LITTLEHORSE

Local Development Tips and Tricks

Become a Local Development Jedi with Strimzi and KIND

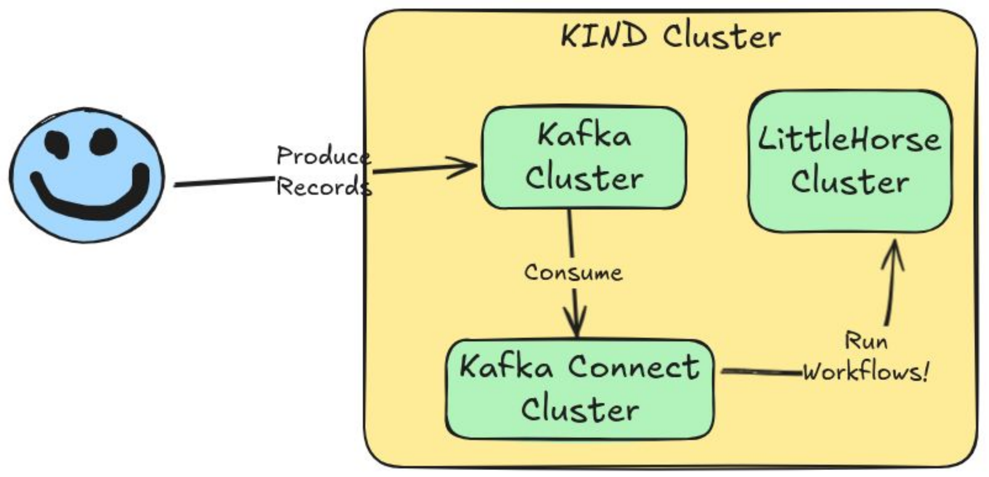
Colt McNealy
Founder, www.littlehorse.io

About Me

- Strimzi user since 2020 (two jobs ago!)
- Founder of LittleHorse Enterprises LLC
- First exposure to Kafka: the Strimzi docs!
- Yaml > bin/kafka-*.sh

The Challenge

- LittleHorse Kafka Connector: read a Kafka Topic, run workflows.
- Want to test locally in KIND: Kubernetes in Docker.
- Problems:
  - Kafka Connect plugins.
  - External Access to the Kafka cluster for quick dev cycles.



The Plan

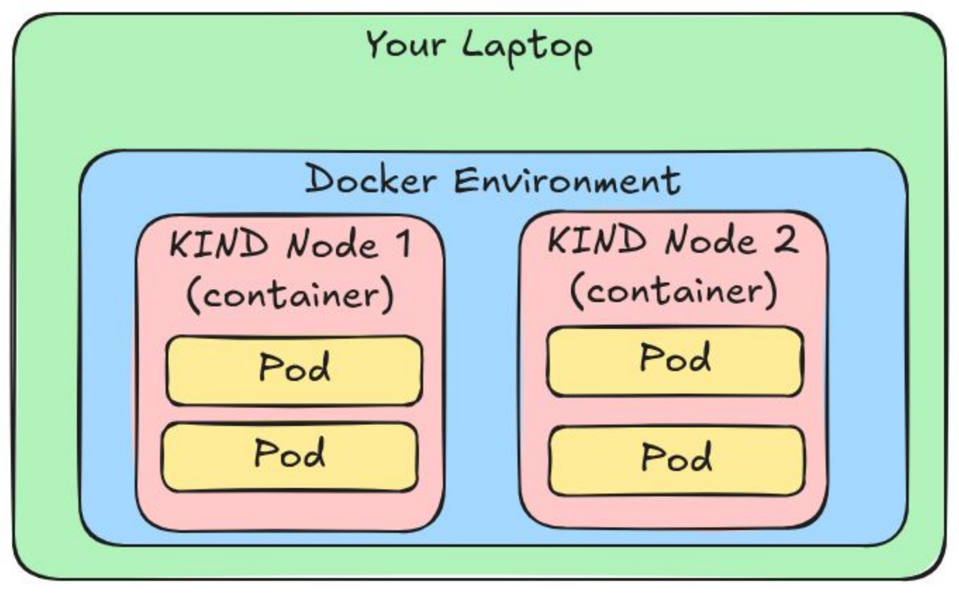
- Deploy Kafka using Strimzi in KIND
- Run Kafka Connect using the LittleHorse Kafka Connector.
- Publish messages to Kafka (from outside the KIND cluster) that trigger workflows in LittleHorse to test the connector!
- Everything local dev, all open source.

