StreamNative

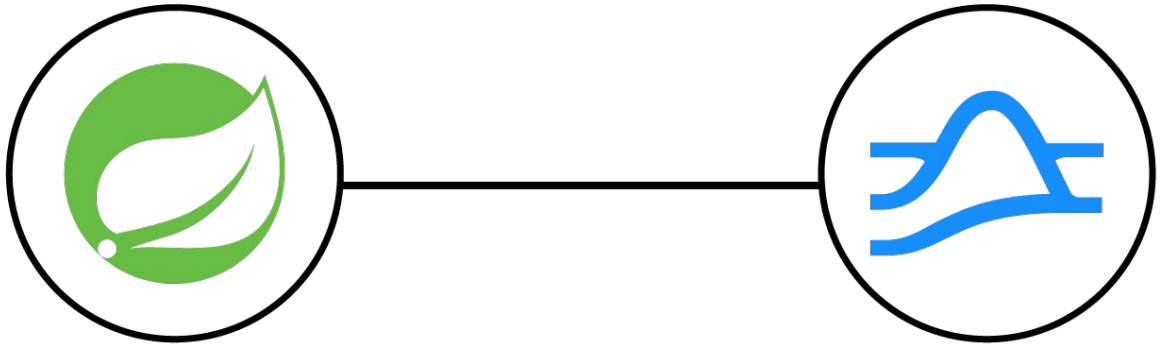
Let's keep it simple and streaming

Tim Spann
Developer Advocate

Tim Spann
Developer Advocate

- FLiP(N) Stack = Flink, Pulsar and NiFi Stack
- Streaming Systems/ Data Architect
- Experience:
  - 15+ years of experience with batch and streaming technologies including Pulsar, Flink, Spark, NiFi, Spring, Java, Big Data, Cloud, MXNet, Hadoop, Datalakes, IoT and more.

StreamNative

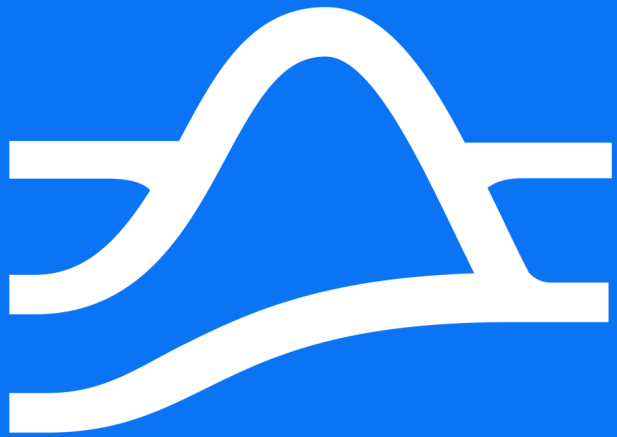


Announcing Spring for Apache Pulsar


APACHE PULSAR
EVERYWHERE
imgflip.com

Agenda

- Introduction
- What is Apache Pulsar?
- Spring Apps
- Pulsar
- AMQP
- MQTT
- Kafka
- Demo



