张振，KAFKA MEETUP

Jitney, Kafka at Airbnb

关于Airbnb爱彼迎

成立于2008年8月，爱彼迎总部位于加利福尼亚州旧金山市。爱彼迎是一个值得信赖的社区型市场，在这里人们可以通过网站、手机或平板电脑发布、发掘和预订世界各地的独特房源。

房客总数
超过
150,000,000个

城市
超过65,000个

城堡
超过1,400个

国家
超过191个

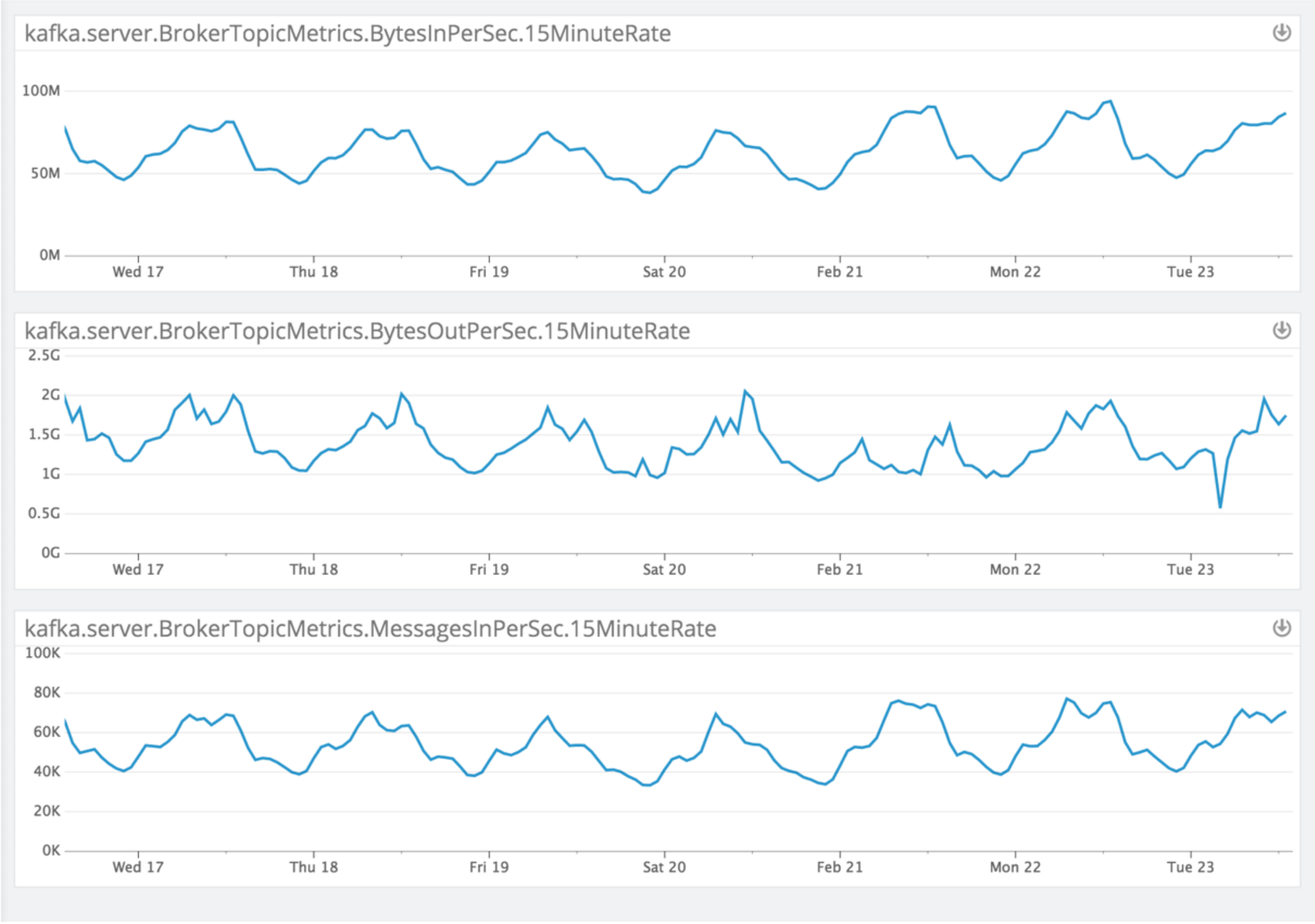
全球房源
3,000,000+

Jitney ?!

