Kafka Connect

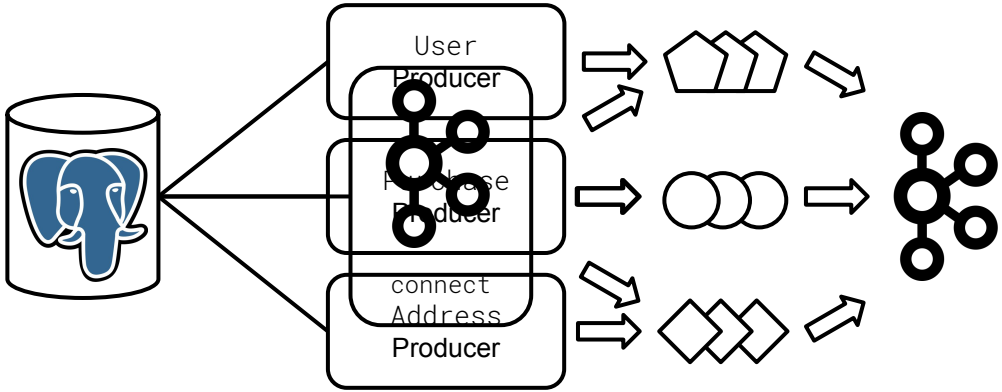
Kafka Connect

A **framework** and **web server** for reusable producers and consumers

- Built at Confluent, now with Apache Foundation
- Built in Java and Scala, runs on the JVM
- Can help promote reusability and keep code simple
- May not even need Kafka Client with Kafka Connect

Kafka Connect

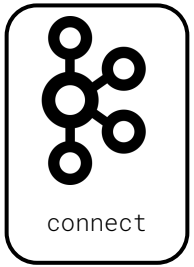
Example: Connect's SQL functionality can save significant effort



Kafka Connect Architecture

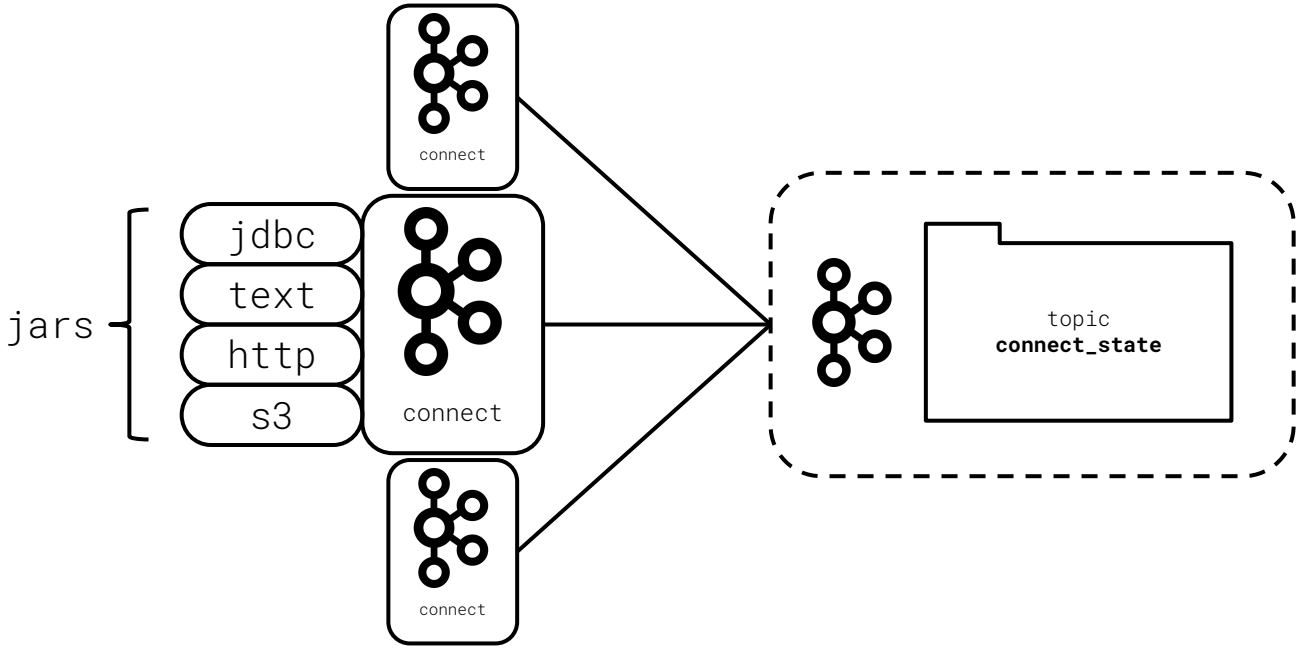
How Kafka Connect Works

Kafka Connect, both webserver and framework, are built on the JVM



How Kafka Connect Works

Kafka Connect, both webserver and framework, are built on the JVM



The Connect Framework

Components

- **Connectors** are abstractions for managing tasks
- **Tasks** contain the production or consumption code
- Kafka and target systems often have different formats
- **Converters** map data formats to and from Connect

Kafka Connect Connectors

Kafka Connect Connectors

Kafka Connect Connectors

Common Connectors

- **Local file source/sink**, ex: useful for logs
- **Cloud Key Value Store source/sink**, ex: AWS S3
- **JDBC source/sink**, ex: PostgreSQL, MySQL
- **HDFS source/sink**, ex: Interacting with Hadoop
- Visit Confluent's web repository for many more!

