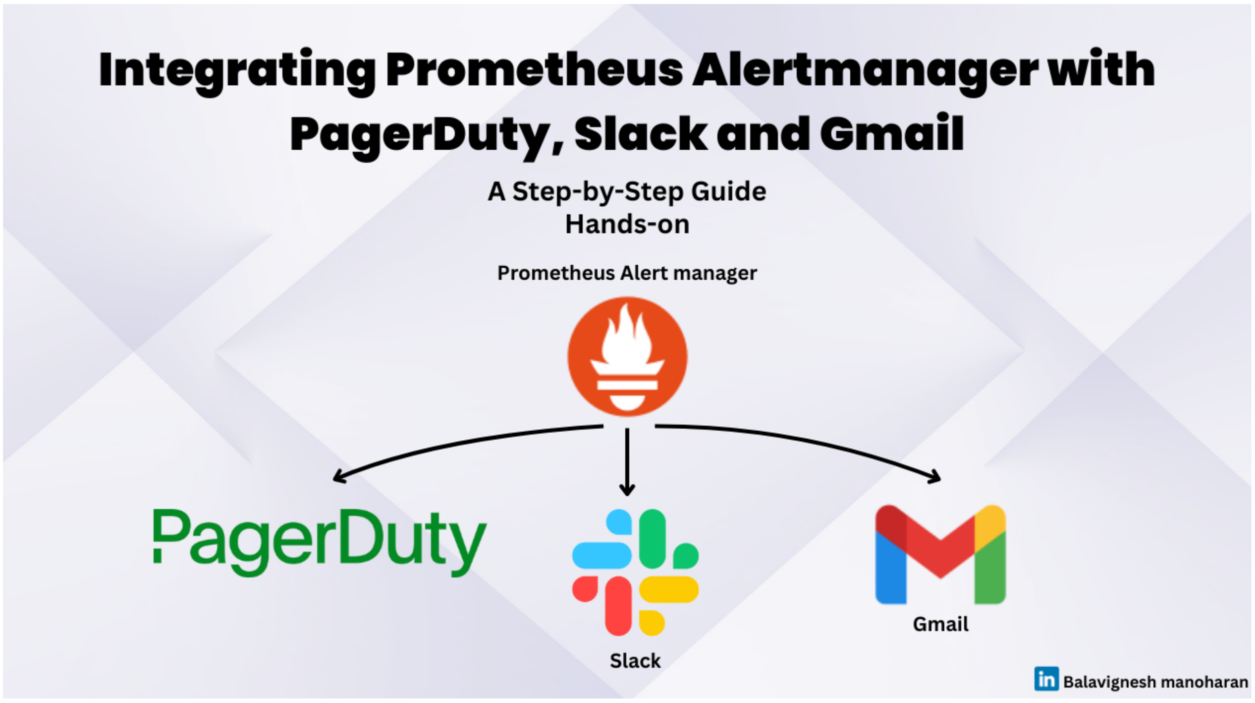
Integrating PagerDuty, Slack and Gmail with Prometheus Alertmanager - Step-by-Step Guide



Introduction: In this guide, I have listed down detailed steps to set up Prometheus with PagerDuty, Gmail and Slack for Alert notification. There are basically two main steps or processes. First, you need to create an Alerting rules file in Prometheus and need to specify under what condition you should be alerted. Second, you need to set up an alert manager which receives the alert specified in the Prometheus. Then the alert manager will manage these alerts by either grouping or sending single notification.

Pre-requisites:

- AWS Account

Requirements:

You will need two servers: The first server will have your application and Node exporter and the second server will have the Prometheus and Alert manager.

Security Ports:

Open the below ports in your inbound rules.

1. Prometheus: Port 9090
2. Node Exporter: Port 9100
3. Alert Manager: Port 9093

Installation

Install & Configure Prometheus, Alert Manager & Node Exporter

- Go to AWS Console and launch two EC2 instances of type t3.medium and 20GB of Storage. Rest of the configuration you can keep it as default.
- One of the server will have the application to monitor and node exporter (Application Server) and the other server will have Prometheus, Alert manager installed in it. (Monitoring Server)
- Once the instances is launched, connect to the monitoring server and update the packages.

```
sudo apt update
```

- We will install Prometheus and Alert manager in this server, go to the official page and get the latest download link of Linux.
- In the terminal download the packages using wget command.

```
wget https://github.com/prometheus/prometheus/releases/download/
```

- Extract the file using tar

```
tar -xvf prometheus-2.53.2.linux-amd64.tar.gz
```

- Remove the tar file, and rename the file.

```
rm prometheus-2.53.2.linux-amd64.tar.gz
mv prometheus-2.53.2.linux-amd64/ prometheus
```

- Go inside the Prometheus folder and run the executable file.

```
cd prometheus
./prometheus &
```

- In the same documentation, look for Alert manager and copy the link address and download and and extract using the same process. Come back to the root location and run the below command.

```
wget https://github.com/prometheus/alertmanager/releases/downloa
```

- Extract the file using tar

```
tar -xvf alertmanager-0.27.0.linux-amd64.tar.gz
```

- Remove the tar file, and rename the file.

```
rm alertmanager-0.27.0.linux-amd64.tar.gz
mv alertmanager-0.27.0.linux-amd64/ alertmanager
```

- Connect to the Application server and download the Node exporter in this server.

```
sudo apt update
wget https://github.com/prometheus/node_exporter/releases/downlo
```

- Extract the file using tar

```
tar -xvf node_exporter-1.8.2.linux-amd64.tar.gz
```

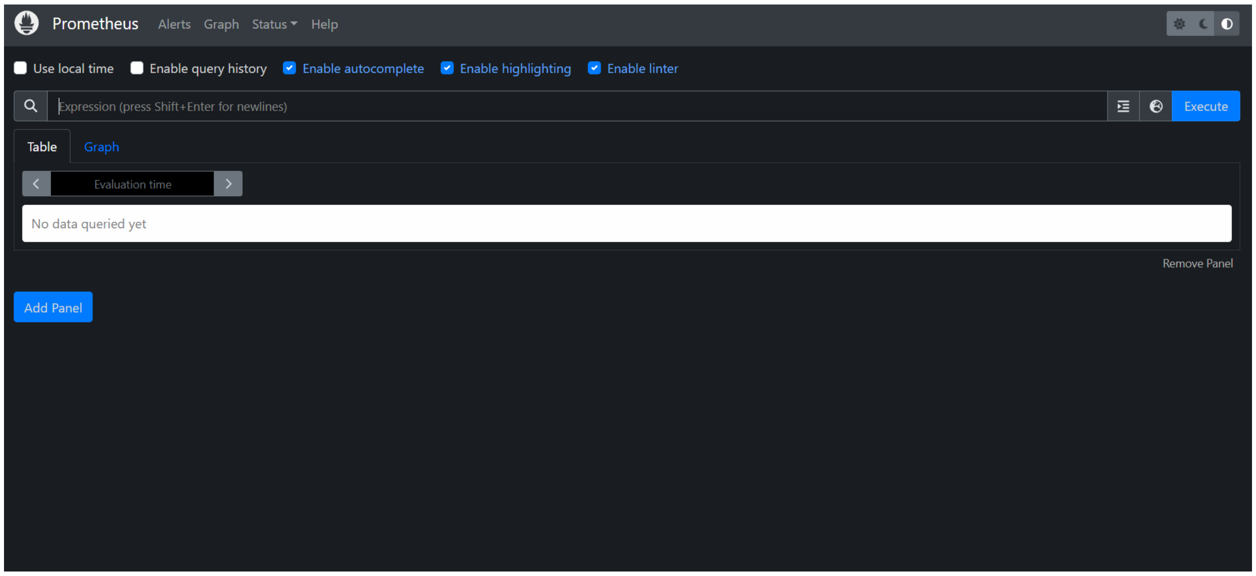
- Remove the tar file, and rename the file.

```
rm node_exporter-1.8.2.linux-amd64.tar.gz
mv node_exporter-1.8.2.linux-amd64/ node_exporter
```

- Go to the Node exporter folder and you can find a file which is executable.

```
cd node_exporter
./node_exporter &
```

- The `&` will run the service in background, and now access the node exporter using the Public IP of your instance with port `9100`. Open the port `9100` in the security group.
- Let us try to access the Prometheus server from Port 9090.

