend Engineer, Lenses.io

Open-Source tooling for Apache Kafka & Kubernetes



>1 Million
Downloads

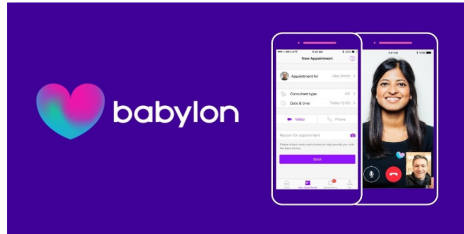


>5000
Github Stars

Commercial offering for Kafka & Kubernetes



We Support Strategic Real-time Data Projects



BABYLON HEALTH

Delivering affordable healthcare to everyone on earth through AI-driven chatbots

Affordable Healthcare



VORTEXA

Tracks the movement of over \$7 trillion worth of seaborne oil-based products in real-time

Analytics



ADUNO GRUPPE

Increasing customer engagement in their marketing campaigns

Customer Engagement

Lenses.io Mission

Reduce the **pain, cost & complexity** of
modern data platforms.



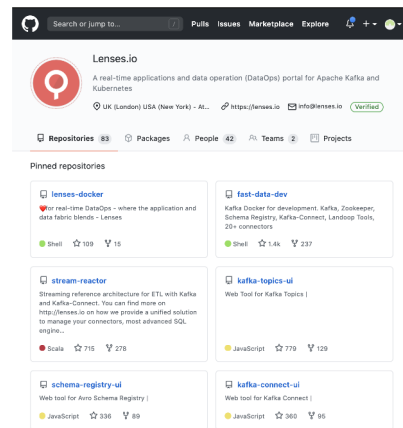
lenses.io/start

All-in-one Kafka + Lenses.io developer Docker box



github.com/lensesio

Open-Source tooling & connectors

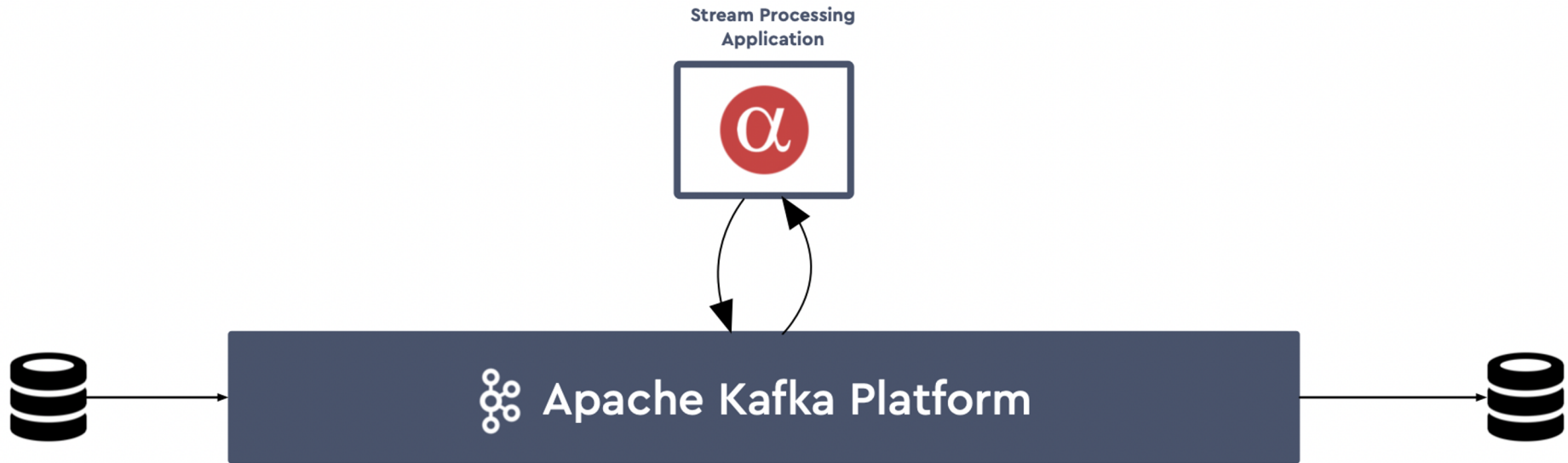


launchpass.com/lensesio

Agenda

- ✓ Overview of a DataOps streaming platform
- ✓ Demo our deployment framework that takes the Ops out of DataOps
- ✓ How we built it and how it scales to any type of application

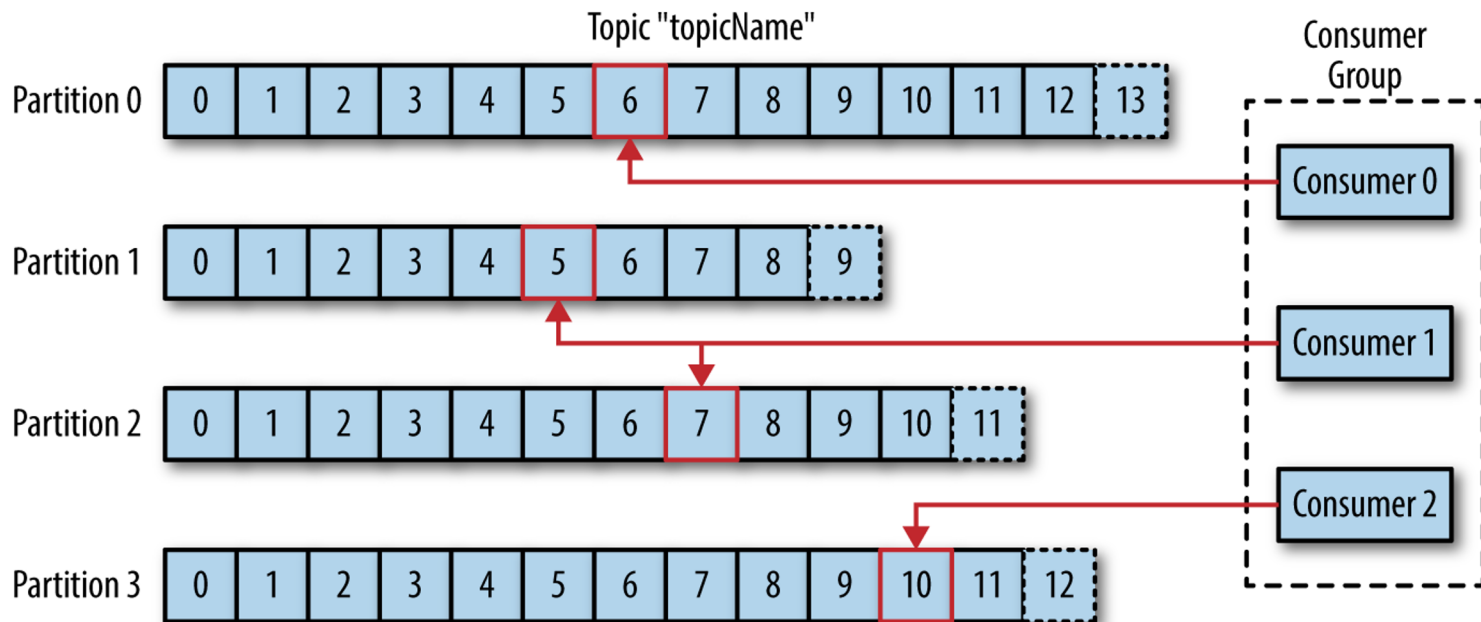
Data platform



What is Kafka?

distributed commit log

Distributed commit log?



*Oreilly - Kafka definitive guide



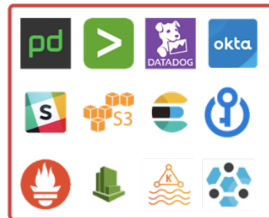
Self-Service
Administration

Data Discovery
& Exploration

Data Processing
& Integration

Monitoring
& Observability

Security
& Governance



Any Kafka

Your data platform



Any Kubernetes

If not kafka, what are we deploying?

