Guia 1

Learning ProxMox, hardening, configuring, monitoring, automatization etc…

I'm using an old laptop

1- I installed proxmox.iso from the official proxmox website.

2- After installing and booting it for the first time i had to do some troubleshooting and performance enhancements.

Since it is a laptop i ran some commands to maintain the os working even with the screen shut off.

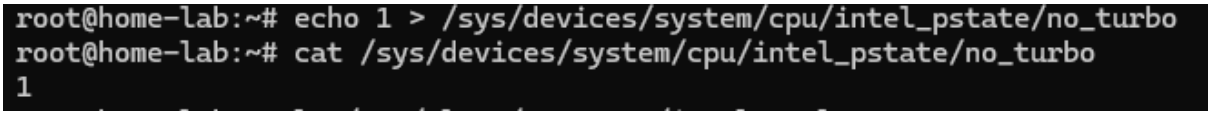
systemctl mask sleep.target suspend.target hibernate.target hybrid-sleep.target

i also set it on powersave 24/7 with this command:

```
for c in /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor; do
  echo powersave > $c
done
```

Limiting the CPU boost to reduce temperature risks

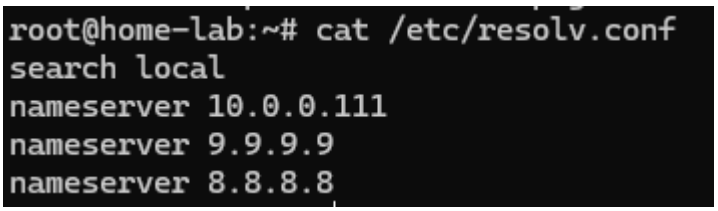
echo 1 > /sys/devices/system/cpu/intel_pstate/no_turbo



3- I want to try to install these apps:
apt install lm-sensors
sensors-detect
sensors

But before that we need to configure the apt source list and our DNS, lets start with the dns first.

first we run nano /etc/resolv.conf
and populate with the dns, i prefer quad9 for security reasons, and cloudflare.



cat /etc/resolv.conf

```
root@home-lab:~# ping -c 3 deb.debian.org
PING debian.map.fastlydns.net (151.101.194.132) 56(84) bytes of data.
64 bytes from 151.101.194.132: icmp_seq=1 ttl=54 time=19.9 ms
64 bytes from 151.101.194.132: icmp_seq=2 ttl=54 time=20.4 ms
^C
--- debian.map.fastlydns.net ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 19.898/20.137/20.377/0.239 ms
root@home-lab:~# apt update
Get:1 http://security.debian.org/debian-security bookworm-security InRele
Get:2 http://deb.debian.org/debian bookworm InRelease [151 kB]
Get:3 http://security.debian.org/debian-security trixie-security InReleas
Get:4 http://deb.debian.org/debian bookworm-updates InRelease [55.4 kB]
Get:5 http://deb.debian.org/debian trixie InRelease [140 kB]
```

Its working now!

lets create our automatic temperature alert
 lets run:
apt install lm-sensors mailutils
sensors-detect

```
root@home-lab:~# apt install lm-sensors mailutils
sensors-detect
Installing:
  lm-sensors  mailutils

Installing dependencies:
  gsasl-common    libgsasl18   libltdl7          libntlm0       mailutils-common
  guile-3.0-libs  libgssglue1  libmailutils9t64  libpq5         mariadb-common
  libgc1          libidn12     libmariadb3       libpython3.13  mysql-common

Suggested packages:
  fancontrol  i2c-tools  read-edid  mailutils-mh  mailutils-doc

Summary:
  Upgrading: 0, Installing: 17, Removing: 0, Not Upgrading: 3
  Download size: 12.8 MB
  Space needed: 74.6 MB / 91.4 GB available

Continue? [Y/n]
```

it says that some south bridges, cpu, memory controllers etc.. has embedded sensors, and asked me if I wanted to scan them, lets say yes.

```
Some south bridges, CPUs or memory controllers contain embedded sensors.
Do you want to scan for them? This is totally safe. (YES/no): YES
Module cpuid loaded successfully.
Silicon Integrated Systems SIS5595...                       No
VIA VT82C686 Integrated Sensors...                          No
VIA VT8231 Integrated Sensors...                            No
AMD K8 thermal sensors...                                   No
AMD Family 10h thermal sensors...                           No
AMD Family 11h thermal sensors...                           No
AMD Family 12h and 14h thermal sensors...                   No
AMD Family 15h thermal sensors...                           No
AMD Family 16h thermal sensors...                           No
AMD Family 17h thermal sensors...                           No
AMD Family 15h power sensors...                             No
AMD Family 16h power sensors...                             No
Hygon Family 18h thermal sensors...                         No
AMD Family 19h thermal sensors...                           No
Intel digital thermal sensor...                             Success!
    (driver `coretemp')
Intel AMB FB-DIMM thermal sensor...                         No
Intel 5500/5520/X58 thermal sensor...                       No
VIA C7 thermal sensor...                                    No
VIA Nano thermal sensor...                                  No

Some Super I/O chips contain embedded sensors. We have to write to
standard I/O ports to probe them. This is usually safe.
Do you want to scan for Super I/O sensors? (YES/no): YES
```

Let’s create our alert script using bash:

nano /usr/local/bin/temp-alert.sh

```
#!/bin/bash

TEMP=$(sensors | awk '/Package id 0/ {gsub("+|°C","",$4); print int($4)}')

LIMIT=80

if [ "$TEMP" -ge "$LIMIT" ]; then
  echo "⚠️ ALERTA: CPU a ${TEMP}°C no Proxmox $(hostname)" \
  | mail -s "🔥 ALERTA TÉRMICO PROXMOX" root
fi
```

e por fim chmod +x /usr/local/bin/temp-alert.sh

define a cron for every 5 minutes.

crontab -e

```
*/5 * * * * /usr/local/bin/temp-alert.sh
```

```
#
# m h  dom mon dow   command

*/5 * * * * /usr/local/bin/temp-alert.sh
```

Success

```
crontab: installing new crontab
```

now lets make an automated backup system.

1- creating a backup folder
mkdir -p /backup/configs

2- adding our bash script

nano /usr/local/bin/backup-configs.sh

#!/bin/bash
tar czf /backup/configs/proxmox-configs-$(date +%F).tar.gz \
/etc/pve /etc/network /etc/ssh

```
 GNU nano 8.4                                  /usr/local/bin/backup-configs.sh *
#!/bin/bash
tar czf /backup/configs/proxmox-configs-$(date +%F).tar.gz \
/etc/pve /etc/network /etc/ssh
```

3- creating weekly cron

0 3 * * 0 /usr/local/bin/backup-configs.sh

```
root@home-lab:~# mkdir -p /backup/configs
root@home-lab:~# nano /usr/local/bin/backup-configs.sh
root@home-lab:~# crontab -e
crontab: installing new crontab
```

—--------

Adjusting the consume/usage per hour day/night using crons

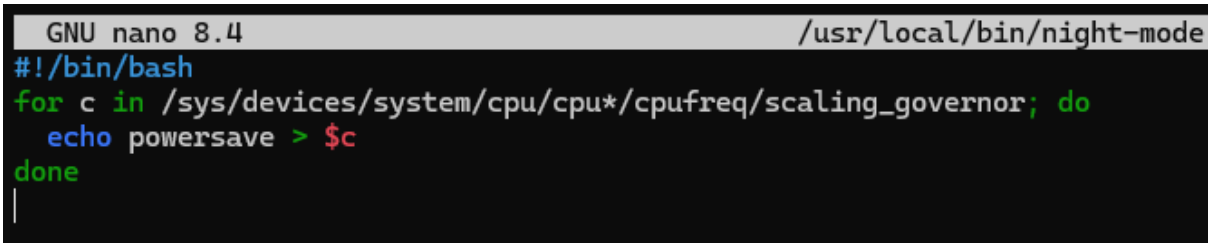
Day moderate performance:

1- nano /usr/local/bin/day-mode.sh

our script:
#!/bin/bash
for c in /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor; do
  echo schedutil > $c
done

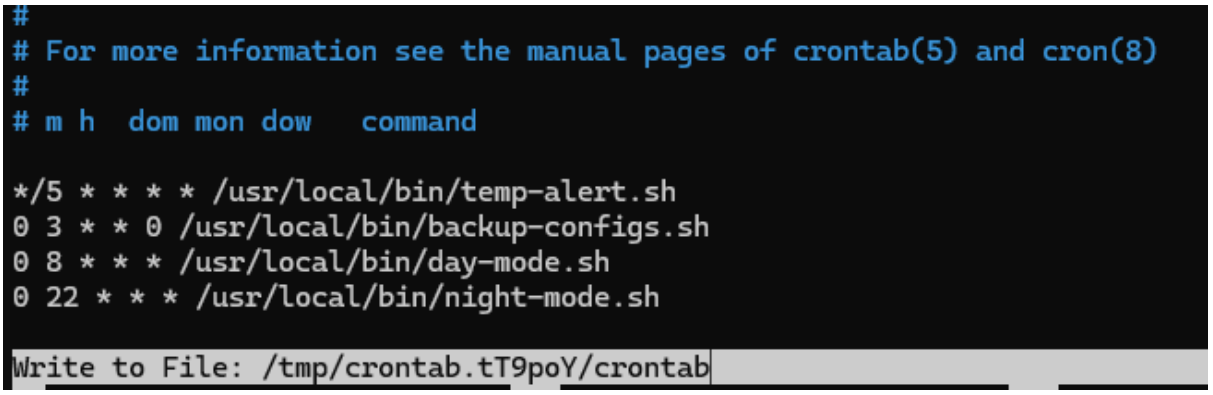
Night (powersave mode)

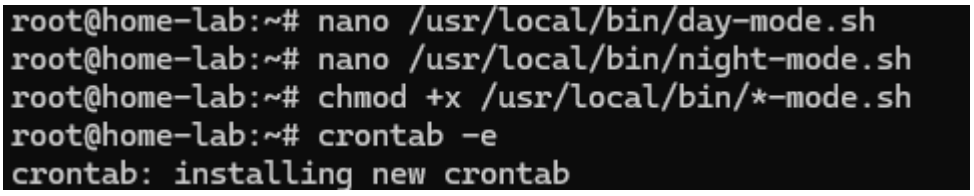
1- nano /usr/local/bin/night-mode.sh
 #!/bin/bash
for c in /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor; do
  echo powersave > $c
done



adding executable permissions:
chmod +x /usr/local/bin/*-mode.sh

writing cron





——————

Hardening

SSH

nano /etc/ssh/sshd_config

```
# Authentication:

#LoginGraceTime 2m
PermitRootLogin prohibit-password
#StrictModes yes
```

&

```
# To disable tunneled clear text passwords, change to "no" here!
PasswordAuthentication no
#PermitEmptyPasswords no
```

PermitRootLogin prohibit-password
PasswordAuthentication no

and then:

systemctl restart ssh

—

Firewall

apt install ufw
ufw default deny incoming
ufw default allow outgoing
ufw allow ssh
ufw allow 8006/tcp
ufw enable

```
Default incoming policy changed to 'deny'
(be sure to update your rules accordingly)
Default outgoing policy changed to 'allow'
(be sure to update your rules accordingly)
Rules updated
Rules updated (v6)
Rules updated
Rules updated (v6)
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
```

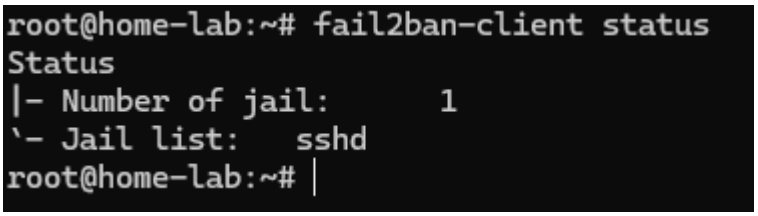
—

Setting up

Fail2Ban

apt install fail2ban
systemctl enable --now fail2ban

we can check using: fail2ban-client status



—

Setting up debian minimal on proxmox to run our docker stack.