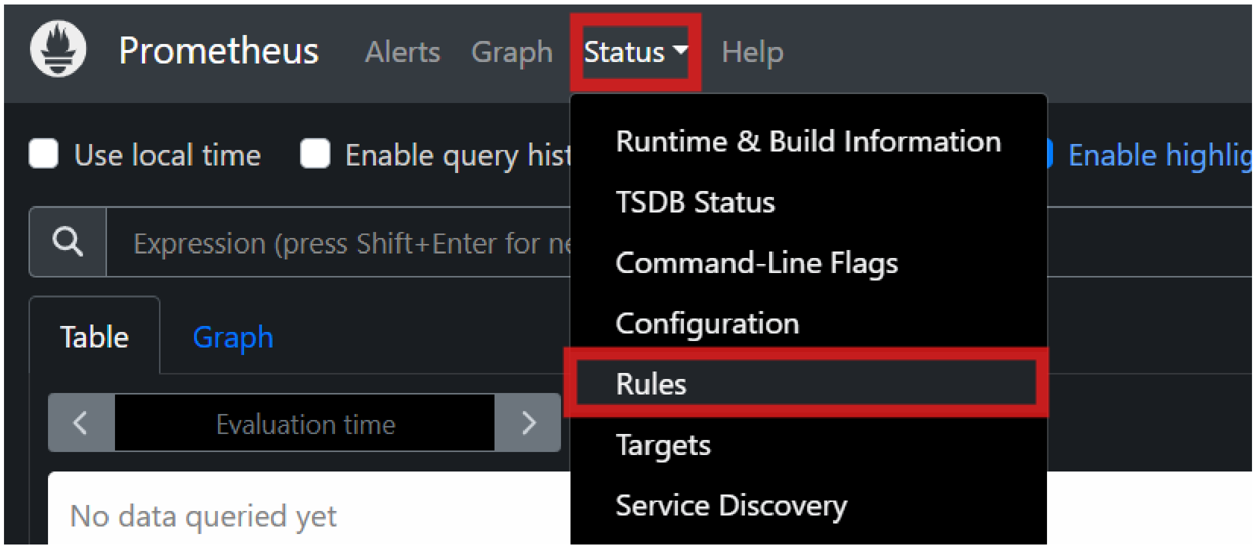


Configuration

Configuring Application and Prometheus Rules

- Go to the Application server and clone your application in this server which you want to monitor.
- Start your application. (In my case, I have a simple Nodejs Application running)

- Copy the IP address of the Monitoring server, open in a new browser with port 9090 (Prometheus port) click on the Status button → Rules. You will observe there isn't any rules configured. Let us setup rules in the next step.



- Go to Prometheus folder and create a new alert_rules.yaml file. Paste the below code into your file.

```
groups:
  - name: AllInstances
    rules:
      - alert: ServiceUnavailable
        expr: up{job="node_exporter"} == 0
        for: 2m
        labels:
          severity: critical
        annotations:
          summary: "Service Unavailable (instance {{ $labels.ins
          description: |-
            The service {{ $labels.job }} is not available.
            VALUE = {{ $value }}
            LABELS: {{ $labels }}
```

Explanation: If the instance is going to be down (up == 0) for 1 minute, then the alert will be firing. In the next alert, we are setting it for a service availability. In this case, I have mentioned node_exporter as the service, you can mention any other service also. Rest we have given the labels of the instance and description for understanding.

> We can have multiple alert rules configured, but in this case we have used only one rule for the service down alert. As in this article we are focusing more on setting up alerts in multi channels.

- Save the above file.
- Now open the prometheus.yaml file and update the name of the alert file

```
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 sec
  evaluation_interval: 15s # Evaluate rules every 15 seconds. De
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
            - 43.204.111.208:9093

# Load rules once and periodically evaluate them according to th
rule_files:
  - alert_rules.yaml
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scra
# Here it's Prometheus itself.
```

```
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any tir
  - job_name: "prometheus"
    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.
    static_configs:
      - targets: ["localhost:9090"]

  - job_name: "node_exporter"
    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.
    static_configs:
      - targets:
          - 13.233.255.2:9100
```

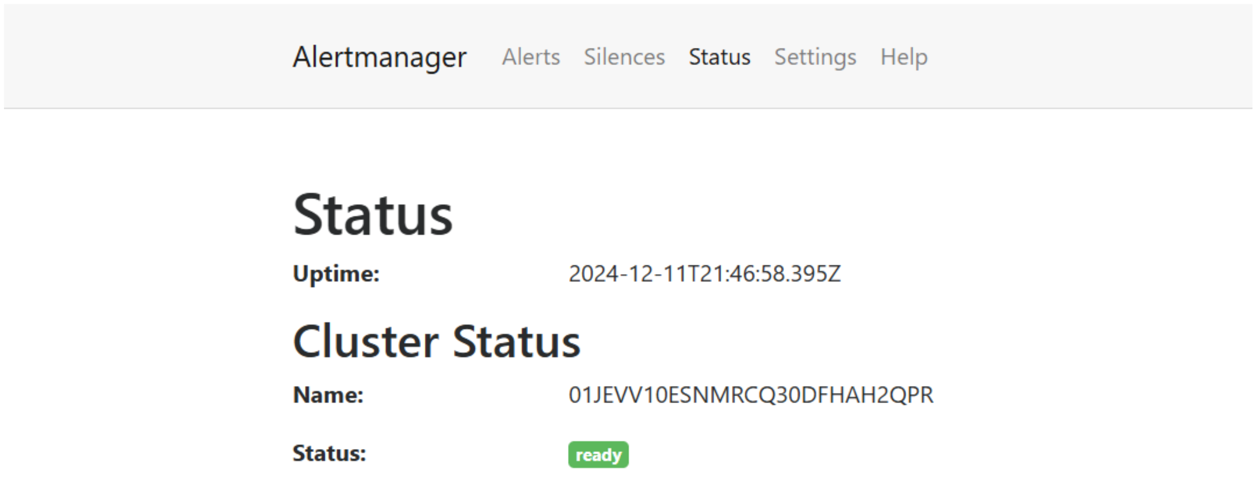
- Replace the Alert manager IP and Node exporter IP address according to your server.
- Restart the Prometheus server once again.

```
pgrep prometheus
#Get the PID number and kill the process
kill PID
#Again start the prometheus server
./prometheus &
```

- Run the Alert manager by navigating through the alert manager folder and running the executable file.

```
./alertmanager &
```

- Open Alert manager by copying IP address with port 9093. Depending on the Alert manager configured you will see the alerts in this page. We will be configuring the alerts in the further steps.



