

Unified Alerting



Luca Gennari
Technical Enablement Manager



Devin Cheevers
Product Manager Grafana Cloud

Agenda

- Unified Alerting Basic Concept
- What's new on Unified Alerting
- Architecture of an Alert
- Troubleshooting Issues
- Demo



Unified Alerting Basic Concepts

Unified Alerting Overview

First of all, why is called Unified Alerting?

When talking of alerting in Grafana, is required to be precise and specify if is about the legacy alerting system before Grafana version 8 or the Unified alerting system released from Grafana version 8.

The alerting logic is almost the same, send an alert by email or other available tools when something happens in the monitored system (*high CPU usage, low memory available, storage almost full, and so on*), what is different is the way this operation is performed and where all alert information can be accessible.

The basic idea behind the unified alerts was to create something consistent across all Grafana products (*Grafana OSS, Grafana Cloud, or Grafana Enterprise*), bringing together the Grafana panel alerts and Prometheus-style alerts.

Basic logic to create an alert is the same (*run a query, get the result and evaluate with an expression*) but now all have in common an identical alert system that behaves exactly the same way from product to product and they can be created where needed (*main alert page or inside a panel configuration*) and managed from a unique place regardless where the alert was created.

Legacy Alerting vs Unified Alerting

Overview

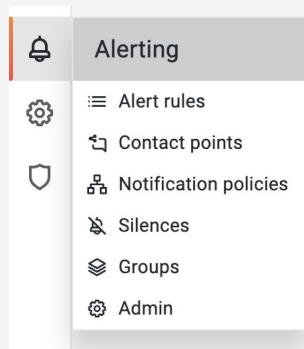
- Unique place to manage and create alerts
- Grafana Alert Manager is totally Prometheus compatible
- If required, external alert managers can be added
- Multi Dimensional Rules
 - multiple alert instances from a single alert rule.
 - expressions now can be created using: Math, Reduce, Resample and Classic
- Manage Cortex and Loki alerts
- Contact Points testing
- Organisation level isolation
- RBAC support (viewer, editor and admin)
- Combine multiple data sources to create alerts

Key Points

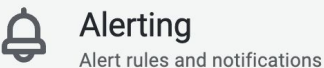
- Unification First
 - consistent throughout all products (OSS, Cloud, Enterprise)
 - new common API backing the new alerting engine
 - new dedicated section for silences and mute alerts
 - Notification Policies and Contact Points are now decoupled
- One page to manage them all
 - powerful interface with much more functionality
 - alerts are no longer exclusively tied to Graph panels
 - easy-to-use experience and less YAML-like textbox editors

Unified Alerting User Interface

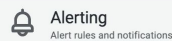
Central Alerting Page - Navigation



From the side menu is possible to select one of the functionalities page.



Alert rules Contact points Notification policies Silences Alert groups Admin



Alert rules Contact points Notification policies Silences Alert groups Admin

Search by data source

All data sources

Search by label

Search

State

Firing

Normal

Pending

Rule type

Alert

Recording

View as

Groups

State

1 rule: 1 normal

New alert rule

Grafana

localTest

1 rule

Cortex / Loki

No rules found.

From any page is possible to navigate between tabs without leaving the alerting page user interface.

Central Alerting Page - Alert Rule



Alerting

Alert rules and notifications

[Alert rules](#) [Contact points](#) [Notification policies](#) [Silences](#) [Alert groups](#) [Admin](#)

Search by data source

All data sources

Search by label

Search

State

Firing

Normal

Pending

Rule type

Alert

Recording

View as

Groups

State

1 rule: 1 normal

+ New alert rule

Grafana

localTest

1 rule | [View](#) [Edit](#)

State

Name

Health

Summary

>

Normal

MyRule

ok

Test on Mac Book Pro

Details of a rule:

- Name
- Health
- Summary
- View
- Edit
- Delete

Find rules using different filters: from data source to label, from type to state.

Create new alert rule. Alert rules can be of type: Grafana, Cortex/Loki rules and Cortex/Loki recording (next slides)

List of existing rules grouped by type.

Grafana

localTest

1 rule | [View](#) [Edit](#)

State

Name

Health

Summary

Normal

MyRule

ok

Test on Mac Book Pro

Silence

Show state history

View

Edit

Delete

Labels

name=luca

Data source

Prometheus

Description

Rule check my local Mac load

Summary

Test on Mac Book Pro

Matching

Instances

Search by label

Search

State

Normal

Alerting

Pending

NoData

Error

State

Labels

Created

Central Alerting Page - Alert Rule Types

1 Rule type

Rule name

MyAmazingRule

Rule type

Choose

- Grafana managed alert**
Classic Grafana alerts based on thresholds.
- Cortex/Loki managed alert**
Alert based on a system or application behavior. Based on Prometheus.
- Cortex/Loki managed recording rule**
Recording rule to pre-compute frequently needed or expensive calculations. Based on Prometheus.

1

Grafana Managed Rule

It is the classic grafana alert and is based on thresholds over a metric (high CPU usage, disk 80%, etc)

2

Cortex/Loki Managed Rule

Rules are based on Prometheus style or Loki LogQL, select this option to configure alerts on an external Cortex or Loki instance

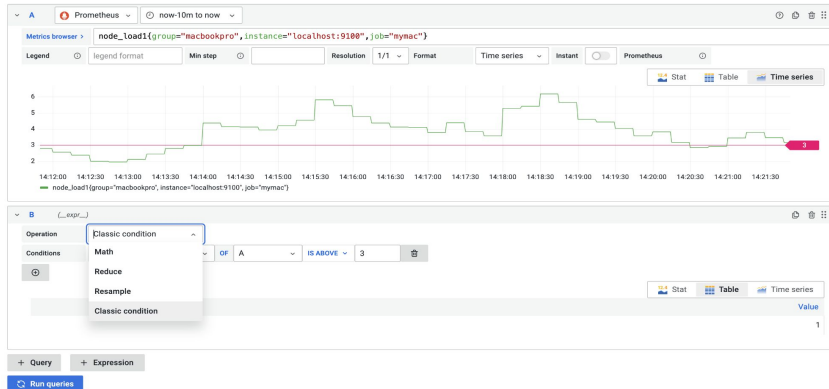
3

Cortex/Loki Recording Rule

If a logic requires an expensive calculations or takes time to be computed, this is the best option. It pre-compute the rule and store the result to Prometheus as a separate time series data to retrieve later.