 Q LENSES

Self-Service
Administration

Data Discovery
& Exploration

Data Processing
& Integration

Monitoring
& Observability

Data Entities

Data Alerting Rules

Data Policies

Data Applications



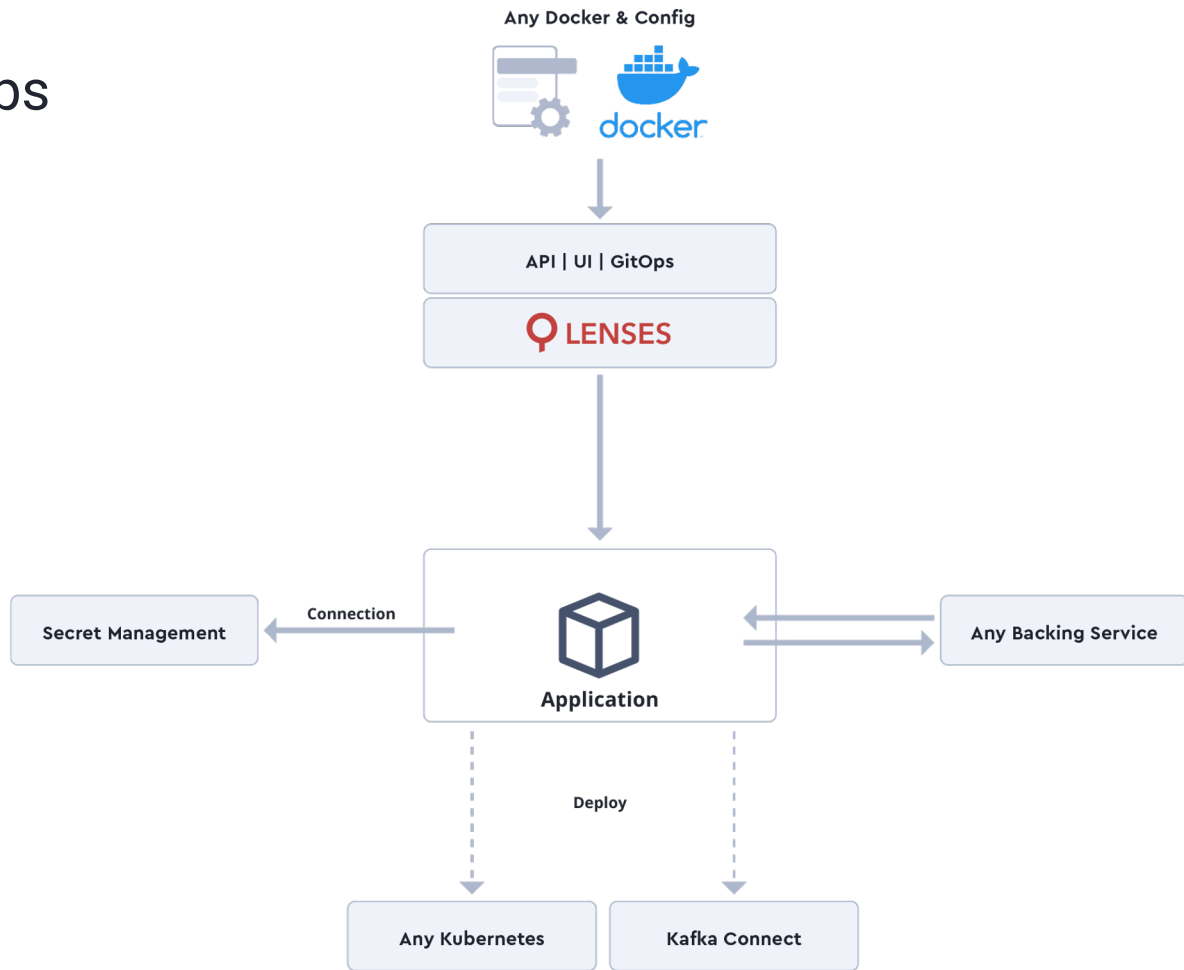
Any Kafka

Your data platform

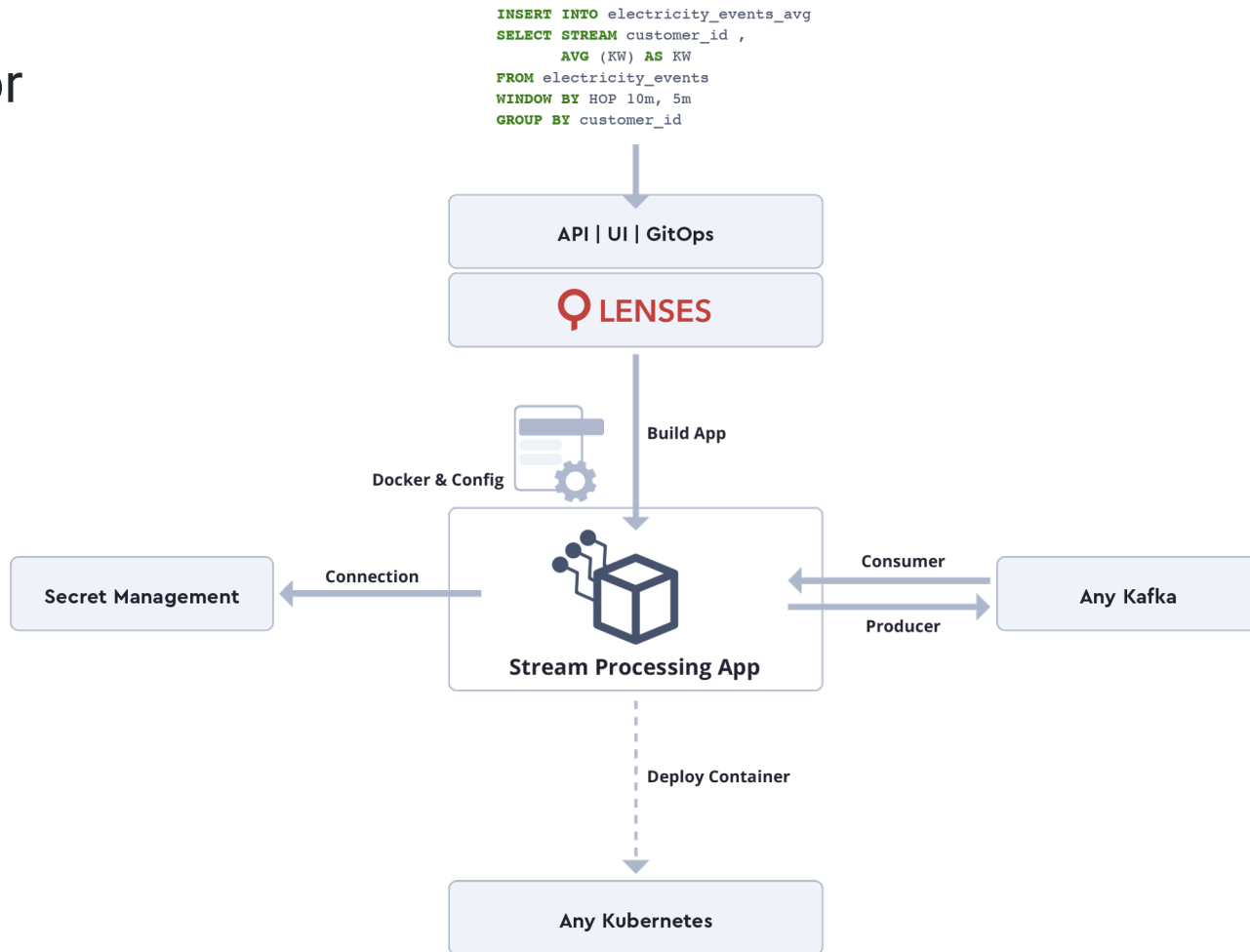


Any Kubernetes

Data intensive apps with Lenses



Streaming SQL for Kubernetes

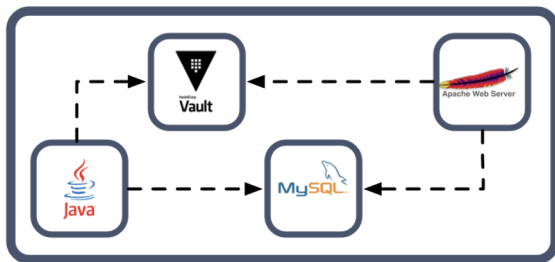




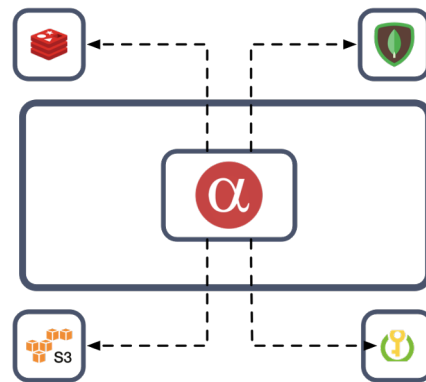
3 minutes demo of deploying a streaming
app

What's special about real-time apps?

