ORACLE

OCI Streaming Service (OSS)

OCI 上完全托管的分布式流数据(消息队列)服务

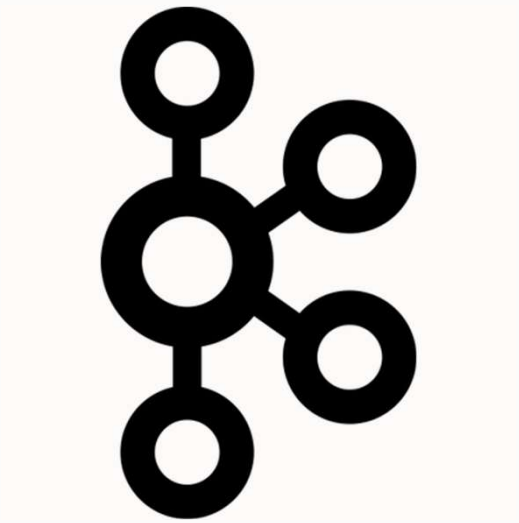
杨鲁
yang.x.lu@oracle.com

Solution Engineer, Technology Hub, JAPAC

议程

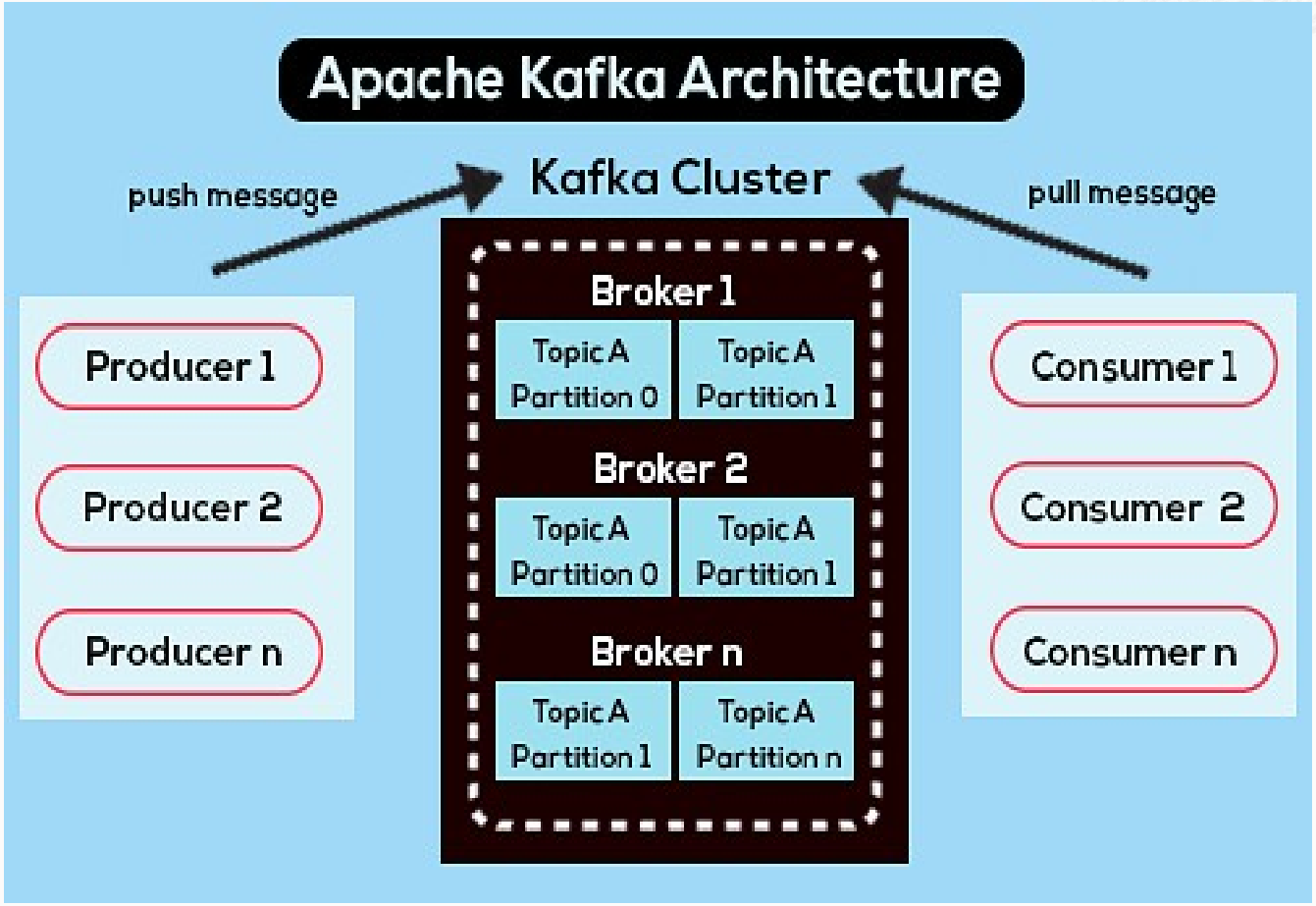
- **从Kafka谈起**
- OSS介绍
  - 云原生，基于开源软件，安全，与OCI及第3方产品深度集成，按使用量计费
- OSS 核心概念
- OSS Kafka 兼容API示例
- OSS 适用场景
- OSS 客户案例

Apache Kafka – 它是什么?



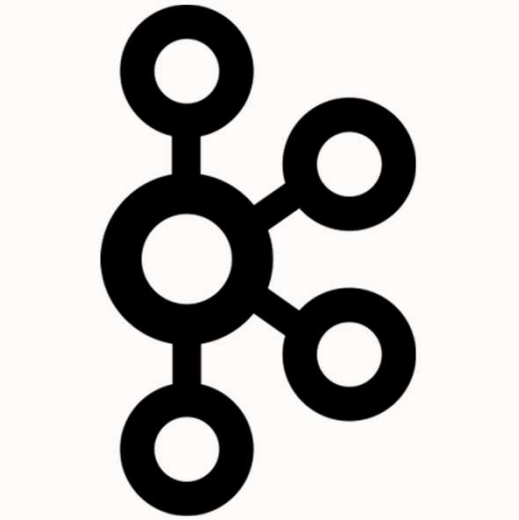
- Apache Kafka 是事实上的标准分布式流数据处理平台，被广泛部署为消息传递系统
- 自 2012 年以来的 Apache 开源项目
- Kafka 旨在解决两个问题
  - 简化数据管道
  - 处理流数据

Apache Kafka – 架构图



- **Broker**
消息中间件处理节点（服务器），一个节点就是一个broker，一个Kafka集群由一个或多个broker组成
- **Topic**
Kafka对消息进行归类，发送到集群的每一条消息都要指定一个topic
- **Partition**
物理上的概念，每个topic包含一个或多个partition，一个partition对应一个文件夹，这个文件夹下存储partition的数据和索引文件，每个partition内部是有序的
- **Producer**
生产者，负责发布消息到broker
- **Consumer**
消费者，从broker读取消息
- **ConsumerGroup**
每个consumer属于一个特定的consumer group，可为每个consumer指定group name。若不指定，则属于默认的group。一条消息可以发送到不同的consumer group，但一个consumer group中只能有一个consumer能消费这条消息

Apache Kafka – 是什么使它如何强大?



