



INSTITUTO SUPERIOR DE ENGENHARIA DE COIMBRA
LICENCIATURA EM ENGENHARIA INFORMÁTICA
RAMO DE REDES E ADMINISTRAÇÃO DE SISTEMAS

Disponibilidade e Desempenho

Proxmox Availability and Failover

Rafael Alves - 2014013189

a21240485@isec.pt

Conteúdo

1	Introdução	1
2	Tecnologias Usadas	2
2.1	Proxmox	2
2.2	Cluster de Alta Disponibilidade	2
2.3	Corosync	2
2.4	Ceph	3
3	Arquitetura para as experiências	4
3.1	Endereçamento	4
3.2	Ambiente Proxmox	5
3.3	Arquitetura dos Servidores	5
4	Trabalho Desenvolvido	6
4.1	Instalação Proxmox	6
4.2	Configuração Ficheiro Hosts	19
4.3	Segunda Placa de Rede	22
4.4	Conectividade	28
4.5	Criação do Cluster	31
4.6	Ceph	39
4.6.1	OSD	44
4.6.2	Pools	50
4.7	Criação Máquina Virtual	52
4.8	Experiência 1 - Migration Ceph	57
4.9	HA	60
4.10	Experiência 2 - High Availability/Automatic Failover	63

Lista de Tabelas

3.1	Tabela endereçamento <i>Internet Protocol</i> (IP)	4
3.2	Tabela endereçamento IP do Cluster	4

Listas de Figuras

3.1 Ambiente Proxmox	5
3.2 Devices	5
4.1 Create a new virtual machine	6
4.2 Tipo de configuração	7
4.3 Install the Operating System later	7
4.4 Tipo Sistema Operativo	8
4.5 Nome e localização da <i>Virtual Machine</i> (VM)	8
4.6 Tamanho Disco VM	9
4.7 Finalização	9
4.8 Inserir <i>International Organization for Standardization</i> (ISO)	10
4.9 Placa de rede	10
4.10 Inicialização da VM	11
4.11 Termos e condições	11
4.12 Escolha do disco rígido	12
4.13 File System	12
4.14 Localização e fuso horário	13
4.15 Password de Administração e Email	13
4.16 Configuração da Rede de Gestão	14
4.17 Resumo da Instalação	14
4.18 Instalação	15
4.19 <i>Proxmox Virtual Environment</i> (PVE) login	15
4.20 Ligação Não Privada	16
4.21 PVE login	16
4.22 Summary DD1	17
4.23 Servidor proxmox DD2	17
4.24 Servidor proxmox DD3	18
4.25 Login Servidor DD1	19
4.26 Configuração do ficheiro Hosts no servidor 1	20
4.27 Configuração do ficheiro Hosts no servidor 2	21
4.28 Configuração do ficheiro Hosts no servidor 3	21
4.29 Definições do servidor proxmox	23
4.30 Add Hardware Wizard	24
4.31 Modo Bridge Placa de Rede	25
4.32 IP Segunda Placa de Rede	26
4.33 Restart Node	26
4.34 Ficheiro configuração interfaces	27
4.35 Ping para DD2 e DD3	28
4.36 Ping para o exterior	28
4.37 Ping para DD1 e DD3	29
4.38 Ping para DD1 e DD3	29
4.39 Ping para segunda interface de rede DD2	30
4.40 Ping para segunda interface de rede DD2	30

4.41 Cluster	31
4.42 Create Cluster	31
4.43 Create Cluster Task Viewer	32
4.44 Cluster Nodes DD1	32
4.45 Copy Information	33
4.46 Join Cluster	33
4.47 Join Cluster	34
4.48 Cluster Nodes DD2	34
4.49 Cluster Nodes DD1	35
4.50 Cluster DD3	35
4.51 Cluster Join	36
4.52 Cluster Nodes DD3	36
4.53 Cluster DD1	37
4.54 Configuração Corosync	37
4.55 Configuração Corosync	38
4.56 Datacenter Summary	38
4.57 Instalação Ceph	39
4.58 Versão do Ceph	39
4.59 Continuação da instalação	40
4.60 Configuração do <i>Ceph</i>	40
4.61 Término da instalação	41
4.62 Monitor DD1	41
4.63 Configuração nos restantes nós	42
4.64 Monitor DD2	42
4.65 Create Monitor DD3 Manualmente	43
4.66 Monitor do Cluster	43
4.67 Configuração do <i>Ceph</i>	44
4.68 Adicionar Hard Disk	44
4.69 Escolha de Tipo de Disco	45
4.70 Create New vitual Disk	45
4.71 Capacidade do disco	46
4.72 Término da criação do novo disco	46
4.73 Create OSD	47
4.74 Escolha do disco	47
4.75 OSD DD1	48
4.76 OSD DD2	48
4.77 OSD Cluster	49
4.78 Health Ceph	49
4.79 Pool Create	50
4.80 Pool Create	50
4.81 Health Ceph	51
4.82 Upload ISO	52
4.83 Task Ok Upload	52
4.84 Create Virtual Machine-General	53
4.85 Create Virtual Machine-OS	53
4.86 Create Virtual Machine-Disks	54
4.87 Create Virtual Machine-CPU	54
4.88 Create Virtual Machine-Memory	55
4.89 Create Virtual Machine-Network	55
4.90 Ifconfig Ubuntu Server	56
4.91 Conectividade com a VM	56
4.92 Ping VM	57

4.93 Migrate VM	57
4.94 Migração em curso	58
4.95 Ping	58
4.96 Migração concluída	59
4.97 Group HA	60
4.98 Add HA	60
4.99 Resources HA	61
4.100Resources HA	61
4.101Resources HA	62
4.102Website	63
4.103Dashboard Uptime Kuma	63
4.104Ping para Container	64
4.105Migração	64
4.106Migração Concluída	65
4.107Website online	65
4.108Overall Cluster	66
4.109Overall Cluster	67
4.110Dashboard Uptime Kuma	67
4.111Dashboard Uptime Kuma	68

Acrónimos

DD *Disponibilidade e Desempenho*

EI *Engenharia Informática*

VM *Virtual Machine*

RADOS *Reliable Autonomic Distributed Object Store*

LAN *Local Area Networks*

IP *Internet Protocol*

ISO *International Organization for Standardization*

SATA *Serial Advanced Technology Attachment*

CD/DVD *Compact Disk/Digital Versatile Disc*

HDD *Hard Disk Drive*

DNS *Domain Name System*

PVE *Proxmox Virtual Environment*

SO *Sistema Operativo*

HA *High Availability*

OSD *Ceph Object Storage Daemons*

SCSI *Small Computer System Interface*

RADOS *Reliable Autonomic Distributed Object Store*

PG *Placement Group*

PC *Personal Computer*

HTML *HyperText Markup Language*

CSS *Cascading Style Sheet*

Capítulo 1

Introdução

Este trabalho foi realizado no âmbito da disciplina de *Disponibilidade e Desempenho* (DD) do 3º ano da licenciatura em *Engenharia Informática* (EI) - Ramo de Redes e Administração de Sistemas do Instituto Superior de Engenharia de Coimbra.

Foi proposto pelo professor Luís Santos o Projeto B onde pretende que se estude uma tecnologia presente na Unidade Curricular de Disponibilidade e Desempenho. Como tal este trabalho tem como objetivo estudar o *Failover* e *Availability* em *Proxmox*.

O principal foco é, criar um *Cluster* em *Proxmox*, criar cenários de falhas e tentar minimizar o *downtime* da(as) VM do *Cluster*.

Capítulo 2

Tecnologias Usadas

2.1 Proxmox

Proxmox VE (PVE) é um sistema de gestão de VM e container (CT) baseado em Linux. Permite criar e gerir VM e container em um único *host* ou em um *cluster* de múltiplos *hosts*. O Proxmox é uma solução de virtualização completa para criar e gerir VM e containeres, incluindo um sistema de gestão de *cluster*, armazenamento partilhado, *backup* e recuperação de desastres e muito mais.

O Proxmox é baseado no Kernel Linux e suporta vários *Sistema Operativo* (SO), incluindo Linux, Windows e muitos outros. Também inclui uma interface gráfica intuitiva que permite toda a gestão do servidor.

2.2 Cluster de Alta Disponibilidade

Um cluster de alta disponibilidade *High Availability* (HA) é um conjunto de máquinas que trabalham em conjunto para fornecer um serviço de alta disponibilidade HA. Isso significa que, se uma das máquinas falhar ou ficar indisponível por qualquer motivo, outra máquina no cluster pode assumir o trabalho sem interrupção do serviço. Dessa forma, um cluster de alta disponibilidade HA garante que um serviço esteja sempre disponível para os utilizadores, mesmo em caso de falhas.

Existem várias maneiras de implementar um cluster de alta disponibilidade HA, mas a maioria das soluções envolve a utilização de dois ou mais nós, cada um com os seus próprios recursos de hardware e software. Cada nó pode ser responsável por executar um serviço ou um conjunto de serviços, e os nós são interligados por uma rede de comunicação para permitir que eles se comuniquem e tomem decisões sobre o estado atual do cluster.

Para gerir um cluster de alta disponibilidade HA, é necessário um software de gestão, como o *Corosync* ou o *Pacemaker*. Este software monitora o estado dos nós e dos serviços no cluster e toma as medidas adequadas em caso de falhas, como transferir os serviços para outro nó ou reiniciá-los num nó que esteja a funcionar corretamente.

2.3 Corosync

Corosync é um software de alta disponibilidade HA que fornece um mecanismo de comunicação confiável para sistemas de alta disponibilidade HA. É projetado para permitir que os serviços de alta disponibilidade HA comuniquem entre si e

tomem decisões sobre o estado atual do sistema e quais ações devem ser tomadas em caso de falhas.

Corosync usa um protocolo de mensagem chamado "*Virtual Synchrony*" para garantir que as mensagens são entregues de forma confiável entre os nós de um cluster de alta disponibilidade HA.

2.4 Ceph

Ceph é um sistema de armazenamento distribuído (*open-source*) que foi desenvolvido para fornecer alta disponibilidade HA, escalabilidade e capacidade de armazenamento. Ele usa um conjunto de máquinas ("nós") para criar uma "*pool*" de armazenamento partilhado que é acessível por meio de uma variedade de protocolos de armazenamento, incluindo o Protocolo de Armazenamento em Blocos (RBD), o Protocolo de Armazenamento em Arquivo (RGW) e o Protocolo de Armazenamento em Objetos (RADOS).

O *Ceph* é projetado para ser escalável e pode ser facilmente expandido adicionando mais nós à sua rede. Também é capaz de tolerar falhas de hardware e oferece alta disponibilidade HA, pois os dados são replicados em vários nós para garantir que eles estejam sempre disponíveis. Além disso, o *Ceph* suporta uma variedade de usos, incluindo armazenamento em *cloud*, backup e recuperação de desastres, armazenamento de arquivos e muito mais.

Capítulo 3

Arquitetura para as experiências

Para a realização deste trabalho prático foi criado um ambiente com várias máquinas virtuais na mesma *Local Area Networks* (LAN), utilizando o virtualizador VMWare.