a bus carrying passengers for a low fare

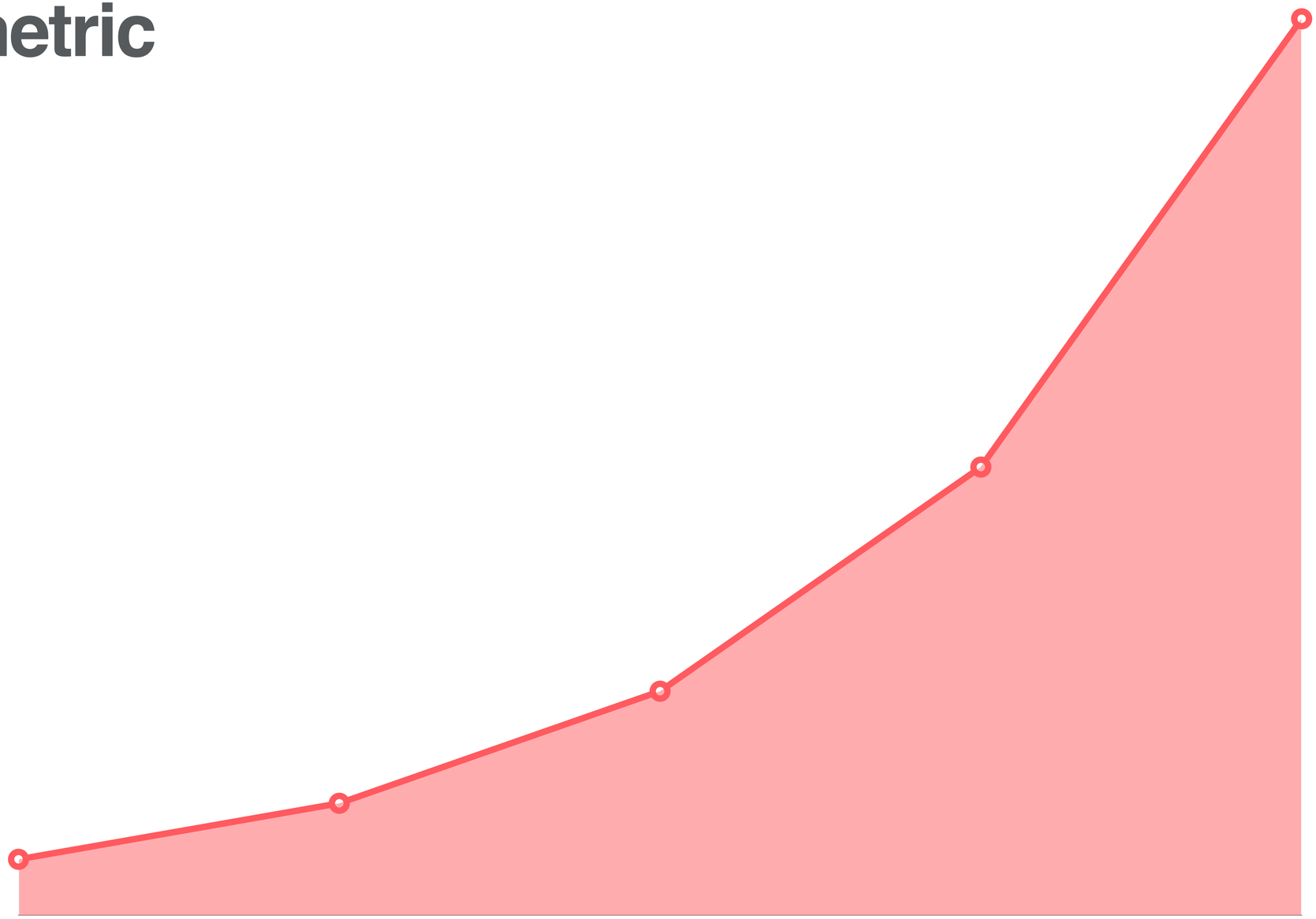
Some Kafka Facts

- 1 Production cluster
- v0.8.2
- 90 “small” brokers, d2.2xlarge
- 70 topics
- Replication Factor of 3
- 5 Billions events / day
- IN: 80MB / second
- OUT: 1.5GB / second
- Network bound
- Super stable



Why Jitney?

