Next generation alerting (ngalert) in Grafana 8

Ngalert (Next generation alert) is the next generation of alerting in Grafana 8.

Overview

The ngalert package can be found in `pkg/services/ngalert` and has the following sub-packages:

```
- api
- eval
- logging
- metrics
- models
- notifier
- schedule
- sender
- state
- store
- tests
```

Scheduling and evaluation of alert rules

The scheduling of alert rules happens in the `schedule` package. This package is responsible for managing the evaluation of alert rules including checking for new alert rules and stopping the evaluation of deleted alert rules.

The scheduler runs at a fixed interval, called its heartbeat, in which it does a number of tasks:

1. Fetch the alert rules for all organizations (excluding disabled)
2. Start a goroutine (if this is a new alert rule or the scheduler has just started) to evaluate the alert rule
3. Send an `*evalContext` event to the goroutine for each alert rule if its interval has elapsed
4. Stop the goroutines for all alert rules that have been deleted since the last heartbeat

The function that evaluates each alert rule is called `ruleRoutine`. It waits for an `*evalContext` event (sent each interval seconds elapsed and is configurable per alert rule) and then evaluates the alert rule. To ensure that the scheduler is evaluating the latest version of the alert rule it compares its local version of the alert rule with that in the `*evalContext` event, fetching the latest version of the alert rule from the database if the version numbers mismatch. It then invokes the Evaluator which evaluates any queries, classic conditions or expressions in alert rule and passes the results of this evaluation to the State Manager. An evaluation can return no results in the case of NoData or Error, a single result in the case of classic conditions, or more than one result if the alert rule is multi-dimensional (i.e. one result per label set). In the case of multi-dimensional alert rules the results from an evaluation should never contain more than one per label set.

The State Manager is responsible for determining the current state of the alert rule (normal, pending, firing, etc) by comparing each evaluation result to the previous evaluations of the same label set in the state cache. Given a label set, it updates the state cache with the new current state, the evaluation time of the current evaluation and appends the current evaluation to the slice of previous evaluations. If the alert changes state (i.e. pending to firing) then it also creates an annotation to mark it on the dashboard and panel for this alert rule.

You might have noticed that so far we have avoided using the word "Alert" and instead talked about evaluation results and the current state of an alert rule. The reason for that is at this time in the evaluation of an alert rule the State Manager does not know about alerts, it just knows for each label set the state of an alert rule, the current evaluation and previous evaluations.

Notification of alerts

When an evaluation transitions the state of an alert rule for a given label set from pending to firing or from firing to normal the scheduler creates an alert instance and passes it to Alertmanager. In the case where a label set is transitioning from pending to firing the state of the alert instance is "Firing" and when transitioning from firing to normal the state of the alert instance is "Normal".

Which Alertmanager?

In ngalert it is possible to send alerts to the internal Alertmanager, an external Alertmanager, or both.

The internal Alertmanager is called `MultiOrgAlertmanager` and creates an Alertmanager for each organization in Grafana to preserve isolation between organizations in Grafana. The `MultiOrgAlertmanager` receives alerts from the scheduler and then forwards the alert to the correct Alertmanager for the organization.

When Grafana is configured to send alerts to an external Alertmanager it does so via the sender which creates an abstraction over notification of alerts and discovery of external Alertmanagers in Prometheus. The sender receives alerts via the `SendAlerts` function and then passes them to Prometheus.

How does Alertmanager turn alerts into notifications?

Alertmanager receives alerts via the `PutAlerts` function. Each alert is validated and its annotations and labels are normalized, then the alerts are put in an in-memory structure. The dispatcher iterates over the alerts and matches it to a route in the configuration as explained here.

The alert is then matched to an alert group depending on the configuration in the route. The alert is then sent through a number of stages including silencing and inhibition and at last the receiver which can include wait, de-duplication, retry.

What are notification channels?

Notification channels receive alerts and turn them into notifications and is often the last callback in the receiver after wait, de-duplication and retry.