

Building Discord-Based Alert Systems

Using Docker, Portainer, and
Uptime Kuma for Real-Time Monitoring

Rune Djurup

Table of Contents

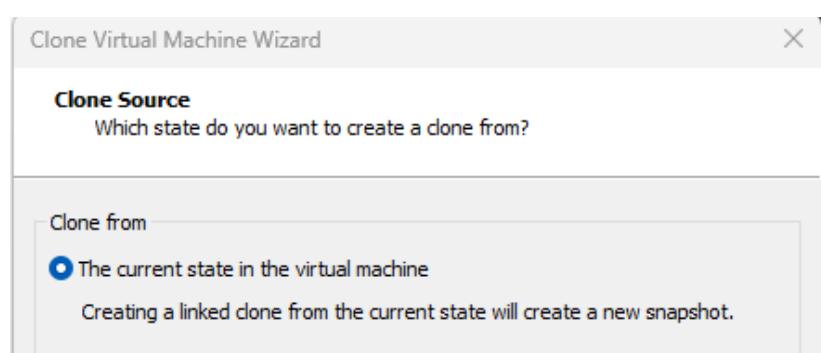
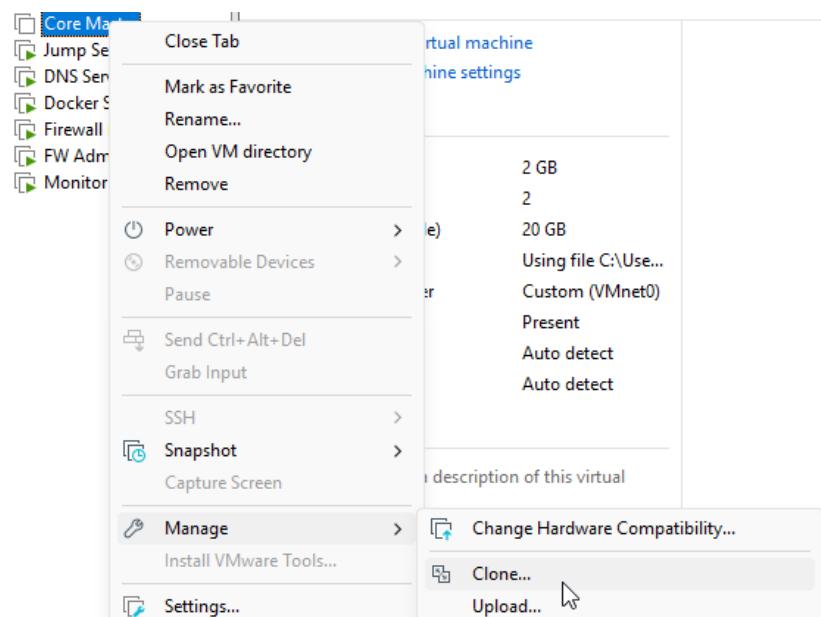
1. Opsætning af Monitor Server VM	1
Lav en klon af din Core Master og kald den Monitor Server.....	1
2. Konfiguration af netværkskortet på VM	3
Nmtui	3
3. Installation af Docker	4
4. Installation af Portainer	7
5. Installation af Uptime Kuma Container	10
6. Opsætning af Discord Server til Uptimekuma.....	13
Webhook	15
7. Opsætning af alarmer i Uptimekuma	18
Alarm.....	18
Notifikation	20
DNS Server Alarm af typen ping.....	23
8. Test af DNS-alarmerne	25
9. TEC H2 Webserver – Total Overvågning	29

1. Opsætning af Monitor Server VM

For at sikre os at vores monitorerings server kan blive ved at monitorere, selvom vores primære docker server går ned, så laver vi en separat docker server til kun monitorering.

OBS! Hvis du er logget på med egen bruger, så husk sudo foran dine kommandoer.

Lav en klon af din Core Master og kald den Monitor Server.



Start nu din nye Monitor Server VM og få ændret hostname på den med det samme.

```
Rocky Linux 9.5 (Blue Onyx)
Kernel 5.14.0-503.38.1.el9_5.x86_64 on an x86_64

core login: _
```

```
core login: root
Password:
Last login: Wed Apr 23 13:03:57 on tty1
[root@core ~]# _
```

Hostname ændres via kommandoen: **hostnamectl set-hostname monitor**

Når hostname er ændret, så skriv exit til du ender ved loginskærmen, så skulle der gerne stå ”monitor login:”

```
Rocky Linux 9.5 (Blue Onyx)
Kernel 5.14.0-503.38.1.el9_5.x86_64 on an x86_64

monitor login: _
```

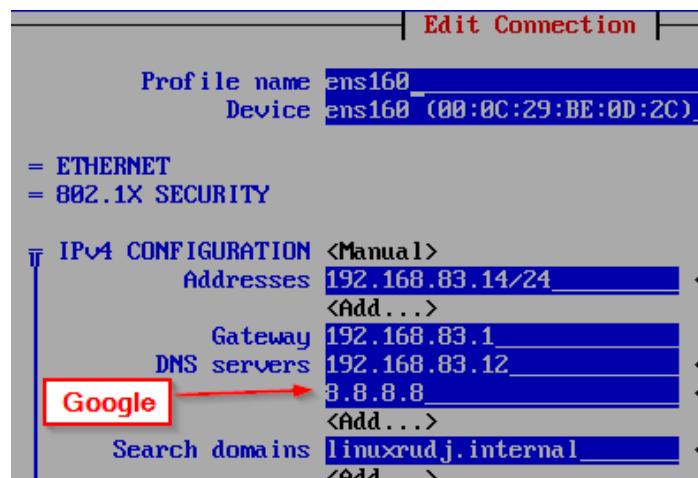
2. Konfiguration af netværkskortet på VM

Log på med root og sæt netværkskortet op.

Netværket konfigureres nemmest med værktøjet nmtui. Kommandoen er **nmtui**.

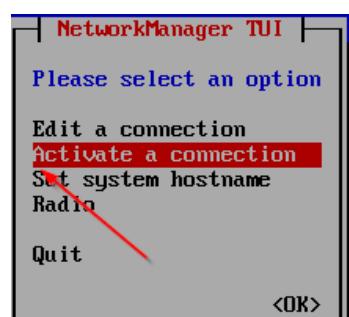
Sørg for at opsætte det korrekt i henhold til din IP-Plan. For en sikkerheds skyld, så lav en sekundær DNS som peger på Googles "8.8.8.8".

Nmtui

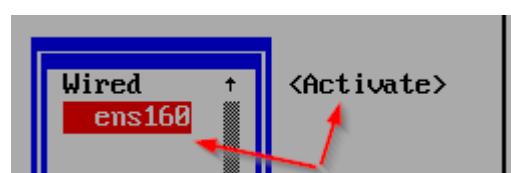


OBS! Husk at aktivere netværkskortet efter du har sat det op!

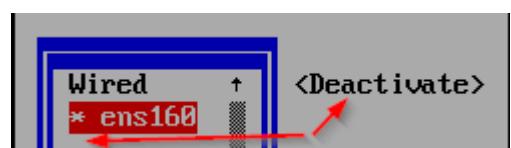
Vælg "Activate a connection"



Tryk på enter for at aktivere når du har valgt netværkskortet ens160.



Når netværkskortet er aktiveret, vil der være en lille stjerne foran ens160:



3. Installation af Docker

Nu er vi klar til at installere docker.

Vi starter med at installere alle eventuelle opdateringer via kommando'en:

dnf update -y

```
[root@monitor ~]# dnf update -y
Last metadata expiration check: 0:22:08 ago on Wed Apr 23 13:13:20 2025.
Dependencies resolved.
Nothing to do.
Complete!
```

Reboot efter opdatering via kommando'en:

reboot

Før vi starter docker installationen, skal vi lige sikre os at podman og buildah er fjernet fra rocky linux, ellers får vi fejl. Det gøres med følgende kommando:

dnf remove podman buildah -y

```
[root@monitor ~]# dnf remove podman buildah -y
No match for argument: podman
No match for argument: buildah
No packages marked for removal.
Dependencies resolved.
Nothing to do.
Complete!
```

Nu er vi klar til at hente den officielle docker pakke ned via kommando'en:

dnf config-manager --add-repo=https://download.docker.com/linux/centos/docker-ce.repo

```
[root@monitor ~]# dnf config-manager --add-repo=https://download.docker.com/linux/centos/docker-ce.repo
Adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
```

Nu kan docker installers via kommando'en:

dnf install -y docker-ce

```
Dependencies resolved.
=====
| Package           | Architecture | Version      | Repository | Size
=====
Installing:
| docker-ce         | x86_64       | 3:20.10.20-3.el9 | docker-ce-stable | 21 M
Installing dependencies:
| container-selinux | noarch      | 3:2.188.0-1.el9_0 | appstream     | 47 k
| containerd.io    | x86_64       | 1.6.8-3.1.el9   | docker-ce-stable | 32 M
| docker-ce-cli    | x86_64       | 1:20.10.20-3.el9 | docker-ce-stable | 29 M
| docker-ce-rootless-extras | x86_64       | 20.10.20-3.el9 | docker-ce-stable | 3.7 M
| fuse-overlayfs  | x86_64       | 1.9-1.el9_0     | appstream     | 71 k
| libslirp          | x86_64       | 4.4.0-7.el9     | appstream     | 68 k
| slirp4netns      | x86_64       | 1.2.0-2.el9_0   | appstream     | 46 k
Installing weak dependencies:
| docker-scan-plugin | x86_64       | 0.17.0-3.el9   | docker-ce-stable | 3.6 M
Transaction Summary
=====
Install 9 Packages
```

Hvis der opstår en container.io fejl når docker installers, så kør følgende commando:

```
dnf install docker-ce -allow-eraser -y
```

Nu kan vi starte og aktivere docker med følgende kommandoer:

```
systemctl start docker (starter docker)
```

```
[root@monitor ~]# systemctl start docker
[ 1027.586621] evm: overlay not supported
[ 1027.632441] Warning: Unmaintained driver is detected: nft_compat
[ 1027.778283] Warning: Unmaintained driver is detected: ip_set
[ 1027.902788] bridge: filtering via arp/ip/ip6tables is no longer available by default. Update your scripts to load br_netfilter if you need this.
[root@monitor ~]#
```

```
systemctl enable docker (aktivere docker)
```

```
[root@monitor ~]# systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[ 1063.907867] systemd-rc-local-generator[2601]: /etc/rc.d/rc.local is not marked executable, skipping.
```

