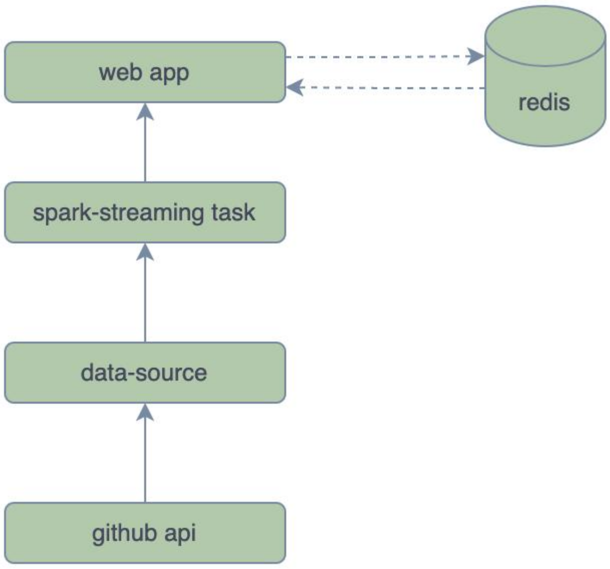

The entire system is implemented using four services and a Redis cache database. The three services are data source, which is responsible for regularly collecting data from github, spark worker, and spark master. They are responsible for running spark stream processing tasks on Spark. After collecting data from data source, they perform various data analysis calculations, and obtain analysis results. The last service is webapp, which provides an interface for data storage and display. After receiving the processing results of spark tasks, Cache to Redis and support visual display.

data processing

1、**Compute the total number**
First, preprocess the data, parse the sent data according to the json format, extract the items field, and do a flat map to obtain that each row of data is a repositoryinformation. Filter out data belonging to three languages based on the language field to avoid dirty data.
Then map the data into a language key with a value of 1, and do a reduceByKey to obtain the number of warehouses in each language. Finally, use updateStateByKey to calculate the total number of warehouses for each language.

2、Compute the average number of stars
The data preprocessing process is the same as the previous step, and then the information of each repositoryis mapped into a language of key and a value of stargazers_count, calculate the groupByKey, and convert the value to a list. Summarize all stargazers in each programming language using updateStateByKey and calculate the count and average. Finally, merge the three records into one rdd through reduce.

3、Find the top 10 most frequent words in the description of all the collected repositories
The data preprocessing process is the same as before. Then, map the information of each repositoryinto language as key and value as description. After replacing the description with non alphabetic characters, divide it according to spaces to obtain ((language, word), 1). After doing reduceByKey and updateStateByKey, obtain the number of times for each word in each language Then, make a reduceByKey based on the language as the key, and take the top10. Finally, merge 30 records into a single rdd through reduce.