Layout:

Proxmox host
 ├─ VM 100 – docker-host
 │   ├─ Portainer
 │   ├─ Monitoramento
 │   ├─ SIEM
 │
 └─ (future) VM 101 – labs/attack

CPU: **2 cores**

Sockets: 1

CPU: **host**

RAM: **4 GB**

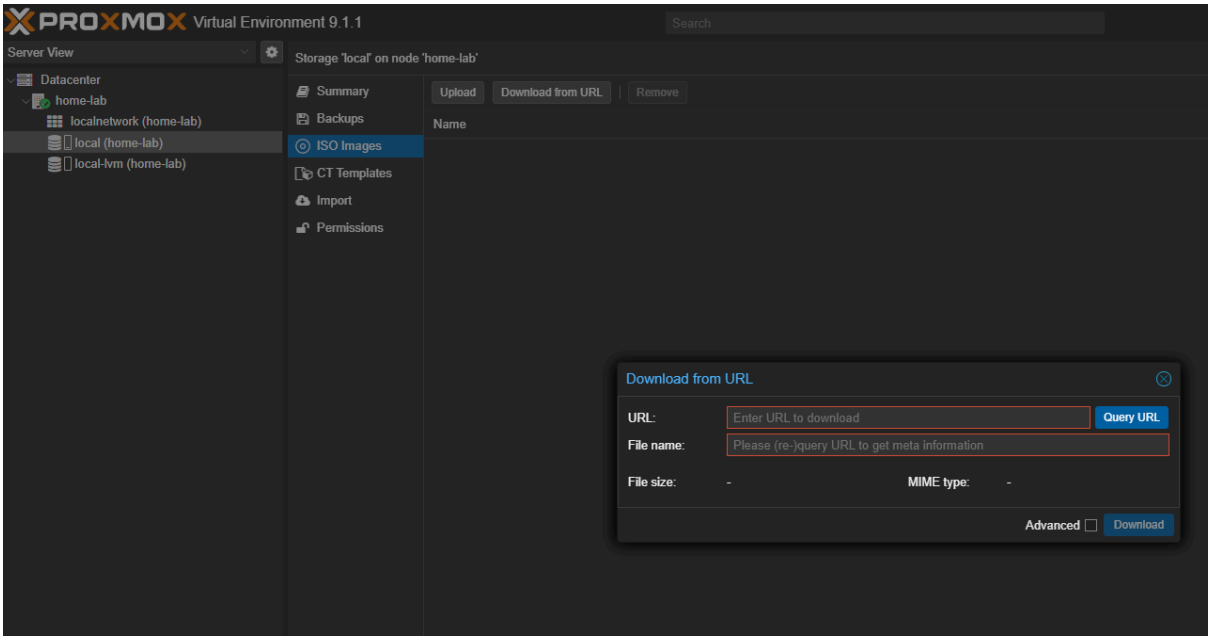
Ballooning: **ON**

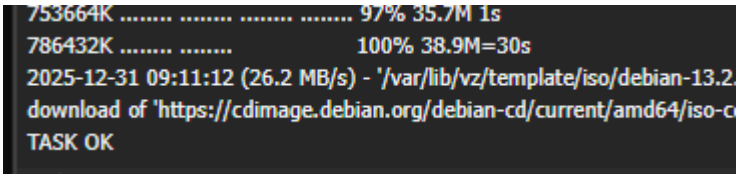
Disk: **40 GB**

BIOS: SeaBIOS

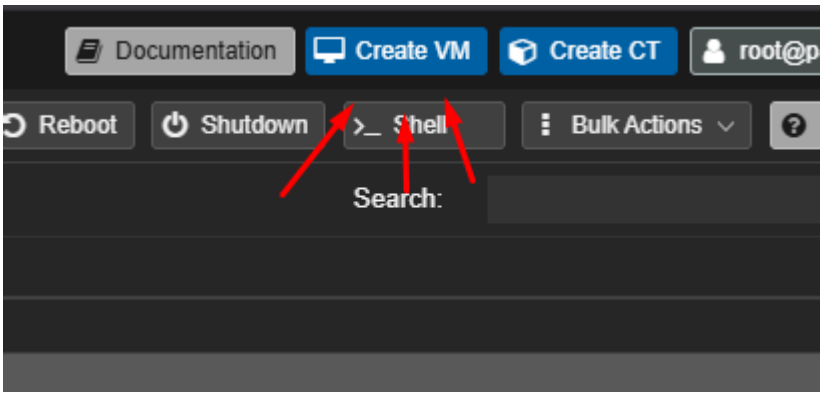
Machine: q35

1- adding debian 13 minimal iso:

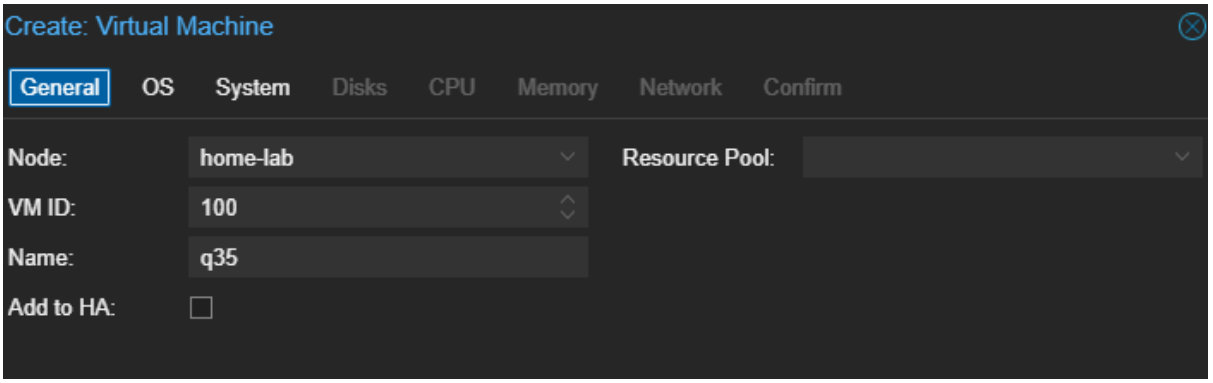




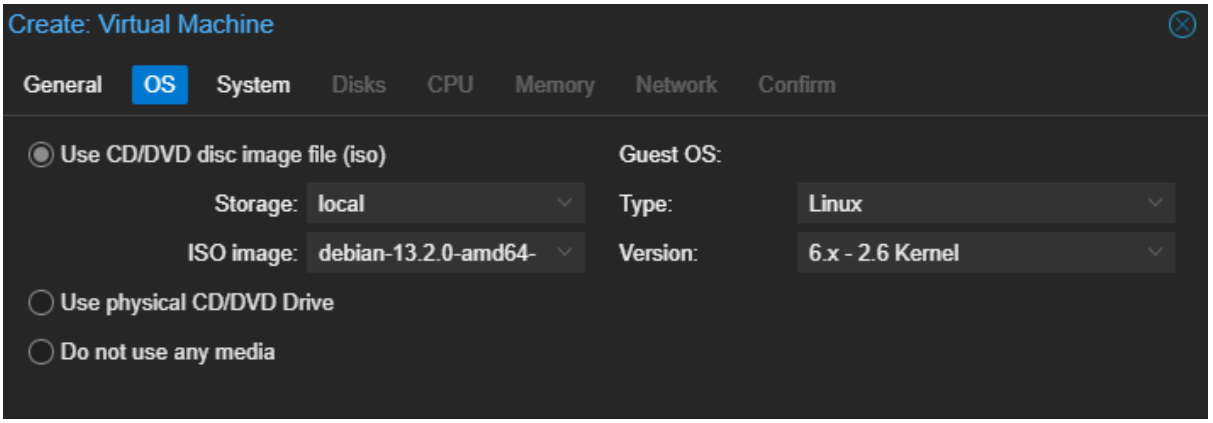
1- Creating our vm:



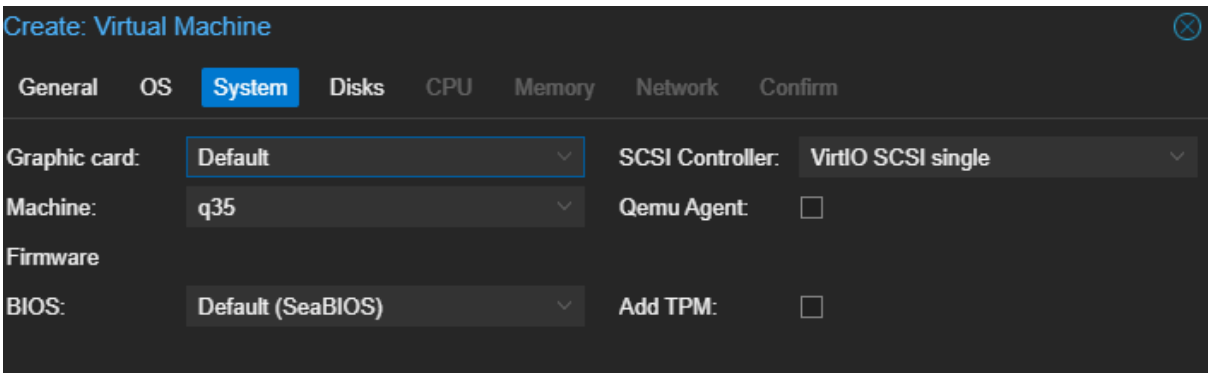
2-



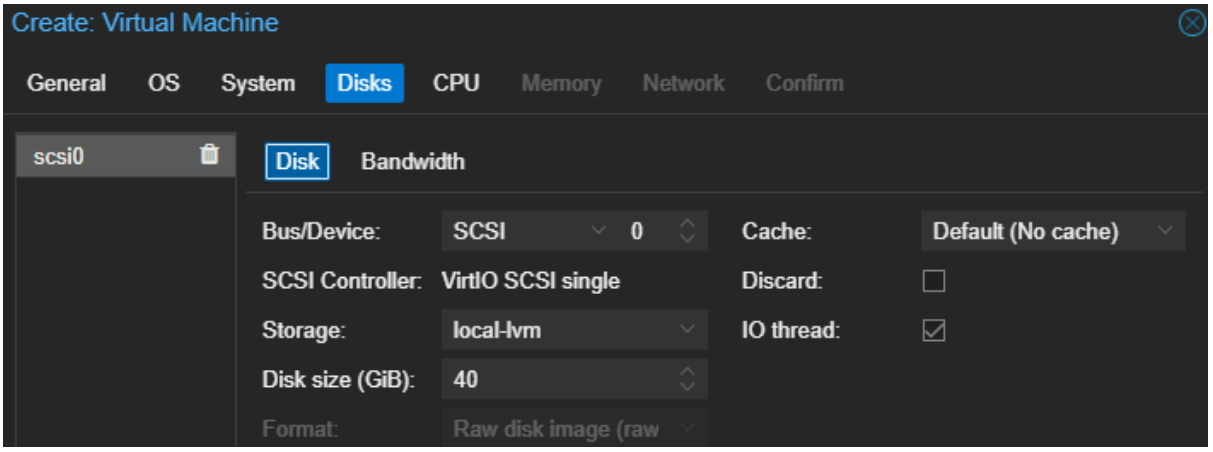
3-



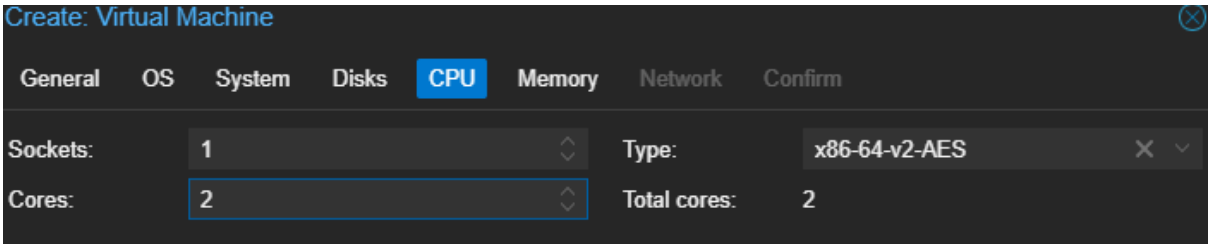
4-


Create: Virtual Machine
General
OS
System
Disks
CPU
Memory
Network
Confirm
Graphic card:
Default
SCSI Controller:
VirtIO SCSI single
Machine:
q35
Qemu Agent:
Firmware
BIOS:
Default (SeaBIOS)
Add TPM:

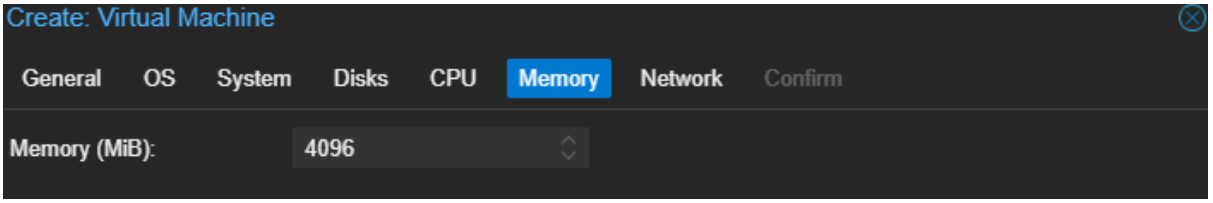
5-


Create: Virtual Machine
General
OS
System
Disks
CPU
Memory
Network
Confirm
scsi0
Disk
Bandwidth
Bus/Device:
SCSI
0
Cache:
Default (No cache)
SCSI Controller:
VirtIO SCSI single
Discard:
Storage:
local-lvm
IO thread:
Disk size (GiB):
40
Format:
Raw disk image (raw

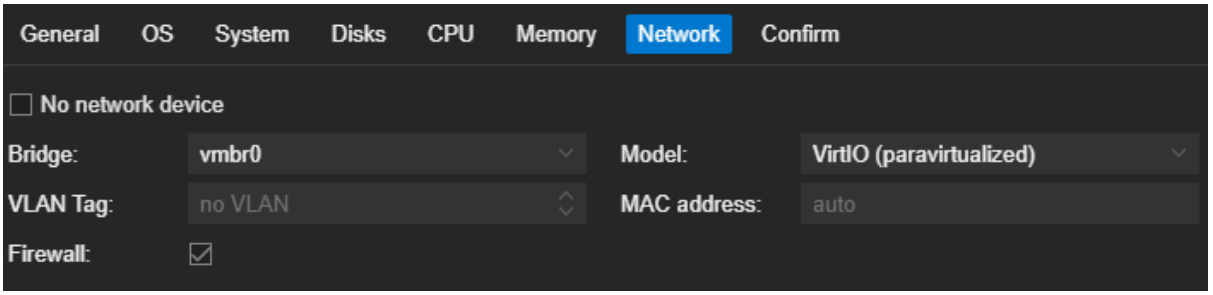
6-


Create: Virtual Machine
General
OS
System
Disks
CPU
Memory
Network
Confirm
Sockets:
1
Type:
x86-64-v2-AES
Cores:
2
Total cores:
2

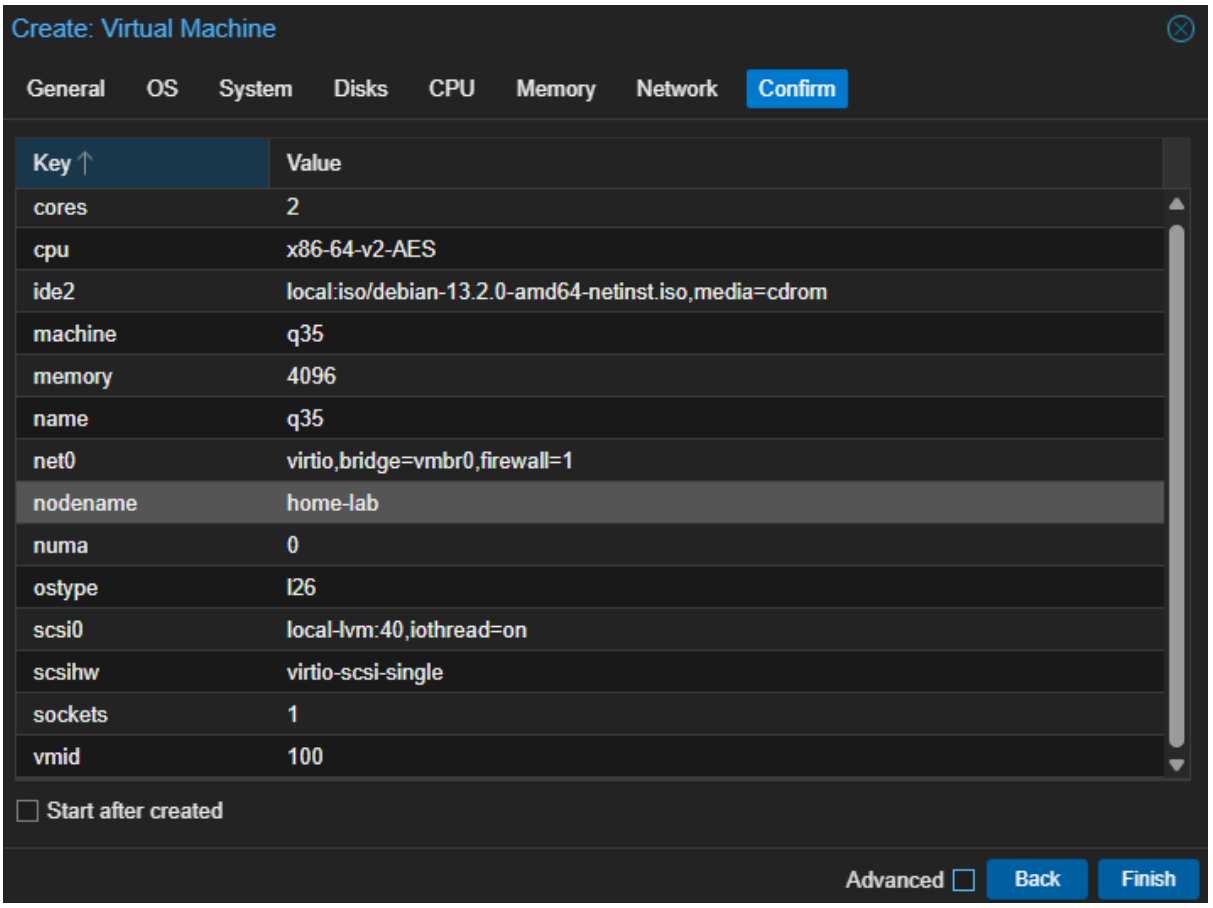
7-


Create: Virtual Machine
General
OS
System
Disks
CPU
Memory
Network
Confirm
Memory (MiB):
4096

8-



Finish



—

Installing docker and some optimizations

updating and installing packages:

apt update && apt upgrade -y
apt install ca-certificates curl gnupg htop -y

setting up Docker:

curl -fsSL https://get.docker.com | sh

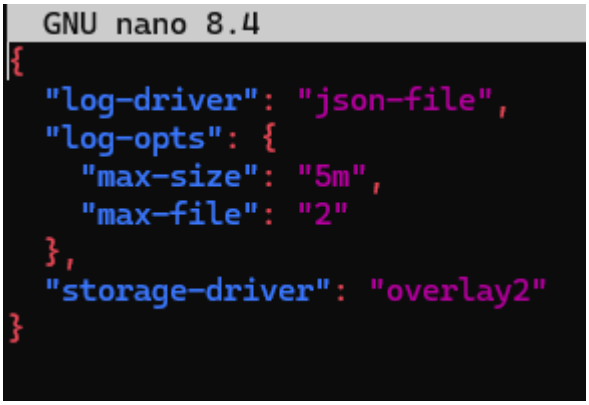
```
Client: Docker Engine - Community
 Version:           29.1.3
 API version:       1.52
 Go version:        go1.25.5
 Git commit:        f52814d
 Built:             Fri Dec 12 14:49:42 2025
 OS/Arch:           linux/amd64
 Context:           default

Server: Docker Engine - Community
 Engine:
  Version:          29.1.3
  API version:      1.52 (minimum version 1.44)
  Go version:       go1.25.5
  Git commit:       fbf3ed2
  Built:            Fri Dec 12 14:49:42 2025
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          v2.2.1
  GitCommit:        dea7da592f5d1d2b7755e3a161be07f43fad8f75
 runc:
  Version:          1.3.4
  GitCommit:        v1.3.4-0-gd6d73eb8
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0

================================================================================

To run Docker as a non-privileged user, consider setting up the
Docker daemon in rootless mode for your user:

    dockerd-rootless-setuptool.sh install

Visit https://docs.docker.com/go/rootless/ to learn about rootless mode.


To run the Docker daemon as a fully privileged service, but granting non-root
users access, refer to https://docs.docker.com/go/daemon-access/

WARNING: Access to the remote API on a privileged Docker daemon is equivalent
         to root access on the host. Refer to the 'Docker daemon attack surface'
         documentation for details: https://docs.docker.com/go/attack-surface/

================================================================================

root@debian:/home/sadly# docker -v
Docker version 29.1.3, build f52814d
root@debian:/home/sadly#
```

adding user:

usermod -aG docker homelab

making smaller logs:

json



nano /etc/docker/daemon.json

```
{
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "5m",
    "max-file": "2"
  },
  "storage-driver": "overlay2"
}
```

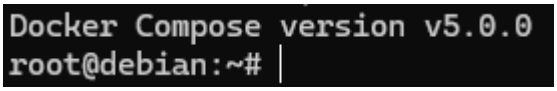
^O + ^X

And then restart docker:

systemctl restart docker

Docker compose:

apt install docker-compose-plugin -y
docker compose version



—

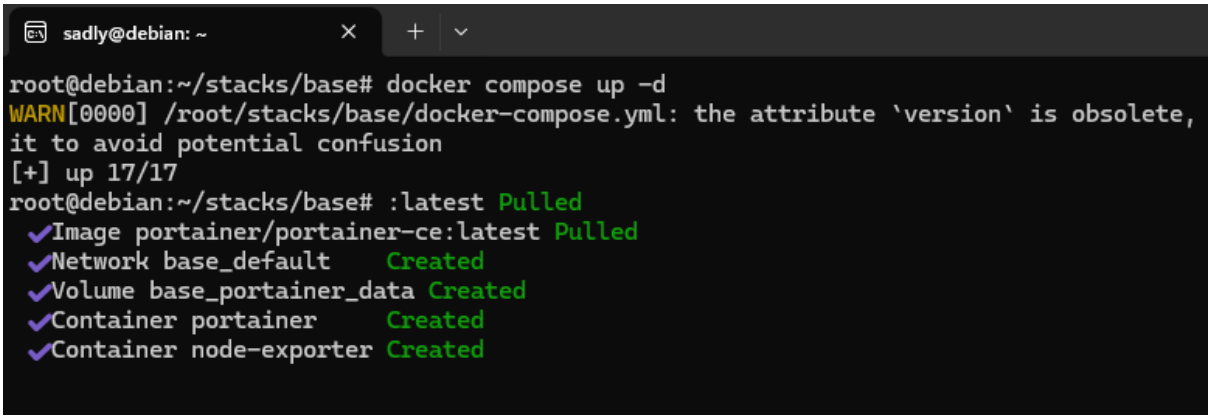
Creating our base stack with portainer:

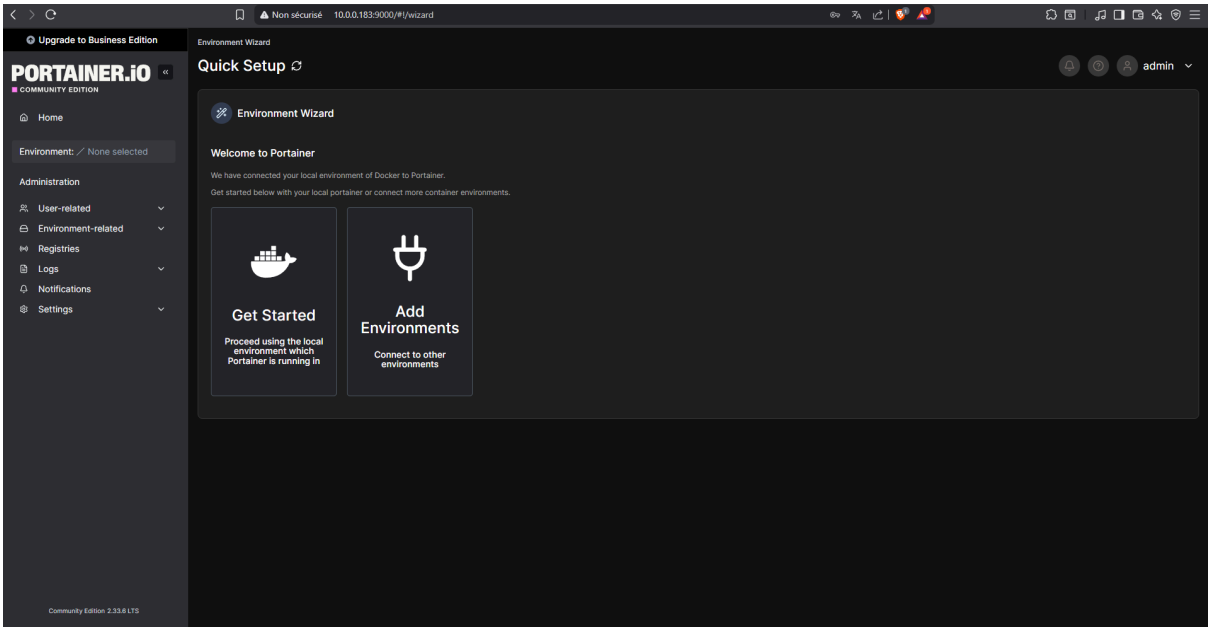
mkdir -p ~/stacks/base
cd ~/stacks/base
nano docker-compose.yml

```
---------------------------------------------------------------
version: "3.9"

services:
  portainer:
    image: portainer/portainer-ce:latest
    container_name: portainer
    restart: unless-stopped
    ports:
      - "9000:9000"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - portainer_data:/data
    deploy:
      resources:
        limits:
          cpus: "0.75"
          memory: 384M

  node-exporter:
    image: prom/node-exporter:latest
    container_name: node-exporter
    restart: unless-stopped
    ports:
      - "9100:9100"
    deploy:
      resources:
        limits:
          cpus: "0.25"
          memory: 64M

volumes:
  portainer_data:
---------------------------------------------------------------
```

Start:

```
docker compose up -d
```





—

GRAFANA

Architecture:

[Proxmox Host]
 ├─ Node Exporter (host)
 ├─ Docker
 │   ├─ Prometheus
 │   └─ cAdvisor
 └─ Grafana

Update and install prometheus node exporter

apt update
apt install prometheus-node-exporter -y

2. Setting up cAdvisor

create directory:
mkdir -p /opt/monitoring/cadvisor
cd /opt/monitoring/cadvisor

docker-compose.yml

```
version: "3.8"

services:
  cadvisor:
    image: gcr.io/cadvisor/cadvisor:v0.49.1
    container_name: cadvisor
    restart: unless-stopped
    ports:
      - "8080:8080"
    volumes:
      - /:/rootfs:ro
      - /var/run:/var/run:ro
      - /sys:/sys:ro
      - /var/lib/docker/:/var/lib/docker:ro
    deploy:
      resources:
        limits:
          cpus: "0.50"
          memory: 256M
```

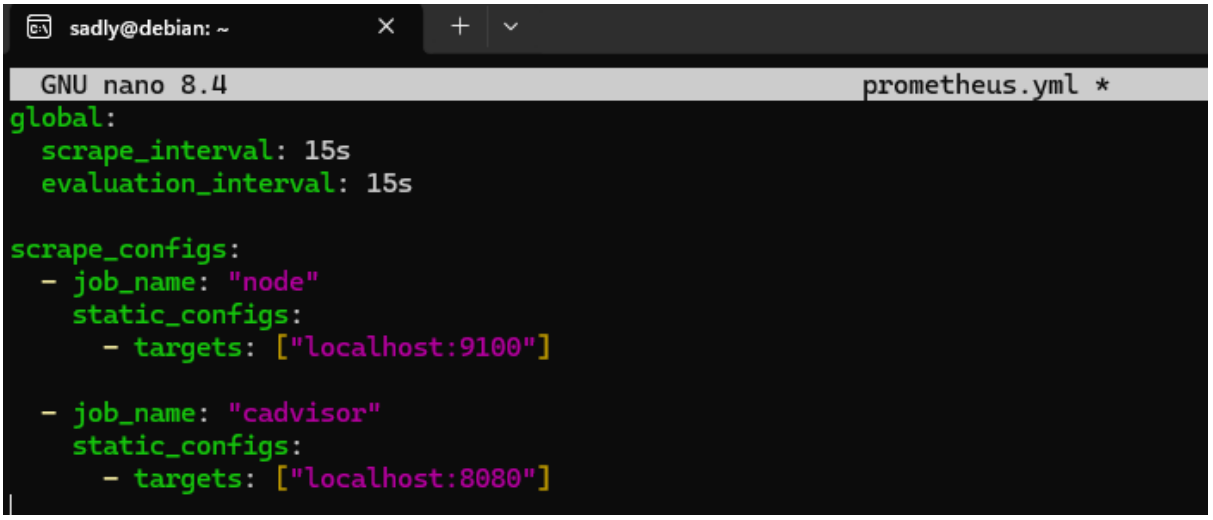
setting up the docker compose up:

docker compose up -d

prometheus lateral collector:

mkdir -p /opt/monitoring/prometheus/{data,config}
cd /opt/monitoring/prometheus

prometheus.yml
```
global:
  scrape_interval: 15s
  evaluation_interval: 15s

scrape_configs:
  - job_name: "node"
    static_configs:
      - targets: ["localhost:9100"]

  - job_name: "cadvisor"
    static_configs:
      - targets: ["localhost:8080"]
```



docker-compose.yml

```
version: "3.8"

services:
  prometheus:
    image: prom/prometheus:v2.52.0
    container_name: prometheus
    restart: unless-stopped
    ports:
      - "9090:9090"
    volumes:
      - ./config/prometheus.yml:/etc/prometheus/prometheus.yml:ro
```

```
    - ./data:/prometheus
  command:
    - "--storage.tsdb.retention.time=7d"
    - "--storage.tsdb.retention.size=1GB"
  deploy:
    resources:
      limits:
        cpus: "1.0"
        memory: 512M
```

setting up prometheus:

docker compose up -d

Useful queries:
CPU host:
100 - (avg by(instance)(irate(node_cpu_seconds_total{mode="idle"}[5m])) * 100)

RAM:
(node_memory_MemTotal_bytes - node_memory_MemAvailable_bytes) /
node_memory_MemTotal_bytes * 100

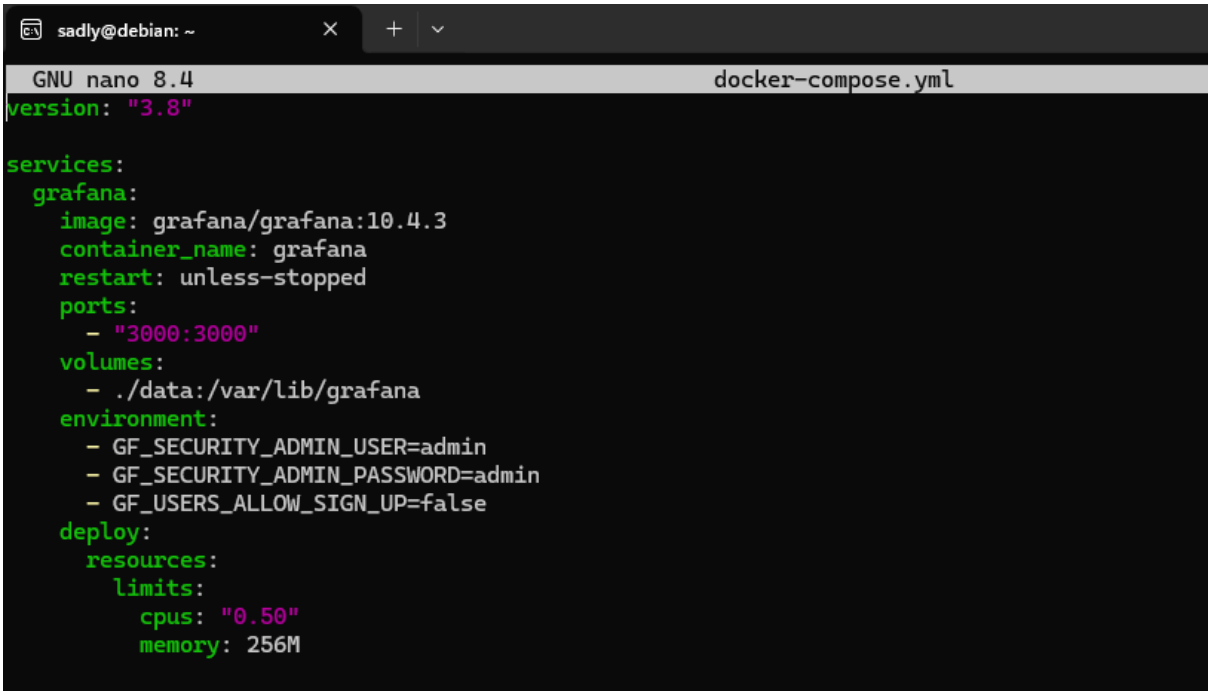
CPU Containers:
rate(container_cpu_usage_seconds_total[5m])