- 多用户生产消费消息
- 容错性
- 水平扩展
- 高性能
- 丰富的生态

在云上运行 Apache Kafka 的开销

部署开销

- 配置虚拟网络
- 选择计算实例型号
- 安装/配置Zookeeper
- 安装/配置Kafka

管理开销

- 使partitions/topics
  在不同结点间进行负载均衡
- OS补丁升级
- Zookeeper补丁
- 当业务增长时动态
  增加更多的计算实例
- 提供高可用性
- 数据加密
- 认证，授仅

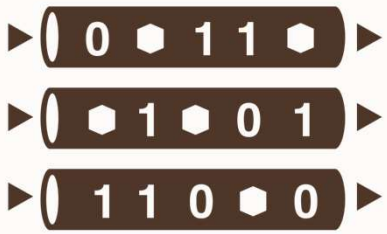
议程

- 从Kafka谈起
- **OSS介绍**
  - **云原生，基于开源软件，安全，与OCI及第3方产品深度集成，按使用量计费**
- OSS 核心概念
- OSS Kafka 兼容API示例
- OSS 适用场景
- OSS 客户案例

OCI Streaming Service介绍

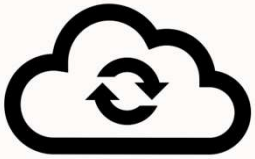
一个真正的无服务器、企业级、基于 Apache Kafka 的分布式消息队列服务平台。

OCI Streaming 特点



Streaming

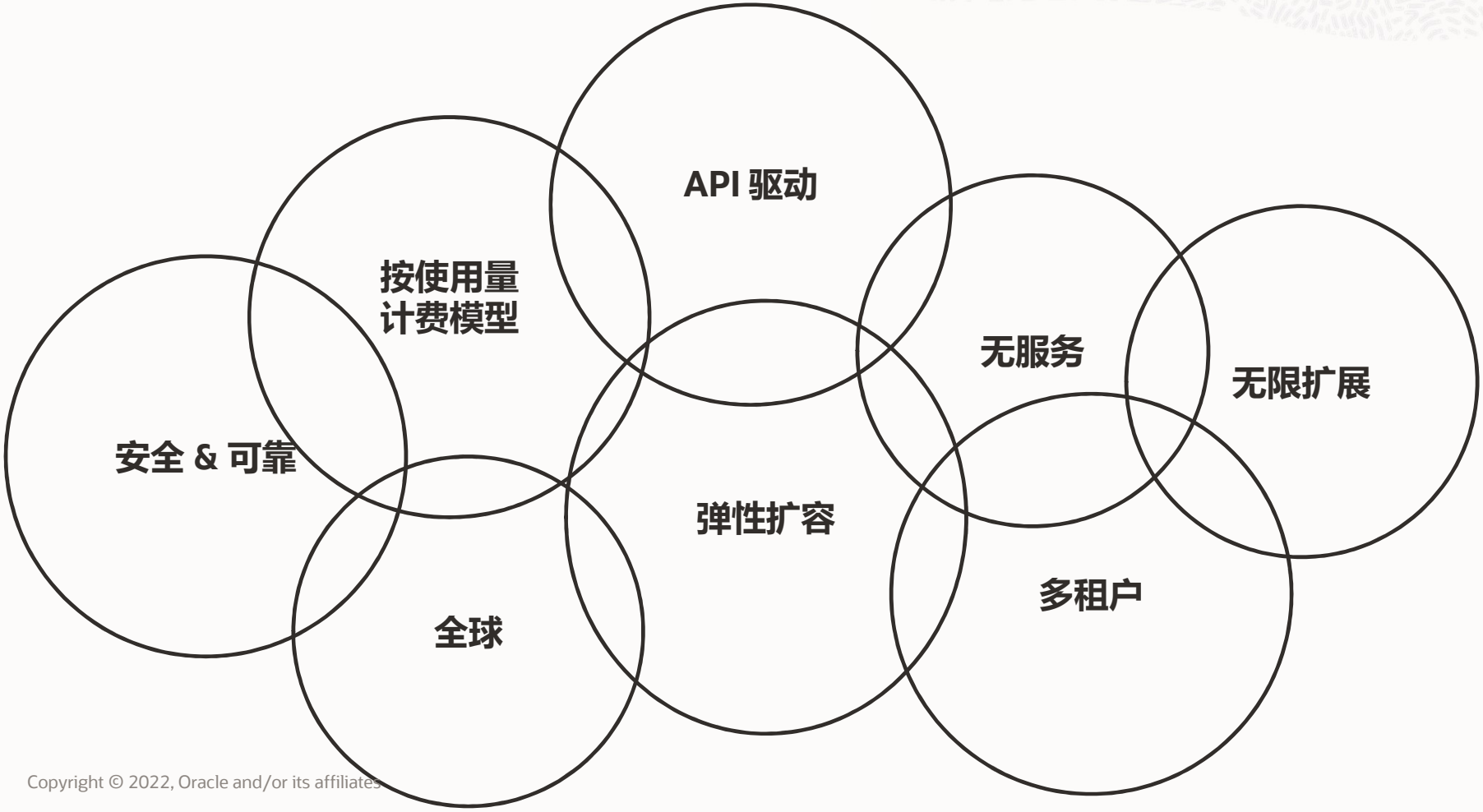
- 云原生
- 基于开源软件及云中立
- 安全
- 与OCI及第3方产品深度集成
- 按使用量计费



