# Observing Cloud Resources

*SRE Project Template*

## Categorize Responsibilities

| Prometheus and Grafana Screenshots |
|---|
| Provide a screenshot of the Prometheus node_exporter service running on the EC2 instance. Use the following command to show that the system is running: `sudo systemctl status node_exporter` |

```
ubuntu@ip-10-100-12-169:~$ sudo systemctl status node_exporter
● node_exporter.service - Node Exporter
     Loaded: loaded (/etc/systemd/system/node_exporter.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2023-03-22 16:05:07 UTC; 1min 26s ago
   Main PID: 11702 (node_exporter)
      Tasks: 3 (limit: 1140)
     CGroup: /system.slice/node_exporter.service
             └─11702 /usr/local/bin/node_exporter

Mar 22 16:05:07 ip-10-100-12-169 node_exporter[11702]: level=info ts=2023-03-22T16:05:07.368Z caller=node_exporter.go:115 collector=thermal_zone
Mar 22 16:05:07 ip-10-100-12-169 node_exporter[11702]: level=info ts=2023-03-22T16:05:07.368Z caller=node_exporter.go:115 collector=time
Mar 22 16:05:07 ip-10-100-12-169 node_exporter[11702]: level=info ts=2023-03-22T16:05:07.368Z caller=node_exporter.go:115 collector=timex
Mar 22 16:05:07 ip-10-100-12-169 node_exporter[11702]: level=info ts=2023-03-22T16:05:07.368Z caller=node_exporter.go:115 collector=udp_queues
Mar 22 16:05:07 ip-10-100-12-169 node_exporter[11702]: level=info ts=2023-03-22T16:05:07.368Z caller=node_exporter.go:115 collector=uname
Mar 22 16:05:07 ip-10-100-12-169 node_exporter[11702]: level=info ts=2023-03-22T16:05:07.368Z caller=node_exporter.go:115 collector=vmstat
Mar 22 16:05:07 ip-10-100-12-169 node_exporter[11702]: level=info ts=2023-03-22T16:05:07.368Z caller=node_exporter.go:115 collector=xfs
Mar 22 16:05:07 ip-10-100-12-169 node_exporter[11702]: level=info ts=2023-03-22T16:05:07.368Z caller=node_exporter.go:115 collector=zfs
Mar 22 16:05:07 ip-10-100-12-169 node_exporter[11702]: level=info ts=2023-03-22T16:05:07.368Z caller=node_exporter.go:199 msg="Listening on" address=:9100
Mar 22 16:05:07 ip-10-100-12-169 node_exporter[11702]: level=info ts=2023-03-22T16:05:07.373Z caller=tls_config.go:191 msg="TLS is disabled." http2=false
ubuntu@ip-10-100-12-169:~$
```
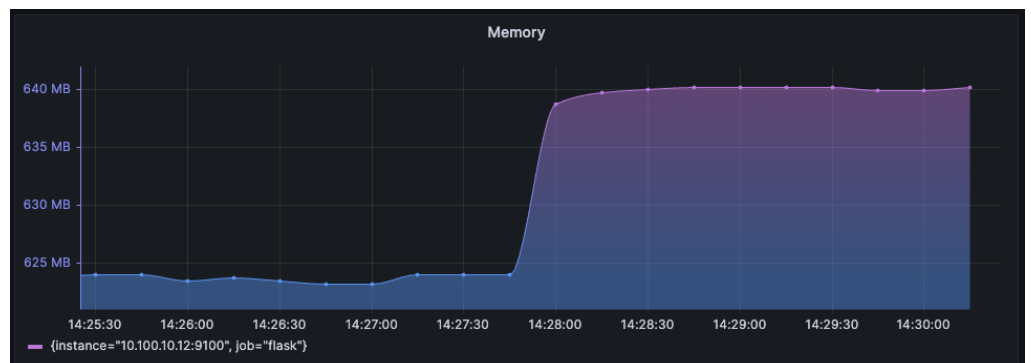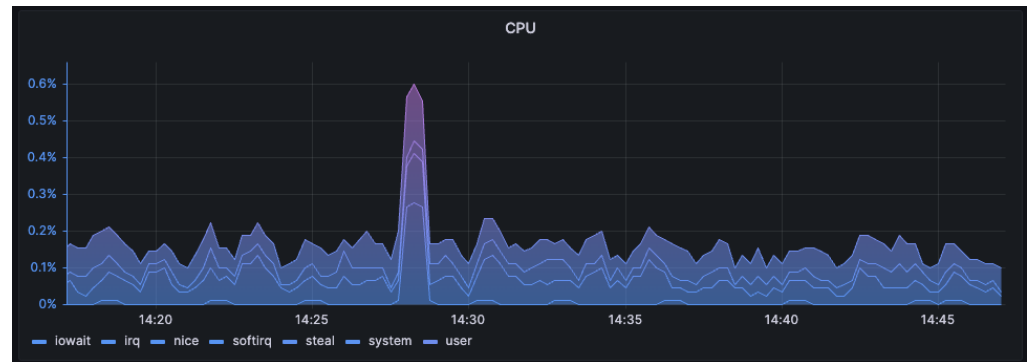
| Host Metric (CPU, RAM, Disk, Network) | Dashboard |
|---|---|
| `node_memory_MemTotal_bytes{job="flask"} - node_memory_MemAvailable_bytes{job="flask"}` |  |

