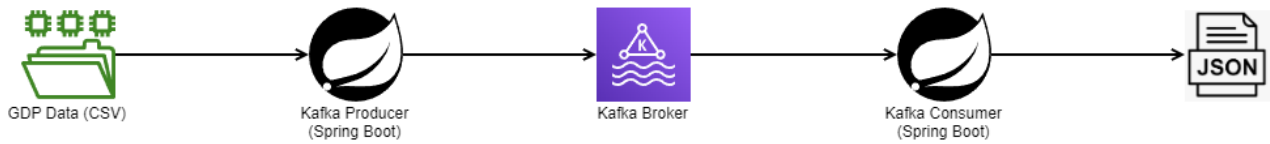
AWS POC 5 - Load Government Debt Total % GDP via Spring Boot & Kafka

Overview



Process Flow

1. Kafka Producer (Java) loads the CSV
2. Kafka Producer pushes the data record by record to the Kafka Broker
3. The Kafka Consumer (Java) listens and pulls the messages from the topic
4. The Kafka Consumer appends the record to the JSON file

Kafka Docker

docker-compose.yml

```yaml
---
version: '2'
services:

  broker:
    image: confluentinc/cp-kafka:7.6.0
    hostname: broker
    container_name: broker
    ports:
      - "9092:9092"
      - "9101:9101"
    environment:
      KAFKA_NODE_ID: 1
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: 'CONTROLLER:PLAINTEXT,PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT'
      KAFKA_ADVERTISED_LISTENERS: 'PLAINTEXT://broker:29092,PLAINTEXT_HOST://localhost:9092'
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_GROUP_INITIAL_REBALANCE_DELAY_MS: 0
      KAFKA_TRANSACTION_STATE_LOG_MIN_ISR: 1
      KAFKA_TRANSACTION_STATE_LOG_REPLICATION_FACTOR: 1
      KAFKA_JMX_PORT: 9101
      KAFKA_JMX_HOSTNAME: localhost
      KAFKA_PROCESS_ROLES: 'broker,controller'
      KAFKA_CONTROLLER_QUORUM_VOTERS: '1@broker:29093'
      KAFKA_LISTENERS: 'PLAINTEXT://broker:29092,CONTROLLER://broker:29093,PLAINTEXT_HOST://0.0.0.0:9092'
      KAFKA_INTER_BROKER_LISTENER_NAME: 'PLAINTEXT'
      KAFKA_CONTROLLER_LISTENER_NAMES: 'CONTROLLER'
      KAFKA_LOG_DIRS: '/tmp/kraft-combined-logs'
      # Replace CLUSTER_ID with a unique base64 UUID using "bin/kafka-storage.sh random-uuid"
      # See https://docs.confluent.io/kafka/operations-tools/kafka-tools.html#kafka-storage-sh
      CLUSTER_ID: 'MkU3OEVBNTcwNTJENDM2Qk'
```

```yaml

  schema-registry:
    image: confluentinc/cp-schema-registry:7.6.0
    hostname: schema-registry
    container_name: schema-registry
    depends_on:
      - broker
    ports:
      - "8081:8081"
    environment:
      SCHEMA_REGISTRY_HOST_NAME: schema-registry
      SCHEMA_REGISTRY_KAFKASTORE_BOOTSTRAP_SERVERS: 'broker:29092'
      SCHEMA_REGISTRY_LISTENERS: http://0.0.0.0:8081

  connect:
    image: cnfldemos/cp-server-connect-datagen:0.6.4-7.6.0
    hostname: connect
    container_name: connect
    depends_on:
      - broker
      - schema-registry
    ports:
      - "8083:8083"
    environment:
      CONNECT_BOOTSTRAP_SERVERS: 'broker:29092'
      CONNECT_REST_ADVERTISED_HOST_NAME: connect
      CONNECT_GROUP_ID: compose-connect-group
      CONNECT_CONFIG_STORAGE_TOPIC: docker-connect-configs
      CONNECT_CONFIG_STORAGE_REPLICATION_FACTOR: 1
      CONNECT_OFFSET_FLUSH_INTERVAL_MS: 10000
      CONNECT_OFFSET_STORAGE_TOPIC: docker-connect-offsets
      CONNECT_OFFSET_STORAGE_REPLICATION_FACTOR: 1
      CONNECT_STATUS_STORAGE_TOPIC: docker-connect-status
      CONNECT_STATUS_STORAGE_REPLICATION_FACTOR: 1
      CONNECT_KEY_CONVERTER: org.apache.kafka.connect.storage.StringConverter
      CONNECT_VALUE_CONVERTER: io.confluent.connect.avro.AvroConverter
      CONNECT_VALUE_CONVERTER_SCHEMA_REGISTRY_URL: http://schema-registry:8081
      # CLASSPATH required due to CC-2422
      CLASSPATH: /usr/share/java/monitoring-interceptors/monitoring-interceptors-7.6.0.jar
      CONNECT_PRODUCER_INTERCEPTOR_CLASSES: "io.confluent.monitoring.clients.interceptor.MonitoringProducerInte
      CONNECT_CONSUMER_INTERCEPTOR_CLASSES: "io.confluent.monitoring.clients.interceptor.MonitoringConsumerInte
      CONNECT_PLUGIN_PATH: "/usr/share/java,/usr/share/confluent-hub-components"
      CONNECT_LOG4J_LOGGERS: org.apache.zookeeper=ERROR,org.I0Itec.zkclient=ERROR,org.reflections=ERROR

  control-center:
    image: confluentinc/cp-enterprise-control-center:7.6.0
    hostname: control-center
    container_name: control-center
    depends_on:
      - broker
      - schema-registry
      - connect
      - ksqldb-server
    ports:
      - "9021:9021"
    environment:
      CONTROL_CENTER_BOOTSTRAP_SERVERS: 'broker:29092'
      CONTROL_CENTER_CONNECT_CONNECT-DEFAULT_CLUSTER: 'connect:8083'
```

```
      CONTROL_CENTER_CONNECT_HEALTHCHECK_ENDPOINT: '/connectors'