The Kafka Connect API

Kafka Connect API

API Capabilities

- Connect is entirely managed through a REST API
- Supports Create, Update, Delete, Read on Connectors
- Can add or remove plugins while server is running
- API aids in monitoring Connectors
- Does not surface logs or metrics

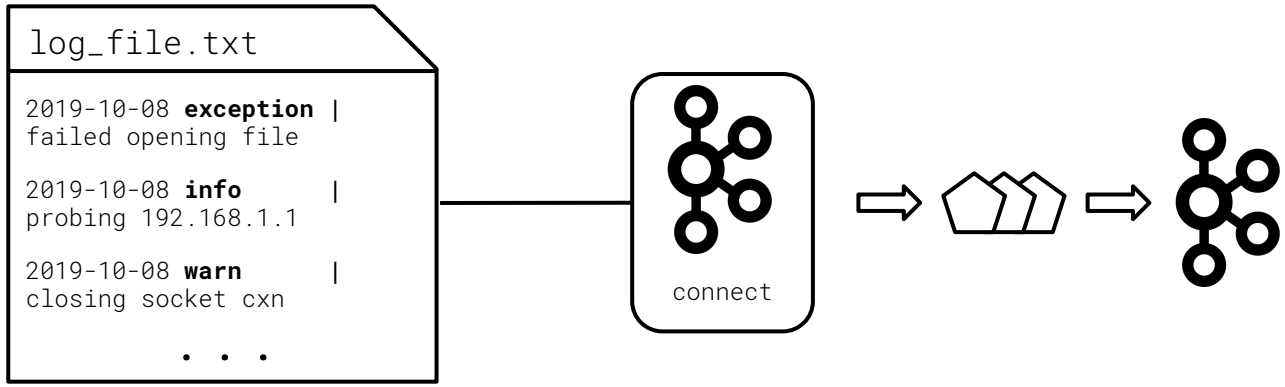
Working with the Kafka Connect API

Demo: Connect API

Key Connectors

Kafka Connect FileStream Source

FileStream Source connector **sends logs** as events to Kafka



Demo: Kafka Connect FileStream

JDBC Source/Sink

Overview

- Uses standard Java SQL Interface (JDBC)
- JDBC Source puts data from a DB *into* Kafka
- JDBC sink takes data *out of* Kafka and into a DB
- Supports table whitelists, blacklists, custom queries
- Commonly used to publish model updates as events
- Commonly used to store Kafka events long term

Demo: Kafka Connect JDBC Source

Kafka REST Proxy

REST Proxy

Overview

- Web server written in Java and Scala, runs on JVM
- May run as one instance or in a cluster
- Allows publish/consumption to Kafka over HTTP REST
- Cannot create topics, may only GET topic metadata
- Can be integrated with Schema Registry and use Avro
- Best in scenarios where you can't use a client library

Demo: Fetching Cluster Metadata from REST Proxy

Using REST Proxy

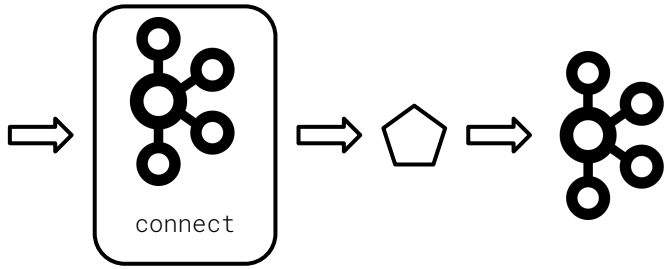
Producing Data with REST Proxy

POST data to Kafka REST Proxy to produce data

```
POST /topics/<topic_name>
```

```
Content-Type:
application/vnd.kafka.json.v2+json
```

```
{
  “records”: [
    {“value”: {“user”: “123”},
    {“value”: {“user”: “456”},
  ]
}
```



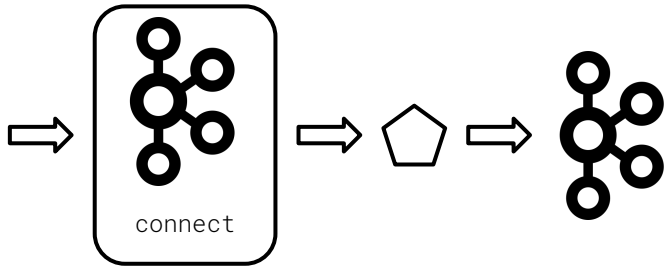
Producing Data with REST Proxy

Avro data may be published but you must always **include the schema**

```
POST /topics/<topic_name>
```

```
Content-Type:
application/vnd.kafka.avro.v2+json
```

```
{
  “value_schema”: “{\"type\": \”r...",
  “records”: [
    {“value”: {“user”: 123},
    {“value”: {“user”: 456},
  ]
}
```



Producing Data with REST Proxy

Always make sure you have the right Content-Type Header!