Memory

640 MB / 635 MB / 630 MB / 625 MB

14:25:30  14:26:00  14:26:30  14:27:00  14:27:30  14:28:00  14:28:30  14:29:00  14:29:30  14:30:00
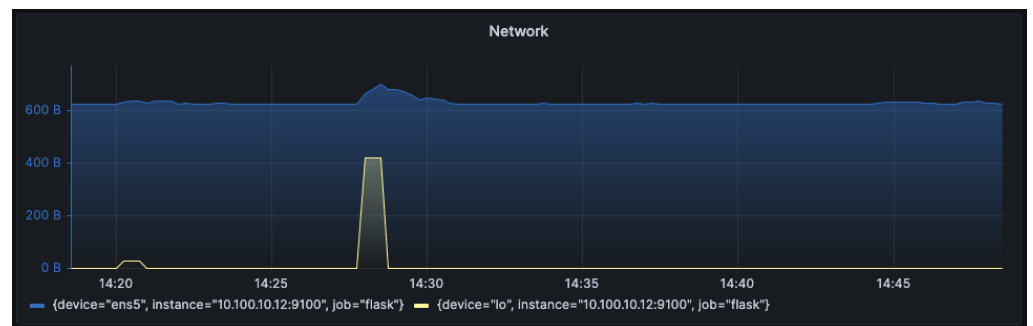
{instance="10.100.10.12:9100", job="flask"}

```
avg(rate(node_cpu_second
s_total{job="flask",mode
!='idle'}[1m])) by
(mode) * 100
```



```
node_disk_io_now{job="fl
ask"}
```



```
rate(node_network_transm
it_bytes_total{job="flas
k"}[1m])
```



## Responsibilities

1. The development team wants to release an emergency hotfix to production. Identify two roles of the SRE team who would be involved in this and why.

*Release Manager: To make sure the change satisfies all dependencies and criteria, prepare a rollback procedure, then execute the release.*
*Monitoring Engineer: To keep an eye on the system metrics and alerts during the release so any possible problems are caught ASAP.*

2. The development team is in the early stages of planning to build a new product. Identify two roles of the SRE team that should be invited to the meeting and why.

*System Architect: To plan necessary infrastructure changes, discuss technology options, prepare documentation for the Developer team.*
*Team Lead: To provide high level contribution to architectural discussions and collect necessary information for the SRE team for further tasks related to the new product.*

3. The emergency hotfix from question 1 was applied and is causing major issues in production. Which SRE role would primarily be involved in mitigating these issues?