Central Alerting Page - Queries & Expressions

2 Create a query to be alerted on



1

Step 1: select a datasource

Once a datasource is selected, in the "metric browser" write the query (in this example is a Prometheus PromQL)

2

Step 2: select the operation

Unified alerting let users create an expression based on a classical condition (min, max, avg, etc) as the legacy alerting but now is also possible to use functions as Reduce, Resample and Math.

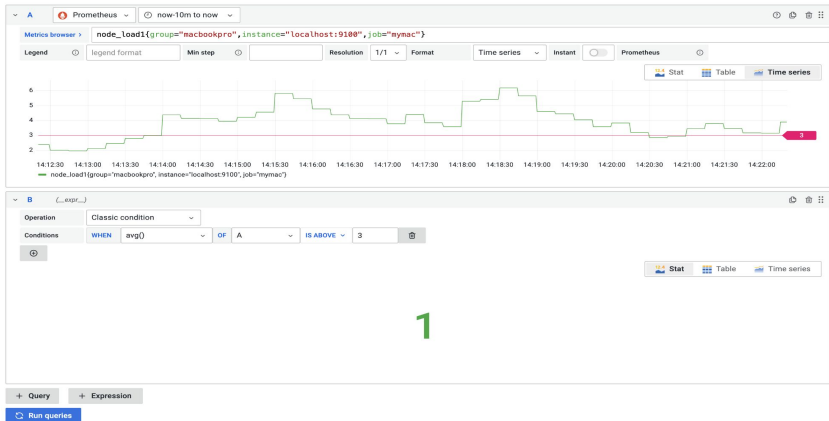
Reduce is a group by function, Resample is used to connect metrics that have different timestamp and Math let you create mathematical operation between queries and expressions results.

3

Step 3: create the alert logic

User interface helps to select components of a condition. Condition can be multiple and linked with an AND/OR logic

2 Create a query to be alerted on



1

Central Alerting Page - Alert Rule Conditions & Exceptions

3 Define alert conditions

Condition

The query or expression that will be alerted on

B

Evaluate

Evaluate every



1m

for



5m

Can be created multiple queries and conditions for an alert, this is called multi-dimensional rule.

They can be linked together to achieve a complex logic but, at the end, one expression or query can be used as source of an alert.

Evaluation Rule

In Grafana managed alert the evaluation has always two components:

- Evaluation Interval
- Evaluation Timeframe

Later in the slides the explanation.

▼ Configure no data and error handling

Alert state if no data or all values are null

No Data

Alert state if execution error or timeout

Alerting

Alerting

OK

Error

By default, in case of missing data, the system fires an alert.

With the new unified alerting is possible to control this behaviour and handle the error in a different way (*example if no data, no error, so no alert*).

Central Alerting Page - Alert Rule Details & Infos

Alert preview is available and let the user know what kind of information will be delivered.

Preview alerts

Preview based on the result of running the query, for this moment. Configuration for 'no data' and 'error handling' is not applied.

State	Info
Alerting	<pre>{ metric='node_load1(group="macbookpro", instance="localhost:9100", job="mymac")'...</pre>

Different information can be added to an Alert

4 Add details for your alert

Write a summary and add labels to help you better manage your alerts

Summary and annotations

Summary

Text that will be shown in the alert page

Description

Any text that has to be shown in the alert. Possible to use templates

Runbook URL

https://

Custom Labels

Labels

key

=

value

Summary

A short text description used in the alert itself

Description

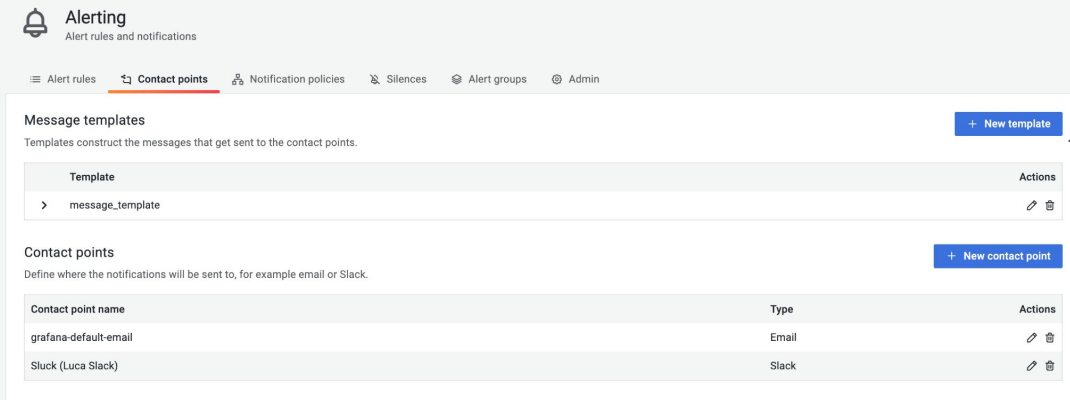
A useful text delivered in case of alert that describe the issue.

Runbook and other infos

Runbook URL is a customisable link to a page (like wiki or documentation) that is immediately accessible from an alert.

In case, is also possible to add other kind of information in form of a textbox or labels in form of key/value pairs.

Central Alerting Page - Contact Point



Alerting
Alert rules and notifications

Alert rules **Contact points** Notification policies Silences Alert groups Admin

Message templates
Templates construct the messages that get sent to the contact points.

+ New template

Template	Actions
> message_template	

Contact points
Define where the notifications will be sent to, for example email or Slack.