—

Setting up Grafana

mkdir -p /opt/monitoring/grafana/data
cd /opt/monitoring/grafana

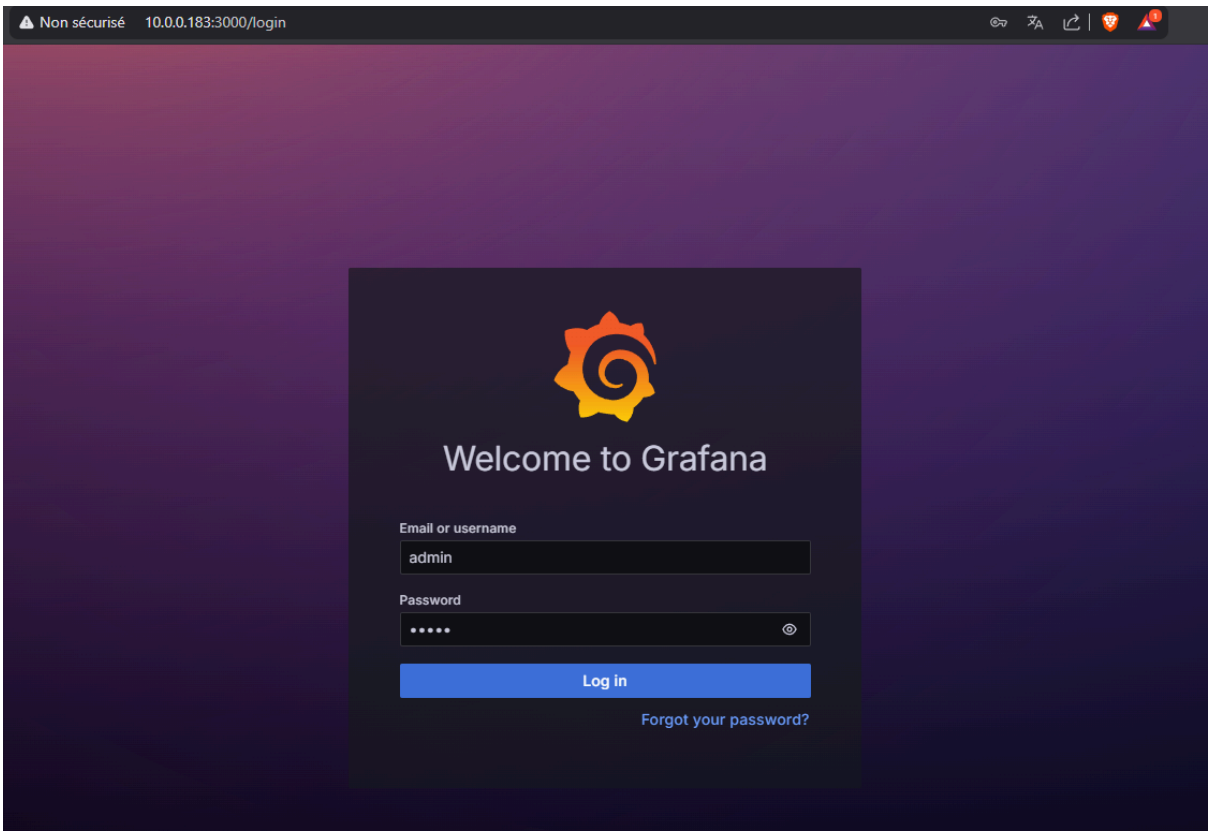
nano docker-compose.yml




```
version: "3.8"

services:
  grafana:
    image: grafana/grafana:10.4.3
    container_name: grafana
    restart: unless-stopped
    ports:
      - "3000:3000"
    volumes:
      - ./data:/var/lib/grafana
    environment:
      - GF_SECURITY_ADMIN_USER=admin
      - GF_SECURITY_ADMIN_PASSWORD=admin
      - GF_USERS_ALLOW_SIGN_UP=false
    deploy:
      resources:
        limits:
          cpus: "0.50"
          memory: 256M
```

Starting up:

```
docker compose up -d
```



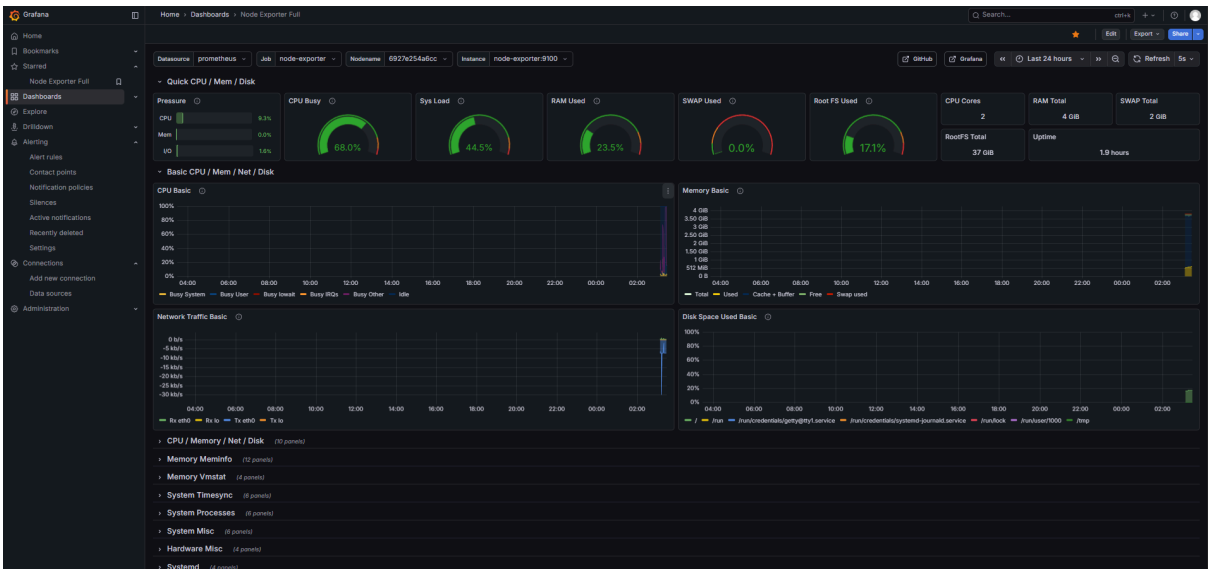
connecting Grafana with Prometheus:

Connections → Data Sources → Add data source → Prometheus

Importing dashboard:

Home > Dashboards > Import dashboard

Dashboard ID: 1860



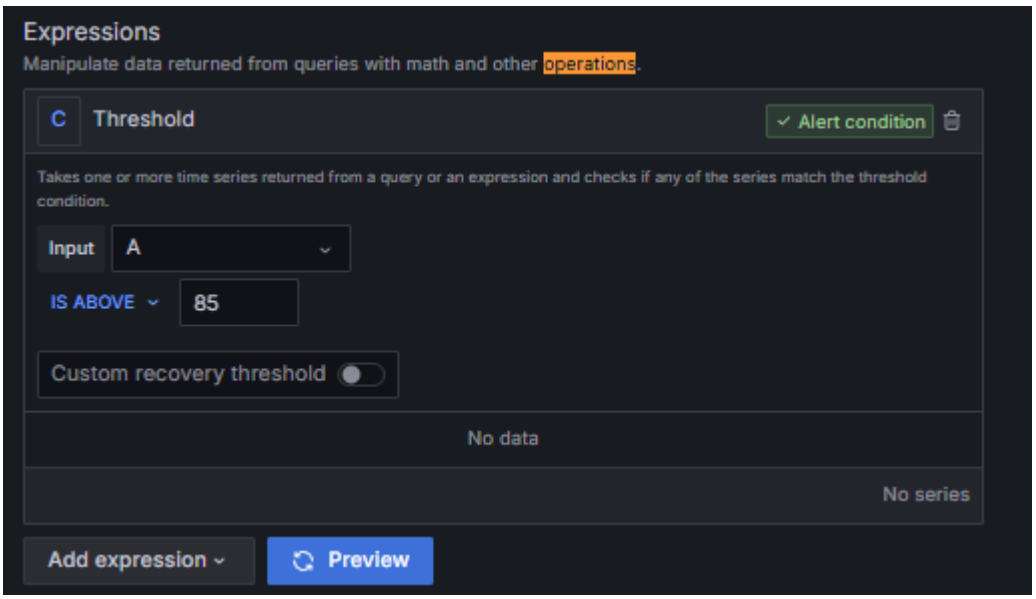
Creating alerts:

Home > Alerting > Alert Rules > New alert rule

Condition:

```
100 - (avg by (instance)
(rate(node_cpu_seconds_total{mode="idle"}[5m])) * 100)
```

Expressions:



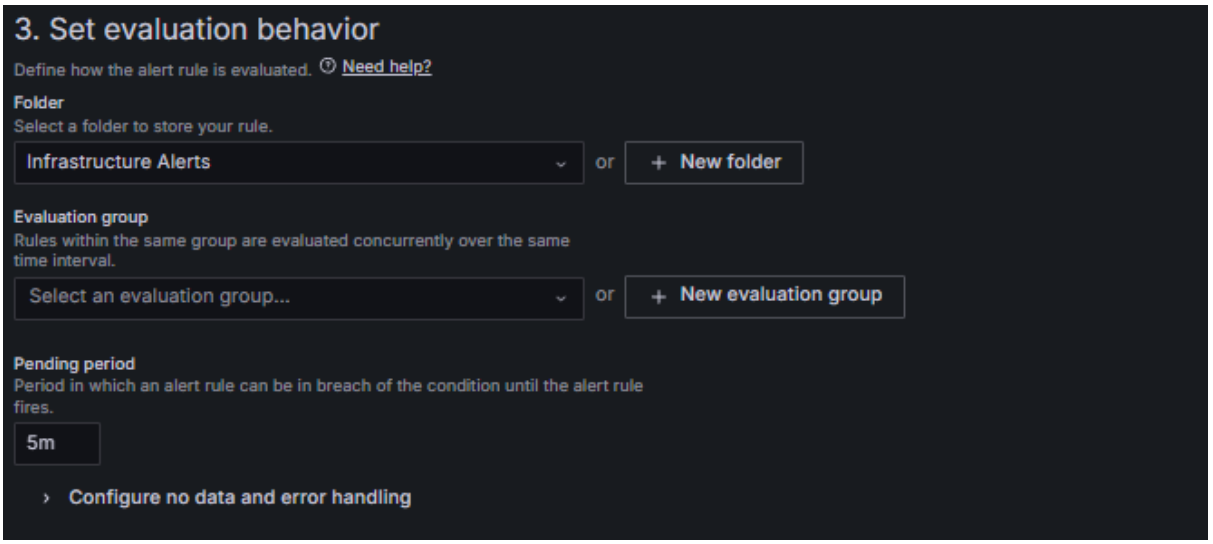
Input: A
IS ABOVE: 85

A (Prometheus query) → C (Threshold: Is above 85)

Set as alert condition.