- ✓ Unbounded data
- ✓ Many sources
- ✓ React to data
- ✓ State
- ✓ Inputs/Outputs
- ✓ Data access
- ✓ Complex security models
- ✓ Connection management



Traditional Self-Contained Microservice



Stream Processing Platform

Platform as a Service



What does it mean

What should we care about

New challenges

12 Factor apps manifest

The Twelve-Factor App

Adam Wiggins

Introduction

In the modern era, software is commonly delivered as a service called *web apps*, or *software-as-a-service*. The twelve-factor app is a methodology for building software-as-a-service apps that:

- Use declarative formats for setup automation, to minimize time and cost for new developers joining the project;
- Have a clean contract with the underlying operating system, offering maximum portability between execution environments;
- Are suitable for deployment on modern cloud platforms, obviating the need for servers and systems administration;
- Minimize divergence between development and production, enabling *continuous deployment* for maximum agility;
- And can scale up without significant changes to tooling, architecture, or development

Page 1

The Twelve-Factor App

I. Codebase

One codebase tracked in revision control, many deploys

A twelve-factor app is always tracked in a version control system, such as Git, Mercurial, or Subversion. A copy of the revision tracking database is known as a *code repository*, often shortened to *code repo* or just *repo*.

A *codebase* is any single repo (in a centralized revision control system like Subversion), or any set of repos who share a root commit (in a decentralized revision control system like Git).

One codebase maps to many deploys

There is always a one-to-one correlation between the codebase and the app:

- If there are multiple codebases, it's not an app – it's a distributed system. Each component in a distributed system is an app, and each can individually comply with twelve-factor.
- Multiple apps sharing the same code is a viola-

Page 5

The Twelve-Factor App

workloads by assigning each type of work to a *process type*. For example, HTTP requests may be handled by a web process, and long-running background tasks handled by a worker process.

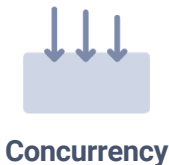
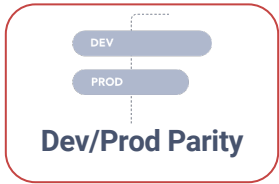
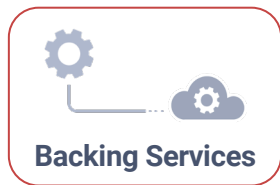
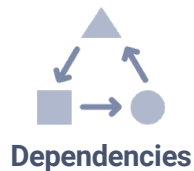
This does not exclude individual processes from handling their own internal multiplexing, via threads inside the runtime VM, or the async/evented model found in tools such as EventMachine, Twisted, or Node.js. But an individual VM can only grow so large (vertical scale), so the application must also be able to span multiple processes running on multiple physical machines.

The process model truly shines when it comes time to scale out. The share-nothing, horizontally partitionable nature of twelve-factor app processes means that adding more concurrency is a simple and reliable operation. The array of process types and number of processes of each type is known as the *process formation*.

Scale is expressed as running processes, workload diversity is expressed as process types.

In the twelve-factor app, processes are a *first class citizen*. Processes in the twelve-factor app take strong cues from the unix process model for running service daemons. Using this model, the developer can architect their app to handle diverse

Page 29



12 Factor Apps Principles

How Lenses approaches it

Connections

Secret Providers

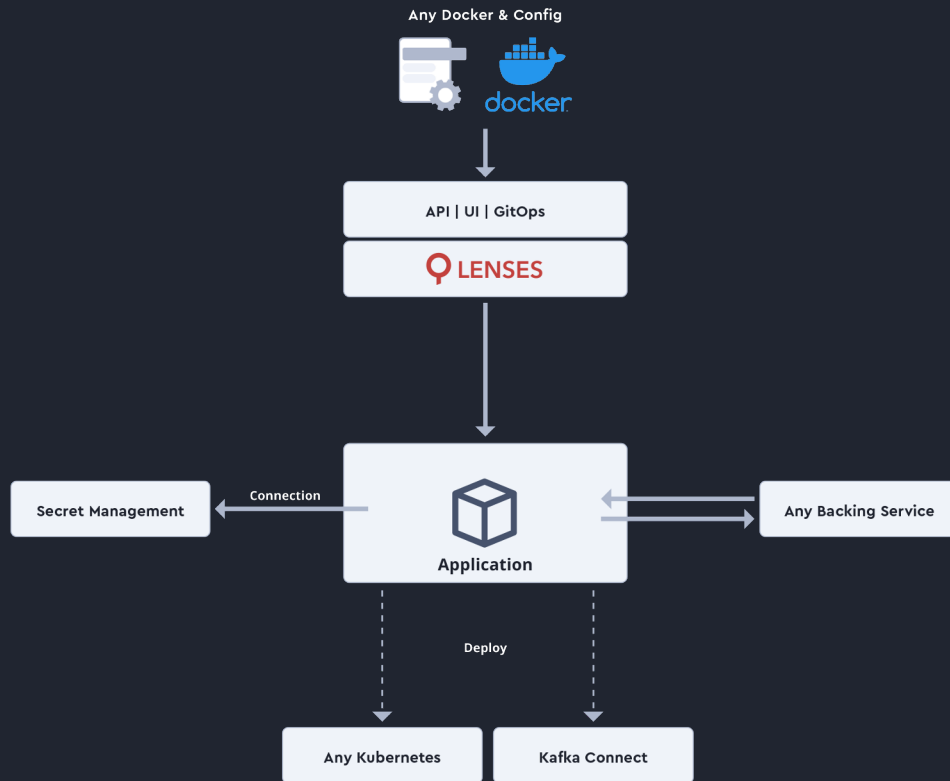
Applications

**Deployment
Targets**

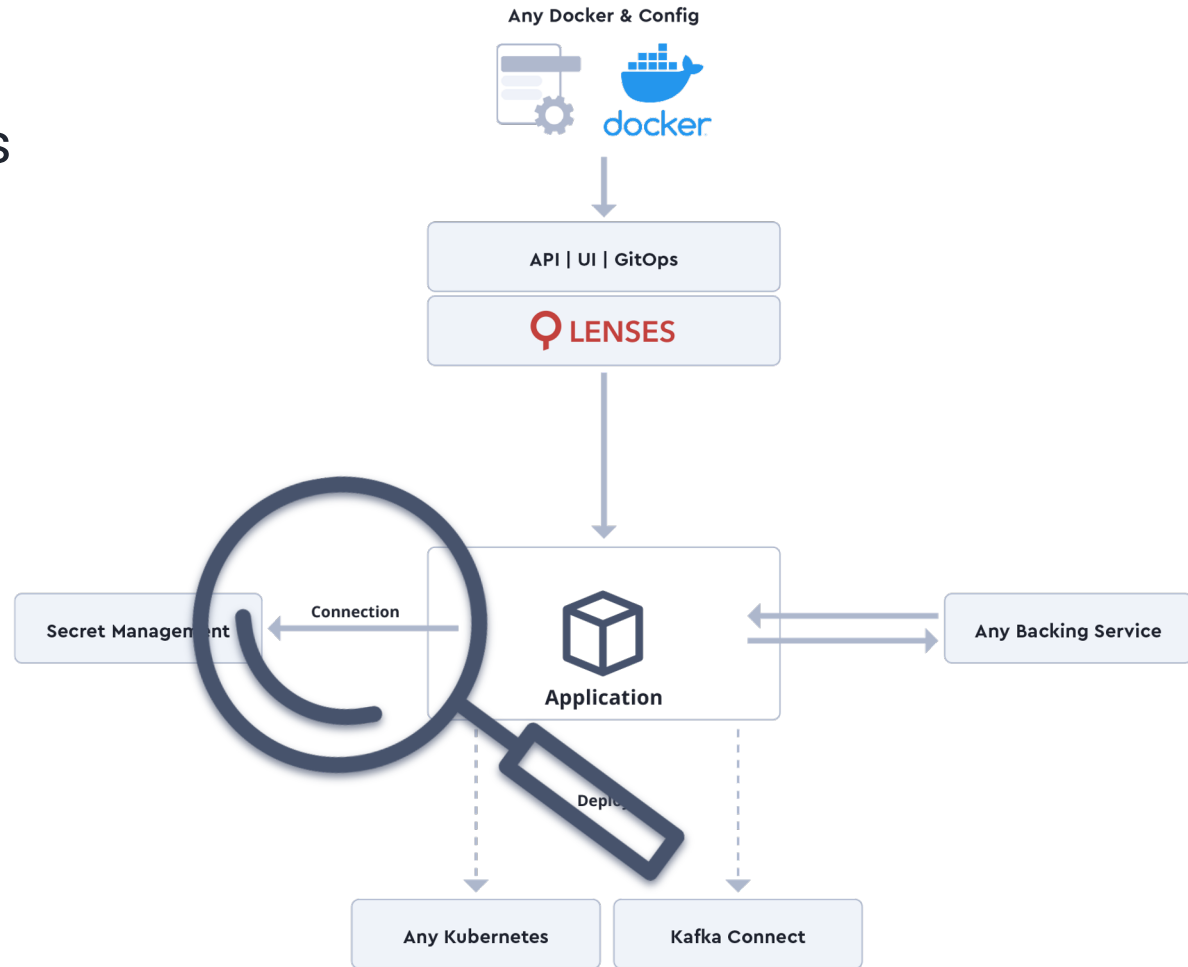
Templating systems <> Streaming apps

Common concerns for any app:

- ✓ Input params
- ✓ Data sources
- ✓ Data sinks
- ✓ Security
- ✓ App image to run
- ✓ Scale
- ✓ Liveness probe
- ...



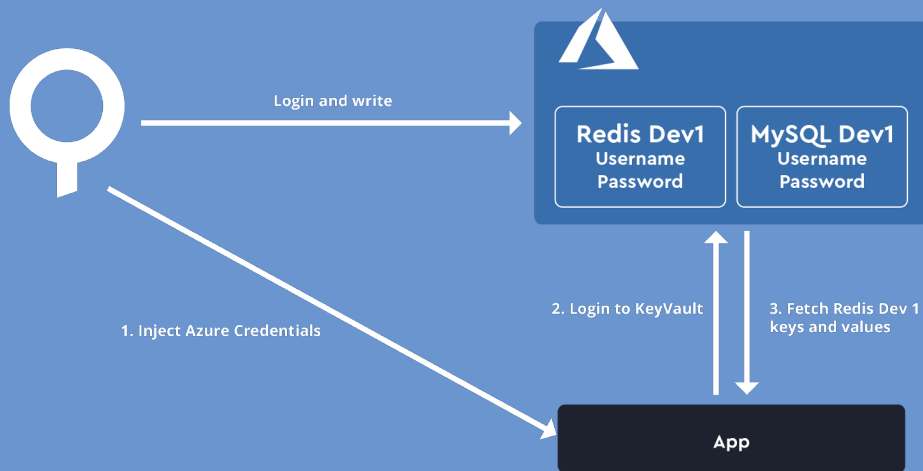
Secret Providers



Secret Providers

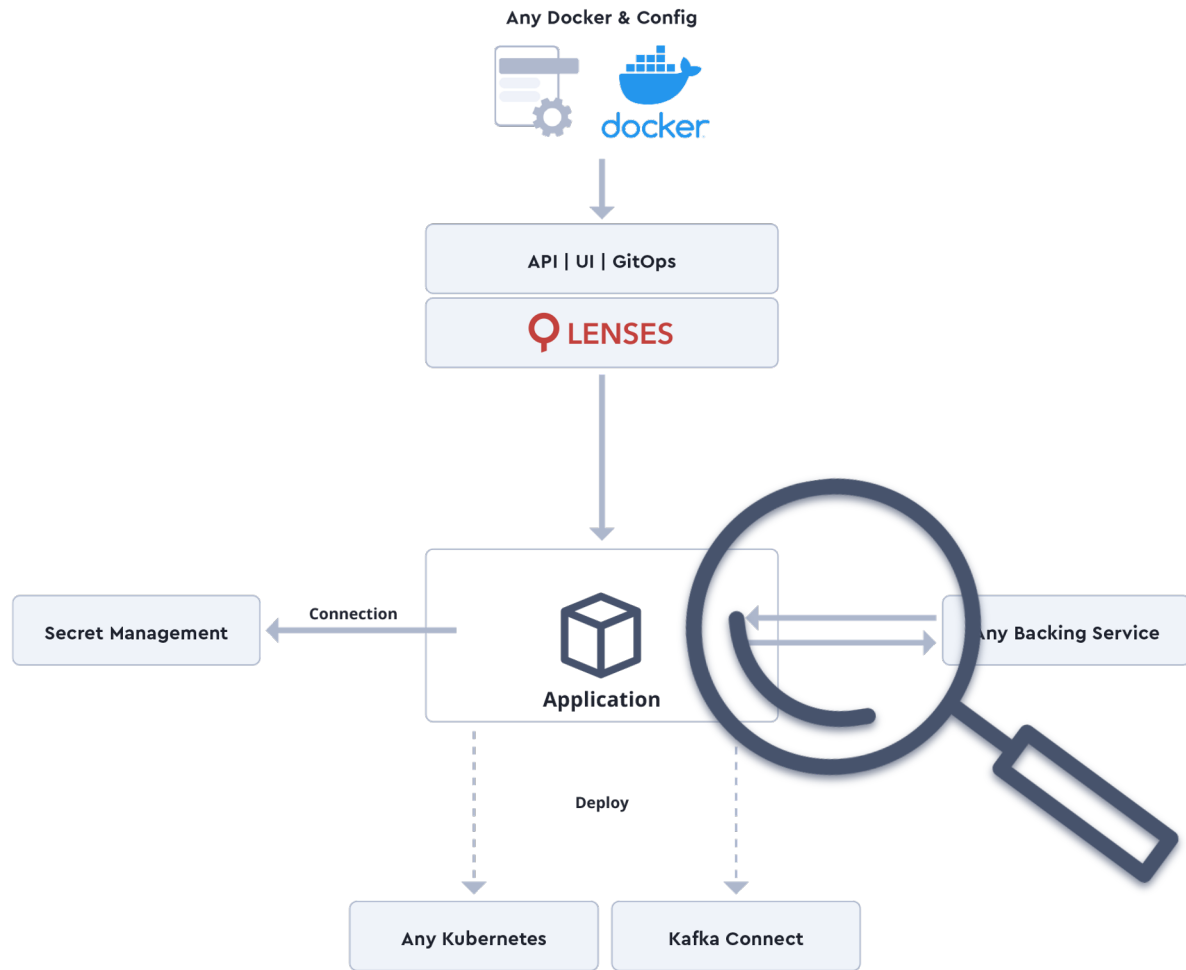
Secret providers. Who (where in the end) is going to keep secrets e.g.:

- ✓ Lenses itself
- ✓ AzureKeyVault
- ✓ HashicorpVault
- ✓ ...