For at verificere status på docker, bruges følgende commando:

```
systemctl status docker
```

```
[root@monitor ~]# systemctl status docker
● docker.service - Docker Application Container Engine
  Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: disabled)
  Active: active (running) since Wed 2025-04-23 13:53:22 CEST; 3min 4s ago
    TriggeredBy: ● docker.socket
    Docs: https://docs.docker.com
   Main PID: 2363 (dockerd)
     Tasks: 9
    Memory: 24.5M
      CPU: 282ms
     CGroup: /system.slice/docker.service
             └─2363 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

For at sikre at en lokal bruger kan administrere og køre docker kommandoer med **sudo**, skal brugeren tilføjes til docker gruppen via følgende kommando:

```
sudo usermod -aG docker $USER
```

```
newgrp docker
```

OBS! Du skal være logget på med din egen bruger, for at køre denne kommando, da variablen \$USER refererer til den bruger som er logget på. (Jeg er logget på som root!).

Du kan eventuelt lige skifte bruger og køre kommandoen:

```
[root@monitor ~]# su rudj
[rudj@monitor root]$ cd
[rudj@monitor ~]$ sudo usermod -aG docker $USER
[sudo] password for rudj:
[rudj@monitor ~]$ newgrp docker
[rudj@monitor ~]$
```

Hvis du vil se hvilken docker version der køres, så brug kommandoen:

```
docker --version
```

```
[root@monitor ~]# docker --version
Docker version 28.1.1, build 4eba377
```

Nu kan vi teste docker installationen ved at køre en hello-world container via følgende kommando:

docker run hello-world

```
[root@monitor ~]# docker run hello-world
Unable to find image 'hello-world:latest' locally ←
latest: Pulling from library/hello-world ←
e6590344b1a5: Pull complete ←
Digest: sha256:c41088499908a59aae84b0a49c70e86f4731e588a737f1637e73c8c09d995
Status: Downloaded newer image for hello-world:latest
[ 1804.442210] docker0: port 1(vethbd068b9) entered blocking state
[ 1804.442490] docker0: port 1(vethbd068b9) entered disabled state
[ 1804.443540] device vethbd068b9 entered promiscuous mode
[ 1804.456188] eth0: renamed from veth027706e
[ 1804.461857] IPv6: ADDRCONF(NETDEV_CHANGE): vethbd068b9: link becomes ready
[ 1804.462183] docker0: port 1(vethbd068b9) entered blocking state
[ 1804.462389] docker0: port 1(vethbd068b9) entered forwarding state
[ 1804.463398] IPv6: ADDRCONF(NETDEV_CHANGE): docker0: link becomes ready

Hello from Docker! ←
This message shows that your installation appears to be working correctly.
```

Nu kan vi installere Docker Compose som tillader at starte flere containere på samme tid. Det gøres ved at bruge følgende kommando:

dnf install docker-compose-plugin -y

```
[root@monitor ~]# dnf install docker-compose-plugin -y
Last metadata expiration check: 0:23:14 ago on Wed Apr 23 13:46:34 2025
Package docker-compose-plugin-2.35.1-1.el9.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

Sidste step i Docker installationen er at verificere docker compose versionen via følgende kommando:

docker compose version

```
[root@monitor ~]# docker compose version
Docker Compose version v2.35.1
```

4. Installation af Portainer

Vi starter med at lave drevet/disken hvor Portainer Serveren har sin database. Det gør vi med følgende kommando:

```
docker volume create portainer_data
```

```
[root@monitor ~]# docker volume create portainer_data  
portainer_data ←
```

Nu kan vi downloade og installere Portainer Server containeren via kommandoen:

```
docker run -d -p 8000:8000 -p 9443:9443 --name portainer --restart=always -v  
/var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data  
portainer/portainer-ce:lts
```

```
[root@monitor ~]# docker run -d -p 8000:8000 -p 9443:9443 --name portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data  
portainer/portainer-ce:lts  
Unable to find image 'portainer/portainer-ce:lts' locally  
lts: Pulling from portainer/portainer-ce  
6970b305841e: Pull complete  
c05d4fb47930: Pull complete  
04de893ad5ed: Pull complete  
a9ff7abff372: Pull complete  
e89df2681140: Pull complete  
df254cde32a0: Pull complete  
6b34a21856c4: Pull complete  
faf3a09a336d: Pull complete  
26693e01e9e2: Pull complete  
4f4fb700ef54: Pull complete  
Digest: sha256:449202d765d28ec443c1657fc1121aff92b8afce6b58bcea36e1f0e81e8297c  
Status: Downloaded newer image for portainer/portainer-ce:lts  
728bd2dc147f4d515007c95932771816faf3e07b99386daeebd5de44757?f33  
[ 2542.964236] docker0: port 1(vethad42dfb) entered blocking state  
[ 2542.964259] docker0: port 1(vethad42dfb) entered disabled state  
[ 2542.964409] device vethad42dfb entered promiscuous mode  
[ 2542.968701] eth0: renamed from vethe7b3958  
[ 2542.975002] IPv6: ADDRCONF(NETDEV_CHANGE): vethad42dfb: link becomes ready  
[ 2542.975141] docker0: port 1(vethad42dfb) entered blocking state  
[ 2542.975152] docker0: port 1(vethad42dfb) entered forwarding state
```

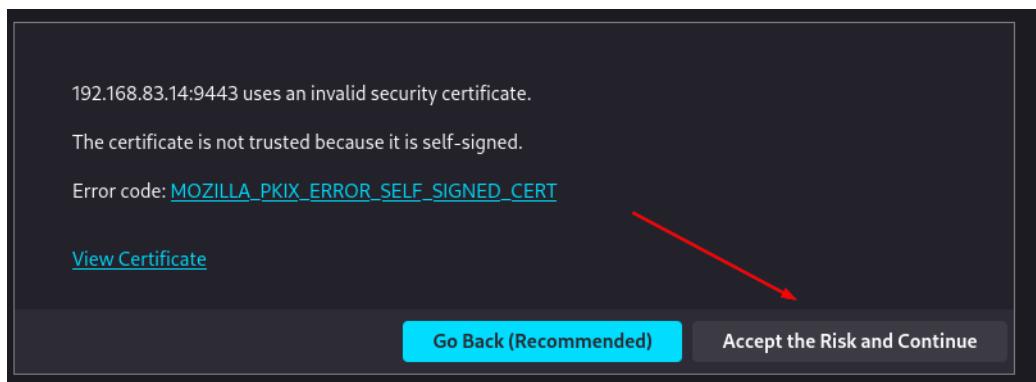
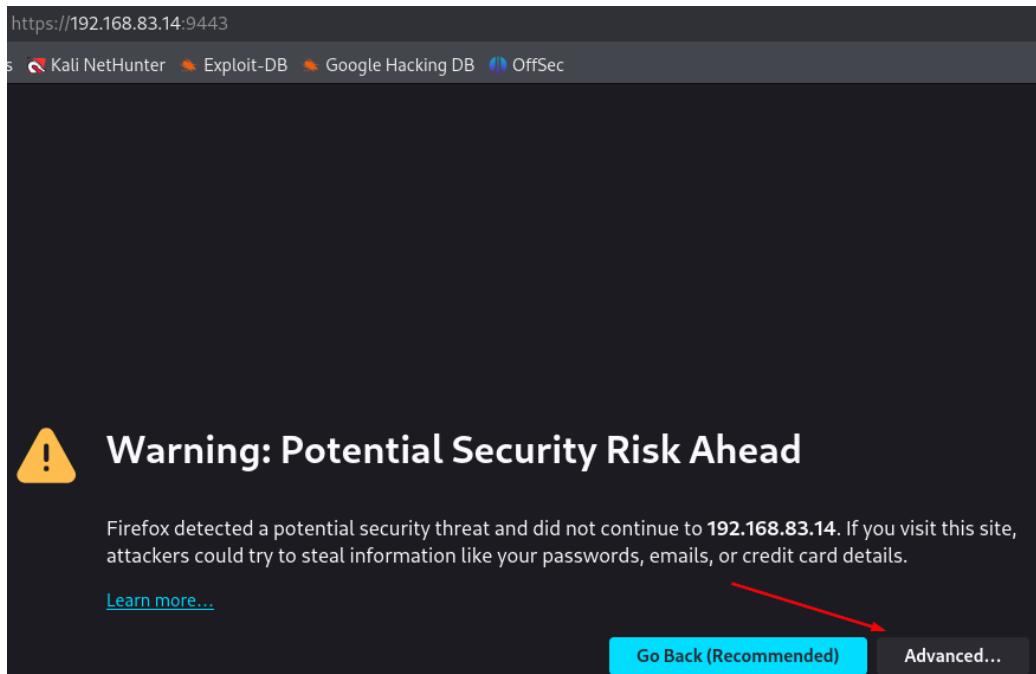
For at tjekke om Portainer Serveren kører, bruges kommandoen:

```
docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
	NAMES				
728bd2dc147f	portainer/portainer-ce:lts	"portainer"	About a minute ago	Up About a minute	0.0.0.0:8000->8000/tcp, [::]:8000->8000/tcp, 0.0.0.0:9443->9443/tcp, [::]:9443->9443/tcp, 9000/tcp portainer

Nu mangler vi bare at logge på via en webbrowser. Det gøres fra din Kali Linux klient maskine. Log på via: <https://localhost:9443> (localhost er din docker servers IP).

Når du ser billedet “Warning: Potential Security Risk Ahead” tryk da på “Advanced” og klik på “Accept the Risk and Continue”.



Nu skal vi lave en bruger til at administrere vores portainer:

▼ New Portainer installation

Please create the initial administrator user.

Username:

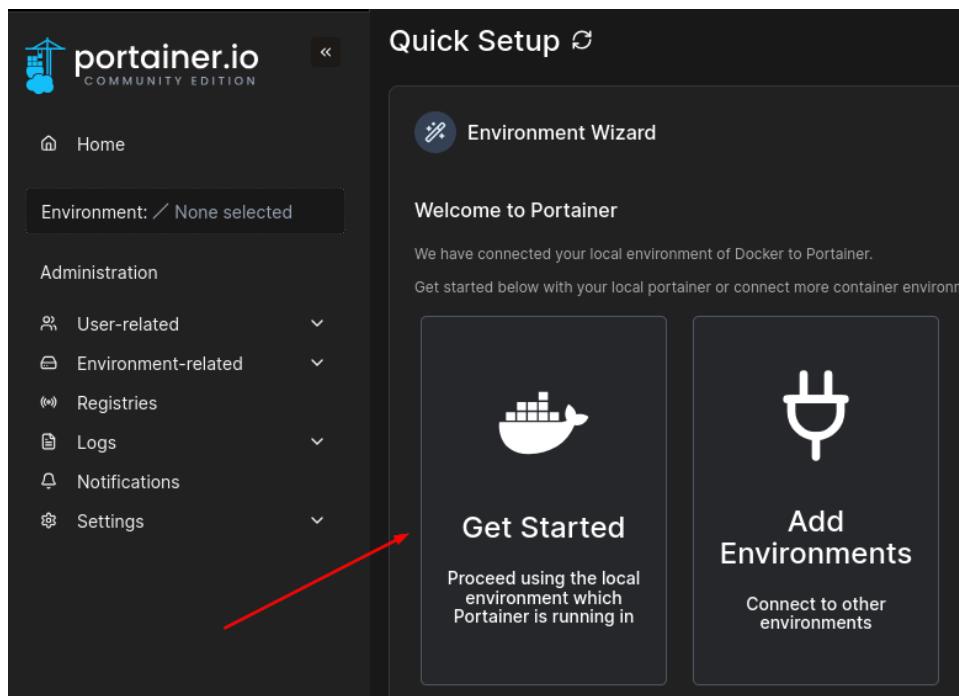
Password:

Confirm password: ✓

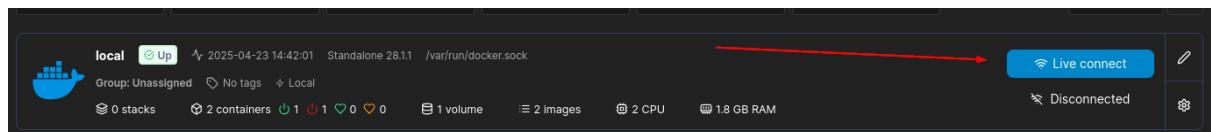
⚠ The password must be at least 12 characters long. ✓

Create User

Nu er det vigtigt at du vælger ”Get Started”



Klik herefter på ”Live Connect”:



5. Installation af Uptime Kuma Container

Nu er vi klar til at installere vores monitorerings værktøj i en container.

Det første vi gør, er at lave mappen til containeren på docker serveren via kommando'en:

mkdir uptimekuma

