

Observing Cloud Resources

SRE Assessment 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`

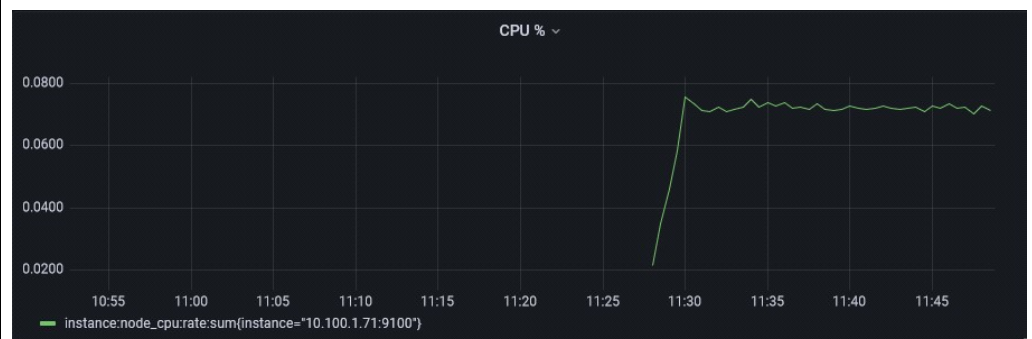
```
ubuntu@ip-172-31-34-33: ~  
ubuntu@ip-172-31-34-33:~$ sudo systemctl status node_exporter  
● node_exporter.service - Node Exporter  
   Loaded: loaded (/etc/systemd/system/node_exporter.service; enabled; vendor preset: enabled)  
   Active: active (running) since Thu 2022-07-14 17:09:55 UTC; 49s ago  
     Main PID: 11583 (node_exporter)  
        Tasks: 5 (limit: 1109)  
       CGroup: /system.slice/node_exporter.service  
              └─11583 /usr/local/bin/node_exporter  
  
Jul 14 17:09:55 ip-172-31-34-33 node_exporter[11583]: level=info ts=2022-07-14T17:09:55.918Z caller=node_exporter.go:115 collector=thermal_zone  
Jul 14 17:09:55 ip-172-31-34-33 node_exporter[11583]: level=info ts=2022-07-14T17:09:55.918Z caller=node_exporter.go:115 collector=time  
Jul 14 17:09:55 ip-172-31-34-33 node_exporter[11583]: level=info ts=2022-07-14T17:09:55.918Z caller=node_exporter.go:115 collector=timex  
Jul 14 17:09:55 ip-172-31-34-33 node_exporter[11583]: level=info ts=2022-07-14T17:09:55.918Z caller=node_exporter.go:115 collector=udp_queues  
Jul 14 17:09:55 ip-172-31-34-33 node_exporter[11583]: level=info ts=2022-07-14T17:09:55.918Z caller=node_exporter.go:115 collector=uname  
Jul 14 17:09:55 ip-172-31-34-33 node_exporter[11583]: level=info ts=2022-07-14T17:09:55.918Z caller=node_exporter.go:115 collector=vmstat  
Jul 14 17:09:55 ip-172-31-34-33 node_exporter[11583]: level=info ts=2022-07-14T17:09:55.918Z caller=node_exporter.go:115 collector=xfs  
Jul 14 17:09:55 ip-172-31-34-33 node_exporter[11583]: level=info ts=2022-07-14T17:09:55.918Z caller=node_exporter.go:115 collector=zfs  
Jul 14 17:09:55 ip-172-31-34-33 node_exporter[11583]: level=info ts=2022-07-14T17:09:55.918Z caller=node_exporter.go:199 msg="Listening on" address=:9100  
Jul 14 17:09:55 ip-172-31-34-33 node_exporter[11583]: level=info ts=2022-07-14T17:09:55.919Z caller=tls_config.go:191 msg="TLS is disabled." http2=false  
ubuntu@ip-172-31-34-33:~$
```

Host Metric

(CPU, RAM, Disk, Network)

Dashboard

CPU



Memory	<p>node_memory_MemAvailable_bytes(container="node-exporter", endpoint="http-metrics", instance="10.100.1.71:9100", job="node-exporter", namespace="mon")</p>
Network	<p>instance:node_network_receive_bytes:rate:sum(instance="10.100.1.71:9100")</p>
Disk I/O	<p>node_disk_io_now(container="node-exporter", device="nvme0n1", endpoint="http-metrics", instance="10.100.1.71:9100", job="node-exporter", namespace="mi")</p>

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.

