



Prometheus & Grafana Alerts

mthree Alumni Training

Site Reliability Engineering



Overview

In this module, you will configure alerts for your environment and application through Prometheus and Grafana and visualize your alerts.

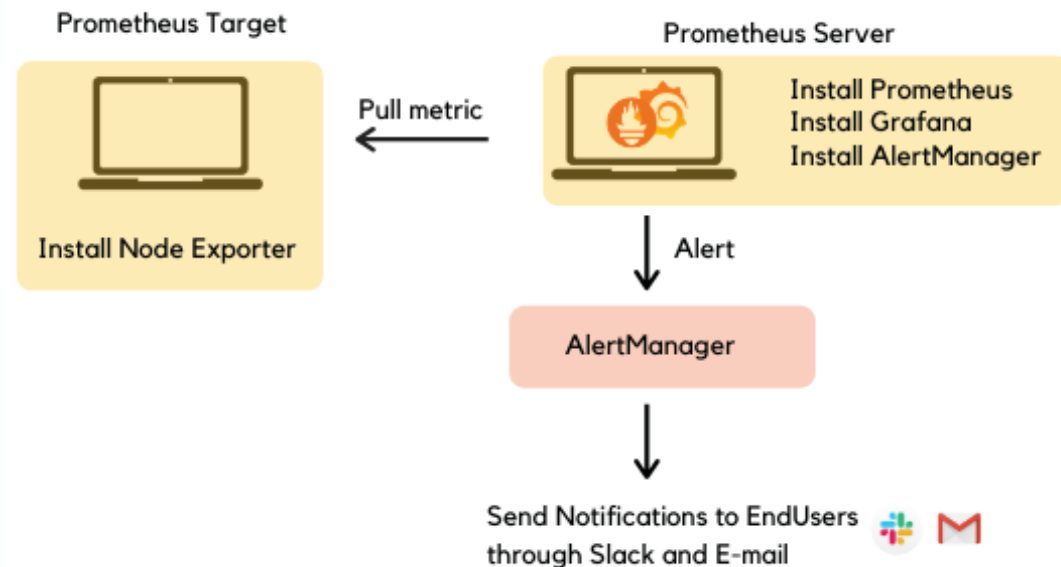
Learning Objectives

- Use Prometheus and AlertManager to create alerts based on your SLOs
- Use Grafana to visualize your SLOs with context
- Create alerts based on SLOs
- Test your alerts before adding to code

Alerts with Prometheus and Grafana

AlertManager handles alerts sent by Prometheus

- ↘ It handles
- >>> Deduplication
 - ~ Reduce toil by sending relevant alerts once
 - >>> Grouping
 - ~ Categorize into single notification
 - >>> Routing to receiver
 - ~ SMS
 - ~ Email
 - ~ PagerDuty
 - ~ OpsGenie



AlertManager

- ↘ Alert rules defined in Prometheus
- ↘ If condition hits during a scrape Prometheus pushes to Alertmanager
- ↘ AlertManager pipeline processes for
 - >>> Silencing
 - ~ Mute for given times otherwise send alert
 - >>> Inhibition
 - ~ Suppress other alerts for those already fired
 - ~ Part of deduplication
 - >>> Grouping
 - ~ Alerts of similar nature grouped preventing multiple notifications firing to receivers

Configuring AlertManager General

- ↘ General configuration
 - >>> Prometheus document: [Alerting Overview](#)
 - >>> [Prometheus Alerting with AlertManager](#)
- ↘ Alert configuration for SRE
 - >>> [Alerting on SLOs like Pros](#)

AlertManager Configuration Lab

- ↘ AlertManager is configured through Kubernetes
- ↘ Only required if you need to send notifications for specific namespaces
- ↘ AlertManagerConfig resource file
 - >>> Examples
 - ~ [RedHat: AlertmanagerConfig](#)
 - ~ [Alerting](#)
- ↘ Created in your project namespace
- ↘ Defines
 - >>> Receivers
 - >>> Routes to send group notifications
 - >>> Matchers to define alert labels to match, e.g. severity=critical