```
[root@monitor ~]# mkdir uptimekuma ←  
[root@monitor ~]# cd uptimekuma  
[root@monitor uptimekuma]# _
```

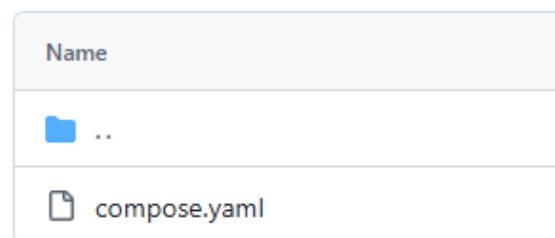
Herefter skiftes til folderen uptimekuma ved brug af kommando'en:

cd uptimekuma

```
[root@monitor ~]# mkdir uptimekuma  
[root@monitor ~]# cd uptimekuma ←  
[root@monitor uptimekuma]# _ ←
```

Nu kan vi hente uptime kuma compose.yaml filen fra følgende link:

<https://github.com/ChristianLempa/boilerplates/tree/main/docker-compose/uptimekuma>

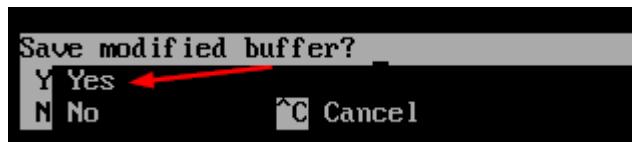


Klik på "compose.yaml" filen og kopier koden, enten via MobaXterm (her kan du copy/paste direkte) eller også lav en compose.yaml fil og skriv koden af.

Der er en lille tilføjelse vi skal bruge for at vores monitorering kan fungere selv når vores DNS går ned. Vi tilføjer en backup DNS i koden, så monitoreringsalarmerne stadigvæk kan sendes til Discord når DNS er nede.

```
volumes:  
  uptimekuma-data:  
    driver: local  
  
services:  
  uptimekuma:  
    image: docker.io/louislam/uptime-kuma:1.23.16  
    container_name: uptimekuma  
    ports:  
      - 3001:3001  
    volumes:  
      - uptimekuma-data:/app/data  
    dns:  
      - 192.168.83.12 ←  
      - 8.8.8.8 ←  
    restart: unless-stopped
```

Tryk på Ctrl + X for at gemme filen. Tryk Y og Enter.



Når den spørger om "File Name to Write: compose.yaml, så tryk Enter.

Nu er vi klar til at starte vores uptimekuma via kommandoen:

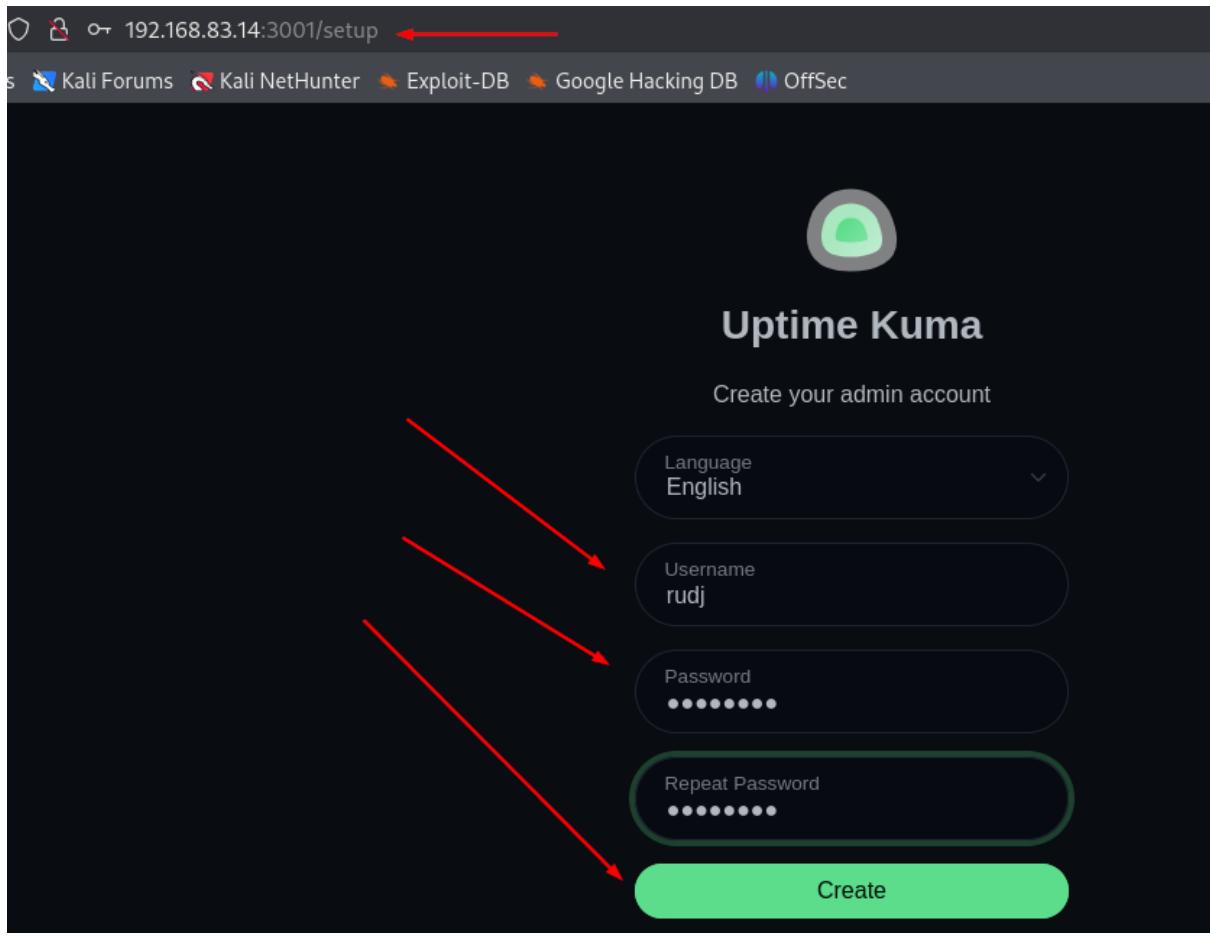
docker compose up -d