*The prepared rollback plan should be executed by the **Release Manager**.*

# Team Formation and Workflow Identification

| API Monitoring and Notifications |
|:---:|

Display the status of an API endpoint: Provide a screenshot of the Grafana dashboard that will show at which point the API is unhealthy (non-200 HTTP code), and when it becomes healthy again (200 HTTP code).
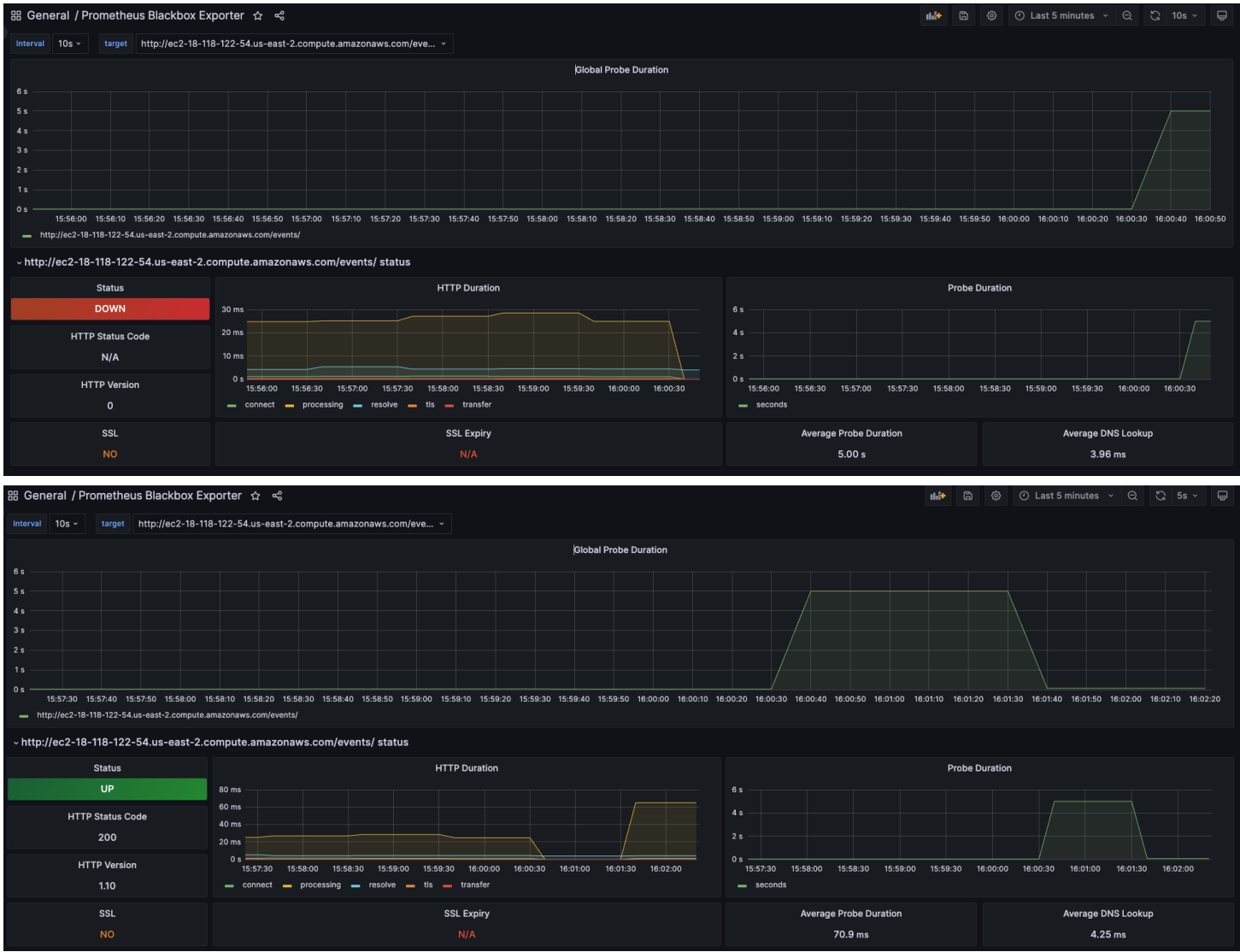




Create a notification channel: Provide a screenshot of the Grafana notification which shows the summary of the issue and when it occurred.

