Sri Lanka Institute of Information Technology

Visual Analytics and User Experience Design – IT4031

Assignment 2

Group ID: 2022-VAUED-G8

Group Members:

Student Name	Registration Number
IT19083742	Vithana K.C.D
IT19064246	Wijesiri M.R.M
IT18228786	Weerarathne D.N.N
IT18037548	Krishan H.A.S

Table of Contents

1. Background

The main objective of this assignment is to develop a deployment to capture metrics from an application and visualize those by using the Grafana dashboards.

Application Metrics consist of many different metric types which can be used to monitor features like the number of requests, the number of user logins over time, the time it takes to perform a database query, CPU utilization, the amount of free RAM, and so on. Mainly application metrics can be classified into 4 types, they are gauges, counters, histograms, and summaries.

To develop this system, an AWS EC2 Ubuntu instance is used to carry out the initial steps of the assignment. An AWS EC2 instance is a virtual server in Amazon's Elastic Compute Cloud (EC2) for running applications on the Amazon Web Services (AWS) infrastructure. After creating the EC2 instance, Docker, Prometheus, Grafana servers and Node Exporter application are downloaded into a Ubuntu server for the use of further implementations.

Docker is a set of platforms as a service (PaaS) product that uses OS-level virtualization to deliver software in packages called containers. First docker is installed, and then an application which provides an API to fetch metrics information is required to deploy. For this assignment Node exporter, a third-party application is used to expose Prometheus metrics. The Prometheus Node Exporter has a wide variety of server metrics that captures and exports all the Ubuntu hardware and kernel-related metrics.

Prometheus is an event monitoring and alerting application. It uses an HTTP pull mechanism to capture real-time measurements in a time series database (allowing for high dimensionality) with configurable searches and real-time alerting. By scraping metrics HTTP endpoints, Prometheus gathers data from monitored targets. Metrics are a type of measurement for the system at a certain moment in time. After selecting the application, which is required to be monitored, the Prometheus server is deployed, and Node Exporter is included as the target to identify metrics types.

After completing these steps Grafana dashboard server is deployed, and the Prometheus server is included as the data source. Grafana is an open-source metric analytics & visualization suite. The purpose of Grafana dashboards is to bring data together in a way that is both efficient and organized. It allows users to better understand the metrics of their data through queries, informative visualizations. As the final step, a dashboard is created to visualize application metrics.

2. Problem

Understanding the state of a system is essential for ensuring the reliability and stability of the applications and services.

Monitoring is mainly concerned with observing the performance of an application over time. As a consequence, it gives crucial information and insights on the application's performance and usage patterns. This might include details on memory utilization and issues, as well as availability, request rates, bottlenecks, and more. Performance metrics can be used to create dashboards to troubleshoot issues or give a summary of the state of your applications and resources. A robust monitoring system that collects metrics and visualizes data is one of the best ways to acquire important insights.

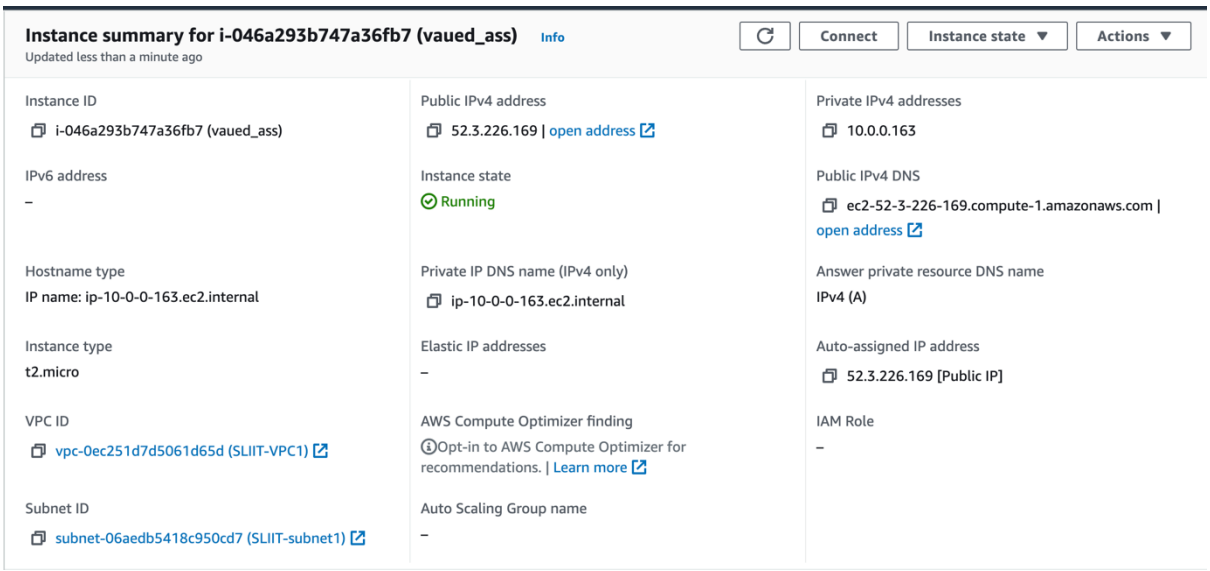
Therefore, it is a necessity to monitor performance of the applications or the services to find any bottlenecks or issues with the application.