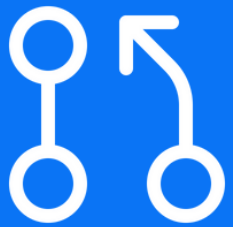
Apache Pulsar has a vibrant community

560+

Contributors

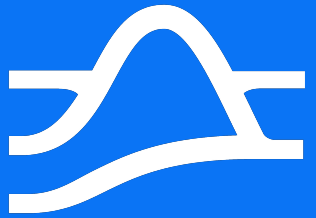
7,000+

Slack Members



10,000+

Commits

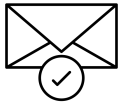


1,000+

Organizations Using Pulsar

PULSAR 101

Unified Messaging Platform

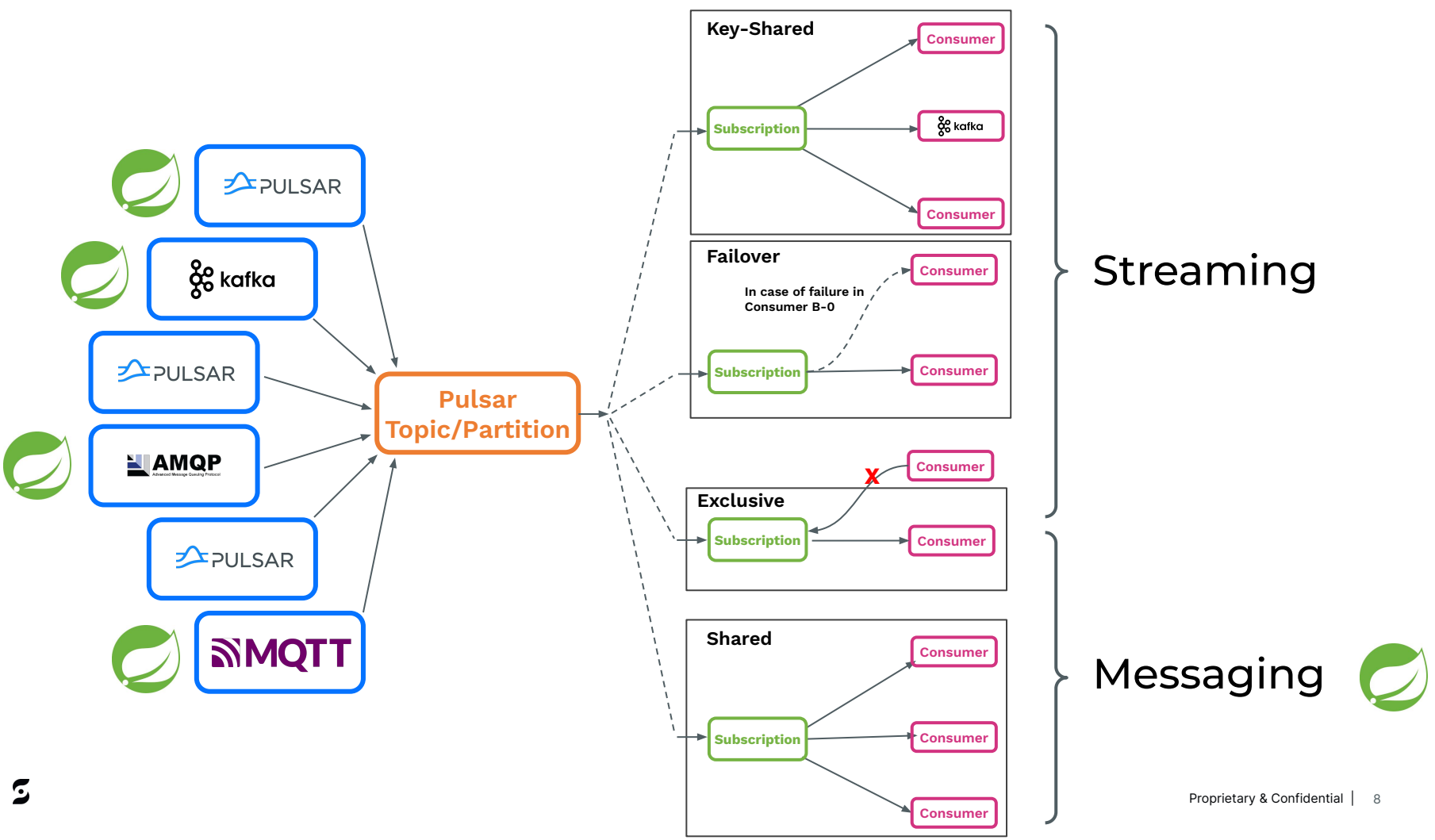


Guaranteed Message Delivery

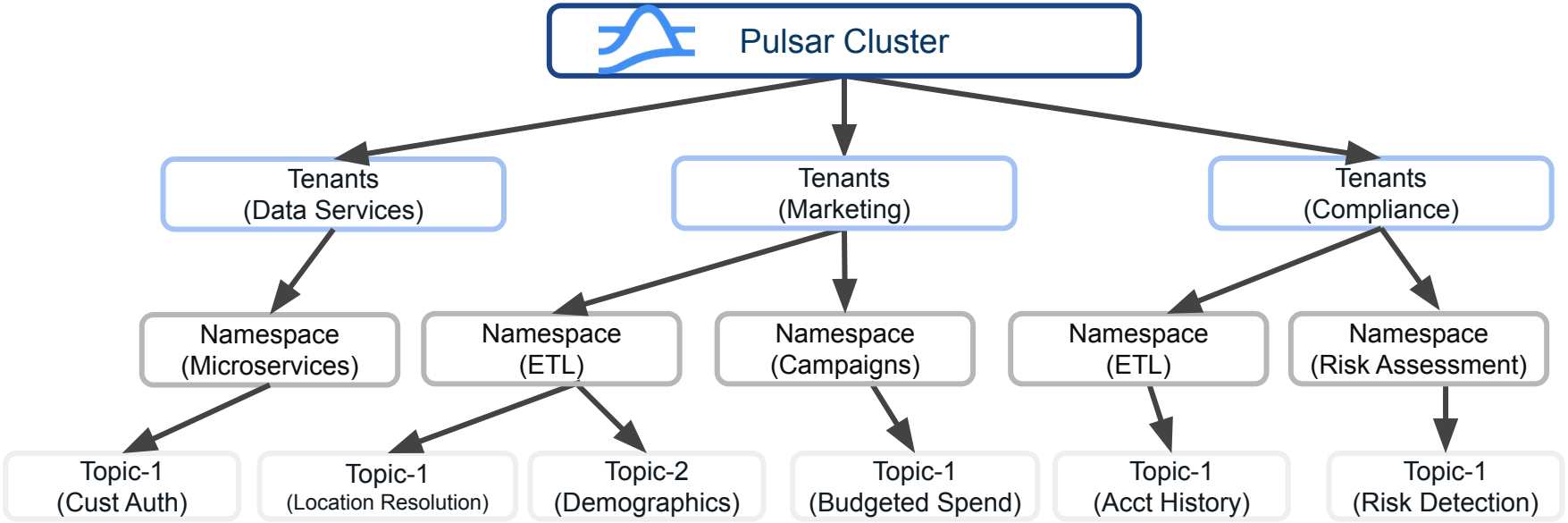
Resiliency

Infinite Scalability

StreamNative


PULSAR
kafka
PULSAR
AMQP
PULSAR
MQTT
Pulsar Topic/Partition
Key-Shared
Subscription
Consumer
kafka
Consumer
Failover
In case of failure in Consumer B-0
Subscription
Consumer
Consumer
Exclusive
Consumer
X
Subscription
Consumer
Shared
Subscription
Consumer
Consumer
Consumer
Streaming
Messaging