AlertManager Alert Configuration Lab

- ↘ Events and alert types are handled by Flux configuration
- ↘ Mostly for the cluster and NGINX proxy
 - >>> External URL requests come through NGINX
- ↘ Configuring your own alerts
 - >>> Prometheus rules define alerts
 - ~ Your application would need a Prometheus library to obtain specifics
 - ~ Requires scraping from application to obtain metrics
 - >>> Alerts can then be defined
 - >>> Common services should not be placed in application namespaces configs
 - ~ For example, MySQL Database alerts should be for all
- ↘ Flux will find and apply

Predefined alerts

- You can find out of the box alerts for commonly used services at
➤➤ <https://awesome-prometheus-alerts.grep.to>
- Alerts are in YAML format

```
- alert: MysqlTooManyConnections(>80%)  
  expr: avg by (instance) (rate(mysql_global_status_threads_connected[1m])) / avg by (instance) (mysql_global_variables_max_connections) * 100 > 80  
  for: 2m  
  labels:  
    severity: warning  
  annotations:  
    summary: MySQL too many connections (> 80%) (instance {{ $labels.instance }})  
    description: "More than 80% of MySQL connections are in use on {{ $labels.instance }}\n VALUE = {{ $value }}\n LABELS = {{ $labels }}"
```

Alert name

The alert rule

How often, and for what type

The message

The expression

```
- alert: MysqlTooManyConnections(>80%)  
  expr: avg by (instance) (rate(mysql_global_status_threads_connected[1m])) / avg by (instance)  
        (mysql_global_variables_max_connections) * 100 > 80
```

- ↘ Uses aggregates such as **avg**, **sum**, etc
- ↘ Define **rates** at which samples are taken
- ↘ Define calculations
 - >>> Handy for SRE SLOs
- ↘ Use fields from Prometheus,
 - >>> For example: `mysql_global_status_threads_connected`

Converting to Kubernetes

Modify the YAML to fit into Kubernetes

```
- alert: MysqlTooManyConnections(>80%)
  expr: avg by (instance) (rate(mysql_global_status_threads_connected[1m])) / avg by (instance) (mysql_global_variables_max_connections) * 100 > 80
  for: 2m
  labels:
    severity: warning
  annotations:
    summary: MySQL too many connections (> 80%) (instance {{ $labels.instance }})
    description: "More than 80% of MySQL connections are in use on {{ $labels.instance }}\nVALUE = {{ $value }}\nLABELS = {{ $labels }}"
```

```
spec:
  groups:
    - name: alertmanager.rules
      rules:
        - alert: MySQLTooManyConnections
          annotations:
            message: |
              MySQL too many connections (> 80%) `{{ $labels.namespace }}/{{ $labels.service }}`
          expr: avg by (instance) (rate(mysql_global_status_threads_connected[1m])) / avg by (instance) (mysql_global_variables_max_connections) * 100 > 80
          for: 2m
          labels:
            severity: warning
```

MySQL Alerts

- ↘ These would be common to all groups
 - >>> Team talk required
 - >>> Likely to be configured by central team, such as DBAs
- ↘ Requires mysqld-exporter to have provided data to Prometheus
 - >>> e.g., Docker container or on another node
- ↘ We created a file for all MySQL alerts in sre-course-infra
 - >>> AlertManager is managed by Kubernetes and Helm
 - >>> An example rule is located in flux/infrastructure/apps/base/prometheus/mysql-rules.yaml
 - ~ Naming convention based on
 - Flux manages deployment
 - Prometheus is performing the monitoring
 - mysql-rules.yaml to define all alerting and rules we wish to create
- ↘ Alert rules yaml should be placed in the project folder for your Kubernetes namespace
- ↘ Commit, Push and Pull Request merge

MySQL Alert yaml - 1

Change to the service to monitor

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  name: prometheus-kube-mysql-alertmanager.rules
  namespace: prometheus
.....
```

Retain the monitoring namespace

{mthree}

MySQL Alert yaml - 2

Names to look for in AlertManager