SECRET_KEY_NAME=[PROVIDER]:[VAULT_URL]:[LOOKUP_KEY]:mounted:type

Connections

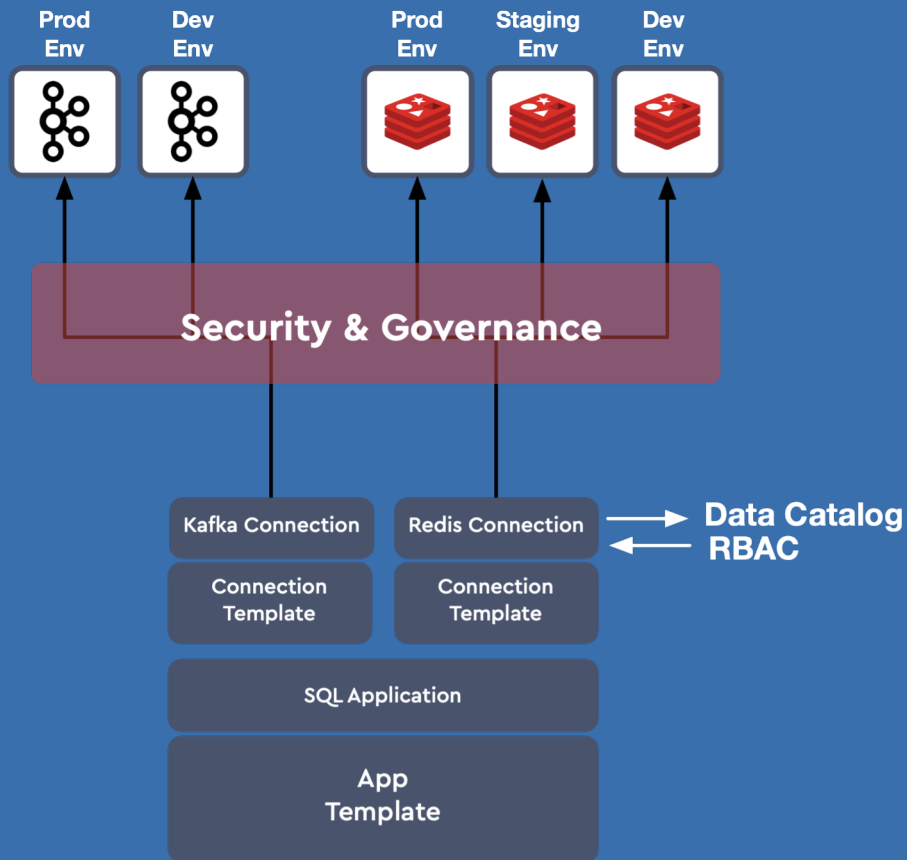


Connections

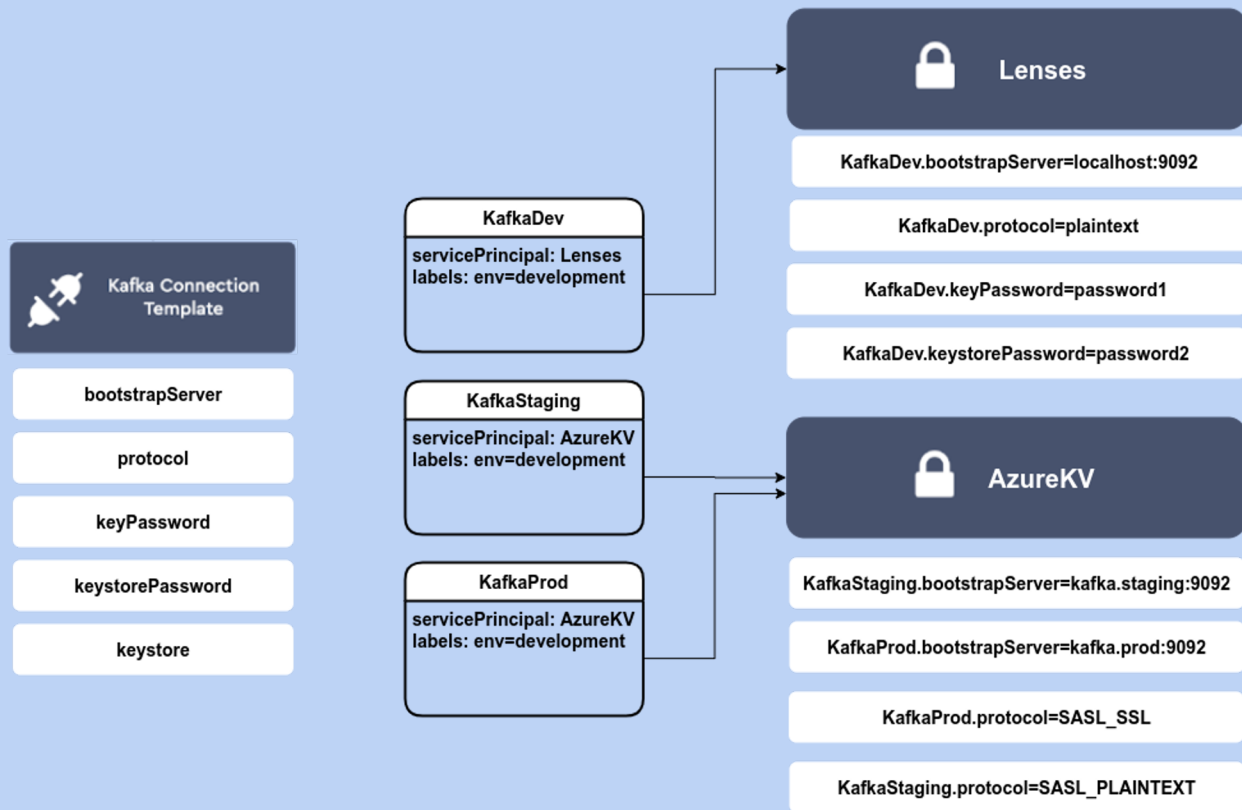
Connection descriptor. Properties needed to connect to a system e.g.:

- ✓ Kafka brokers
- ✓ DB connection string

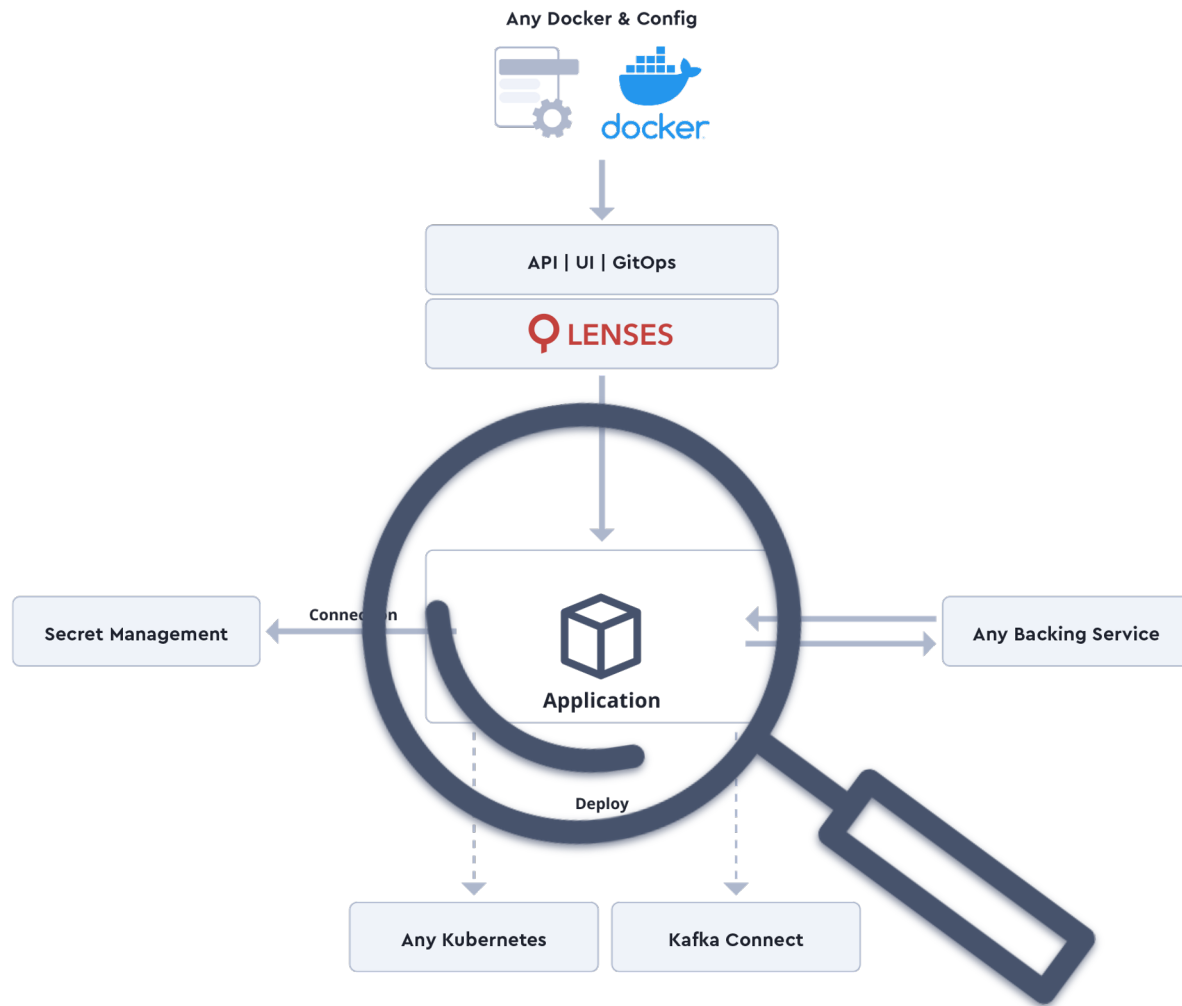
*Connections are referenced by Names



Storing connection details



Applications



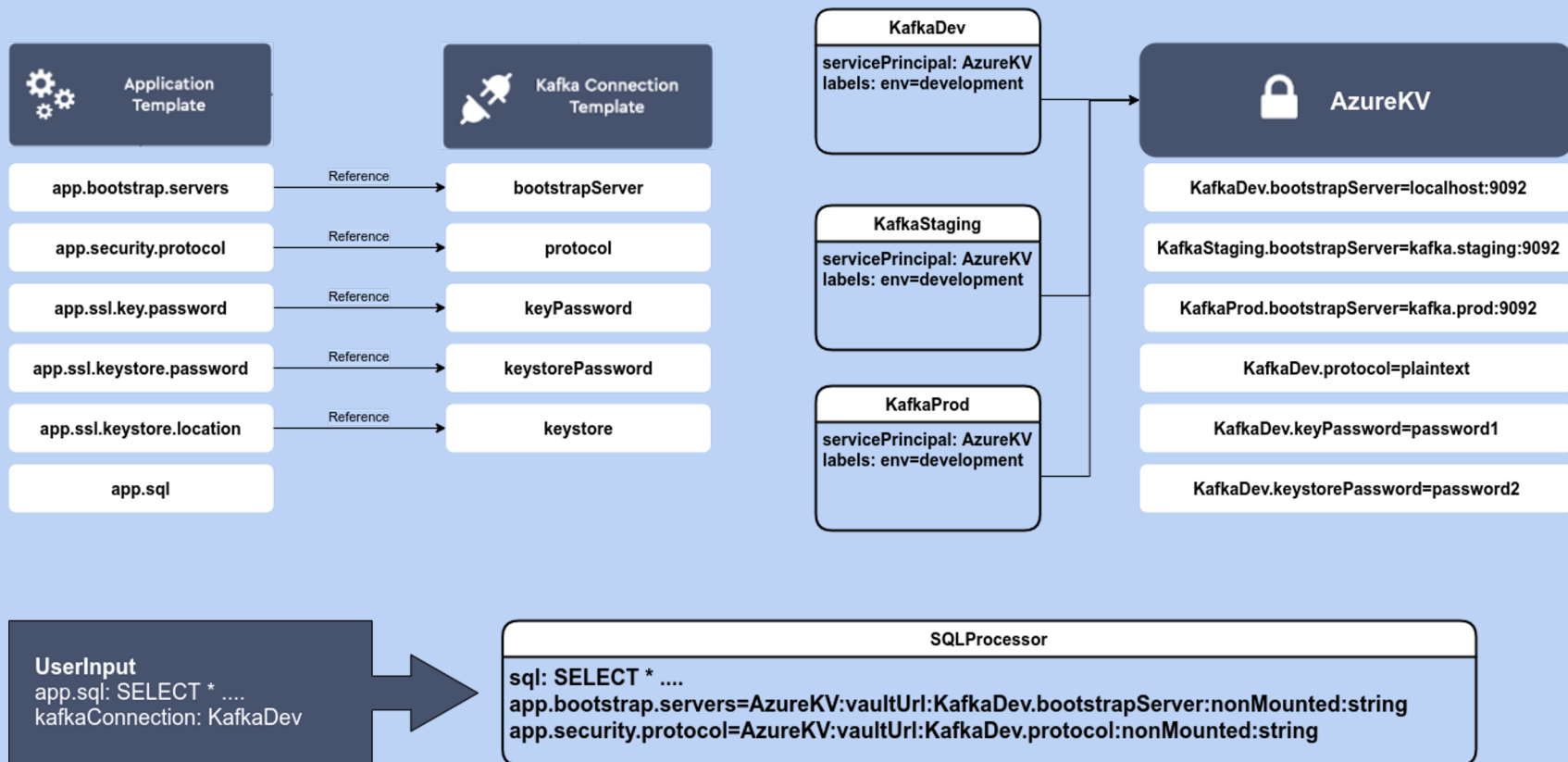
Applications

Application descriptor. Properties needed by an Application to be executed, including references to Connection instances based on existing templates e.g.:

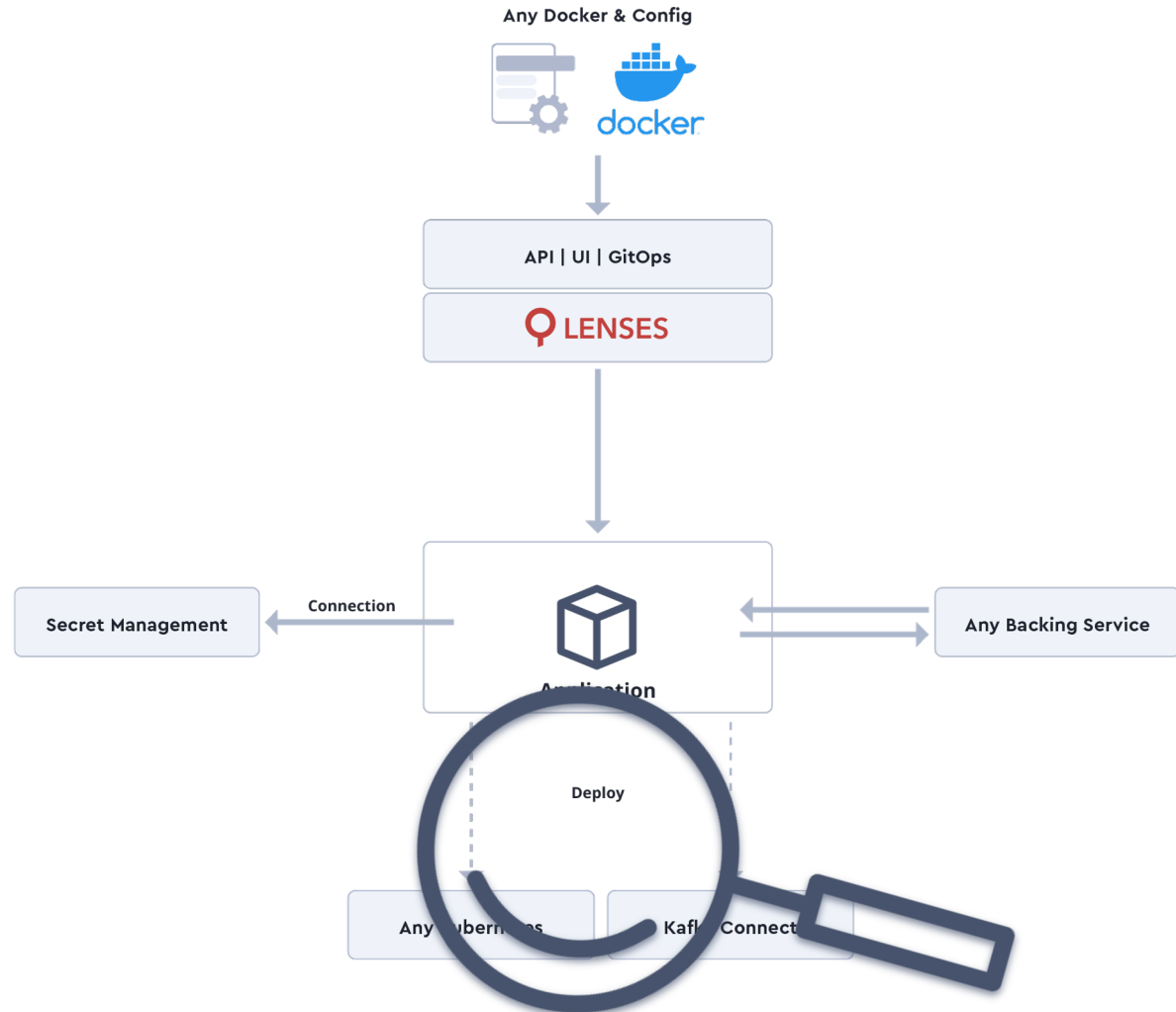
- ✓ SQL query for SQL Processors
- ✓ KCQL query for Kafka Connector
- ✓ Twitter bot: hashtag to listen to
- ✓ Twitter bot: twitter connection as Connection reference

```
name: SQLProcessor
author: Lenses.io
apLanguage: Scala
enabled: true
description: SQL Processor.
templateVersion: 1
createdBy: lenses
category: Application
configurationKeys:
- key: kafka.bootstrap.servers
  mounted: false
  required: true
  connectionReference:
    templateName: Kafka
    propertyKey: kafkaBootstrapServers
  category: Reference
  createdBy: lenses
  displayName: Kafka Bootstrap Servers
  placeholder: PLAINTEXT://host1:port1,SSL://host2:port2
  description: Comma separated list of protocol://host:port to use for initial connection
    to Kafka
```