LITTLEHORSE

Envoy Gateway

KIND: Kubernetes in Docker

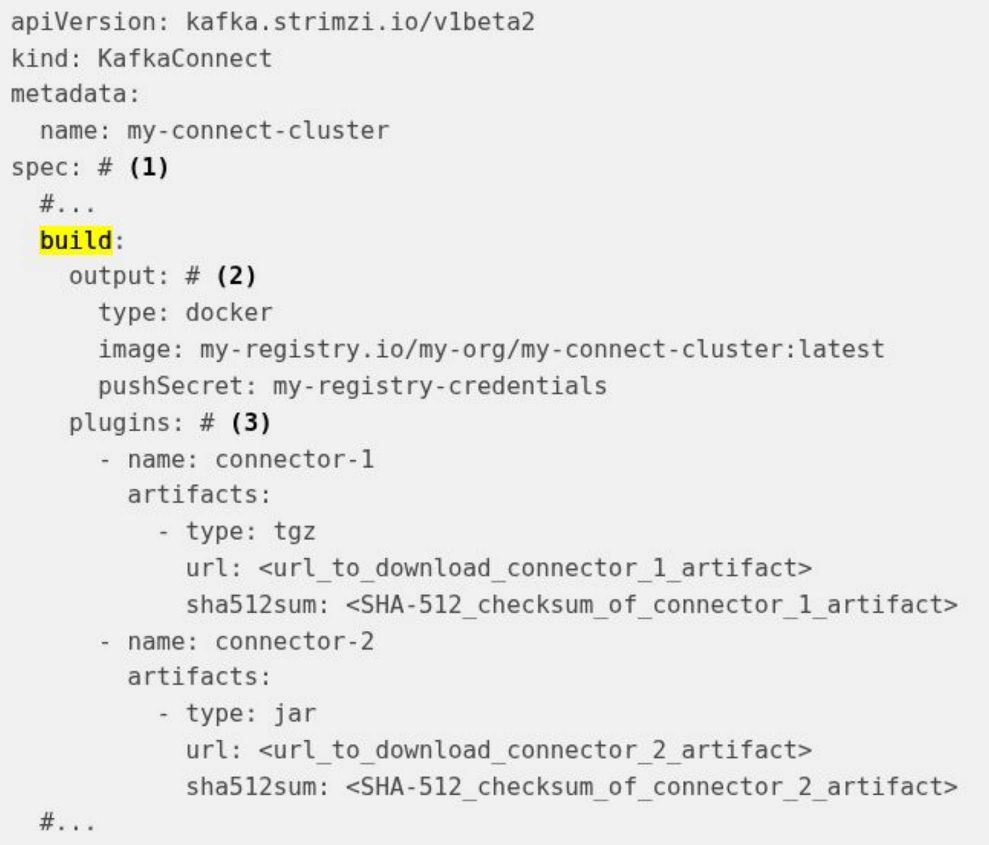
- Tool to run Kubernetes locally (Minikube alternative)
- Kubernetes nodes are just docker containers!
- Allows flexibility: docker networking, custom port mapping.
- Also friendly to running tests in pipelines. We are working on this in our K8s Operator!



Challenge 1: Kafka Connect

Getting Access to Plugins

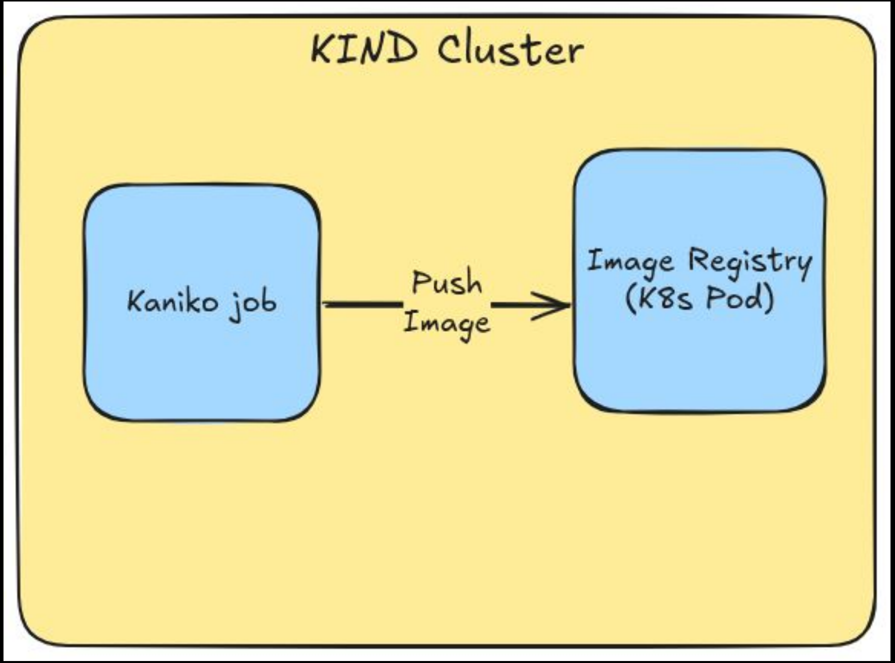
- Every Kafka Connector is a plugin—a Java class loaded at runtime.
- Need to add Connector Plugins to the class path of the Connect Workers.
- Strimzi ‘KafkaConnect’ has a wonderful “build” feature!
- Upcoming Image Volume feature, not available yet.



```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnect
metadata:
  name: my-connect-cluster
spec: # (1)
  #...
  build:
    output: # (2)
      type: docker
      image: my-registry.io/my-org/my-connect-cluster:latest
      pushSecret: my-registry-credentials
    plugins: # (3)
      - name: connector-1
        artifacts:
          - type: tgz
            url: <url_to_download_connector_1_artifact>
            sha512sum: <SHA-512_checksum_of_connector_1_artifact>
      - name: connector-2
        artifacts:
          - type: jar
            url: <url_to_download_connector_2_artifact>
            sha512sum: <SHA-512_checksum_of_connector_2_artifact>
  #...
```

Attempt: Docker Registry in K8s

- Using external service: not exactly local development!
- First solution attempt: deploy Docker Registry inside Kubernetes.
- Kaniko build succeeded!
- KIND was unable to deploy the Connect pods, though):

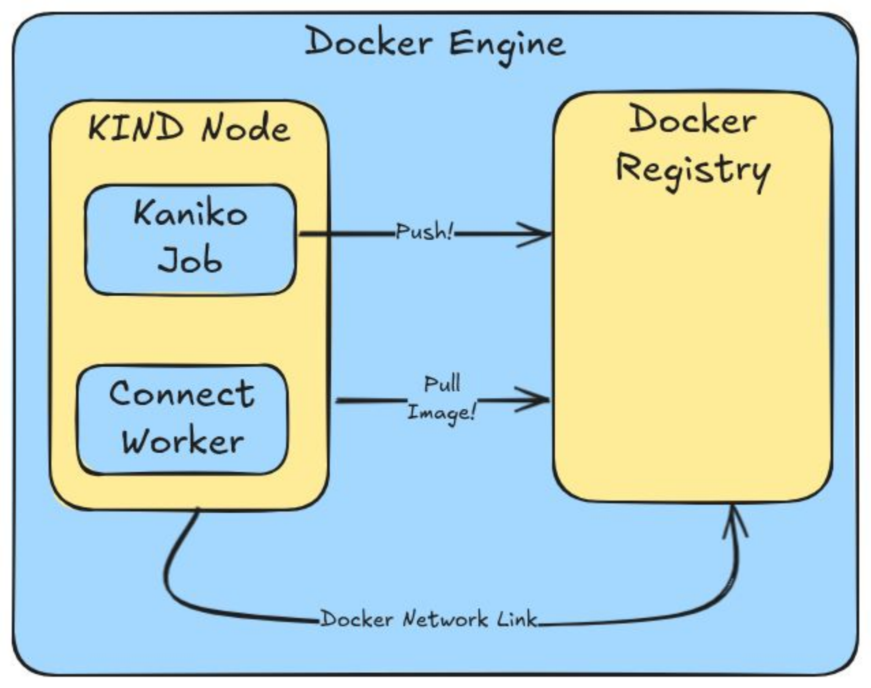


Solution: Docker Registry in Docker

- Docker Registry runs as a normal container (as a sibling to the KIND nodes)

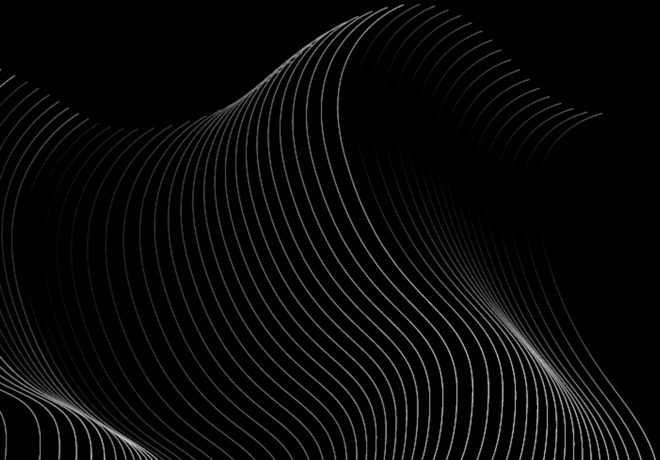