```
froot@monitor uptimekuma$ docker compose up -d
[+] Running 13/13
  ▲ uptimekuma Pulled
    ▲ b338562f40a7 Pull complete
    ▲ 874bf4d93728 Pull complete
    ▲ b16337721583 Pull complete
    ▲ 7d955db85b95 Pull complete
    ▲ 2c786596bd17 Pull complete
    ▲ 88a5c59ed14f Pull complete
    ▲ 5a1d0a896c33 Pull complete
    ▲ e68c2f25b946 Pull complete
    ▲ 2e6c90f010d6 Pull complete
    ▲ ff15b10fabb8 Pull complete
    ▲ 4f4fb700ef54 Pull complete
    ▲ d2a900cc8adb Pull complete
[+] Running 2/3
  ▲ Network uptimekuma_default          Created
  ▲ Volume "uptimekuma_uptimekuma-data"   Created
  ▲ Container uptimekuma                Starting
[ 5388.775439] br-f4039dfbe2c9: port 1(veth988821f) entered blocking state
[ 5388.775570] br-f4039dfbe2c9: port 1(veth988821f) entered disabled state
[ 5388.776008] device veth988821f entered promiscuous mode
[ 5388.776084] br-f4039dfbe2c9: port 1(veth988821f) entered blocking state
[ 5388.776695] br-f4039dfbe2c9: port 1(veth988821f) entered forwarding state
[ 5388.777648] br-f4039dfbe2c9: port 1(veth988821f) entered disabled state
[ 5388.882378] eth0: renamed from vetha291389
[+] Running 3/3 IPv6: ADDRCONF(NETDEV_CHANGE): veth988821f: link becomes ready
  ▲ Network uptimekuma_default          Created
  ▲ Volume "uptimekuma_uptimekuma-data"   Created
  ▲ Container uptimekuma                Started
froot@monitor uptimekuma$
```

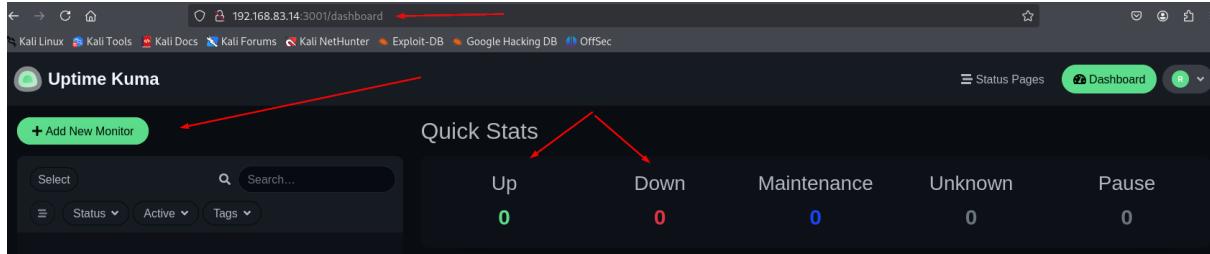
Nu skulle vi gerne kunne se uptimekuma i fanen "Containers" i portainer:

Name	State	Image	Created	IP Address	Published Ports
heuristic-taussig	exited - code 0	hello-world	2025-04-23 14:06:18	-	-
portainer	running	portainer/portainer-ce:its	2025-04-23 14:18:37	172.17.0.2	8000:8000
uptimekuma	healthy	uptimekuma	2025-04-23 15:06:02	172.18.0.2	3001:3001

Nu kan vi logge ind på vores monitorerings server og lave en bruger. Du logger på via en webbrowser fra din klient maskine og bruger monitor serverens IP + port 3001. I mit tilfælde 192.168.83.14:3001.



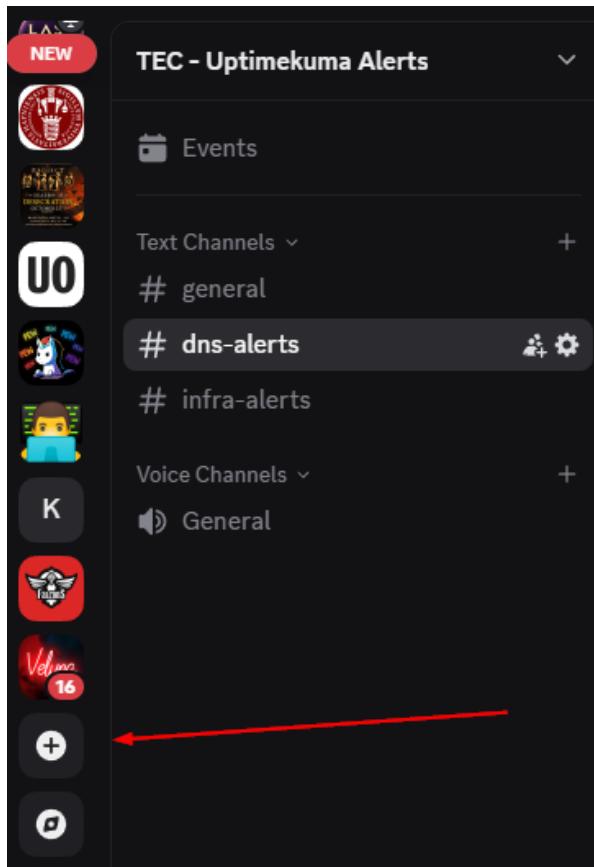
Nu er vi inde på Uptime Kuma Dashboard og så kan vi gå i gang med at sætte alarmer op og få sendt dem direkte til en Discord kanal.



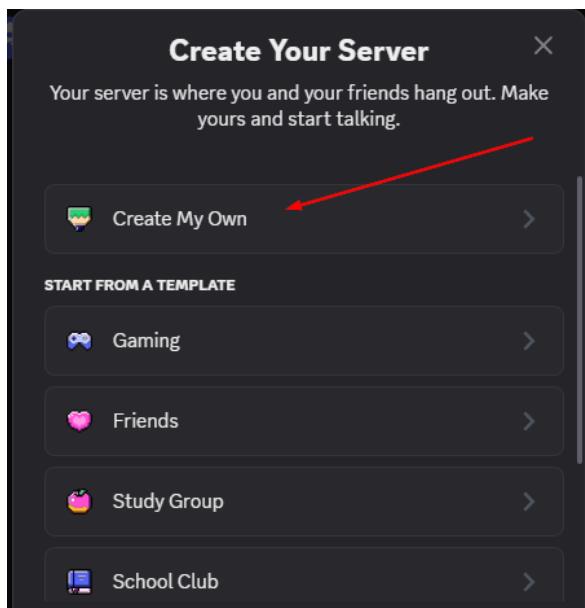
6. Opsætning af Discord Server til Uptimekuma.

Det første skridt på vejen til at sætte alarmer op, er at lave din egen discord server, så du kan få sendt alle dine alarmer dertil.

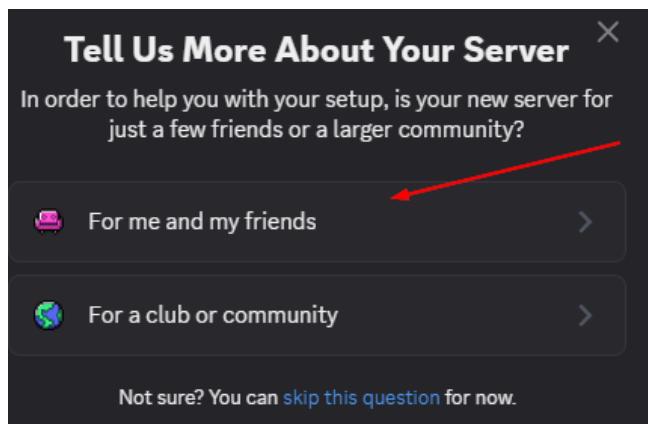
I fanen til venstre, skal du scrollle helt ned og trykke på + for at lave en ny Discord Server.



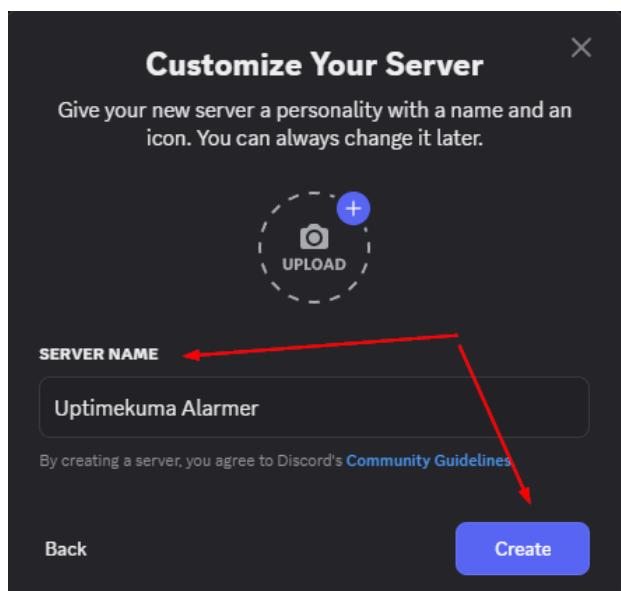
Vælg "Create My Own"



Vælg "For me and my friends"

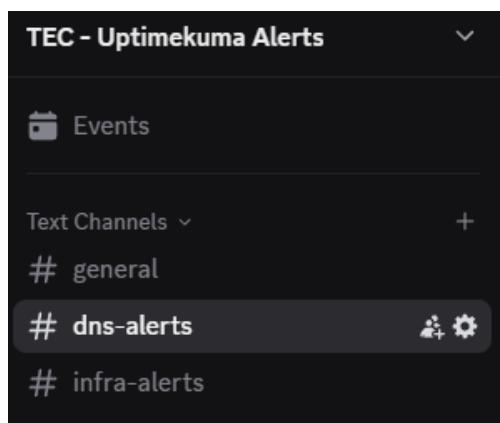


Giv den et navn og tryk på create:



Når serveren er lavet, har du nogen muligheder. Jeg har personligt valgt at have forskellige kanaler til forskellige kategorier af servers.

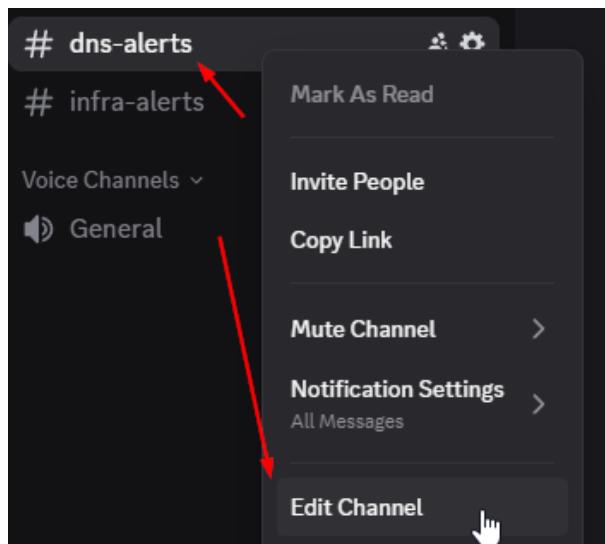
Jeg har indtil videre delt dem op i DNS-alarmer og Infrastruktur-alarmer (De resterende servers som fx Jump og Docker). Senere kan man tilføje DC, DHCP og mange flere.



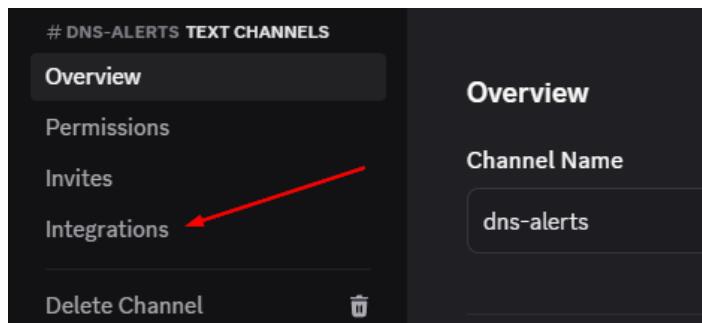
Webhook

Nu er det vigtigt at få lavet en webhook i sine kanaler, så man kan binde alarmerne til de korrekte discord kanaler.

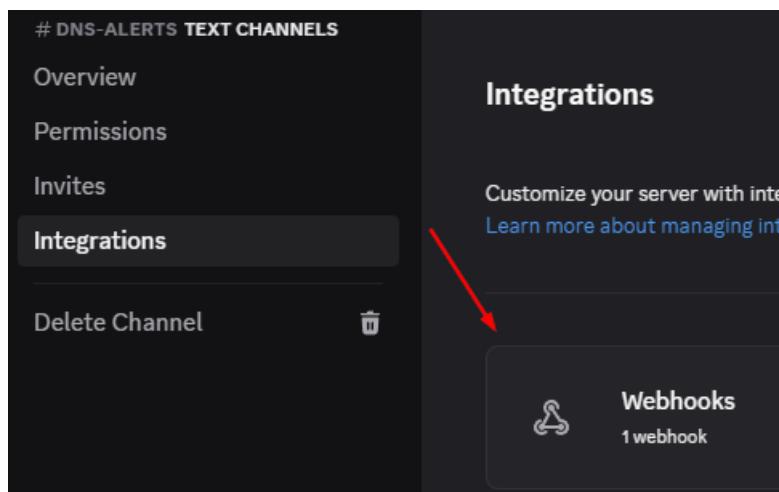
For at lave en webhook i Discord skal du højreklikke på kanalnavnet og vælge ”Edit Channel”



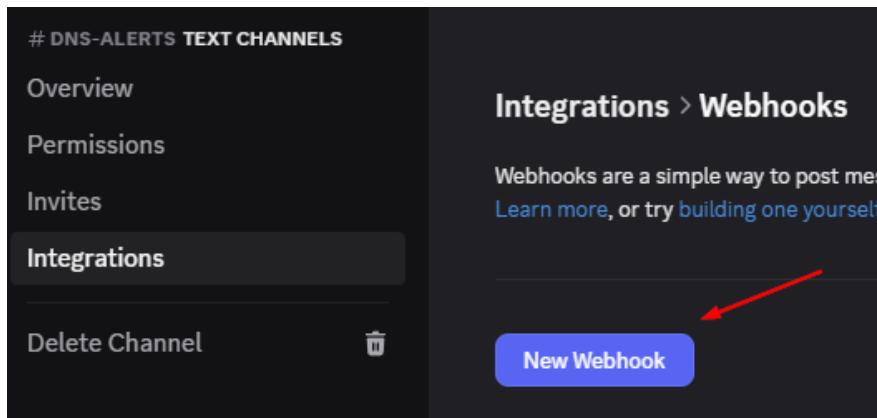
Vælg Integrations:



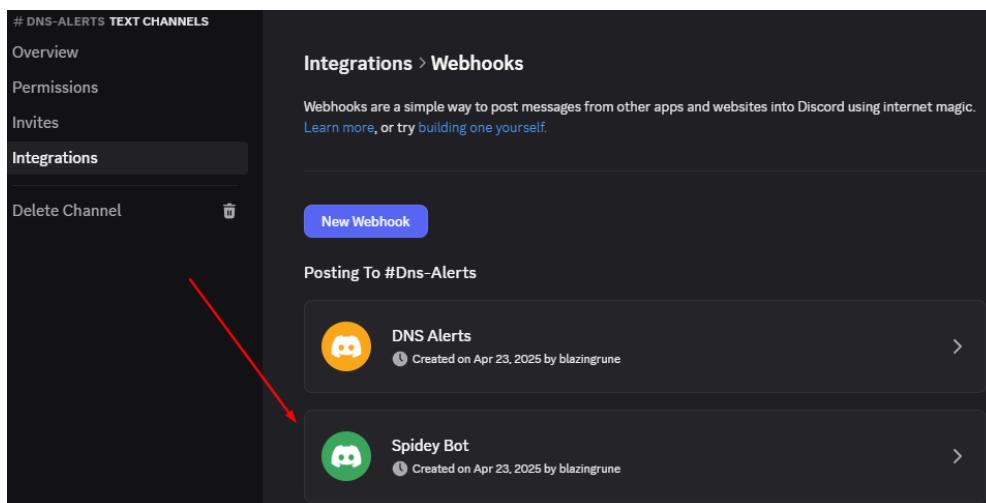
Vælg Webhooks:



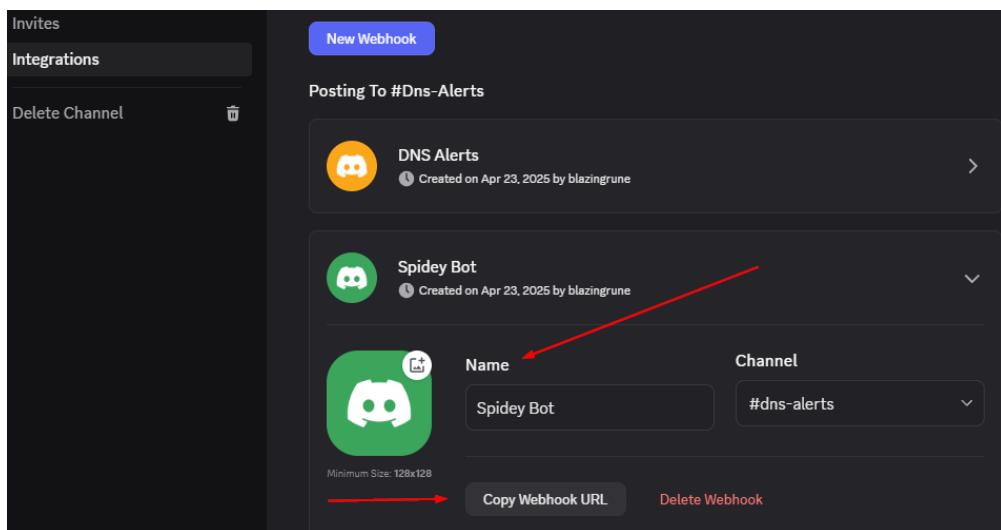
Vælg ”New Webhook”



Åben din nye Webhook:



Giv den et navn, f.eks. DNS Alarm hvis det er DNS du sætter alarmer for. Derefter skal du trykke på ”Copy Webhook URL” og skrive paste det ind i et notepad dokument, så du har det til senere brug når der opsættes alarmer.



