chronosphere

Alert Execution Engine - Take Home

Overview

Implement a simple alert execution engine that runs alerts periodically by checking the alerts queries against the provided query API and sends notifications if alerts fire.

Environment

A helper/validator alerts server is available in the docker container provided here. It exposes REST APIs to retrieve alerts, make queries, send notifications and resolve an alert. You will make use of these to build the alert execution engine. In the background it runs a validation loop to ensure the right notification and resolution messages are being sent. Information about any failures are printed out to the alerts' server console output.

1. Ensure you have docker installed on your machine.
2. Start the engine using - `docker run --name alerts_server -p 9001:9001 quay.io/chronosphereiotest/interview-alerts-engine:latest`
    a. You can pass in "-d" while starting the container to see extra debug output from the alerts server that will help seeing traffic and responses to the alerts server

You are free to use either the sample alerts clients (in Python, Go or Java) provided or the HTTP API directly if you prefer to use any other language to solve the problem.

Alerts Client in Go/Python/Java

We have sample alerts' client implemented in Go and Python that wrap the HTTP endpoints exposed on the HTTP server. The sample clients will be provided to you in a zip file by the recruiter along with this exercise.

Alerts HTTP API

We also use cURL to demonstrate the provided APIs in the Appendix.

Requirements

Alert Engine Basics

The exercise requires you to develop an alert execution engine that executes alerts at the specified interval checks. Some engine basics:

1. One alert execution cycle involves:
    a. Making an API call to query the value of the metric
    b. Comparing the return value against thresholds

   c. Determining the state of the alert
   d. Make a notify API call if the alert is transitioning to WARN / CRITICAL state
   e. Make a resolve API call if the alert is transitioning to PASS state
2. Alert can be in different states based on the current value of the query and the thresholds.
   a. It is considered PASS if value <= warn threshold
   b. It is considered WARN if value > warn threshold and value <= critical threshold
   c. It is considered CRITICAL if value > critical threshold

Functionality

Functionality of the alert execution engine are described below

1. Build an alert execution engine
   a. Queries for alerts using the above API and execute them at the specified interval
   b. Alerts will not change over time, so only need to be loaded once at start
   c. The basic alert engine will send notifications whenever it sees a value that exceeds the critical threshold.
   d. The engine should have the ability to execute alerts in parallel with the amount of parallelization being configurable
2. Introduce alert state transitions and repeat intervals
   a. Incorporate using the the warn threshold in the alerting engine - now an alert can go between states PASS <-> WARN <-> CRITICAL <-> PASS
   b. Add support for repeat intervals, so that if an alert is continuously firing in the same state it will only re-fire after the repeat interval

Submission Requirements

- The program should pass alongside the provided validation docker container.
- The code should include sufficient documentation and comments helpful for the reviewer to understand the code.
- Unit tests are not required, but please include any tests that you end up writing.

Validation

The docker container comes with validation code that runs some end to end tests on 5 minute loops and validates the different alert messages received. Any errors in validation will be printed in the docker container logs as errors. These can be accessed using ”`docker logs alerts_engine -f`”.

Submission

You are free to use any language that you find appropriate to solve the exercise using your IDE of choice. Please submit all files along with instructions of how to run the code and a README that talks about any trade-offs made in the code in a zip file once the exercise is complete.

Questions

If you have any questions while solving the exercise contact eng_interviews@chronosphere.io.

Appendix

Alerts HTTP API

QueryAlerts

This allows you to retrieve the current alerts in the system. `intervalSecs` and `repeatIntervalSecs` are second durations.

Description of alerts fields

- Name - is the name of an alert, is unique and can be used as an identifier
- Query - query is the query that the alert is monitoring.
- IntervalSecs - how often the alert should execute.
- RepeatIntervalSecs - how often should messages be re-sent for an alert that is violating thresholds (default 5m).
- Warn/Critical threshold - the threshold above which the respective message should fire.
- Warn/Critical message - the message to be sent corresponding to the threshold being violated

```
shreyas:/Users/shreyas/scripts$  curl http://127.0.0.1:9001/alerts | jq .
[
  {
    "name": "alert-1",
    "query": "test-query-1",
    "intervalSecs": 30,
    "repeatIntervalSecs": 300,
    "warn": {
      "value": 100,
      "message": "alert-1-warning"
    },
    "critical": {
      "value": 200,
      "message": "alert-1-critical"
    }
  },
  {
    "name": "alert-2",
    "query": "test-query-2",
    "intervalSecs": 30,
    "repeatIntervalSecs": 300,
    "warn": {
      "value": 100,
      "message": "alert-2-warning"
    },
    "critical": {
      "value": 200,
      "message": "alert-2-critical"
    }
  }
]
```

Query

For a given target, query returns the value of the for that or an error.

```
shreyas:/Users/shreyas/scripts$  curl
http://127.0.0.1:9001/query?target=test-query-1 | jq .
{
  "value": 179.26082
}
```

Notify

Returns success or failure depending on if the notify succeeded.

```
shreyas:/Users/shreyas/scripts$  curl -X POST http://127.0.0.1:9001/notify -d
'{"alertName": "test-alert", "message": "test message"}'
```

Resolve

Returns success or failure depending on if the resolve succeeded.

```
shreyas:/Users/shreyas/scripts$  curl -X POST http://127.0.0.1:9001/resolve -d
'{"alertName": "test-alert"}'
```