- Docker network link + some KIND magic to set the same host name

- Now, Kaniko can still push and KIND is able to find the image!



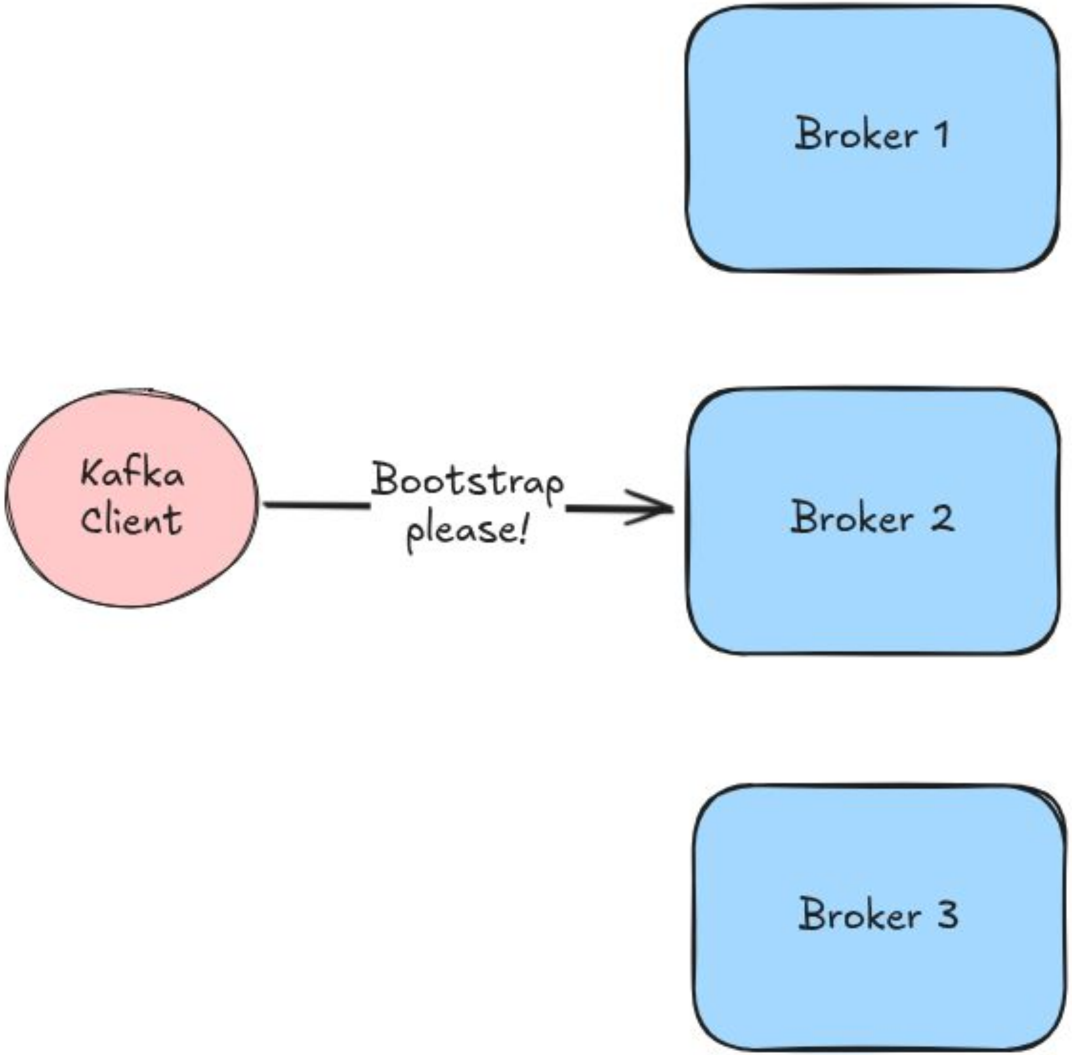
Live Demo (Uh Oh!)

