Alarm digest

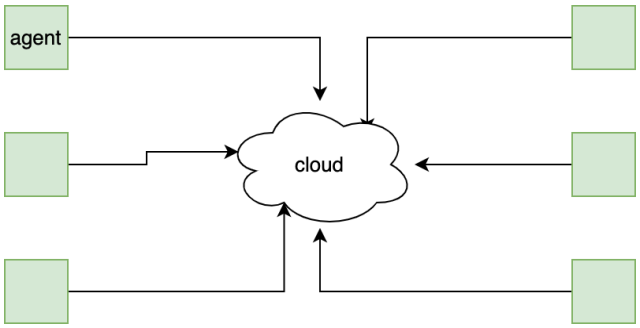
Cloud backend challenge

NETDATA

Intro

Netdata agent is an open-source tool designed to collect real-time metrics, such as CPU usage, disk activity, bandwidth usage, website visits, etc., and then display them in live, easy-to-interpret charts. The tool is designed to visualize activity in the greatest possible detail, allowing the user to obtain an overview of what is happening and what has just happened in their system or application.

Netdata agent periodically checks for anomalies on the aforementioned metrics and based on a set of configurations (both default and custom) reports each detected anomaly in the form of an Alarm. To do so the agent sends the alarms to the cloud backend microservices which are responsible to notify the appropriate user.



An alarm can be in one of the following statuses:

- CLEARED: the alarm is not triggered
- WARNING: first threshold is reached
- CRITICAL: second threshold is reached

We consider an alarm active when its status is either WARNING or CRITICAL. Please consider that an ALARM can go through all the above statuses multiple times during its lifetime, so even after it was CLEARED it could become CRITICAL again.

Deliverables

1. The microservice written in golang
2. A brief documentation outlining your design/implementation decisions
3. A unit and end-to-end test suite for your implementation

Your service

In this assignment we want you to design a new feature for Netdata cloud that allows users to receive notifications for the triggered alarms in a non-intrusive manner.

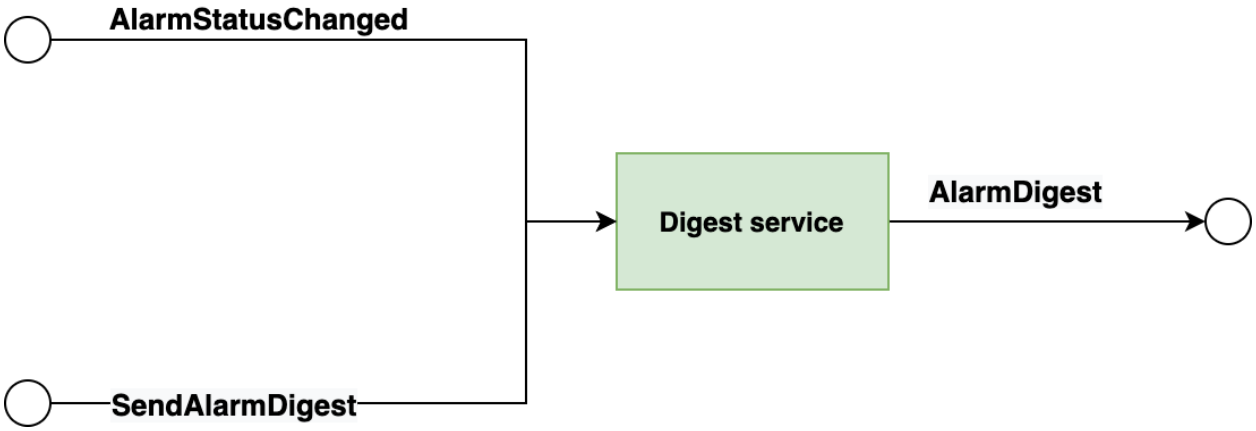
To achieve that you need to create a microservice which will consume events from two topics (as depicted below).

Topic Name	Description
AlarmStatusChanged	This event indicates that an agent has detected a status change on a specific Alarm and a certain user should be notified by this status change.
SendAlarmDigest	This is an internal event triggered by one of the cloud's microservices instructing your microservice to flush all gathered alarms in the form of an AlarmDigest to the appropriate users

You will also need to create a publisher to send the AlarmDigest to the notification service (see image below)