3. Solution

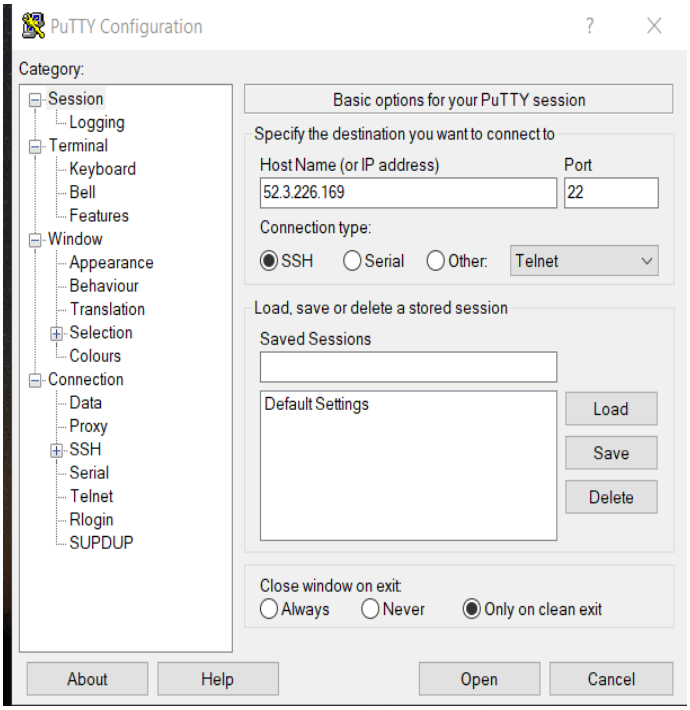
A robust monitoring system that collects metrics and visualizes data is deployed to acquire important insights and find any bottlenecks or issues with the application in real-time and improve operational efficiency of the application.

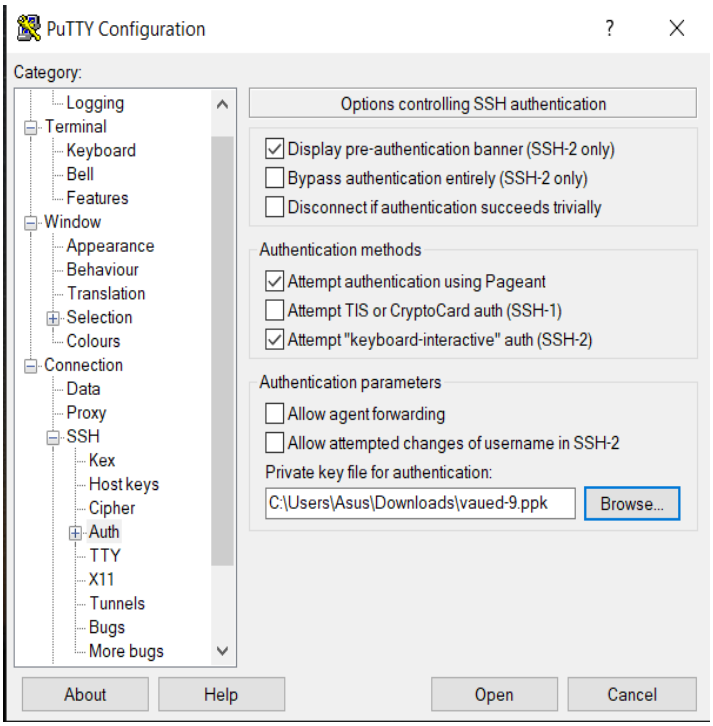
4. Process

1. Create an EC2 Ubuntu instance performing the necessary configurations



2. Connect to the EC2 instance via Putty





3. Update your machine using **sudo apt-get update**

```
ubuntu@ip-10-0-0-234: ~
login as: ubuntu
Authenticating with public key "imported-openssh-key"
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.15.0-1004-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sat May 21 08:51:04 UTC 2022

  System load:  0.05615234375     Processes:             103
  Usage of /:   19.0% of 7.58GB   Users logged in:       0
  Memory usage: 21%               IPv4 address for eth0: 10.0.0.234
  Swap usage:   0%

0 updates can be applied immediately.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-0-0-234:~$ sudo apt-get update
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease [270 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [10
9 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [
99.8 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packag
es [14.1 MB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-
en [5652 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f
Metadata [286 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Pack
ages [217 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translatio
n-en [112 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n
```

4. Install docker using **sudo apt-get install docker.io** command

```
root@ip-10-0-0-163: /home/ubuntu
Reading package lists... Done
root@ip-10-0-0-163:/home/ubuntu# sudo apt-get install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse
  | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 39 not upgraded.
Need to get 65.6 MB of archives.
After this operation, 283 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [6
3.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 bridge-utils amd64 1.7-
1ubuntu3 [34.4 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 runc amd64 1.1.0-0ubunt
u1 [4087 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 containerd amd64 1.5.9-
0ubuntu3 [27.0 MB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 dns-root-data all 20210
11101 [5256 B]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 dnsmasq-base am
d64 2.86-1.1ubuntu0.1 [354 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 docker.io amd64 20.
10.12-0ubuntu4 [34.0 MB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 ubuntu-fan all 0.12
.16 [35.2 kB]
Fetched 65.6 MB in 1s (57.9 MB/s)
Preconfiguring packages ...
Selecting previously unselected package pigz.
(Reading database ... 63599 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.6-1_amd64.deb ...
Unpacking pigz (2.6-1) ...
Selecting previously unselected package bridge-utils.
Preparing to unpack .../1-bridge-utils_1.7-1ubuntu3_amd64.deb ...
Unpacking bridge-utils (1.7-1ubuntu3) ...
Selecting previously unselected package runc.
Preparing to unpack .../2-runc_1.1.0-0ubuntu1_amd64.deb ...
Unpacking runc (1.1.0-0ubuntu1) ...
Selecting previously unselected package containerd.
Preparing to unpack .../3-containerd_1.5.9-0ubuntu3_amd64.deb ...
Unpacking containerd (1.5.9-0ubuntu3) ...
Selecting previously unselected package dns-root-data.
Preparing to unpack .../4-dns-root-data_2021011101_all.deb ...
Unpacking dns-root-data (2021011101) ...
```