3.1 Endereçamento

Administração:

Nome	IP
Servidor DD1	192.168.1.129
Servidor DD2	192.168.1.130
Servidor DD3	192.168.1.131
VM	192.168.1.128
CT	192.168.1.140

Tabela 3.1: Tabela endereçamento IP

Rede do Cluster:

Nome	IP
Servidor DD1	10.10.10.129
Servidor DD2	10.10.10.130
Servidor DD3	10.10.10.131

Tabela 3.2: Tabela endereçamento IP do Cluster

3.2 Ambiente Proxmox

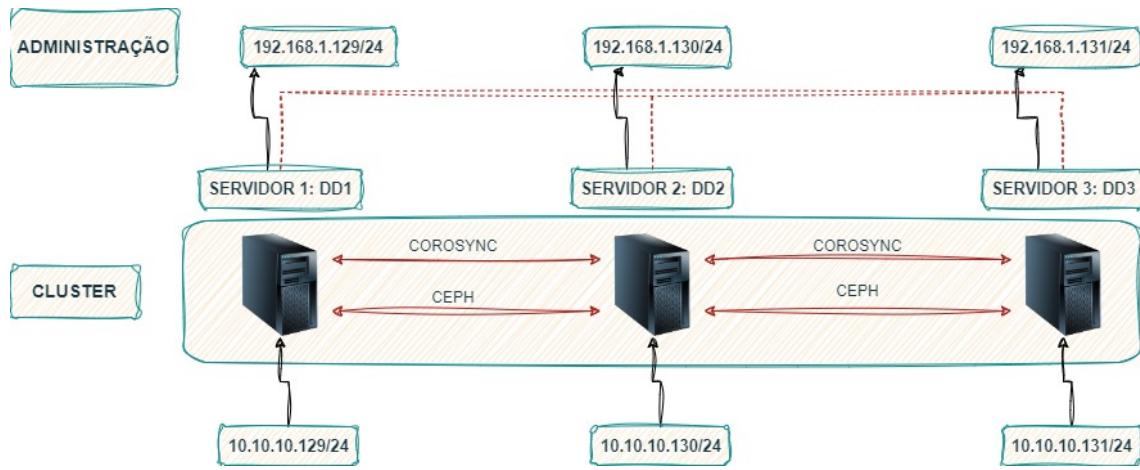


Figura 3.1: Ambiente Proxmox

3.3 Arquitetura dos Servidores

- CPU: 2;
- RAM: 4GB;
- Placa de Rede: 2;
- Disco: 120GB;

▼ Devices

	Memory	4 GB
	Processors	2
	Hard Disk (SCSI)	120 GB
	Hard Disk 2 (SCSI)	80 GB
	CD/DVD (SATA)	Auto detect
	Network Adapter	Bridged (Autom...
	Network Adapter 2	Bridged (Autom...
	USB Controller	Present
	Display	Auto detect

Figura 3.2: Devices

Capítulo 4

Trabalho Desenvolvido

4.1 Instalação Proxmox

Inicialmente foram criados os vários servidores proxmox (DD1, DD2, DD3) e para isso fica registrado passo a passo a instalação.

Passo 1: "Create a new virtual machine".

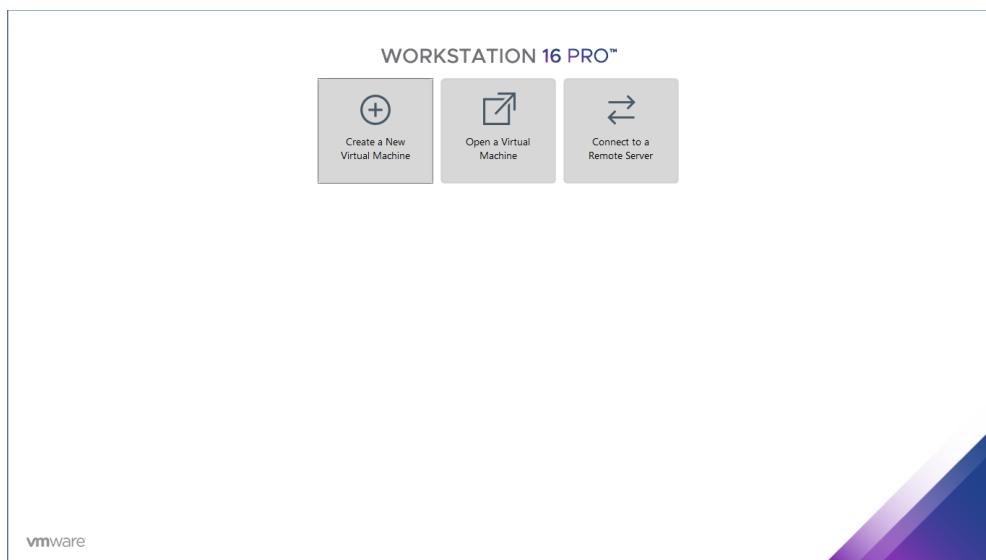


Figura 4.1: Create a new virtual machine

Passo 2: Aqui é escolha pessoal, eu escolhi "Typical".

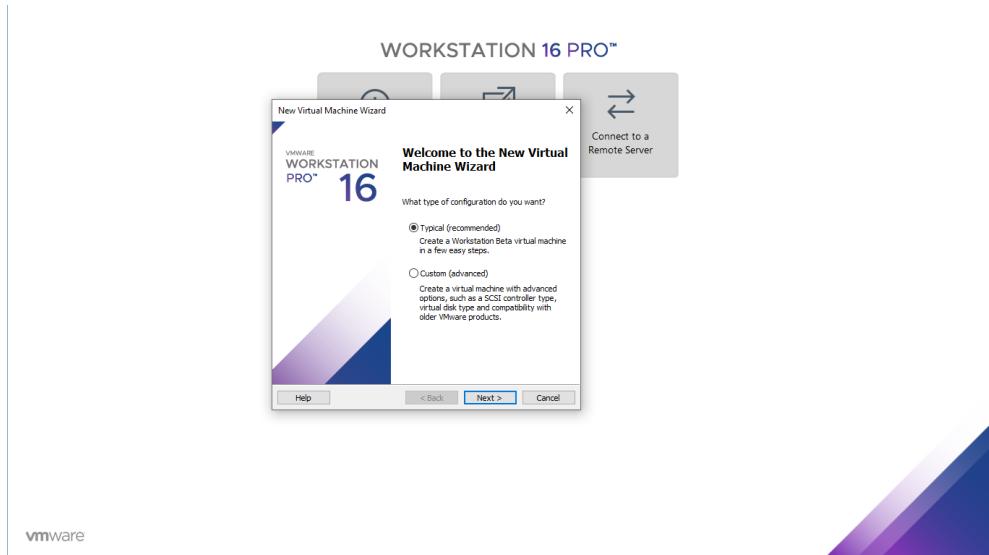


Figura 4.2: Tipo de configuração

Passo 3: Escolhi instalar o Sistema Operativo mais tarde.

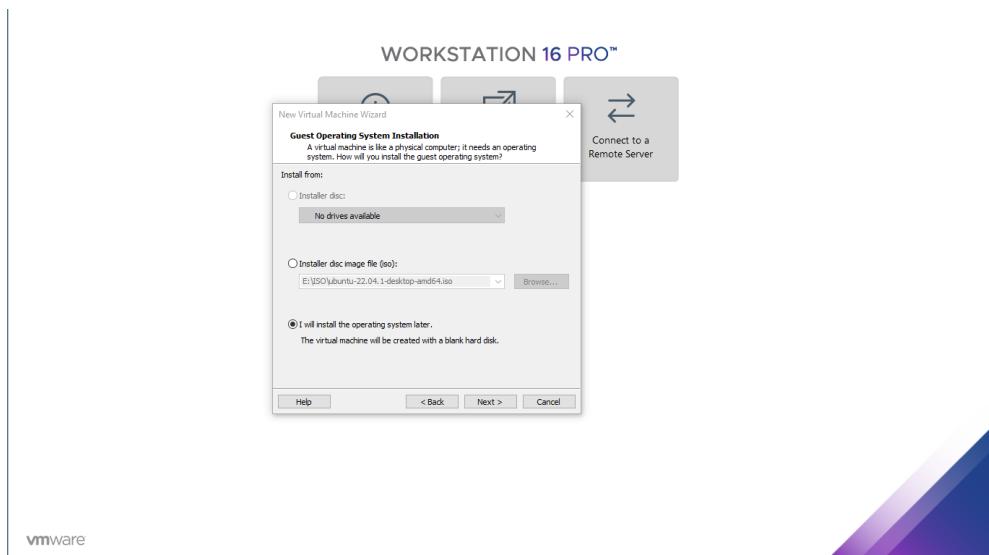


Figura 4.3: Install the Operating System later

Passo 4: Por predefinição deixei ficar as opções dadas uma vez que o proxmox é baseado em Linux.

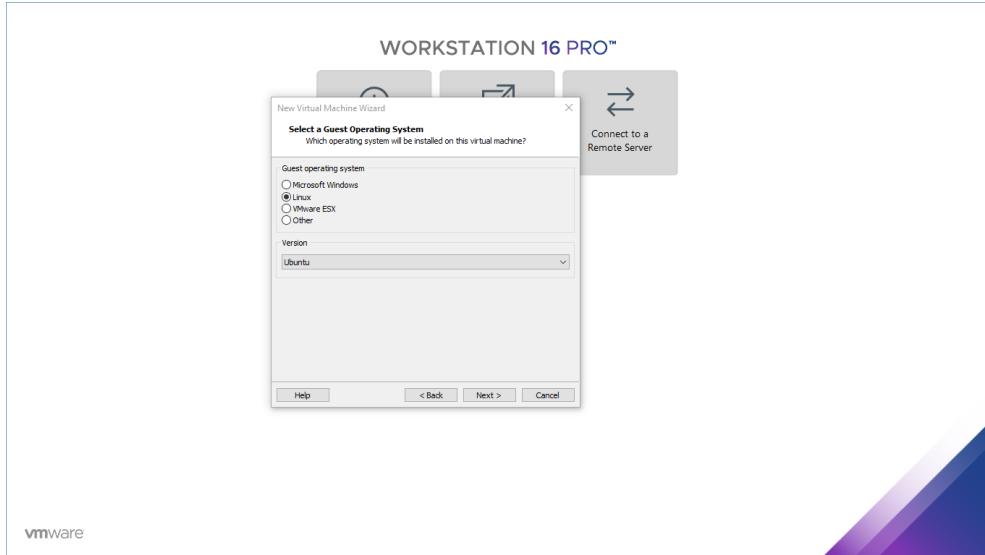


Figura 4.4: Tipo Sistema Operativo

Passo 5: Escolher o nome (Proxmox) para a VM e a localização. Neste caso utilizei um disco externo de 1TB.

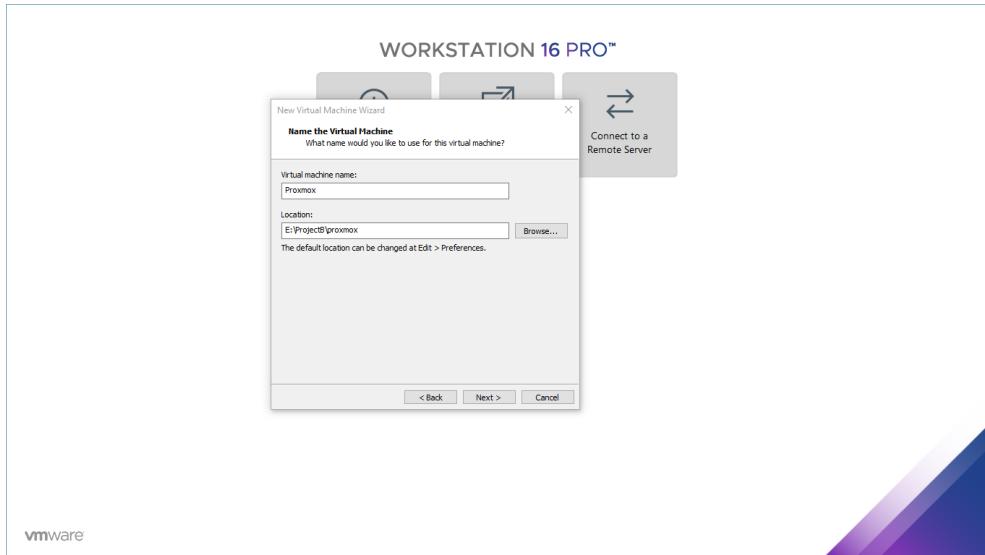


Figura 4.5: Nome e localização da VM

Passo 6: Definir o tamanho do disco para a VM. Neste caso foi 120Gb.