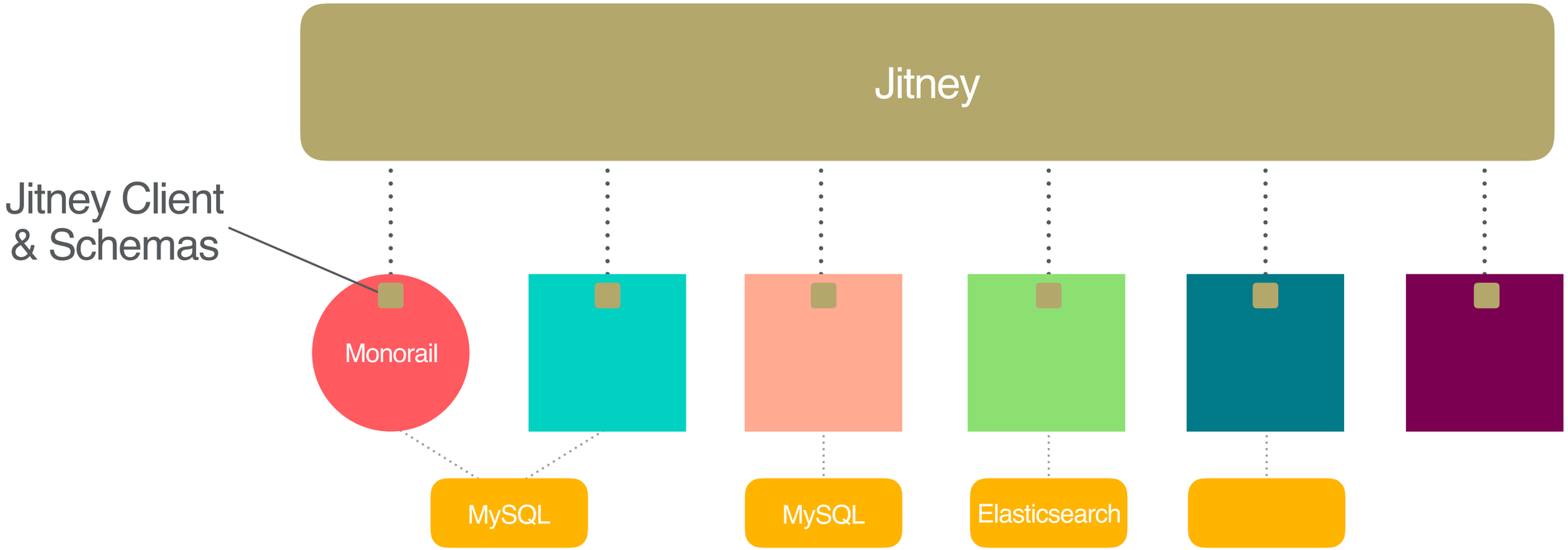
Pick any metric



Standardization!

What Use Cases ?

Classic Message Bus



Message Bus

- Decouple Services
- Standard Events
- At-least once delivery
- Standard clients, for Java and Ruby
- Easy to use
- Conventions over configuration

User Activity Logging

- Site and image load times, OOM events
- Searches, requests, bookings, etc.
- Experiment assignments

- Event data is critical for building data products
- Data ingestion should be reliable: timely and complete

Challenges

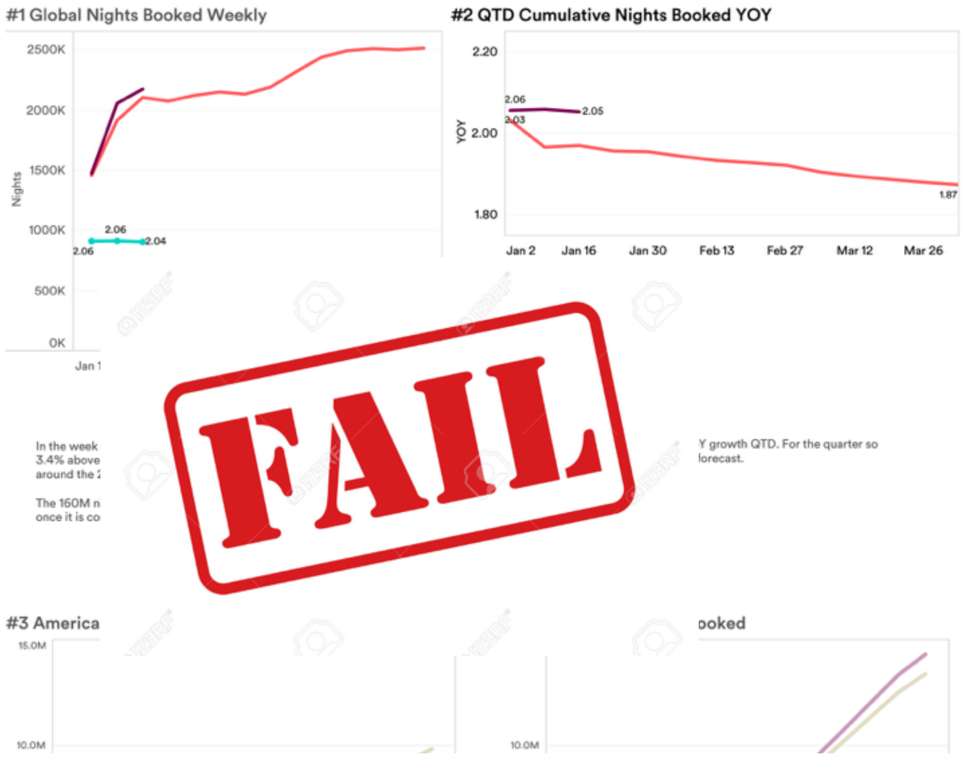
- JSON events without schemas
- Easy to break events during evolution/code changes
- One topic overall for 800+ event types
- Improper producer configs
- Lack of monitoring

- Lead to:
- Too many data outages, data loss incidents
- Lack of trust on data systems

Data Stability

A Year Ago

CEO dashboard and Magical booking dashboards were regularly broken.



