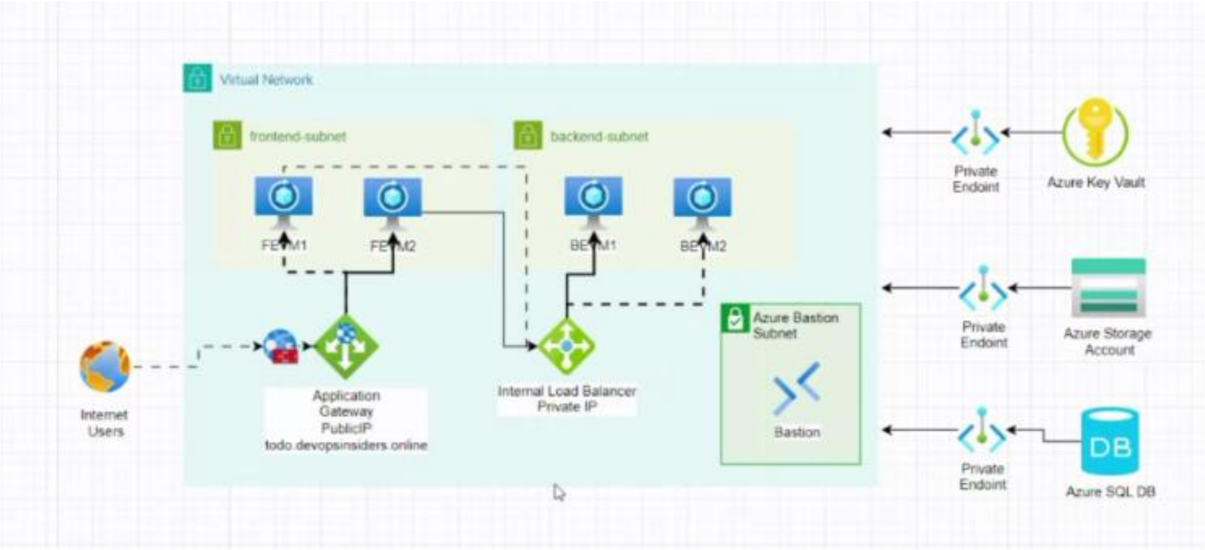
PROMETHEUS AND GRAFANA

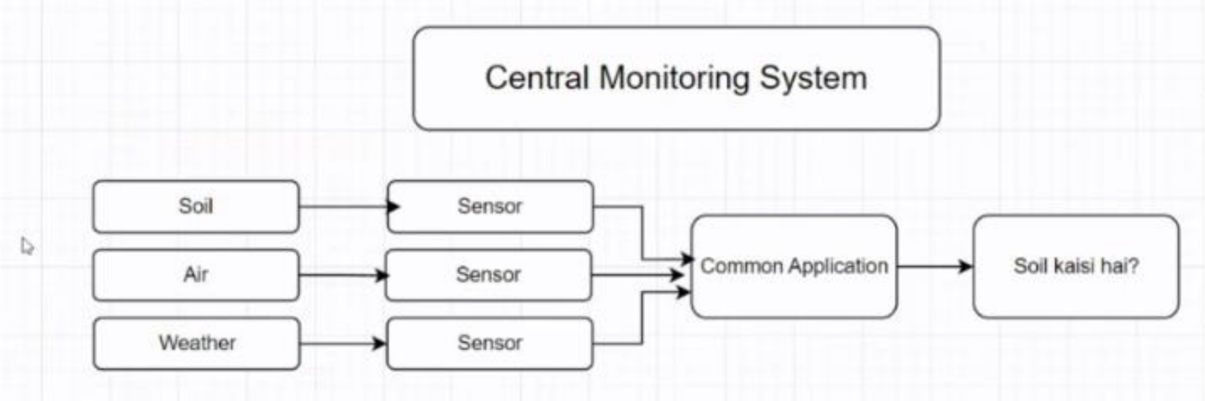
1)

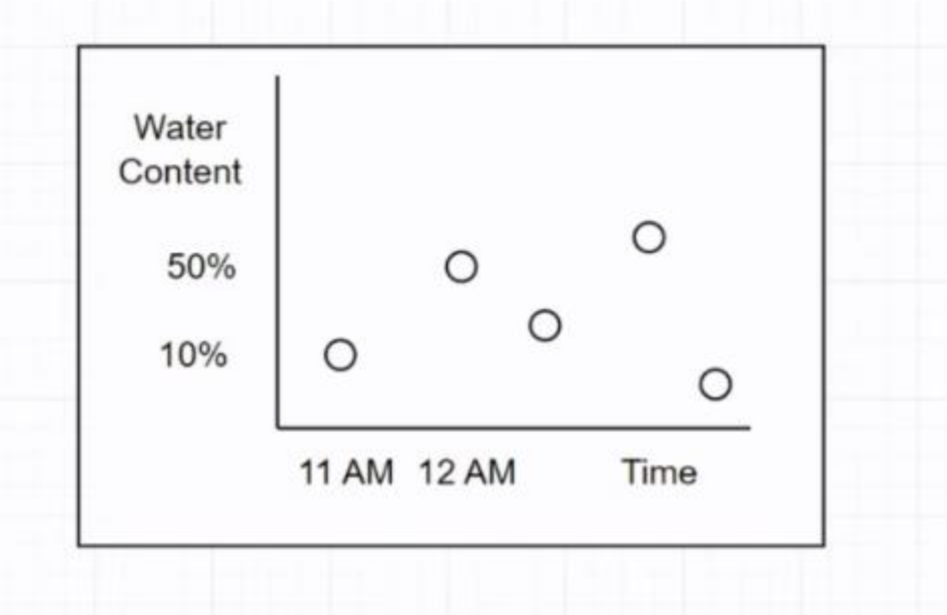


2) Internal loadbalancer is that which has private IP assigned to it.

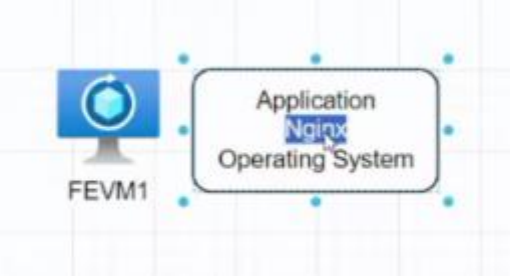
3) Frontend VMs will not directly communicate to backend VMs, they actually always communicate via Internal Load balancer, which is fitted in backend subnet.

4)

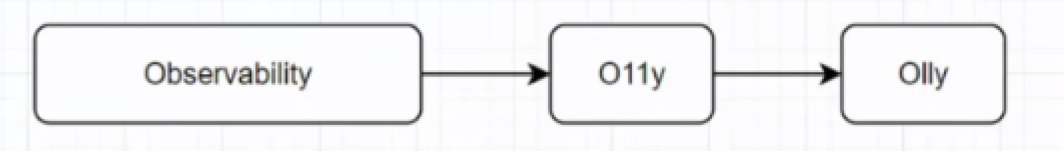




5)

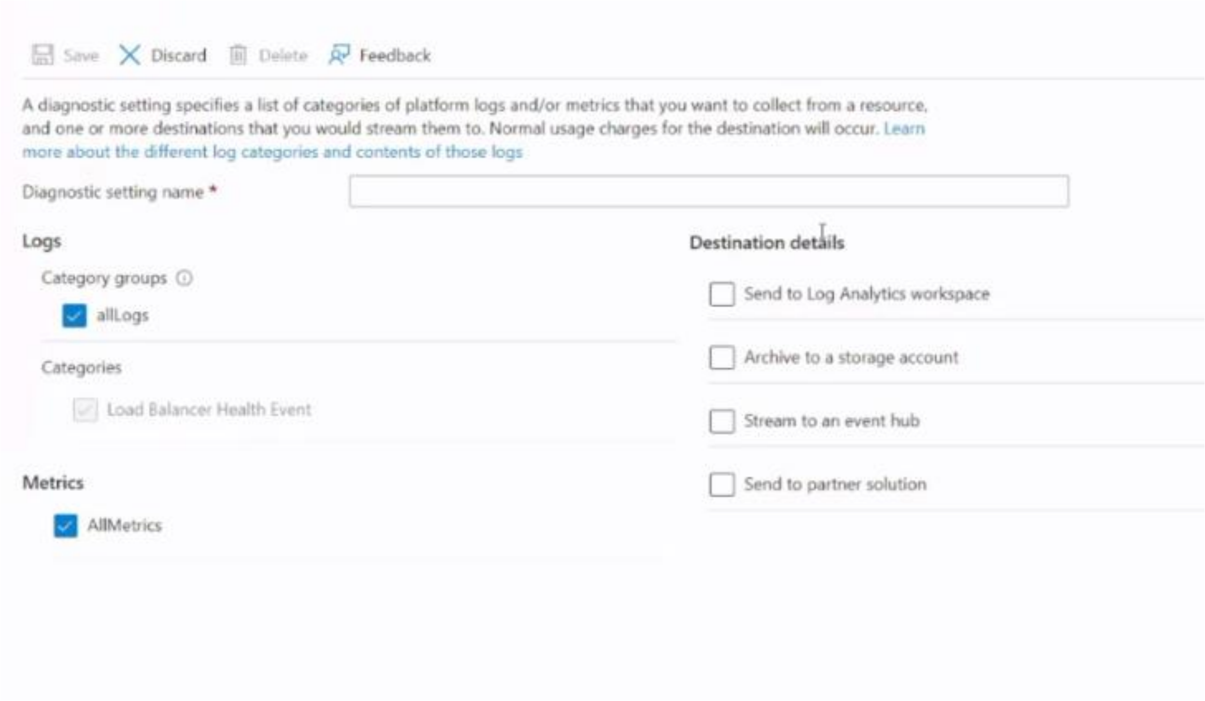


6)



7) To monitor logs of every component in azure, we have “diagnostic settings” option

8) We can send our logs to different places as shown below



A diagnostic setting specifies a list of categories of platform logs and/or metrics that you want to collect from a resource, and one or more destinations that you would stream them to. Normal usage charges for the destination will occur. Learn more about the different log categories and contents of those logs