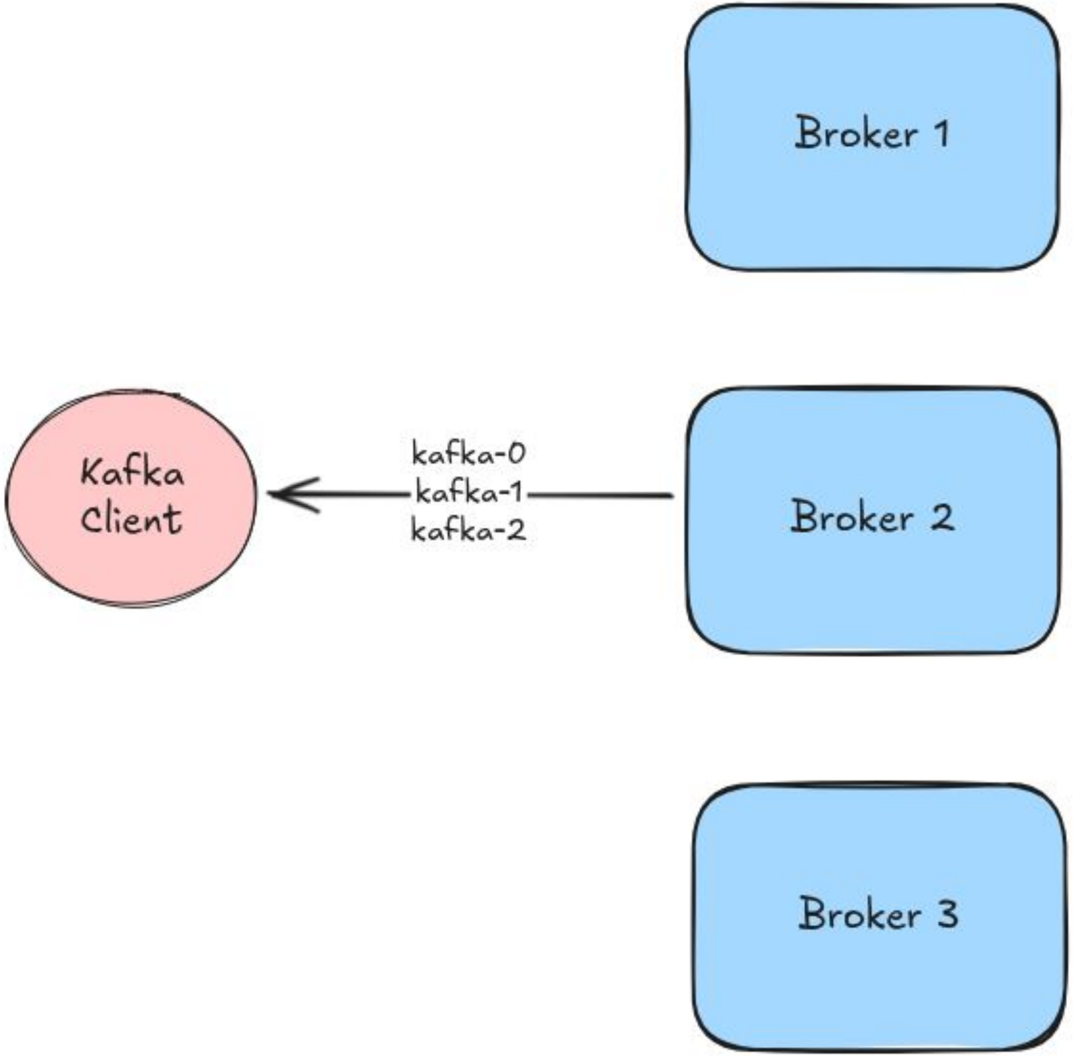
Challenge 2: External Access



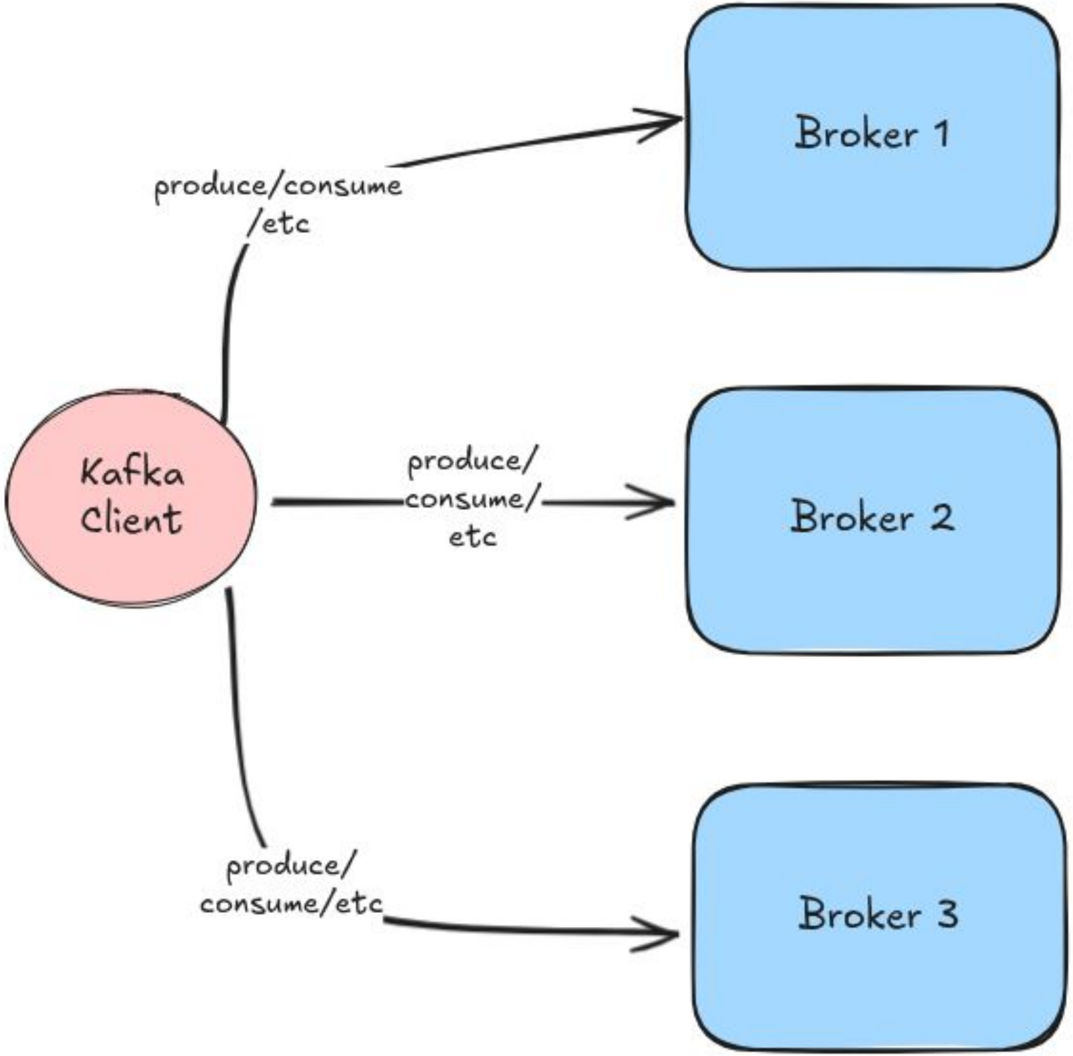
Kafka Ingress Challenges

- Ingress is built for HTTP 1 traffic.
- Standard Service routing assumes that all Pods are the same.
- Kafka’s bootstrap protocol relies on clients addressing brokers individually.
- Kafka is not HTTP!


Broker 1
Kafka Client
Bootstrap please!