      CONTROL_CENTER_KSQL_KSQLDB1_URL: "http://ksqldb-server:8088"
      CONTROL_CENTER_KSQL_KSQLDB1_ADVERTISED_URL: "http://localhost:8088"
      CONTROL_CENTER_SCHEMA_REGISTRY_URL: "http://schema-registry:8081"
      CONTROL_CENTER_REPLICATION_FACTOR: 1
      CONTROL_CENTER_INTERNAL_TOPICS_PARTITIONS: 1
      CONTROL_CENTER_MONITORING_INTERCEPTOR_TOPIC_PARTITIONS: 1
      CONFLUENT_METRICS_TOPIC_REPLICATION: 1
      PORT: 9021

  ksqldb-server:
    image: confluentinc/cp-ksqldb-server:7.6.0
    hostname: ksqldb-server
    container_name: ksqldb-server
    depends_on:
      - broker
      - connect
    ports:
      - "8088:8088"
    environment:
      KSQL_CONFIG_DIR: "/etc/ksql"
      KSQL_BOOTSTRAP_SERVERS: "broker:29092"
      KSQL_HOST_NAME: ksqldb-server
      KSQL_LISTENERS: "http://0.0.0.0:8088"
      KSQL_CACHE_MAX_BYTES_BUFFERING: 0
      KSQL_KSQL_SCHEMA_REGISTRY_URL: "http://schema-registry:8081"
      KSQL_PRODUCER_INTERCEPTOR_CLASSES: "io.confluent.monitoring.clients.interceptor.MonitoringProducerInterce
      KSQL_CONSUMER_INTERCEPTOR_CLASSES: "io.confluent.monitoring.clients.interceptor.MonitoringConsumerInterce
      KSQL_KSQL_CONNECT_URL: "http://connect:8083"
      KSQL_KSQL_LOGGING_PROCESSING_TOPIC_REPLICATION_FACTOR: 1
      KSQL_KSQL_LOGGING_PROCESSING_TOPIC_AUTO_CREATE: 'true'
      KSQL_KSQL_LOGGING_PROCESSING_STREAM_AUTO_CREATE: 'true'

  ksqldb-cli:
    image: confluentinc/cp-ksqldb-cli:7.6.0
    container_name: ksqldb-cli
    depends_on:
      - broker
      - connect
      - ksqldb-server
    entrypoint: /bin/sh
    tty: true

  ksql-datagen:
    image: confluentinc/ksqldb-examples:7.6.0
    hostname: ksql-datagen
    container_name: ksql-datagen
    depends_on:
      - ksqldb-server
      - broker
      - schema-registry
      - connect
    command: "bash -c 'echo Waiting for Kafka to be ready... && \
                        cub kafka-ready -b broker:29092 1 40 && \
                        echo Waiting for Confluent Schema Registry to be ready... && \
                        cub sr-ready schema-registry 8081 40 && \
                        echo Waiting a few seconds for topic creation to finish... && \
                        sleep 11 && \
```

```
                        tail -f /dev/null'"
    environment:
      KSQL_CONFIG_DIR: "/etc/ksql"
      STREAMS_BOOTSTRAP_SERVERS: broker:29092
      STREAMS_SCHEMA_REGISTRY_HOST: schema-registry
      STREAMS_SCHEMA_REGISTRY_PORT: 8081

  rest-proxy:
    image: confluentinc/cp-kafka-rest:7.6.0
    depends_on:
      - broker
      - schema-registry
    ports:
      - 8082:8082
    hostname: rest-proxy
    container_name: rest-proxy
    environment:
      KAFKA_REST_HOST_NAME: rest-proxy
      KAFKA_REST_BOOTSTRAP_SERVERS: 'broker:29092'
      KAFKA_REST_LISTENERS: "http://0.0.0.0:8082"
      KAFKA_REST_SCHEMA_REGISTRY_URL: 'http://schema-registry:8081'
```

Start the Kafka Docker Server

```
docker-compose up -d
```

Java Spring Boot Application

RestServiceApplication.java

```
@SpringBootApplication
public class RestServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(RestServiceApplication.class, args);
    }

}
```

CountryGDPController.java

```
@RestController
@RequestMapping("/gdp")
public class CountryGDPController {
    @Autowired
    private CountryGDPMessagePublisher publisher;

    @GetMapping("/is_running")
    public GDPServiceStatus serviceStatus() {
        return new GDPServiceStatus("GDP Service", "Running");
    }

    @GetMapping("/simulate_process_file")
    public GDPDetailRecord simulateProcessFile() {
        try {
            var gdpDetailRecord = new GDPDetailRecord("Australia");
            //publisher.sendGDPDetailRecordToTopic(gdpDetailRecord);
            return gdpDetailRecord;
```

```
        } catch (Exception ex) {
            return new GDPDetailRecord("Failed to process");
        }
    }

}
```

marthinusswart/aws-proof-of-concepts

Created by marthinusswart • Updated 12 hours ago

Various Proof of Concept using the AWS Tech Stack

Github

Kafka Docker Container