3. Set evaluation behavior

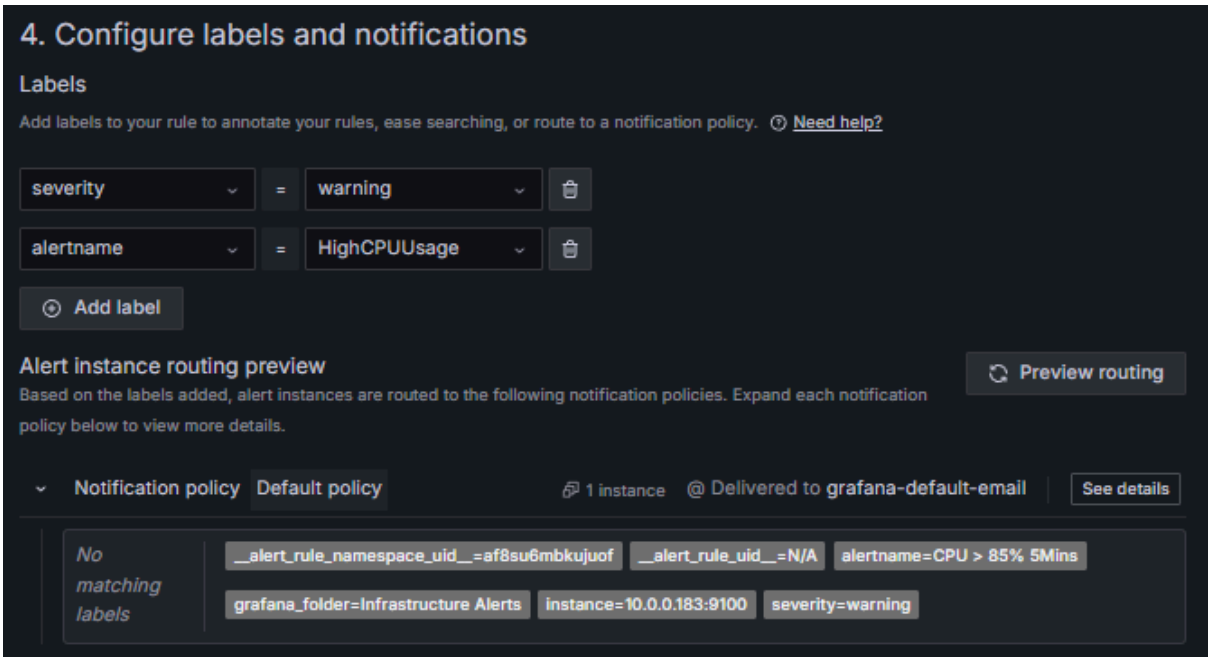
create new folder called Infrastructure Alerts



Evaluation group= Default : 5m
Pending period= 5m

4. Configure Labels and notifications

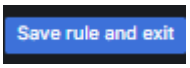
key: severity
Value: warning
key: alertname
Value: HighCPUUsage



7. Add annotations

summary: High CPU usage on {{ $labels.instance }}
description: CPU usage is at {{ $values.A }}% for 5 minutes

Save rule and exit.



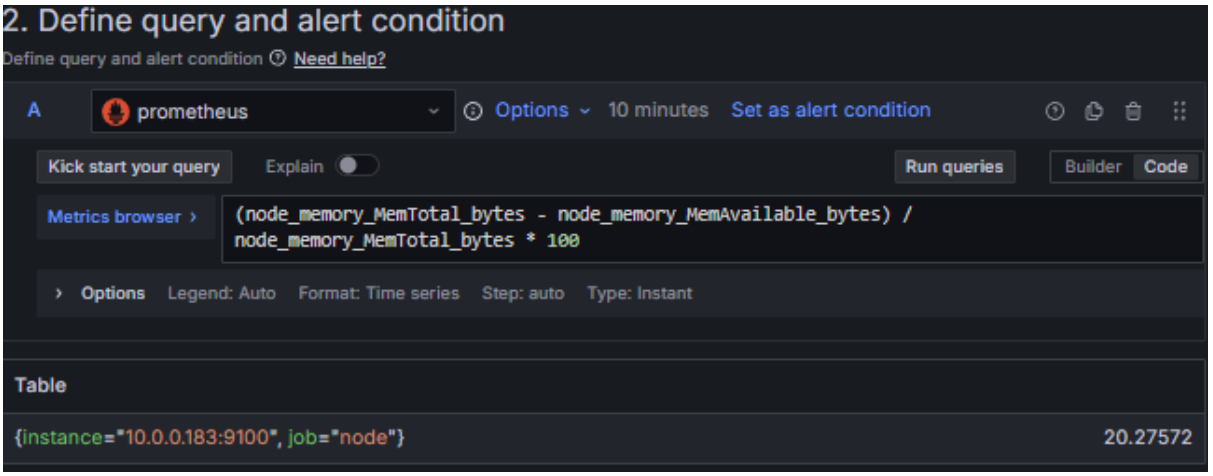
__

RAM:
RAM > 85% for 5 minutes
Query A: (node_memory_MemTotal_bytes -
node_memory_MemAvailable_bytes) / node_memory_MemTotal_bytes * 100

Threshold C: Is above 85

Labels: severity=warning, alertname=HighMemoryUsage

Summary: High memory usage on {{ $labels.instance }}

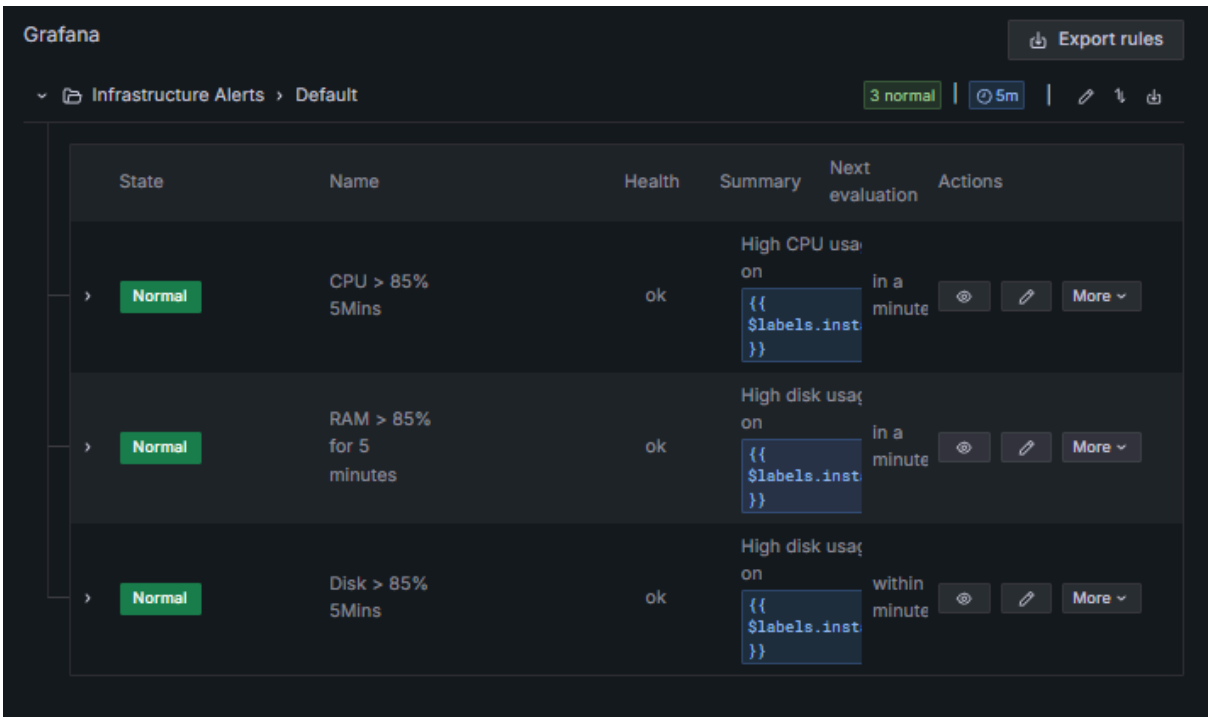


Disk:
Query A: (node_filesystem_size_bytes{mountpoint="/"} - node_filesystem_free_bytes{mountpoint="/"}) / node_filesystem_size_bytes{mountpoint="/"} * 100

Threshold C: Is above 85

Labels: severity=warning, alertname=HighDiskUsage

Summary: High disk usage on {{ $labels.instance }}



—-

Stack de Monitoramento

• Node Exporter: métricas do host
• cAdvisor: métricas de containers
• Prometheus: coleta central
• Grafana: visualização e alertas

Consumo médio:
• RAM: ~700 MB total
• CPU: <5% idle

Dashboards:
• Node Exporter Full (ID 1860)
• Docker cAdvisor (ID 14282)