Wiring Apps with real properties



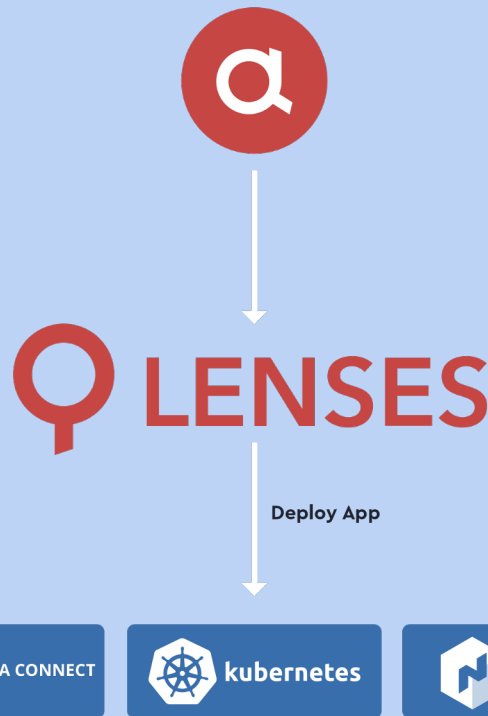
Deployments



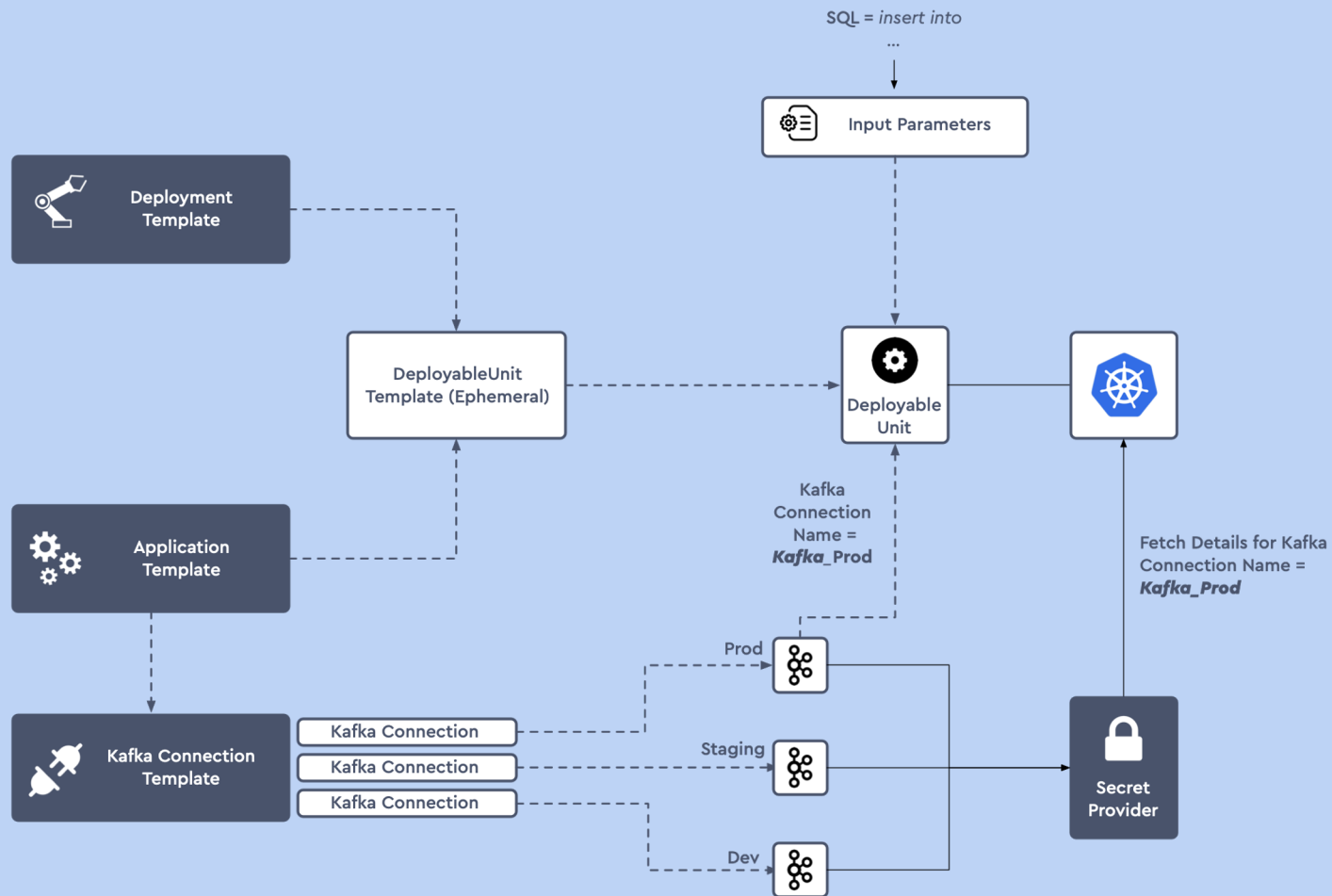
Deployments

Deployment descriptor. Properties needed to deploy an application e.g.:

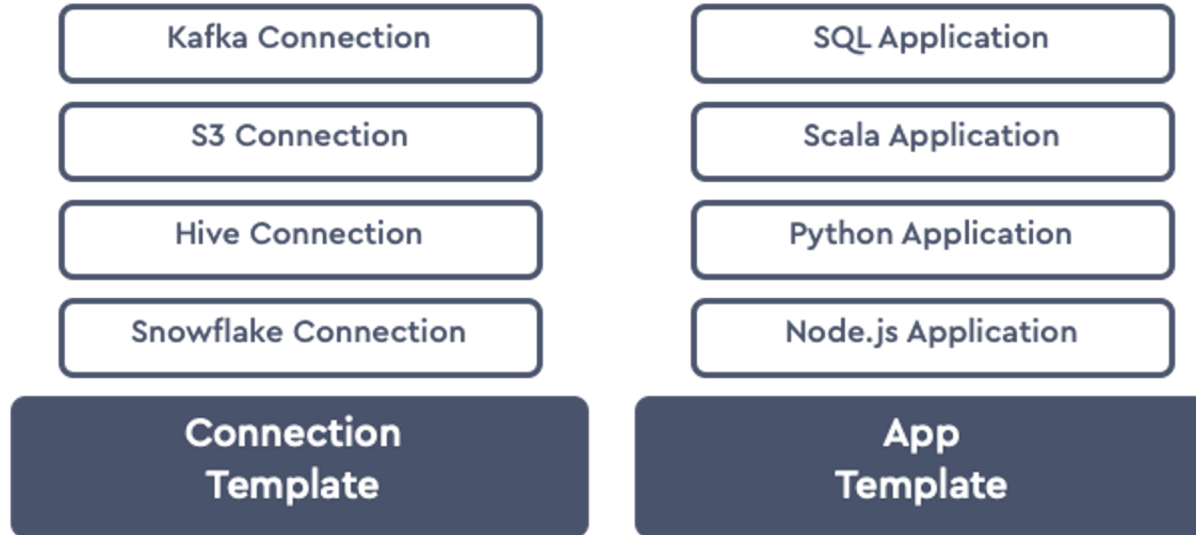
- ✓ Docker Image
- ✓ Number of runners
- ✓ Namespace (kubernetes)
- ✓ Shutdown hook
- ✓ Liveness probe