Eksempel på Webhook URL fra Spidey Bot (Lad nu være med at bruge dette link, da det er et jeg har genereret på min Discord Server!)

Husk at få lavet en Webhook for hver kanal kategori du opretter, da uptimekuma alarmerne skal bruge en Webhook for at kunne sende til en specifik kanal.

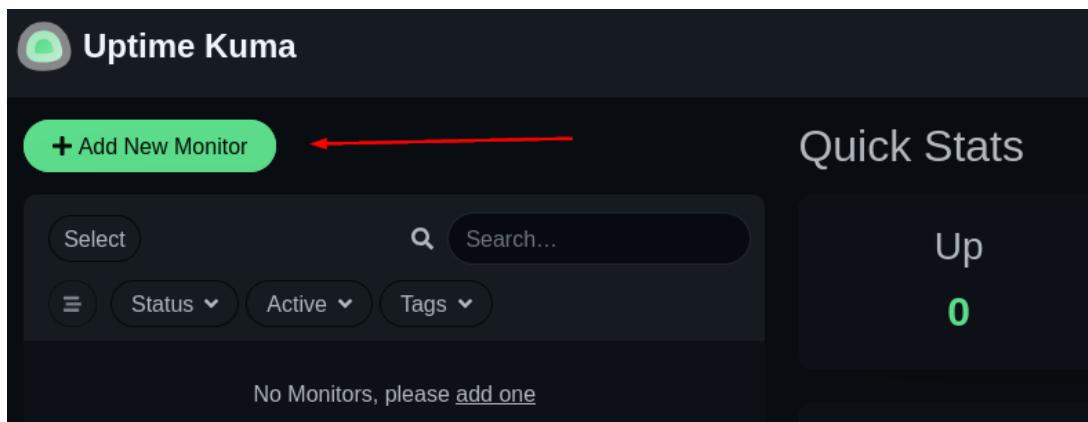
https://discord.com/api/webhooks/1364605792258359367/Kc_pF2mSQLit6EEPD36WkkRlTDs4QWUBYOOvvS7NIRGEfBi3bFd8uUgruTySFevJQzpR

7. Opsætning af alarmer i Uptimekuma

Nu er det blevet tid til at sætte alarmer op i Uptimekuma. Jeg viser kun hvordan jeg har du sætter DNS alarmerne op, gruppere dine alarmer og binder dem til en Webhook som smider alarmerne på din Discord Server når noget går ned og kommer op igen.

Alarm

Vi starter med at oprette en ny alarm via vores Uptimekuma dashboard:



Vi laver først en alarm for vores DNS og kalder den for DNS Resolver.

Her skal vi vælge/indtaste følgende:

Monitor Type: DNS

Friendly Name: DNS Resolver – linuxrudj.internal (Bare for overblikkets skyld!)

Hostname: dns.linuxrudj.internal (FQDN – Fully Qualified Domain Name)

Resolver Server: 192.168.83.12 (IP-adressen på din DNS Server)

A screenshot of the 'Add New Monitor' form. The title 'Add New Monitor' is at the top. Below it is a 'General' section. The first field is 'Monitor Type' with the value 'DNS' selected, indicated by a red arrow. The next field is 'Friendly Name' with the value 'DNS Resolver - linuxrudj.internal', indicated by a red arrow. The third field is 'Hostname' with the value 'dns.linuxrudj.internal', indicated by a red arrow. The fourth field is 'Resolver Server' with the value '192.168.83.12', indicated by a red arrow. At the bottom of the form, a note says 'Cloudflare is the default server. You can change the resolver server anytime.'

Vi er ikke færdige endnu, der var bare ikke plads til resten af skærmklippet 😊 Se næste side.

Port: 53

Ressource Record Type: A

Heartbeat Interval: 20 sekunder

Retries: 2 (ved tredje forsøg uden svar, sendes alarmen)

Monitor Group: Tryk på det grønne plus og navngiv gruppen, f.eks. DNS Services.

Description: Monitors DNS Services

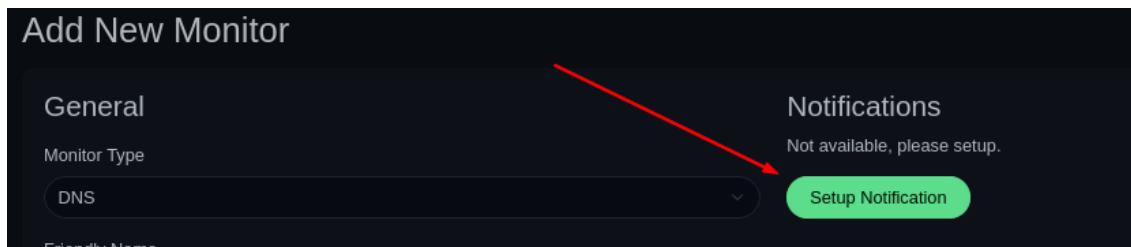
OBS! Klik IKKE på Save endnu. Vi skal lige have tilføjet Discord notifikationen først.

The screenshot shows a configuration interface for a monitoring service. Several fields have been highlighted with red arrows:

- Port:** Set to 53.
- Resource Record Type:** Set to A.
- Heartbeat Interval:** Set to 20.
- Retries:** Set to 2.
- Monitor Group:** Set to DNS Services.
- Description:** Set to Monitors DNS Services.

Notifikation

Nu tilføjer vi notifikationen og det er her vi skal bruge vores Discord Webhook fra tidligere. Klik på "Setup Notification".



Vi starter med at sætte selve DNS-alarmen op, altså reserveren. Her skal vi vælge følgende:

Notification Type: Discord

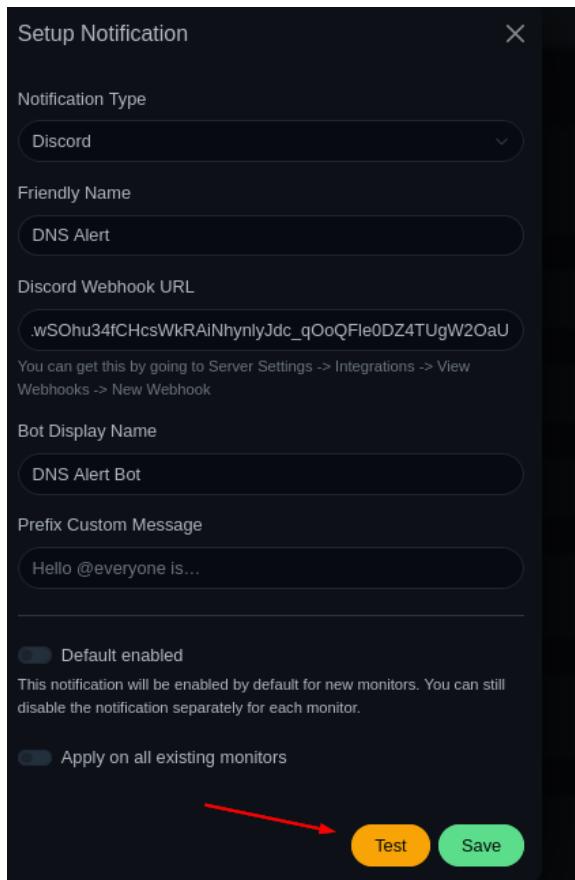
Friendly Name: DNS Alert

Discord Webhook:

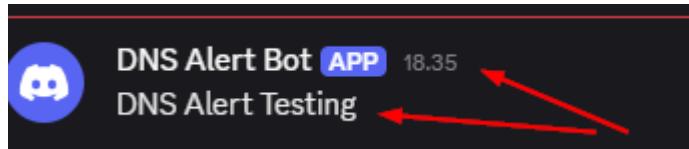
https://discord.com/api/webhooks/1364531711928701059/6dzTRU_sulhAxeNlj6-B1wSOhu34fCHcsWkRAiNhynlyJdc_qOoQFle0DZ4TUgW2OaU (Se forrige afsnit).

Bot Display Name: DNS Alert Bot (Brug et navn som giver mening).

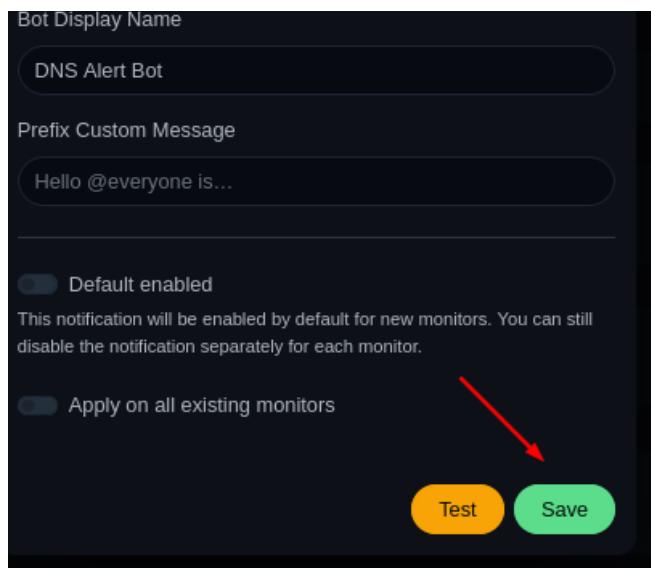
Når ovenstående ting er udfyldt, trykker du på Test knappen for at se om den sender besked direkte til den rigtige Discord kanal.



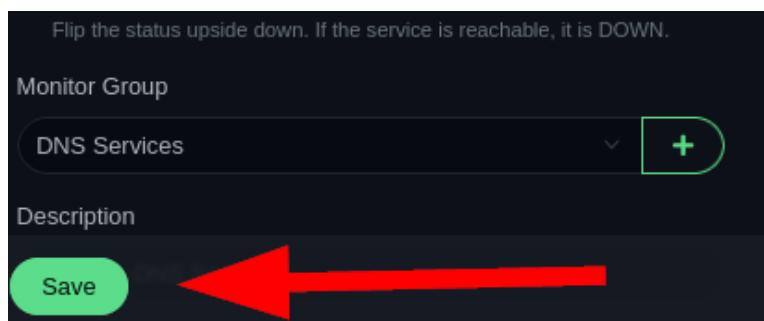
Hvis det virker, kan vi med det samme se i Discord kanalen, at der kommer en test besked fra DNS Alert Bot. På Discord kan vi se både Timestamp for DNS Alert Bot og det den skriver (DNS Alert Testing).