Alertas:
• CPU > 85% por 5 min
• RAM > 85% por 5 min
• DISK > 85% por 5 min

—-

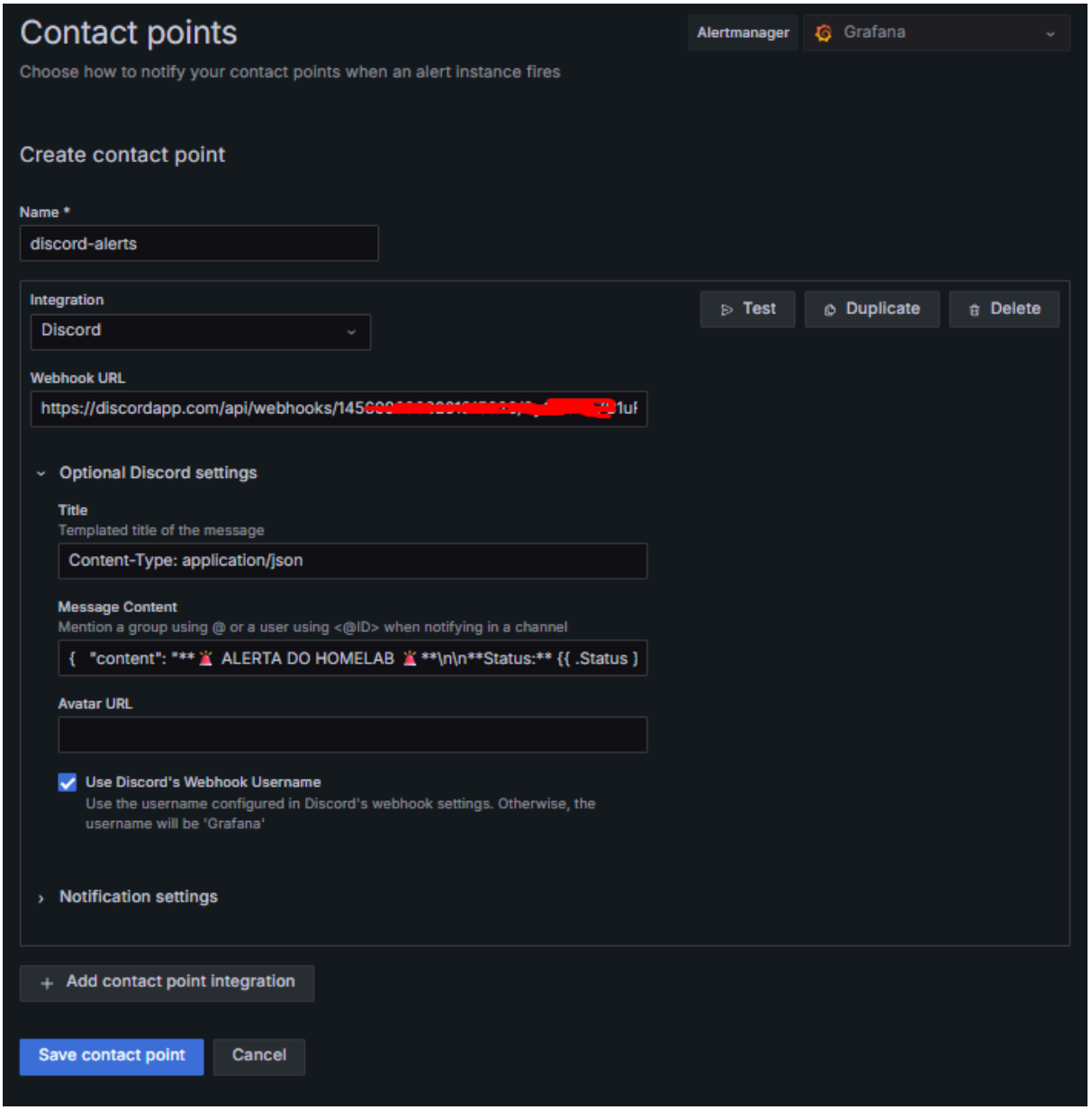
Discord Webhook contact point

Alerting → Contact points → New contact point

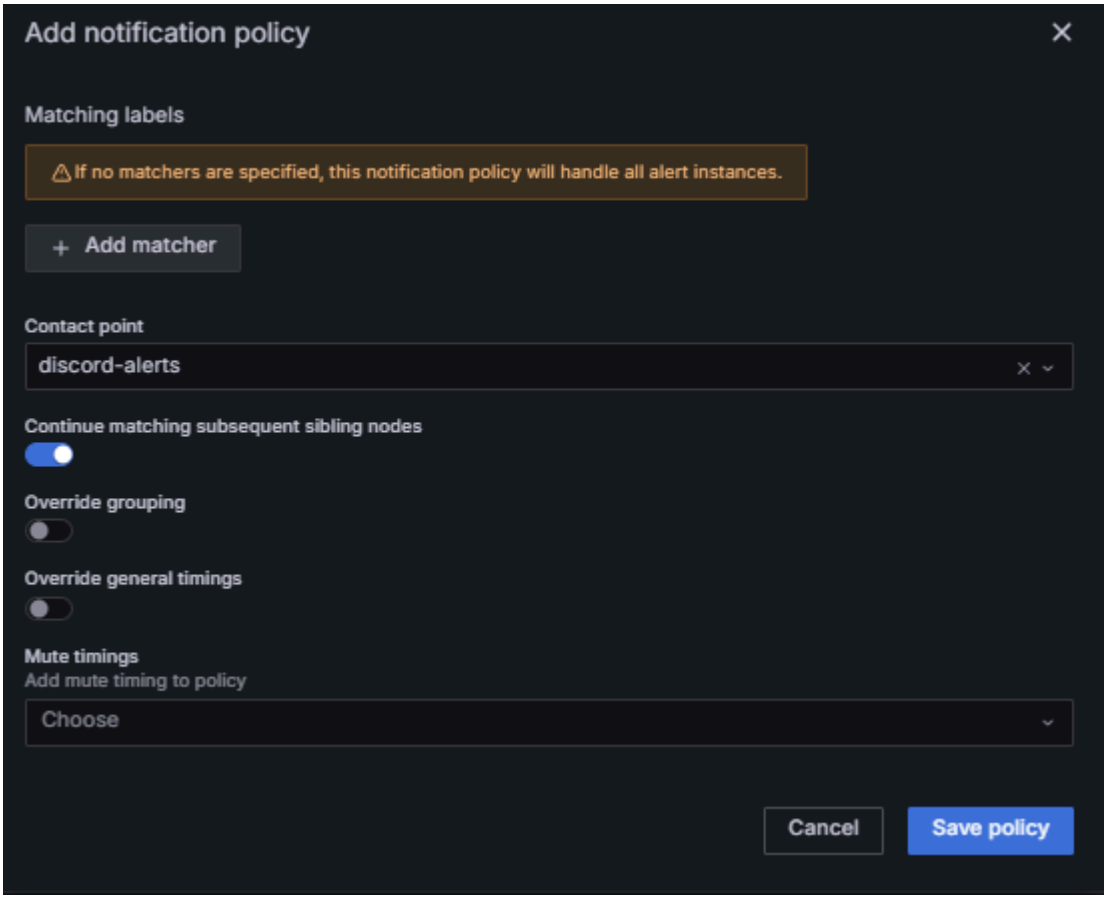
Custom HTTP Headers:

Content-Type: application/json

```
{
  "content": "🚨 **ALERTA DO HOMELAB** 🚨\n\n**Status:** {{
.Status }}\n**Alerta:** {{ .Annotations.summary }}\n**Descrição:**
{{ .Annotations.description }}\n**Valor:** {{ .Values.A
}}\n**Instância:** {{ .Labels.instance }}\n**Início:** {{
.StartsAt.Format \"2006-01-02 15:04:05\" }}"
}
```



setting up notification policy:



—

Instead of the grafano webhook service i will utilise the alertmanager docker.

Setting up Alertmanager

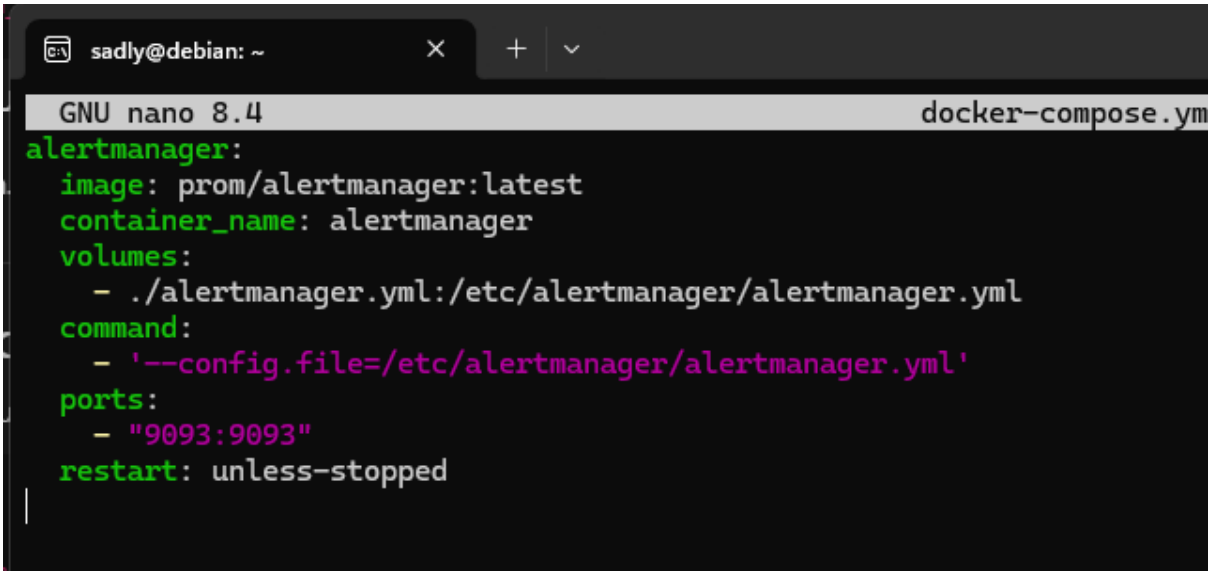
nano alertmanager.yml

```
global:
  resolve_timeout: 5m

route:
  receiver: 'discord-notifications'
  group_wait: 10s
  group_interval: 30s
  repeat_interval: 4h

receivers:
  - name: 'discord-notifications'
    webhook_configs:
      - url: 'https://discord.com/api/webhooks/ID/TOKEN'
```

```
        send_resolved: true
```

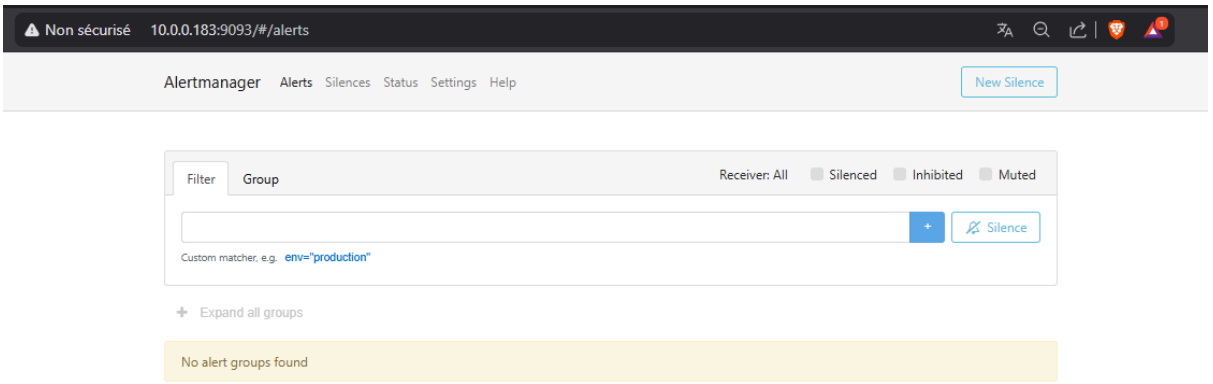
```
nano docker-compose.yml
```

```
alertmanager:
  image: prom/alertmanager:latest
  container_name: alertmanager
  volumes:
    - ./alertmanager.yml:/etc/alertmanager/alertmanager.yml
  command:
    - '--config.file=/etc/alertmanager/alertmanager.yml'
  ports:
    - "9093:9093"
  restart: unless-stopped
```




```
Running alertmanager
```

```
docker compose up -d alertmanager
```

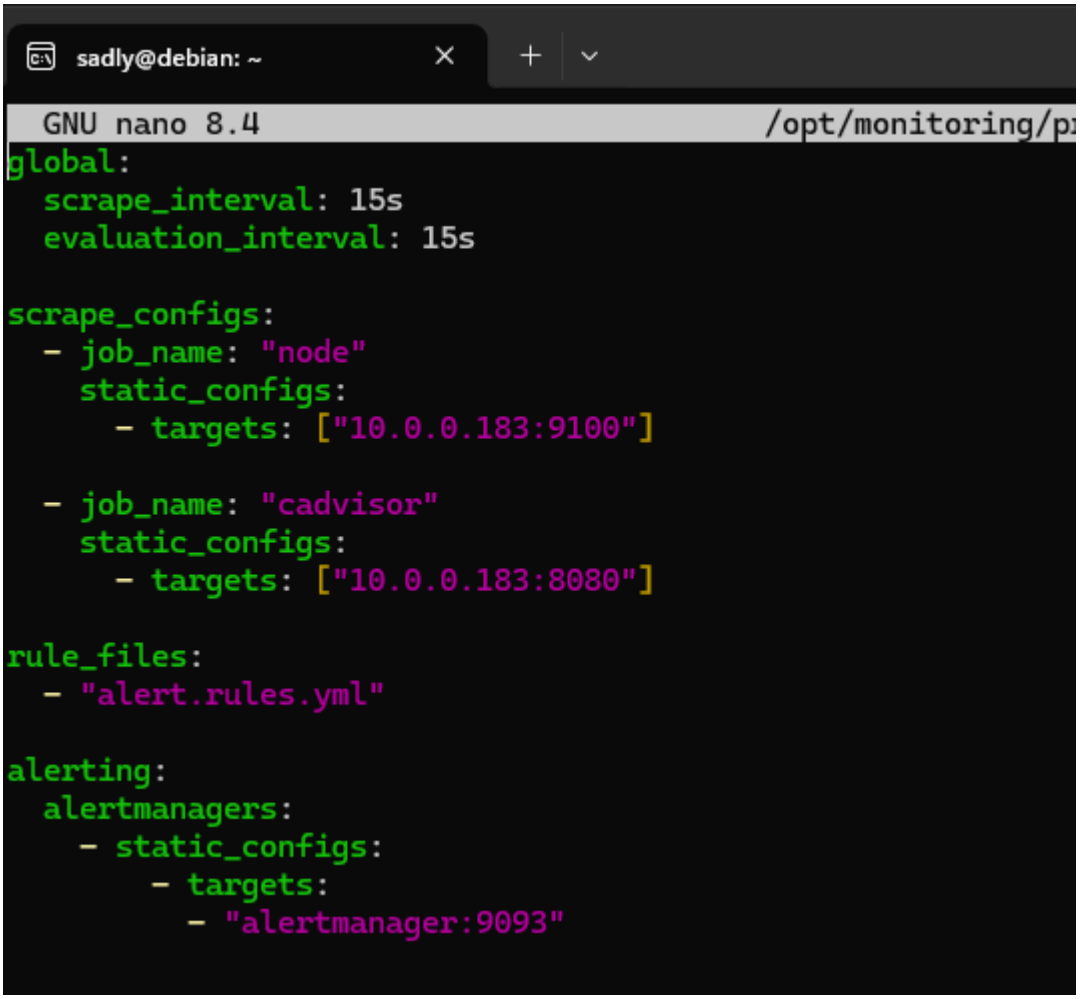


we have to configure prometheus to send discord notifications via webhook.