Let's cook

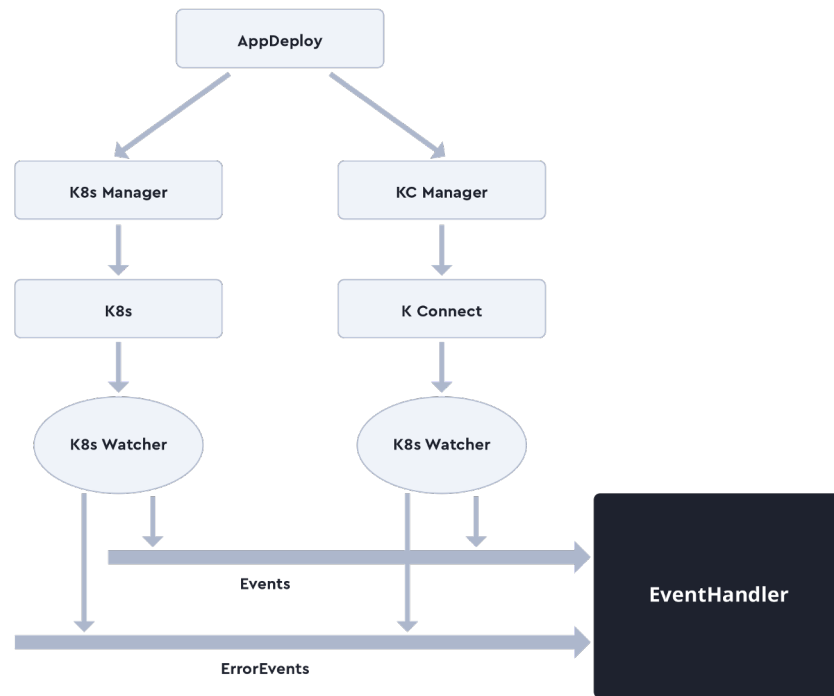


Scalability - Can be for any app not just SQL processors

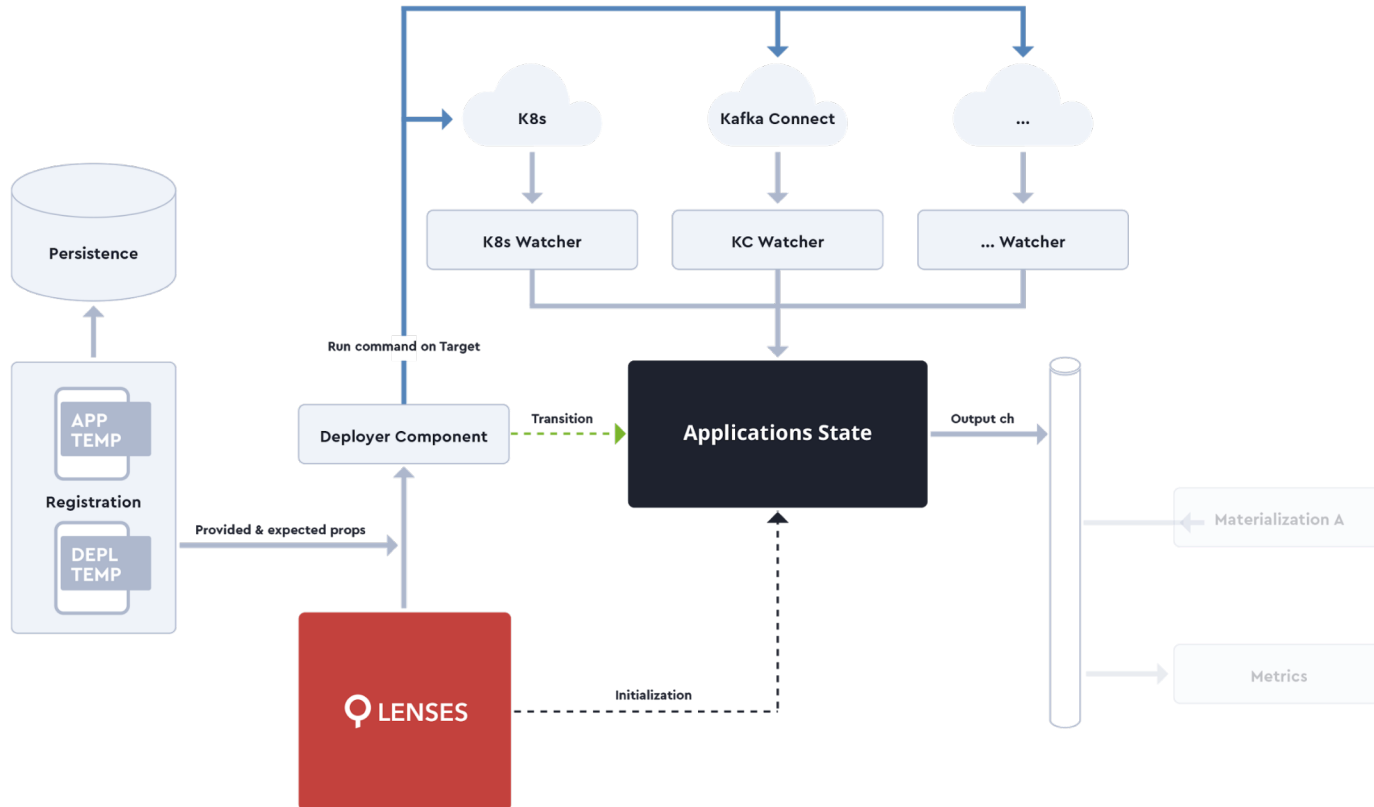


Observability

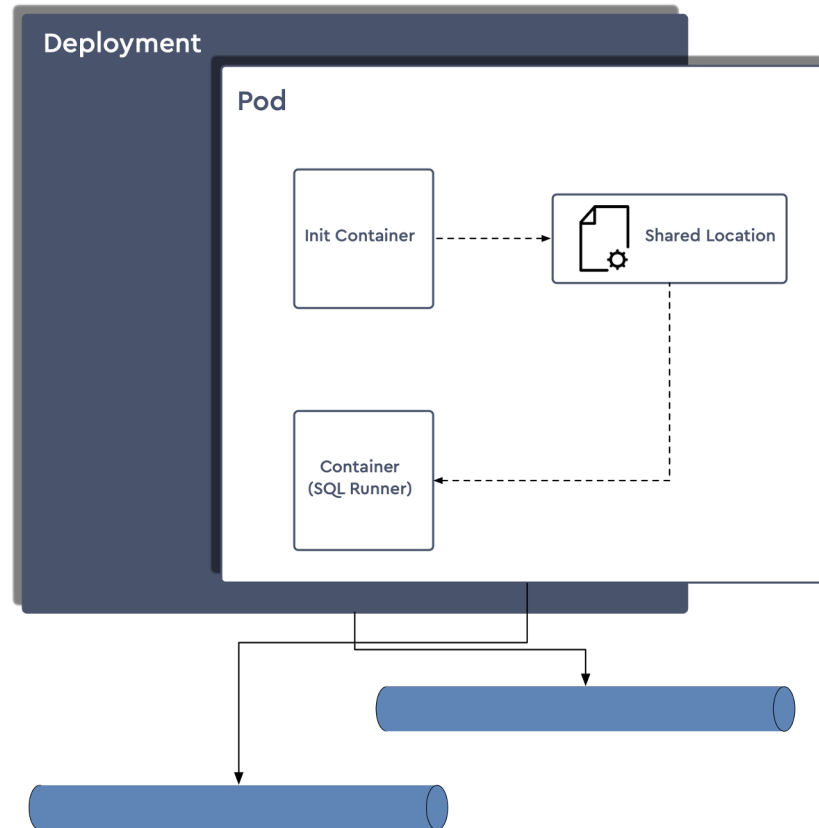
- ✓ We (Lenses) watch what's happening with the apps we deployed
- ✓ We can also expose through our templates ports where the application is already emitting more biz specific metrics



Architecture Overview (draft, needs modifications)



Kubernetes Write side (draft, needs modifications.)



Dev Experience

- ✓ Import / Export applications (enable GitOps)
- ✓ Move apps without worrying about knowing backend service
- ✓ Promotion workflows eg. from Dev -> Prod
- ✓ Governance & security
- ✓ Streaming Apps marketplace
- ✓ Enable moving apps around environments by running Lenses-cli commands (or clicks if you are using our UI)

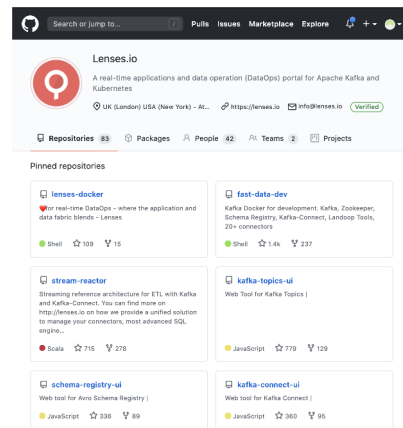
lenses.io/start

All-in-one Kafka + Lenses.io developer Docker box



github.com/lensesio

Open-Source tooling & connectors



launchpass.com/lensesio

**THANK
YOU**