5. Create the **prometheus.yml** configuration file in the root directory

```
root@ip-10-0-0-163:/home/ubuntu# sudo vim /root/prometheus.yml
```

6. Edit the /root/Prometheus.yml and add the two targets (localhost, public IP of Ubuntu instance) for the Prometheus server

```
root@ip-10-0-0-163: /home/ubuntu
global:
  scrape_interval: 15s
  evaluation_interval: 15s

scrape_configs:
  - job_name: 'prometheus'

    static_configs:
      - targets: ['127.0.0.1:9090', '52.3.226.169:9090']
        labels:
          group: 'prometheus'
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                          11,30          All
```

7. Launch Prometheus container - The image to be pulled is prom/prometheus from docker.hub and the Prometheus server will run on port number 9090, that binds to the Ubuntu machine port 9090

```
root@ip-10-0-0-163:/home/ubuntu# docker run -p 9090:9090 -v /root/prometheus.yml:/etc/prometheus/prometheus.yml prom/prometheus
ts=2022-05-21T10:11:15.793Z caller=main.go:488 level=info msg="No time or size retention was set so using the default time retention" duration=15d
ts=2022-05-21T10:11:15.793Z caller=main.go:525 level=info msg="Starting Prometheus" version="(version=2.35.0, branch=HEAD, revision=6656cd29fe6ac92bab91ecec0fe162ef0f187654)"
ts=2022-05-21T10:11:15.794Z caller=main.go:530 level=info build_context="(go=go1.18.1, user=root@cf6852b14d68, date=20220421-09:53:42)"
ts=2022-05-21T10:11:15.794Z caller=main.go:531 level=info host_details="(Linux 5.15.0-1004-aws #6-Ubuntu SMP Thu Mar 31 09:44:20 UTC 2022 x86_64 8cd4c560077e (none))"
ts=2022-05-21T10:11:15.794Z caller=main.go:532 level=info fd_limits="(soft=1048576, hard=1048576)"
ts=2022-05-21T10:11:15.794Z caller=main.go:533 level=info vm_limits="(soft=unlimited, hard=unlimited)"
ts=2022-05-21T10:11:15.801Z caller=web.go:541 level=info component=web msg="Start listening for connections" address=0.0.0.0:9090
ts=2022-05-21T10:11:15.803Z caller=main.go:957 level=info msg="Starting TSDB ..."
ts=2022-05-21T10:11:15.814Z caller=head.go:493 level=info component=tsdb msg="Replaying on-disk memory mappable chunks if any"
ts=2022-05-21T10:11:15.814Z caller=head.go:536 level=info component=tsdb msg="On-disk memory mappable chunks replay completed" duration=5.605µs
ts=2022-05-21T10:11:15.814Z caller=head.go:542 level=info component=tsdb msg="Replaying WAL, this may take a while"
ts=2022-05-21T10:11:15.822Z caller=tls_config.go:195 level=info component=web msg="TLS is disabled." http2=false
ts=2022-05-21T10:11:15.822Z caller=head.go:613 level=info component=tsdb msg="WAL segment loaded" segment=0 maxSegment=0
ts=2022-05-21T10:11:15.823Z caller=head.go:619 level=info component=tsdb msg="WAL replay completed" checkpoint_replay_duration=68.889µs wal_replay_duration=8.105702ms total_replay_duration=
8.668107ms
ts=2022-05-21T10:11:15.826Z caller=main.go:978 level=info fs_type=EXT4_SUPER_MAGIC
ts=2022-05-21T10:11:15.826Z caller=main.go:981 level=info msg="TSDB started"
ts=2022-05-21T10:11:15.826Z caller=main.go:1162 level=info msg="Loading configuration file" filename=/etc/prometheus/prometheus.yml
ts=2022-05-21T10:11:15.832Z caller=main.go:1199 level=info msg="Completed loading of configuration file" filename=/etc/prometheus/prometheus.yml totalDuration=5.526031ms db_storage=40.87µs
remote_storage=2.617µs web_handler=1.041µs query_engine=1.411µs scrape=4.821177ms scrape_sd=22.375µs notify=1.498µs notify_sd=2.324µs rules=1.758µs tracing=6.68µs
ts=2022-05-21T10:11:15.832Z caller=main.go:930 level=info msg="Server is ready to receive web requests."
```

8. Configure the security groups of the EC2 instance

▼ Inbound rules

Filter rules

Security group rule ID	Port range	Protocol	Source	Security groups
sgr-0387d1af3c18c4b23	22	TCP	0.0.0.0/0	launch-wizard-13
sgr-0bcdf34b79dad36a5	80	TCP	::/0	launch-wizard-13
sgr-0e714f7d5503f9daf	9100	TCP	0.0.0.0/0	launch-wizard-13
sgr-0b66bbf7800b3e482	9090	TCP	0.0.0.0/0	launch-wizard-13
sgr-06f391d978a15e22c	3000	TCP	0.0.0.0/0	launch-wizard-13
sgr-0dc28273c2e0f773e	80	TCP	0.0.0.0/0	launch-wizard-13