Data Stability

A Year Ago

ERF was unstable and experimentation culture was weak

Hi team,

This is partly a PSA to let you know ERF dashboard data hasn't been up to date/ accurate for several weeks now. **Do not rely on the ERF dashboard for information about your experiment.**

Jitney Components

Jitney Components

Schema Repository

Thrift Schema Repository

Why Thrift?

- Easy syntax
- Good performance in Ruby
- Ubiquitous

Advantages of schema repo?

- Great Catalyst for communication, documentation, etc
- it ships jar and gems

- **Will developers hate you for this? no**

Schema Evolution

- **Standard Field in the event schema**
- **Managed Explicitly**
- **use Semantic Versioning:**

1.0.0 = MODEL . REVISION . ADDITION

MODEL is a change which breaks the rules of backward compatibility.

Example: changing the type of a field.

REVISION is a change which is backward compatible but not forward compatible.

Example: adding a new field to a union type.

ADDITION is a change which is both backward compatible and forward compatible.

Example: adding a new optional field.

Example of Thrift Event

```
# package names must include the major version
namespace java com.airbnb.jitney.event.di_canary.v1
namespace rb airbnb.jitney.event.di_canary.v1

struct CanaryEvent {
    31337: optional string schema = "com.airbnb.jitney.event.di_canary:CanaryEvent:1.0.0"

    1: required i64 canary_created_at,
    2: required string canary_role,
    3: required string canary_seqid,
    4: required i64 canary_seqn,
    5: optional map<string, string> extras,
}
```

Jitney Components

Schema Repository

Topic Repository

Topic Repository

- **Declare all Jitney topics**
- Aggregate all **characteristics** of a topic:
  - name
  - ordering (partitioning function)
  - whitelist of accepted schemas
- Great for documentation purposes
- DRY

Example of a Topic

```
JitneyTopic.builder()
    .withRouter(SpinaltapHelper::getRouter)
    .setName("spinaltap")
    .addWhitelistedSchemas(
        "com.airbnb.jitney.event.spinaltap.v1",
        "com.airbnb.jitney.event.spinaltap.v2"
    )
    .withKeyProvider(SpinaltapHelper::getPartitionKey)
    .build()
```

Jitney Components

Jitney Clients

- **Kafka clients** are hard to use correctly
- it’s better with 0.9

- **Committing offsets** is tricky, someone will get it wrong
- even with 0.9

- **Configuration** is a mess

Jitney Clients

it provides:

- metrics reporting: github.com/airbnb/kafka-statsd-metrics2
- configuration for default clusters
- built-in support for Schema Repository and Topic Repository

Consumer:

- offset management to implement at-least once delivery
- polymorphic dispatching to event handler