Diagnostic setting name *

Logs

Category groups

☑ allLogs

Categories

☑ Load Balancer Health Event

Metrics

☑ AllMetrics

Destination details

□ Send to Log Analytics workspace

□ Archive to a storage account

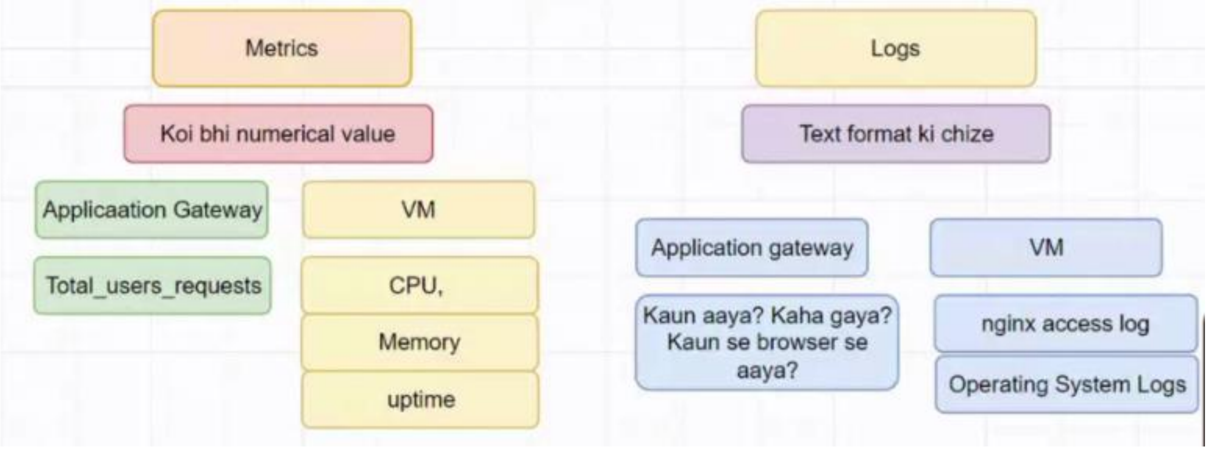
□ Stream to an event hub

□ Send to partner solution

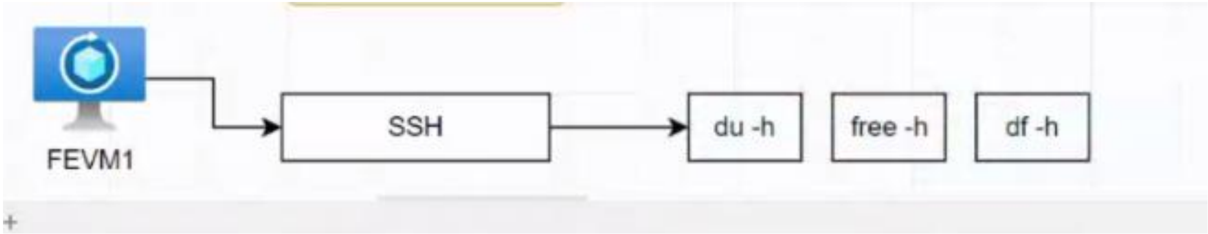
9) We monitors

Metrics
Logs

10)

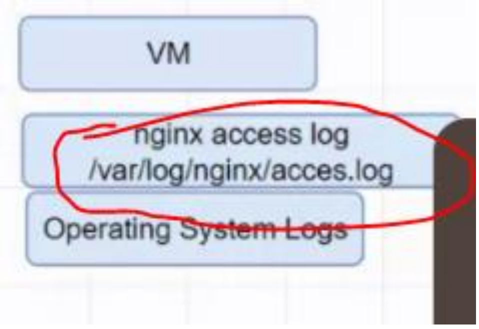


11) Now we are looking for VM monitoring



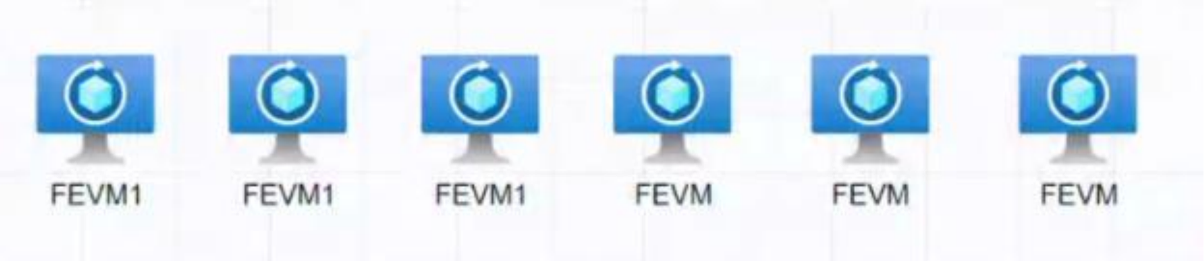
```
du -h
free -h
df -h
```

12) The log is made inside file

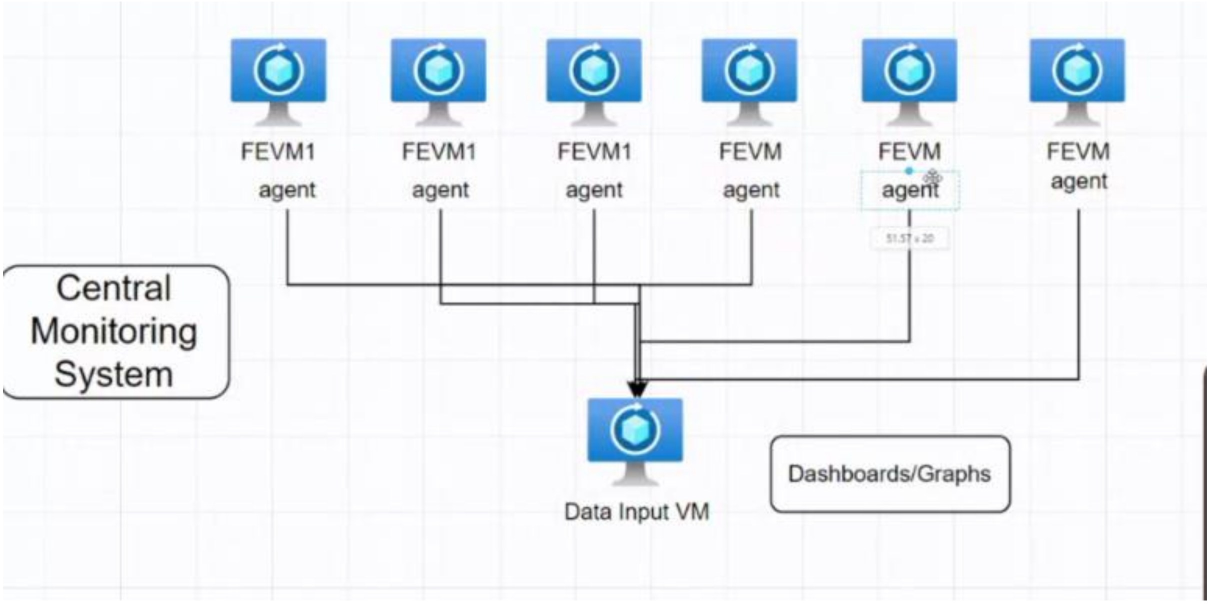


```
cat access.log
```

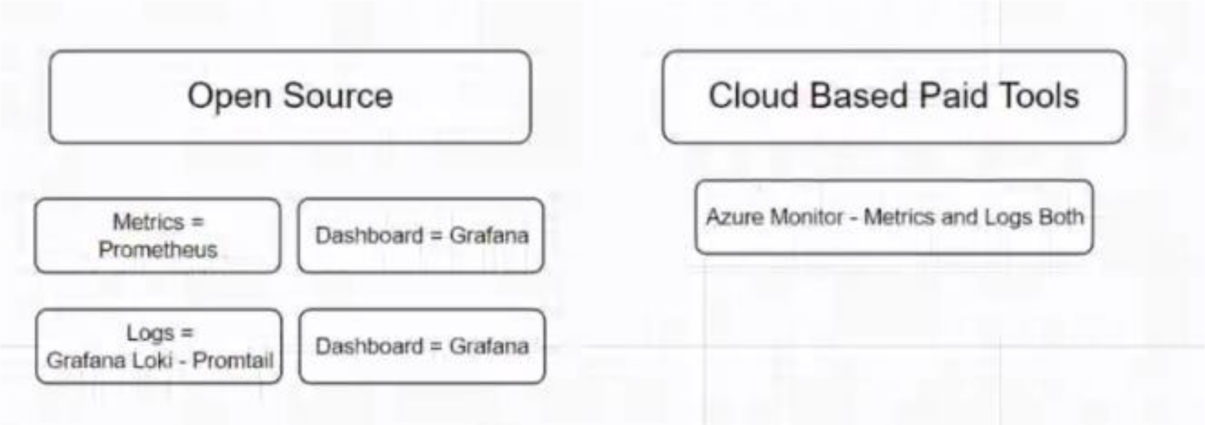
13)



14)



15)



Open Source

Metrics = Prometheus

Dashboard = Grafana

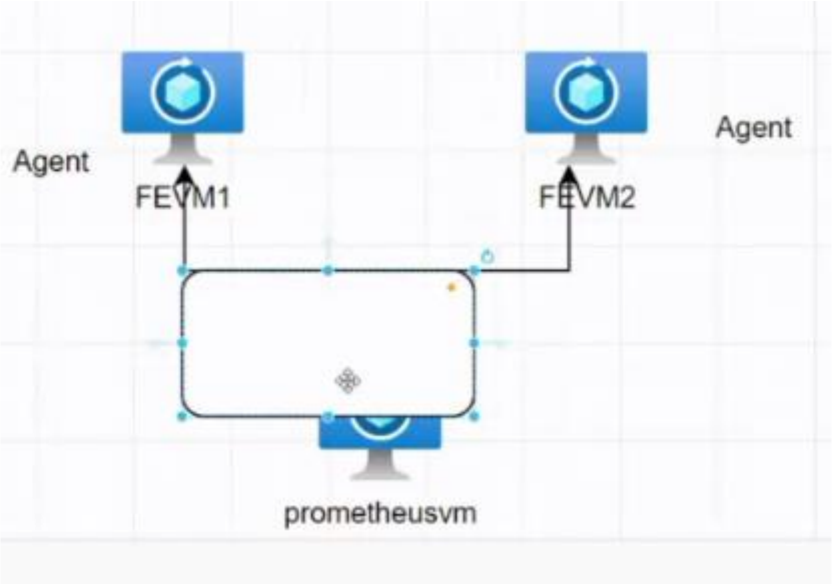
Logs = Grafana Loki - Promtail

Dashboard = Grafana

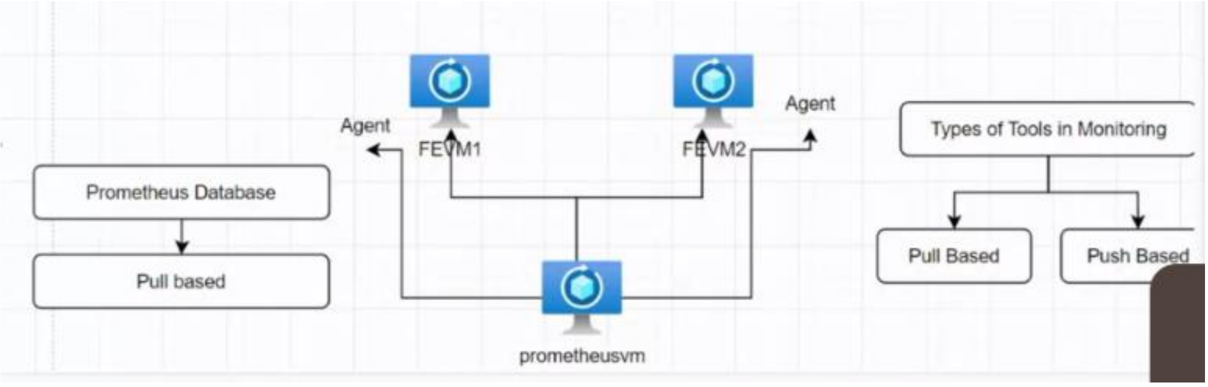
Cloud Based Paid Tools

Azure Monitor - Metrics and Logs Both

16)



17) In this Prometheus server will go to agent and then it will ask for data i.e. pull based. If agent itself goes to Prometheus server, then it will be push based.