Nu trykker du på "Save" og enabler notifikationen for denne alarm, hvis den ikke automatisk er slået til (den plejer at være slået til når du opretter den, men tjek lige for en god ordens skyld!).

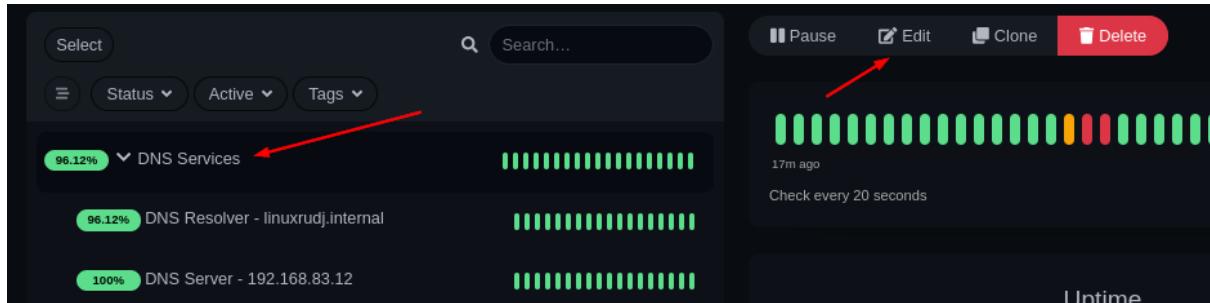


Når alt dette er sat op og testen går igennem til Discord, skal du huske at gemme hele alarmen nede i bunden.

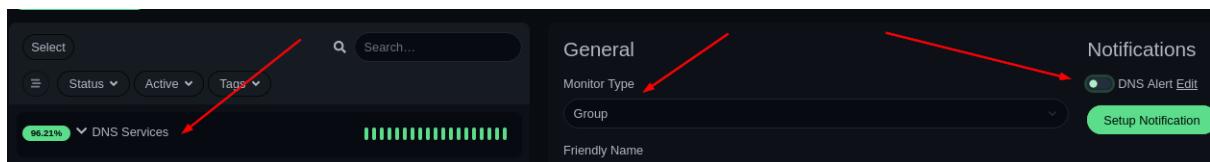


Når notifikationen og Monitor Group er lavet, er det vigtigt lige at tjekke at selve gruppen ikke har fået notifikation på også, da du ellers vil få dobbelt alarmer i Discord.

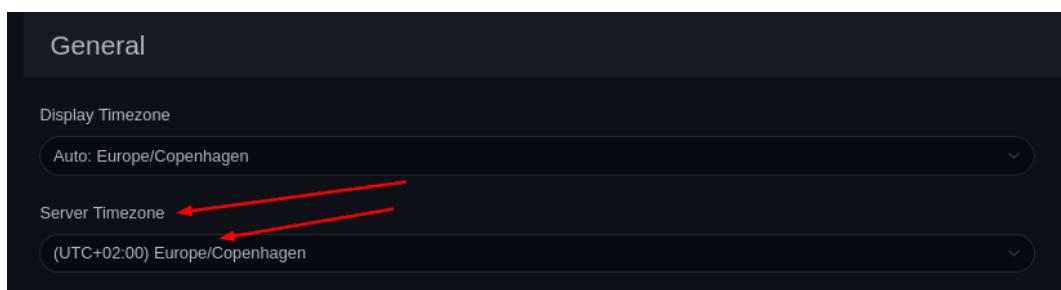
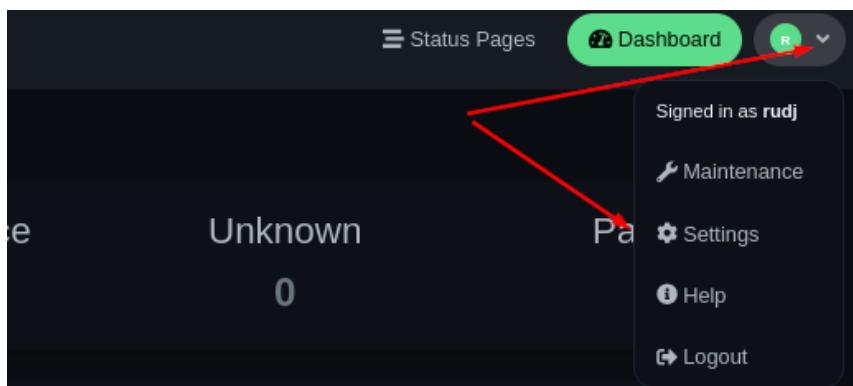
Markér gruppenavnet ”DNS Services” og klik på Edit.



Her er det vigtigt at slå DNS Alert notifikationen fra, så gruppen ikke også spammer Discord med alarmer, hver gang der sker noget!



En sidste god ting at vide, hvis man ikke skal have sine alarmer vist med amerikanske timestamps, så skal man lige ind i indstillinger og indstille ”Server Timezone” korrekt.



Heresfter trykker du bare på Save nede i bunden og så vil de alarmer som sendes til Discord, have de rigtige timestamps.

Hvis vores DNS Resolver ikke virker, hvordan kan vi så få at vide om det er selve resolveren eller om hele serveren er gået ned?

For at få et bedre indblik i hvad der kan være galt, kan det være en god idé at sætte en DNS Server alarm op sammen med Resolver alarmen. Dette gør vi ved at lave en ny alarm, af typen "Ping".

På den måde vil vi få at vide hvis selve DNS Serveren ikke kan nås via ping, hvilket er tegn på at serveren er gået ned eller forbindelsen er røget. Dermed får vi et lille tip om, hvor vi skal begynde fejlsøgningen!

Opsætningen af de næste alarmer er helt identiske til det jeg lige har gennemgået så derfor nøjes jeg med at vise nogle få skærmlip af opsætningen og til sidst teste begge alarmer, så man kan se hvordan det endelige resultat bliver.

Når vi stadigvæk bevæger os indenfor kategorien DNS-alarmer, så er det ikke nødvendigt at lave en ny notifikation. Det er kun hvis vi laver en ny kategori som skal smide alarmen til en ny kanal i Discord. I et sådan tilfælde skal vi lave en ny Monitor Group, Discord Kanal og Discord Webhook.

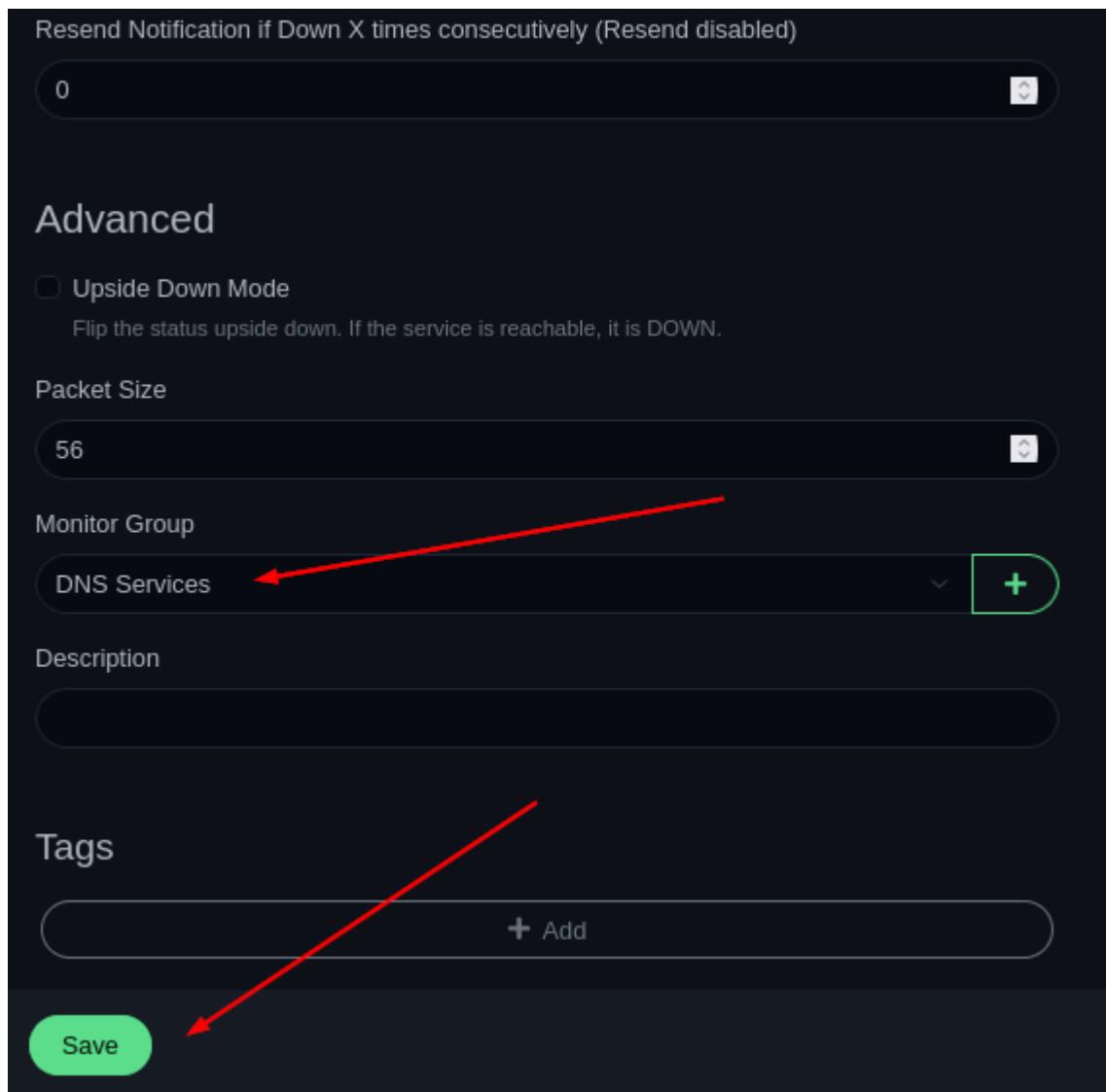
DNS Server Alarm af typen ping

Her ses opsætningen af ping alarmen. Der er valgt "Ping" og den har fået et mere sigende navn og et andet hostname. Det kan ses på billedet at notifikationen allerede er slået til, hvilket er fint da vi er ved at lave endnu en DNS-alarm.

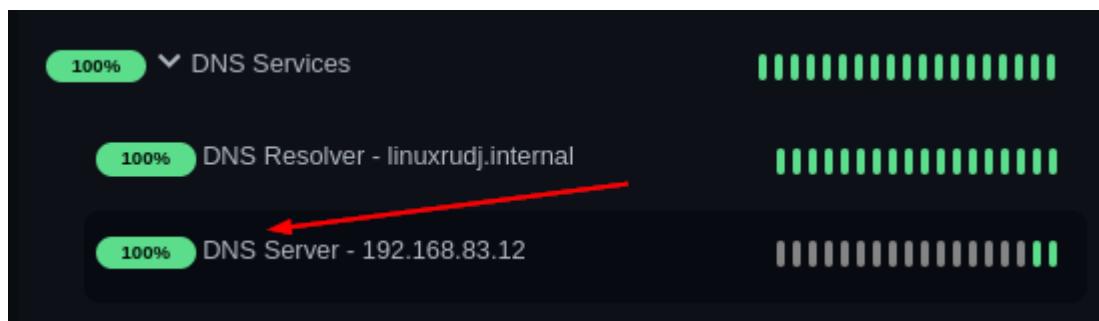
The screenshot shows the 'Add New Monitor' configuration screen. The 'Monitor Type' is set to 'Ping'. The 'Friendly Name' is 'DNS Server - 192.168.83.12'. The 'Hostname' is '192.168.83.12'. The 'Heartbeat Interval (Check every 20 seconds)' is '20'. The 'Retries' is '2'. The 'Heartbeat Retry Interval (Retry every 20 seconds)' is '20'. The 'Resend Notification if Down X times consecutively (Resend disabled)' is '0'. The 'Notifications' section has a 'DNS Alert Edit' toggle switch which is turned on, and a 'Setup Notification' button.

Her kan det ses at samme Monitor Group (DNS Services) er valgt som før.

Husk at gemme den nye alarm!



Her kan det ses at vi nu har tilføjet endnu en alarm til gruppen DNS Services:

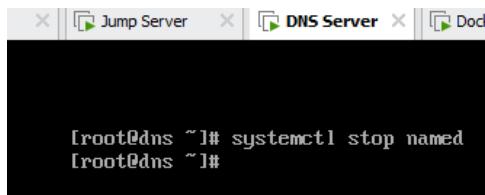


Nu er begge DNS-alarmer sat op og det er tid til at teste og se om de virker.

8. Test af DNS-alarmerne

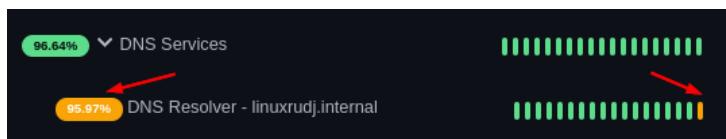
For at teste DNS Resolver alarmen, er det nok at gå ind på DNS Serveren og slukke for "named" ved kommandoen:

systemctl stop named

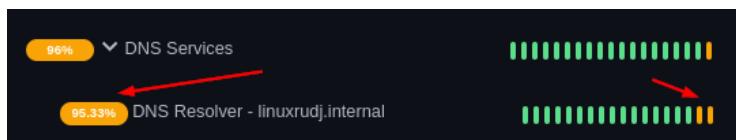