云原生

构建具有无限可扩展性、完全弹性和完全无服务器的真正云原生产品

OCI Streaming – 真正的云原生数据系统



OCI Streaming – Cloud Native

无服务

自动化所有基础设施
和平台维护

- 无服务器，无需在线维护和修补软件
- 操作简便 – 与 OCI 监控集成

按使用量付费

最佳性价比
保证

- 不为预置容量付费，只为使用付费
- 简单和最小的定价维度，
- 数据传输无需额外费用

无限扩展

在线扩展以获得
最佳性能及最低成本

- 根据需要扩展和扩展
- 容错 - 跨 AD 复制数据
- 99.9% 的可用性 SLA 保证

构建于Oracle Cloud Infrastructure之上

物理服务器，网络虚拟化，高性能，高安全，高可用性

完全专用*

业界第一台完全专用的服务器——没有管理程序、代理、嘈杂的邻居或共享资源

企业级

耐用性和可用性保证；专为支持企业工作负载而设计

性能优先*

性能优先的方法，其性能明显高于现有云选项

自动化和 API 驱动

创建 OCI 服务，就像使用 REST API 的公有云 VM 一样

快速创建*

在不到 5 分钟的时间内启动 OCI 服务 – 无需再等待几天的时间来配置服务器

按使用量付费模式*

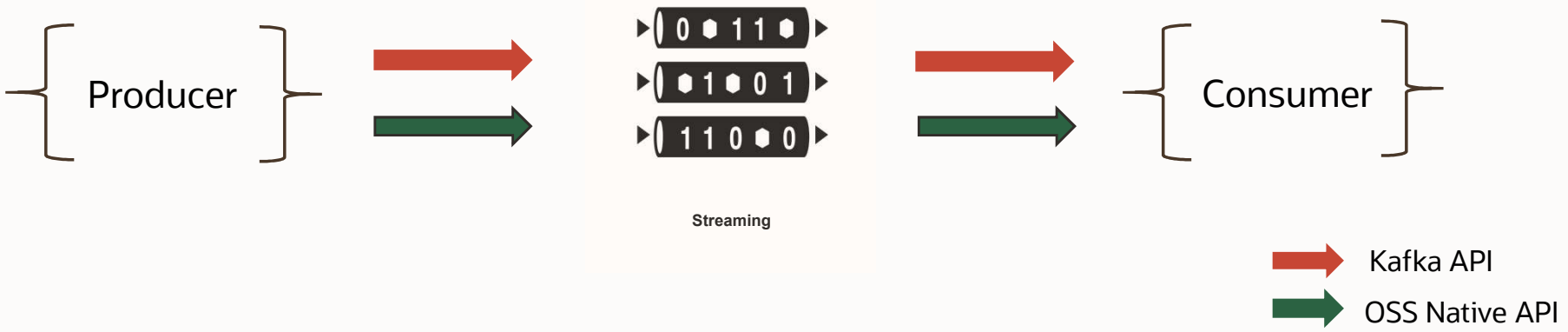
一切都按小时付费：计算、IP 地址和块存储——突然增加或减少

*** 针对大量数据的工作负载进行的优化**

开源软件

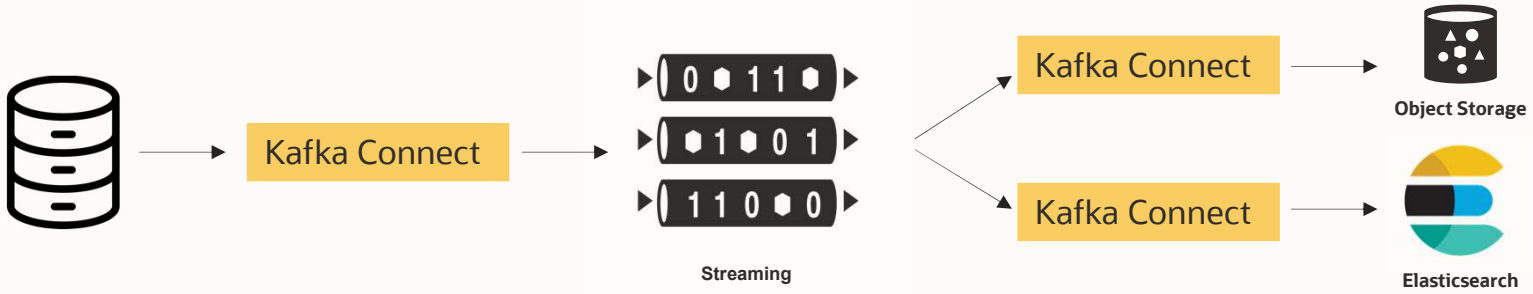
与Apache Kafka兼容，没有任何锁定

OCI Streaming - Kafka 兼容



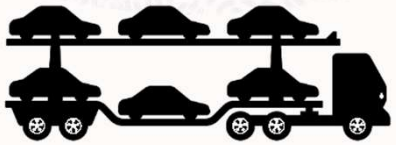
- 没有锁定和访问丰富的开发者生态系统
- 具有全面管理和云定价的开源软件的灵活性
- 混合 API 使用——同时使用 OCI 原生 API 和 Kafka API

OCI Streaming - Kafka Connect



- 丰富的生态系统 - 能与100多个不同的产品结合(作为源，或目标端)
- 易于使用 - 能够轻松地插入数据

OCI Streaming – 很容易从现有的Kafka集群迁移过来



- 从现有的 Kafka 实现迁移简单快速
- 拥有现有 Kafka 应用程序的客户只需更改其配置的以下参数即可迁移到 OSS*

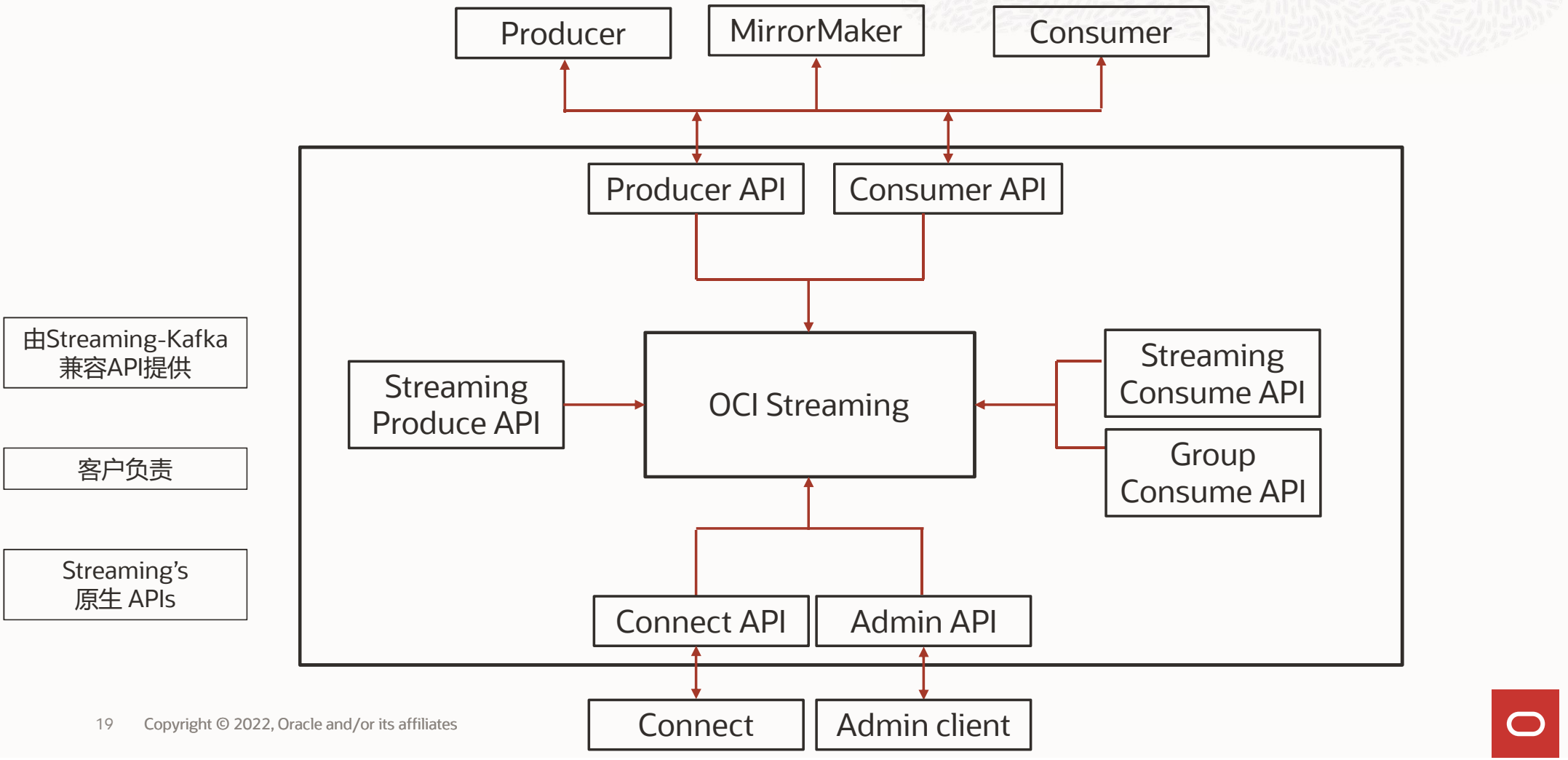
```
# Kafka settings
security.protocol: SASL_SSL

sasl.mechanism: PLAIN

sasl.jaas.config: org.apache.kafka.common.security.plain.PlainLoginModule required
username="{username}" password="{pwd}";

bootstrap.servers: kafka.streaming.{region}.com:9092
# Application settings
topicName: [streamOcid]
```

后面章节API示例有更详细的配置示例

OCI Streaming - Kafka API 支持

Feature	Available since	Use case	API support
Produce	Jan 2012 – v 0.7.0	产生消息到topic	✅ 支持
Consume	Jan 2012 – v 0.7.0	从topic消费消息	✅ 支持
Group Management	Jan 2012 – v 0.7.0	利用消费组从topic消费消息	✅ 支持
Kafka Connect	Nov 2015 – v 0.9.0.0	Kafka Connectors (Source 和 Sink)	✅ 支持
Admin APIs	Oct 2016 – 0.10.1.0	程序创建/删除消息	✅ 支持
Add partitions	Jan 2012 – v 0.7.0	增加分区数	✅ 支持
Compaction	March 2014 – v 0.8.1	仅支持Kafka Connect 用例	⚠️ 部分支持
Kafka Streams	May 2016 – v 0.10.0.0	Kafka Streams 以进行stream处理	❌ 不支持
Transaction	Jun 2017 – v 0.11.0.0	跨多个topic/partition原子写入	❌ 不支持
Idempotent produce	Jun 2017 – v 0.11.0.0	Exactly-once(避免重复）产生消息机制	❌ 不支持