The monitoring engineer and release manager are responsible for releasing hotfixes to production. Monitoring engineers manage alerting, are usually the first to know of an incident (in case the hotfix fails), and the release manager is involved in change management, code releases, and executing releases.

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.

In the early stages of planning the team lead, and system architect should be invited to the meeting. The team lead directs the work, contributes to architectural meetings, and creates workflows for the team. The architect is involved in creating/documenting the infrastructure, and making recommendations for new tech.

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 release manager is responsible as they are in charge of executing a release, and rolling back in case of issues.

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.

The screenshot shows a Grafana notification message for "testslackapp" (APP) at 2:36 PM. The message is titled "[FIRING:1] (HttpSuccess)" and contains the following text:

```
**Firing**
```

Value: [var='BO' metric='probe_http_status_code(instance="http://3.144.120.111", job="blackbox")' labels={'__name__': 'probe_http_status_code', instance='http://3.144.120.111', job='blackbox'} value=200]

Labels:

- alertname = HttpSuccess

Annotations:



Source: <http://localhost:3000/alerting/grafana/JY-zkigVk/view>

[Show more](#)

Grafana v9.0.2 | Today at 2:36 PM

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

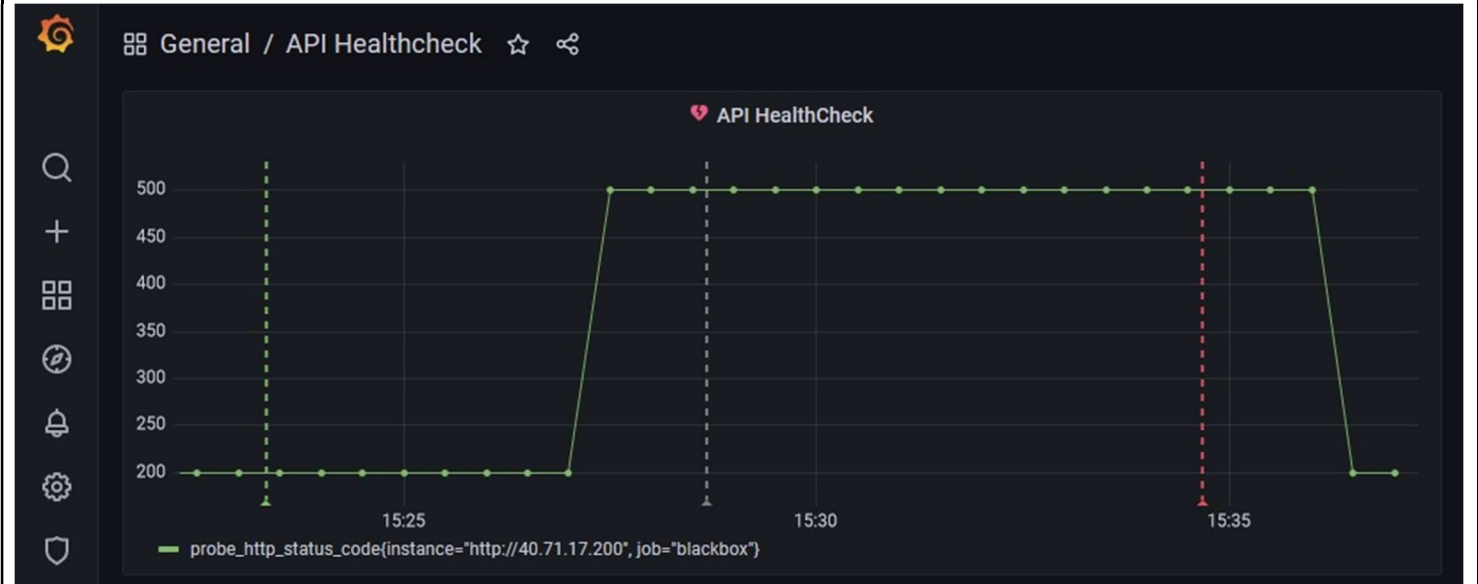
AlertFolder