```
[root@dns ~]# systemctl stop named
[root@dns ~]#
```

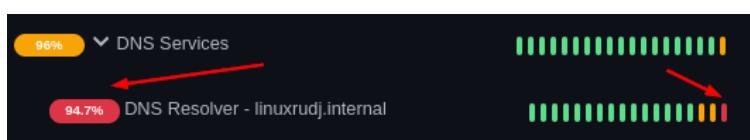
Det første vi vil se når vi kigger på uptimekuma overvågningen, er første heartbeat som vises med 1 orange streg.



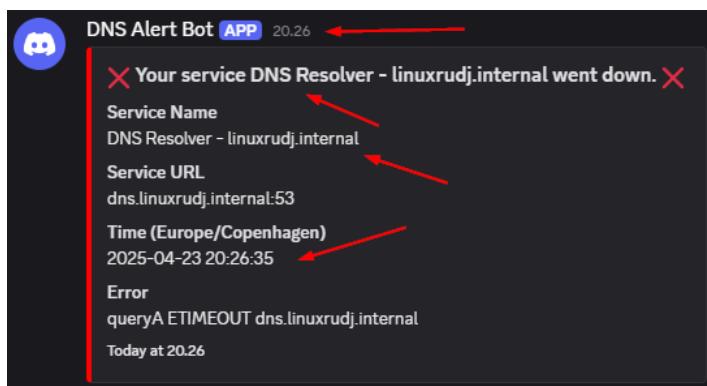
Det næste vi vil se, er andet heartbeat som vises med 2 orange streger.



Det sidste vi vil se, lige inden alarmen sendes til Discord er tredje heartbeat, som er det vi har sat til at indikere når servicen er nede. Det vises med 2 orange og 1 rød streg.



Nu skulle der så gerne komme en alarm i Discord kanalen.



Hermed kan vi konkludere at alarmen virker og kan sende til discord. Årsagen til den stadig sender, er at vi i compose.yaml filen definerede primær og sekundær DNS, så uptimekuma altid kan sende, selvom den lokale DNS går ned.

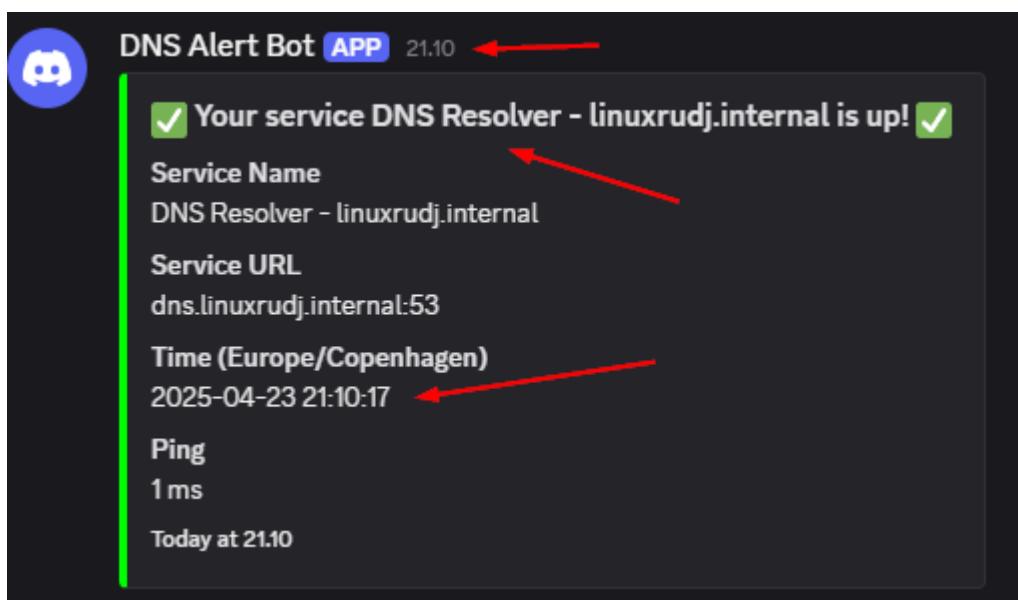
Nu kan vi teste den anden vej og se hvad der sker når vi tænder for "named" igen.

Billedet viser to ting. Det første vi kan se, er at DNS Resolver er oppe igen og har været nede i noget tid.

Det næste vi kan se, er at selve DNS Serveren IKKE har været nede.



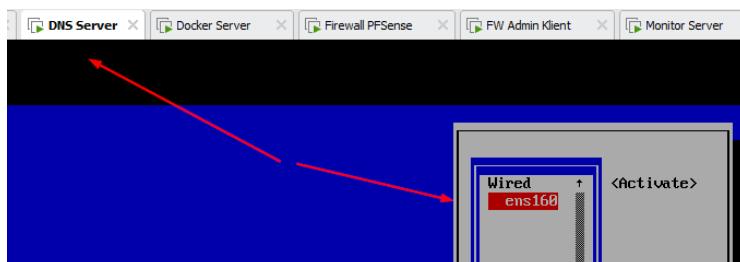
Name	Status	DateTime	Message
DNS Services	Up	2025-04-23 21:10:37	All children up and running
DNS Resolver - linuxrudj.internal	Up	2025-04-23 21:10:17	Records: 192.168.83.12



Hermed ses det at hele DNS Resolver alarmen fungere perfekt og viser både når servicen går ned og kommer op igen, i real-time.

Nu kan vi teste Ping alarmen også. Når Serveren går ned, skulle begge alarmer gerne gå!

For at simulere et nedbrud, deaktivere jeg netværkskortet via værktøjet nmtui.



Her kan vi se at det første som fejler er DNS Server, da forbindelsen er røget.

Det næste vi kan se, er at DNS Resolver er nået til Heartbeat nummer 2 og snart vil blive rød efter Heartbeat 3. Derefter vil den også give alarm på Discord.



På næste billede er det tydeligt at se, at den første alarm som går på Discord, er DNS Server alarmen. Et halvt minuts tid senere, går DNS Resolver alarmen også.

Fidusen her, er at vi nu ved at serveren eller forbindelsen er røget, men ikke nødvendigvis selve vores DNS.

Derfor bliver det noget nemmere at vide hvor man skal starte sin fejlsøgning.

DNS Alert Bot APP 21.18

X Your service DNS Server - 192.168.83.12 went down. X

Service Name
DNS Server - 192.168.83.12

Service URL
192.168.83.12

Time (Europe/Copenhagen)
2025-04-23 21:18:36

Error