++

AGENDA – Create 2 VMs and 1 Prometheus VM

1) Now creating 2 Vms

VMPRO1 and **VMPRO2** and **VMCMSPRO** (Central monitoring system)

UN - azureuser

PW - Mommy7Daddy!

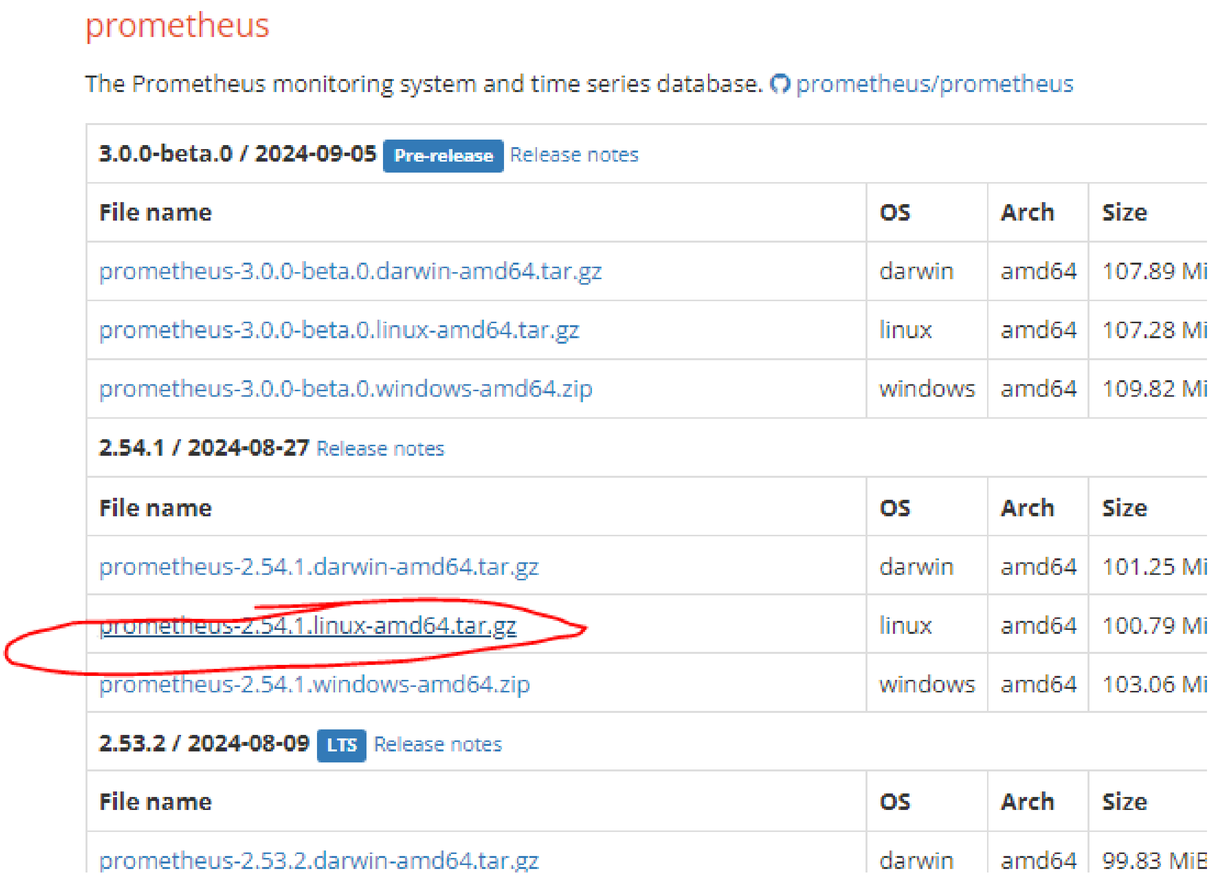
2) We are making a kind of setup that will be open source and by making 1 computer we will be able to do everything on it.

3) So basically we have to monitor vm1 and vm2 by using our central monitor computer or vm (vmcmspro)

4) SEARCH PROMETHEUS URL - https://prometheus.io/docs/prometheus/latest/getting_started/

5) Now ssh to **VMCMSPRO** (Central monitoring system)

6) **wget** https://github.com/prometheus/prometheus/releases/download/v2.54.1/prometheus-2.54.1.linux-amd64.tar.gz



prometheus

The Prometheus monitoring system and time series database. prometheus/prometheus

3.0.0-beta.0 / 2024-09-05 Pre-release Release notes

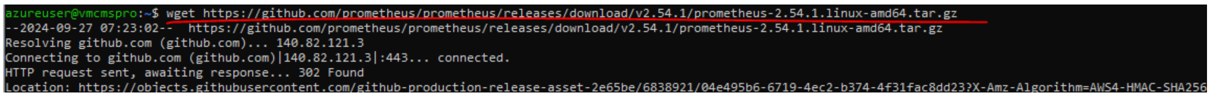
File name	OS	Arch	Size
prometheus-3.0.0-beta.0.darwin-amd64.tar.gz	darwin	amd64	107.89 Mi
prometheus-3.0.0-beta.0.linux-amd64.tar.gz	linux	amd64	107.28 Mi
prometheus-3.0.0-beta.0.windows-amd64.zip	windows	amd64	109.82 Mi

2.54.1 / 2024-08-27 Release notes

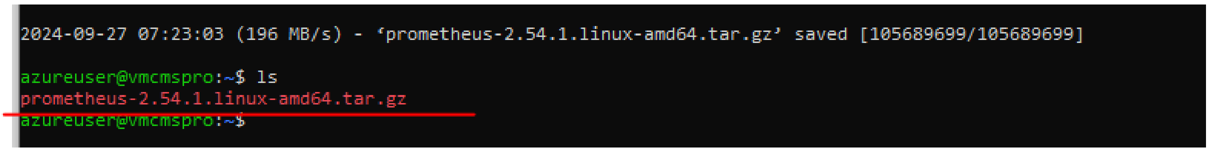
File name	OS	Arch	Size
prometheus-2.54.1.darwin-amd64.tar.gz	darwin	amd64	101.25 Mi
prometheus-2.54.1.linux-amd64.tar.gz	linux	amd64	100.79 Mi
prometheus-2.54.1.windows-amd64.zip	windows	amd64	103.06 Mi

2.53.2 / 2024-08-09 LTS Release notes

File name	OS	Arch	Size
prometheus-2.53.2.darwin-amd64.tar.gz	darwin	amd64	99.83 MiE



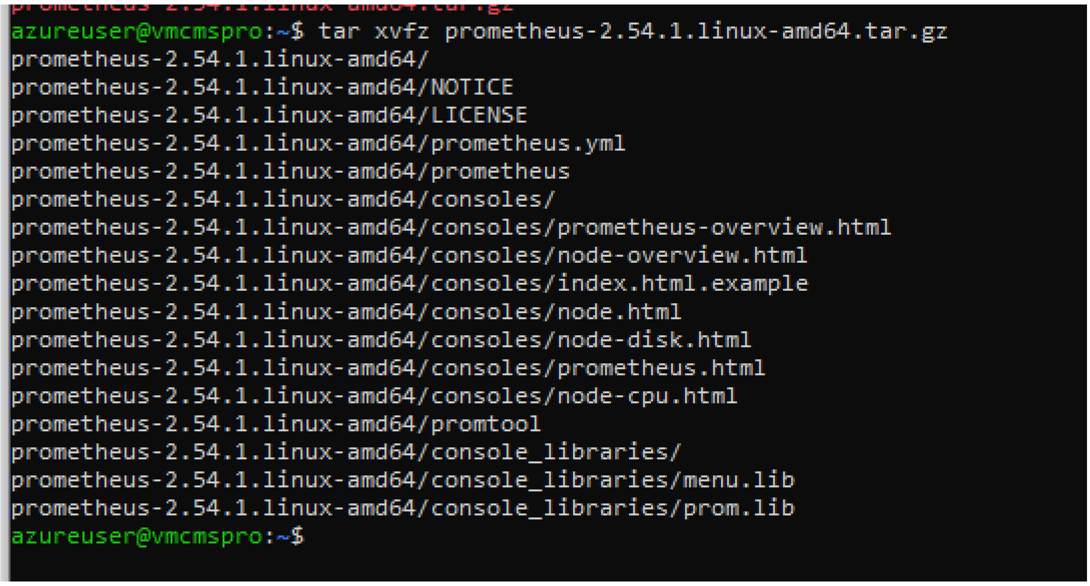
```
azureuser@vmcmspro:~$ wget https://github.com/prometheus/prometheus/releases/download/v2.54.1/prometheus-2.54.1.linux-amd64.tar.gz
--2024-09-27 07:23:02--  https://github.com/prometheus/prometheus/releases/download/v2.54.1/prometheus-2.54.1.linux-amd64.tar.gz
Resolving github.com (github.com)... 140.82.121.3
Connecting to github.com (github.com)|140.82.121.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/6838921/04e495b6-6719-4ec2-b374-4f31fac8dd23?X-Amz-Algorithm=AWS4-HMAC-SHA256
```



```
2024-09-27 07:23:03 (196 MB/s) - ‘prometheus-2.54.1.linux-amd64.tar.gz’ saved [105689699/105689699]

azureuser@vmcmspro:~$ ls
prometheus-2.54.1.linux-amd64.tar.gz
azureuser@vmcmspro:~$
```