3 rules: 1 firing |  

State	Name	Health	Summary
> Normal	HttpFail	ok	
> Firing for 5m	HttpSuccess	ok	
> Normal	LowMemoryUsage	ok	

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?

In the above graph, the API endpoint is down at 15:29, and is healthy again after :1537.

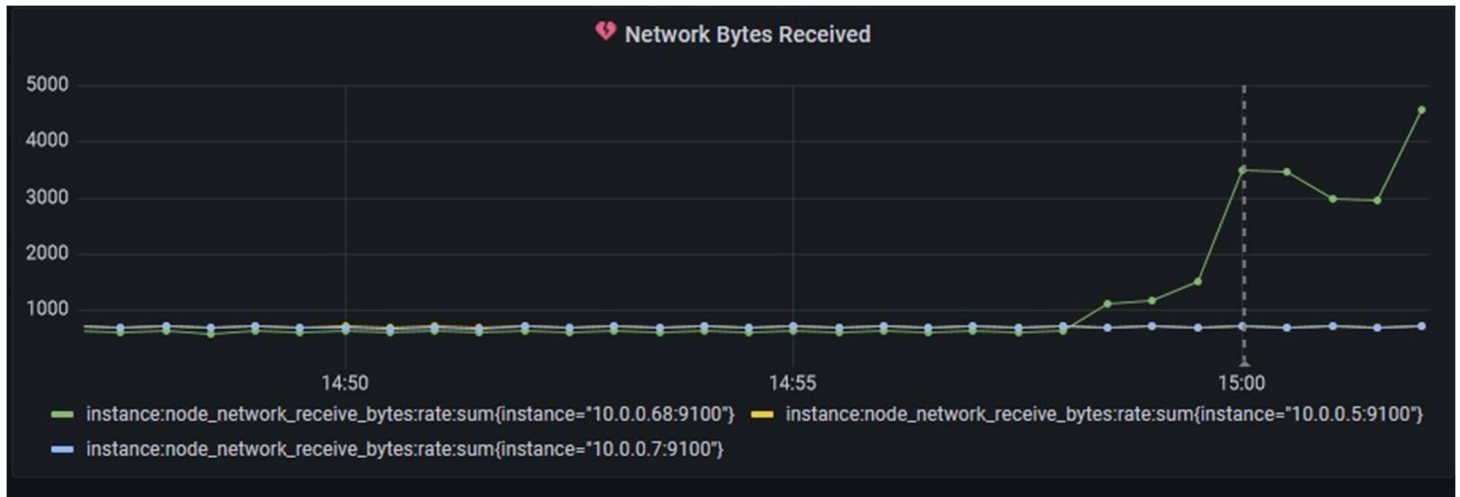
4b. If there was no SRE team, how would this outage affect customers?

Assuming there was no other team structure, individual(s) responsible, the outage could result in loss of data, frustration among customers causing loss of reputation. Without some kind of monitoring the outage might go unnoticed.

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

Synthetic monitoring could be implemented to know if the endpoint is down, and when the endpoint goes down. In Grafana, the metric `probe_dns_lookup_time_seconds` can be monitored, for any number of endpoints.

Graph 2



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 10.0.0.68:9100 instance experienced the increase in network traffic. It peaked at just under 5000 bytes. During the ramp up, it peaked at around 3500 bytes, dropped to 3000 bytes (before peaking again).

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

The infrastructure engineer would be interested as they are involved in development and operations. They could research the current system patches and updates for potential issues. From an architectural standpoint, the system architect would be interested as maybe it indicates a needed update to the design for performance reasons, maybe add a load balancer.