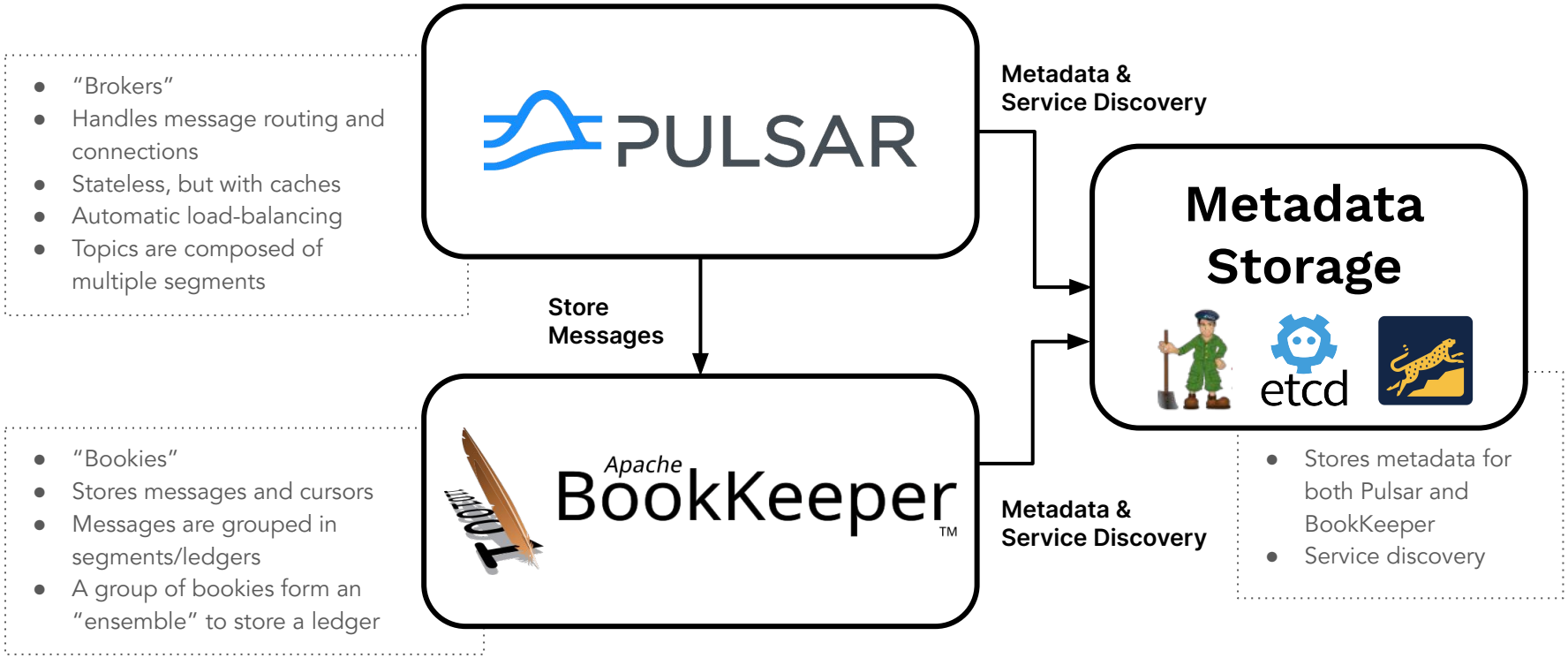
Tenants / Namespaces / Topics



Messages - the basic unit of Pulsar

Component	Description
Value / data payload	The data carried by the message. All Pulsar messages contain raw bytes, although message data can also conform to data schemas.
Key	Messages are optionally tagged with keys, used in partitioning and also is useful for things like topic compaction.
Properties	An optional key/value map of user-defined properties.
Producer name	The name of the producer who produces the message. If you do not specify a producer name, the default name is used.
Sequence ID	Each Pulsar message belongs to an ordered sequence on its topic. The sequence ID of the message is its order in that sequence.

Pulsar Cluster

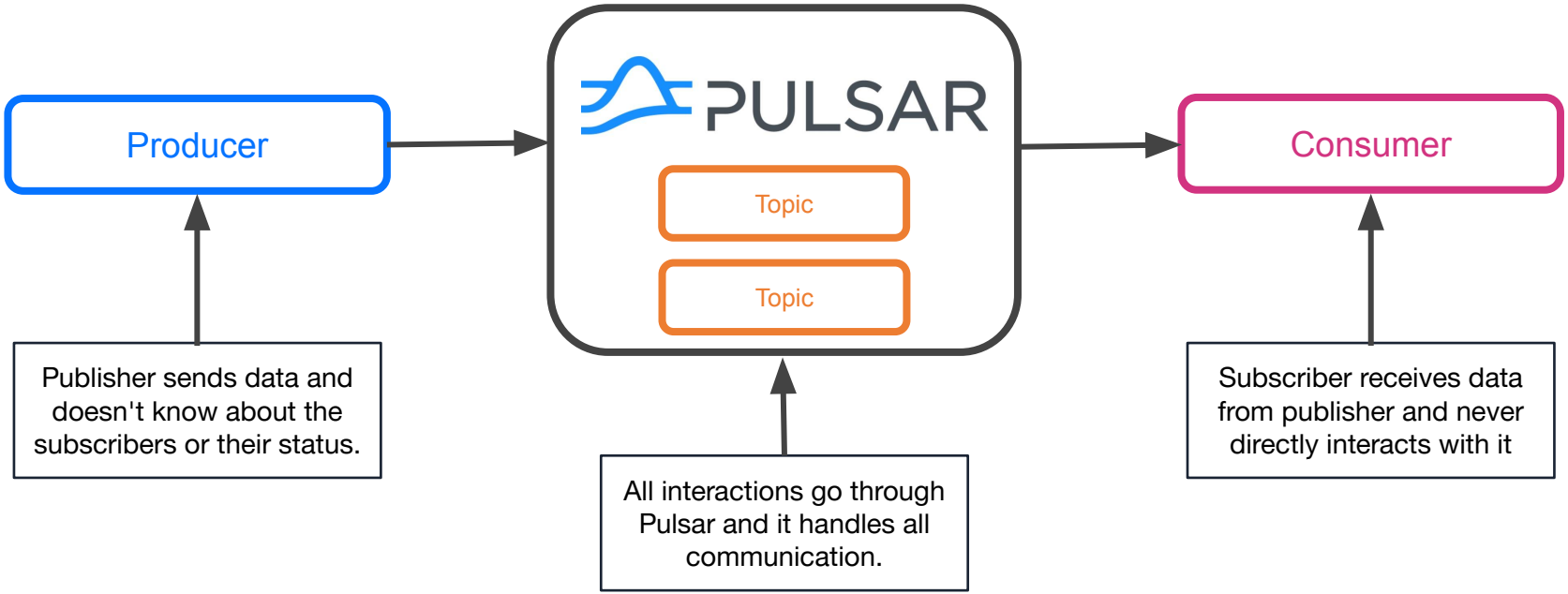


- "Brokers"
- Handles message routing and connections
- Stateless, but with caches
- Automatic load-balancing
- Topics are composed of multiple segments

- "Bookies"
- Stores messages and cursors
- Messages are grouped in segments/ledgers
- A group of bookies form an "ensemble" to store a ledger

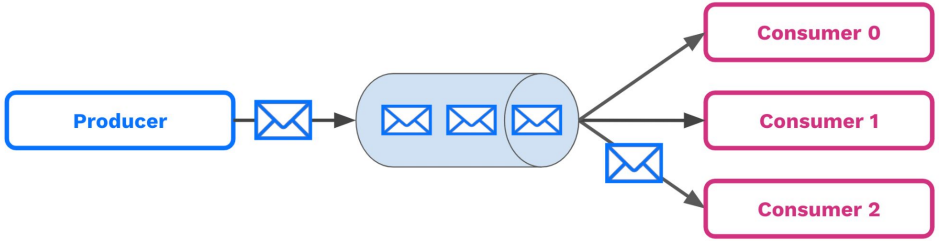
- Stores metadata for both Pulsar and BookKeeper
- Service discovery

Producer-Consumer

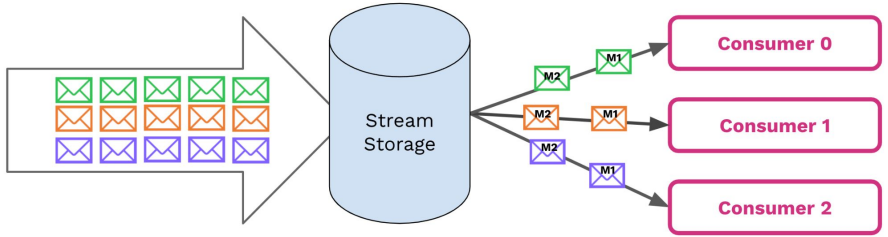


Apache Pulsar: Messaging vs Streaming

Message Queueing - Queueing systems are ideal for work queues that do not require tasks to be performed in a particular order.