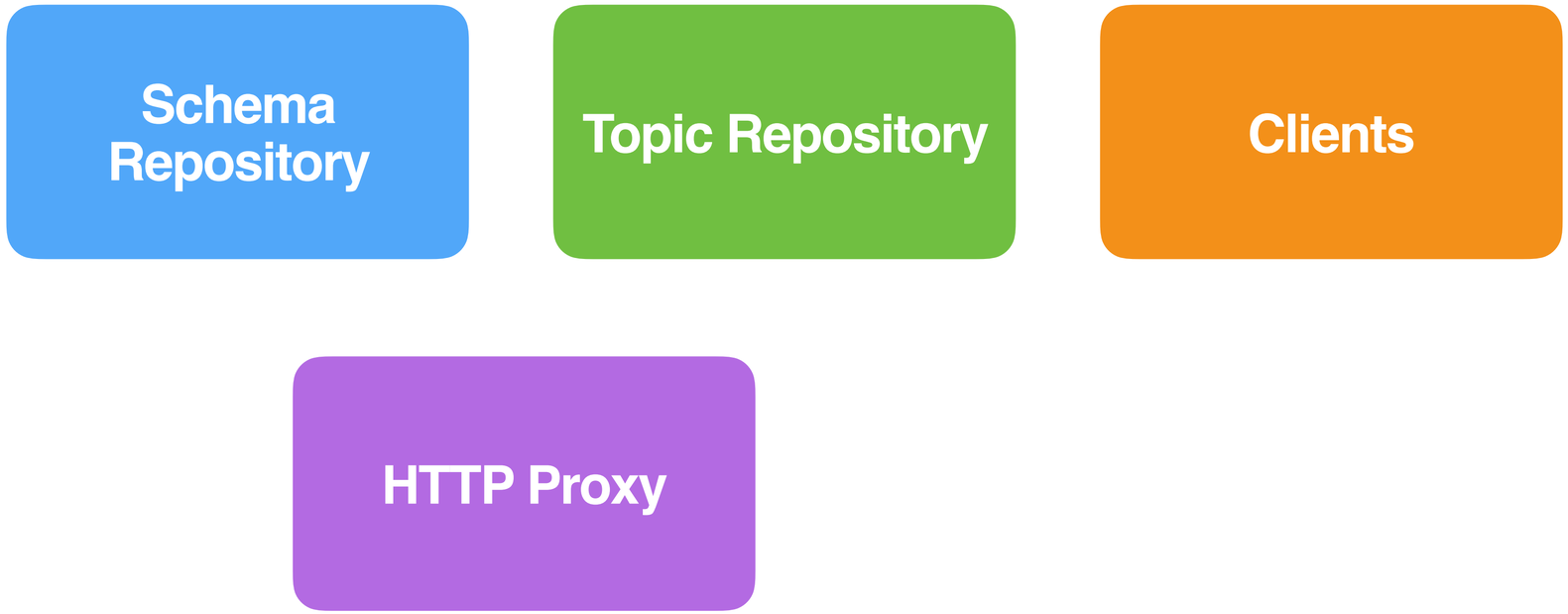
Example of a Java Producer

```
55    JitneyProducer producer = JitneyProducerFactory.create(
56            new ProducerConfig("my-client-id"),
57            new JitneyBusConfig("localhost"),
58            JitneyRepository.getTopic("canary")
59          );
60
61     producer.startAsync();
62
63     // publish
64     final TBase event = new CanaryEvent(...);
65     producer.publish(event);
66
67     producer.stopAsync();
```

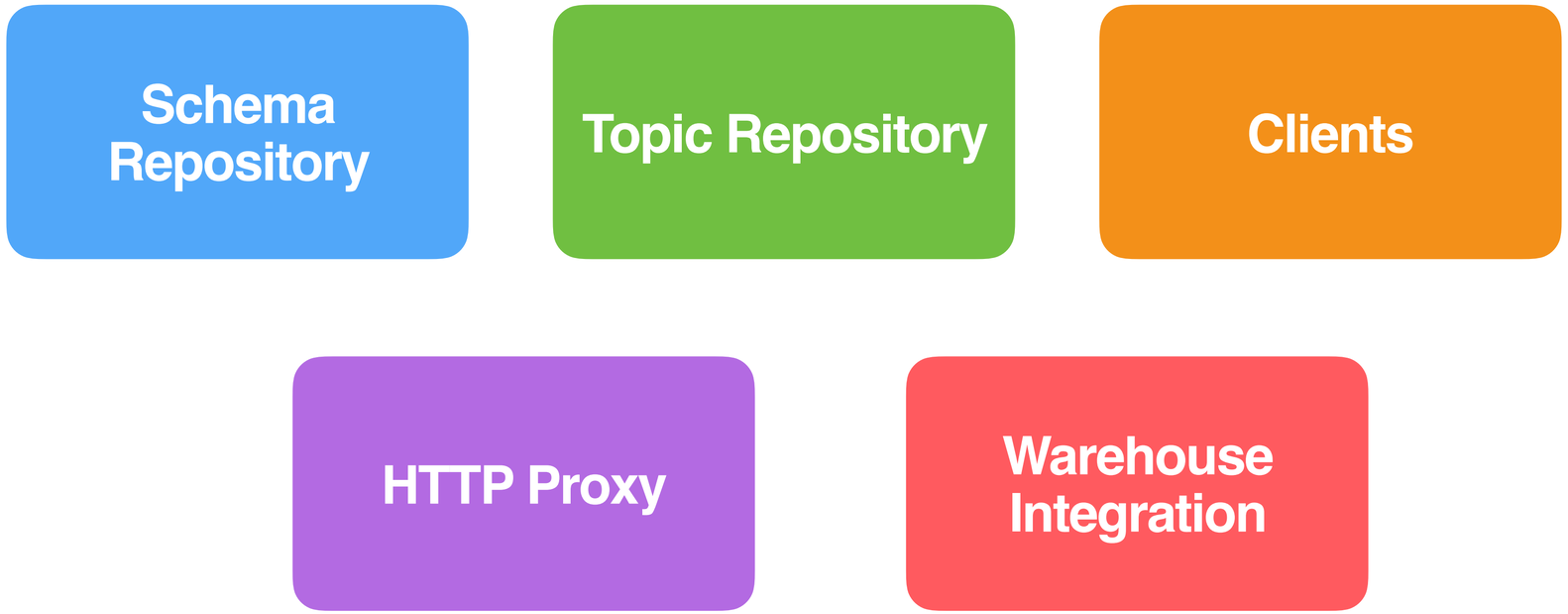
Example of a Java Consumer

```
54
55      final Dispatcher dispatcher = ClassBasedDispatcher.builder()
56          .addHandler(CanaryEvent.class, new Handler<CanaryEvent>(){
57            public void handle(Message<CanaryEvent> message) {
58                System.out.println("message received!");
59            }
60          )
61          .build();
62
63      consumer = JitneyConsumerFactory.create(
64          ...,
65          ...,
66          Arrays.asList(Subscription.of(Route.of("canary"), dispatcher)));
67
68
69      consumer.startAsync();
70      // worker threads are created and process incoming messages
```

Jitney Components



Jitney Components

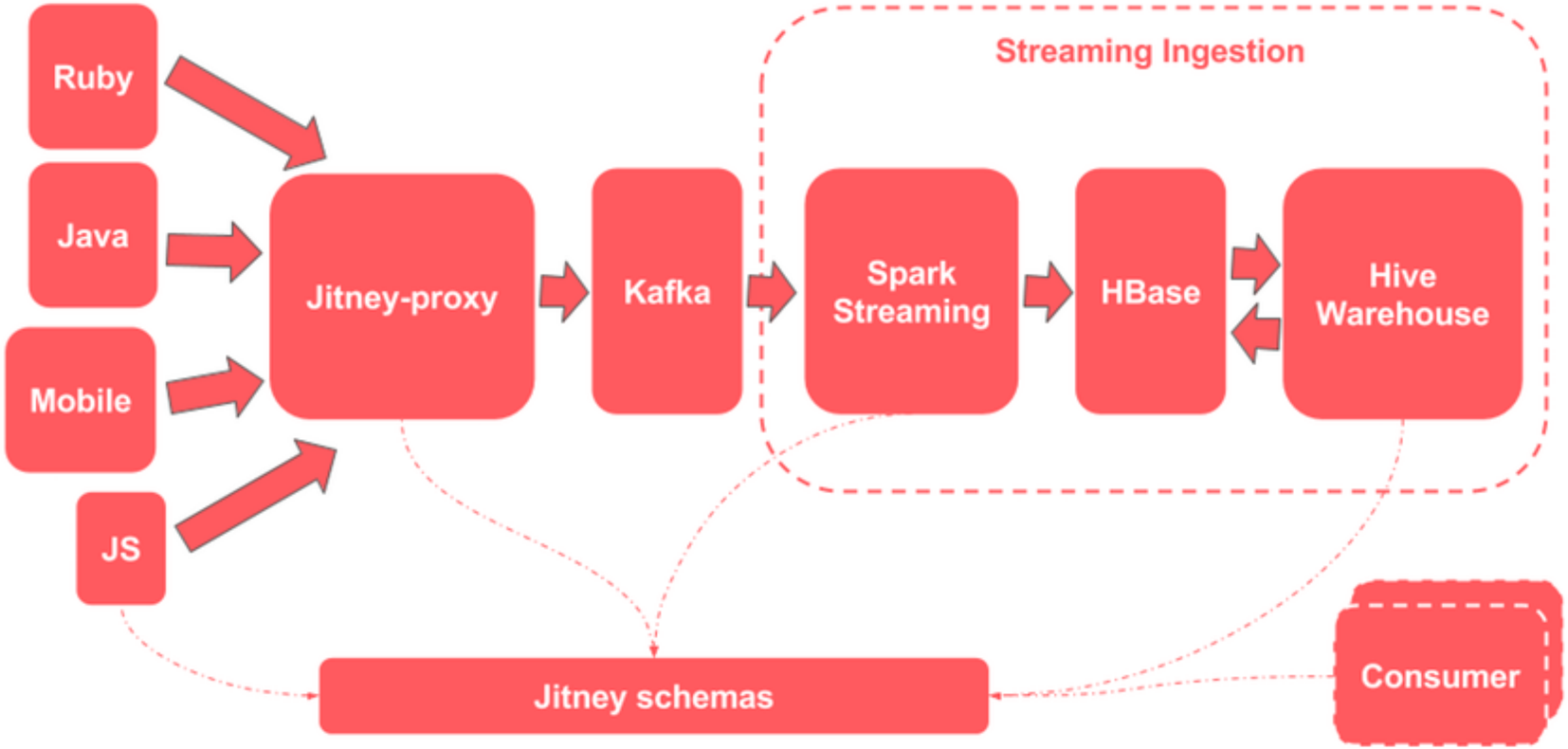


Data Ingestion Pipeline

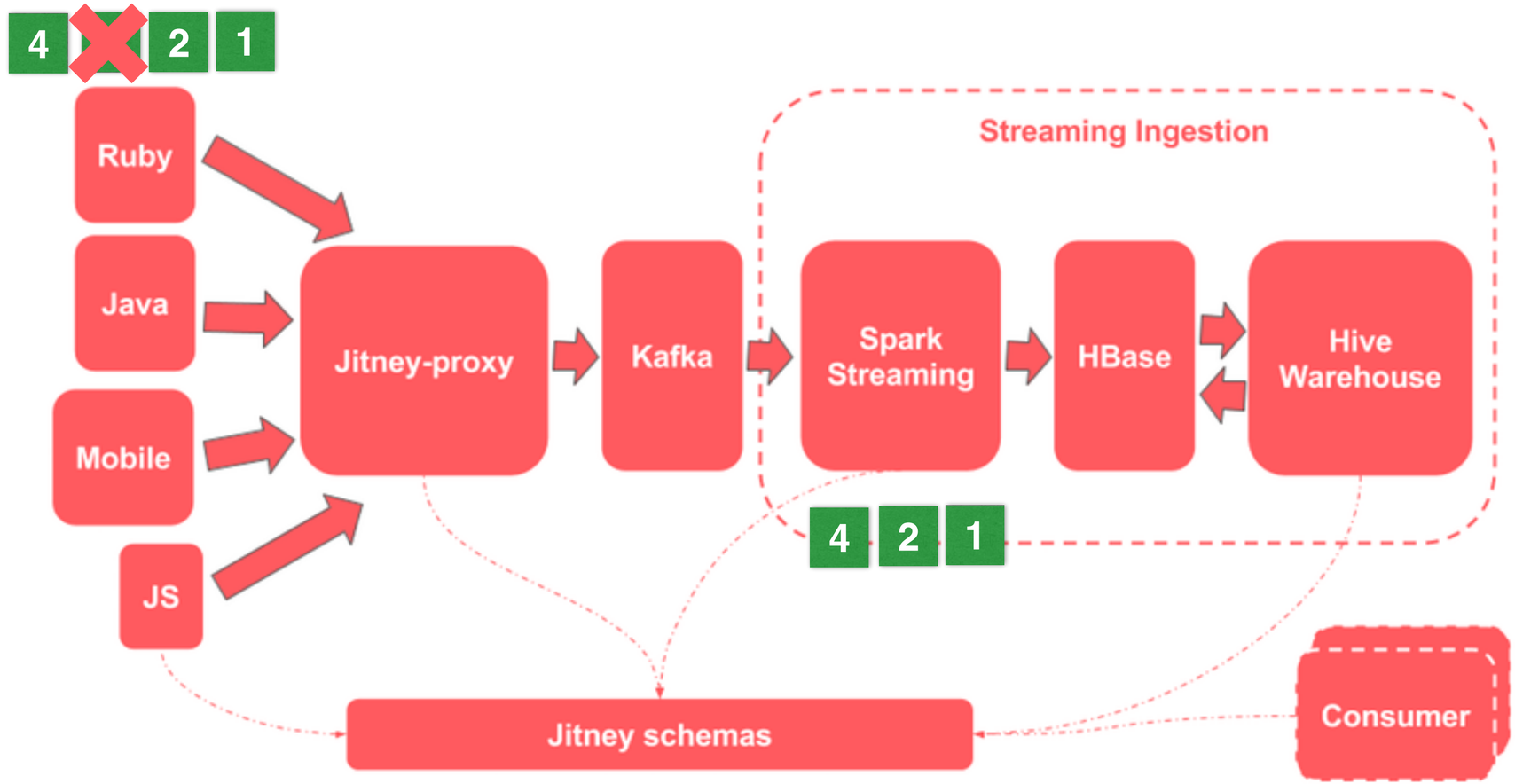
- Stack: Jitney, Spark Streaming, HBase, HDFS
- Spark Streaming 1.5 with Kafka “direct” connect
- Process 1 minute batches
- Write to HBase after deserializing with the right schema
- Dump data to HDFS every hour (with dedup) and add a Hive partition
- But live data can be queried via “current” partition