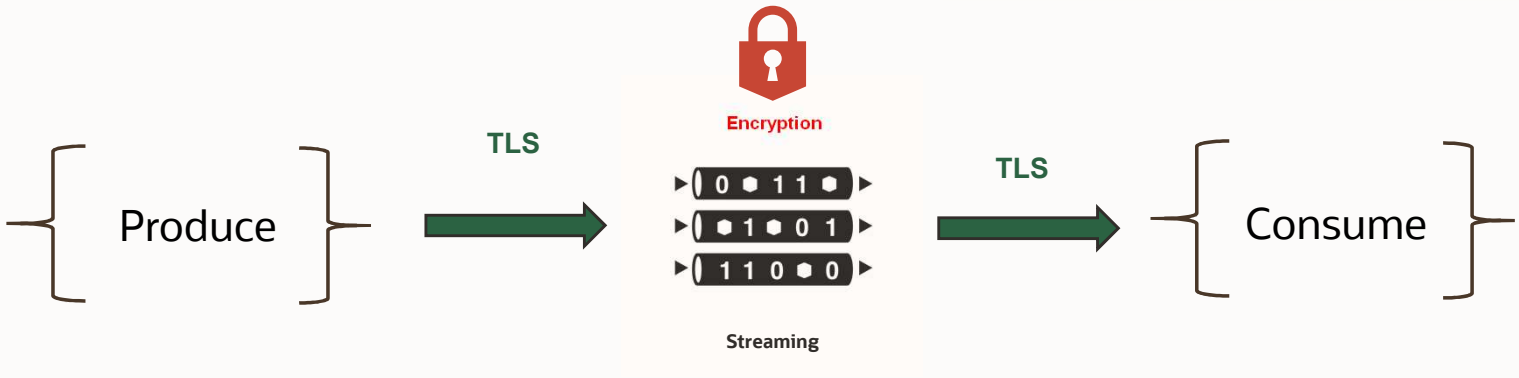
OCI Streaming - Kafka 兼容



安全

部署到 Oracle Oracle Cloud Infrastructure，
具有企业级安全性、性能和容错性

Security – 提供了端到端的加密

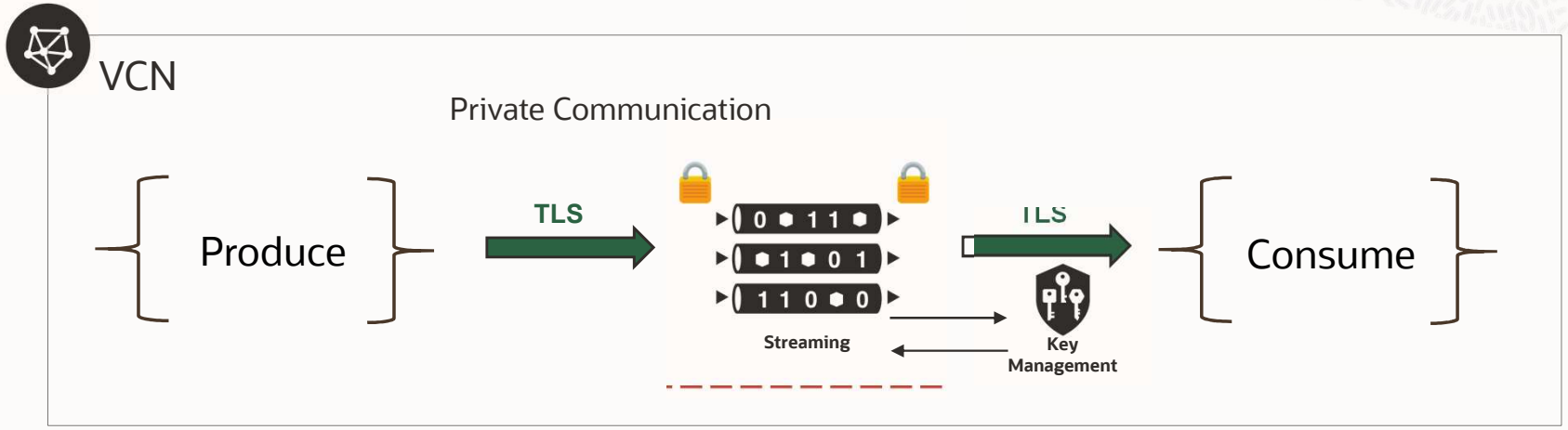


- 流数据总是加密的 – 对用户透明，后台完成
- Oracle 自动为整个堆栈应用安全更新 - 每月或针对高影响安全漏洞的非周期更新

Security – 与OCI IAM完全集成

- OSS与 OCI IAM 完全集成
- 客户通过 OCI IAM 策略获得细粒度的访问控制——从流创建到生产和消费流数据。
- Kafka 用户可以使用 Streaming 的 Kafka 端点并使用 PLAIN SASL 进行身份验证

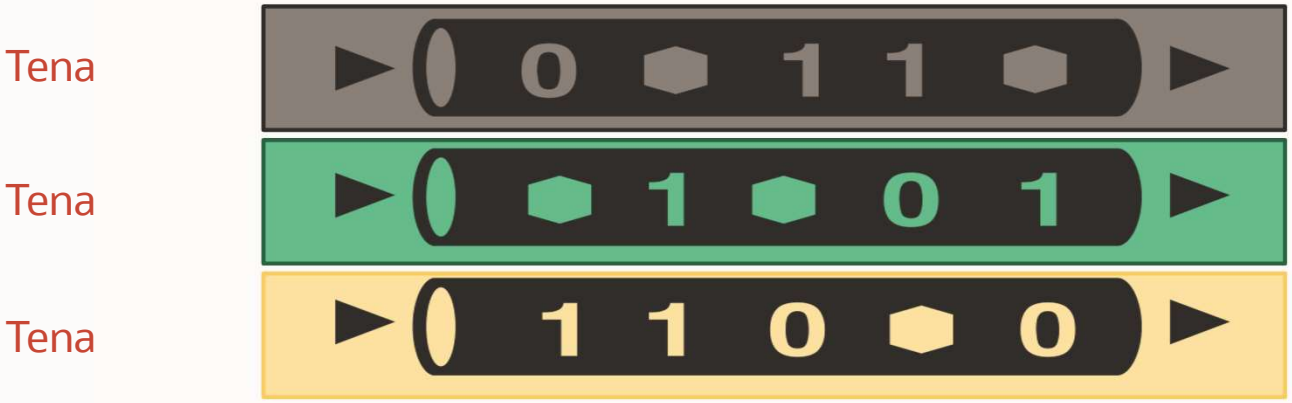
Security – 私有Endpoints， 以及KMS (Vault) 支持



- 当用户为stream pool启用私有端点时，流式传输流量会被路由到他们的stream实例，而无需遍历互联网
  - 这提供了完全阻止从公共端点访问Stream资源