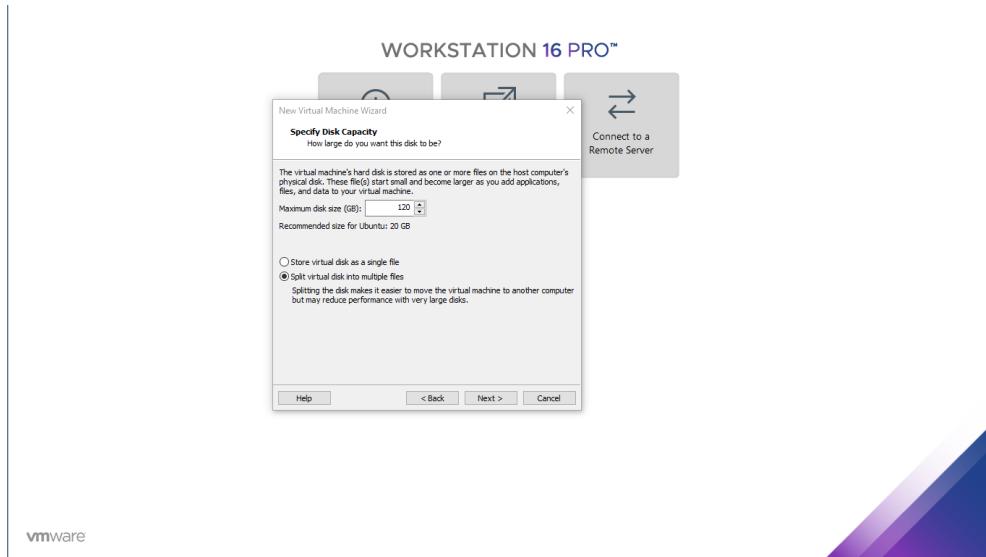


Figura 4.6: Tamanho Disco VM

Passo 7: Resumo da configuração e finalização.

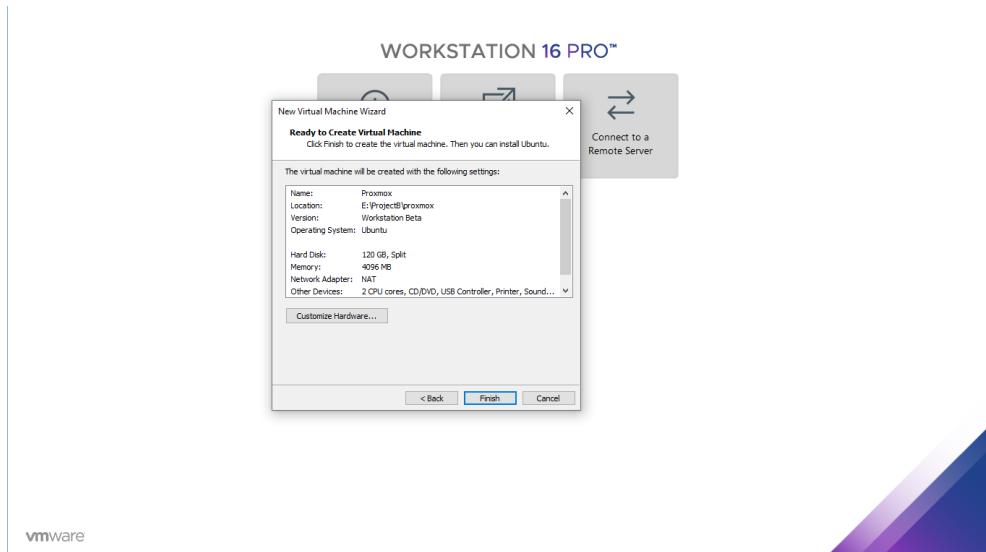


Figura 4.7: Finalização

Passo 8: Nas propriedades da VM, inseri a ISO do proxmox em "Compact Disk/Digital Versatile Disc (CD/DVD) (Serial Advanced Technology Attachment (SATA))".

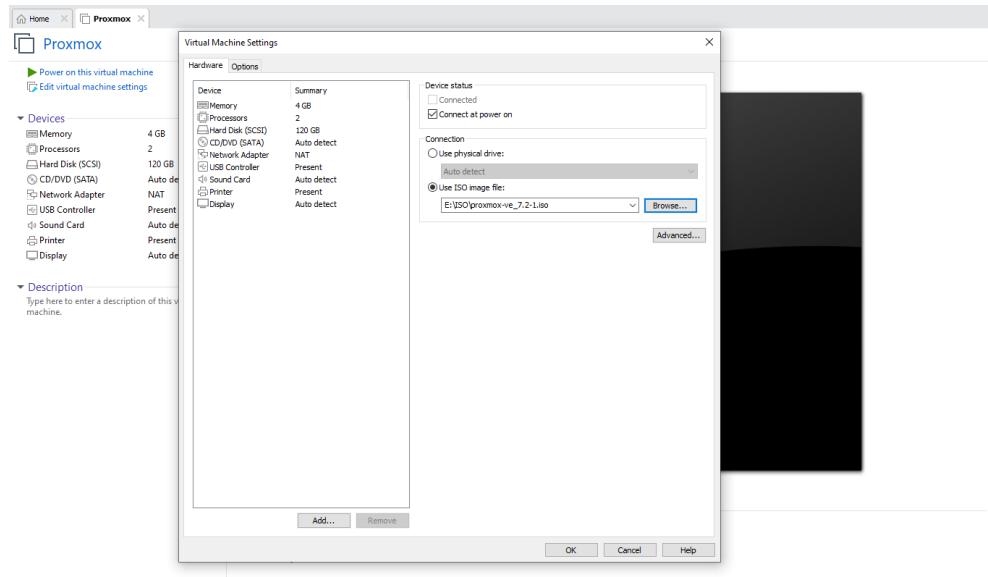


Figura 4.8: Inserir ISO

Passo 9: Ainda nas propriedades da VM, a placa de rede tem de estar em modo bridge.

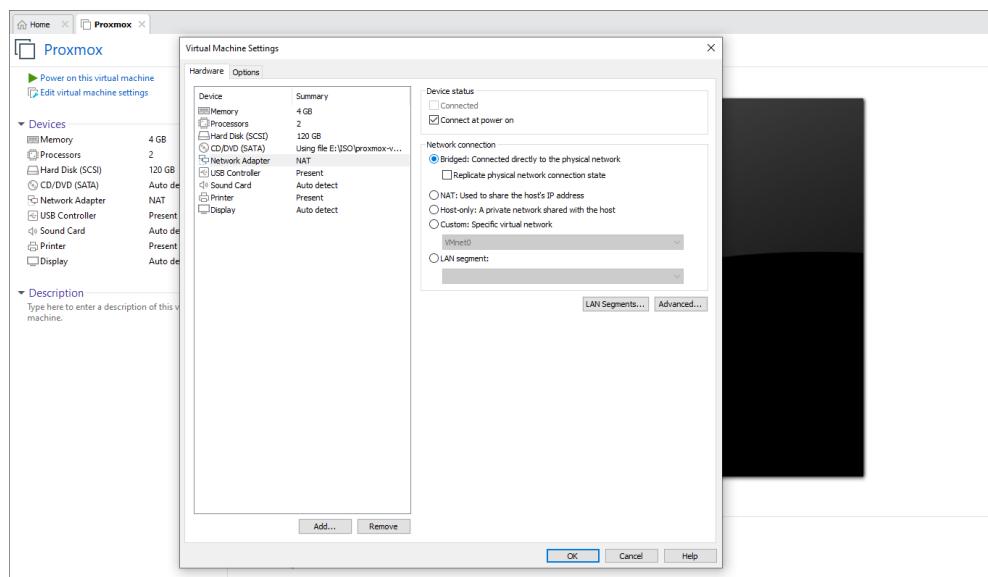


Figura 4.9: Placa de rede

Passo 10: Depois de alterar algumas configurações iniciais, podemos iniciar a VM e iniciar a instalação.

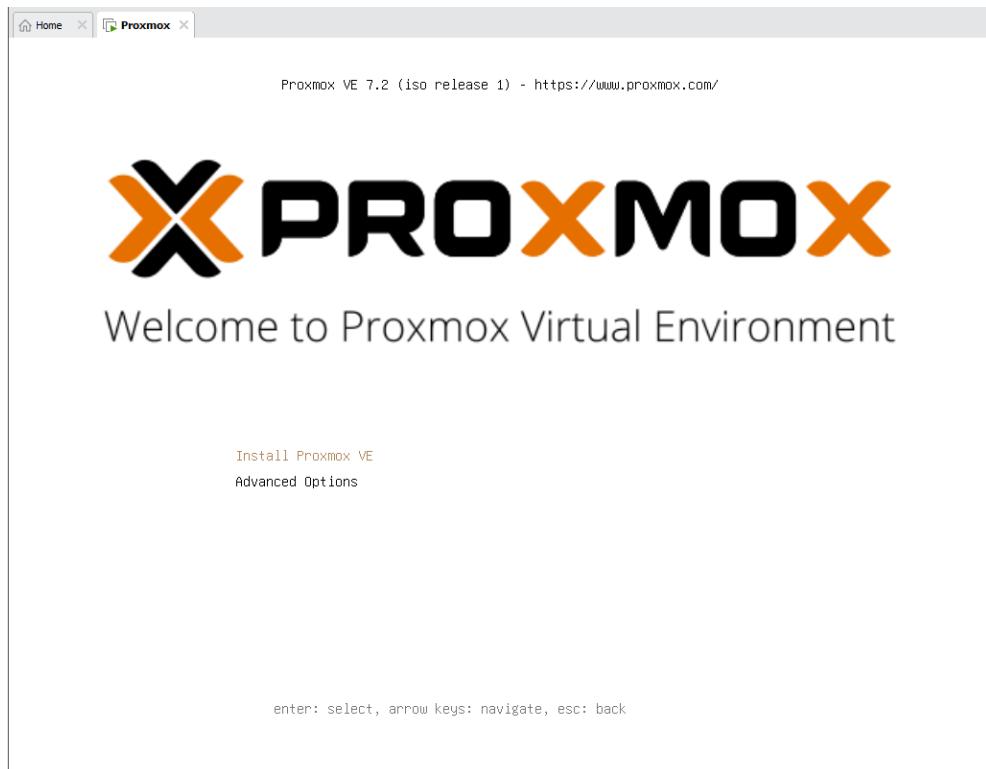


Figura 4.10: Inicialização da VM

Passo 11: Aceitar os termos e condições do proxmox.

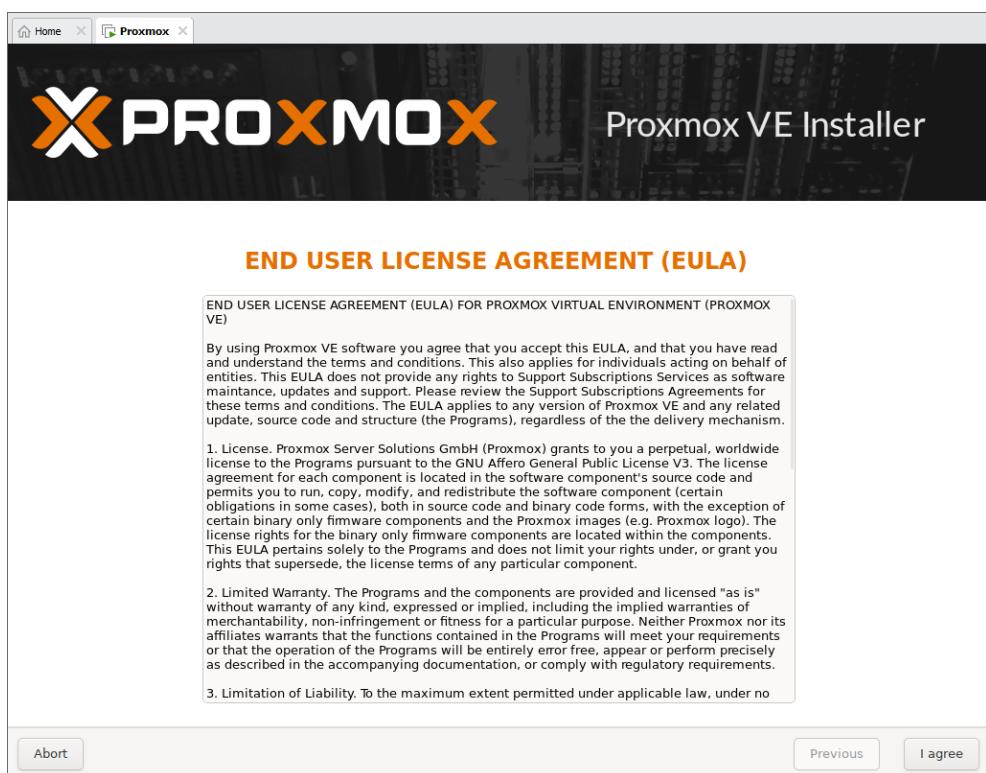


Figura 4.11: Termos e condições

Passo 12: Escolha do disco para a instalação do servidor. No meu caso estou a usar um disco *Hard Disk Drive* (HDD) de 1TB e os 120Gb foi a "partição" que criamos inicialmente na criação da VM.



Figura 4.12: Escolha do disco rígido

O File System usado foi *ext4*.



Figura 4.13: File System

Passo 13: Escolha do país, fuso horário e *layout* do teclado.

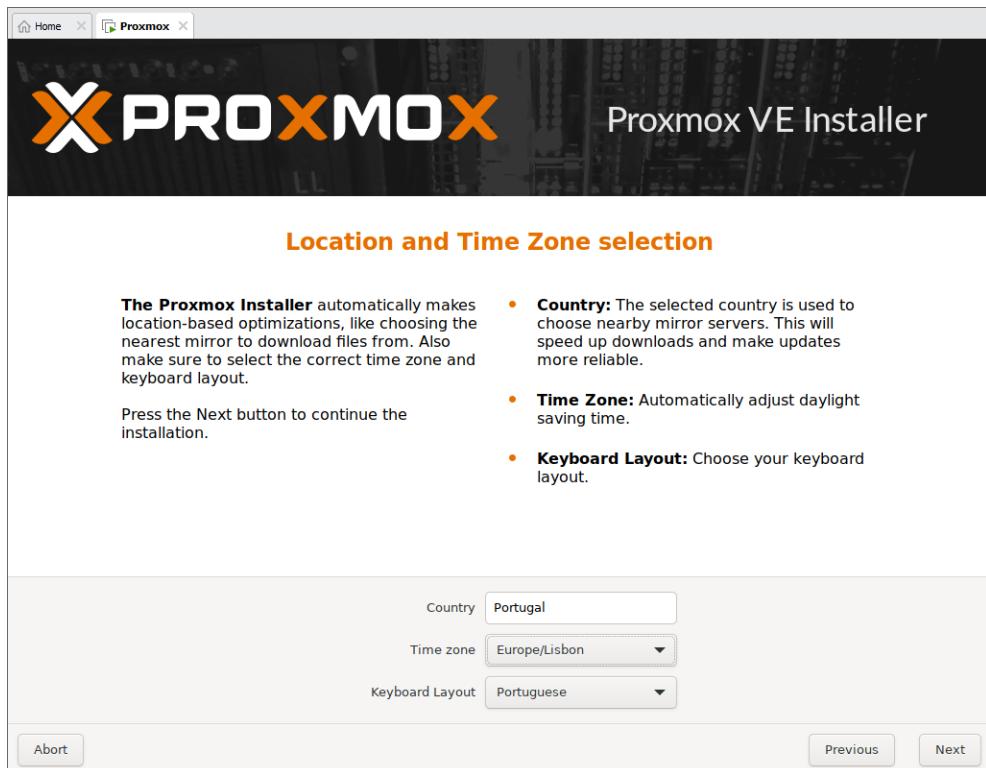


Figura 4.14: Localização e fuso horário

Passo 14: Definir uma *password* de administração e *email*.



Figura 4.15: Password de Administração e Email

Passo 15: Para a configuração da Rede de Gestão foi definido a placa de rede (ens33), hostname (DD1.VM), IP (192.168.1.129/24), gateway e *Domain Name System* (DNS).

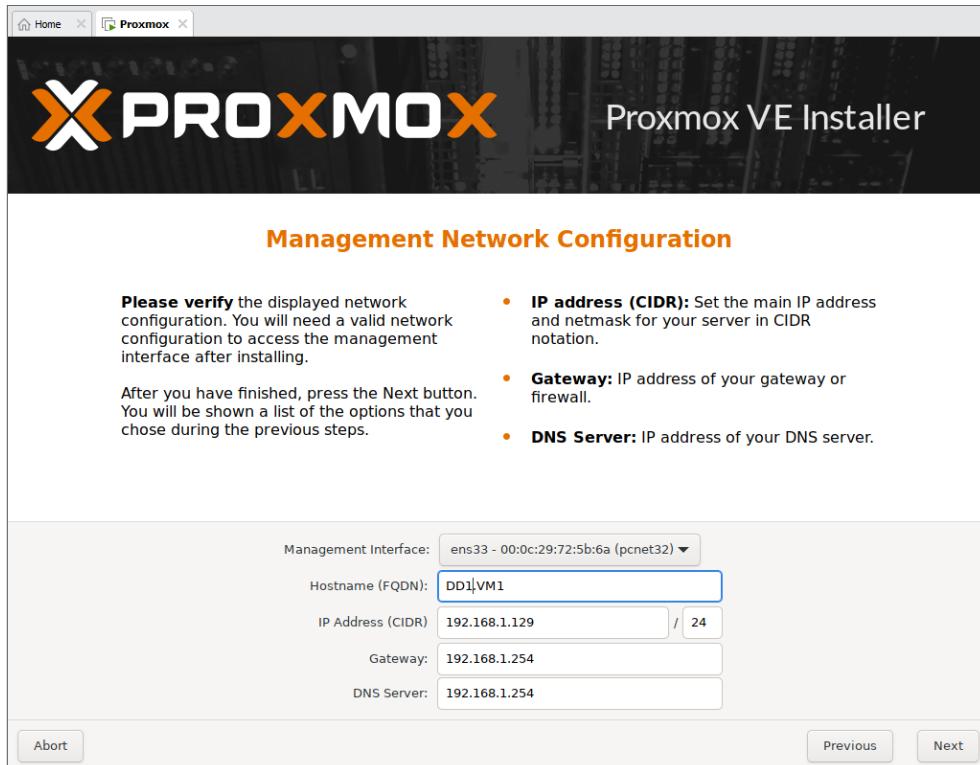


Figura 4.16: Configuração da Rede de Gestão

Passo 16: Aqui podemos verificar o resumo das opções de instalação e prosseguir com a instalação.

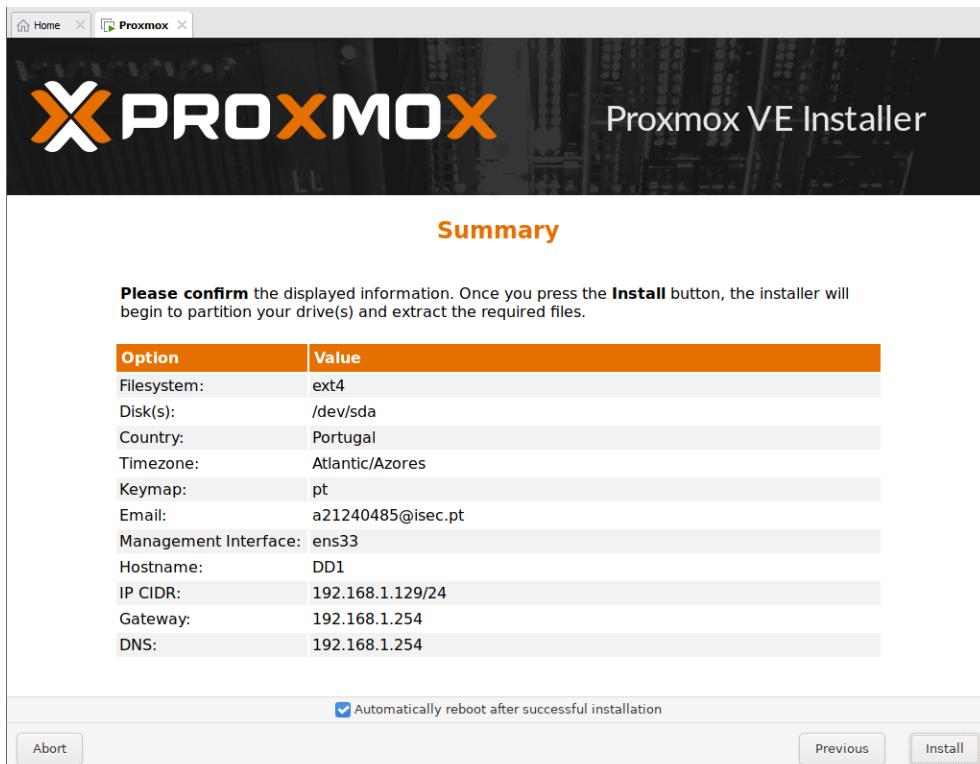


Figura 4.17: Resumo da Instalação

Passo 17: Instalação do proxmox.



Figura 4.18: Instalação

Passo 18: Após a instalação do proxmox é necessário reiniciar o servidor e de seguida fica pronto a ser utilizado. Na tela de apresentação do servidor é apresentado o endereço para poder aceder ao servidor proxmox através da interface web.

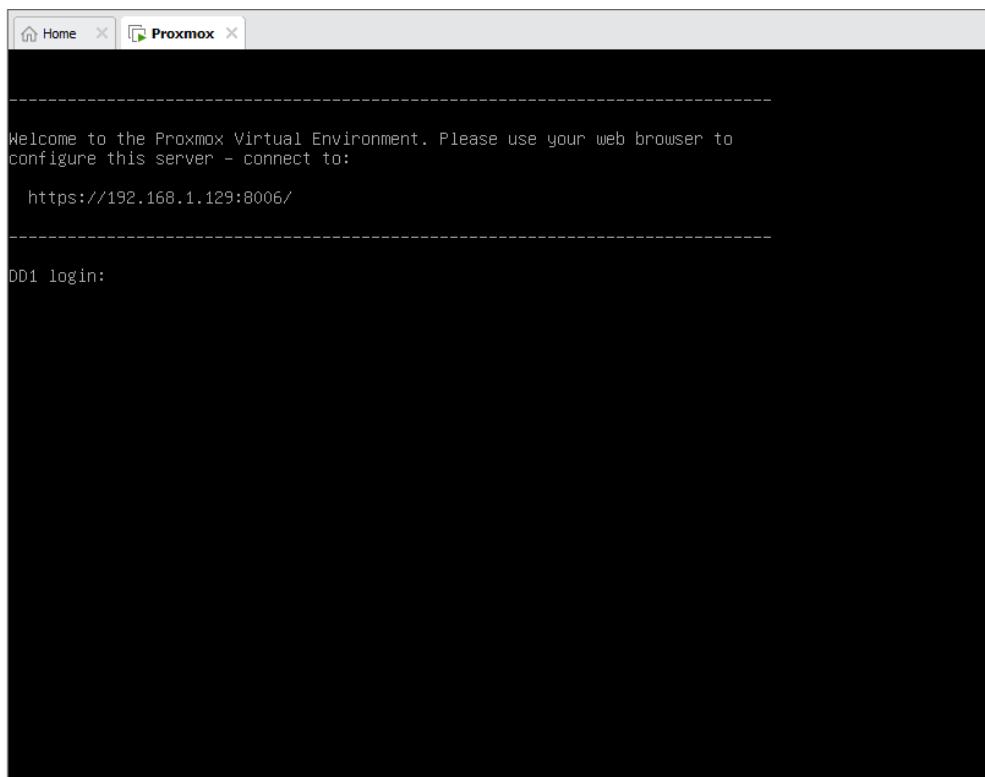


Figura 4.19: PVE login

Passo 19: Depois de verificar o PVE login no passo anterior, o mesmo nos indica o endereço para poder aceder à interface web. Num browser à escolha apenas basta conectar ao IP do servidor mais a porta 8006 do proxmox.

<https://192.168.1.129:8006>

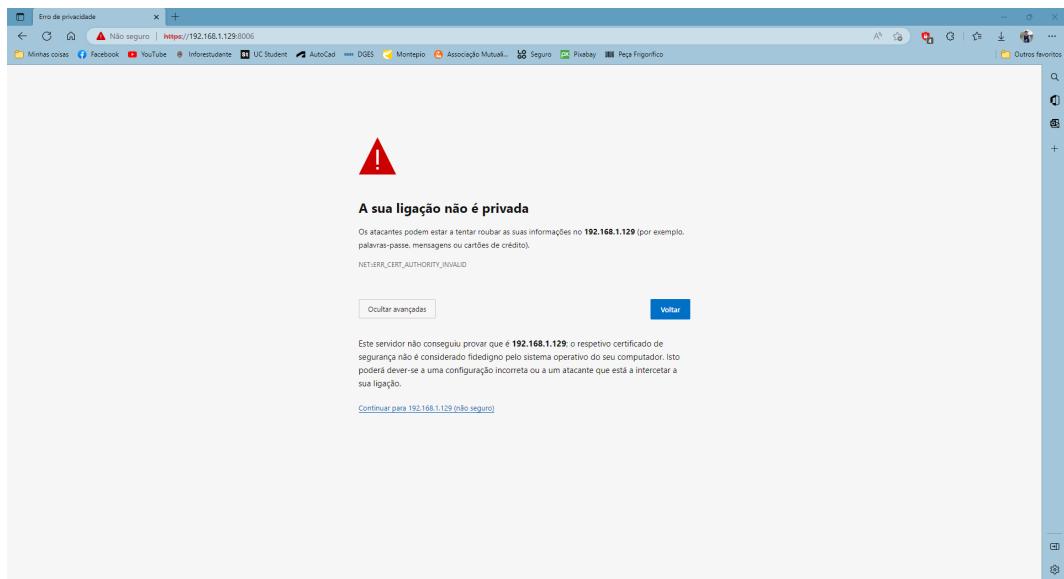


Figura 4.20: Ligação Não Privada

O browser vai mostrar o erro de que a ligação não é privada porque não existe um certificado válido mas prosseguimos carregando em: "**Continuar para 192.168.1.129 (não seguro)**". De seguida passamos para a tela inicial de login do servidor proxmox.

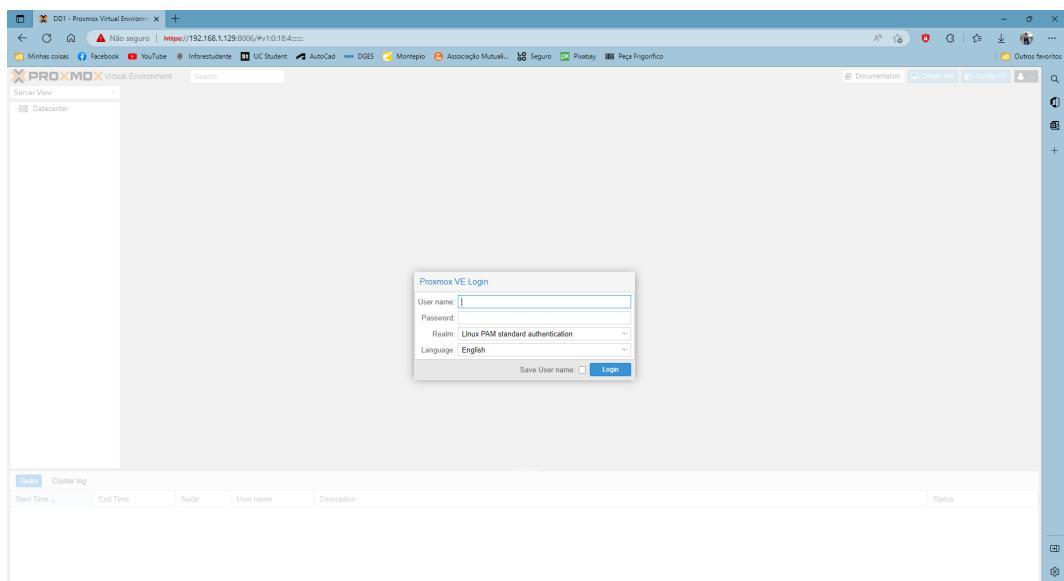


Figura 4.21: PVE login

O *username* por predefinição é *root* e a *password* foi definida no passo 14.

Passo 20: Depois da autenticação temos acesso completo ao servidor proxmox via interface web.

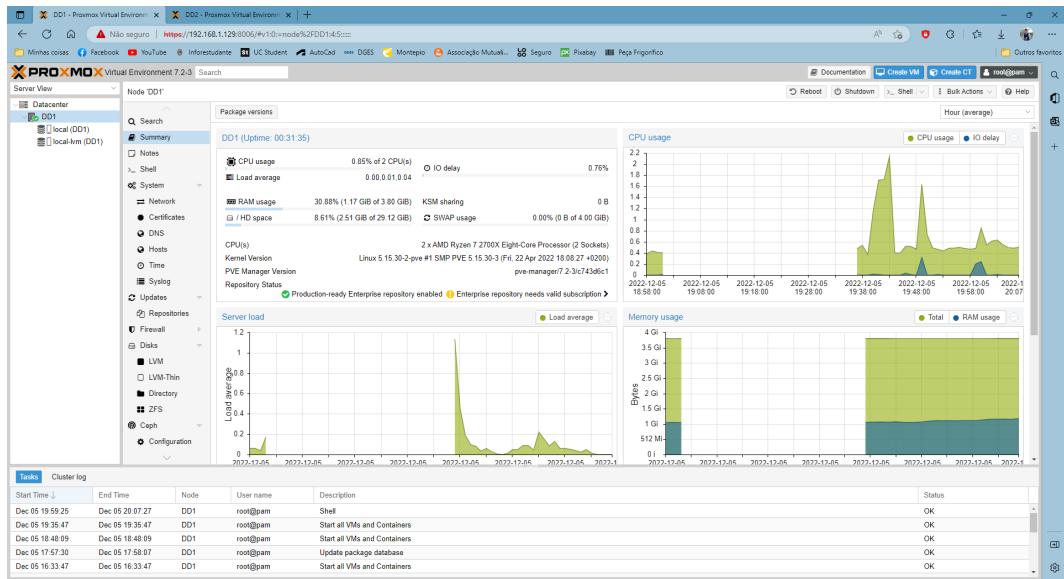


Figura 4.22: Summary DD1

O objetivo é a construção de um *cluster* e para isso é preciso no mínimo 3 servidores. De seguida foi adicionado mais 2 servidores proxmox seguindo todos os passos anteriores. Depois de repetir todos os passos anteriores o objetivo é ter 3 servidores proxmox a funcionar em simultâneo (DD1, DD2 e DD3) no VMWare:

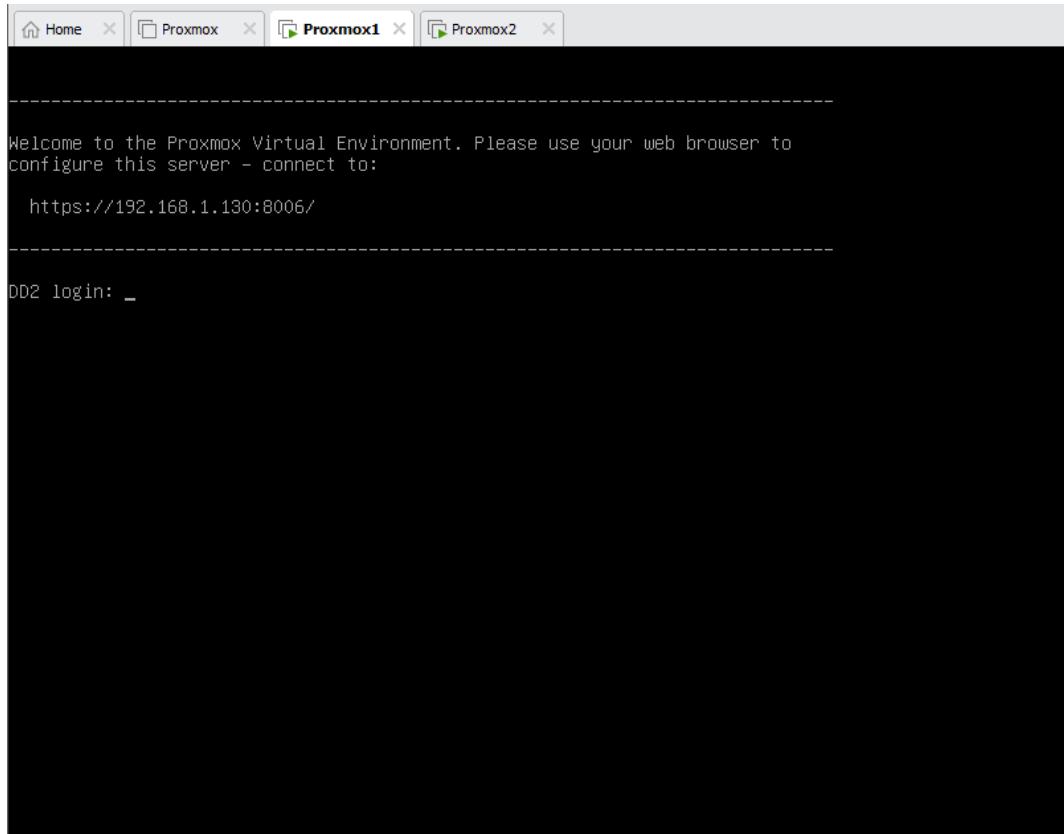


Figura 4.23: Servidor proxmox DD2

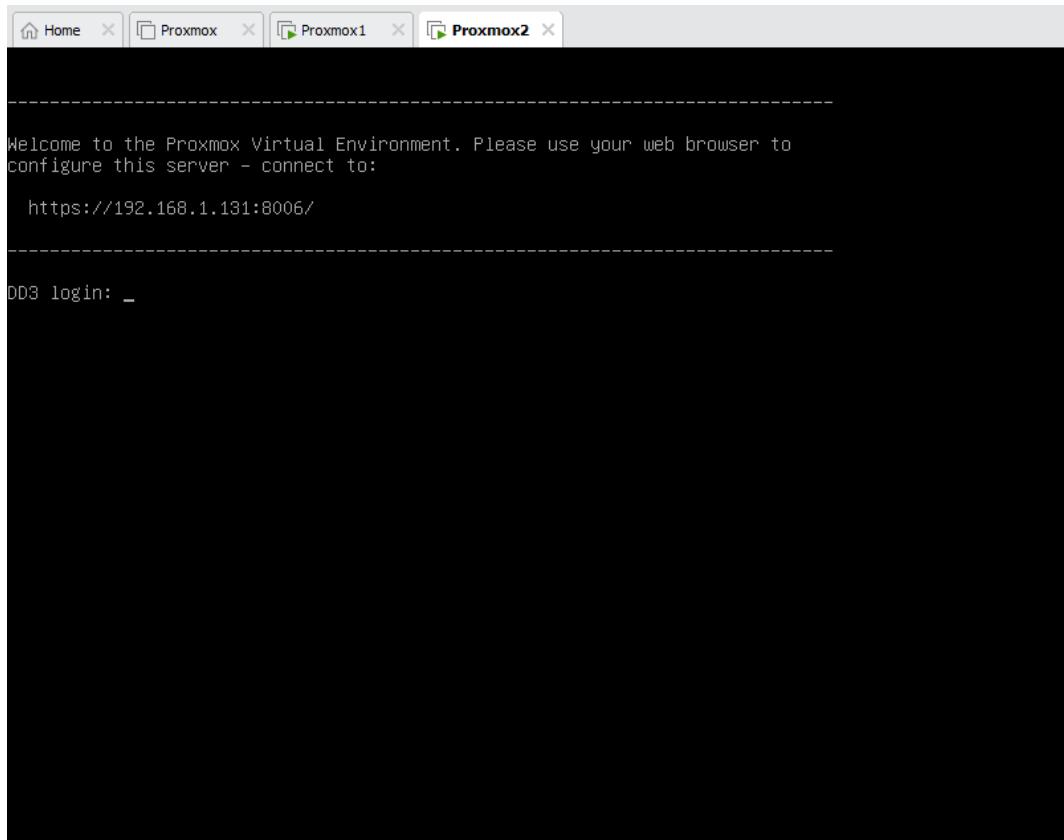


Figura 4.24: Servidor proxmox DD3

4.2 Configuração Ficheiro Hosts

O ficheiro "hosts" é usado principalmente para resolver nomes de host localmente, sem precisar de uma consulta ao servidor DNS. Ele também pode ser usado para bloquear o acesso a sites específicos, redirecionando o nome de host para um endereço IP inválido. Além disso, é frequentemente usado em ambientes de desenvolvimento para mapear nomes de host para endereços IP locais.

Desta forma prosseguimos com a configuração do ficheiro "hosts" em cada servidor. O ficheiro "hosts" encontra-se na seguinte diretoria:

```
/etc/hosts
```

Esta configuração foi realizada através da linha de comandos do servidor proxmox e para isso basta efetuar o *login* no servidor com o *username root* e a *password*.

Passo 1: Como foi dito anteriormente, é preciso ter acesso ao servidor proxmox e efetuar o *login*.

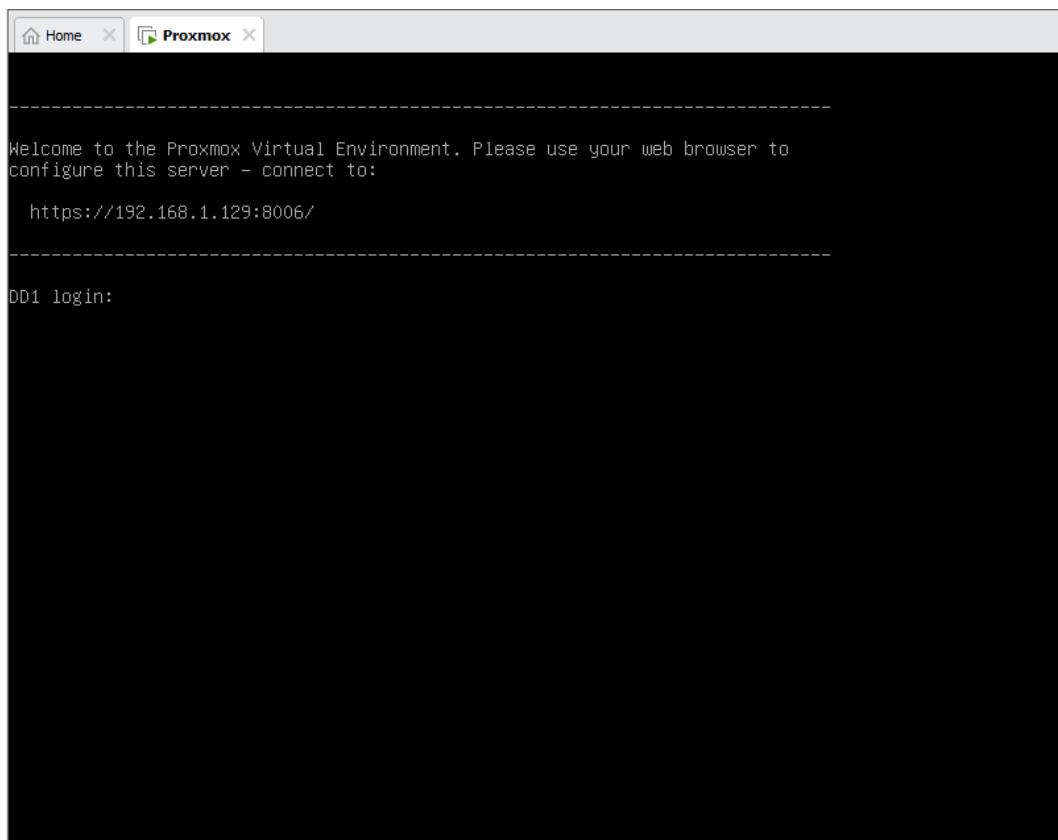


Figura 4.25: Login Servidor DD1

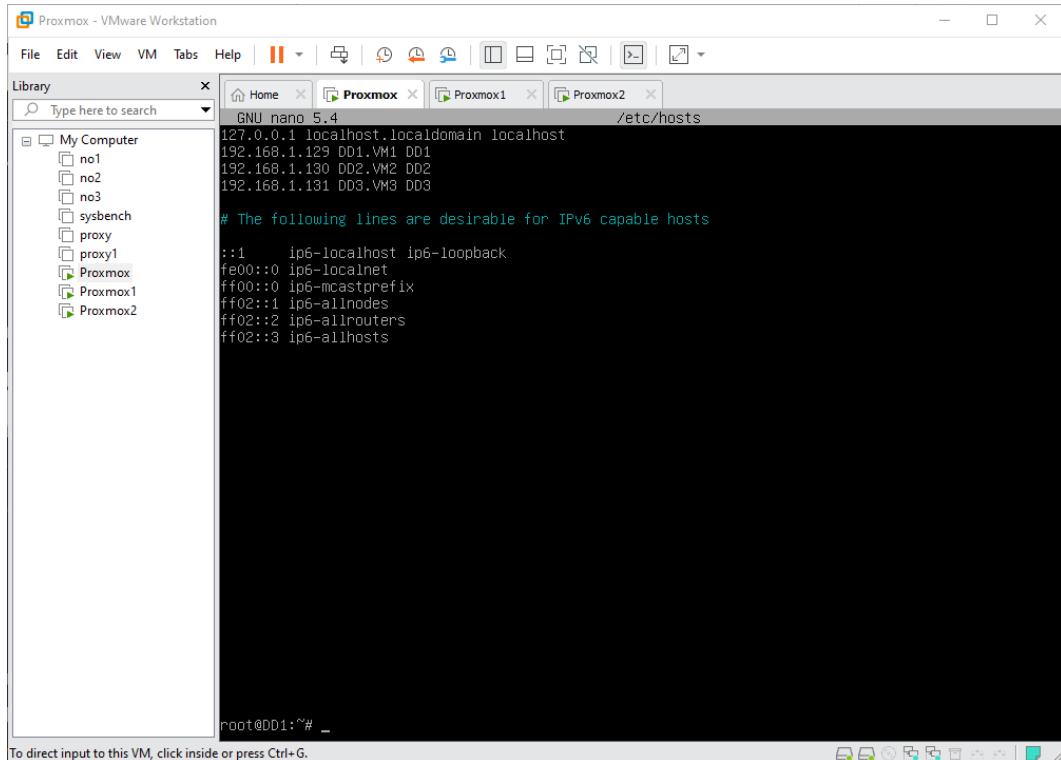
Passo 2: Depois de efetuar o *login*, com recurso ao editor de texto nano do linux, configuramos o ficheiro "hosts":

```
nano /etc/hosts
```

No servidor 1, Proxmox (DD1), foi adicionado o servidor 2 e 3.

```
192.168.1.130 DD2.VM2 DD2
```

```
192.168.1.131 DD3.VM3 DD3
```



The screenshot shows a VMware Workstation interface with a terminal window titled "Proxmox". The terminal is running the command "nano /etc/hosts". The contents of the file are as follows:

```
GNU nano 5.4
127.0.0.1 localhost.localdomain localhost
192.168.1.129 DD1.VM1 DD1
192.168.1.130 DD2.VM2 DD2
192.168.1.131 DD3.VM3 DD3

# The following lines are desirable for IPv6 capable hosts
::1      ip6-localhost ip6-loopback
fe00::0  ip6-localnet
ff00::0  ip6-mcastprefix
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters
ff02::3  ip6-allhosts
```

The terminal prompt at the bottom is "root@DD1:~#". A message at the bottom of the terminal window says "To direct input to this VM, click inside or press Ctrl+G."

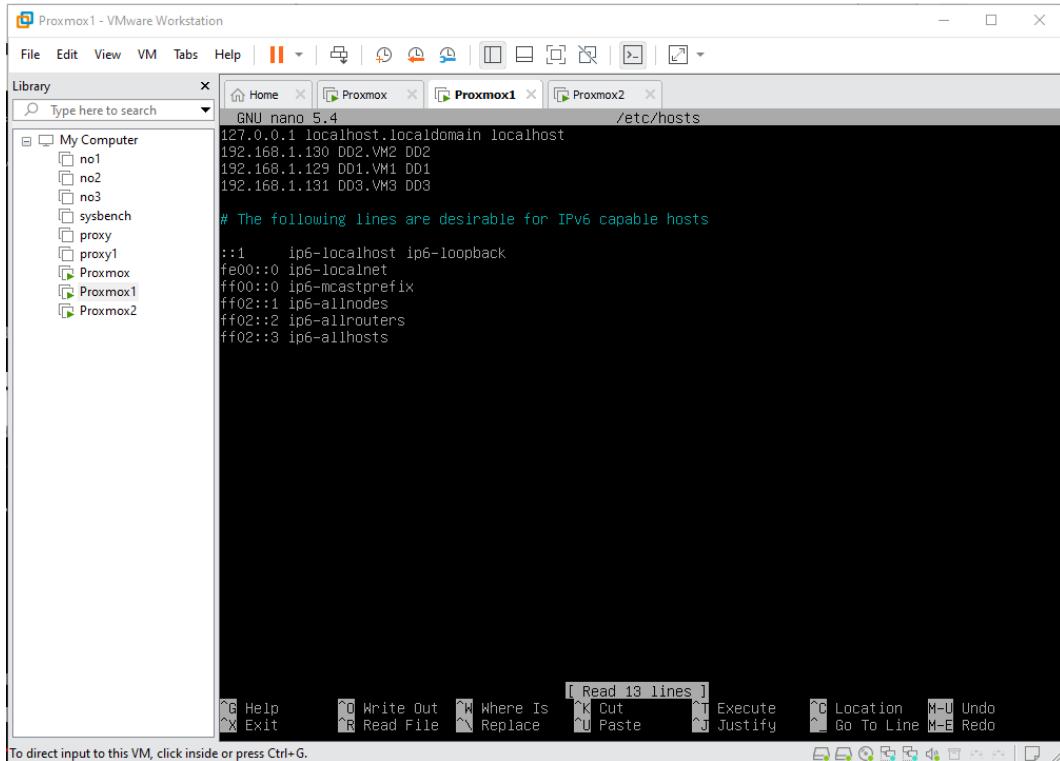
Figura 4.26: Configuração do ficheiro Hosts no servidor 1

Nos restantes servidores basta seguir os passos anteriores e adicionar os servidores corretos.

Servidor 2, Proxmox1 (DD2), foi adicionado o servidor 1 e 3.

192.168.1.129 DD1.VM1 DD1

192.168.1.131 DD3.VM3 DD3



```
GNU nano 5.4
127.0.0.1 localhost.localdomain localhost
192.168.1.130 DD2.VM2 DD2
192.168.1.129 DD1.VM1 DD1
192.168.1.131 DD3.VM3 DD3

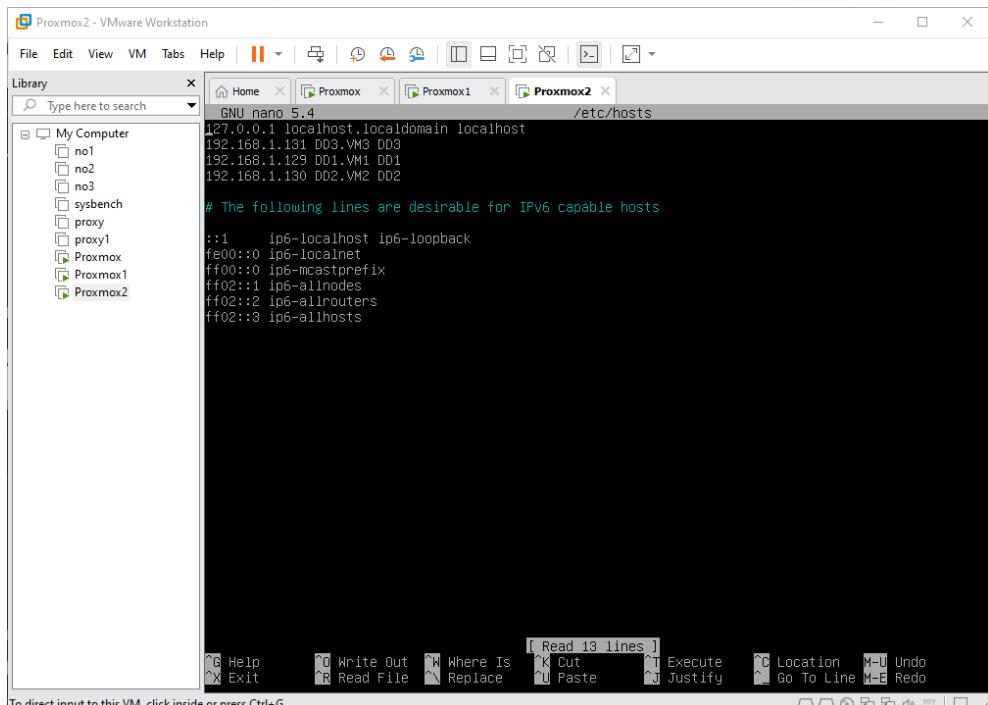
# The following lines are desirable for IPv6 capable hosts
::1      ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

Figura 4.27: Configuração do ficheiro Hosts no servidor 2

Servidor 3, Proxmox2 (DD3), foi adicionado o servidor 1 e 2.

192.168.1.129 DD1.VM1 DD1

192.168.1.130 DD2.VM2 DD2



```
GNU nano 5.4
127.0.0.1 localhost.localdomain localhost
192.168.1.131 DD3.VM3 DD3
192.168.1.129 DD1.VM1 DD1
192.168.1.130 DD2.VM2 DD2

# The following lines are desirable for IPv6 capable hosts
::1      ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

Figura 4.28: Configuração do ficheiro Hosts no servidor 3

4.3 Segunda Placa de Rede

Adicionar uma segunda placa de rede a um sistema é uma maneira de criar redundância e garantir a disponibilidade da rede em caso de falha de uma das placas de rede. É especialmente útil em ambientes de alta disponibilidade HA, onde é importante minimizar o tempo de inatividade (*downtime*).

Para criar redundância com duas ou mais placas de rede, precisamos de configurar as placas de rede de forma a trabalharem em conjunto. Isso pode ser feito de várias maneiras, dependendo do SO e do hardware utilizado. Algumas opções comuns são:

1. Configurar as placas de rede para trabalhar em conjunto como um link ativo-passivo: uma placa de rede é configurada como principal e a outra é configurada como secundária. A placa principal é usada para o tráfego de rede, enquanto a placa secundária é usada apenas em caso de falha da placa principal.
2. Configurar as placas de rede para trabalhar em conjunto como um link ativo-ativo: ambas as placas de rede são configuradas para trabalhar ao mesmo tempo e dividir o tráfego de rede. Isso pode ser feito usando técnicas como balanceamento de carga ou agregação de link.
3. Configurar as placas de rede como um *bond*: Um *bond* é uma configuração de rede que permite combinar múltiplas placas de rede num único link. Isso permite criar uma conexão mais rápida e mais confiável, pois as placas de rede trabalham em conjunto para transmitir e receber dados.

O meu objetivo com a segunda placa de rede foi criar boas práticas de implementação/configuração de rede apesar de no trabalho não aprofundar muito este tema (agregação de placas de rede) porque o principal foco não era este. Então, como acima mostrei (ambiente proxmox), tentei isolar a rede do cluster da rede de administração. Para isso criei uma segunda placa de rede para a rede do cluster (10.10.10.0).

De seguida, mostro passo a passo da criação e configuração da segunda placa de rede.

Passo 1: Em **Virtual Machine Settings** do servidor proxmox, na aba de **Hardware** carregar em **Add....**

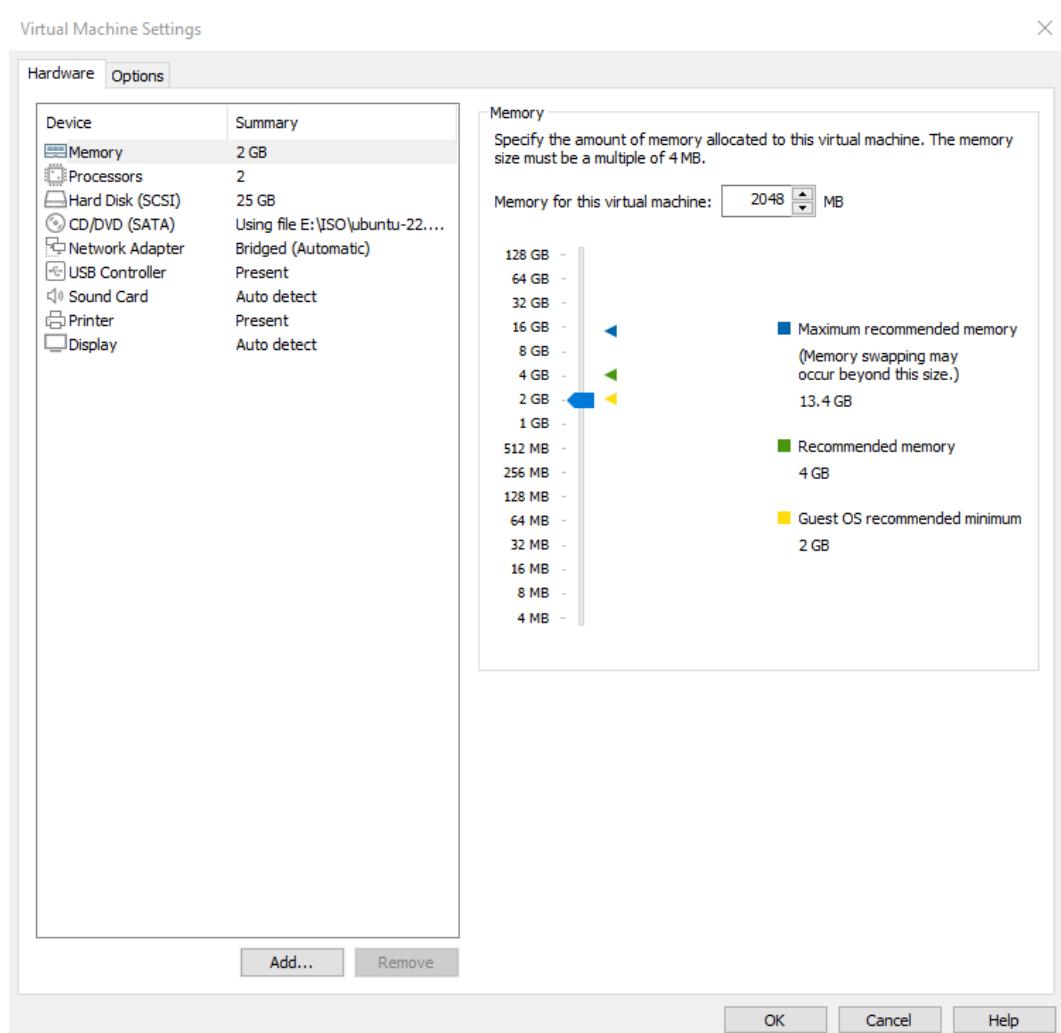


Figura 4.29: Definições do servidor proxmox

Passo 2: Como é uma placa de rede escolhemos **Network Adapter**.

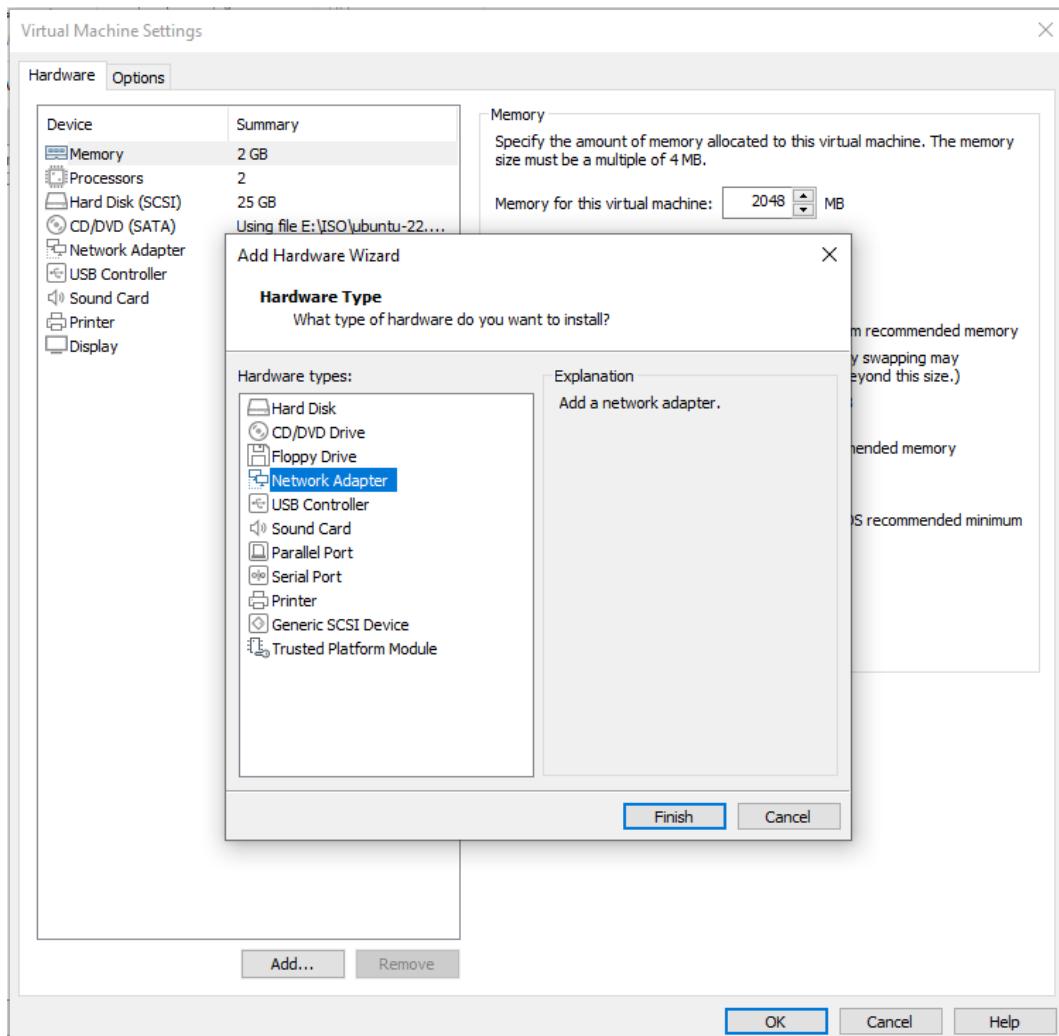


Figura 4.30: Add Hardware Wizard

Passo 3: Por fim, colocar a nova placa de rede (**Network Adapter 2**) em modo *bridged*.