▼ Outbound rules

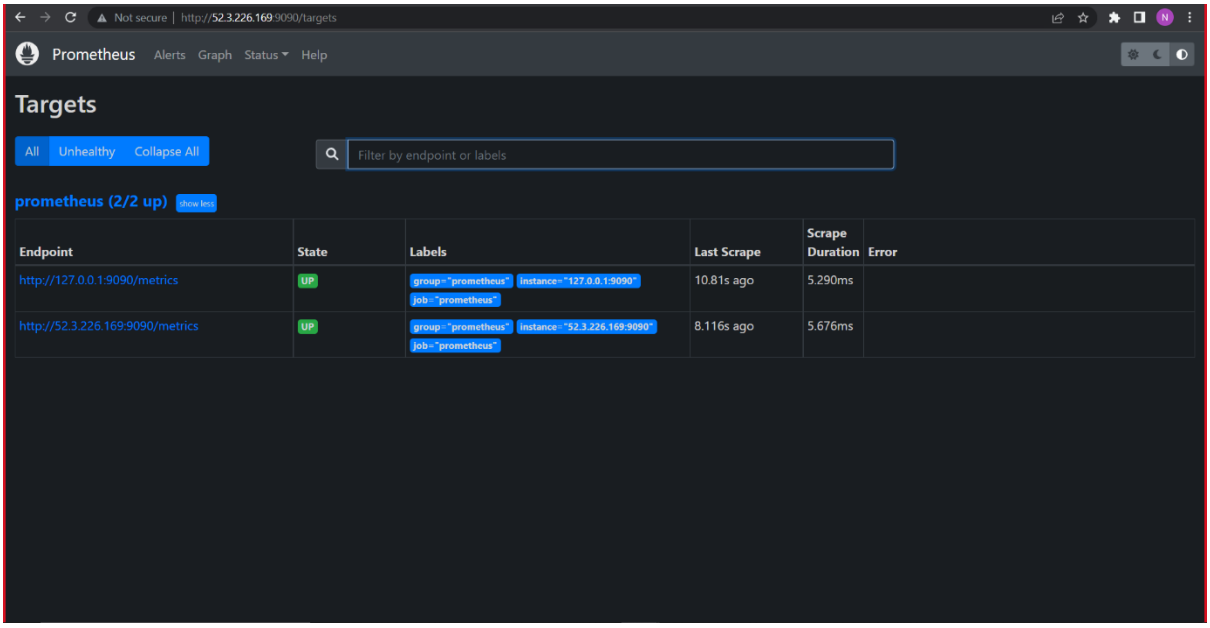
Filter rules

Security group rule ID	Port range	Protocol	Destination	Security groups
sgr-06074c3da3b44332b	All	All	0.0.0.0/0	launch-wizard-13

9. Set the Node Exporter

```
root@ip-10-0-0-163: /home/ubuntu
login as: ubuntu
Authenticating with public key "imported-openssh-key"
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.15.0-1004-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sat May 21 10:16:24 UTC 2022

  System load:  0.0               Processes:               112
  Usage of /:   28.1% of 7.58GB   Users logged in:         1
  Memory usage: 33%               IPv4 address for docker0: 172.17.0.1
  Swap usage:   0%                IPv4 address for eth0:    10.0.0.163

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

43 updates can be applied immediately.
30 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable


Last login: Sat May 21 09:30:08 2022 from 8.38.147.32
ubuntu@ip-10-0-0-163:~$ sudo su
root@ip-10-0-0-163:/home/ubuntu# vi root/prometheus.yml
root@ip-10-0-0-163:/home/ubuntu# vi/ root/prometheus.yml
bash: vi/: No such file or directory
root@ip-10-0-0-163:/home/ubuntu# vi /root/prometheus.yml
root@ip-10-0-0-163:/home/ubuntu# docker run -d -v "/proc:/host/proc" -v "/sys:/h
ost/sys" -v "/:/rootfs" --net="host" --name=promnode quay.io/prometheus/node-exp
orter:v0.13.0 -collector.procfs /host/proc -collector.sysfs /host/sys -collector
.filesystem.ignored-mount-points "^/(sys|proc|dev|host|etc) ($|/)"
Unable to find image 'quay.io/prometheus/node-exporter:v0.13.0' locally
v0.13.0: Pulling from prometheus/node-exporter
8ddc19f16526: Pull complete
a3ed95caeb02: Pull complete
8279f336cdd3: Pull complete
81998f54d5a6: Pull complete
Digest: sha256:8f083b308a39bdd5bd42759b67e20e19398e98b291648f397d98ef4b99d9a318
Status: Downloaded newer image for quay.io/prometheus/node-exporter:v0.13.0
87fe1d6f4e2c59caf5684bb53dcd1a53cea21486bfdfb303de39fa1bfafff63f
root@ip-10-0-0-163:/home/ubuntu#
```

10. Launch the Prometheus dashboard



11. Create the **docker-compose.yml** file that defines the Prometheus and node-exporter service and monitoring in the root directory

root@ip-10-0-0-163: /home/ubuntu

```yaml
version: '3.8'

networks:
  monitoring:
    driver: bridge

volumes:
  prometheus_data: {}

services:
  node-exporter:
    image: prom/node-exporter:latest
    container_name: node-exporter
    restart: unless-stopped
    volumes:
      - /proc:/host/proc:ro
      - /sys:/host/sys:ro
      - /:/rootfs:ro
    command:
      - '--path.procfs=/host/proc'
      - '--path.rootfs=/rootfs'
      - '--path.sysfs=/host/sys'
      - '--collector.filesystem.mount-points-exclude=^/(sys|proc|dev|host|etc)($$|/)'
    expose:
      - 9100
    networks:
      - monitoring

  prometheus:
    image: prom/prometheus:latest
    container_name: prometheus
    restart: unless-stopped
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
      - prometheus_data:/prometheus
    command:
      - '--config.file=/etc/prometheus/prometheus.yml'
      - '--storage.tsdb.path=/prometheus'
      - '--web.console.libraries=/etc/prometheus/console_libraries'
      - '--web.console.templates=/etc/prometheus/consoles'
      - '--web.enable-lifecycle'
    expose:
      - 9090
    networks:
      - monitoring
~
~
~
~
"~/docker-compose.yml" 45L, 1108B
```