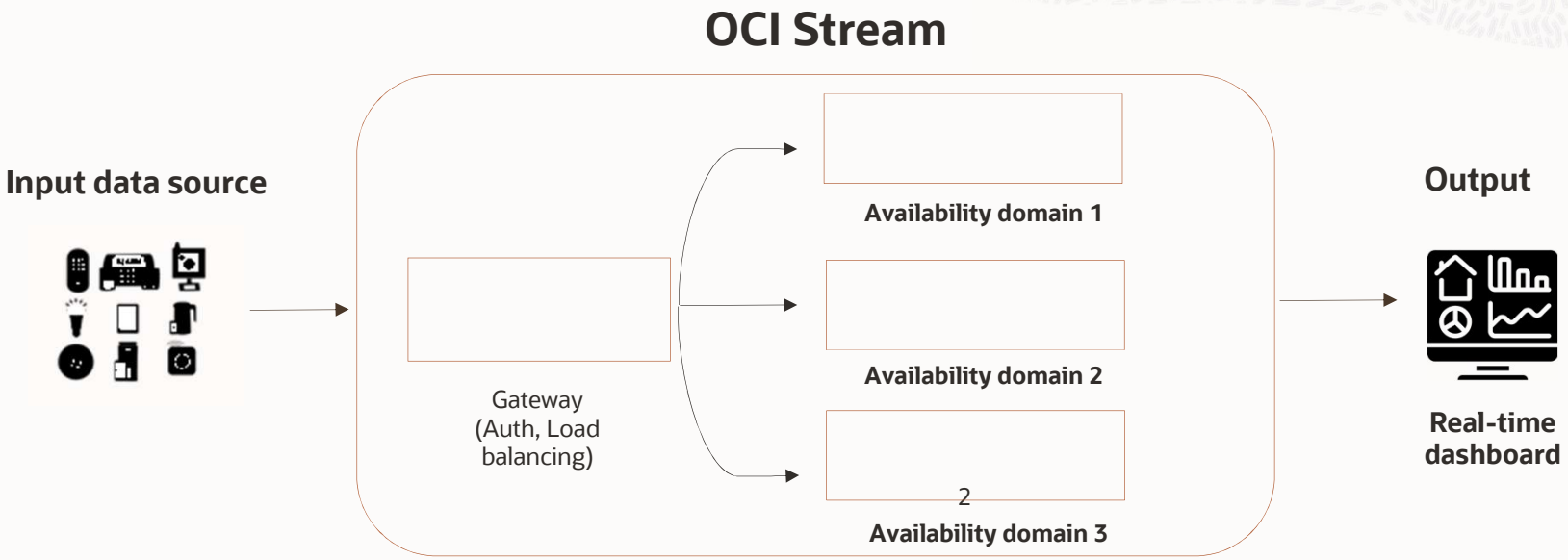
- 客户可以使用 Oracle 提供的密钥（通过 KMS）或使用他们自己的密钥

Security – 完全隔离



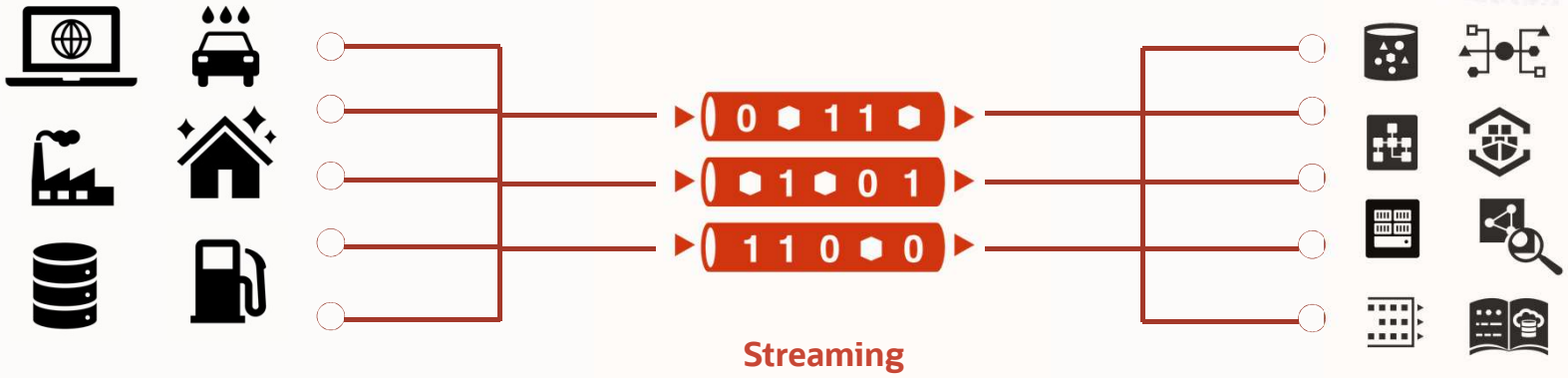
- 流服务内部架构保证数据的完全租户级隔离。
- 无论使用情况如何，该服务都保证没有嘈杂的邻居问题

Security – 弹性处理异常



- 流服务在地理分布的 AD 之间同步复制数据
  - 这提供了容错和持久性保证

- 流媒体服务提供 99.95% 的可用性 SLA 保证

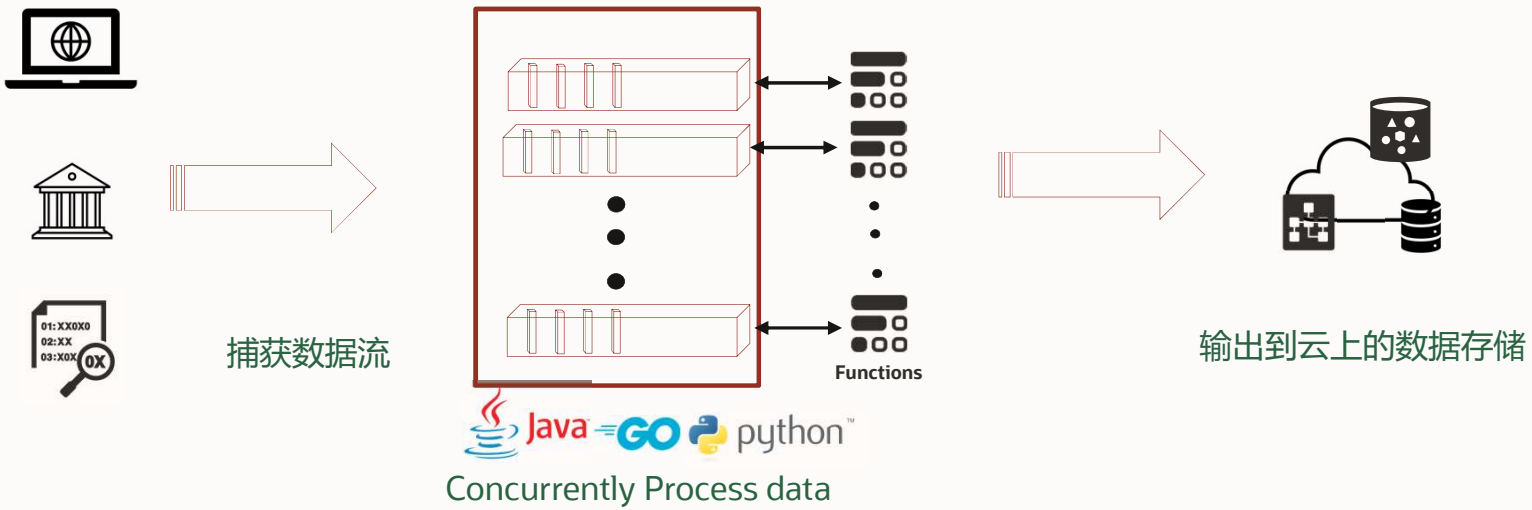


深度集成

与 100 种第一方和第三方产品的开箱即用集成

OCI Streaming– 与Functions集成

构建无服务器实时数据处理管道 OCI Streaming 和 Oracle Functions



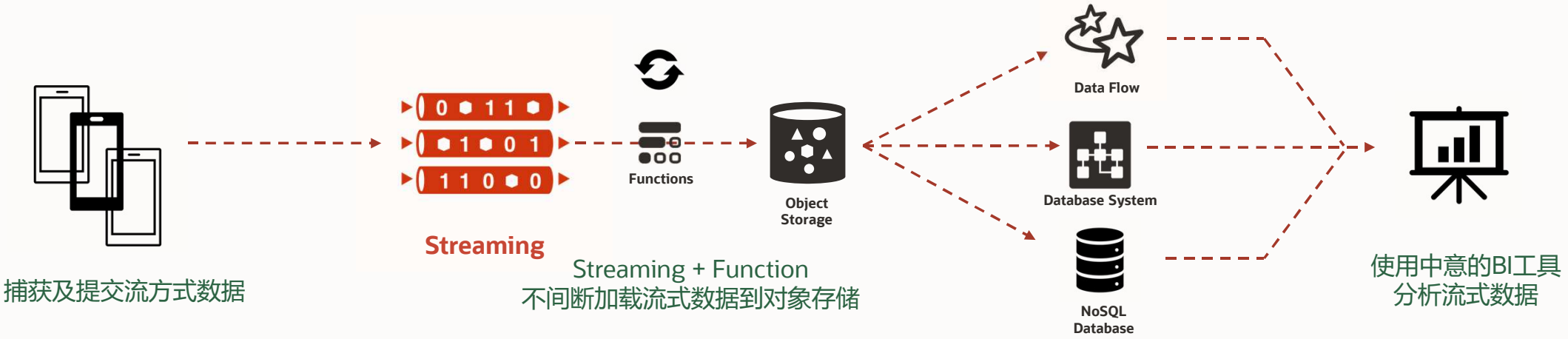
易用性 专注于快速启动数据流应用程序，而不是管理基础设施

并发处理 并行处理流中所有分区的数据

实时 采集实时数据流，实时及时响应关键业务事件和运营触发

OCI Streaming– 与Object Storage集成

使用对象存储将大量流式数据加载到数据湖中，并运行分析



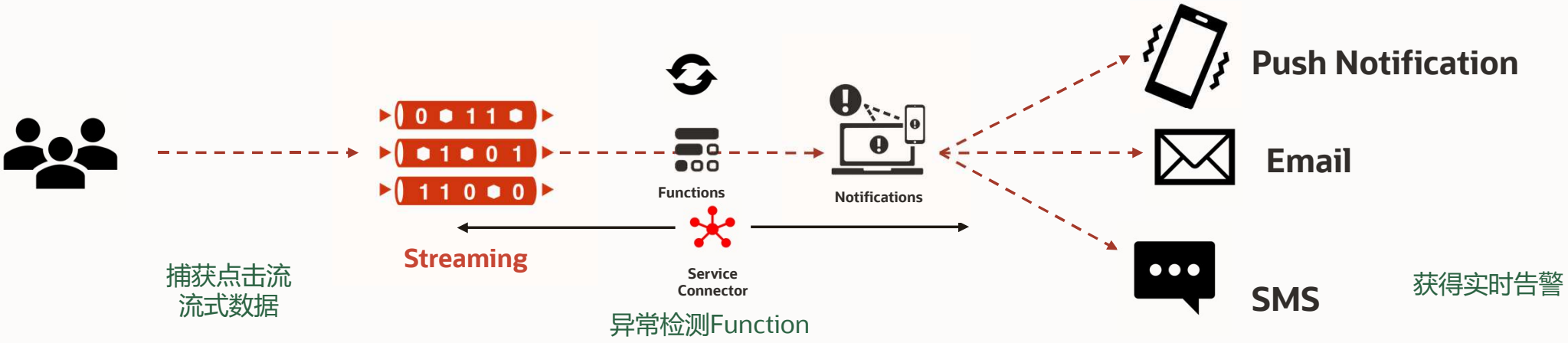
零管理 无需编写应用程序或管理基础架构，即可将流数据直接传送到对象存储。

无缝弹性 无缝扩展以匹配数据吞吐量，无需干预

使用 Oracle Function的无服务器 ETL：无缝调用 Oracle Function以在发送到对象存储之前转换传入的源数据

OCI Streaming– 与Notifications集成

创建可扩展的，健壮的高吞吐量实时通知管道



- **易于使用** 从流中轻松构建事件序列，例如点击流中的用户会话或通过日志提取出的用户行为
- **端到端无服务器** 识别感兴趣的事件（或一系列事件）并通过警报和通知对数据做出反应，无需管理任何基础设施。

OCI Streaming– 与第3方产品集成

Streaming Service可以使用 Kafka ConnectTM 框架连接 1

节约成本

提供业内最佳的性价比组合

OCI Streaming –减少总的数据摄取和数据移

通过按使用量计费的模式提供在线弹性 +

减少峰值/平均值负载的总体成本

+

无数据移动成本+

减少40%的运行时成本

+

无维护成本

减少昂贵的运维成本

更重要的是:

- 按使用量定价与竞争对手预置定价的比较
- 简单且最小的定价维度

+ Oracle唯一

OCI Streaming – 计费模式

- Oracle 云基础设施流式处理 PUT/GET：
  - 价格为每 GB PUT/GET 的数据 0.025 美元
    （PUT 和 GET 记录的大小四舍五入到最接近的 4KB 倍数）

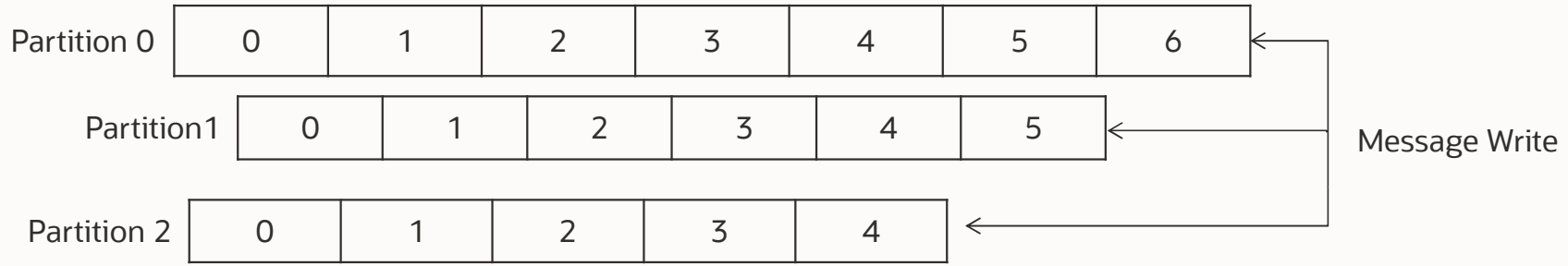
- Oracle 云基础设施流式存储：
  - 价格 $0.0002 每 GB 每小时

议程

- 从Kafka谈起
- OSS介绍
  - 云原生，基于开源软件，安全，与OCI及第3方产品深度集成，按使用量计费
- **OSS 核心概念**
- OSS Kafka 兼容API示例
- OSS 适用场景
- OSS 客户案例

OCI Streaming – 核心概念

- **Message**: 64 位编码记录或字节数组
- **Key**: 对相关消息进行分组的标识符
- **Stream**: append-only的消息日志
- **Partitions**: : 数据流还被分解成多个分区



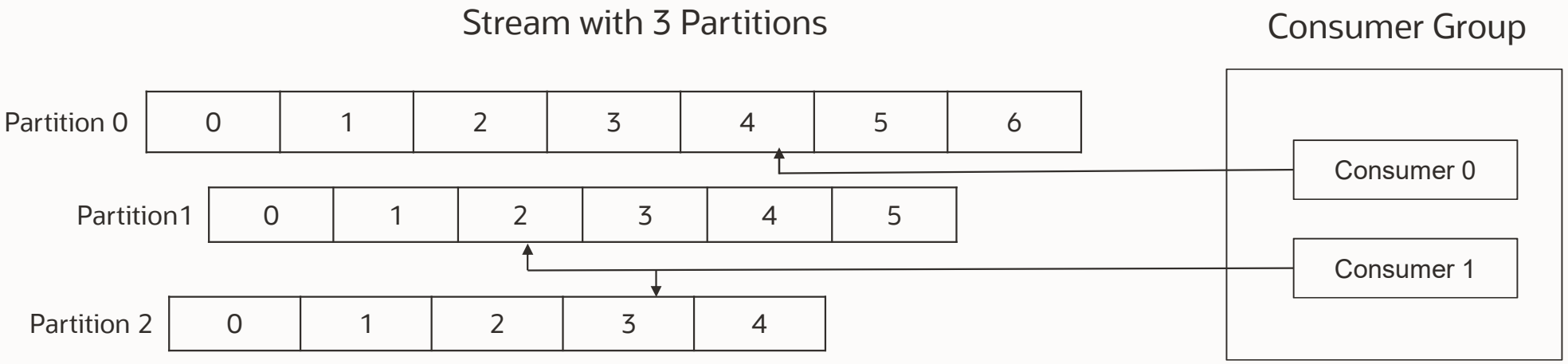
➢ 每个分区可以托管在不同的服务器上（不同的 AD，在一个region内），这意味着单个流可以跨多个服务器水平扩展，以提供远远超出单个服务器能力的性能

OCI Streaming – 核心概念

- **Producer**: 创建新消息的实体。 一般来说，一条消息被写入一个特定的分区
- **Consumer**: 从一个或多个流中读取消息的实体
  - 一般来说，消费者订阅一个或多个分区，并按照消息产生的顺序读取消息
  - 消费者通过跟踪消息偏移量来跟踪它已经消费了哪些消息
  - **Offset:** 消息在流/分区中的位置
  - 通过存储最后一条消费消息的偏移量，消费者可以停止和重启