Topic Name	Description
AlarmDigest	This message should contain the latest statuses for each alarm per user (please refer to the "Topics and message schemas" for

	details of the payloads

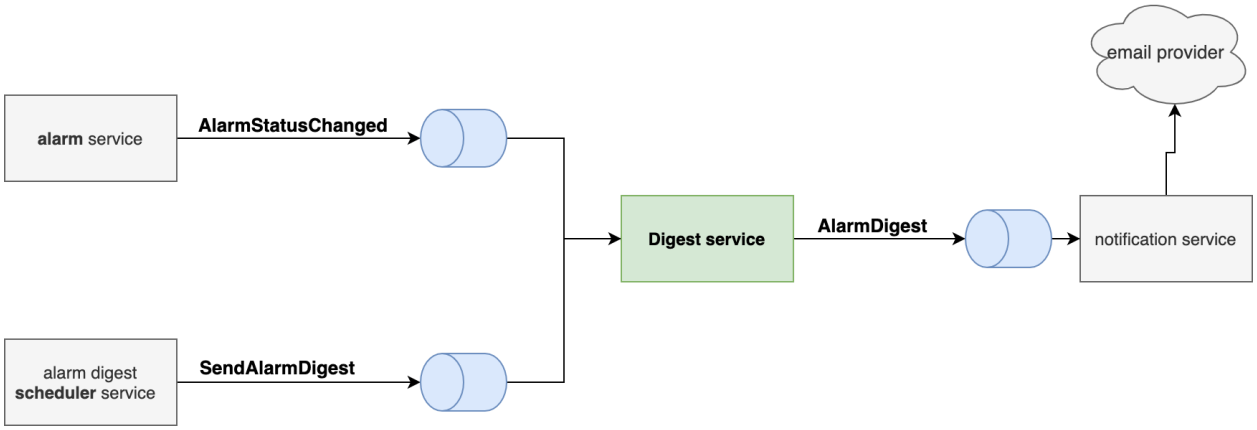


When a *SendAlarmDigest* for a given user is received the service must send an *AlarmDigest* for that user.

The *AlarmDigest* message should include the active alarms as explained in product description.

Notes

1. For the purposes of this challenge you can assume that your service will be part of an event driven system, with multiple microservices that communicate via messaging.
2. In the context of this challenge, nats will act as the message broker.
3. Our broker cannot guarantee in-order delivery of the messages
4. Our broker can guarantee at-least-once delivery for all topics
5. Your solution should be capable of handling any amount of messages
6. Your solution should be able to scale horizontally.
7. It is very important that **all active** alarms are eventually sent to the user, alarms should not be lost.
8. Ideally a user shouldn't receive the same alarm twice, if its status has not changed since the last digest email.
9. Active alarms should be ordered chronologically (oldest to newest)



Getting Started

To help you get things going and get a hint if you are in the right direction, we have prepared a docker image that includes the nats server and a verification tool.

The verification tool runs a very simple tests senerario, publishes some messages and expects to consume an *AlarmDigest* message.

Your service should pass the verification, but it is not enough. You need to write your own tests and you need to consider all the technical and non-technical requirements, along with edge cases.

We suggest you to run the following:

In one terminal:

```
docker run -p 4222:4222 --rm -it \
    --name be-challenge \
    netdata/be-challenge:latest
```

Then run your service in an second terminal, for example:

```
go run ./cmd/service
```

And while your service and the container are both running:

```
docker exec -it be-challenge verify
```

Topics and message schemas

Topic: AlarmStatusChanged

JSON example payload

```
{
    AlarmID: "e36a6c22-ece6-46eb-9016-9303273edbfe",
    UserID: "e859dab9-66d4-4bf4-a578-113d223b94f0",
    Status: "WARNING",
    ChangedAt: "2021-06-07T20:40:15.598765212Z"
}
```

Topic: SendAlarmDigest

JSON example payload

```
{
    UserID: "e859dab9-66d4-4bf4-a578-113d223b94f0",
}
```

Topic: AlarmDigest

JSON example payload

```
{
    UserID: "e859dab9-66d4-4bf4-a578-113d223b94f0",
    ActiveAlarms: [{
        AlarmID: "e36a6c22-ece6-46eb-9016-9303273edbfe",
        Status: "WARNING",
        LatestChangedAt: "2021-06-07T20:40:15.598765212Z"
    }, {
        AlarmID: "29f717c1-70b8-4f42-b2f5-89bf21f560e9",
        Status: "WARNING",
        LatestChangedAt: "2021-06-07T21:23:01.828161292Z"
    }]
}
```

Communication

Please feel free to reach us with any questions regarding this at be@netdata.cloud

How to submit

1. Zip your working directory along with any other related files, like documentation or diagrams.
2. Upload the zip to https://upload.disroot.org/ (set retention of 7 days) or unlisted to a file sharing of your choosing.
3. Share the link with us at `be@netdata.cloud` with the topic “BE challenge <your name>”