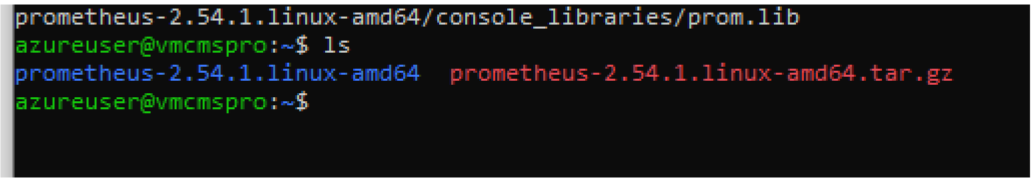
7) Now extract that downloaded file

tar xvfz prometheus-2.54.1.linux-amd64.tar.gz



```
prometheus-2.54.1.linux-amd64.tar.gz
azureuser@vmcmspro:~$ tar xvfz prometheus-2.54.1.linux-amd64.tar.gz
prometheus-2.54.1.linux-amd64/
prometheus-2.54.1.linux-amd64/NOTICE
prometheus-2.54.1.linux-amd64/LICENSE
prometheus-2.54.1.linux-amd64/prometheus.yml
prometheus-2.54.1.linux-amd64/prometheus
prometheus-2.54.1.linux-amd64/consoles/
prometheus-2.54.1.linux-amd64/consoles/prometheus-overview.html
prometheus-2.54.1.linux-amd64/consoles/node-overview.html
prometheus-2.54.1.linux-amd64/consoles/index.html.example
prometheus-2.54.1.linux-amd64/consoles/node.html
prometheus-2.54.1.linux-amd64/consoles/node-disk.html
prometheus-2.54.1.linux-amd64/consoles/prometheus.html
prometheus-2.54.1.linux-amd64/consoles/node-cpu.html
prometheus-2.54.1.linux-amd64/promtool
prometheus-2.54.1.linux-amd64/console_libraries/
prometheus-2.54.1.linux-amd64/console_libraries/menu.lib
prometheus-2.54.1.linux-amd64/console_libraries/prom.lib
azureuser@vmcmspro:~$
```

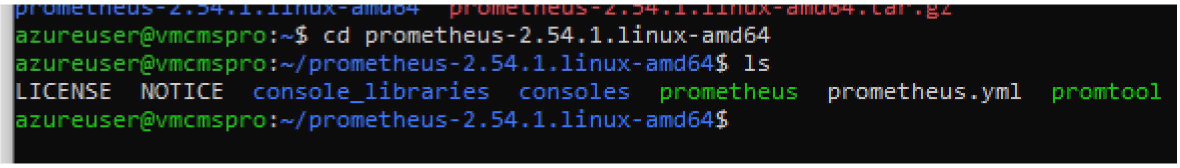
8) **ls**



```
prometheus-2.54.1.linux-amd64/console_libraries/prom.lib
azureuser@vmcmspro:~$ ls
prometheus-2.54.1.linux-amd64  prometheus-2.54.1.linux-amd64.tar.gz
azureuser@vmcmspro:~$
```

9) **cd prometheus-2.54.1.linux-amd64**

ls

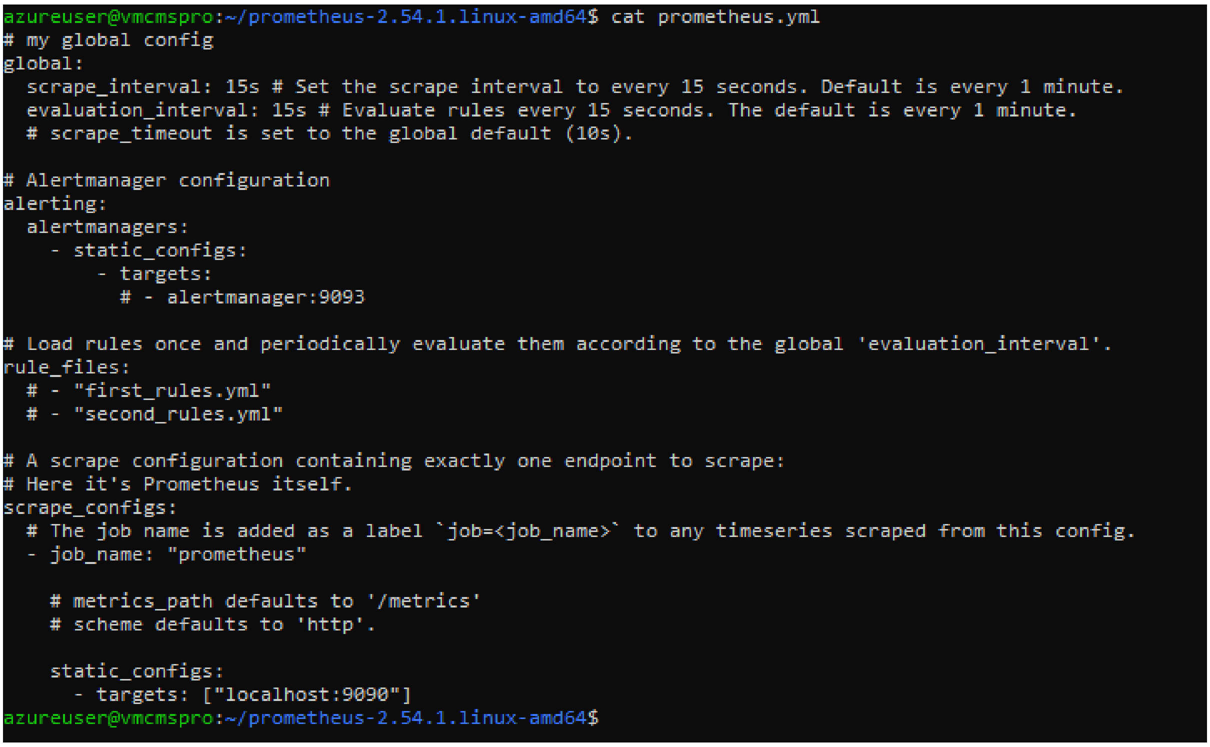


```
prometheus-2.54.1.linux-amd64  prometheus-2.54.1.linux-amd64.tar.gz
azureuser@vmcmspro:~$ cd prometheus-2.54.1.linux-amd64
azureuser@vmcmspro:~/prometheus-2.54.1.linux-amd64$ ls
LICENSE  NOTICE  console_libraries  consoles  prometheus  prometheus.yml  promtool
azureuser@vmcmspro:~/prometheus-2.54.1.linux-amd64$
```

10) Prometheus works on which port by default – **9090**

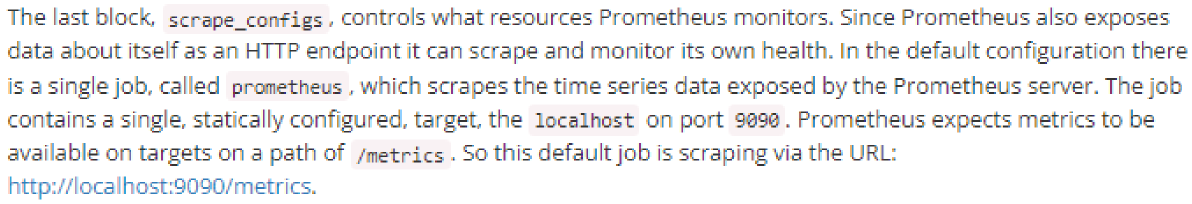
11) Prometheus configuration is YAML - YAML Ain't Markup Language. YAML is a human-friendly data serialization language for all programming languages.

12) **cat prometheus.yml**



```
azureuser@vmcmspro:~/prometheus-2.54.1.linux-amd64$ cat prometheus.yml
# 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"]
azureuser@vmcmspro:~/prometheus-2.54.1.linux-amd64$
```

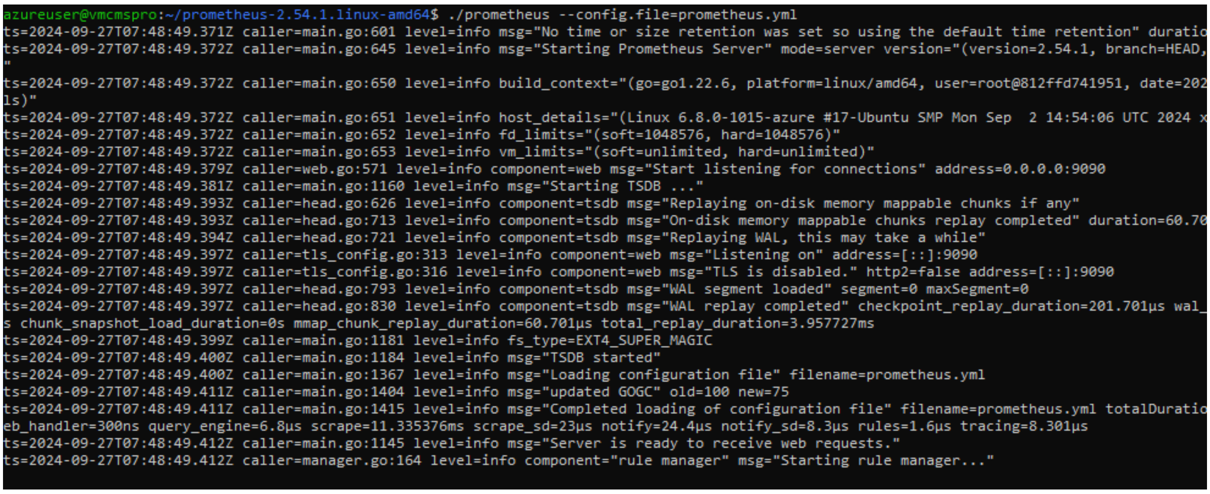
13)



The last block, `scrape_configs`, controls what resources Prometheus monitors. Since Prometheus also exposes data about itself as an HTTP endpoint it can scrape and monitor its own health. In the default configuration there is a single job, called `prometheus`, which scrapes the time series data exposed by the Prometheus server. The job contains a single, statically configured, target, the `localhost` on port `9090`. Prometheus expects metrics to be available on targets on a path of `/metrics`. So this default job is scraping via the URL: http://localhost:9090/metrics.

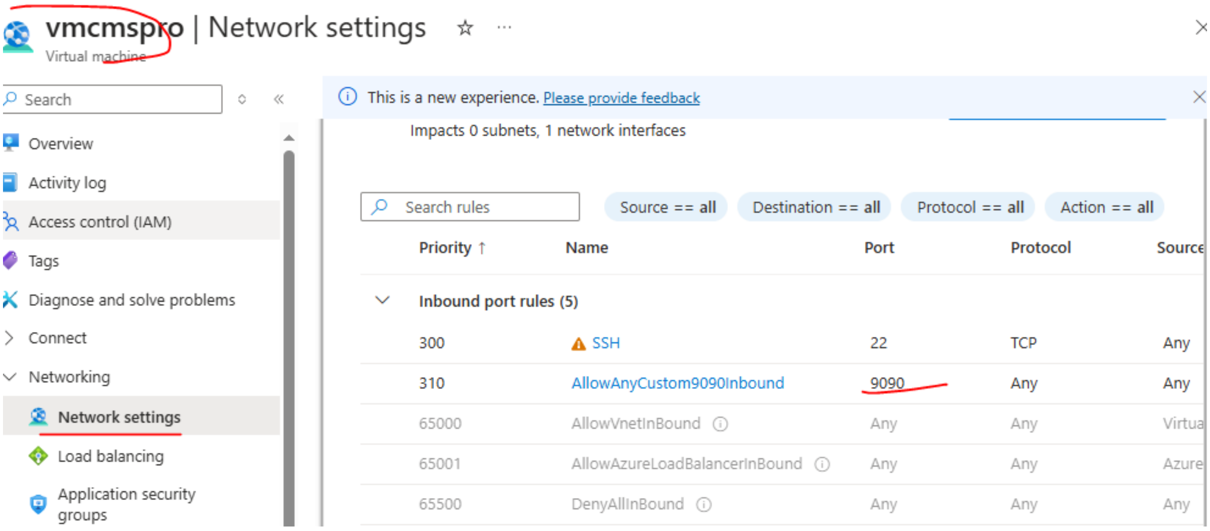
14) To start Prometheus with our newly created configuration file, change to the directory containing the Prometheus binary and run:

./prometheus --config.file=prometheus.yml



```
azureuser@vmcmspro:~/prometheus-2.54.1.linux-amd64$ ./prometheus --config.file=prometheus.yml
ts=2024-09-27T07:48:49.371Z caller=main.go:601 level=info msg="No time or size retention was set so using the default time retention" duratio
ts=2024-09-27T07:48:49.372Z caller=main.go:645 level=info msg="Starting Prometheus Server" mode=server version="(version=2.54.1, branch=HEAD,
"
ts=2024-09-27T07:48:49.372Z caller=main.go:650 level=info build_context="(go=go1.22.6, platform=linux/amd64, user=root@812ffd741951, date=202
ls)"
ts=2024-09-27T07:48:49.372Z caller=main.go:651 level=info host_details="(Linux 6.8.0-1015-azure #17-Ubuntu SMP Mon Sep  2 14:54:06 UTC 2024 x
ts=2024-09-27T07:48:49.372Z caller=main.go:652 level=info fd_limits="(soft=1048576, hard=1048576)"
ts=2024-09-27T07:48:49.372Z caller=main.go:653 level=info vm_limits="(soft=unlimited, hard=unlimited)"
ts=2024-09-27T07:48:49.379Z caller=web.go:571 level=info component=web msg="Start listening for connections" address=0.0.0.0:9090
ts=2024-09-27T07:48:49.381Z caller=main.go:1160 level=info msg="Starting TSDB ..."
ts=2024-09-27T07:48:49.393Z caller=head.go:626 level=info component=tsdb msg="Replaying on-disk memory mappable chunks if any"
ts=2024-09-27T07:48:49.393Z caller=head.go:713 level=info component=tsdb msg="On-disk memory mappable chunks replay completed" duration=60.70
ts=2024-09-27T07:48:49.394Z caller=head.go:721 level=info component=tsdb msg="Replaying WAL, this may take a while"
ts=2024-09-27T07:48:49.397Z caller=tls_config.go:313 level=info component=web msg="Listening on" address=[::]:9090
ts=2024-09-27T07:48:49.397Z caller=tls_config.go:316 level=info component=web msg="TLS is disabled." http2=false address=[::]:9090
ts=2024-09-27T07:48:49.397Z caller=head.go:793 level=info component=tsdb msg="WAL segment loaded" segment=0 maxSegment=0
ts=2024-09-27T07:48:49.397Z caller=head.go:830 level=info component=tsdb msg="WAL replay completed" checkpoint_replay_duration=201.701µs wal_
s chunk_snapshot_load_duration=0s mmap_chunk_replay_duration=60.701µs total_replay_duration=3.957727ms
ts=2024-09-27T07:48:49.399Z caller=main.go:1181 level=info fs_type=EXT4_SUPER_MAGIC
ts=2024-09-27T07:48:49.400Z caller=main.go:1184 level=info msg="TSDB started"
ts=2024-09-27T07:48:49.400Z caller=main.go:1367 level=info msg="Loading configuration file" filename=prometheus.yml
ts=2024-09-27T07:48:49.411Z caller=main.go:1404 level=info msg="updated GOGC" old=100 new=75
ts=2024-09-27T07:48:49.411Z caller=main.go:1415 level=info msg="Completed loading of configuration file" filename=prometheus.yml totalDuratio
eb_handler=300ns query_engine=6.8µs scrape=11.335376ms scrape_sd=23µs notify=24.4µs notify_sd=8.3µs rules=1.6µs tracing=8.301µs
ts=2024-09-27T07:48:49.412Z caller=main.go:1145 level=info msg="Server is ready to receive web requests."
ts=2024-09-27T07:48:49.412Z caller=manager.go:164 level=info component="rule manager" msg="Starting rule manager..."
```

15) Now in azure portal enable port 9090



This is a new experience. Please provide feedback

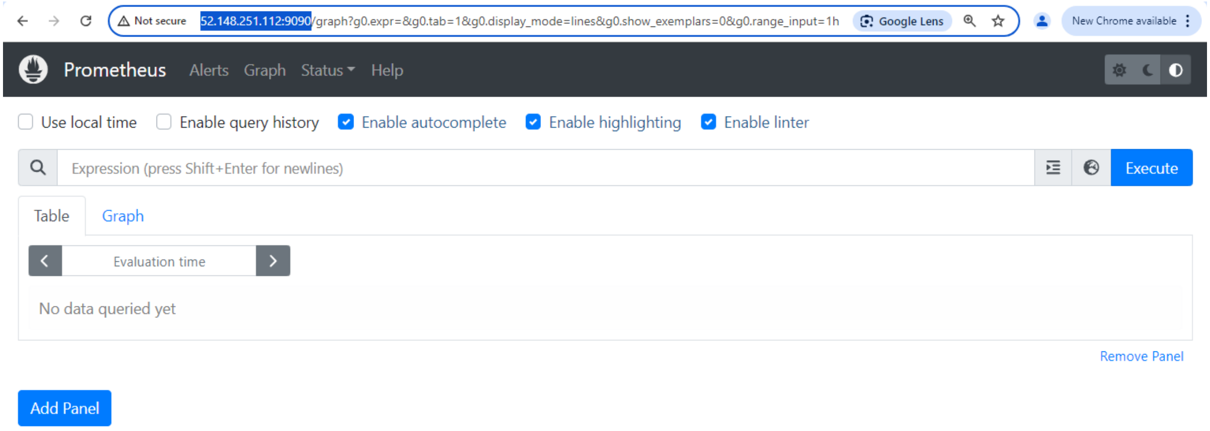
Impacts 0 subnets, 1 network interfaces

Search rules · Source == all · Destination == all · Protocol == all · Action == all

Priority ↑	Name	Port	Protocol	Source
Inbound port rules (5)				
300	⚠ SSH	22	TCP	Any
310	AllowAnyCustom9090Inbound	9090	Any	Any
65000	AllowVnetInBound ⓘ	Any	Any	Virtua
65001	AllowAzureLoadBalancerInBound ⓘ	Any	Any	Azure
65500	DenyAllInBound ⓘ	Any	Any	Any

16) Take public ip of vmcmspro and port no 9090 of Prometheus

http://52.148.251.112:9090/



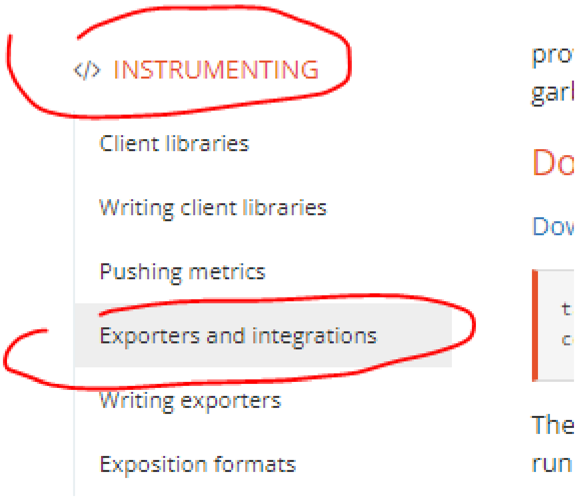
52.148.251.112:9090/graph?g0.expr=&g0.tab=1&g0.display_mode=lines&g0.show_exemplars=0&g0.range_input=1h

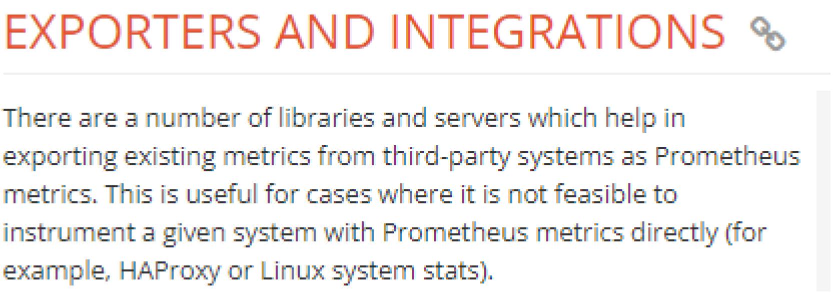
□ Use local time □ Enable query history ☑ Enable autocomplete ☑ Enable highlighting ☑ Enable linter

Expression (press Shift+Enter for newlines)

No data queried yet

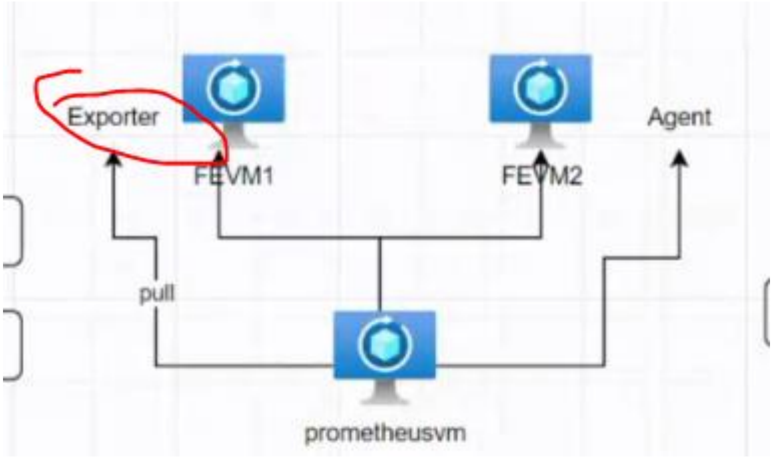
17) Now we need exporter – Means from how many places we can bring data or export



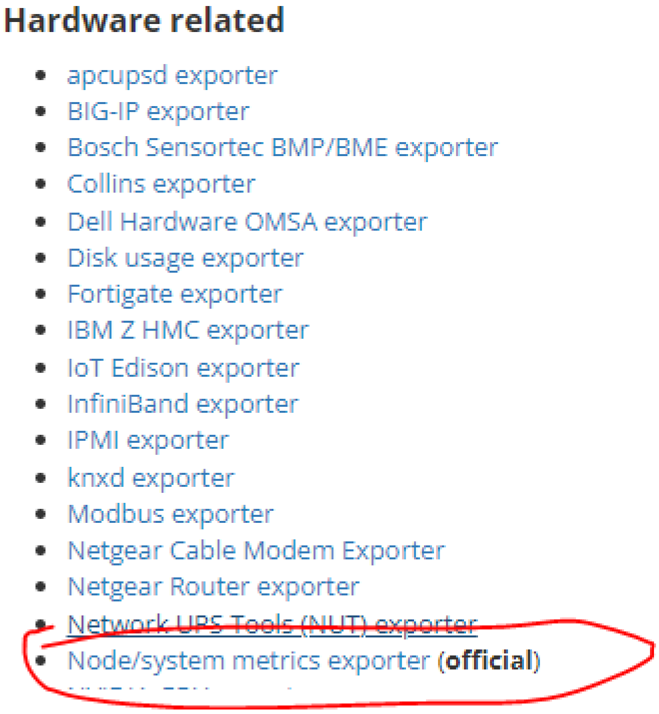


EXPORTERS AND INTEGRATIONS

There are a number of libraries and servers which help in exporting existing metrics from third-party systems as Prometheus metrics. This is useful for cases where it is not feasible to instrument a given system with Prometheus metrics directly (for example, HAProxy or Linux system stats).

