Operations

Handling alerts

2/18/2026

Contents

Learn about deploying service alerts.

Related documentation:

-
-

RSS:

- For private edition

Introduction

Alerts notify you when certain metrics exceed specified thresholds. In some services, alerting is enabled by default; in others, you must enable alerting when you deploy the service. See the respective service guides (listed here) for details about service-specific support for alerting.

Alert rules

By default, most services define alerts for certain key operational parameters. The alerts are **PrometheusRule** objects that are defined in a YAML file. The metrics collected from the applicable service are evaluated based on the expression specified in the rule. An alert is triggered if the value of the expression is `true`.

Private edition does not support custom alerts triggered by rules you define yourself. However, some services — for example, Designer — enable you to modify certain parameters in the **values.yaml** file to customize the predefined alerts by modifying the values that trigger the alert. See the respective service-level guides for information about the limited customization each service might support.

Prometheus / Alertmanager

Enable ServiceMonitor or PodMonitor to scrape metrics from the cluster. To import custom alerts or notification configurations, follow these steps.

Alerting Rules

This section describes how to create alert rules and import custom rules.

1. Create alert rules. These rules triggers alerts based on the values.

```
apiVersion: "monitoring.coreos.com/v1"
kind: PrometheusRule
metadata:
  name: -alertrules
  labels:
    genesysengage/monitoring: prometheus
    service:
    servicename:
    tenant:  --> Ex: shared
spec:
  groups:
  - name: -alert
    rules:
    - alert:
      expr:
      for:  For ex: 5m
      labels:
        severity:  For ex: critical
        service:
        servicename:
      annotations:
        summary: ""
```

2. Import the custom rule.

```
kubectl apply -f  -n monitoring
```

Customizing Alertmanager configuration for notifications

Alertmanager sends notifications to the notification provider such as email or Webhook (PagerDuty) when an alert is triggered.

Alertmanager configuration for Notifications

Alertmanager sends notifications to the notification provider (such as email or PagerDuty) when an alert is triggered.

To add notification configuration, edit **alertmanager.yaml** using the following steps:

1. Load the configuration map into a file using the following command.

```
kubectl get configmap prometheus-alertmanager --namespace=monitoring -o yaml >
alertmanager.yaml
```

2. Add the configuration in **alertmanager.yaml**.

```
global:
   resolve_timeout: 5m
route:
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 12h
  receiver: default
  routes:
  - match:
      alertname: Watchdog
    repeat_interval: 5m
```

```
      receiver: watchdog
    - match:
        service:
      routes:
      - match:

        receiver:
receivers:
- name: default
- name: watchdog
- name:
```

3. Save the changes in the file and replace the configuration map.

```
 kubectl replace configmaps prometheus-alertmanager --namespace=monitoring -f
alertmanager.yaml
```

For more details about configuring receivers for alert notification and how the receiver types are created/configured, refer to Configuring alert notifications.

GKE platform

Google Cloud operations suite – Alerting

Google Cloud operations suite is backed by Stackdriver which ingests and processes alerts based on predefined policy configuration.

Stackdriver utilizes Google Cloud Monitoring API for management of metric and alert policies within the operation suite.

Here are some key features provided by Google Cloud operation suite:

- Google Cloud API supports over 1,500 Cloud Monitoring metrics.
- Alert policies are configured as a resource object in cloud monitoring API.
- Unlike Alert Manager, policies are defined directly through GCP Cloud Monitoring API via REST or GRCP request. There are no custom resource objects in Kubernetes for alert polices in GKE.
- Defining alert policies allows you to define specific conditions and actions to take in reaction to key metrics and other criteria.
- Notification channels are used to specify where alerts should be sent when an incident occurs. For example:
  - Webhook
  - Email
  - PagerDuty

For more details, refer to the following Google document pages:

- Introduction to alerting

- Resource: AlertPolicy
- Resource: NotificationChannel
- Resource: UptimeCheckConfig

Google Cloud Monitoring API - Alert Policy

Alert Policy REST API

All API requests to Google Cloud Monitoring API require proper authentication before you query and apply configuration.

See Google authentication for further details.

Here are various functions that are available for creation of custom alert policy.

projects.alertPolicies.create

POST https://monitoring.googleapis.com/v3/{name}/alertPolicies

projects.alertPolicies.delete

DELETE https://monitoring.googleapis.com/v3/{name}

projects.alertPolicies.get

GET https://monitoring.googleapis.com/v3/{name}

projects.alertPolicies.list

GET https://monitoring.googleapis.com/v3/{name}/alertPolicies

projects.alertPolicies.patch

PATCH https://monitoring.googleapis.com/v3/{alertPolicy.name}

Alert Policy example

This example assumes you have created notification channel and uptime check prior to deployment.

AlertPolicy - NGINX Ingress Uptime Check

```
{
  "displayName": "Uptime-Test uptime failure- Ingress",
  "documentation": {
    "content": "Indicates issue with NGINX Ingress availability.  Check ingress-nginx-
controller-* in the 'ingress-nginx' namespace",
    "mimeType": "text/markdown"
  },
  "conditions": [
    {
      "displayName": "Failure of uptime check_id uptime-test",
      "conditionThreshold": {
        "aggregations": [
          {
```

```
            "alignmentPeriod": "1200s",
            "crossSeriesReducer": "REDUCE_COUNT_FALSE",
            "groupByFields": [
              "resource.label.*"
            ],
            "perSeriesAligner": "ALIGN_NEXT_OLDER"
          }
        ],
        "comparison": "COMPARISON_GT",
        "duration": "60s",
        "filter": "metric.type=\"monitoring.googleapis.com/uptime_check/check_passed\" AND
metric.label.check_id=\" pod " AND resource.type=\"k8s_service\"",
        "thresholdValue": 1,
        "trigger": {
          "count": 1
        }
      }
    }
  ],
  "combiner": "OR",
  "enabled": true,
  "notificationChannels": [
    "projects/gcpe0001/notificationChannels/"
  ]
}
```