Data Serialization Format	Content-Type Header
Binary (Base64-encoded String, etc)	`application/vnd.kafka.`**`binary`**`.v2+json`
JSON	`application/vnd.kafka.`**`json`**`.v2+json`
Avro	`application/vnd.kafka.`**`avro`**`.v2+json`

Demo: Producing JSON Data with REST Proxy

Demo: Producing Avro Data with REST Proxy

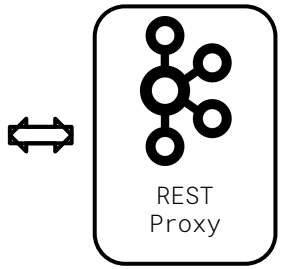
Consuming Data with REST Proxy

Consumption begins with a POST to create a Consumer Group

```
POST /consumers/<group_name>
HTTP 1.1 200 OK
```

```
Content-Type:
application/vnd.kafka.v2+json
```

```
{
  “instance_id”: “<consumer_name>”,
  “base_uri”: “http://<rest_uri>”
{
}
  “name”: “<consumer_name>”,
  “format”: “<binary/json/avro>”,
}
```



Consuming Data with REST Proxy

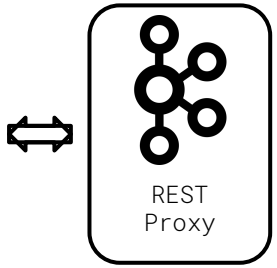
Next, POST to the subscribe endpoint

```
POST /consumers/<group_name>/instances/<instance_id>/subscription

Content-Type: application/vnd.kafka.v2+json

{
  "topics":[
    "<topic_0>",
    "<topic_1>",
    ... ,
    "<topic_n>"
  ]
}
```

```
HTTP 1.1 204 No Content
```



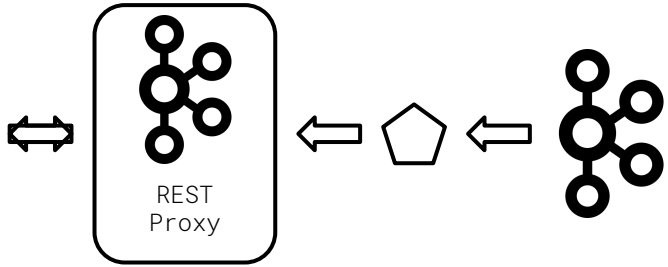
Consuming Data with REST Proxy

Once the consumer is subscribed, we can use HTTP GET to fetch records

```
GET /consumers/<group_name>/instances/<instance_id>/records
Accept: application/vnd.kafka.json.v2+json
```

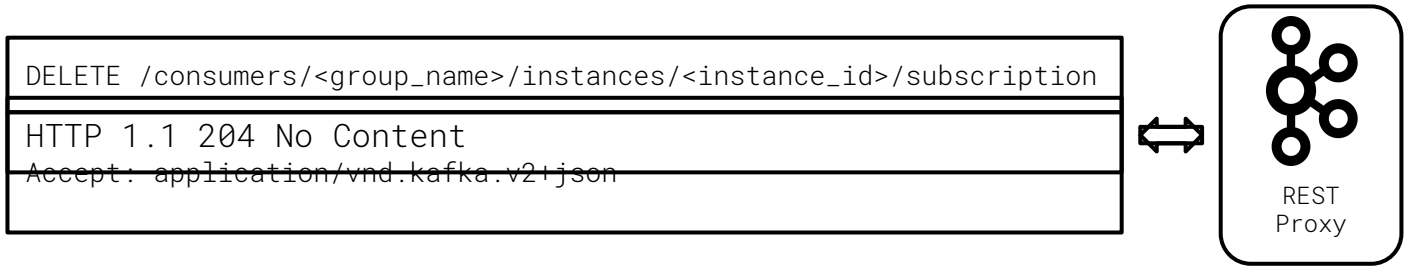
```
HTTP 1.1 200 OK

[
  {
    "topic": <topic_name>,
    "key": "<key>",
    "value": "<value>",
    "partition": <partition>,
    "offset": <offset>
  }, {
    ...
  }
]
```



Consuming Data with REST Proxy

DELETE your consumer subscription when shutting down



Demo: Consuming JSON Data with REST Proxy

Demo: Consuming Avro Data with REST Proxy

Summary