Streaming - Streaming works best in situations where the order of messages is important.



StreamNative

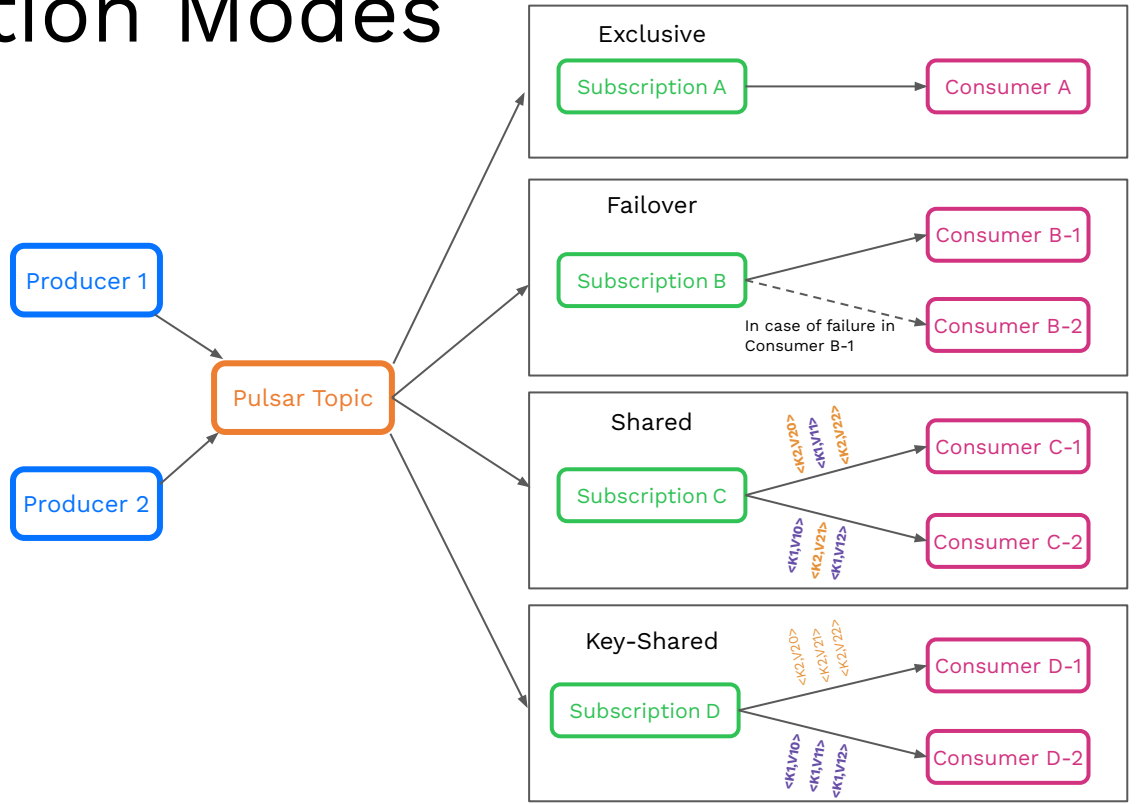
Pulsar Subscription Modes

Different subscription modes have different semantics:

Exclusive/Failover - guaranteed order, single active consumer

Shared - multiple active consumers, no order

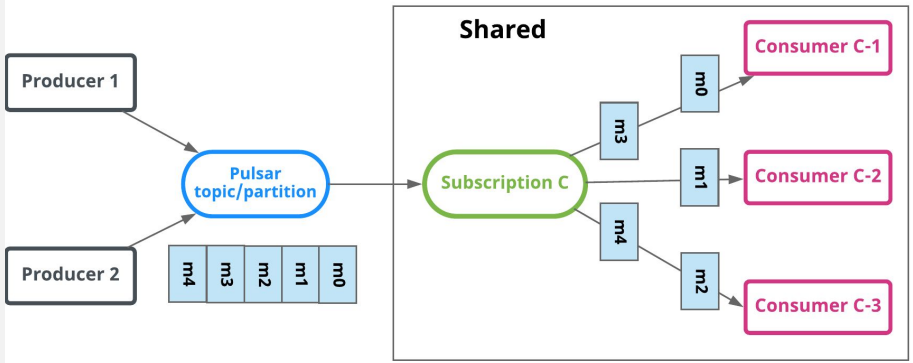
Key_Shared - multiple active consumers, order for given key



StreamNative

Flexible Pub/Sub API for Pulsar - Shared

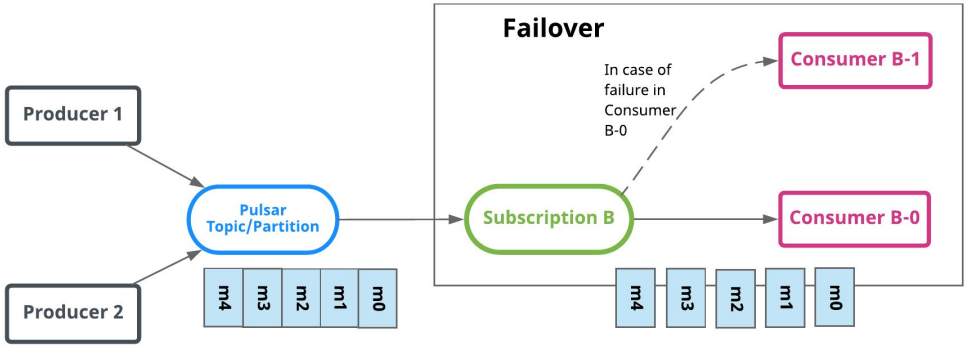
```
Consumer consumer = client.newConsumer()
        .topic("my-topic")
        .subscriptionName("work-q-1")
        .subscriptionType(SubType.Shared)
        .subscribe();
```