Data Ingestion Pipeline

end to end



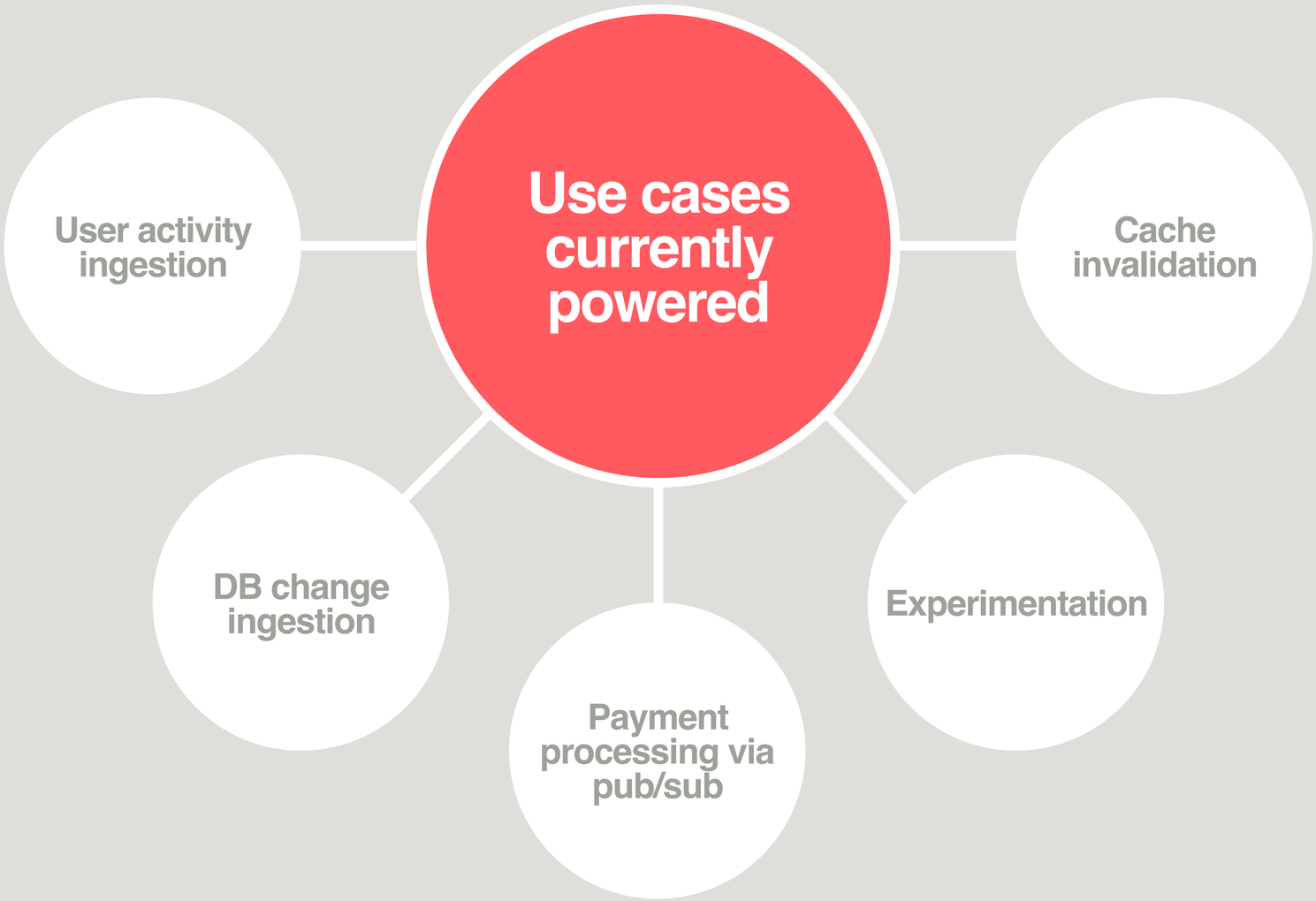
Audit


4
2
1
Ruby
Java
Mobile
JS
Jitney-proxy
Kafka
Streaming Ingestion
Spark Streaming
HBase
Hive Warehouse
4
2
1
Jitney schemas
Consumer

Event Schema for Audit Metadata


```
{
  "eventSchema":"com.airbnb.jitney.event.di_canary:CanaryEvent:1.0.0",
  "id":"a941b1d3-67c8-41b5-af20-adb176bb53ee",
  "tiers":[
    {
      "id":"e2d43d8b-d23e-4bde-907f-7526ffd41a24",
      "type":"jitney-ruby-client",
      "hostname":"i-8c516c5f",
      "ipAddress":"10.169.141.54",
      "timestamp":1456262927715,
      "sequences":[
        {
          "key":[
            "global"
          ],
          "value":0,
          "bytesSent":141
        }
      ]
    },
    { }
  ]
}
```

How is Jitney used in the org?


Use cases currently powered
User activity ingestion
Cache invalidation
DB change ingestion
Payment processing via pub/sub
Experimentation

Key take aways

1

Standardization!

2

Auditing Pipeline

3

Huge Advantage for the organization

Join Airbnb

Learn More

https://www.airbnb.com/careers/locations/beijing-china