```
spec:
  groups:
  - name: alertmanager.rules
    rules:
    - alert: MySQLTooManyConnections
      annotations:
        message: |
          MySQL too many connections (> 80%) `{{ $labels.namespace }}/{{ $labels.service }}`
      expr: avg by (instance) (rate(mysql_global_status_threads_connected[1m])) / avg by (instance)
(mysql_global_variables_max_connections) * 100 > 80
      for: 2m
      labels:
        severity: warning
```

MySQL Alert in AlertManager

➤ Once Flux has updated the system you will see your alert

>>> Go to <https://prometheus.computerlab.online/alerts>

.metadata.name



✓ Inactive (96) ✓ Pending (0) ✓ Firing (1) ✓ Show annotations

/etc/prometheus/rules/prometheus-prometheus-kube-prometheus-prometheus-rulefiles-0/prometheus-prometheus-kube-mysql-alertmanager.rules.yaml > alertmanager.rules Inactive

> MySQLTooManyConnections (0 active)

.spec.groups.rules.alert



Viewing Configuration

- ↘ Your Prometheus server has the following URLs
 - >>> /alerts
 - ~ Shows which alerts are firing
 - >>> /config
 - ~ The Prometheus configuration, including alerting
 - >>> /rules
 - ~ Alert rules
 - >>> /targets
 - ~ Nodes being scraped for metrics
- ↘ Kubernetes stores the rules in **kind: PrometheusRule**

Activity: Working Example of 404 Issues

- We'll create this alert in our sre-course-infra repository
- It will be added to your application directory under **flux**
- Name it *application-dev-alerts.yaml*

Activity: Working Example of 404 Issues - 2

- ↘ An alert when the number of 404 status compared to all status in 1 hour is greater than 50%
- ↘ Naming the rule
 - >>> team?? matching your team number

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  name: prometheus-kube-team??-
  alertmanager.rules
  namespace: team??-dev
.....
```

Activity: Working out the Values

- ↘ Get our metrics
 - >>> <https://prometheus.computerlab.online>
- ↘ Use the expression boxes to find the metrics
 - >>> Let's start with **nginx**
 - >>> You'll see a list appear
 - >>> Type a space and then **request**
 - >>> Select **nginx_ingress_controller_requests**
 - >>> Click **Execute**
- ↘ A lot of data appears
 - >>> Important keys that appear
 - ~ **exported_namespace**
 - ~ **status**

Activity: Working out the Values - 2

↘ Let's filter that data to our namespace

>>> Add the following in the expression box

~ {exported_namespace="sre-example-dev"}

>>> Your expression should now be

~ nginx_ingress_controller_requests{exported_namespace="team??-dev"}

~ This tells us how many of each type of request

>>> Let's get just the 404 status message count

~ nginx_ingress_controller_requests{exported_namespace="team??-dev",status="404"}

- We add the , and then the next key and value to compare

↘ Now we want to know how many in the last 1 hour

>>> For this we need the rate() function

>>> rate(expression[timePeriod])

```
rate/nginx_ingress_controller_requests{exported_namespace="team??-dev",status="404"}[1h]
```

{mthree}

Activity: Working out the Values – 3

- ↘ Click the **Add panel** button
- ↘ Now we want all status requests for our namespace per hour
 - >>> What do you think the expression will be?

Activity: Working out the Values - 4

- Now let's get the 2 together to give us a percentage
- Click **Add panel**
- Let's call our first expression `expr1` and the other `expr2`
`>>> (expr1/expr2)*100 > 50`
- To ensure we are capturing the sum of all status

```
(sum by (all) (rate(nginx_ingress_controller_requests{exported_namespace="team??-dev"}[1h])))
```

- We must sum both sides to ensure single value used in the division

Activity: Working out the Values - 5

Final expression

```
((sum by (all) (rate(nginx_ingress_controller_requests{ exported_namespace="team??-dev",status="404"}[1h])))/(sum by (all) (rate(nginx_ingress_controller_requests{exported_namespace="team??-dev"}[1h]))))*100 > 50
```

Activity: Working Example of 404 Issues – 3

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  name: prometheus-kube-team??-alertmanager.rules
  namespace: team??-dev
spec:
  groups:
    - name: alertmanager.rules
      rules:
        - alert: TooMany404ForTeam??
          annotations:
            message: |
              Too many 404 requests `{{ $labels.namespace }}/{{ $labels.service }}`
            expr: ((sum by (all) (rate(nginx_ingress_controller_requests{ exported_namespace="team??-dev",status="404"}[1h])))/(sum by (all) (rate(nginx_ingress_controller_requests{exported_namespace="team??-dev"}[1h]))))*100 > 50
            for: 2m
          labels:
            severity: warning
```

Activity: Triggering the Alert

- To trigger the alert we need to generate some 404 URLs
- We could run the following shell script

```
while (( x < 50 ))  
do  
    curl sreexampledev.computerlab.online/xyz  
    (( x=x+1 ))  
    sleep 2  
done
```


Pending Alert

<https://prometheus.computerlab.online/alerts>

/etc/prometheus/rules/prometheus-prometheus-kube-prometheus-prometheus-rulefiles-0/sre-example-dev-prometheus-kube-sreexampledev-alertmanager.rules.yaml > alertmanager.rulespending (1)

▼ TooMany404ForSREExampleDev (1 active)

```
name: TooMany404ForSREExampleDev
expr: ((sum by(all) (rate(nginx_ingress_controller_requests{exported_namespace="sre-example-dev",status="404"}[1h]))) / (sum by(all) (rate(nginx_ingress_controller_requests{exported_namespace="sre-example-d
for: 2m
labels:
  severity: critical
annotations:
  message: Too many 404 requests `{{ $labels.namespace }}/{{ $labels.service }}`
```

Labels	State	Active Since	Value
alertname=TooMany404ForSREExampleDev severity=critical	PENDING	2021-05-18T14:29:03.779485606Z	100
Annotations			
message Too many 404 requests `/'			

Fired Alert

<https://prometheus.computerlab.online/alerts>

/etc/prometheus/rules/prometheus-prometheus-kube-prometheus-prometheus-rulefiles-0/sre-example-dev-prometheus-kube-sreexampledev-alertmanager.rules.yaml > alertmanager.rules

firing (1)

▼ TooMany404ForSREExampleDev (1 active)

```
name: TooMany404ForSREExampleDev
expr: ((sum by(all) (rate(nginx_ingress_controller_requests{exported_namespace="sre-example-dev",status="404"}[1h]))) / (sum by(all) (rate(nginx_ingress_controller_requests{exported_namespace="sre-example-d
for: 2m
labels:
  severity: critical
annotations:
  message: Too many 404 requests `{{ $labels.namespace }}/{{ $labels.service }}`
```

Labels	State	Active Since	Value
alertname=TooMany404ForSREExampleDev severity=critical	FIRING	2021-05-18T14:29:03.779485606Z	100
Annotations			
message Too many 404 requests `			

AlertManager Example

Alertmanager Alerts Silences Status Help

New Silence

Filter

Group

Receiver: All ☐ Silenced ☐ Inhibited

+

Silence

Custom matcher, e.g. `env="production"`

+ Expand all groups

+ Not grouped 1 alert

- Not grouped 1 alert

14:23:17, 2021-05-18 (UTC) [+ Info](#) [📈 Source](#) [🔕 Silence](#)

alername="TooMany404ForSREExampleDev" +

prometheus="prometheus/prometheus-kube-prometheus-prometheus" +

severity="critical" +

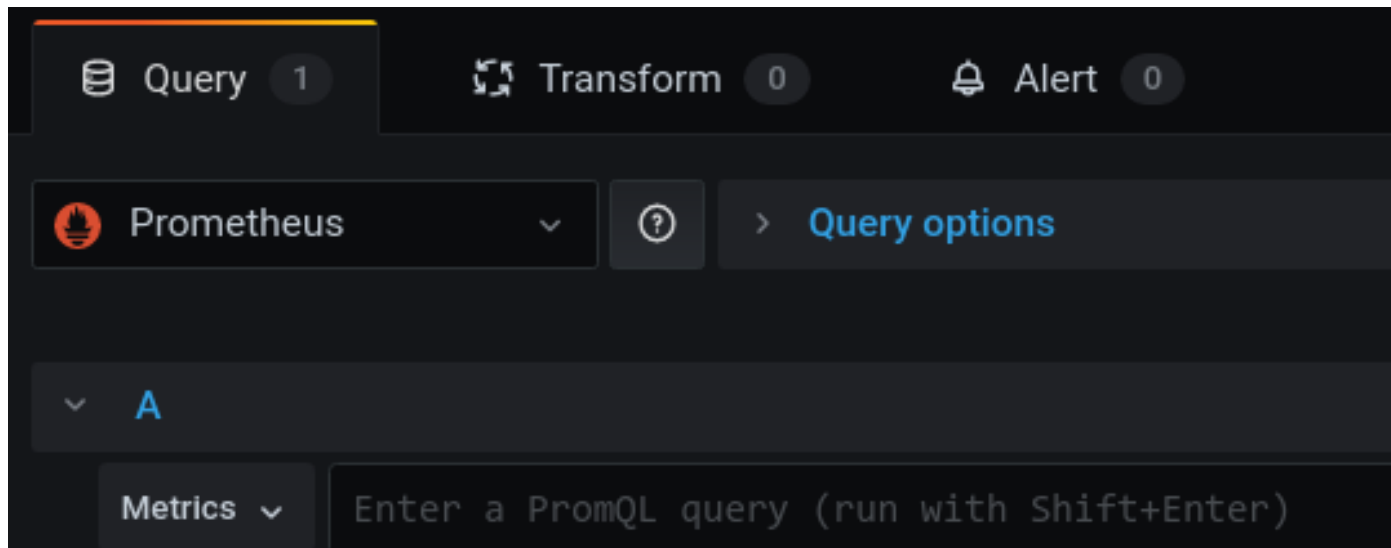
Activity: Grafana Alert

- ↘ Let's use the same SRE http error rate
 - >>> A useful help page
 - ~ [Grafana Alerting](#)
- ↘ Go to <https://grafana.computerlab.online>
- ↘ Go to your dashboard
- ↘ Click the Add panel icon
- ↘ Click the Add new panel button

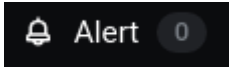


Activity: Grafana Alert - 2

- ↘ Copy and paste your expression into the text box below the graph
 - >>> Enter a PromQL query (run with Shift+Enter)
 - ~ Place it where you see the above text



Activity: Grafana Alert - 3

- ↘ Name your panel
 - >>> 404 Requests per hour
- ↘ Make sure it is a Graph visualization
- ↘ Scroll down to Display on the right of the screen
 - >>> Turn on Staircase
 - ~ This will show all graduation on the Y-axis
- ↘ Under the graph select Alert 
- ↘ Click Create Alert button
 - >>> Here you can set the durations
 - >>> Set the IS ABOVE box to 50
 - >>> Note the slider with the heart to the right of the graph
 - ~ This can be slid up and down to change the value
- ↘ Click Apply button top left
 - >>> Arrange your dashboard and save

References

- ↘ Custom alerting in Kubernetes with Prometheus
 - >>> [Custom Alerts Using Prometheus Queries](#)
- ↘ Out of the box alerts
 - >>> [Awesome Prometheus alerts](#)
- ↘ Alert rule examples
 - >>> [Prometheus alerts examples](#)
 - >>> [Prometheus Example Queries](#)
- ↘ Prometheus alert rule functions
 - >>> [Prometheus: Functions](#)
- ↘ Creating alerts in Grafana
 - >>> [Grafana Alerting](#)



Summary Q&A

Additional References

- Awesome Prometheus Alerts (n.d.). rule-mysql-1-2. Retrieved from <https://awesome-prometheus-alerts.grep.to/rules#rule-mysql-1-2>
- Sylia CH. (2020, May 25). Prometheus Alerting with AlertManager. Retrieved from <https://medium.com/devops-dudes/prometheus-alerting-with-alertmanager-e1bbba8e6a8e>