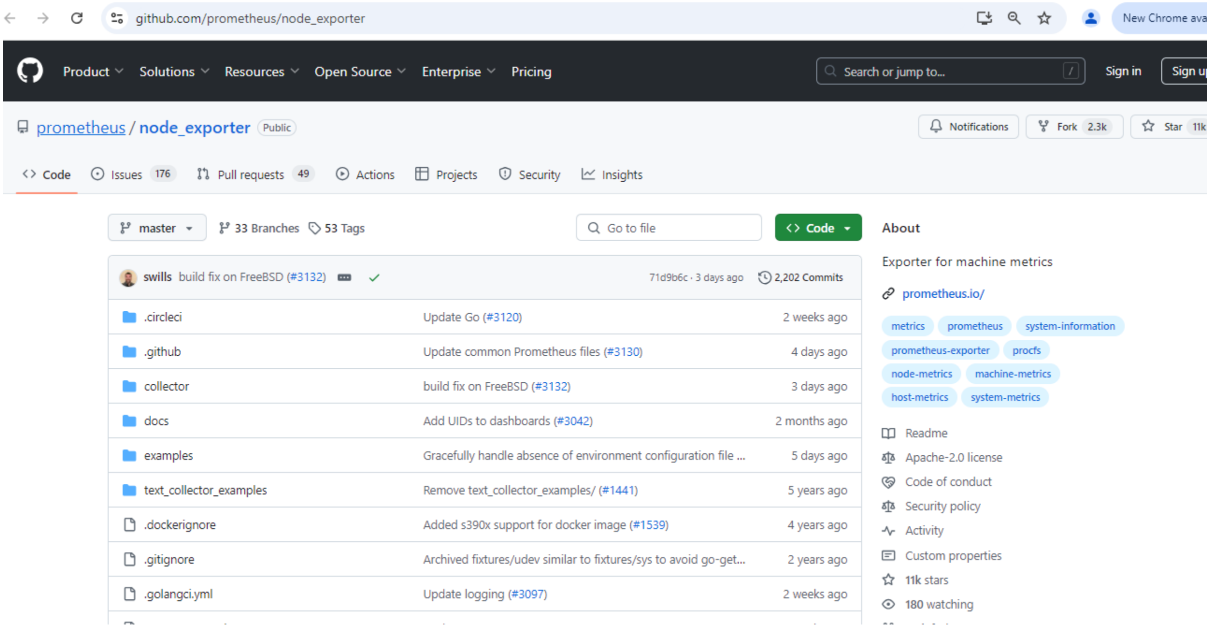


18) Now we have to put exporter on VM1 which will help us to monitor that machine



Hardware related

- apcupsd exporter
- BIG-IP exporter
- Bosch Sensortec BMP/BME exporter
- Collins exporter
- Dell Hardware OMSA exporter
- Disk usage exporter
- Fortigate exporter
- IBM Z HMC exporter
- IoT Edison exporter
- InfiniBand exporter
- IPMI exporter
- knxd exporter
- Modbus exporter
- Netgear Cable Modem Exporter
- Netgear Router exporter
- Network UPS Tools (NUT) exporter
- Node/system metrics exporter (**official**)



swills build fix on FreeBSD (#3132) — 71d9b6c · 3 days ago — 2,202 Commits

.circleci	Update Go (#3120)	2 weeks ago
.github	Update common Prometheus files (#3130)	4 days ago
collector	build fix on FreeBSD (#3132)	3 days ago
docs	Add UIDs to dashboards (#3042)	2 months ago
examples	Gracefully handle absence of environment configuration file ...	5 days ago
text_collector_examples	Remove text_collector_examples/ (#1441)	5 years ago
.dockerignore	Added s390x support for docker image (#1539)	4 years ago
.gitignore	Archived fixtures/udev similar to fixtures/sys to avoid go-get...	2 years ago
.golangci.yml	Update logging (#3097)	2 weeks ago

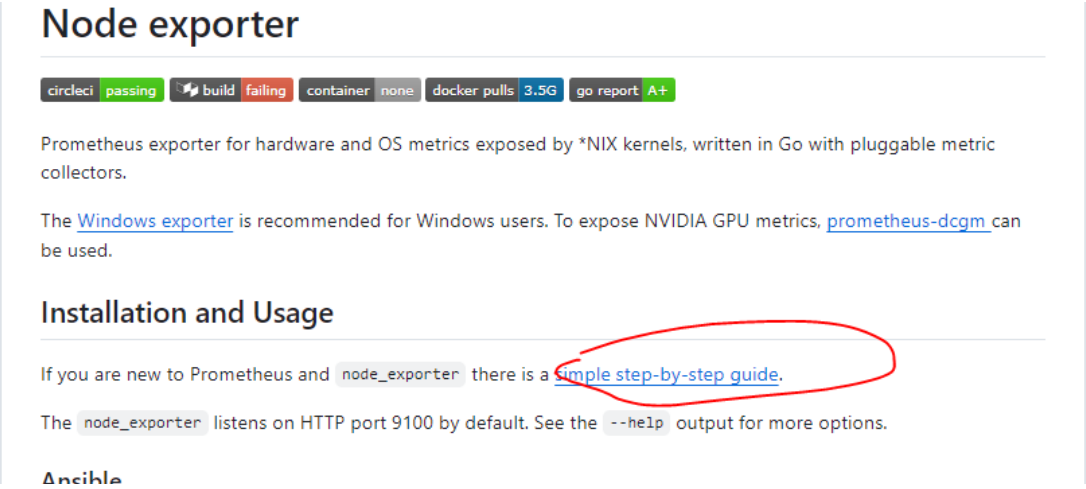
About

Exporter for machine metrics

prometheus.io/

metrics prometheus system-information prometheus-exporter procfs node-metrics machine-metrics host-metrics system-metrics

Readme
Apache-2.0 license
Code of conduct
Security policy
Activity
Custom properties
11k stars
180 watching



Node exporter

Prometheus exporter for hardware and OS metrics exposed by *NIX kernels, written in Go with pluggable metric collectors.

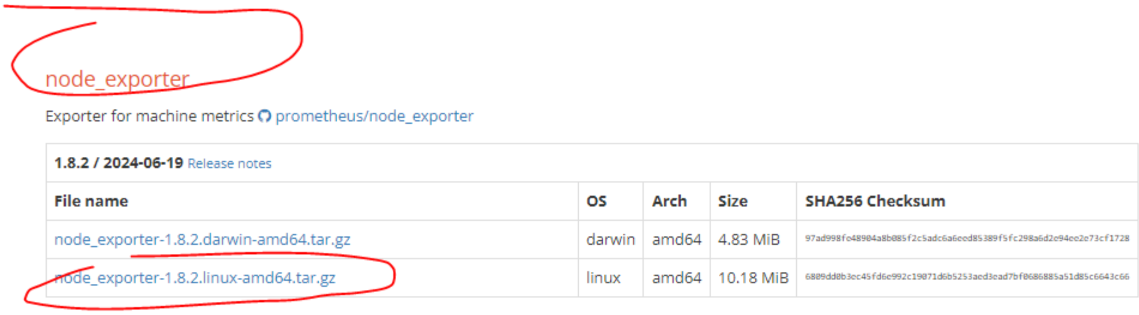
The Windows exporter is recommended for Windows users. To expose NVIDIA GPU metrics, prometheus-dcgm can be used.

Installation and Usage

If you are new to Prometheus and `node_exporter` there is a simple step-by-step guide.

The `node_exporter` listens on HTTP port 9100 by default. See the `--help` output for more options.

Ansible



node_exporter

Exporter for machine metrics prometheus/node_exporter

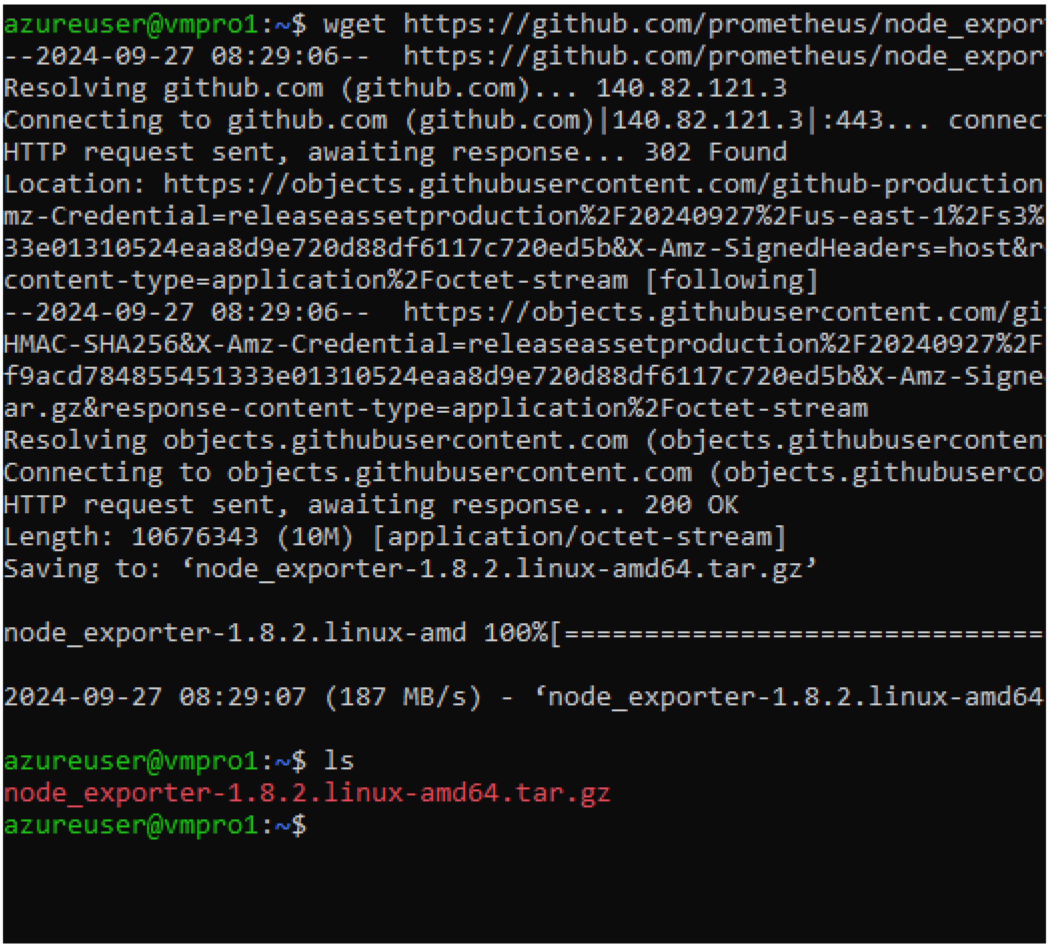
1.8.2 / 2024-06-19 Release notes

File name	OS	Arch	Size	SHA256 Checksum
node_exporter-1.8.2.darwin-amd64.tar.gz	darwin	amd64	4.83 MiB	97ad998fc48984a8b885f2c5adc6a6eed85389f5fc298a6d2e94ee2e73cf1728
node_exporter-1.8.2.linux-amd64.tar.gz	linux	amd64	10.18 MiB	6809dd0b3ec45fd6e992c19071d6b5253aed3ead7bf0686885a51d85c6643c66