+ New contact point

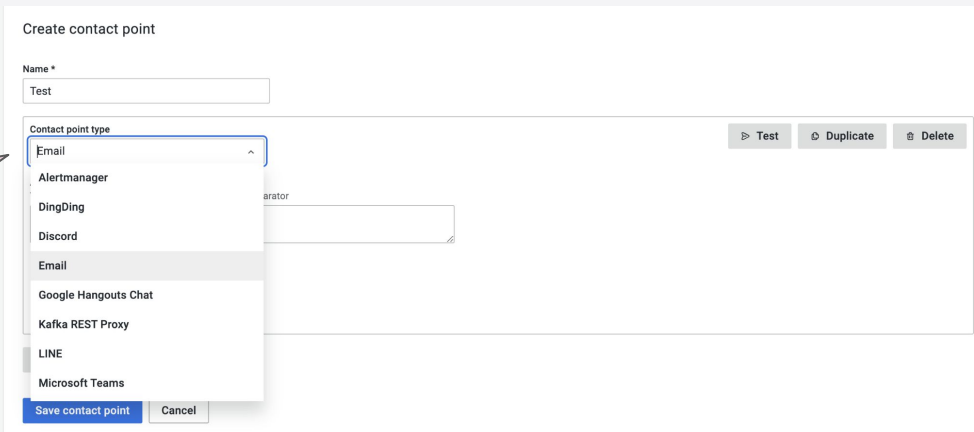
Contact point name	Type	Actions
grafana-default-email	Email	
Slack (Luca Slack)	Slack	

It is also possible to create different templates used by the alert manager to delivery the notification.

Templates are Go templates logic.

Contact points represents the final destination of an alert firing. Is where the alert notification is received and is managed by the Alert manager.

Grafana has different contact points from where to choice, the most used are email, slack, PagerDuty, Kafka.



Create contact point

Name *

Test

Contact point type

Email

Alertmanager

DingDing

Discord

Email

Google Hangouts Chat

Kafka REST Proxy

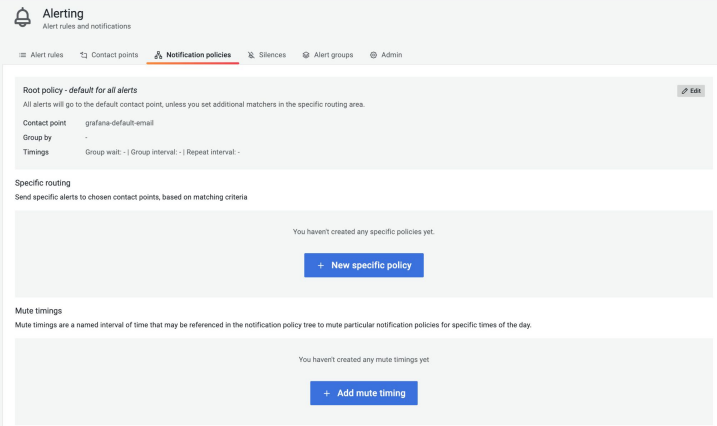
LINE

Microsoft Teams

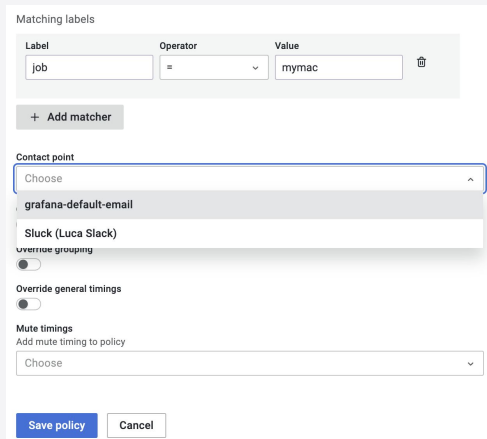
Save contact point Cancel

Test Duplicate Delete

Central Alerting Page - Notification Policies



The screenshot shows the 'Alerting' section of a dashboard, specifically the 'Notification policies' tab. The page has a sidebar with navigation links: 'Alert rules', 'Contact points', 'Notification policies' (active), 'Silences', 'Alert groups', and 'Admin'. The main content area is divided into three sections: 'Root policy - default for all alerts', 'Specific routing', and 'Mute timings'. The 'Root policy' section shows a default contact point 'grafana-default-email' and a 'Timings' section with 'Group wait', 'Group interval', and 'Repeat interval'. The 'Specific routing' section has a message 'You haven't created any specific policies yet.' and a '+ New specific policy' button. The 'Mute timings' section has a message 'You haven't created any mute timings yet.' and an '+ Add mute timing' button.



The screenshot shows the 'Matching labels' form. It has a table with columns 'Label', 'Operator', and 'Value'. The first row has 'job' in the 'Label' column, '=' in the 'Operator' column, and 'mymac' in the 'Value' column. There is a '+ Add matcher' button below the table. Below the table is a 'Contact point' dropdown menu with 'Choose' as the selected option. Below the dropdown is a 'Slack (Luca Slack)' option. There is a 'Override general timings' section with a radio button. Below that is a 'Mute timings' section with a dropdown menu. At the bottom are 'Save policy' and 'Cancel' buttons.

1

Root Policy: default policy

This is the default delivery method used by the alert manager. Can be mail, slack or any other configured contact point.

2

Routing: custom policy

It is possible to create different policies for a notification delivery.

For example when a label match a particular value, instead to use the default delivery method, is possible to routing the message via Slack.

3

Muting: muting timeframes

There are cases in which the user doesn't want to receive an alert, maybe because it's known that parameters are outside a range. In this case is possible to configure that in a specific time, in a particular day, month, year, no notification is needed despite an anomaly is detected.

Central Alerting Page - Silences



Alerting

Alert rules and notifications

Alert rules Contact points Notification policies **Silences** Alert groups Admin

Create silence

Silence start and end

2022-03-22 13:12:40 to 2022-03-22 15:12:40

Duration

2h

Matching labels *

Label

label

Operator

=

Value

value

+ Add matcher

Comment *

created 2022-03-22 13:12

Affected alerts

Add a valid matcher to see affected alerts

Submit

Cancel

Define when a notification should mute by stating the start and end datetimes and refine with customise the duration field.

The logic for when the silence rule should be applied.

Can be one single match or a combination of multiple factors (*by adding other matchers*).