**prometheus-alertmanager** 16:42

**[RESOLVED] Flask GetEvents is unhealthy Flask application (http://ec2-18-118-122-54.us-east-2.compute.amazonaws.com/events/ blackbox)**

**Resolved**

Value: B=200, C=0 Labels:

- alertname = Flask GetEvents is unhealthy
- grafana_folder = Flask application
- instance = http://ec2-18-118-122-54.us-east-2.compute.amazonaws.com/events/
- job = blackbox

Annotations:

- description = Endpoint is responding with non 200 status code
- summary = Flash application GET events endpoint is down

Source: http://localhost:3000/alerting/grafana/t_V5u5fVk/view

Silence: http://localhost:3000/alerting/silence/new?
alertmanager=grafana&matcher=alertname%3DFlask+GetEvents+is+unhealthy&matche
r=grafana_folder%3DFlask+application&matcher=instance%3Dhttp%3A%2F%2Fec2-
18-118-122-54.us-east-
2.compute.amazonaws.com%2Fevents%2F&matcher=job%3Dblackbox

**View URL**

Configure alert rules: Provide a screenshot of the alert rules list in Grafana.

UDACITY

# Applying the Concepts

**Graph 1**
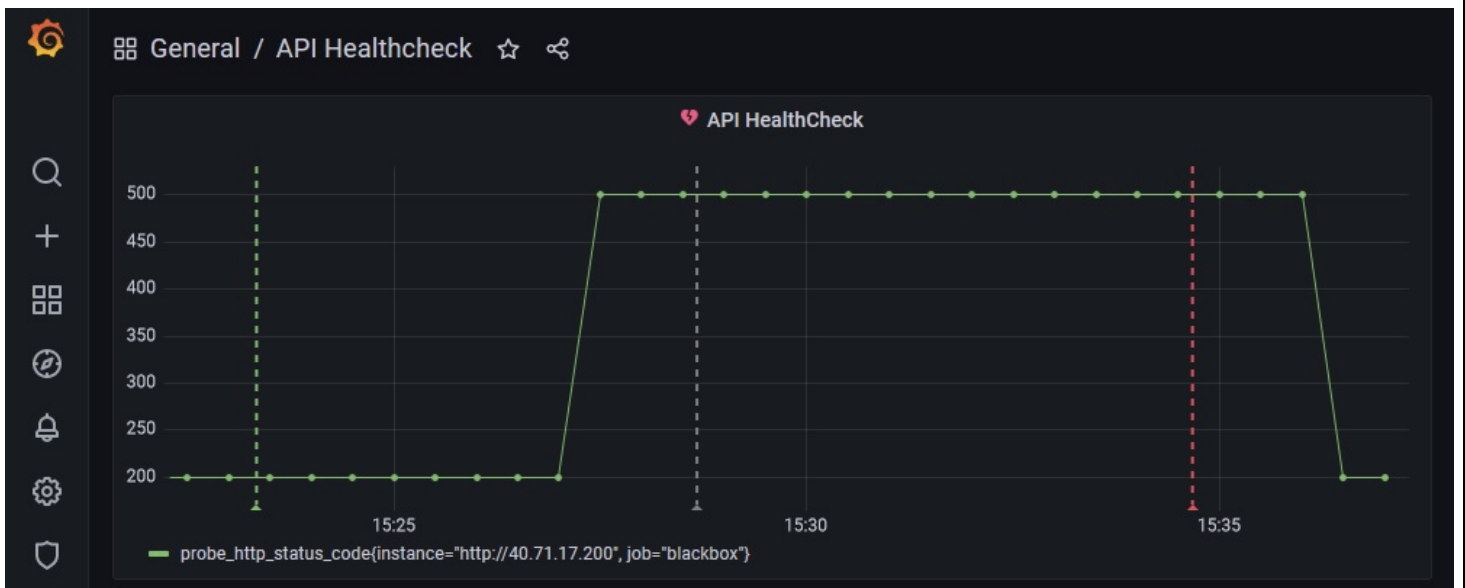
**4a. Given the above graph, where does it show that the API endpoint is down? Where on the graph does this show that the API is healthy again?**