19) Copy above link and ssh to vm1 and vm2

wget https://github.com/prometheus/node_exporter/releases/download/v1.8.2/node_exporter-1.8.2.linux-amd64.tar.gz

ls

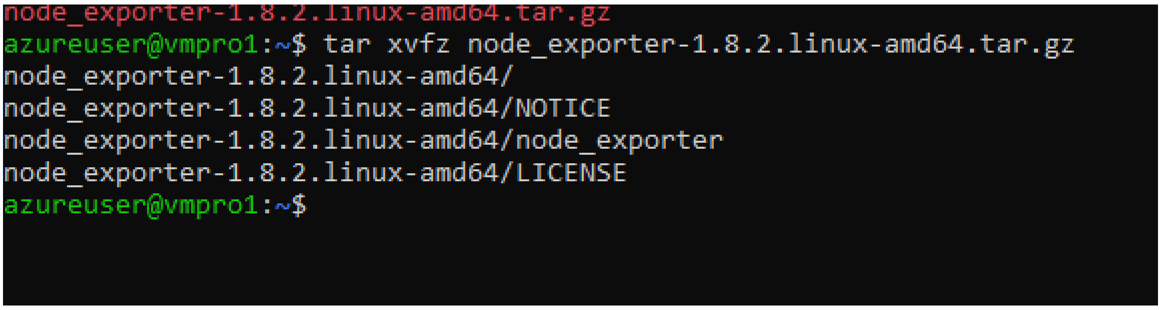


```
azureuser@vmpro1:~$ wget https://github.com/prometheus/node_expor
--2024-09-27 08:29:06--  https://github.com/prometheus/node_expor
Resolving github.com (github.com)... 140.82.121.3
Connecting to github.com (github.com)|140.82.121.3|:443... connec
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production
mz-Credential=releaseassetproduction%2F20240927%2Fus-east-1%2Fs3%
33e01310524eaa8d9e720d88df6117c720ed5b&X-Amz-SignedHeaders=host&r
content-type=application%2Foctet-stream [following]
--2024-09-27 08:29:06--  https://objects.githubusercontent.com/gi
HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20240927%2F
f9acd784855451333e01310524eaa8d9e720d88df6117c720ed5b&X-Amz-Signe
ar.gz&response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubuserconten
Connecting to objects.githubusercontent.com (objects.githubuserco
HTTP request sent, awaiting response... 200 OK
Length: 10676343 (10M) [application/octet-stream]
Saving to: ‘node_exporter-1.8.2.linux-amd64.tar.gz’

node_exporter-1.8.2.linux-amd 100%[=============================

2024-09-27 08:29:07 (187 MB/s) - ‘node_exporter-1.8.2.linux-amd64

azureuser@vmpro1:~$ ls
node_exporter-1.8.2.linux-amd64.tar.gz
azureuser@vmpro1:~$
```

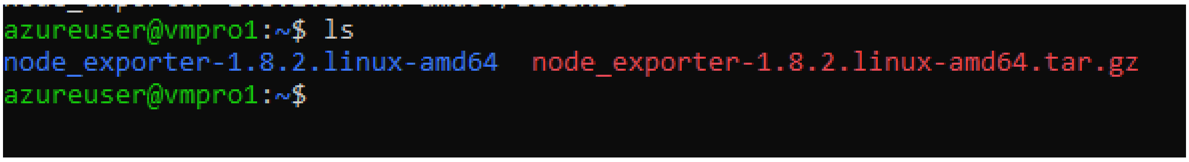
20) Now extract downloaded file

tar xvfz node_exporter-1.8.2.linux-amd64.tar.gz

ls



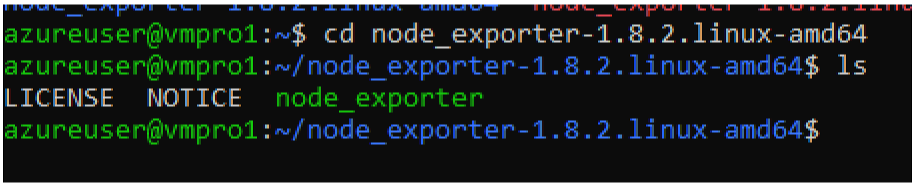
```
node_exporter-1.8.2.linux-amd64.tar.gz
azureuser@vmpro1:~$ tar xvfz node_exporter-1.8.2.linux-amd64.tar.gz
node_exporter-1.8.2.linux-amd64/
node_exporter-1.8.2.linux-amd64/NOTICE
node_exporter-1.8.2.linux-amd64/node_exporter
node_exporter-1.8.2.linux-amd64/LICENSE
azureuser@vmpro1:~$
```



```
azureuser@vmpro1:~$ ls
node_exporter-1.8.2.linux-amd64  node_exporter-1.8.2.linux-amd64.tar.gz
azureuser@vmpro1:~$
```

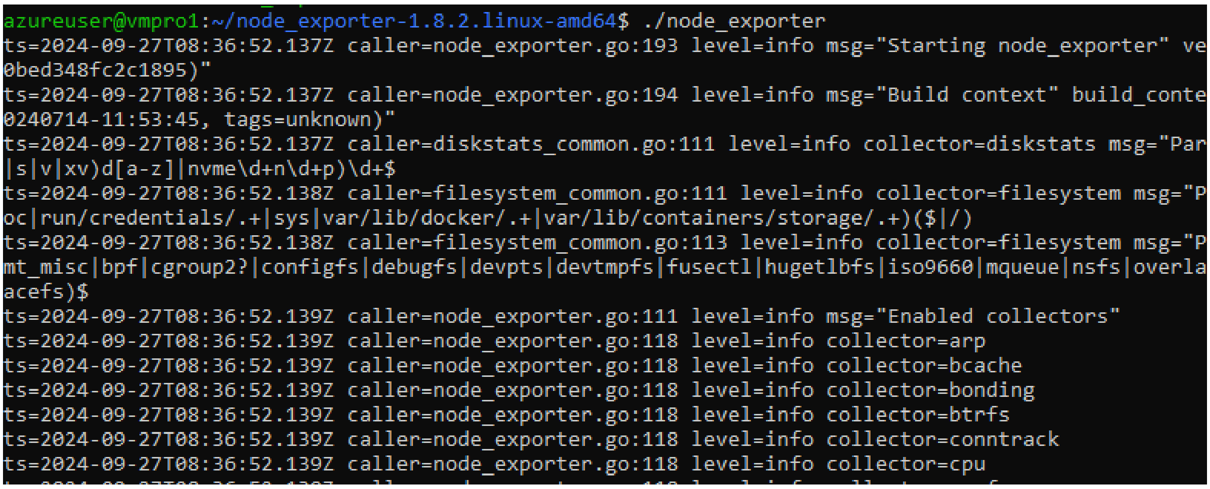
21) **cd node_exporter-1.8.2.linux-amd64**

ls



```
azureuser@vmpro1:~$ cd node_exporter-1.8.2.linux-amd64
azureuser@vmpro1:~/node_exporter-1.8.2.linux-amd64$ ls
LICENSE  NOTICE  node_exporter
azureuser@vmpro1:~/node_exporter-1.8.2.linux-amd64$
```

22) **./node_exporter** – Enter means run node exporter



```
azureuser@vmpro1:~/node_exporter-1.8.2.linux-amd64$ ./node_exporter
ts=2024-09-27T08:36:52.137Z caller=node_exporter.go:193 level=info msg="Starting node_exporter" ve
0bed348fc2c1895)"
ts=2024-09-27T08:36:52.137Z caller=node_exporter.go:194 level=info msg="Build context" build_conte
0240714-11:53:45, tags=unknown)"
ts=2024-09-27T08:36:52.137Z caller=diskstats_common.go:111 level=info collector=diskstats msg="Par
|s|v|xv)d[a-z]|nvme\d+n\d+p)\d+$
ts=2024-09-27T08:36:52.138Z caller=filesystem_common.go:111 level=info collector=filesystem msg="P
oc|run/credentials/.+|sys|var/lib/docker/.+|var/lib/containers/storage/.+)($|/)
ts=2024-09-27T08:36:52.138Z caller=filesystem_common.go:113 level=info collector=filesystem msg="P
mt_misc|bpf|cgroup2?|configfs|debugfs|devpts|devtmpfs|fusectl|hugetlbfs|iso9660|mqueue|nsfs|overla
acefs)$
ts=2024-09-27T08:36:52.139Z caller=node_exporter.go:111 level=info msg="Enabled collectors"
ts=2024-09-27T08:36:52.139Z caller=node_exporter.go:118 level=info collector=arp
ts=2024-09-27T08:36:52.139Z caller=node_exporter.go:118 level=info collector=bcache
ts=2024-09-27T08:36:52.139Z caller=node_exporter.go:118 level=info collector=bonding
ts=2024-09-27T08:36:52.139Z caller=node_exporter.go:118 level=info collector=btrfs
ts=2024-09-27T08:36:52.139Z caller=node_exporter.go:118 level=info collector=conntrack
ts=2024-09-27T08:36:52.139Z caller=node_exporter.go:118 level=info collector=cpu
```