Comment is mandatory and should report why the silence rule is created and what is its purpose.

Central Alerting Page - Admin Tab



Alerting

Alert rules and notifications

Alert rules Contact points Notification policies Silences Alert groups **Admin**

Configuration

```
{
  "template_files": {
    "message_template": "{{ define \"message_template\" }}\n {{ if gt (len .Alerts.Firing) 0 }}\n {{ len .Alerts.Firing }} firing:\n {{ range .Alerts.Firing }} {{ template \"alert\" . }} {{ end }}\n {{ end }}\n {{ if gt (len .Alerts.Resolved) 0 }}\n {{ len .Alerts.Resolved }} resolved:\n {{ range .Alerts.Resolved }} {{ template \"alert\" . }} {{ end }}\n {{ end }}\n{{ end }}"
  },
  "alertmanager_config": {
    "route": {
      "receiver": "grafana-default-email"
    },
    "templates": [
      "message_template"
    ],
    "receivers": [
      {
        "name": "grafana-default-email",
        "grafana_managed_receiver_configs": [
          {
            "uid": "ca0wRBEnk",
            "name": "grafana-default-email",
            "type": "email",
            "disableResolveMessage": false,
            "settings": {
              "addresses": "luca.gennari@grafana.com",
```

Save

Reset configuration

Basic configuration

Basic Configuration is a template that records all changes a user does in all other tabs.

Basically every time a contact point changes or a policy is created and so on, all these actions are stored in this JSON configuration.

External Alertmanagers

You can have your Grafana managed alerts be delivered to one or many external Alertmanager(s) in addition to the internal Alertmanager by specifying their URLs below.

Add Alertmanager		
Url	Status	Action
http://localhost:9093	🟢	✎ 🗑️

External Alertmanager

It is possible to configure and add as many external alertmanager are required.

By default Grafana use its own alertmanager but is possible to add any alertmanager (for example *Prometheus*) and send there the notifications.

Unified Alerting Terminology

Unified Alerting - Terminology

➤ **Alert Rule:**

Alert Rule(s) lets the user define which kind of queries the alerting system has to evaluate periodically to find the anomaly(ies) that trigger the alert(s).

In short, alert rules are the “laws” that govern the alerting system and that are taken into consideration periodically to check and verify that the events exposed by the queries do not occur. If at least one query returns a result, then the alert starts.

➤ **Alert Instance:**

Grafana calls Alert Instance an alert before that is delivered to the Alertmanager and exists locally to the scheduling system.

An alert instance is generated by a set of alert rules and can be modified by changing the rules that create it.

Unified Alerting - Terminology

➤ Alert Evaluations:

Alert evaluations are one of the most critical parts of unified alerting and refer to a single execution of an Alert Rule.

An alert is typically composed of two parts:

- ***Evaluation interval***
- ***Hold duration***

Evaluation interval on which the query(s) runs and is evaluated (for example, every 30 seconds)

Hold duration refers to the timeframe the system waits before firing the alarm.

For example, suppose a timeframe of 2 minutes for holding duration is configured, and the evaluation interval is set to 30 seconds. In that case, the alerting system evaluates the query(s) every 30 seconds for a maximum time of 2 minutes and, if evaluations are positive, the alarm is fired. That means there is four evaluation within the timeframe.

This behavior is because sometimes a metric can slightly change for a brief period and then go back to normal almost immediately. The system is designed to void to spread alarms that can result false from a logistical point of view.

Unified Alerting - Terminology

➤ **Alert:**

When the system fires an alarm, the alert instance that is just locally is delivered to the Alertmanager.

When an alarm is passed to the Alertmanager, the responsibility to deliver the notification is delegated. The alert system notes that an alert was fired and returns to its primary task: evaluating rules.

➤ **Contact Point or Receiver:**

Referring to the previous point, now the alert is in charge of the Alertmanager. Its main task is to spread the notification about the fired alarm through a series of configured receivers.

A receiver is an external system in charge of receiving the notification and delivering it to the last stage: the user(s). A list of the possible receivers, also known as contact points, can be integrated with Grafana. The most used are Slack, PagerDuty, Microsoft Teams, Kafka, etc.

Unified Alerting - Terminology

➤ Notification:

Notification does not refer to the alerting itself but the payload delivered by the Alertmanager to the receiver (*3rd party integration*).

It is usually in JSON or a compatible deliverable structure.

➤ Expressions:

As the name states, this is a series of expressions used by Grafana (v8+) server-side to transform language that allows manipulating data returned from queries with math and other types of operations.

What's Next

Unified Alerting - What's Next on Grafana 9

➤ Legacy Alerting:

- Old legacy alerting system won't be available anymore, a migration to the unified alerting is required.

➤ Recurring Silence:

- With recurring silence will be possible to schedule a mute notification with a series of presetting (*every day, every week or every month*) by a specific time range.
- It is possible to schedule a silence for a particular time for some specific day of the month (*example the 15th and 21st of every month notification will be muted*).
- It is possible to mute a notification by month or year (*every March and November or for the year 2022 notification are on mute*).

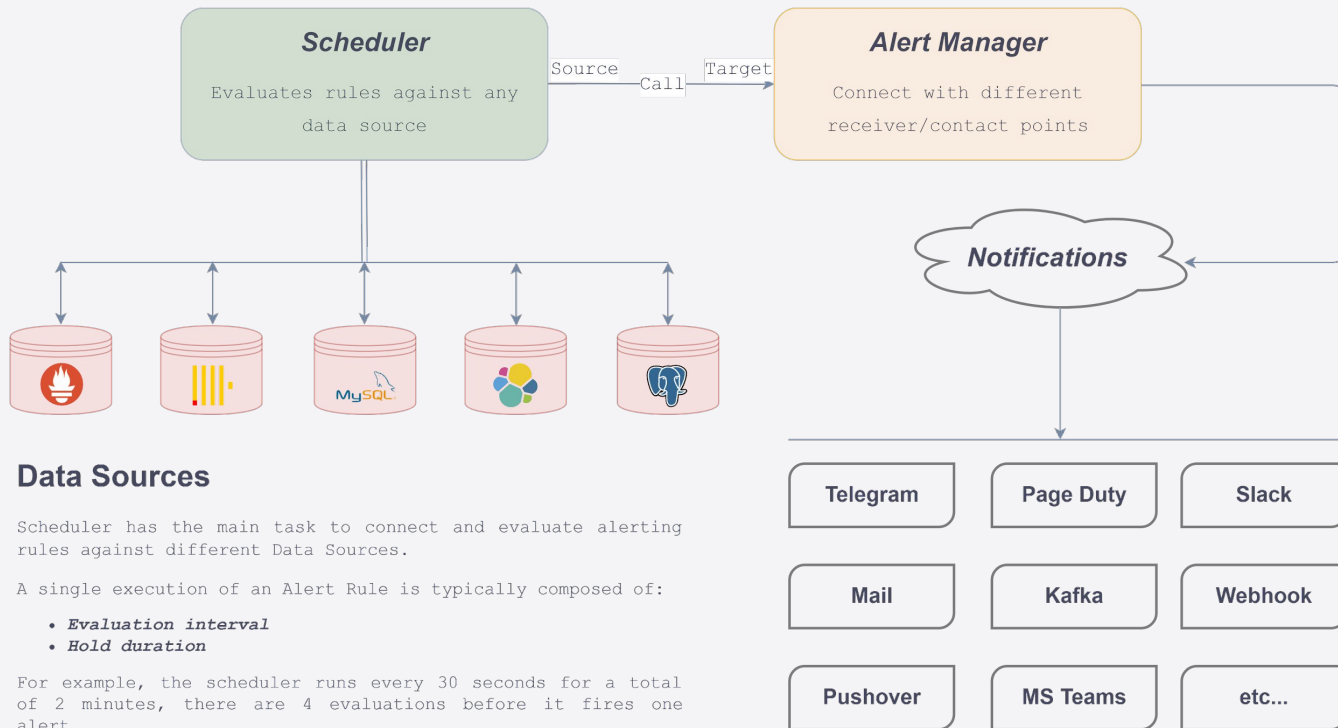
Architecture of an Alert

How Alerts Work

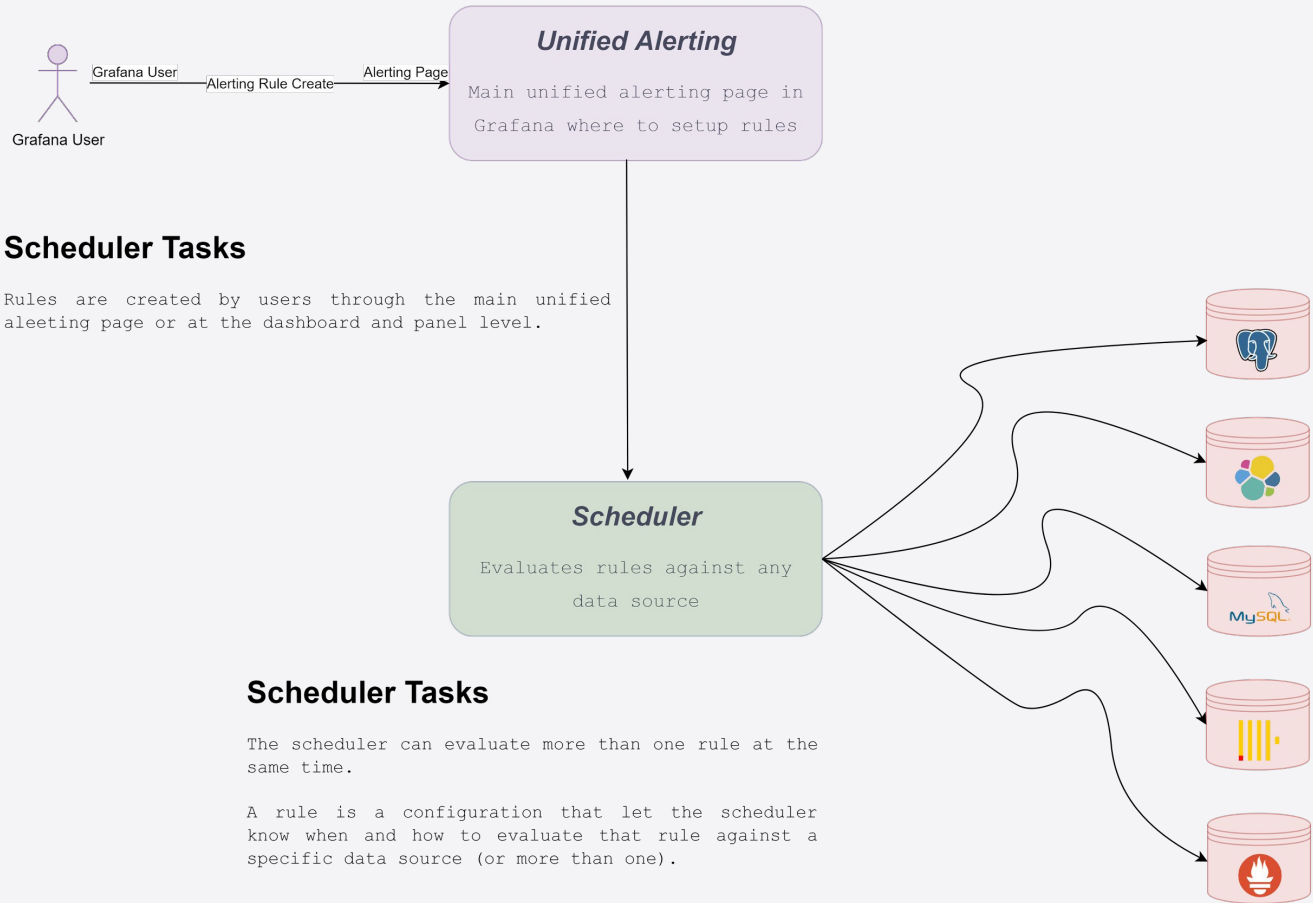
Unified Alerting - Architecture

Alerting Evaluation Flow - Grafana

Diagram describes the process between the Scheduler, its interaction with the Alert Manager, and the final Notification to one of the supported tools (aka receivers).



Unified Alerting - Architecture



Unified Alerting - Architecture

Notifications

The scheduler can notify the internal, default Grafana alert manager or can notify an external alert manager like Prometheus.

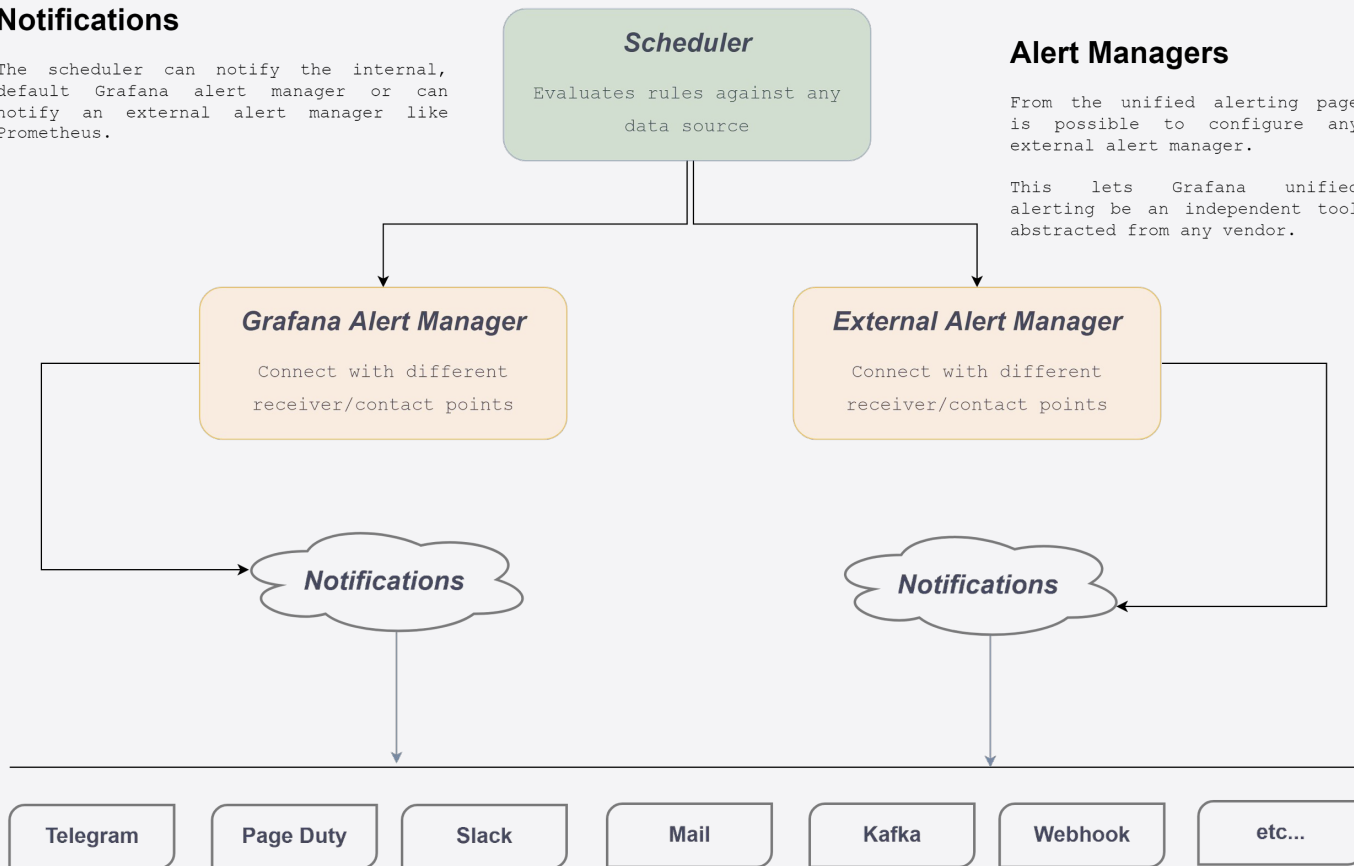
Scheduler

Evaluates rules against any data source

Alert Managers

From the unified alerting page is possible to configure any external alert manager.

This lets Grafana unified alerting be an independent tool abstracted from any vendor.



Types of Rules

Unified Alerting - Rule Types

➤ Grafana Managed Rule:

Grafana managed rule allows the user create alerting rules that query one or more data sources. In the logic of Multi dimensional rules, every query that refers to a single data source is independent by the other but they can be linked together using different expressions.

An expression has the main task to reduce, transform or perform calculation on the query results and compare them to each other or to fix thresholds.

When an expression find a match (for example the avg of the value sent by a query is bigger than a reference number), these are executed and Grafana sends notifications to the contact point.

It is possible to create a complex logic using as many expressions and queries are needed, but they all have to reduce to a single final expression that is the one that is considered by the scheduler.

Unified Alerting - Troubleshooting & Debug

With Grafana managed rule, in most of the cases, the source of an issue can be found in 3 ways:

- One of the query against a data source does not return any result or return unexpected value:
 - verify that the query is correct (*ask customers what they want to achieve and what they expect to see*)
 - verify that all data required to run the query are available in the datasource
 - verify that data are accessible from Grafana (*data source configuration, RBCA, etc*).
- One of the expressions against a query or a series of queries does not return any result or return unexpected value:
 - verify the validity of the expression and that the right query is used in the logic (*and not another one*)
 - verify that your Reduce, Resample or your Math logic are correct (*for example Math uses \$ symbol and the name of the query/expression to get the result*).
 - verify that the right expression / query name that trigger the alert is setup in “**Define Alert Condition**” (step 3, page 15 on this deck: create a new rule phase 3 in the user interface).
- Everything stated here above is correct and the system works fine, but the expected value is never reached (*for example disk usage 90% never reached*).
 - verify with customers if this is the case, sometimes they are sure to monitoring the right resource but in real they are querying another one, it happens when the architecture under monitoring is complex.

Unified Alerting - Troubleshooting & Debug

What if an alert is in status **Firing** but notifications are not sent out:

- Are contact points configured correctly?
 - verify that the contact point that is expected to receive the notification is working: go to **contact points tab** and pick up the contact point under debug, click on the pencil icon (*edit*) and send a test clicking the test button.
 - in case the contact point is the email, check:
 - is the SMTP server configured correctly? (config file under **[smtp]** block)
 - is the mailing address that expects to receive notification in the list?
 - in case the contact point is Slack or any other receiver, check:
 - is the webhook still valid and available
 - Is there an issue to reach the contact point from Grafana installation? (*maybe a firewall or security restrictions*).
- Is there any configuration policy that block the notification to a certain contact point?
 - is the root policy (*the default one*) configured correctly and still valid?
 - verify that the logic of any existing notification policy does not send the notification to a contact point that has been deleted
 - verify that there is not a silence logic in place that is evaluated and block the alert.

Unified Alerting - Rule Types

➤ Cortex/Loki Managed Rule:

Unified Alerts allow the user to create alert rules for an external Cortex or Loki instance. Cortex is a Prometheus instance aggregator while Prometheus itself is a metrics aggregator. Loki is the Grafana tool that allows users to collect logs from different locations and expose them in the Explore session (left menu).

Alerts can be configured based on logs collected by Loki (some anomalies in logs) or based on Prometheus metrics collected by Cortex from different instances in the cluster, that means is possible to keep under observability different instances of Prometheus in one single place.

When an alert belongs to Cortex or Loki, it can be configured and edited directly on the main unified alert page, you can view and edit the rules. The same does not happen if the alert is configured directly in Prometheus, which can be viewed on the alerts page but cannot be deleted or modified, to do this requires a modification in the Prometheus YAML file. This can be an additional step during the debug phase.

Configuring alerts in Cortex or Loki is similar to configuring alerts using Grafana's standard managed rule, the difference is that they are managed by the two tools and Cortex requires PromQL while Loki requires LogQL.

Unified Alerting - Prometheus Rules

YAML file where the rule is defined.
In case of debug the access to that
file is required

▼ /etc/prometheus/rules.yml > workshop 6 rules: 1 firing

State	Name	Health	Summary
▼ Normal	APIHighRequestLatency	ok	High request latency on {{ \$labels.instance }}

See graph View

Expression `api_http_request_latencies_second{quantile="0.5"} > 1` Data source Workshop

Description `{{ $labels.instance }} has a median request latency above 1s (current value: {{ $value }}s)`

Summary High request latency on {{ \$labels.instance }}

Matching instances

Search by label

Search

State

Normal Alerting Pending NoData Error

State Labels Created

From unified alerting
main page the only
operation allowed is
viewing

Unified Alerting - Cortex and Loki Rules

The screenshot shows the Prometheus alerting interface for a Cortex Managed Rule. The rule is named "KubePersistentVolumeFillingUp" and is in a "Normal" state. The expression is a complex PromQL query. The description explains that the rule triggers when a PersistentVolume is claimed in a namespace that is only free. The interface includes buttons for "See graph", "View runbook", "View", "Edit", and "Delete".

State: Normal

Name: KubePersistentVolumeFillingUp

Health: ok

Summary: PersistentVolume is filling up.

Labels: severity=critical

Expression:

```
(kubelet_volume_stats_available_bytes{job="kube-system/kubelet"} / kubelet_volume_stats_capacity_bytes{job="kube-system/kubelet"}) <= 0.01 and kubelet_volume_stats_used_bytes{job="kube-system/kubelet"} > 0 unless on(namespace, persistentvolumeclaim, cluster) kube_persistentvolumeclaim_access_mode{access_mode="ReadOnlyMany"} == 1 unless on(namespace, persistentvolumeclaim, cluster) kube_persistentvolumeclaim_labels{label_excluded_from_alerts="true"} == 1
```

Description: The PersistentVolume claimed by {{ \$labels.persistentvolumeclaim }} in Namespace {{ \$labels.namespace }} is only {{ \$value | humanizePercentage }} free.

Runbook URL: <https://github.com/kubernetes-monitoring/kubernetes-mixin/tree/master/runbook.md#alert-name-kubepersistentvolumeclaimfillingup>

Summary: PersistentVolume is filling up.

Matching Instances: Search by label, Search, State: Normal, Alerting, Pending, NoData, Error

State: Labels Created

Cortex Managed Rule

There are far more operation that can be done with a Cortex managed rule: it is possible to change, view and delete a rule directly from the unified alerting page, it is possible to see the graph and open the documentation resource called Runbook

Loki Managed Rule

As in the Cortex manage rule, same types of operation are available from the alert interface: change, view and delete.

The screenshot shows the Loki alerting interface for a Loki Managed Rule. The rule is named "webServerErrorRateTooHigh" and is in a "Normal" state. The expression is a PromQL query. The description explains that the rule triggers when an application is receiving 404 errors at a rate above 1/sec. The interface includes buttons for "See graph", "View", "Edit", and "Delete".

State: Normal

Name: webServerErrorRateTooHigh

Health: ok

Summary: Application {{ \$labels.application }} 404 rate above 1/sec

Labels: namespace=grafana_cloud, severity=high

Expression:

```
sum by (application, environment) (rate(application="sample_app" | json | res_statusCode=404 | __error__ ~ "*" [!m]) > 0
```

Description: Application {{ \$labels.application }} is receiving 404's at a rate of {{ printf "%.1f" \$value }}%.

Summary: Application {{ \$labels.application }} 404 rate above 1/sec

Matching Instances: Search by label, Search, State: Normal, Alerting, Pending, NoData, Error

State: Labels Created

Unified Alerting - Rule Types

➤ Cortex/Loki Recording Rule:

The user can create and manage registration rules for an external Cortex or Loki instance like a normal managed rule.

The main difference here is that a registration rule calculates frequently needed expressions or computationally expensive expressions in advance and saves the result as a new set of time series.

Querying this new time series is faster, especially for dashboards as they query the same expression every time the dashboards refresh.

This option should be used in case it is really needed and the logic to acquire the metric is complex and requires difficult mathematical calculations

Unified Alerting - Recording Rule

The screenshot shows the Cortex Recording Rule configuration page. At the top, the breadcrumb is 'synthetic_monitoring > default' and there are '4 rules' with an edit icon. Below this is a table with columns: State, Name, Health, and Summary. A single rule is listed with State 'Recording rule', Name 'instance_job_severity:probe_success:mean5m', and Health 'ok'. Below the table, there is a 'See graph' button. To the right of the expression field are 'View', 'Edit', and 'Delete' buttons. The 'Expression' field contains a complex query. The 'Data source' is listed as 'wengelbrecht-prom'.

State	Name	Health	Summary
Recording rule	instance_job_severity:probe_success:mean5m	ok	

[See graph](#) [View](#) [Edit](#) [Delete](#)

Expression

```
(sum without(probe, config_version) (rate(probe_all_success_sum[5m]) *  
on(instance, job, probe) group_left(alert_sensitivity) max by(instance, job,  
probe, alert_sensitivity) (sm_check_info{alert_sensitivity!=""})) / sum  
without(probe, config_version) (rate(probe_all_success_count[5m]) *  
on(instance, job, probe) group_left(alert_sensitivity) max by(instance, job,  
probe, alert_sensitivity) (sm_check_info{alert_sensitivity!=""}))) * 100
```

Data source
wengelbrecht-prom

Cortex Recording Rule

Complex query expression can be hard to compute, as the one in the example here.

This kind of rules should be used for performance reason and to have all data calculated available in case a graph is required

Same access and functionality as a standard managed rule, does not require access to any external configuration files to modify or delete the rule, and can be updated at any time.

Unified Alerting - Troubleshooting & Debug

With Cortex/Loki managed and recording rules, it is required to first understand in which context the issue is:

- Issue is related to Prometheus rules:
 - debug requires the customer send the configuration file (YAML), it is not possible to access from unified alerting page.
- Issue is related to Cortex or Loki rules:
 - verify to have the access in edit mode in order to change the rule
 - verify with the customer which is the goal of the rule and check the validity
 - verify that all data are accessible
- For both cases above, the following steps can help to debug the issue:
 - verify with customers what they are trying to achieve, understanding the meaning of the rule is the first step.
 - verify that data on which the rule is based on are available and accessible.
 - if previous steps are ok (rule is correct and data are accessible) but the notification is not sent:
 - verify is the contact point exists and is correctly configured (*run a test, now it is easy with unified alerting*)
 - verify that there are no notification policies that block the notification:
 - send notification to a contact point wrongly configured or inexistent
 - there is in place a silence logic that mute the notification

Troubleshooting Issues

Debugging Best Practices

Unified Alerting - Troubleshooting & Debug

Basic configuration parameters (default.ini):

- `org_alert_rule = 100` : attribute that limits the number of alerts per Org
- `global_alert_rule = -1` : attribute that limits the number of alert globally (-1 means no limit)
- `level = debug` : under the **[log]** block. Default to info, set to debug to get extra log for alerting

*Note: Instead to set all system to debug, it is possible to enable to debug just a set of loggers: In **[log]** block, the parameter **filter** can contain the follow:*

- `filters = alerting.scheduler:debug \`
 `alerting.engine:debug \`
 `alerting.resultHandler:debug \`
 `alerting.evalHandler:debug \`
 `alerting.evalContext:debug \`
 `alerting.extractor:debug \`
 `alerting.notifier:debug \`
 `alerting.notifier.slack:debug \`
 `alerting.notifier.pagerduty:debug \`
 `alerting.notifier.email:debug \`
 `alerting.notifier.webhook:debug \`

This configuration will exposes inside the log file some value as the follow:

`logger=alerting.scheduler`
etc...

Ask the customer just to enable the one needed

Unified Alerting - Troubleshooting & Debug

Under the block [unified_alerting] :

- **enabled** = true : enable unified alerting and disable the legacy alerting (under **[alerting]** enabled parameter is for legacy. It is opt-in for OSS, so legacy is the default).
- **disabled_orgs** : value is a comma separated list of organizations on which not to send alerts.
- **execute_alerts** = true : enable or disable alerting rule execution keeping visible the UI. The same parameter in legacy alerting block [alerting] takes precedence
- **max_attempts** = 3 : number of times system attempts to evaluate an alert rule before giving up on that evaluation. The failure is on the log *(that need to be set to debug)*
- **listen_address** = "0.0.0.0:9094" : address/hostname and port where to receive unified alerting messages for other Grafana instances *(it consider that other Grafana use the same port, if not need to change accordingly)*.

Unified Alerting - Troubleshooting & Debug

External Alert Managers Issues

It is possible to connect as many external alert managers as needed, if one of these does not work correctly from Grafana point of view (*that means doesn't receive the alert notification, how it manages the received alert is outside the scope*) the following steps can help debug the issue.

- verify that the external alert manager is listed in the **Admin** tab and its status is green (*icon has a heart shape*).
- verify that the external alert manager is configured as Contact Point
- verify that no policies or silence rules block the firing of an alert to the external alert manager.

Resources

Unified Alerting - Resources

Where to go from here on:

- Alerting Documentation:
 - <https://grafana.com/docs/grafana/latest/alerting/unified-alerting/>
 - <https://grafana.com/docs/grafana/latest/alerting/unified-alerting/notifications/>
- Expressions:
 - <https://grafana.com/docs/grafana/next/panels/query-a-data-source/use-expressions-to-manipulate-data/about-expressions/>
- Managed & Recording Rule:
 - Grafana: <https://grafana.com/docs/grafana/latest/alerting/unified-alerting/alerting-rules/create-grafana-managed-rule/>
 - Cortex/Loki: <https://grafana.com/docs/grafana/latest/alerting/unified-alerting/alerting-rules/create-cortex-loki-managed-rule/>
- Loki:
 - Configuration: <https://grafana.com/docs/loki/latest/rules/>
 - LogQL: <https://grafana.com/docs/loki/latest/logql/>
- Code:
 - GitHub: <https://github.com/grafana/grafana/tree/main/pkg/services/alerting>