12. Update the /root/prometheus.yml file

```
root@ip-10-0-0-163: /home/ubuntu
global:
  scrape_interval: 1m

scrape_configs:
  - job_name: "prometheus"
    scrape_interval: 1m
    static_configs:
      - targets: ["localhost:9090", "52.3.226.169:9090"]

  - job_name: "node"
    static_configs:
      - targets: ["node-exporter:9100", "52.3.226.169:9100"]

remote_write:
  - url: "https://prometheus-prod-10-prod-us-central-0.grafana.net/api/prom/push"
    basic_auth:
      username: "437352"
      password: "eyJrIjoiM2Q2YzEyNWFhYzVlOTY0ZTY5ODhmYjgwZWM4Zjk2ODI3M2FmODM1NCIsIm4iOiJ2YXVlZF91c2VyIiwiaWQiOjY1MDY1Nn0="
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"~/prometheus.yml" 18L, 539B
```

13. Start the Prometheus and node-exporter containers using **docker-compose up -d** command

```
root@ip-10-0-0-163:/home/ubuntu# cd /root
root@ip-10-0-0-163:~# ls
docker-compose.yml  prometheus.yml  snap
root@ip-10-0-0-163:~# docker-compose up -d
Creating network "root_monitoring" with driver "bridge"
Creating volume "root_prometheus_data" with default driver
Pulling node-exporter (prom/node-exporter:latest)...
latest: Pulling from prom/node-exporter
aa2a8d90b84c: Pull complete
b45d31ee2d7f: Pull complete
b5db1e299295: Pull complete
Digest: sha256:f2269e73124dd0f60a7d19a2ce1264d33d08a985aed0ee6b0b89d0be470592cd
Status: Downloaded newer image for prom/node-exporter:latest
Creating prometheus    ... done
Creating node-exporter ... done
root@ip-10-0-0-163:~#
```

14. Check docker status using **docker-compose ps**

```
root@ip-10-0-0-163:~# docker-compose ps
    Name                   Command               State    Ports
-----------------------------------------------------------------
node-exporter   /bin/node_exporter --path. ...   Up      9100/tcp
prometheus      /bin/prometheus --config.f ...   Up      9090/tcp
root@ip-10-0-0-163:~# docker-compose logs -f prometheus
Attaching to prometheus
prometheus       | ts=2022-05-21T10:39:17.294Z caller=main.go:488 level=info msg="No time or
size retention was set so using the default time retention" duration=15d
prometheus       | ts=2022-05-21T10:39:17.294Z caller=main.go:525 level=info msg="Starting Pr
ometheus" version="(version=2.35.0, branch=HEAD, revision=6656cd29fe6ac92bab91ecec0fe162ef0f1
87654)"
prometheus       | ts=2022-05-21T10:39:17.294Z caller=main.go:530 level=info build_context="(
go=go1.18.1, user=root@cf6852b14d68, date=20220421-09:53:42)"
prometheus       | ts=2022-05-21T10:39:17.295Z caller=main.go:531 level=info host_details="(L
inux 5.15.0-1004-aws #6-Ubuntu SMP Thu Mar 31 09:44:20 UTC 2022 x86_64 881343aceel5 (none))"
prometheus       | ts=2022-05-21T10:39:17.295Z caller=main.go:532 level=info fd_limits="(soft
=1048576, hard=1048576)"
prometheus       | ts=2022-05-21T10:39:17.295Z caller=main.go:533 level=info vm_limits="(soft
=unlimited, hard=unlimited)"
prometheus       | ts=2022-05-21T10:39:17.297Z caller=web.go:541 level=info component=web msg
="Start listening for connections" address=0.0.0.0:9090
prometheus       | ts=2022-05-21T10:39:17.298Z caller=main.go:957 level=info msg="Starting TS
DB ..."
prometheus       | ts=2022-05-21T10:39:17.302Z caller=head.go:493 level=info component=tsdb m
sg="Replaying on-disk memory mappable chunks if any"
prometheus       | ts=2022-05-21T10:39:17.302Z caller=head.go:536 level=info component=tsdb m
sg="On-disk memory mappable chunks replay completed" duration=4.481µs
```

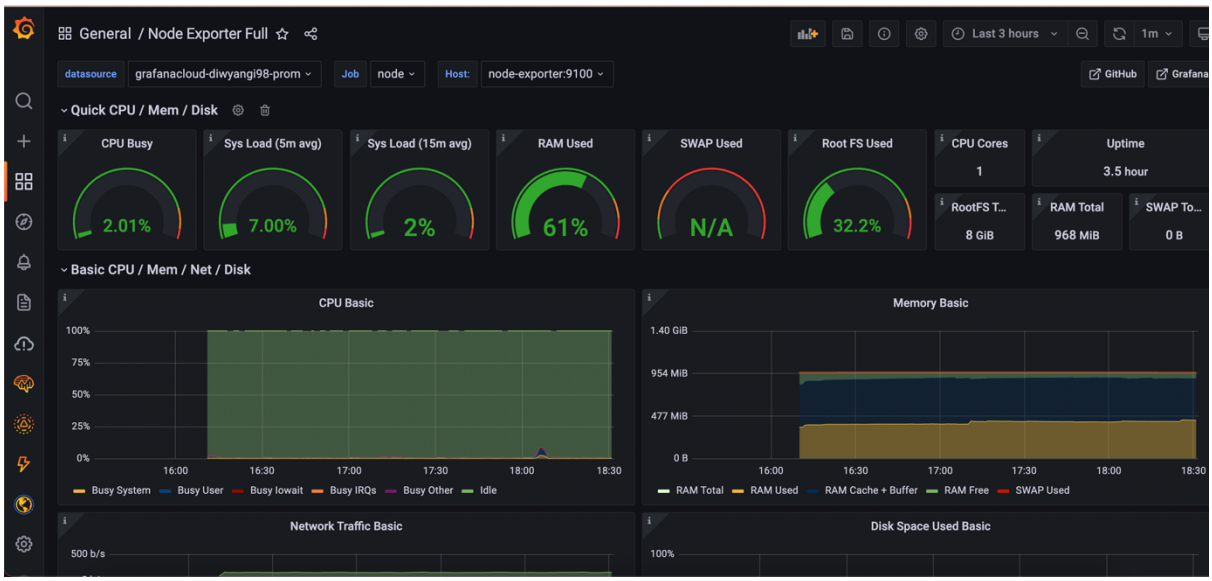
15. Check status of Prometheus by checking its logs using **docker-compose logs -f Prometheus**

```
root@ip-10-0-0-163:~# docker-compose logs -f prometheus
Attaching to prometheus
prometheus       | ts=2022-05-21T10:39:17.294Z caller=main.go:488 level=info msg="No time or
size retention was set so using the default time retention" duration=15d
prometheus       | ts=2022-05-21T10:39:17.294Z caller=main.go:525 level=info msg="Starting Pr
ometheus" version="(version=2.35.0, branch=HEAD, revision=6656cd29fe6ac92bab91ecec0fe162ef0f1
87654)"
prometheus       | ts=2022-05-21T10:39:17.294Z caller=main.go:530 level=info build_context="(
go=go1.18.1, user=root@cf6852b14d68, date=20220421-09:53:42)"
prometheus       | ts=2022-05-21T10:39:17.295Z caller=main.go:531 level=info host_details="(L
inux 5.15.0-1004-aws #6-Ubuntu SMP Thu Mar 31 09:44:20 UTC 2022 x86_64 881343acee15 (none))"
prometheus       | ts=2022-05-21T10:39:17.295Z caller=main.go:532 level=info fd_limits="(soft
=1048576, hard=1048576)"
prometheus       | ts=2022-05-21T10:39:17.295Z caller=main.go:533 level=info vm_limits="(soft
=unlimited, hard=unlimited)"
prometheus       | ts=2022-05-21T10:39:17.297Z caller=web.go:541 level=info component=web msg
="Start listening for connections" address=0.0.0.0:9090
prometheus       | ts=2022-05-21T10:39:17.298Z caller=main.go:957 level=info msg="Starting TS
DB ..."
prometheus       | ts=2022-05-21T10:39:17.302Z caller=head.go:493 level=info component=tsdb m
sg="Replaying on-disk memory mappable chunks if any"
prometheus       | ts=2022-05-21T10:39:17.302Z caller=head.go:536 level=info component=tsdb m
sg="On-disk memory mappable chunks replay completed" duration=4.481µs
```

16. Confirm the status of node-exporter by checking the logs using **docker-compose logs -f node-exporter**

```
root@ip-10-0-0-163: ~

Last login: Sat May 21 10:16:26 2022 from 8.38.147.32
ubuntu@ip-10-0-0-163:~$ ls
ubuntu@ip-10-0-0-163:~$ sudo su
root@ip-10-0-0-163:/home/ubuntu# cd /root
root@ip-10-0-0-163:~# docker-compose logs -f node-exporter
Attaching to node-exporter
node-exporter    | ts=2022-05-21T10:39:17.133Z caller=node_exporter.go:182 level=info msg="St
arting node_exporter" version="(version=1.3.1, branch=HEAD, revision=a2321e7b940ddcff26873612
bccdf7cd4c42b6b6)"
node-exporter    | ts=2022-05-21T10:39:17.133Z caller=node_exporter.go:183 level=info msg="Bu
ild context" build_context="(go=go1.17.3, user=root@243aafa5525c, date=20211205-11:09:49)"
node-exporter    | ts=2022-05-21T10:39:17.134Z caller=filesystem_common.go:111 level=info col
lector=filesystem msg="Parsed flag --collector.filesystem.mount-points-exclude" flag=^/(sys|p
roc|dev|host|etc)($|/)
node-exporter    | ts=2022-05-21T10:39:17.134Z caller=filesystem_common.go:113 level=info col
lector=filesystem msg="Parsed flag --collector.filesystem.fs-types-exclude" flag=^(autofs|bin
fmt_misc|bpf|cgroup2?|configfs|debugfs|devpts|devtmpfs|fusectl|hugetlbfs|iso9660|mqueue|nsfs|
overlay|proc|procfs|pstore|rpc_pipefs|securityfs|selinuxfs|squashfs|sysfs|tracefs)$
node-exporter    | ts=2022-05-21T10:39:17.136Z caller=node_exporter.go:108 level=info msg="En
abled collectors"
node-exporter    | ts=2022-05-21T10:39:17.136Z caller=node_exporter.go:115 level=info collect
or=arp
node-exporter    | ts=2022-05-21T10:39:17.136Z caller=node_exporter.go:115 level=info collect
or=bcache
node-exporter    | ts=2022-05-21T10:39:17.136Z caller=node_exporter.go:115 level=info collect
or=bonding
node-exporter    | ts=2022-05-21T10:39:17.136Z caller=node_exporter.go:115 level=info collect
or=btrfs
node-exporter    | ts=2022-05-21T10:39:17.136Z caller=node_exporter.go:115 level=info collect
or=conntrack
node-exporter    | ts=2022-05-21T10:39:17.136Z caller=node_exporter.go:115 level=info collect
or=cpu
node-exporter    | ts=2022-05-21T10:39:17.136Z caller=node_exporter.go:115 level=info collect
or=cpufreq
node-exporter    | ts=2022-05-21T10:39:17.136Z caller=node_exporter.go:115 level=info collect
or=diskstats
node-exporter    | ts=2022-05-21T10:39:17.136Z caller=node_exporter.go:115 level=info collect
or=dmi
node-exporter    | ts=2022-05-21T10:39:17.136Z caller=node_exporter.go:115 level=info collect
or=edac
node-exporter    | ts=2022-05-21T10:39:17.136Z caller=node_exporter.go:115 level=info collect
or=entropy
node-exporter    | ts=2022-05-21T10:39:17.136Z caller=node_exporter.go:115 level=info collect
or=fibrechannel
node-exporter    | ts=2022-05-21T10:39:17.136Z caller=node_exporter.go:115 level=info collect
or=filefd
node-exporter    | ts=2022-05-21T10:39:17.136Z caller=node_exporter.go:115 level=info collect
or=filesystem
```

17. Select appropriate metrics and verify the metrics by checking the Grafana dashboard



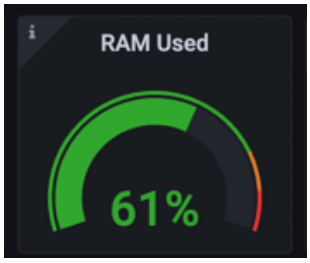
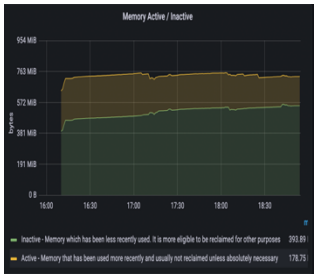
5. Results

Memory was identified as important feature through research

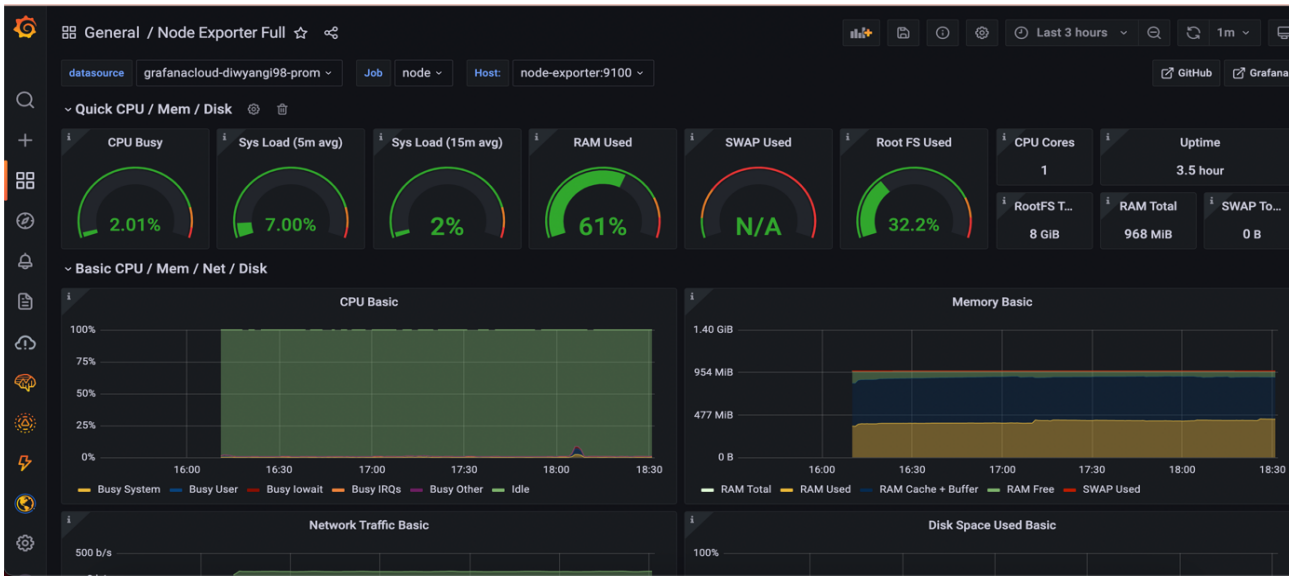
Metrics Identified

- Ram Used
- Swap Used
- Memory Pages In/ Out
- Memory Pages Swap In/Out
- Memory Active/ Inactive Stats
- Memory Stack

Used Metric Types

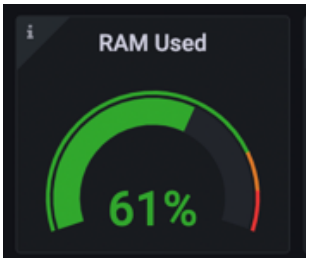
Gauge	Counter	Histogram
Metrics whose values are subject to change throughout time 	Metrics whose value increases over time 	It entails counting and adding samples. 

Full Dashboard (Screenshot)



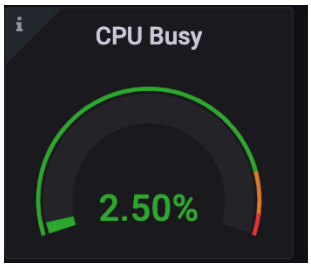
Metrics in Detail

RAM Used



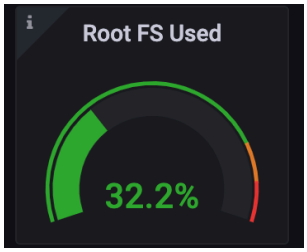
- node_memory_MemTotal_bytes
- node_memory_MemFree_bytes
- node_memory_MemAvailable_bytes

CPU Busy



- node_cpu_seconds_total

Root FS Used

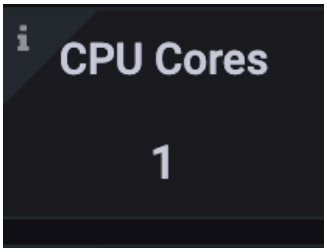


- node_filesystem_avail_bytes

RAM Total

- node_memory_MemTotal_bytes

CPU Cores

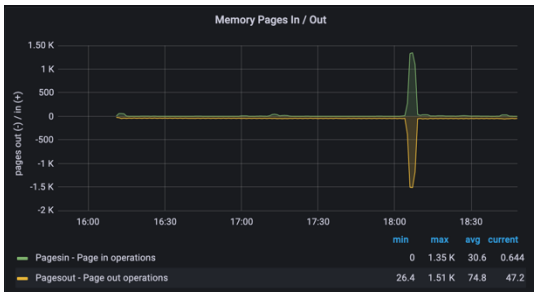


- node_cpu_seconds_total

Root FS Used

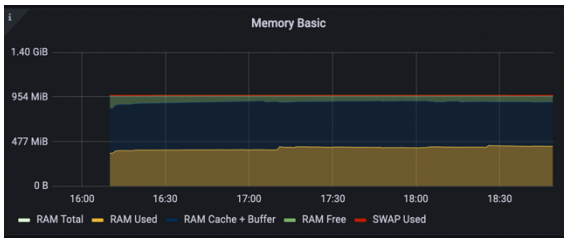
- node_filesystem_size_bytes

Memory Pages In / Out



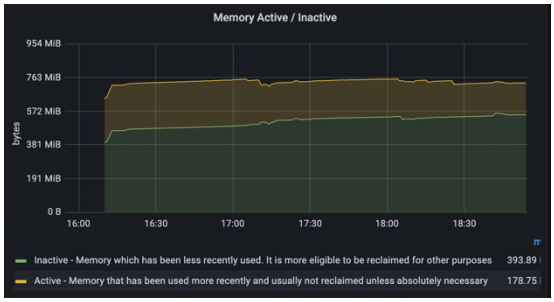
- node_vmstat_pgpgin
- node_vmstat_pgpgout

Memory Basic



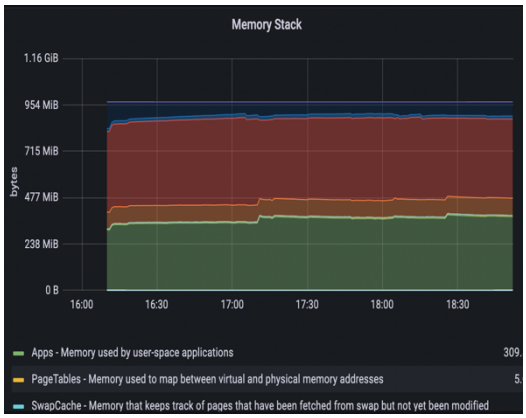
- node_memory_MemTotal_bytes
- node_memory_MemFree_bytes
- node_memory_Cached_bytes
- node_memory_Buffers_bytes
- node_memory_SwapTotal_bytes
- node_memory_SwapFree_bytes

Memory Active / Inactive Detail



- node_memory_Inactive_file_bytes
- node_memory_Inactive_anon_bytes
- node_memory_Active_file_bytes
- node_memory_Active_anon_bytes

Memory Stack



- node_memory_MemTotal_bytes
- node_memory_MemFree_bytes
- node_memory_Buffers_bytes
- node_memory_Cached_bytes
- node_memory_Slab_bytes
- node_memory_PageTables_bytes
- node_memory_SwapCached_bytes
- node_memory_SwapTotal_bytes
- node_memory_SwapFree_bytes
- node_memory_HardwareCorrupted_bytes

6. Work Breakdown

Student Name	Registration Number	Contribution
IT19083742	Vithana K.C.D	Prometheus Installation and Manage Prometheus UI
IT19064246	Wijesiri M.R.M	Grafana Dashboard Development
IT18228786	Weerarathne D.N.N	EC2 Instance Implementation And Docker Installation into instance
IT18037548	Krishan H.A.S	Node-Exporter installation and Docker Setup for Prometheus, Grafana and Exporter

7. References

[1] A. Srivastava, "How install and Configure Prometheus in a Docker Container on AWS EC2 instance.," 28 05 2020. [Online]. Available: https://medium.com/devtorq/how-install-and-configure-prometheus-in-a-docker-container-on-aws-ec2-instance-84d1e35cc8b0. [Accessed 10 05 2022].

[2] G. Labs, "Monitoring a Linux host with Prometheus, Node Exporter, and Docker Compose," [Online]. Available: https://grafana.com/docs/grafana-cloud/quickstart/docker-compose-linux/. [Accessed 13 05 2022].