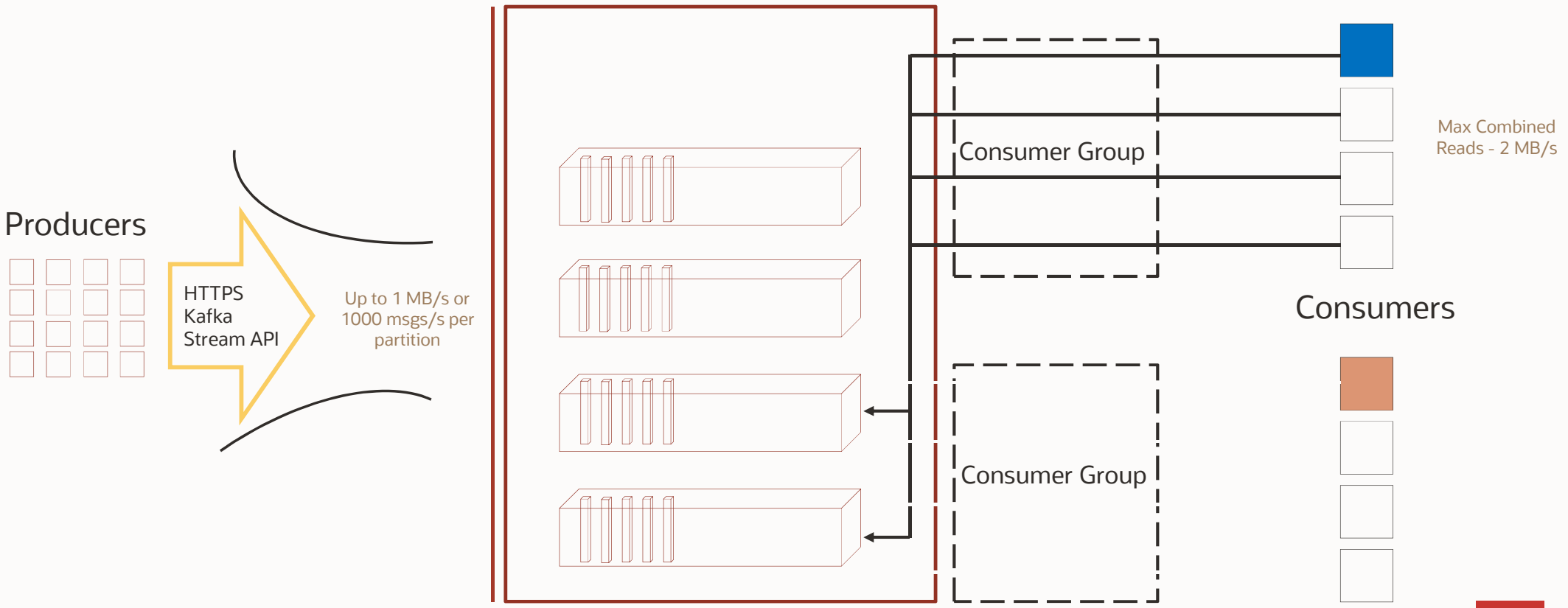
- **Consumer Group**: 消费者组是一组消费者，它们协调消费来自流中所有分区的消息
  - 每个分区仅由组中的一个成员使用

OCI Streaming – Consumer Group



- **Consumer Group**: 消费者组是一组消费者，它们协调消费来自流中所有分区的消息
- Consumer Groups 提供:
  - Rebalancing –随着新消费者加入或离开群体
  - Load balancing

OCI Streaming – 概念架构



议程

- 从Kafka谈起
- OSS介绍
  - 云原生，基于开源软件，安全，与OCI及第3方产品深度集成，按使用量计费
- OSS 核心概念
- **OSS Kafka 兼容API示例**
- OSS 适用场景
- OSS 客户案例

OCI Streaming Kafaka 兼容API示例 - 生产

Kafka API

```
import java.util.Properties;import
org.apache.kafka.clients.producer.Producer;import
org.apache.kafka.clients.producer.KafkaProducer;import
org.apache.kafka.clients.producer.ProducerRecord;


String topicName = "yangluTestStream";
Properties properties = new Properties();


properties.put("bootstrap.servers",
"10.1.0.156:9092,10.1.2.140:9092,10.1.2.187:9092");
properties.put("key.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
properties.put("value.serializer",
"org.apache.kafka.common.serialization.StringSerializer");


Producer<String, String> producer = new KafkaProducer<String,
String>(properties);


String str =  getRandomString (istr_size);
int i=0;
for(; i < iMsg; i++)
    producer.send(new ProducerRecord<String, String>(topicName,
  "key:"+Integer.toString(i), "value:"+str+Integer.toString(i)));
System.out.println("Message sent successfully, count: "+ i);
producer.flush();
producer.close();
```

OSS AP

```
String topicName = "yangluTestStream";
Properties properties = new Properties();


properties.put("bootstrap.servers", "cell-1.streaming.ap-seoul-
1.oci.oraclecloud.com:9092");
properties.put("security.protocol", "SASL_SSL");
properties.put("sasl.mechanism", "PLAIN");
properties.put("sasl.jaas.config",
"org.apache.kafka.common.security.plain.PlainLoginModule required
username=\"sehubjapacprod/oracleidentitycloudservice/yang.x.lu@oracle
.com/ocid1.streampool.oc1.ap-seoul-
1.amaaaaaaak7gbriajzeiregi6e4zr2jvdxtmjkniadd2sngyzmphy4emurya\"
password=\"FOlDWp6bc+iC5c#zZ1:C\";");
properties.put("key.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
properties.put("value.serializer",
"org.apache.kafka.common.serialization.StringSerializer");


Producer<String, String> producer = new KafkaProducer<String,
String>(properties);


String str =  getRandomString (istr_size);
int i=0;
for(; i < iMsg; i++)
 producer.send(new ProducerRecord<String, String>(topicName,
   "key:"+Integer.toString(i), "value:"+str+Integer.toString(i)));
 System.out.println("Message sent successfully, count: "+ i);
 producer.flush();
 producer.close();
```

OCI Streaming Kafka 兼容API示例 - 消费者

Kafka API

```
String topic = "yangluTestStream";

Properties props = new Properties();
props.put("bootstrap.servers",
"10.1.0.156:9092,10.1.2.140:9092,10.1.2.187:9092");
props.put("group.id", "group-0");
props.put("enable.auto.commit", "true");
props.put("auto.commit.interval.ms", "1000");
props.put("session.timeout.ms", "30000");
props.put("key.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
props.put("value.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
KafkaConsumer<String, String> consumer = new KafkaConsumer<String,
String>(props);

consumer.subscribe(Arrays.asList(topic));
while (True) {
  ConsumerRecords<String, String> records =
consumer.poll(Duration.ofMillis(1000));
 for (ConsumerRecord<String, String> record : records)
 System.out.printf("offset = %d, key = %s, value = %s\n",
    record.offset(), record.key(), record.value());
}
consumer.close();
```

OSS API

```
String topic = "yangluTestStream";

Properties props = new Properties();
props.put("bootstrap.servers", "cell-1.streaming.ap-seoul-
1.oci.oraclecloud.com:9092");
props.put("security.protocol", "SASL_SSL");
props.put("sasl.mechanism", "PLAIN");
props.put("sasl.jaas.config",
"org.apache.kafka.common.security.plain.PlainLoginModule required
username=\"sehubjapacprod/oracleidentitycloudservice/yang.x.lu@oracle.co
m/ocid1.streampool.oc1.ap-seoul-
1.amaaaaaaak7gbriajzeiregi6e4zr2jvdxtmjkniadd2sngyzmphy4emurya\"
password=\"FOlDWp6bc+iC5c#zZ1:C\";");
props.put("group.id", "group-0");
props.put("enable.auto.commit", "true");
props.put("auto.commit.interval.ms", "1000");
props.put("session.timeout.ms", "30000");
props.put("key.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
props.put("value.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
KafkaConsumer<String, String> consumer = new KafkaConsumer<String,
String>(props);

consumer.subscribe(Arrays.asList(topic));
while (True) {
  ConsumerRecords<String, String> records =
consumer.poll(Duration.ofMillis(1000));
 for (ConsumerRecord<String, String> record : records)
 System.out.printf("offset = %d, key = %s, value = %s\n",
    record.offset(), record.key(), record.value());
}
consumer.close();
```

议程

- 从Kafka谈起
- OSS介绍
  - 云原生，基于开源软件，安全，与OCI及第3方产品深度集成，按使用量计费
- OSS 核心概念
- OSS Kafka 兼容API示例
- **OSS 适用场景**
- OSS 客户案例