Status

Uptime: 2024-12-11T21:46:58.395Z

Cluster Status

Name: 01JEVV10ESNMRCQ30DFHAH2QPR

Status: ready

- Now we have set up Prometheus with Alert manager, next we will look and set up different ways from where we can send out alert notifications.

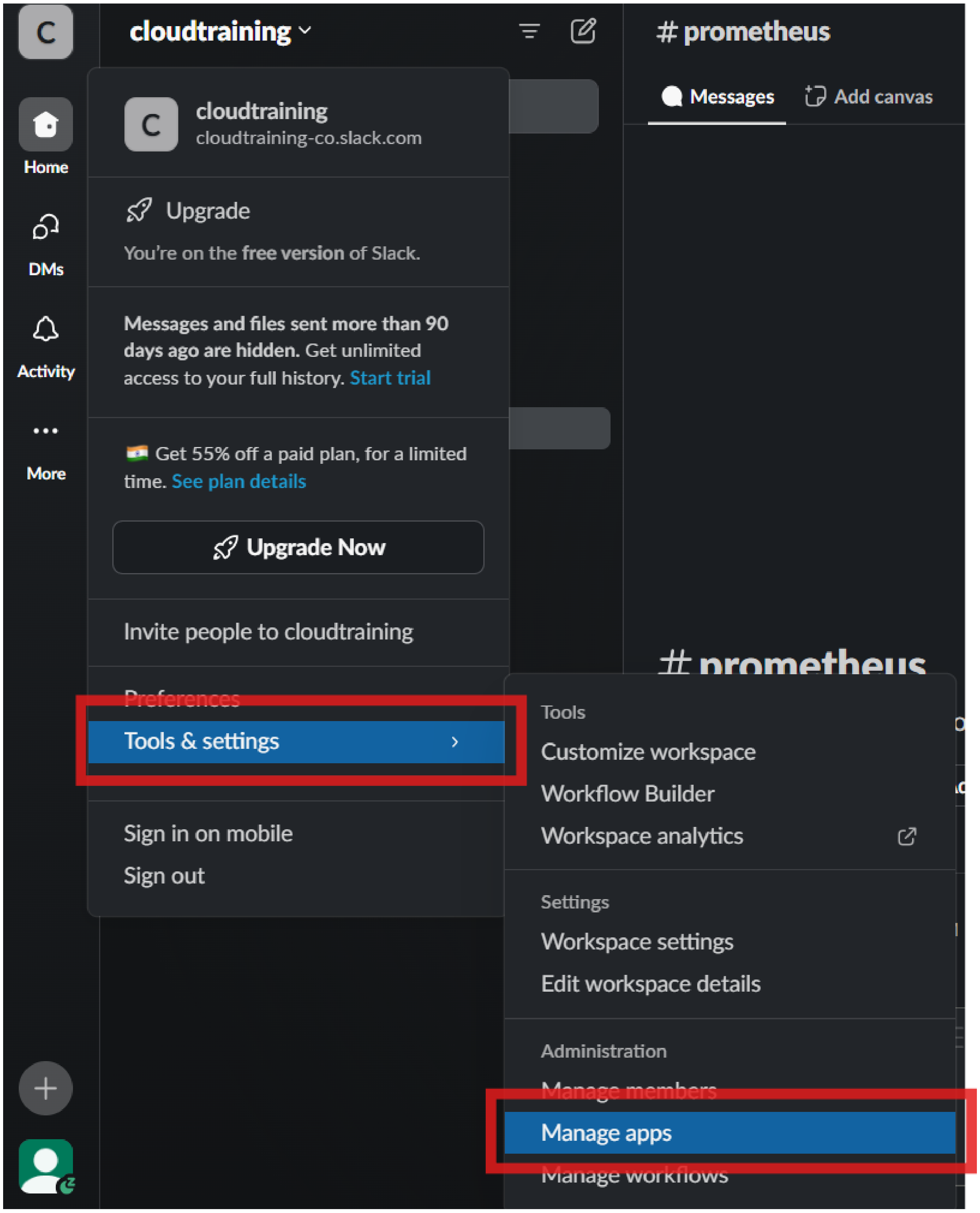
Setting Up Slack Alerts

To get started with Slack, you need to have Slack Account. You can get one created from here.

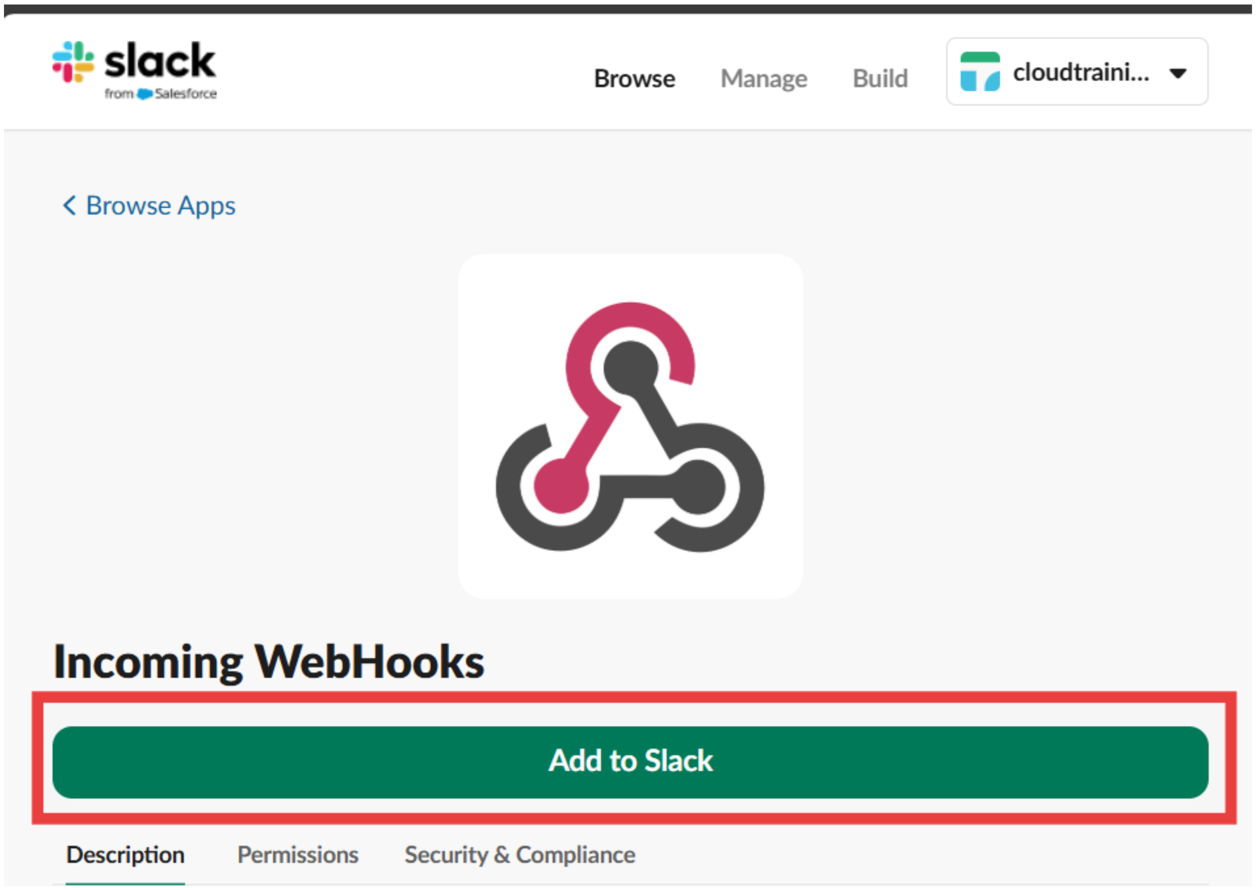
We need a Slack API URL to set up the alerting. Follow the below steps to create an API URL.