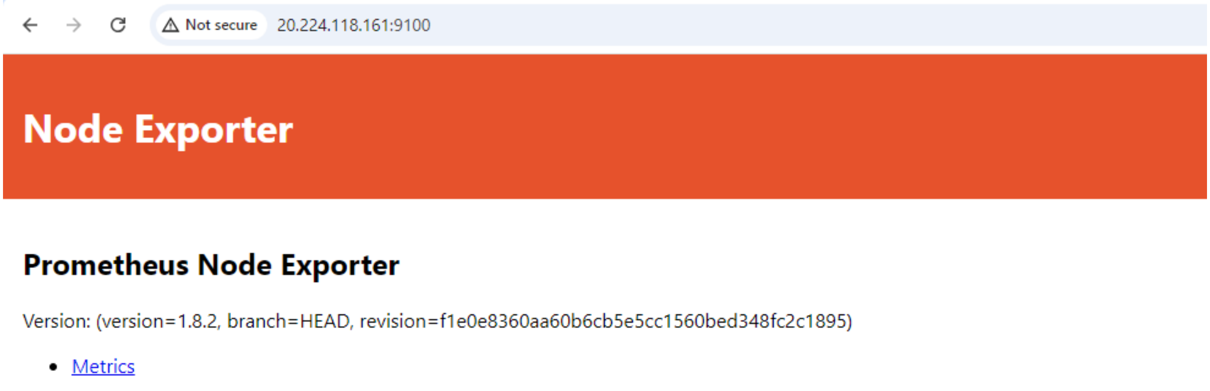
23) Now node exporter is running on both VM1 and VM2. Basically node exporter extracts out all metrics of VM1 and VM2 and will run a small server inside itself on port 9100 as per below details and will public all data on 9100 port of VM

```
...
INFO[0000] Listening on :9100
```

24) Now opening port 9100 of exporter on both frontend vm1 and vm2, and run on browser

Vmpro1 - 20.23.244.216:9100

Vmpro2 - 20.224.118.161:9100



Node Exporter

Prometheus Node Exporter

Version: (version=1.8.2, branch=HEAD, revision=f1e0e8360aa60b6cb5e5cc1560bed348fc2c1895)

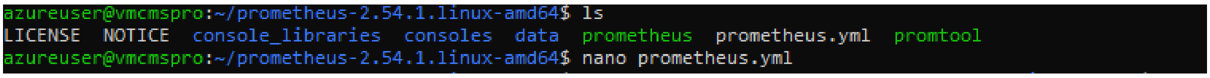
- Metrics

25) Now stop vm of Prometheus

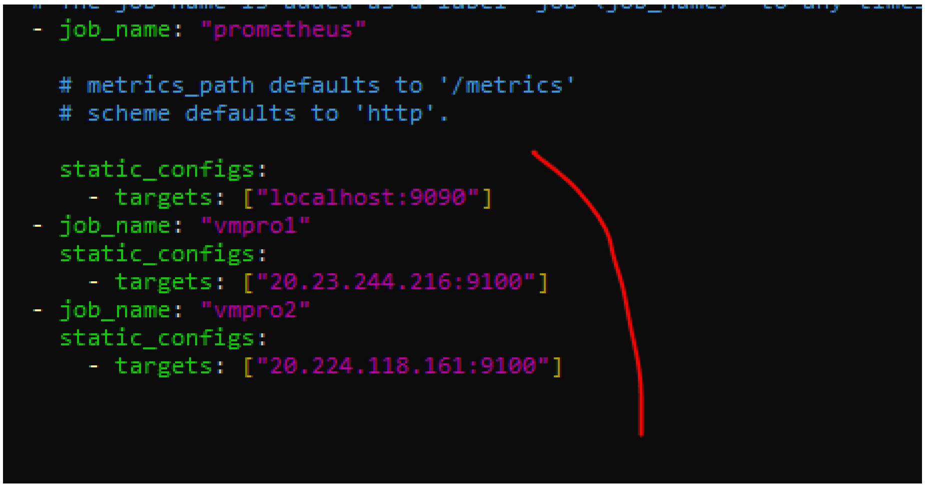
ls

nano prometheus.yml

add both vm1 and vm2 details – ctrlO - ctrlx

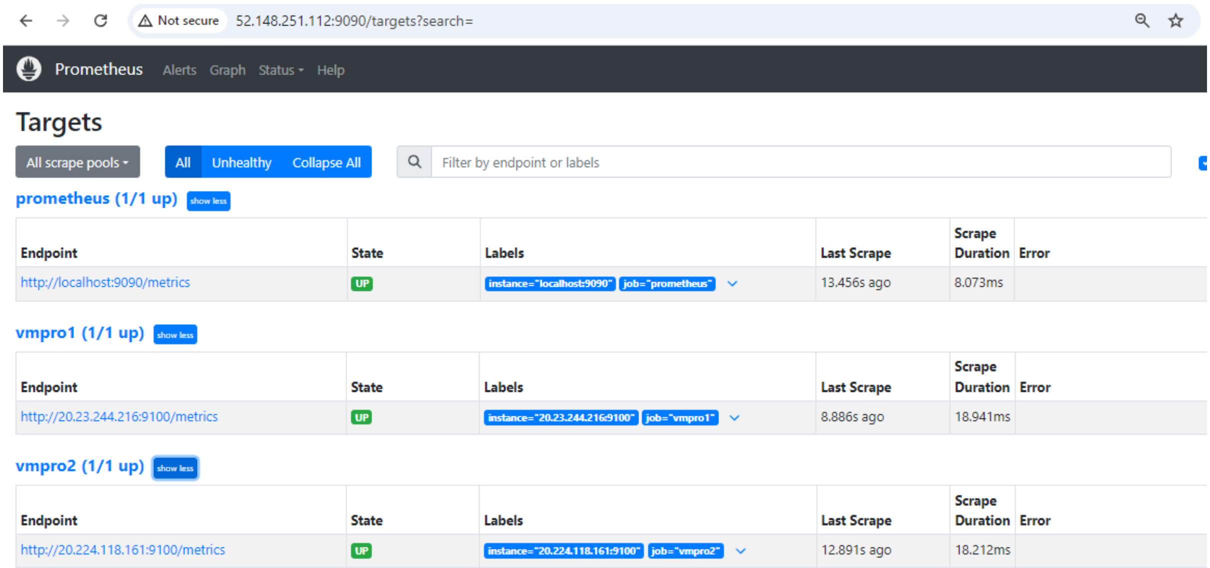


```
azureuser@vmcmspro:~/prometheus-2.54.1.linux-amd64$ ls
LICENSE  NOTICE  console_libraries  consoles  data  prometheus  prometheus.yml  promtool
azureuser@vmcmspro:~/prometheus-2.54.1.linux-amd64$ nano prometheus.yml
```



```
  # The job name is added as a label `job=<job_name>` to any time
  - job_name: "prometheus"

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

    static_configs:
      - targets: ["localhost:9090"]
  - job_name: "vmpro1"
    static_configs:
      - targets: ["20.23.244.216:9100"]
  - job_name: "vmpro2"
    static_configs:
      - targets: ["20.224.118.161:9100"]
```

Start Prometheus -



Targets

prometheus (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	13.456s ago	8.073ms	

vmpro1 (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://20.23.244.216:9100/metrics	UP	instance="20.23.244.216:9100" job="vmpro1"	8.886s ago	18.941ms	

vmpro2 (1/1 up)

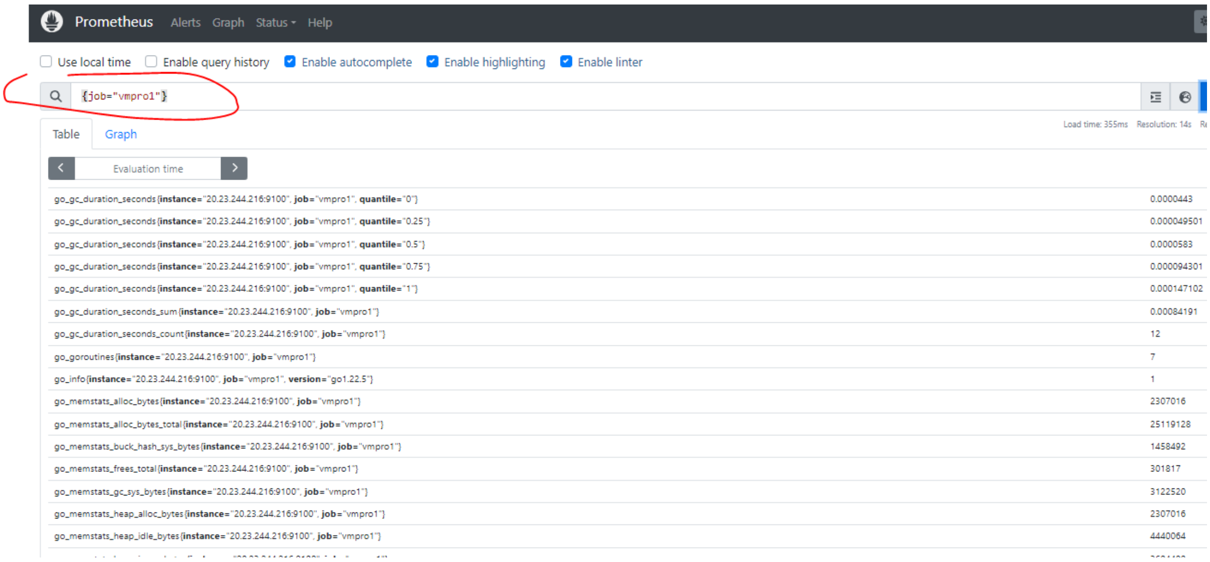
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://20.224.118.161:9100/metrics	UP	instance="20.224.118.161:9100" job="vmpro2"	12.891s ago	18.212ms	

26) {job="vmpro1"}

Or

{job="vmpro2"}

So below we can check and and get logs



□ Use local time □ Enable query history ☑ Enable autocomplete ☑ Enable highlighting ☑ Enable linter

```
{job="vmpro1"}
```

Load time: 355ms Resolution: 14s

go_gc_duration_seconds{instance="20.23.244.216:9100", job="vmpro1", quantile="0"}	0.0000443
go_gc_duration_seconds{instance="20.23.244.216:9100", job="vmpro1", quantile="0.25"}	0.000049501
go_gc_duration_seconds{instance="20.23.244.216:9100", job="vmpro1", quantile="0.5"}	0.0000583
go_gc_duration_seconds{instance="20.23.244.216:9100", job="vmpro1", quantile="0.75"}	0.000094301
go_gc_duration_seconds{instance="20.23.244.216:9100", job="vmpro1", quantile="1"}	0.000147102
go_gc_duration_seconds_sum{instance="20.23.244.216:9100", job="vmpro1"}	0.00084191
go_gc_duration_seconds_count{instance="20.23.244.216:9100", job="vmpro1"}	12
go_goroutines{instance="20.23.244.216:9100", job="vmpro1"}	7
go_info{instance="20.23.244.216:9100", job="vmpro1", version="go1.22.5"}	1
go_memstats_alloc_bytes{instance="20.23.244.216:9100", job="vmpro1"}	2307016
go_memstats_alloc_bytes_total{instance="20.23.244.216:9100", job="vmpro1"}	25119128
go_memstats_buck_hash_sys_bytes{instance="20.23.244.216:9100", job="vmpro1"}	1458492
go_memstats_frees_total{instance="20.23.244.216:9100", job="vmpro1"}	301817
go_memstats_gc_sys_bytes{instance="20.23.244.216:9100", job="vmpro1"}	3122520
go_memstats_heap_alloc_bytes{instance="20.23.244.216:9100", job="vmpro1"}	2307016
go_memstats_heap_idle_bytes{instance="20.23.244.216:9100", job="vmpro1"}	4440064

27)