StreamNative

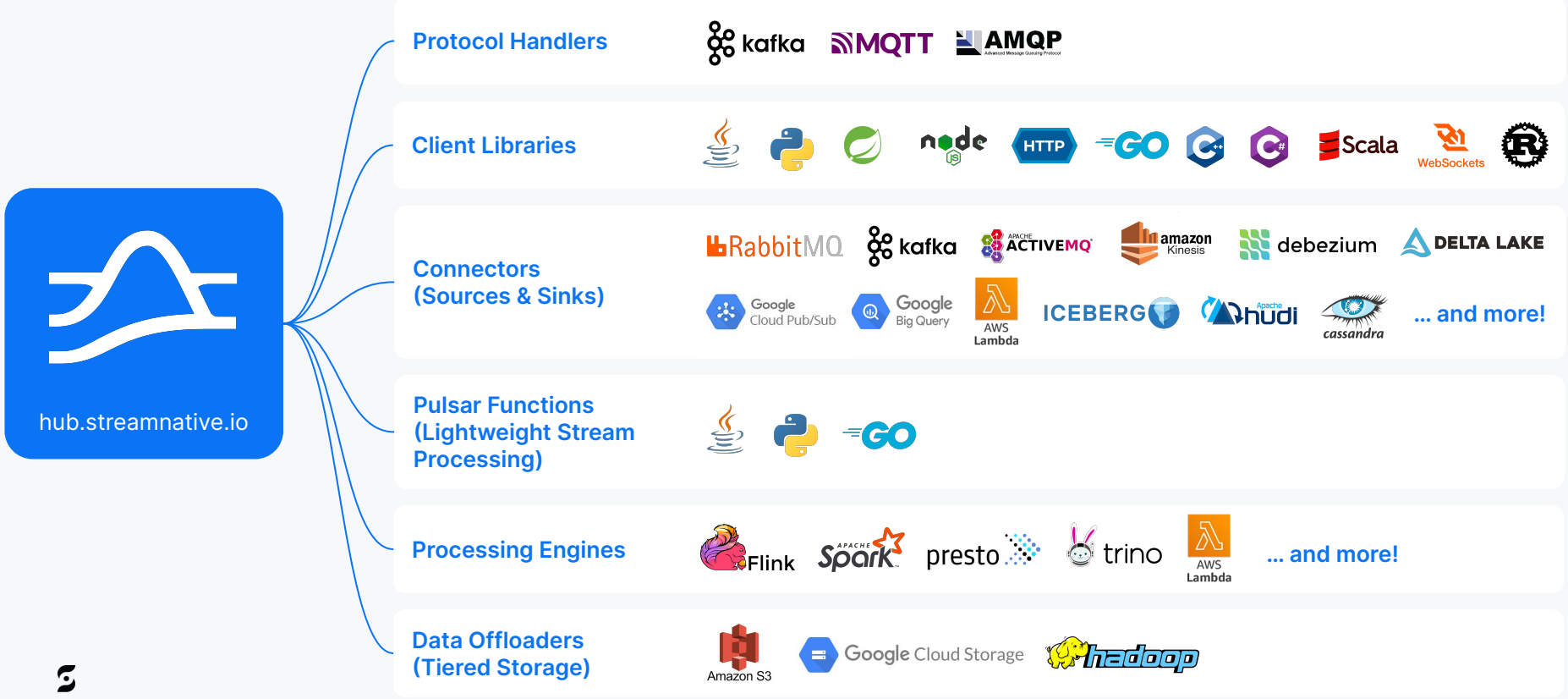
Flexible Pub/Sub API for Pulsar - Failover

```
Consumer consumer = client.newConsumer()
        .topic("my-topic")
        .subscriptionName("stream-1")
        .subscriptionType(SubType.Failover)
        .subscribe();
```