nano /opt/monitoring/prometheus/alert.rules.yml

```
groups:
- name: test-alerts
  rules:
  - alert: InstanceDown
    expr: up == 0
    for: 10s
    labels:
      severity: critical
    annotations:
      summary: "Instance Down"
      description: "Target {{ $labels.instance }} is offline!"
```

nano /opt/monitoring/prometheus/prometheus.yml



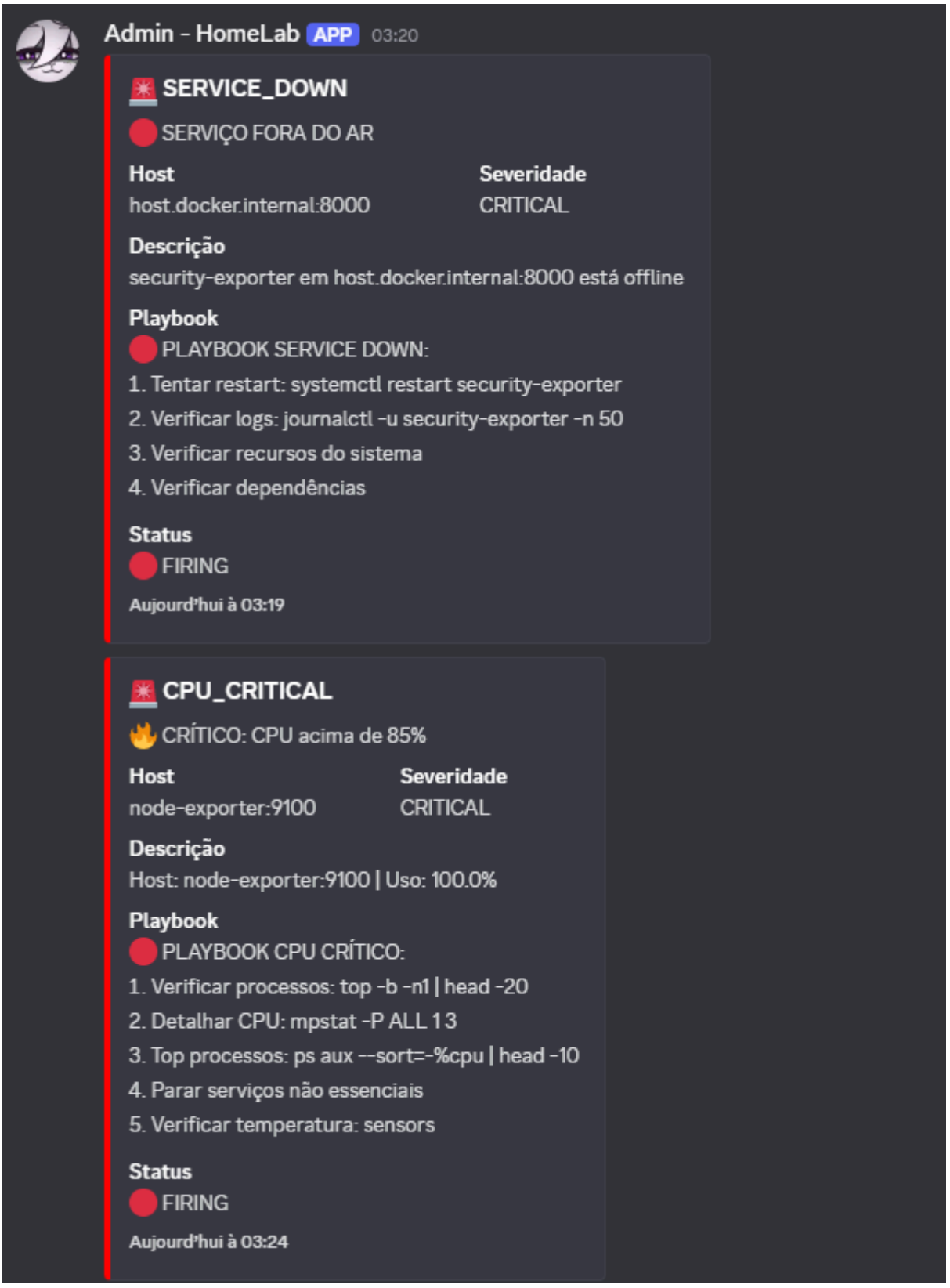

```
  GNU nano 8.4
/opt/monitoring/prometheus/prometheus.yml
global:
  scrape_interval: 15s
  evaluation_interval: 15s

scrape_configs:
  - job_name: "node"
    static_configs:
      - targets: ["10.0.0.183:9100"]

  - job_name: "cadvisor"
    static_configs:
      - targets: ["10.0.0.183:8080"]

rule_files:
  - "alert.rules.yml"

alerting:
  alertmanagers:
```

```
  - static_configs:
      - targets:
        - "alertmanager:9093"
```

Restart prometheus

```
docker compose restart prometheus
```

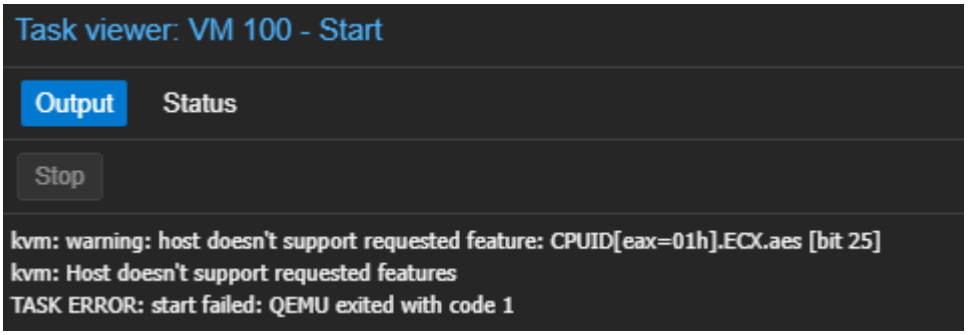
Discord webhook notification:


Admin - HomeLab APP 03:20
SERVICE_DOWN
SERVIÇO FORA DO AR
Host
host.docker.internal:8000
Severidade
CRITICAL
Descrição
security-exporter em host.docker.internal:8000 está offline
Playbook
PLAYBOOK SERVICE DOWN:
1. Tentar restart: systemctl restart security-exporter
2. Verificar logs: journalctl -u security-exporter -n 50
3. Verificar recursos do sistema
4. Verificar dependências
Status
FIRING
Aujourd'hui à 03:19
CPU_CRITICAL
CRÍTICO: CPU acima de 85%
Host
node-exporter:9100
Severidade
CRITICAL
Descrição
Host: node-exporter:9100 | Uso: 100.0%
Playbook
PLAYBOOK CPU CRÍTICO:
1. Verificar processos: top -b -n1 | head -20
2. Detalhar CPU: mpstat -P ALL 1 3
3. Top processos: ps aux --sort=-%cpu | head -10
4. Parar serviços não essenciais
5. Verificar temperatura: sensors
Status
FIRING
Aujourd'hui à 03:24

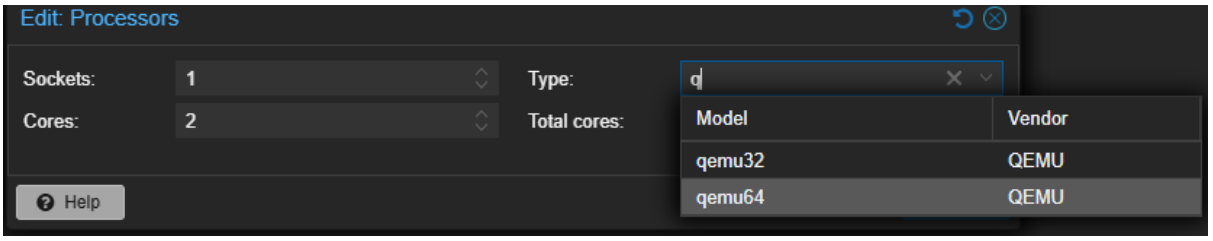
Guia 2

Erros i encountered and troubleshooted—

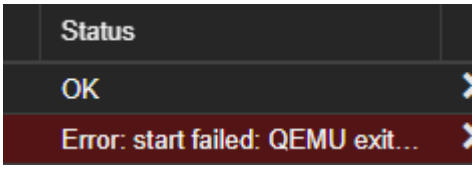
while starting vm100 for the first time:



fix:

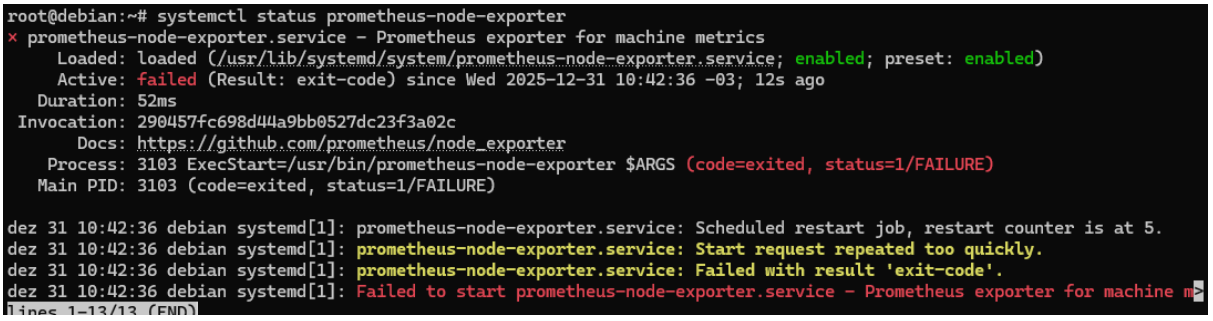


changing cpu type to qemu64
evidence:

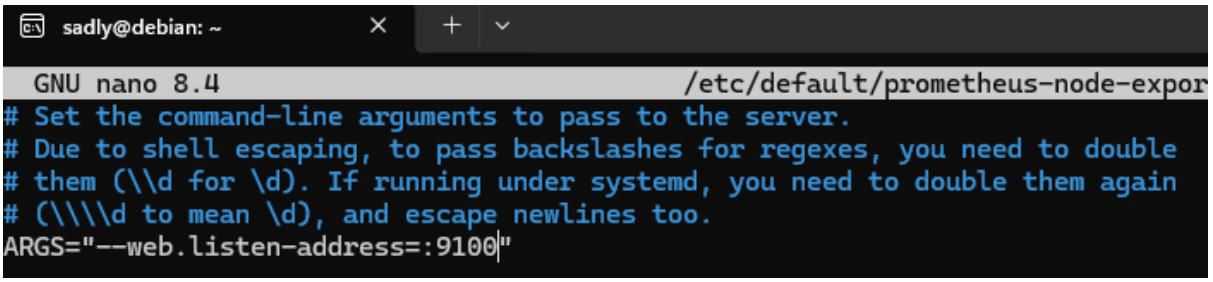


—

prometheus error



Fix:



sudo nano /etc/default/prometheus-node-exporter

Evidence: