😊 Node Exporter

Node Exporter is a Prometheus exporter that collects various system-level metrics from a host machine and makes them available for Prometheus to scrape. These metrics include CPU usage, memory usage, disk usage, network statistics, and other system-level information.

Node Exporter runs as a separate process on the host machine and exposes an HTTP endpoint that Prometheus can scrape to collect these metrics. By installing and configuring Node Exporter on each host machine in a system, administrators can gather detailed insights into the performance and health of individual nodes.

Node Exporter is a crucial component in monitoring infrastructure and is often used alongside Prometheus to create comprehensive monitoring solutions for DevOps environments.

😄 Use cases of Node exporter:

Node Exporter has several key use cases in the realm of monitoring and observability:

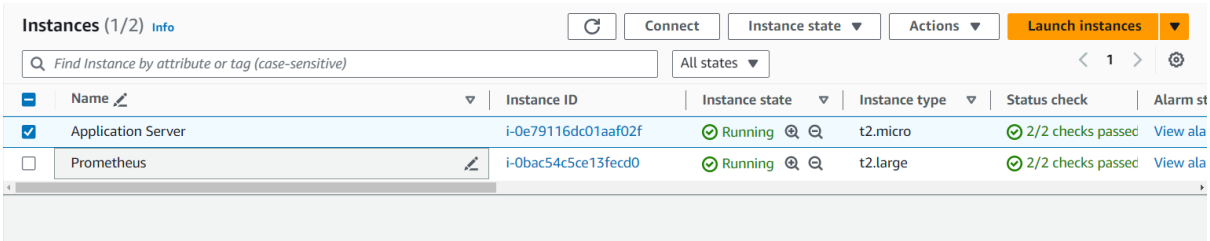
1. **Infrastructure Monitoring:** Node Exporter enables administrators to monitor the health and performance of individual servers and virtual machines within their infrastructure. It provides insights into CPU usage, memory usage, disk I/O, network traffic, and other system-level metrics, helping identify bottlenecks, resource constraints, and potential issues.
2. **Capacity Planning:** By collecting metrics on resource utilization from Node Exporter, teams can perform capacity planning and resource allocation effectively. This involves analyzing historical data to understand trends and patterns, which can inform decisions on scaling infrastructure to meet growing demand or optimizing resource allocation for cost-efficiency.
3. **Performance Analysis:** Node Exporter facilitates performance analysis by providing granular metrics on various aspects of system performance. Administrators can use these metrics to identify performance anomalies, diagnose issues, and optimize system configurations to improve overall performance and reliability.
4. **Alerting and Incident Response:** Node Exporter integrates seamlessly with Prometheus for alerting and incident management. Administrators can define alerting rules based on thresholds and conditions derived from Node Exporter metrics. When anomalies or issues are detected, alerts can be triggered, notifying relevant stakeholders for timely response and resolution.
5. **Security Monitoring:** Node Exporter can also play a role in security monitoring by providing visibility into system-level metrics that can indicate potential security threats or breaches. Metrics such as network connections, process activity, and file system changes can be monitored for abnormal behavior, aiding in intrusion detection and response efforts.

6. **Containerized Environments:** In containerized environments like Docker and Kubernetes, Node Exporter can be deployed as a sidecar container or as a standalone service to monitor individual nodes or worker instances. It provides visibility into resource usage and performance metrics for both host systems and containers, facilitating efficient resource management and troubleshooting.
7. **Cloud Monitoring:** Node Exporter can be deployed on cloud instances to monitor virtual machines and instances hosted on cloud platforms such as AWS, Azure, and Google Cloud Platform. It enables administrators to monitor resource utilization, network traffic, and other system-level metrics in cloud environments, ensuring optimal performance and reliability.

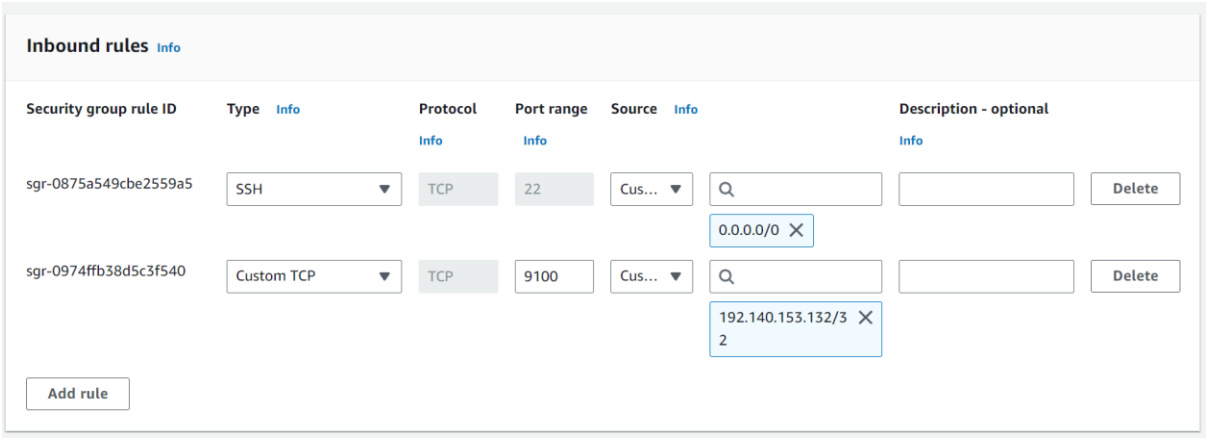
In this guide, we're setting up Node Exporter, which collects system-level metrics from a host machine, and integrating it with Prometheus, a monitoring tool. The end goal is to gather detailed insights into the performance and health of individual nodes in a system. By installing Node Exporter on each host machine and configuring Prometheus to scrape its metrics, administrators can monitor infrastructure, perform capacity planning, analyze performance, set up alerting, and enhance security monitoring. This setup provides a comprehensive monitoring solution for DevOps environments, ensuring optimal performance, reliability, and timely incident response.

😄 To begin with the Lab:

1. In the previous lab we build our Prometheus instance and installed it. Now your Prometheus instance should be running for this lab too.
2. In this lab, first we need to create another instance on the Ubuntu Operating System. Give its name as application server.
3. Then choose the instance type as t2.micro and then launch your instance.

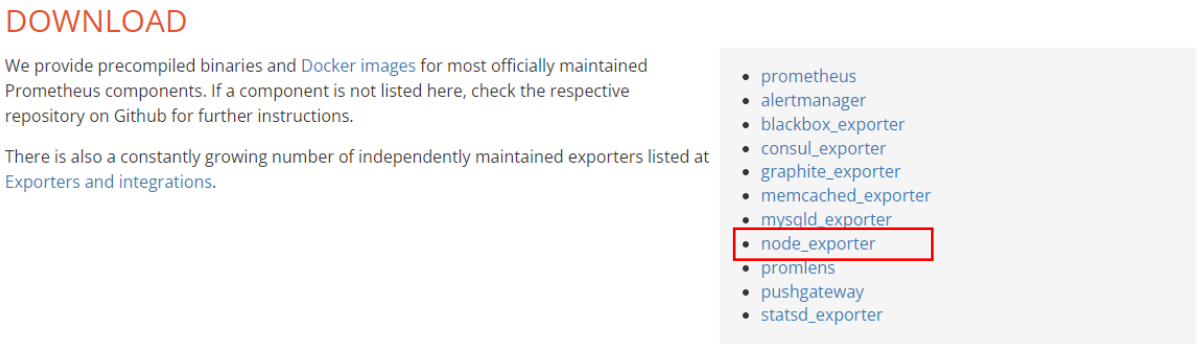


4. After that once your instance has been launched open its security group.
5. Then add the inbound rule for port 9100 from your IP address. Then click on save.



6. Now you need to connect to your instance via SSH and run the update and upgrade command.

   sudo apt-get update
   sudo apt-get upgrade
7. Then you need to open the Prometheus download page and there you will what services you can download. But you have to click on node_exporter.



8. You will be on a new page and you will see node_exporter there. Now you just need to right click on the link and choose copy link address.

9. Below you can see that we pasted the link with sudo wget command.

```
ubuntu@ip-172-31-20-58:~$ sudo wget https://github.com/prometheus/node_exporter/releases/download/v1.8.0/node_exporter-1.8.0.linux-amd64.tar.gz
--2024-05-03 14:50:53--  https://github.com/prometheus/node_exporter/releases/download/v1.8.0/node_exporter-1.8.0.linux-amd64.tar.gz
Resolving github.com (github.com)... 20.205.243.166
Connecting to github.com (github.com)|20.205.243.166|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/9524057/1db6b1dc-654d-4b8f-917f-9dcf28d6b3da?X-Amz-Algorithm=AWS4-HMAC
-SHA256&X-Amz-Credential=AKIAVCODYLSA53PQK4ZA%2F20240503%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20240503T145053Z&X-Amz-Expires=300&X-Amz-Signature=7bd6d8
cfd5b25afb610a90e26dd9a31ffc0375e8a1d5b816c41973011b6e514b&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=9524057&response-content-disposition=attachme
nt%3B%20filename%3Dnode_exporter-1.8.0.linux-amd64.tar.gz&response-content-type=application%2Foctet-stream [following]
--2024-05-03 14:50:53--  https://objects.githubusercontent.com/github-production-release-asset-2e65be/9524057/1db6b1dc-654d-4b8f-917f-9dcf28d6b3da?X-Amz-Algo
rithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAVCODYLSA53PQK4ZA%2F20240503%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20240503T145053Z&X-Amz-Expires=300&X-Amz-S
ignature=7bd6d8cfd5b25afb610a90e26dd9a31ffc0375e8a1d5b816c41973011b6e514b&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=9524057&response-content-dispo
sition=attachment%3B%20filename%3Dnode_exporter-1.8.0.linux-amd64.tar.gz&response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.110.133, 185.199.111.133, 185.199.108.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10675730 (10M) [application/octet-stream]
Saving to: 'node_exporter-1.8.0.linux-amd64.tar.gz'

node_exporter-1.8.0.linux-amd64.tar.gz              100%[=============================================================================================
================>]  10.18M  --.-KB/s    in 0.07s

2024-05-03 14:50:55 (153 MB/s) - 'node_exporter-1.8.0.linux-amd64.tar.gz' saved [10675730/10675730]

ubuntu@ip-172-31-20-58:~$
```

10. Now if you do a listing of objects then you will see your zipped file. Then you need to unzip that file using the below command.

    sudo tar -xvf

```
ubuntu@ip-172-31-20-58:~$ ls
node_exporter-1.8.0.linux-amd64.tar.gz
ubuntu@ip-172-31-20-58:~$ sudo tar -xvf node_exporter-1.8.0.linux-amd64.tar.gz
node_exporter-1.8.0.linux-amd64/
node_exporter-1.8.0.linux-amd64/NOTICE
node_exporter-1.8.0.linux-amd64/node_exporter
node_exporter-1.8.0.linux-amd64/LICENSE
ubuntu@ip-172-31-20-58:~$ ls
node_exporter-1.8.0.linux-amd64  node_exporter-1.8.0.linux-amd64.tar.gz
ubuntu@ip-172-31-20-58:~$
```

11. Now if we go inside of our unzipped file and list the objects then we can see our node exporter file.

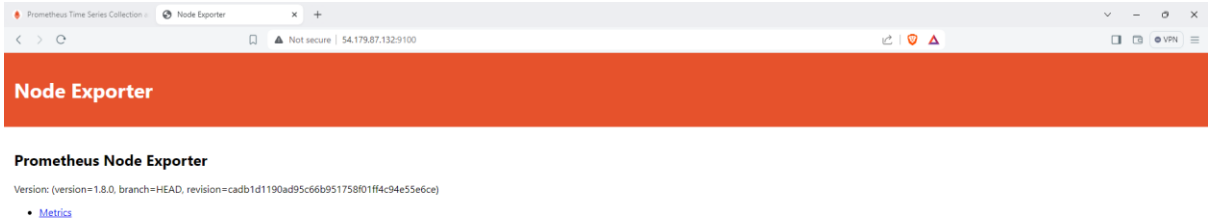
```
ubuntu@ip-172-31-20-58:~$ cd node_exporter-1.8.0.linux-amd64/
ubuntu@ip-172-31-20-58:~/node_exporter-1.8.0.linux-amd64$ ls
LICENSE  NOTICE  node_exporter
ubuntu@ip-172-31-20-58:~/node_exporter-1.8.0.linux-amd64$
```

12. Then if you run this command, you will see that we get a very long output and in the bottom of that you will see that it is listening on this address.

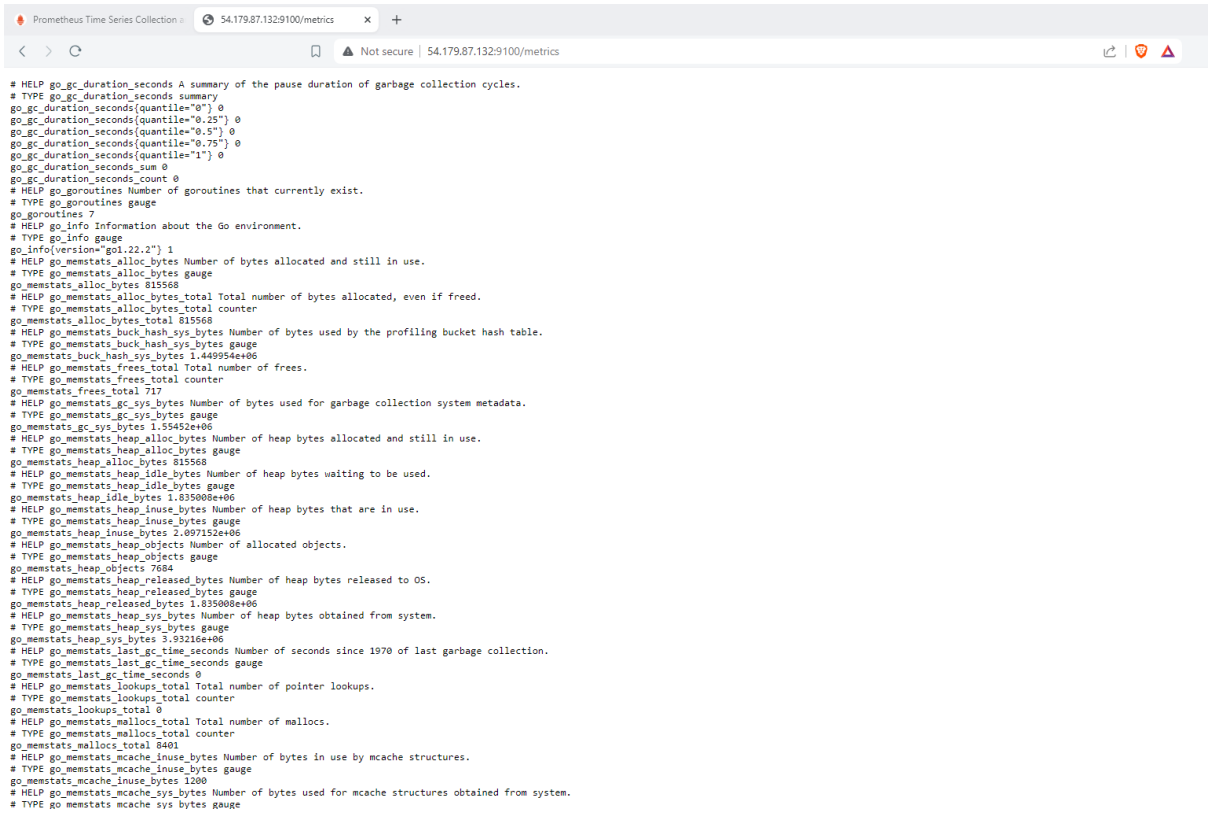
    ./node_exporter

```
ts=2024-05-03T14:55:44.251Z caller=node_exporter.go:118 level=info collector=zfs
ts=2024-05-03T14:55:44.251Z caller=tls_config.go:313 level=info msg="Listening on" address=[::]:9100
ts=2024-05-03T14:55:44.252Z caller=tls_config.go:316 level=info msg="TLS is disabled." http2=false address=[::]:9100
```

13. Now you need to go to your browser and copy the public IP address of your new instance which you created for node exporter. Then paste it into a new tab with port 9100 appended.
14. You will see a webpage like this, here click on the Metrics.



15. Here you can see the metrics.



16. Now we want to configure Prometheus so that it can scrape the metrics from the application server.
17. For that, we have to remotely connect to Prometheus server to this one, update the Prometheus config file so that it knows that we have set up node exporter on the application server, and then it starts scraping.
18. Now SSH into your Prometheus instance. From your home directory go to this location mentioned below and do the listing of objects.
19. Now we need to update the prometheus.yml file and every time we run an exporter we need to update this file.

    cd /etc
    cd prometheus/

```
ubuntu@ip-172-31-43-89:~$ cd /etc
ubuntu@ip-172-31-43-89:/etc$ cd prometheus/
ubuntu@ip-172-31-43-89:/etc/prometheus$ ls
files_sd  prometheus.yml  rules  rules.s
ubuntu@ip-172-31-43-89:/etc/prometheus$
```

20. Now to edit this file you need to use this command.

    sudo nano prometheus.yml
21. Once you are in the file you will see this type of code written in it.
22. Now you need to add a job name in scrape configs. You can keep everything the same but you need to change the IP address in the target. Choose the Private IP address of your application server instance.
23. Most important thing, this is a YAML file you need to take care of the indentations.
24. Once you have pasted the code in the file then you need to press CTRL + O to save the file after that CTRL + X to exit.

```
- job_name: "application_server"

  # metrics_path defaults to '/metrics'
  # scheme defaults to 'http'.

  static_configs:
   - targets: ["172.31.20.58:9100"]
```

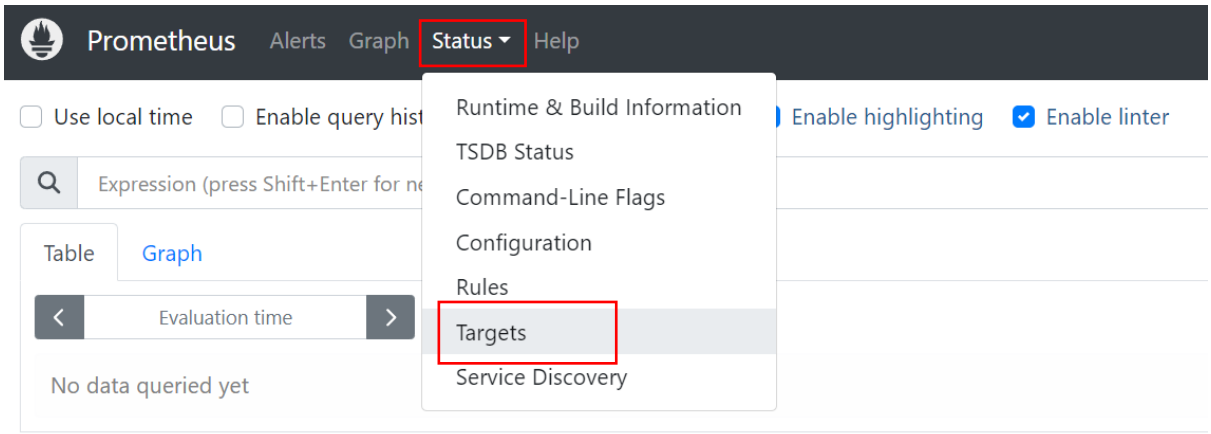
```
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

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

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:9090"]
  - job_name: "application_server"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["172.31.20.58:9100"]
```

25. Now you need to stop and restart the Prometheus service.

    sudo systemctl stop prometheus
    sudo systemctl start prometheus

```
ubuntu@ip-172-31-43-89:/etc/prometheus$ sudo systemctl stop prometheus
ubuntu@ip-172-31-43-89:/etc/prometheus$ sudo systemctl start prometheus
ubuntu@ip-172-31-43-89:/etc/prometheus$
```

26. Open your browser and access your Prometheus webpage then click on status can choose targets.



27. Then you will see that we have two targets up and running.
28. Every target in here is basically one of those entries in the script config in your Yaml file in the Prometheus config file. The first one is the one that Prometheus uses to scrape its own metrics, and it is up, obviously, because that exporter is on the same server as Prometheus, and the other one is the one that we added.