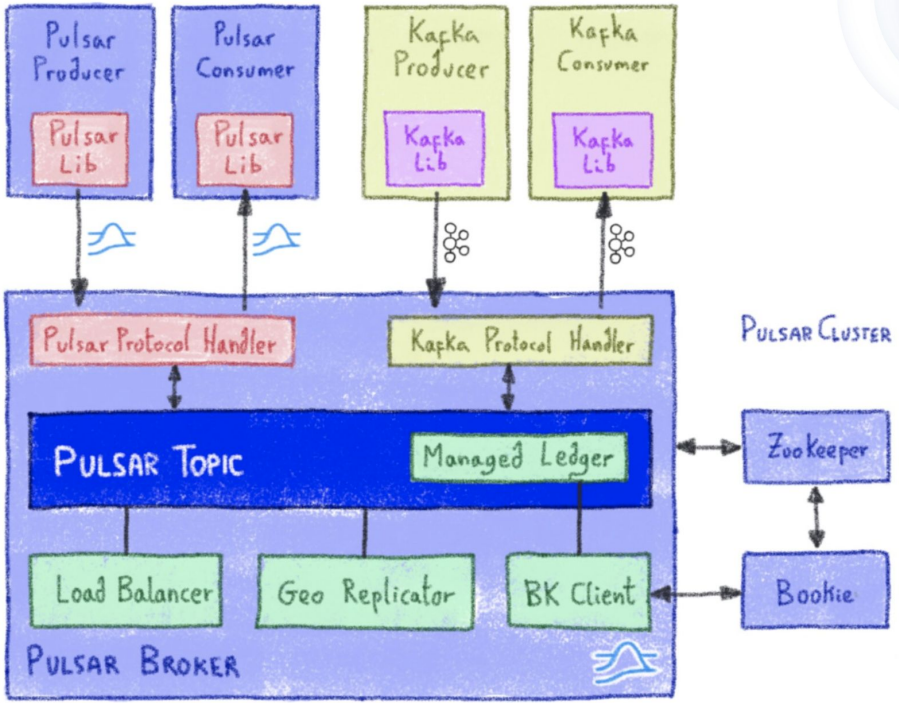


StreamNative

StreamNative Pulsar ecosystem

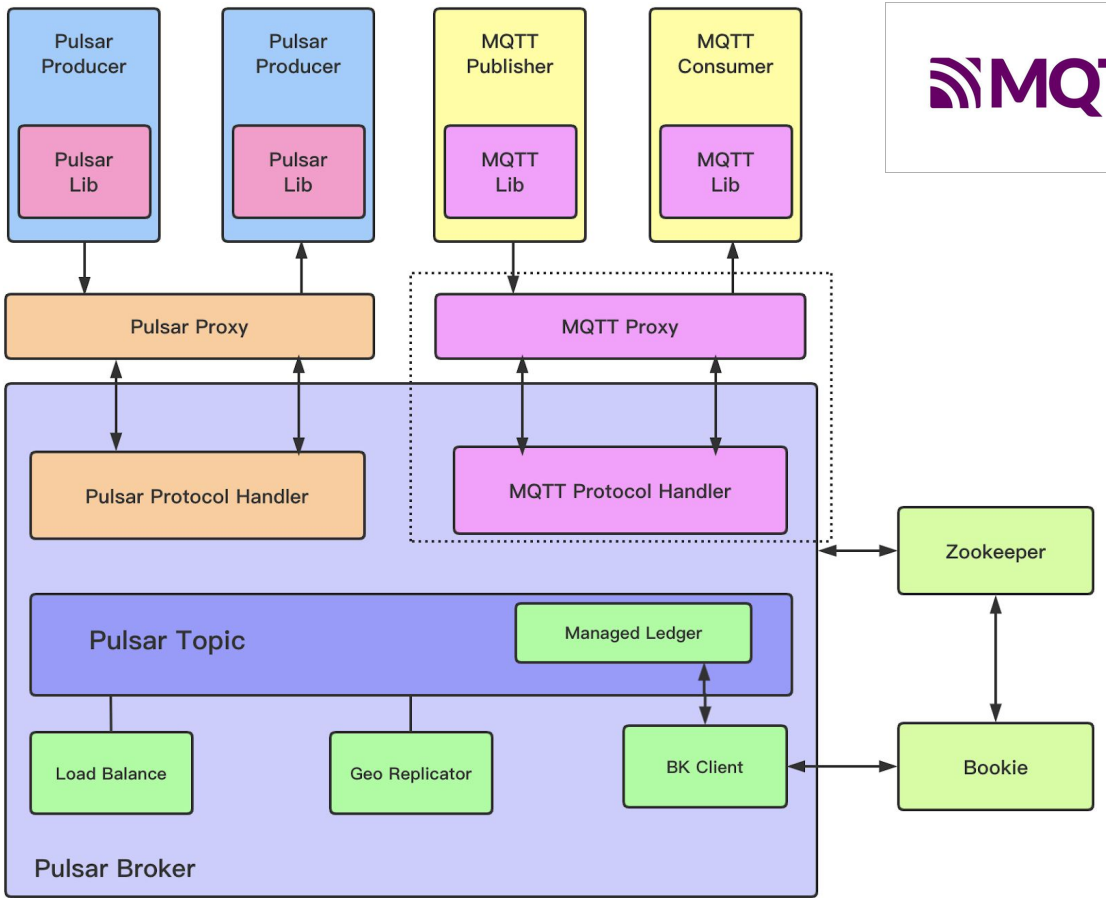


Kafka On Pulsar (KoP)



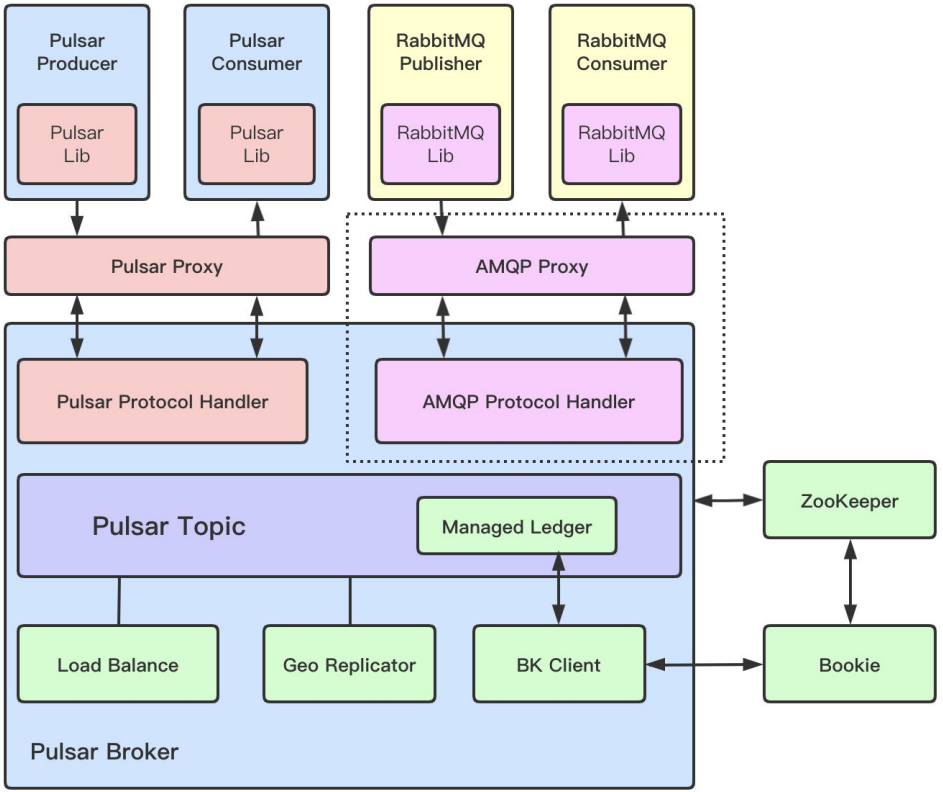
StreamNative

MQTT On Pulsar (MoP)