OCI Streaming– 用例

与本地应用程序和数据库的集成

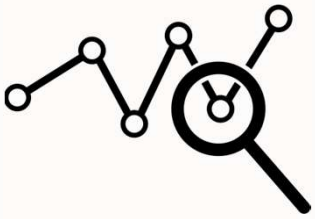
示例:

- **数据变更抓取:** 从本地数据库捕获 CDC 日志并将其备份到 OCI 对象存储上.
- **SaaS 连接:** 集成 Oracle EBS、Peoplesoft、Salesforce 等本地 SaaS 应用程序，使用 Kafka 连接器通过 OIC 和 OSS 连接到云。

云原生高吞吐消息总线

示例:

- **微服务:** 使用消息来解耦大型系统的组件，以构建健壮的微服务架构。
- **Kappa Architectures**: 使用 Streaming 和 DataFlow 聚合批处理和流管道，以创建强大的数据处理管道。



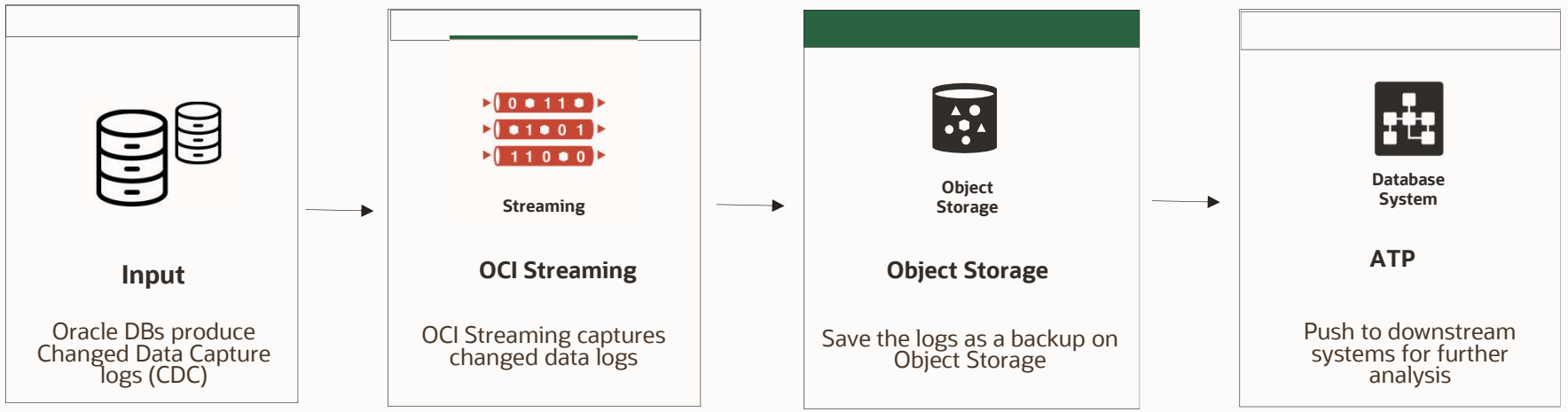
实时分析引擎

示例:

- **应用日志:** 通过 OSS 流式传输 OCI 服务和自定义应用程序日志，并使用 Kafka connect for Splunk，实时分析它们。
- **E-commerce orders:** 将电子商务交易订单流式传输到 OSS，使用 Oracle Functions 处理它们并将数据存储在后端数据库中。

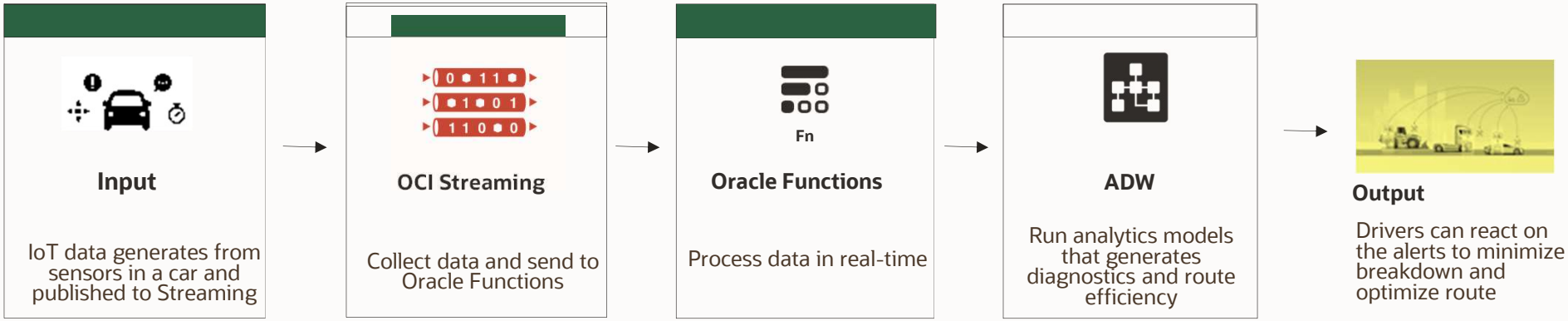
用例1 - Oracle DB 使用Streaming抓取数据变更

- 使用 OCI Streaming捕获从本地数据库生成的更改数据，并使其可供其他云和本地系统使用以使用该数据



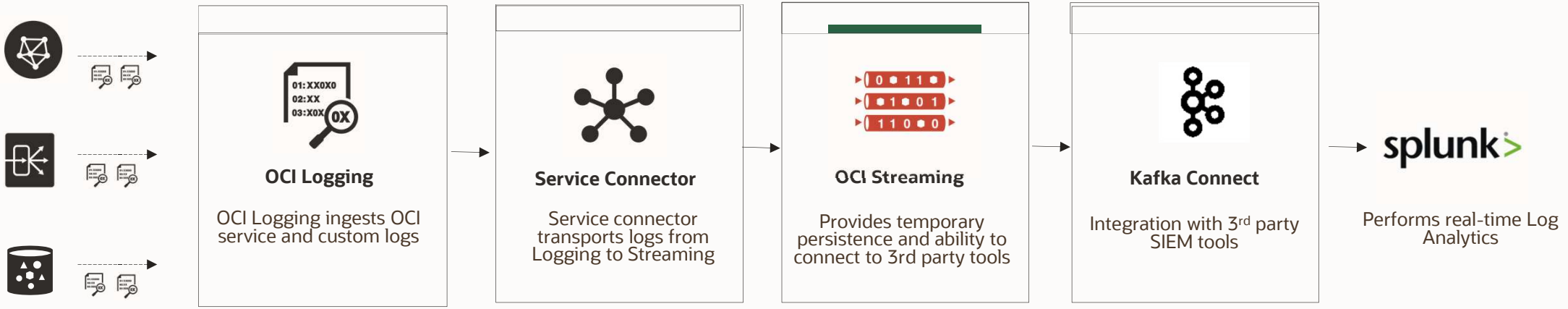
用例2 - Analyze IoT data using Streaming

- 使用 OCI 流捕获从 IoT 设备生成的事件，并使用 Oracle Functions 和 ADW 的处理能力来生成洞察以发送警报。
- 例如：自动驾驶汽车



用例3 - Real-Time Log Analytics

通过 OSS 流式传输 OCI 服务和自定义应用程序日志，并使用 Kafka connect for Splunk 实时分析它们。



用例4 – Streaming as a High Throughput Message Bus

- 使用 OCI 流作为异步消息传递通道，以松散耦合分布式 SaaS 应用程序的组件并提供实时数据传输

议程

- 从Kafka谈起
- OSS介绍
  - 云原生，基于开源软件，安全，与OCI及第3方产品深度集成，按使用量计费
- OSS 核心概念
- OSS Kafka 兼容API示例
- OSS 适用场景
- **OSS 客户案例**

LOLC FINANCE

LOLC 集团是斯里兰卡最大的非银行金融机构和最多元化的企业集团之一。 除了一系列金融产品和服务外，他们的产品组合还包括休闲、种植园和农业投入品。

用例: Change Data Capture


ORACLE