Broker 2
Broker 3



LITTLEHORSE



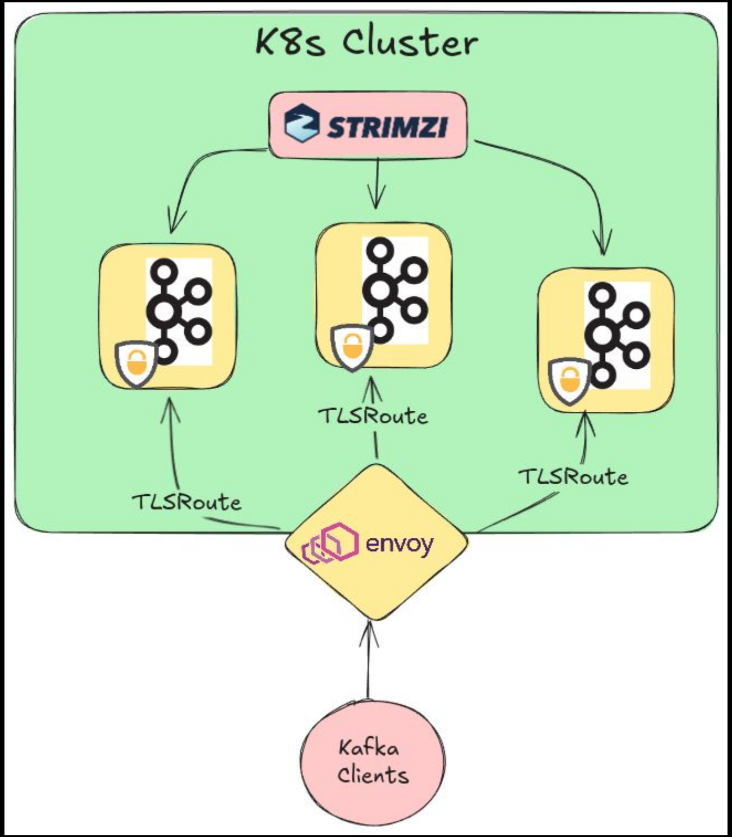
LITTLEHORSE

Using the Gateway API

- Gateway API has native support for TCP or TLS traffic that isn’t HTTP-based.

- We will use Envoy Gateway as a Gateway Controller and implement a TLSRoute

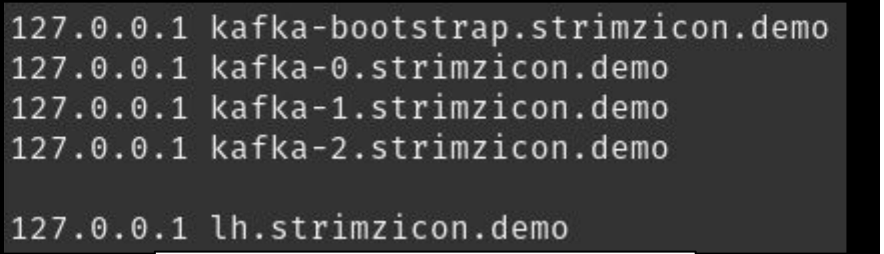
- Traffic will be encrypted end-to-end, terminated at the Kafka Brokers.