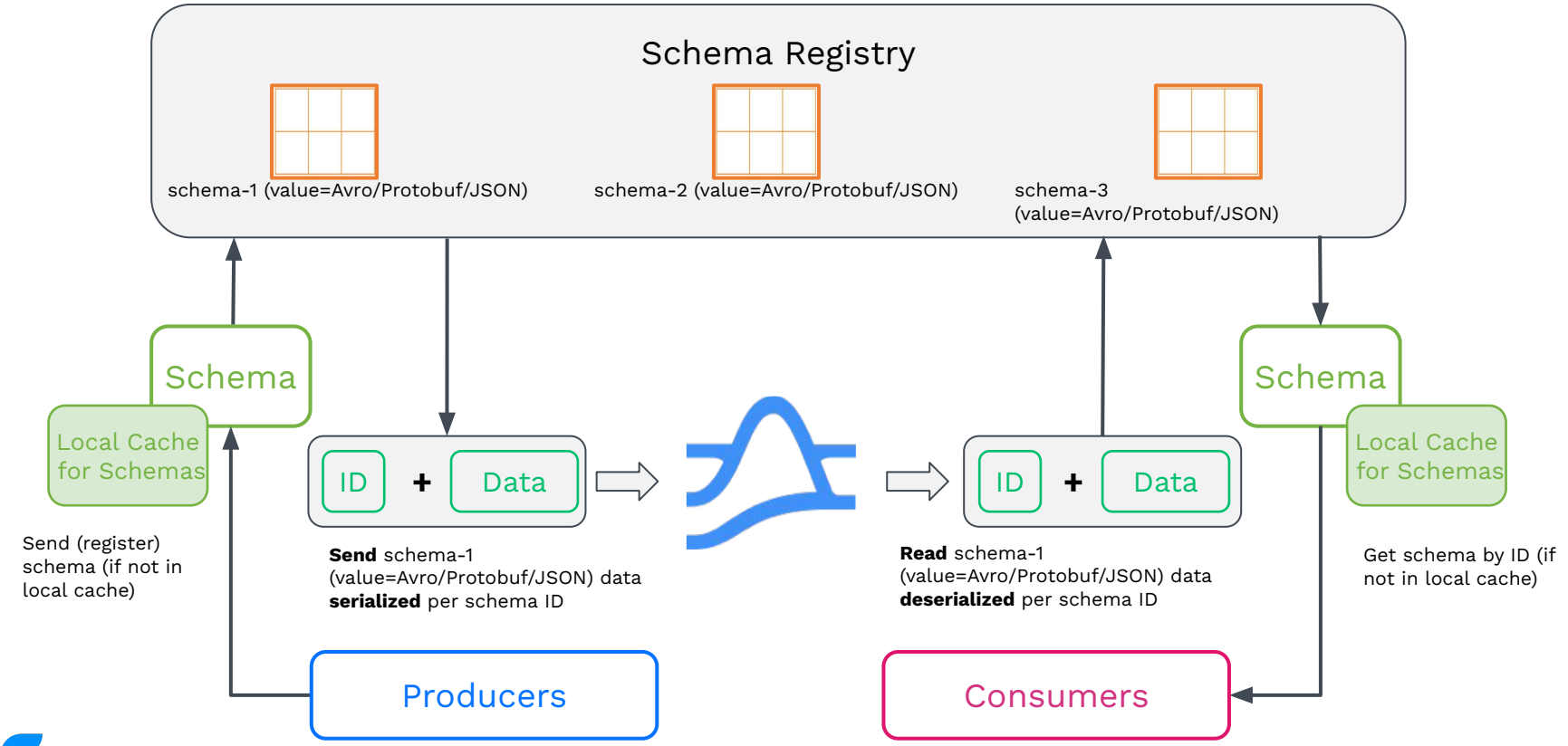
StreamNative

AMQP On Pulsar (AoP)



StreamNative

Schema Registry



StreamNative

Building Real-Time Requires a Team

StreamNative

Pulsar - Spring

https://docs.spring.io/spring-pulsar/docs/current/reference/html/

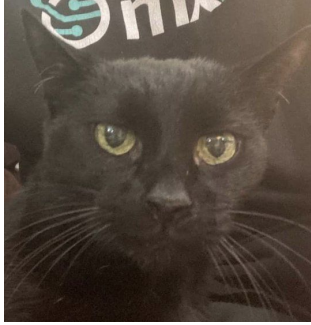
Pulsar - Spring - Code

```
@Autowired
private PulsarTemplate<Observation> pulsarTemplate;

this.pulsarTemplate.setSchema(Schema.JSON(Observation.class));

MessageId msgid = pulsarTemplate.newMessage(observation)
      .withMessageCustomizer((mb) -> mb.key(uidKey.toString()))
      .send();
```

```
@PulsarListener(subscriptionName = "aq-spring-reader", subscriptionType = Shared,
schemaType = SchemaType.JSON, topics = "persistent://public/default/aq-pm25")
void echoObservation(Observation message) {
  this.log.info("PM2.5 Message received: {}", message);
}
```

StreamNative

Pulsar - Spring - Configuration



```yaml
spring:
   pulsar:
     client:
       service-url: pulsar+ssl://sn-academy.sndevadvocate.snio.cloud:6651
       auth-plugin-class-name: org.apache.pulsar.client.impl.auth.oauth2.AuthenticationOAuth2
       authentication:
          issuer-url: https://auth.streamnative.cloud/
          private-key: file:///scr/sndevadvocate-tspann.json
          audience: urn:sn:pulsar:sndevadvocate:my-instance
     producer:
       batching-enabled: false
       send-timeout-ms: 90000
       producer-name: airqualityjava
       topic-name: persistent://public/default/airquality
```

StreamNative

Spring - Pulsar as Kafka

```
@Bean
public KafkaTemplate<String, Observation> kafkaTemplate() {
   KafkaTemplate<String, Observation> kafkaTemplate =
        new KafkaTemplate<String, Observation>(producerFactory());
   return kafkaTemplate;
}
```

```
ProducerRecord<String, Observation> producerRecord = new ProducerRecord<>(topicName,
       uuidKey.toString(),
       message);
```

```
kafkaTemplate.send(producerRecord);
```

StreamNative

https://www.baeldung.com/spring-kafka

Spring - MQTT - Pulsar

```
@Bean
public IMqttClient mqttClient(
@Value("${mqtt.clientId}") String clientId,
@Value("${mqtt.hostname}") String hostname,
@Value("${mqtt.port}") int port)
throws MqttException {
    IMqttClient mqttClient = new MqttClient(
        "tcp://" + hostname + ":" + port, clientId);
    mqttClient.connect(mqttConnectOptions());
    return mqttClient;
}
MqttMessage mqttMessage = new MqttMessage();
mqttMessage.setPayload(DataUtility.serialize(payload));
mqttMessage.setQos(0);
mqttMessage.setRetained(true);
mqttClient.publish(topicName, mqttMessage);
```

StreamNative

https://roytuts.com/publish-subscribe-message-onto-mqtt-using-spring/

Spring - AMQP - Pulsar

```
@Bean
public CachingConnectionFactory
        connectionFactory() {
    CachingConnectionFactory ccf =
new CachingConnectionFactory();
    ccf.setAddresses(serverName);
    return ccf;
}
```

```
rabbitTemplate.convertAndSend(topicName,
        DataUtility.serializeToJSON(observation));
```

