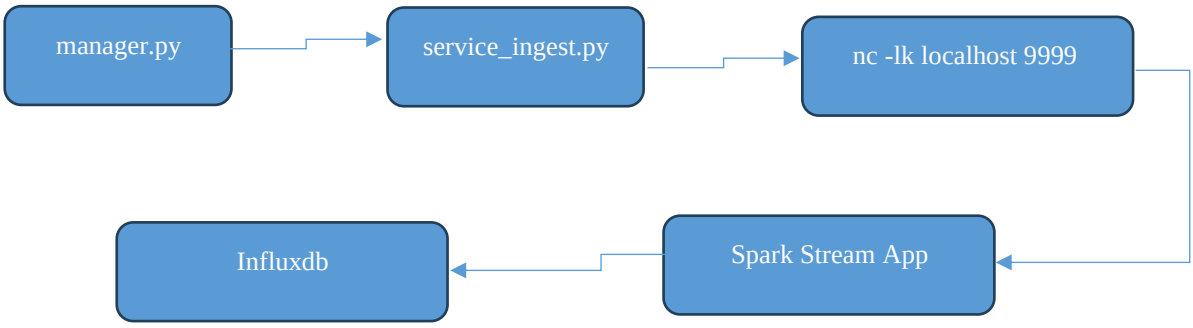
۱. معماری کلی سامانه

جهت پردازش داده‌های جریانی ابتدا از یک کد پایتونی به عنوان واسط استفاده شده است تا اطلاعات ارسالی از طرف manager.sh دریافت و برای کلاستر پردازش جریانی ارسال نماید. جهت پردازش جریانی از Spark استفاده شده است. Spark داده‌های جریانی پردازش نموده و معیارهای EMS، MS و RSI را محاسبه می‌کند و خروجی را به Influxdb جهت بصری‌سازی بلادرنگ ارسال می‌نماید. شماتیک کلی معماری سامانه در شکل زیر آمده است.

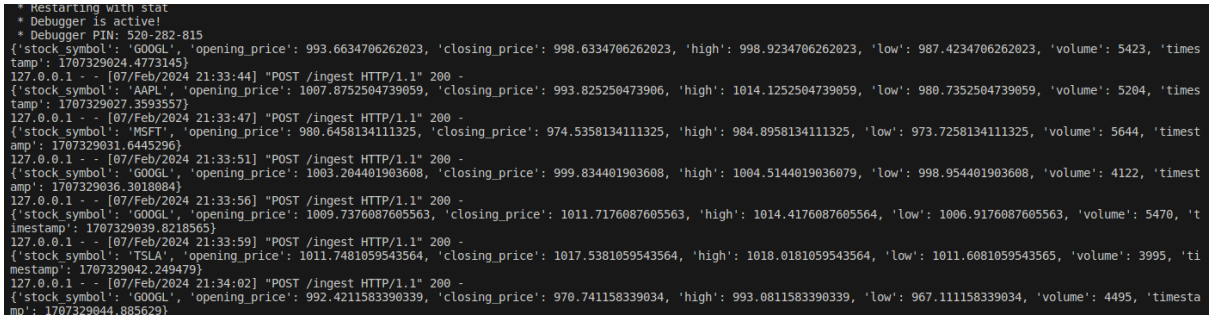


۲. service_ingest.py

همانطور که بیام شد این بخش وظیفه دریافت داده‌های ارسالی از manager.py را بر عهده دارد.

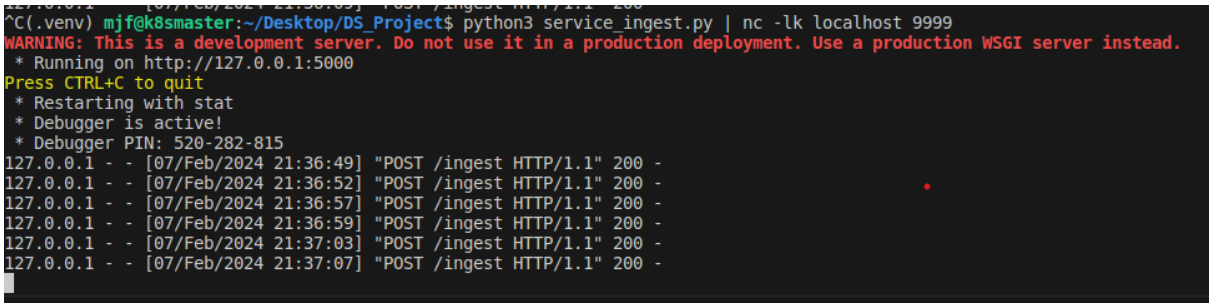
```python
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route('/ingest', methods=['POST'])
def ingest():
    try:
        data = request.json  # Assumes the incoming data is JSON
        print(f"{data}")

        return jsonify(data)
    except Exception as e:
        return jsonify({"status": "error", "message": str(e)}), 500

if __name__ == '__main__':
    app.run(port=5000, debug=True)
```

این برنامه ورودی دریافتی را در قالب json چاپ می‌نماید. نمونه‌ای از خروجی کد بالا در ادامه آمده است.



۳. استفاده از دستور netstat

در ادامه با استفاده از دستور زیر داده‌ها را برای Spark ارسال می‌نماییم. به‌طور پیش‌فرض Spark داده‌ها را از روی پورت 9999 می‌خواند.



۴. برنامه پردازش جریانی داده‌ها مالی

در این مرحله نرم‌افزاری با استفاده از pyspark توسعه داده شده که داده‌ها را به‌طور مداوم از پورت ۹۹۹۹ می‌خواند و برای پردازش به تابع محاسبه کننده معیارها و استخراج سیگنال ارسال می‌نماید. در بخش main برنامه داده‌ها به‌صورت جریانی توسط Spark خوانده می‌شوند و به ازای هر Batch برای تابع update_dict فراخوانی خواهد شد.

```python

if __name__ == "__main__":


    sc = SparkContext("local[2]", "SocketStreamExample")
    # set batch interval to 1 second (minimum value)
    ssc = StreamingContext(sc, 1)

    # Create a DStream that will connect to a socket and receive raw binary data
    samples = ssc.socketTextStream("localhost", 9999)

    #json_data_stream = samples.map(lambda line: json.loads(line))
    #print(json_data_stream)

    # Update dictionary with new data
    # samples.foreachRDD(lambda rdd: rdd.foreach(update_dict))
    samples.foreachRDD(update_dict)

    # Process each sample
    #samples.foreachRDD(lambda rdd: rdd.foreach(process_sample))

    # Start the streaming computation
    ssc.start()

    # Keep the program running
    try:
        # Sleep for a long time to keep the Spark Streaming context running
        # You can interrupt the program when you want to stop it
        ssc.awaitTermination()  # Timeout set to 24 hours (in seconds)
    except KeyboardInterrupt:
        # Stop the Spark Streaming context if interrupted
        ssc.stop()

```

بخش اصلی کد به همراه گزارش ارسال شده است و در اینجا جهت پرهیز از طولانی شدن از آوردن آن صرفنظر می‌شود. در نهایت خروجی برنامه اسپارک در کنسول اطلاعات زیر را چاپ می‌نماید.