Steps:

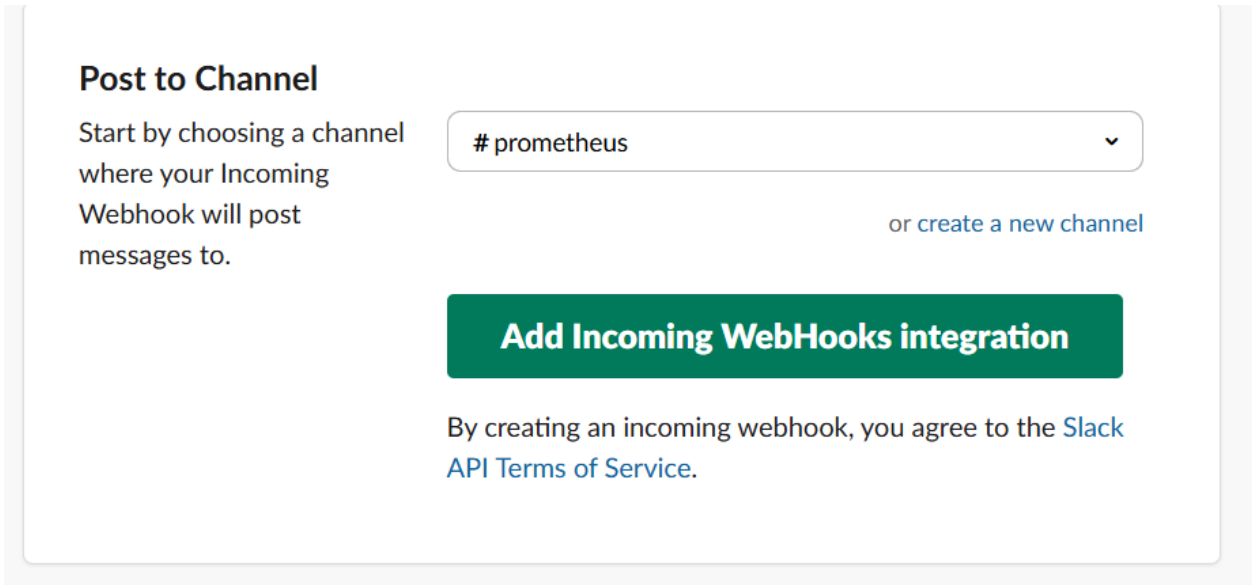
1. Open your Slack app → Workspace Name→ tools & settings → Manage Apps.



2. In manage apps, search for **Incoming Webhooks** and add it to Slack.



3. Select your channel name in which you would like to receive the notification. I have given my channel name as Prometheus.



Post to Channel

Start by choosing a channel where your Incoming Webhook will post messages to.

prometheus

or create a new channel

Add Incoming WebHooks integration

By creating an incoming webhook, you agree to the Slack API Terms of Service.

4. You will receive a Webhook URL, Copy it.
5. Navigate to the Alert manager folder and open alertmanager.yaml file and use the URL which you have copied in the previous step.

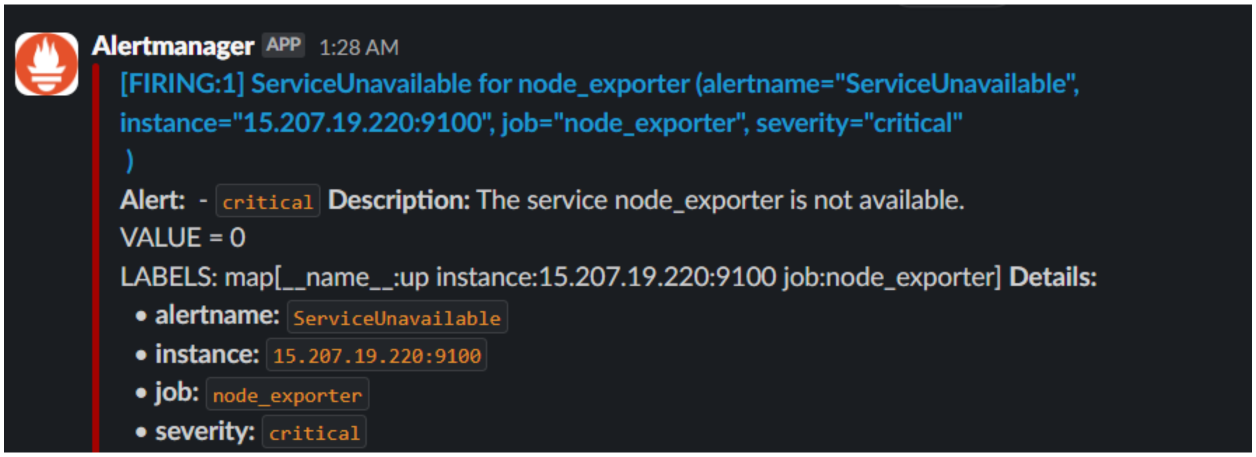
```
global:
  resolve_timeout: 1m
  slack_api_url: 'YOURAPIURL'

route:
  receiver: 'slack-notifications'

receivers:
  - name: 'slack-notifications'
    slack_configs:
      - channel: '#YOURCHANNELNAME'
        send_resolved: true
        icon_url: 'https://avatars3.githubusercontent.com/u/338
        title: |-
          [{{ .Status | toUpper }}{{ if eq .Status "firing" }}:
          {{- 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: >-
          {{ range .Alerts -}}
          *Alert:* {{ .Annotations.title }}{{ if .Labels.severi
          *Description:* {{ .Annotations.description }}
          *Details:*
            {{ range .Labels.SortedPairs }} • *{{ .Name }}:* `{
              {{ end }}
            {{ end }}
```

6. Save it.
7. Restart the Prometheus server and alert manager.

Now if you stop your service, you will receive a notification on Slack. It should look like the below.



```
Alertmanager APP 1:28 AM
[FIRING:1] ServiceUnavailable for node_exporter (alertname="ServiceUnavailable",
instance="15.207.19.220:9100", job="node_exporter", severity="critical"
)
Alert: - critical Description: The service node_exporter is not available.
VALUE = 0
LABELS: map[__name__:up instance:15.207.19.220:9100 job:node_exporter] Details:
 • alertname: ServiceUnavailable
 • instance: 15.207.19.220:9100
 • job: node_exporter
 • severity: critical
```

Setting Up Pager Duty Alerts

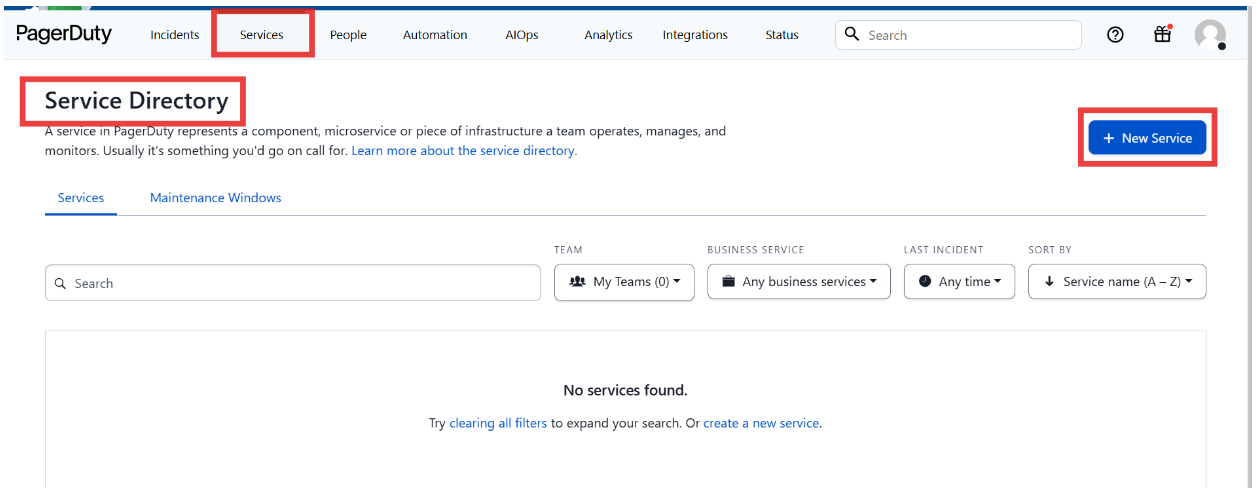
What is Pager Duty ?

Pager Duty is a cloud based incident management platform which is designed to detect , respond, resolve IT issues and other operational incidents quickly.

To set up Pager Duty alerts, we first need to have an account created. It is a paid service, however a 14 day trial is available.

Steps:

1. After account creation, login into the service.
2. Go to Services → Service Discovery → New Service.



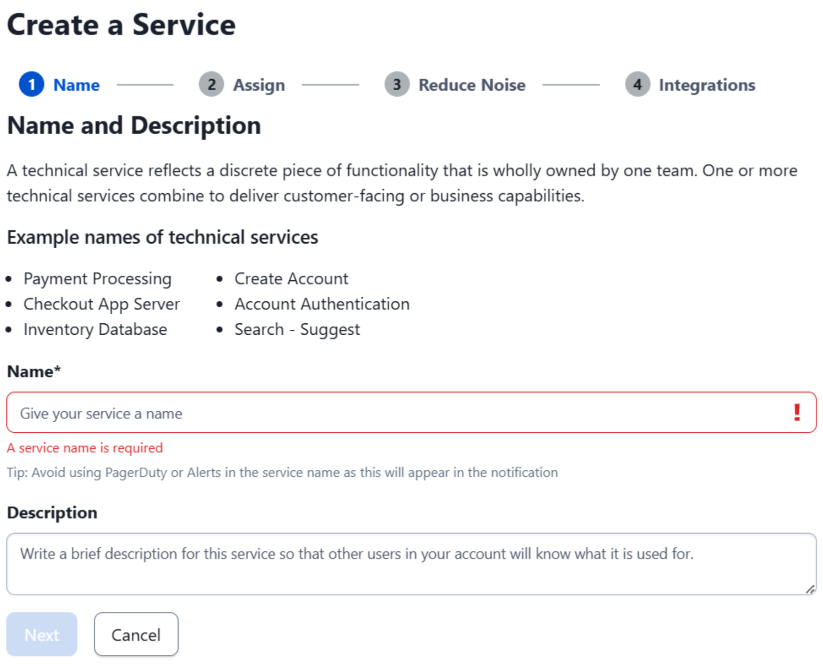
Service Directory

A service in PagerDuty represents a component, microservice or piece of infrastructure a team operates, manages, and monitors. Usually it's something you'd go on call for. Learn more about the service directory.

No services found.

Try clearing all filters to expand your search. Or create a new service.

3. Give a name to the service.



Create a Service

Name and Description

A technical service reflects a discrete piece of functionality that is wholly owned by one team. One or more technical services combine to deliver customer-facing or business capabilities.

Example names of technical services

- Payment Processing
- Checkout App Server
- Inventory Database
- Create Account
- Account Authentication
- Search - Suggest

Name*

Give your service a name

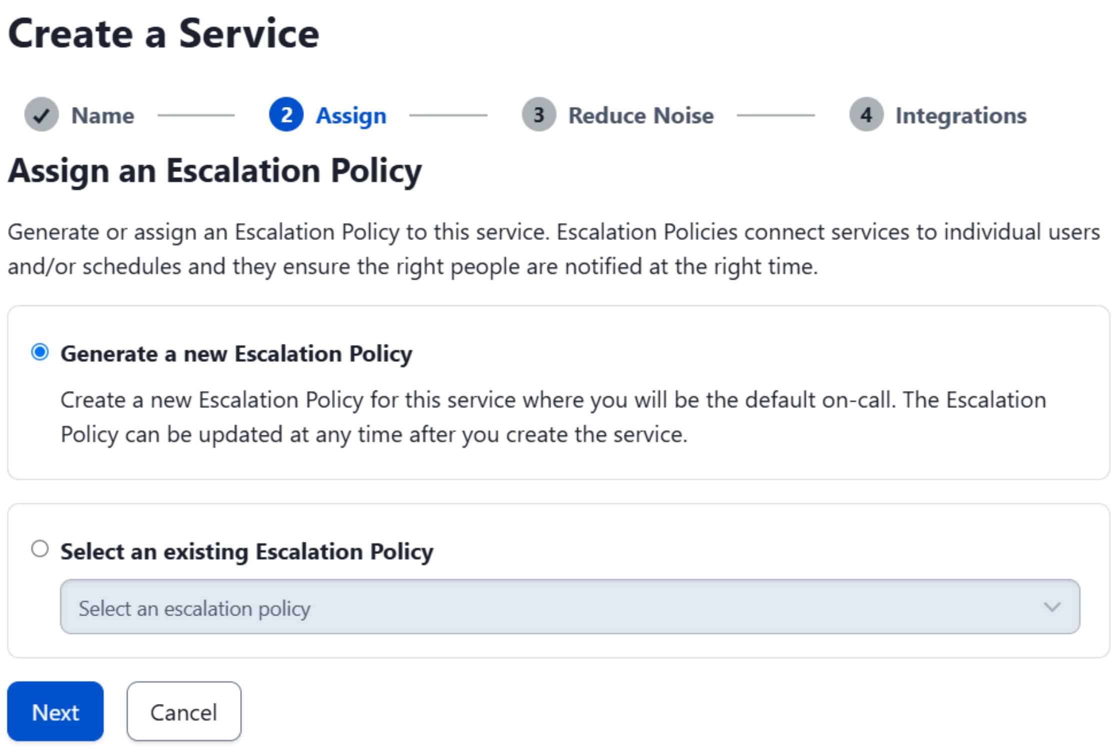
A service name is required

Tip: Avoid using PagerDuty or Alerts in the service name as this will appear in the notification

Description

Write a brief description for this service so that other users in your account will know what it is used for.

4. Assign an Escalation Policy.



Create a Service

✓ Name — 2 Assign — 3 Reduce Noise — 4 Integrations

Assign an Escalation Policy

Generate or assign an Escalation Policy to this service. Escalation Policies connect services to individual users and/or schedules and they ensure the right people are notified at the right time.

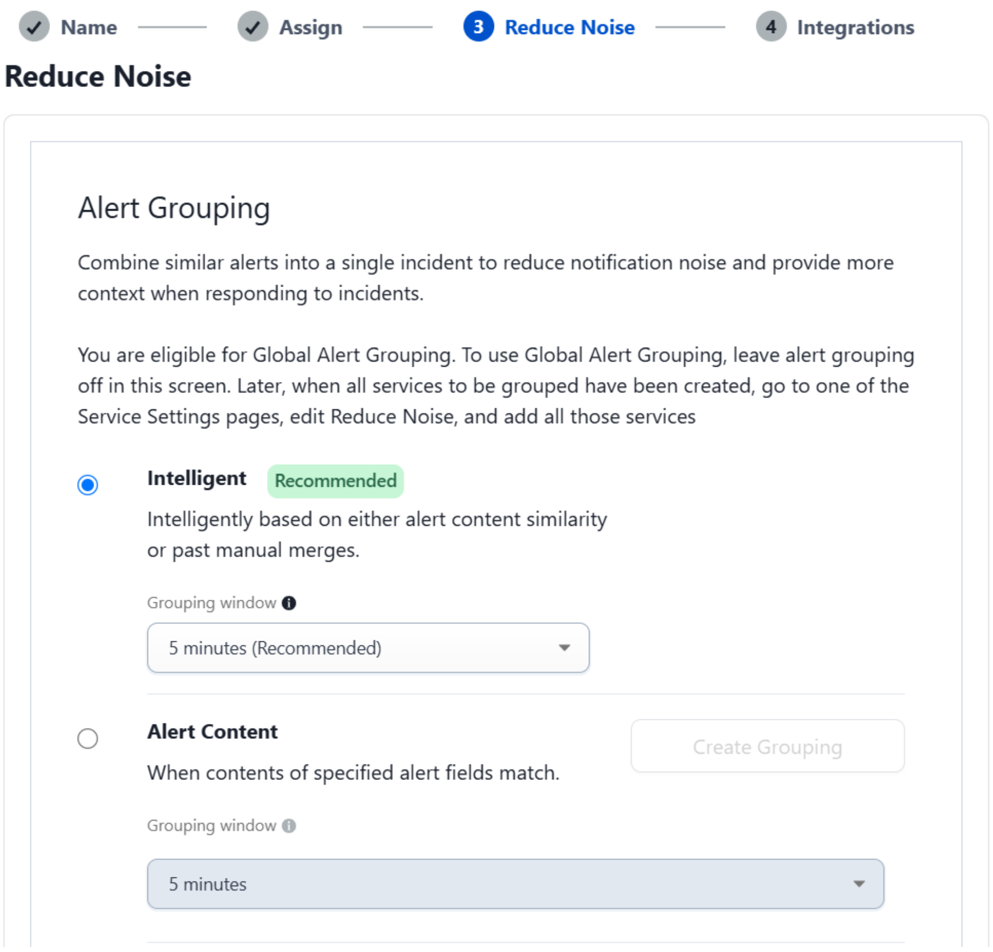
● Generate a new Escalation Policy

Create a new Escalation Policy for this service where you will be the default on-call. The Escalation Policy can be updated at any time after you create the service.

○ Select an existing Escalation Policy

Select an escalation policy

5. In Reduce Noise, keep the value as defaults.



Reduce Noise

Alert Grouping

Combine similar alerts into a single incident to reduce notification noise and provide more context when responding to incidents.

You are eligible for Global Alert Grouping. To use Global Alert Grouping, leave alert grouping off in this screen. Later, when all services to be grouped have been created, go to one of the Service Settings pages, edit Reduce Noise, and add all those services

[✓ **Intelligent**] Recommended

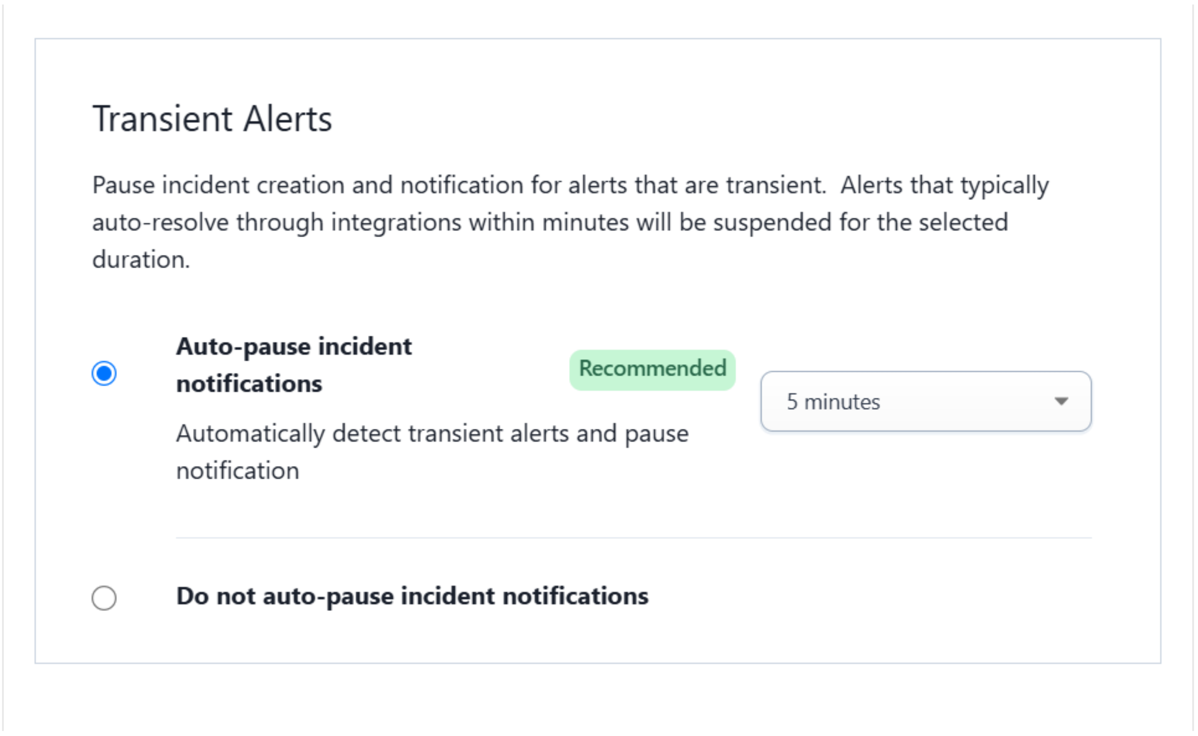
Intelligently based on either alert content similarity or past manual merges.

Grouping window: 5 minutes (Recommended)

Alert Content

When contents of specified alert fields match.

Grouping window: 5 minutes



Transient Alerts

Pause incident creation and notification for alerts that are transient. Alerts that typically auto-resolve through integrations within minutes will be suspended for the selected duration.

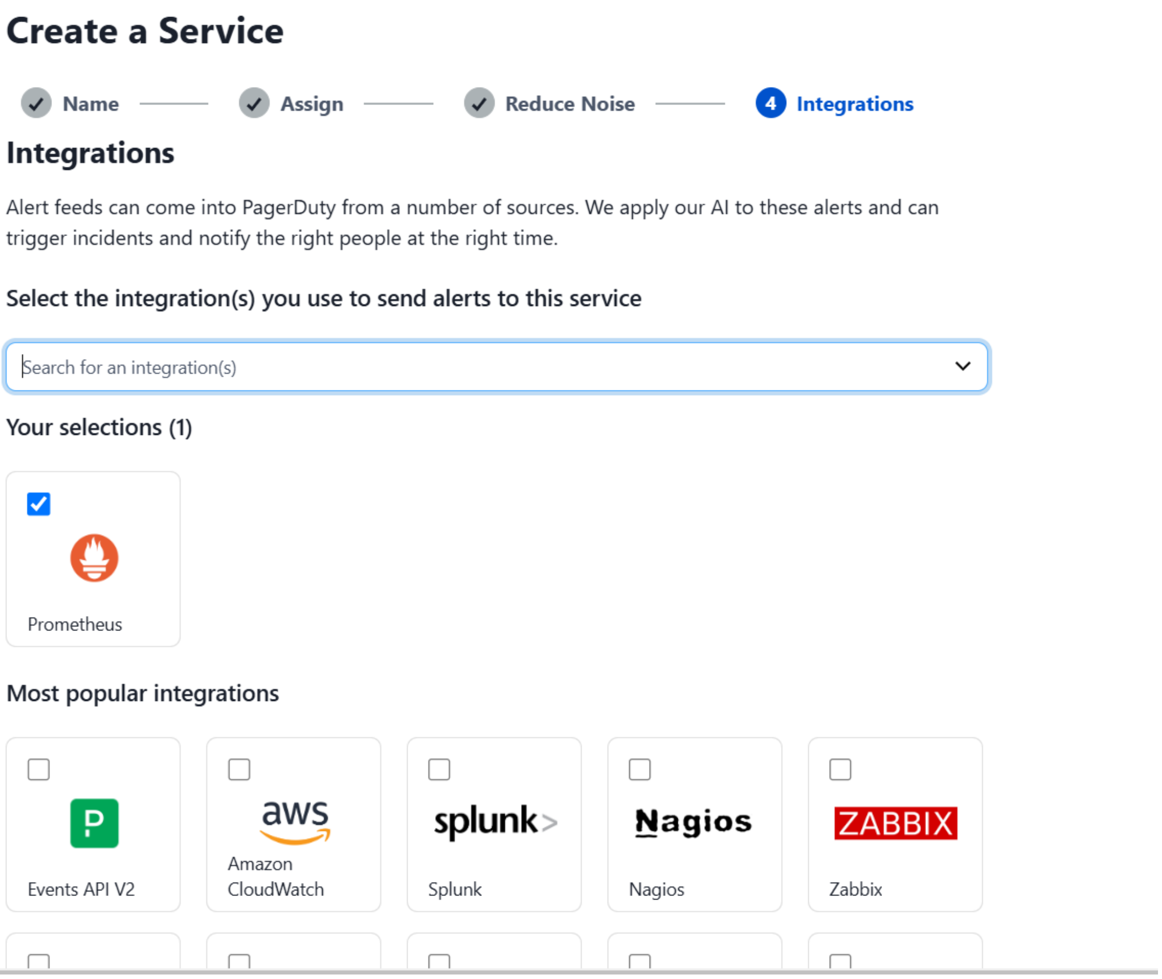
[✓ **Auto-pause incident notifications**] Recommended

Automatically detect transient alerts and pause notification

5 minutes

Do not auto-pause incident notifications

6. In the Integration, select **Prometheus.**



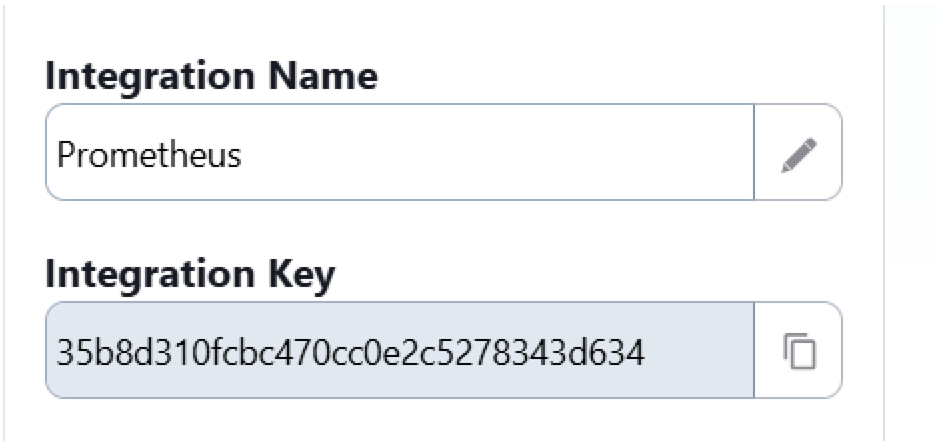
Create a Service

Integrations

Alert feeds can come into PagerDuty from a number of sources. We apply our AI to these alerts and can trigger incidents and notify the right people at the right time.

Select the integration(s) you use to send alerts to this service

7. Click on create service.
8. An integration key will be displayed, Copy it.



Integration Name

Prometheus

Integration Key

35b8d310fcbc470cc0e2c5278343d634

9. We have to update the alertmanager.yaml file and use the routing_key (integration key) and set the Pagerduty_URL same as default which is " https://events.pagerduty.com/v2/enqueue ".
10. The config will look like the below.

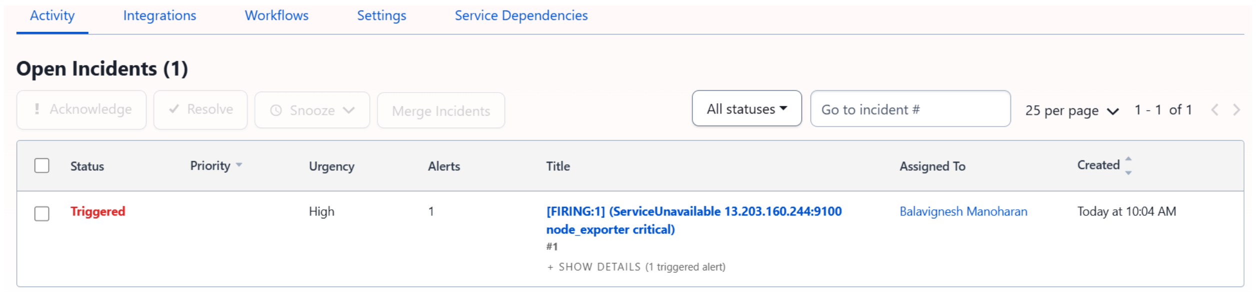
```
global:
  resolve_timeout: 1m
  pagerduty_url: 'https://events.pagerduty.com/v2/enqueue'

route:
  receiver: 'pagerduty-notifications'

receivers:
  - name: 'pagerduty-notifications'
    pagerduty_configs:
      - routing_key: '35b8d310fcbc470cc0e2c5278343d634'
        send_resolved: true
```

11. Add the above code to the alertmanager.yaml and save it.
12. Restart the Prometheus server and alert manager.

Now if you stop your service, you will receive a notification on Pager duty. It should look like the below.



Open Incidents (1)

	Status	Priority	Urgency	Alerts	Title	Assigned To	Created
☐	Triggered		High	1	[FIRING:1] (ServiceUnavailable 13.203.160.244:9100 node_exporter critical) #1 + SHOW DETAILS (1 triggered alert)	Balavignesh Manoharan	Today at 10:04 AM

You can configure the PagerDuty user settings on how you should be notified. I have set both email and phone call. I got notified using both the ways.

To set up a phone call.

Go to My profile → Contact Information → Request for your number at first.

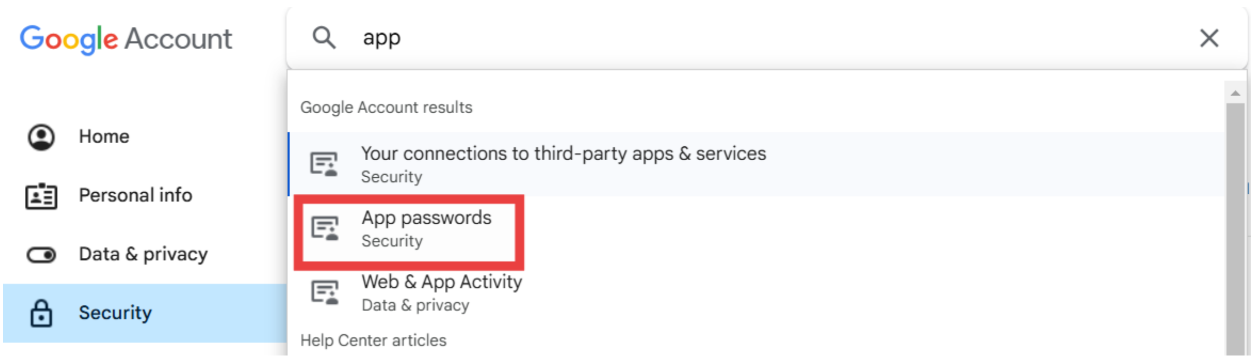
You will receive a confirmation email. After that you can add your phone number with the country code. Once added, whenever there is an alert triggered you will receive both email and phone call.

Setting Up Gmail Alerts

To get started with Gmail alerts, we need to create an App Password. For App password to be created, you need to enable 2-Step verification.

Steps:

1. Go to https://myaccount.google.com/ → Security → 2-Step verification - Turn ON.
2. Below the 2-step Verification, you will find App password. If you could not find App password, just type in the search bar and create a new App password.



3. Click on the App password, and create a new password.
4. Copy the password.
5. Go to alertmanager.yaml file and paste the below code.
6. We have to update the app password in the auth_password parameter.
7. The alertmanager config will look like the below.

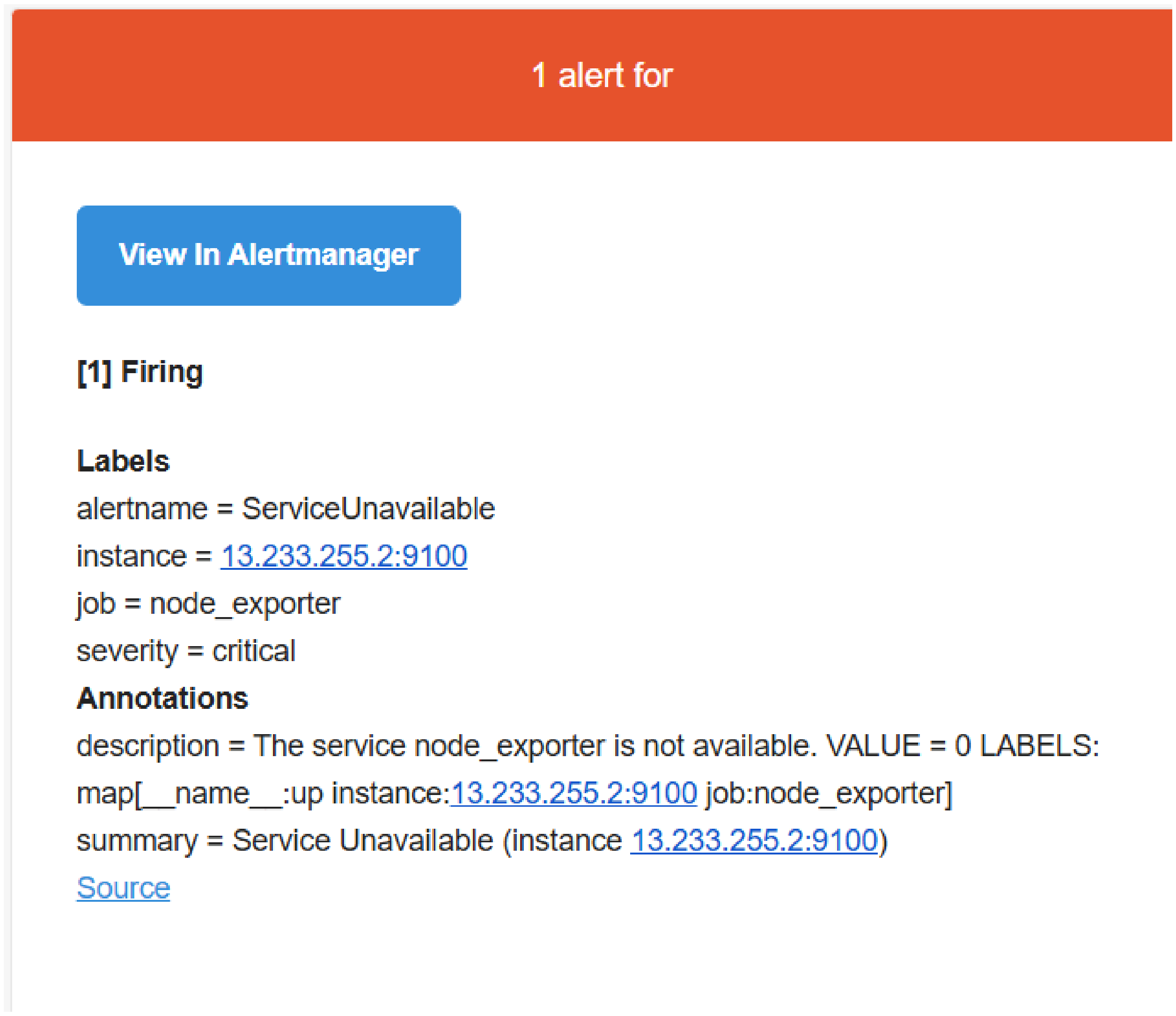
```
global:
    resolve_timeout: 1m

   route:
    receiver: 'gmail-notifications'
```

```
receivers:
- name: 'gmail-notifications'
 email_configs:
 - to: YOUREMAIL@gmail.com
   from: monitoringinstances@gmail.com
   smarthost: smtp.gmail.com:587
   auth_username: YOUREMAIL@gmail.com
   auth_identity: YOUREMAIL@gmail.com
   auth_password: password
   send_resolved: true
```

8. Save it.
9. Restart the Prometheus server and alert manager.

One last time if you stop your service, you will receive a notification on your Gmail. It should look like the below.


1 alert for

View In Alertmanager

[1] Firing

Labels
alertname = ServiceUnavailable
instance = 13.233.255.2:9100
job = node_exporter
severity = critical
Annotations
description = The service node_exporter is not available. VALUE = 0 LABELS: map[__name__:up instance:13.233.255.2:9100 job:node_exporter]
summary = Service Unavailable (instance 13.233.255.2:9100)
Source

Voila! Its working for gmail as expected.

Thus in this article, we have successfully configured Alertmanager to handle multi channel alerting for the monitoring system across PagerDuty, Slack and Gmail.

Conclusion

We have successfully setup a robust alerting system which ensures that the application monitoring is highly efficient and responsive. With these integrations you are ready to handle real-world scenarios for detecting and resoling incidents.

Do try all the different methods of sending an alert and let me know which one of the above does your organization use for Alerting.

Thanks for reading !

If you found this post useful, give it a like👍

Repost♻️

Follow @Bala Vignesh for more such posts 💯🚀