First Magic: /etc/hosts

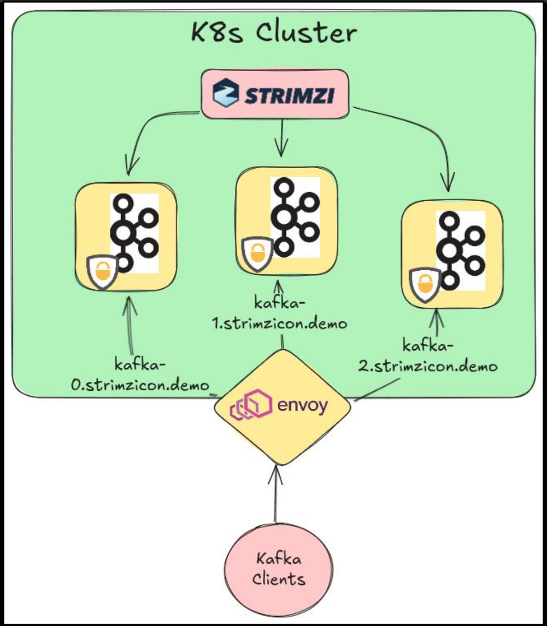
- Kafka brokers need to advertise unique addresses.
  - Just ‘localhost’ doesn’t work!

- /etc/hosts allows us to point certain hostnames to localhost.
  - Allows different brokers to have unique addresses that end up local.



```
127.0.0.1 kafka-bootstrap.strimzicon.demo
127.0.0.1 kafka-0.strimzicon.demo
127.0.0.1 kafka-1.strimzicon.demo
127.0.0.1 kafka-2.strimzicon.demo

127.0.0.1 lh.strimzicon.demo
```

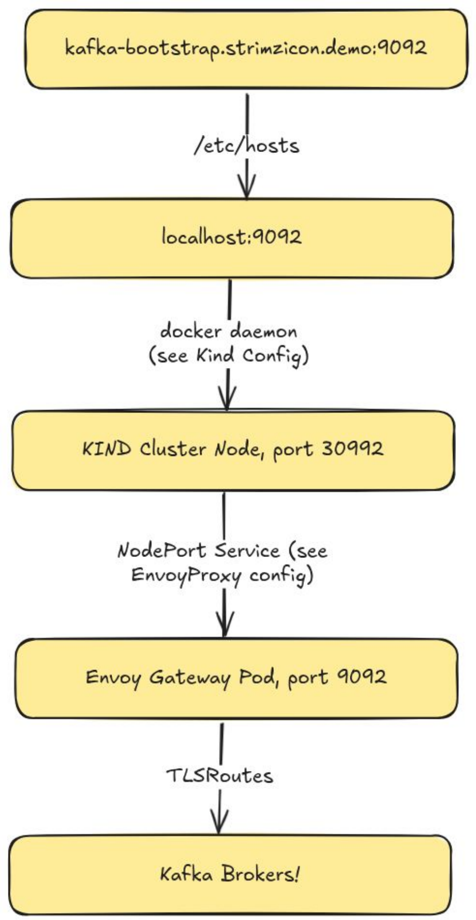


Second Trick: KIND Port Mappings

- We deployed an Envoy Gateway instance as a NodePort service.

- KIND clusters are just docker containers running locally.
    - A port on the docker container is a NodePort on the K8s node.

- Allows us to map localhost ports to ports on the Envoy Gateway!


kafka-bootstrap.strimzicon.demo:9092
/etc/hosts
localhost:9092
docker daemon (see Kind Config)
KIND Cluster Node, port 30992
NodePort Service (see EnvoyProxy config)
Envoy Gateway Pod, port 9092
TLSRoutes
Kafka Brokers!

Demo!

Thank You!

Get in touch!

My Blog: Accessing Kafka with Gateway API

LittleHorse QuickStart!

All the Code From Today

A Blog Series by Jakub Scholz: Accessing Kafka