```
PING 192.168.83.12 (192.168.83.12) 56(84) bytes of data.  
From 192.168.83.14 icmp_seq=1 Destination Host Unreachable  
From 192.168.83.14 icmp_seq=2 Destination Host Unreachable
```

--- 192.168.83.12 ping statistics ---
2 packets transmitted, 0 received, +2 errors, 100% packet loss, time 41ms
pipe 2

Today at 21.18

21.19

X Your service DNS Resolver - linuxrudj.internal went down. X

Service Name
DNS Resolver - linuxrudj.internal

Service URL
dns.linuxrudj.internal:53

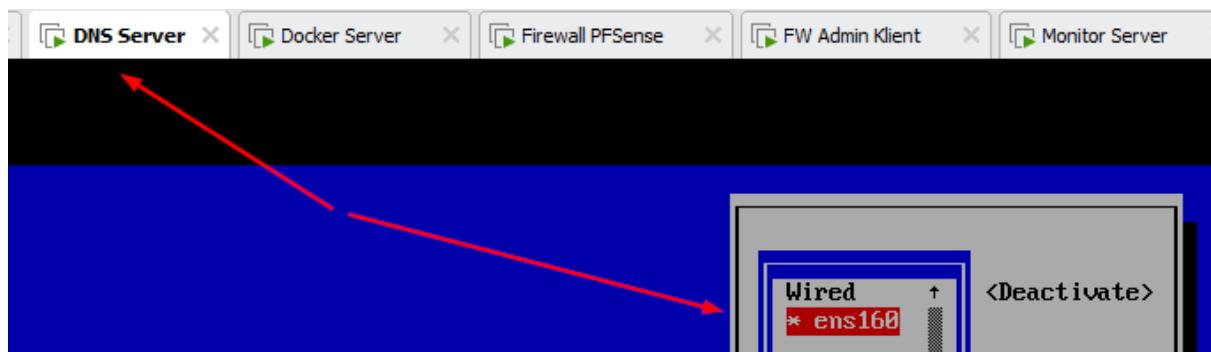
Time (Europe/Copenhagen)
2025-04-23 21:19:04

Error

```
queryA ETIMEDOUT dns.linuxrudj.internal
```

Today at 21.19

Nu mangler vi bare at aktivere netværkskortet igen via nmtui og se hvad der sker.



Nu kan vi se at Discord har fået at vide, at både DNS Server og DNS Resolver kører igen.

A screenshot of a monitoring interface showing two service status cards. The top card is for the "DNS Server" service, which is up at IP 192.168.83.12. The bottom card is for the "DNS Resolver" service, which is up at dns.linuxrudj.internal:53. Both cards provide details like service name, URL, time, and ping results. Red arrows point from the text "Nu kan vi se at Discord har fået at vide, at både DNS Server og DNS Resolver kører igen." to both service status cards.

Service Name	Service URL	Last Check	Ping
DNS Server - 192.168.83.12	192.168.83.12	2025-04-23 21:23:35	2.12 ms
DNS Resolver - linuxrudj.internal	dns.linuxrudj.internal:53	2025-04-23 21:23:41	2 ms

9. TEC H2 Webserver – Total Overvågning

Nedestående billede viser hvordan en eventuel overvågning kan se ud for Web-server forløbet på TEC H2.

