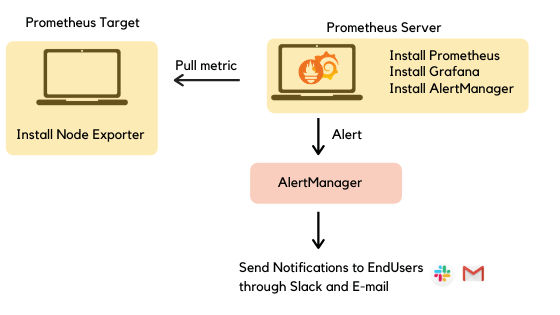
**ALERTMANAGER:**

The Alertmanager handles alerts sent by client applications such as the Prometheus server. It takes care of deduplicating, grouping, and routing them to the correct receiver integration such as Email, PagerDuty, or Slack. It also takes care of silencing and inhibition of alerts.

**ARCHITECTURE OF ALERTMANAGER WITH PROMETHEUS:**



Alert rules are defined in Prometheus configuration. Prometheus just scrapes (pull) metrics from its client application (the Node Exporter). However, if any alert condition hits, Prometheus pushes it to the AlertManager which manages the alerts through its pipeline of **silencing, inhibition, grouping and sending out notifications**.

**SILENCING:**

Silencing is to mute alerts for a given time. Alerts are checked to match against active silent alerts, if a match is found then no notifications are sent.

**INHIBITION:**

Inhibitionis to suppress notifications for certain alerts if other alerts are already fired.

**GROUPING:**

Grouping group alerts of similar nature into a single notification. This helps prevent firing multiple notifications simultaneously to the receivers like Mail or Slack.

**Setting up alerts with Prometheus is a two-step process:**

First, you need to create your alerting rules in Prometheus, and specify under what conditions you want to be alerted (such as when an instance is down).

Second, you need to set up Alertmanager, which receives the alerts specified in Prometheus. Alertmanager will then be able to do a variety of things, including grouping alerts of similar nature into a single notification; silencing alerts for a specific time; muting notifications for certain alerts if other specified alerts are already firing; and picking which receivers receive a particular alert.

**Step One: Create Alerting Rules in Prometheus:**

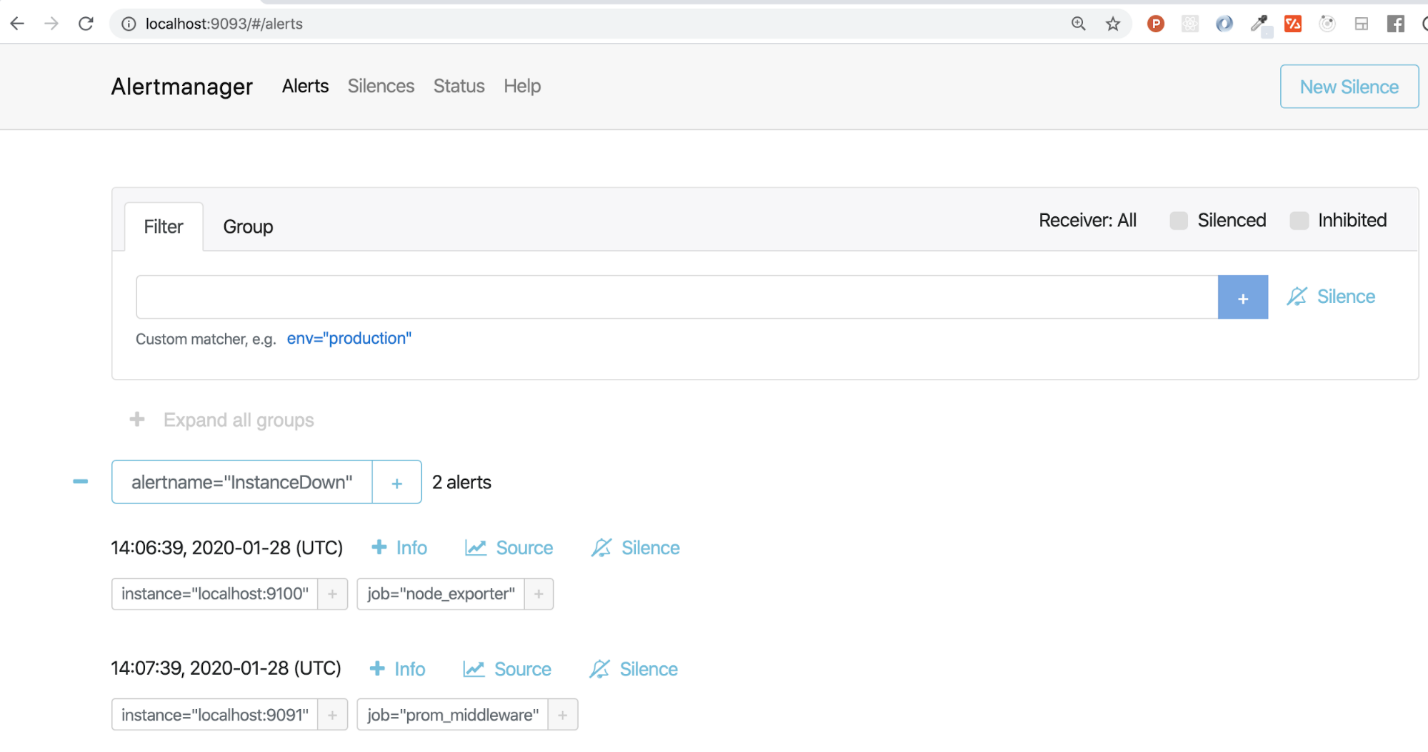
The below is the sample configuration file for the alert rules specified in

Prometheus named as alert-rules.yaml

|  |
| --- |
| groups: |
|  | - name: alert.rules |
|  | rules: |
|  | - alert: InstanceDown |
|  | expr: up == 0 |
|  | for: 1m |
|  | labels: |
|  | severity: "critical" |
|  | annotations: |
|  | summary: "Endpoint {{ $labels.instance }} down" |
|  | description: "{{ $labels.instance }} of job {{ $labels.job }} has been down for more than 1 minutes." |
|  |  |
|  | - alert: HostOutOfMemory |
|  | expr: node\_memory\_MemAvailable / node\_memory\_MemTotal \* 100 < 25 |
|  | for: 5m |
|  | labels: |
|  | severity: warning |
|  | annotations: |
|  | summary: "Host out of memory (instance {{ $labels.instance }})" |
|  | description: "Node memory is filling up (< 25% left)\n VALUE = {{ $value }}\n LABELS: {{ $labels }}" |
|  |  |
|  |  |
|  | - alert: HostOutOfDiskSpace |
|  | expr: (node\_filesystem\_avail{mountpoint="/"} \* 100) / node\_filesystem\_size{mountpoint="/"} < 50 |
|  | for: 1s |
|  | labels: |
|  | severity: warning |
|  | annotations: |
|  | summary: "Host out of disk space (instance {{ $labels.instance }})" |
|  | description: "Disk is almost full (< 50% left)\n VALUE = {{ $value }}\n LABELS: {{ $labels }}" |
|  |  |
|  |  |
|  | - alert: HostHighCpuLoad |
|  | expr: (sum by (instance) (irate(node\_cpu{job="node\_exporter\_metrics",mode="idle"}[5m]))) > 80 |
|  | for: 5m |
|  | labels: |
|  | severity: warning |
|  | annotations: |
|  | summary: "Host high CPU load (instance {{ $labels.instance }})" |
|  | description: "CPU load is > 80%\n VALUE = {{ $value }}\n LABELS: {{ $labels }}" |

## Step Two: Set up Alertmanager:

## Depending on whether or not you have any active alerts, Alertmanager should be properly set up and look something like the image below. To see the annotations that you added in the step above, you’d click on the ****+**Info** button.



**SETUP GMAIL ALERTS:**

If we prefer to have our notifications come directly through an email service, the setup is even easier. It isn’t recommended that you use your personal password for this, so you should create an [App Password](https://support.google.com/accounts/answer/185833?hl=en). To do that, go to **Account Settings -> Security -> Signing in to Google -> App password** (if you don’t see App password as an option, you probably haven’t set up 2-Step Verification and will need to do that first). Copy the newly-created password.

The Configuration file of alertmanager with email is as shown below:

#alertmanager.yaml:

global:

resolve\_timeout: 1m

route:

group\_by: [Alertname]

group\_wait: 10s

group\_interval: 10s

repeat\_interval: 1h

# Send all notifications to me.

receiver: email-me

receivers:

- name: email-me

email\_configs:

- to: receiver@gmail.com

from: sender@gmail.com

smarthost: smtp.gmail.com:587

auth\_username: "sender@gmail.com"

auth\_identity: "sender@gmail.com"

auth\_password: "xxxxxxxxxxxxxxxx"

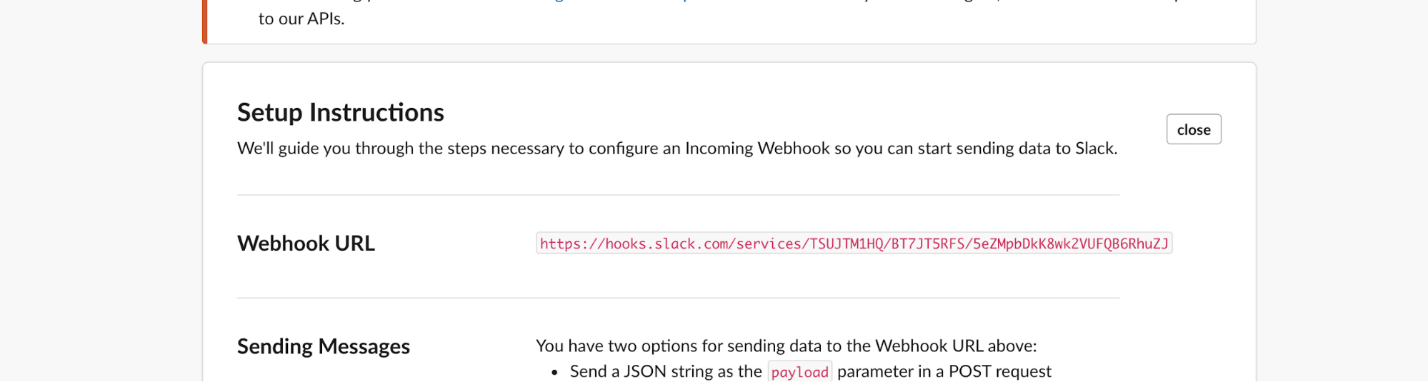
Once again, after a couple of minutes (if any alert rule is triggered), alert notifications should be sent to your Gmail.

**SET UP STACK ALERTS:**

If we want to receive notifications via Slack, you should be part of a Slack workspace. To set up alerting in your Slack workspace, you’re going to need a **Slack API URL.** Go to **Slack -> Administration -> Manage apps.**

In the Manage apps directory, search for **Incoming WebHooks** and add it to your Slack workspace.

Next, specify in which channel you’d like to receive notifications from Alertmanager. (I’ve created #monitoring-infrastructure channel.) After you confirm and add Incoming WebHooks integration, webhook URL (which is your Slack API URL) is displayed. Copy it.



Then you need to modify the **alertmanager.yml** file as follows:

global:

resolve\_timeout: 1m

route:

group\_by: ['alertname']

group\_wait: 10s

group\_interval: 10s

repeat\_interval: 1h

receiver: 'slack-notifications'

receivers:

- name: 'slack-notifications'

slack\_configs:

- api\_url: "https://hooks.slack.com/services/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"

channel: '#monitoring-instances'

send\_resolved: true

icon\_url: https://avatars3.githubusercontent.com/u/3380462

title: |-

[{{ .Status | toUpper }}{{ if eq .Status "firing" }}:{{ .Alerts.Firing | len }}{{ end }}] {{ .CommonLabels.alertname }} for {{ .CommonLabels.job }}

{{- if gt (len .CommonLabels) (len .GroupLabels) -}}

{{" "}}(

{{- with .CommonLabels.Remove .GroupLabels.Names }}

{{- range $index, $label := .SortedPairs -}}

{{ if $index }}, {{ end }}

{{- $label.Name }}="{{ $label.Value -}}"

{{- end }}

{{- end -}}

)

{{- end }}

text: >-

{{ with index .Alerts 0 -}}

:chart\_with\_upwards\_trend: \*<{{ .GeneratorURL }}|Graph>\*

{{- if .Annotations.runbook }} :notebook: \*<{{ .Annotations.runbook }}|Runbook>\*{{ end }}

{{ end }}

\*Alert details\*:

{{ range .Alerts -}}

\*Alert:\* {{ .Annotations.title }}{{ if .Labels.severity }} - `{{ .Labels.severity }}`{{ end }}

\*Description:\* {{ .Annotations.description }}

\*Details:\*

{{ range .Labels.SortedPairs }} • \*{{ .Name }}:\* `{{ .Value }}`

{{ end }}

{{ end }}

inhibit\_rules:

- source\_match:

severity: 'critical'

target\_match:

severity: 'warning'

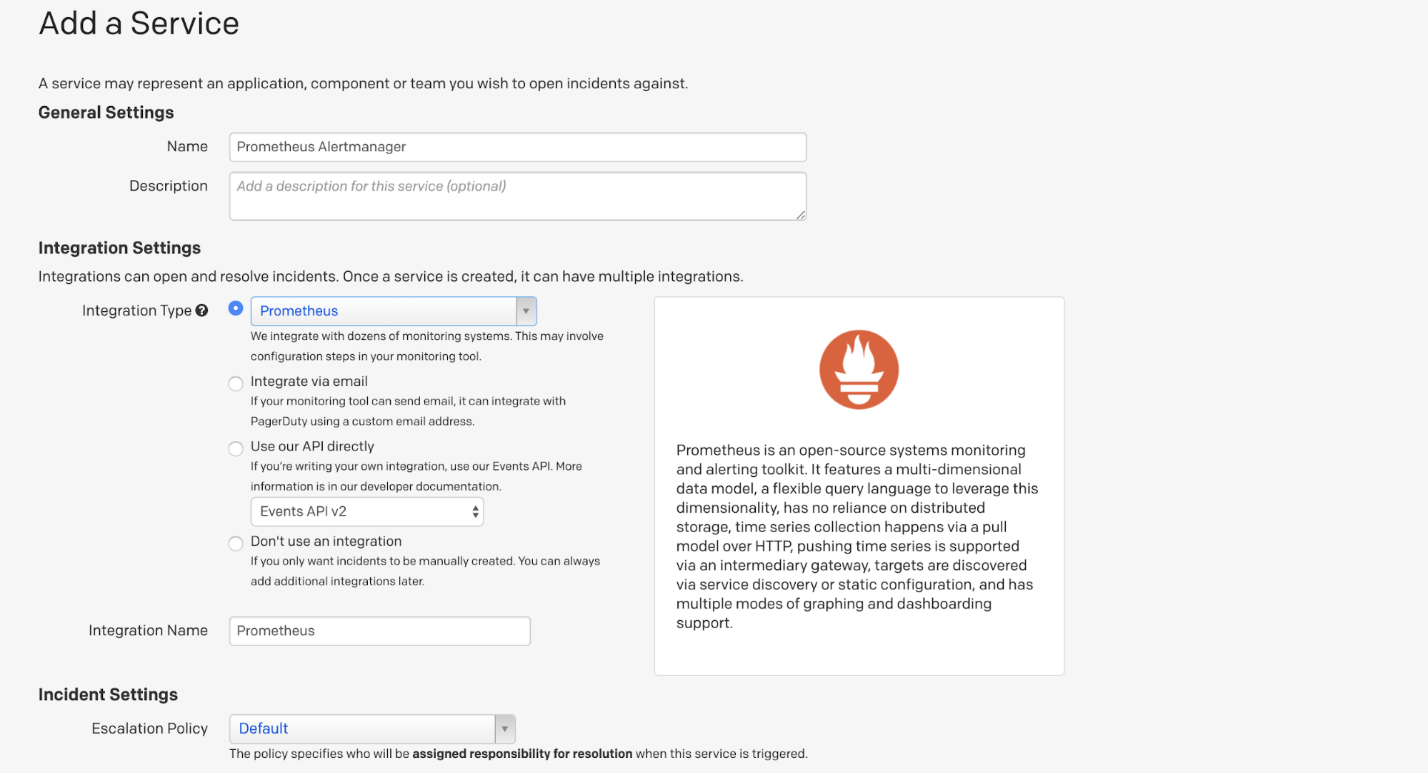
equal: ['alertname', 'dev', 'instance']

Once again, after a couple of minutes (if any alert rule is triggered), alert notifications should be sent to your Slack.

**SETUP PAGERDUTY ALERTS:**

PagerDuty is one of the most well-known incident response platforms for IT departments. Once you’re logged in, go to **Configuration -> Services -> + New Service.**

Choose Prometheus from the Integration types list and give the service a name – I decided to call mine Prometheus Alertmanager. (You can also customize the incident settings, but I went with the default setup.) Then click save.



The Integration Key will be displayed in view integrations. Copy the key.

we’ll need to update the content of our **alertmanager.yml.**

#pagerduty-alertmanager.yaml

global:

resolve\_timeout: 1m

pagerduty\_url: https://events.pagerduty.com/v2/enqueue

route:

group\_by: [cluster]

# If an alert isn't caught by a route, send it to the pager.

receiver: pagerduty-notifications

routes:

# Send severity=page alerts to the pager.

- match:

severity: page

receiver: pagerduty-notifications

receivers:

- name: pagerduty-notifications

pagerduty\_configs:

- service\_key: 0e6e358a4dfa4396868cc817d79c1d81

send\_resolved: true

In PagerDuty user settings, you can decide how you’d like to be notified: email and/or phone call. I chose both and they each worked successfully.

*Reference links:* https://grafana.com/blog/2020/02/25/step-by-step-guide-to-setting-up-prometheus-alertmanager-with-slack-pagerduty-and-gmail/