StreamNative

https://www.baeldung.com/spring-amqp

Reactive Spring - Pulsar

```
2 usages
@Autowired
ReactivePulsarTemplate<String> reactivePulsarTemplate;

Timothy Spann *
@Scheduled(initialDelay = 1000, fixedRate = 1000)
public void getRows() {
    reactivePulsarTemplate.setSchema(Schema.STRING);

    for ( int rowCounter = 0; rowCounter < 20; rowCounter++ ) {
        System.out.println("sending " + rowCounter);
        reactivePulsarTemplate.newMessage(messageBuilder())
                .withMessageCustomizer((mb) -> mb.key(UUID.randomUUID().toString()))
                .withSenderCustomizer((sc) -> sc.accessMode(ProducerAccessMode.Shared))
                .withSenderCustomizer((sc2) -> sc2.producerName("ReactiveProducer"))
                .withSenderCustomizer((sc3) -> sc3.sendTimeout(Duration.ofSeconds(60L)))
                .withSenderCustomizer((sc4) -> sc4.maxInflight( i: 100))
                .send().subscribe();
    }
}
```

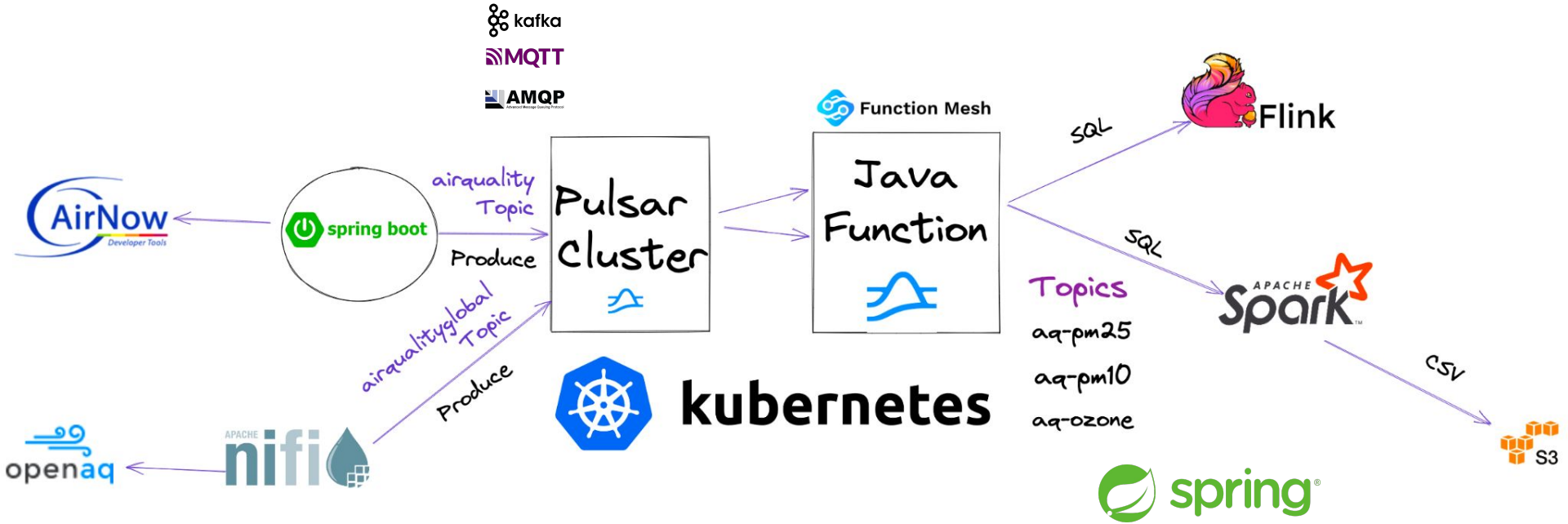
StreamNative

Reactive Spring - Pulsar

```
@ReactivePulsarListener(topics = "persistent://public/default/reactivefaker", stream = true)
Flux<MessageResult<Void>> listen(Flux<Message<String>> messages) {
    return messages
            .doOnNext((msg) -> System.out.println("Stream Received: " + msg.getValue()))
            .map(MessageResult::acknowledge);
}
```

StreamNative

REST + Spring Boot + Pulsar + Friends



StreamNative

FLiP Stack Weekly

https://bit.ly/32dAJft

This week in Apache Flink, Apache Pulsar, Apache NiFi, Apache Spark, Java and Open Source friends.

StreamNative

Demo

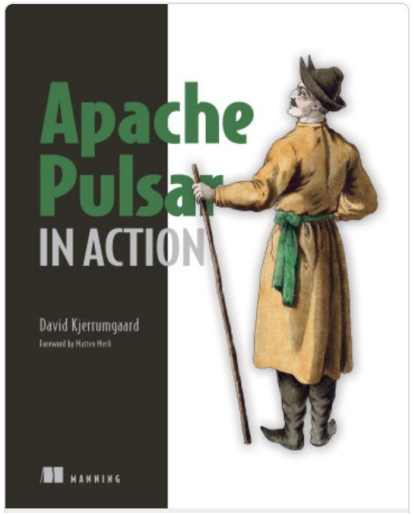
StreamNative

Spring + Pulsar References

- https://streamnative.io/blog/engineering/2022-11-29-spring-into-pulsar-part-2-spring-based-microservices-for-multiple-protocols-with-apache-pulsar/
- https://streamnative.io/blog/release/2022-09-21-announcing-spring-for-apache-pulsar/
- https://docs.spring.io/spring-pulsar/docs/current-SNAPSHOT/reference/html/
- https://spring.io/blog/2022/08/16/introducing-experimental-spring-support-for-apache-pulsar
- https://medium.com/@tspann/using-the-new-spring-boot-apache-pulsar-integration-8a38447dce7b

Spring Things

- https://spring.io/guides/gs/spring-boot/
- https://spring.io/projects/spring-amqp/
- https://spring.io/projects/spring-kafka/
- https://github.com/spring-projects/spring-integration-kafka
- https://github.com/spring-projects/spring-integration
- https://github.com/spring-projects/spring-data-relational
- https://github.com/spring-projects/spring-kafka
- https://github.com/spring-projects/spring-amqp



Apache Pulsar in Action

Please enjoy David's complete book which is the ultimate guide to Pulsar.

Tim Spann

Developer Advocate
at StreamNative

https://www.linkedin.com/in/timothyspann

@PaaSDev

https://github.com/tspannhw

StreamNative

Notices

Apache Pulsar™