*This seems like a synthetic test checking the endpoint under [http://40.71.17.200](http://40.71.17.200), running in 30s frequency. The graph displayed the returned http status code by the endpoint. In the example at 15:27 the endpoint stated to return 500, Server Error. I lasted for 9 minutes, ending at 15:36, where the received response code went back to 200, OK again.*
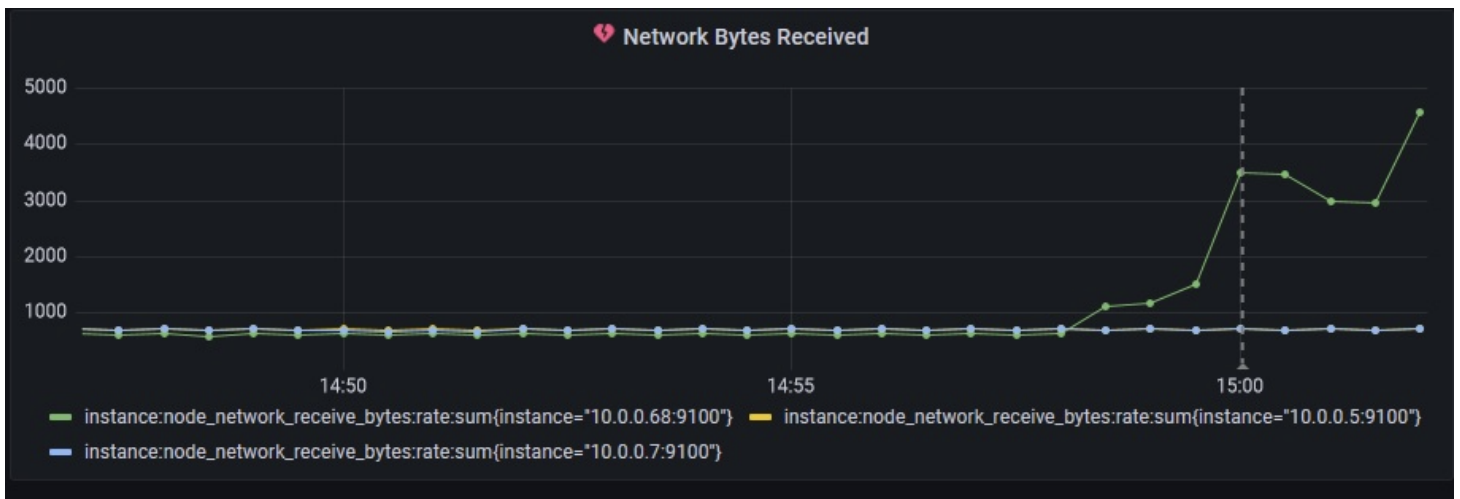
**4b. If there was no SRE team, how would this outage affect customers?**

*Most possibly some of the system functionalities were affected. From the GET request and 200 status code, I suppose this endpoint might be used to serve some kind of information to the end user. The SRE team hopefully previously set up some kind of redundant configuration for this API so this outage did not affect the users. Otherwise the outage must be followed up with the Post Incident procedures.*

**4c. What could be put in place so that the SRE team could know of the outage before the customer does?**

*In my view the following things can be done:*
*System level monitoring can help to catch the root case of the issue even before the symptom occurs.*
*Synthetic tests with alerting, as in the example, can be configured in order to know about the issue as soon as the symptom is present.*
*Infrastructure configuration with redundancy can help to reduce the scale of the outage.*

## Graph 2



Network Bytes Received

Legend:
- instance:node_network_receive_bytes:rate:sum{instance="10.0.0.68:9100"}
- instance:node_network_receive_bytes:rate:sum{instance="10.0.0.5:9100"}
- instance:node_network_receive_bytes:rate:sum{instance="10.0.0.7:9100"}

5a. Given the above graph, which instance had the increase in traffic, and approximately how many bytes did it receive (feel free to round)?

*The instance with IP 10.0.0.68 on port 9100 has the increased traffic. It experienced around 3000 bytes / second for the visible 2 minutes (120s) so the overall traffic must be around: 120 * 3000 = 360000 bytes = 360k*

5b. Which team members on the SRE team would be interested in this graph and why?

*Probably the Monitoring Engineer would be interested to consider whether setting up an alerting rule would make sense.*