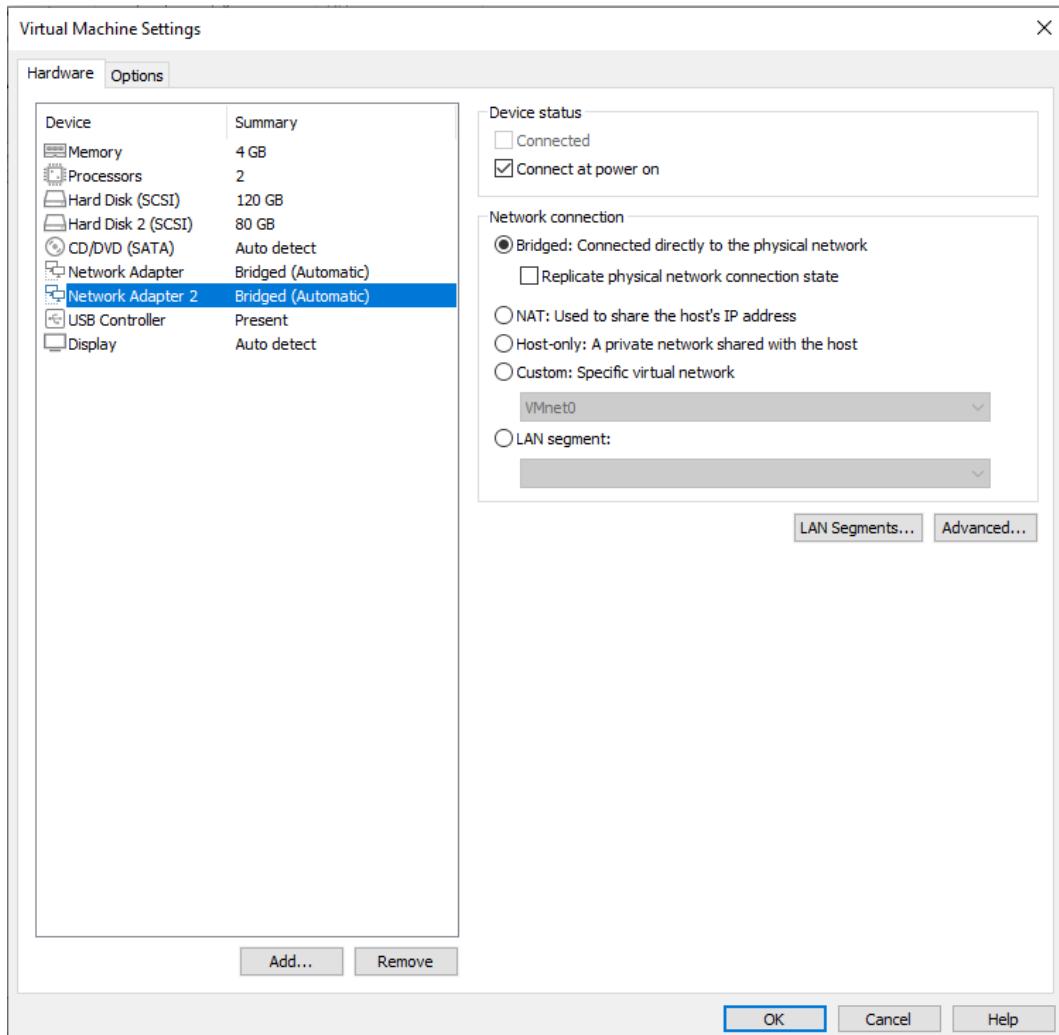


Figura 4.31: Modo Bridge Placa de Rede

Para os outros 2 servidores é só repetir estes 3 passos anteriores.

Passo 4: Depois de adicionar a nova placa de rede aos servidores e colocar em modo *bridged*, na interface web dos 3 servidores proxmox já conseguimos ver a nova placa de rede (ens37). Para fazer a gestão da placas de rede no proxmox vamos ao nosso **Datacenter** e escolhemos o nosso servidor, no meu caso DD2, depois **System** e **Network**. Por fim, foi só atribuir o IP (10.10.10.130) à nova placa de rede carregando em **Edit**. Não esquecer de colocar a *checkbox Autostart* marcada para a placa de rede iniciar sempre com o servidor.

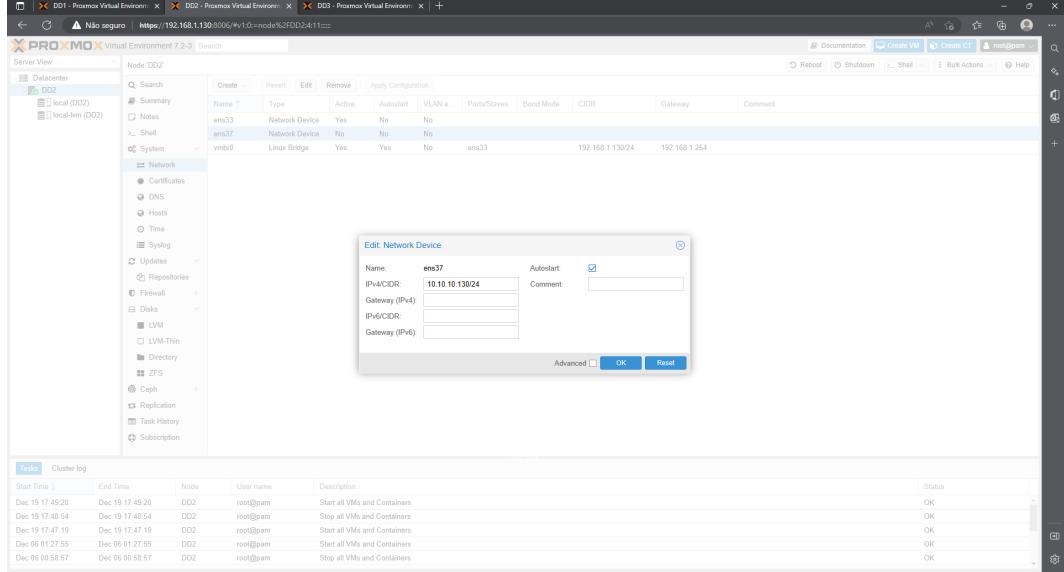


Figura 4.32: IP Segunda Placa de Rede

Passo 5: Depois de atribuir o IP à placa de rede é sempre importante e necessário reiniciar o sistema.

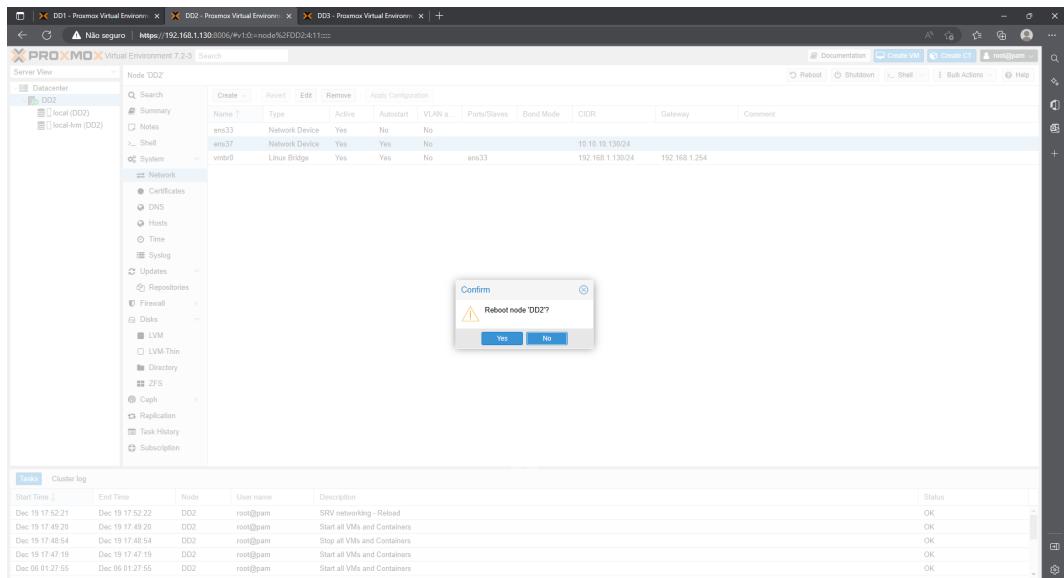


Figura 4.33: Restart Node

Os passos anteriores são repetidos nos restantes servidores.

Sem recorrer à interface gráfica do servidor também podemos configurar os passos anteriores através de linha de comandos, basta abrir a **shell** do servidor e temos acesso ao SO. Como fiz a configuração na interface gráfica podemos verificar o ficheiro *interfaces* do linux que contém a configuração das placas de rede.

```
nano /etc/network/interfaces
```

The screenshot shows the Proxmox VE 7.2-3 interface. On the left, the Server View sidebar shows a cluster named 'Datacenter (ClusterTPB)' with three nodes: D01, D02, and D03. The 'Shell' option under the 'System' section is selected. In the main pane, a terminal window displays the contents of the /etc/network/interfaces file. The file contains configuration for several interfaces, including auto lo, iface ens33, iface ens34, and auto vmbro. Below the terminal, a 'Cluster log' table lists recent events:

Start Time	End Time	Node	User name	Description	Status
Dec 20 16:37:54		D01	root@team	Shell	
Dec 20 16:31:21	Dec 20 16:31:34	D03	root@team	Join Cluster	OK
Dec 20 16:28:32	Dec 20 16:28:44	D02	root@team	Join Cluster	OK
Dec 20 16:26:43	Dec 20 16:26:45	D01	root@team	Create Cluster	OK
Dec 20 16:21:36	Dec 20 16:21:37	D01	root@team	Shell	OK

Figura 4.34: Ficheiro configuração interfaces

4.4 Conectividade

Feitas as instalações e configurações necessárias para obter um sistema funcional e comunicativo precisamos de verificar se os servidores conseguem comunicar entre si para poder avançar no projeto. Na **shell** de cada servidor foi feito um *ping* para os restantes servidores para testar a conectividade.

DD1: Conseguiu "pingar" o servidor DD2 e DD3 e ainda o exterior (1.1.1.1).

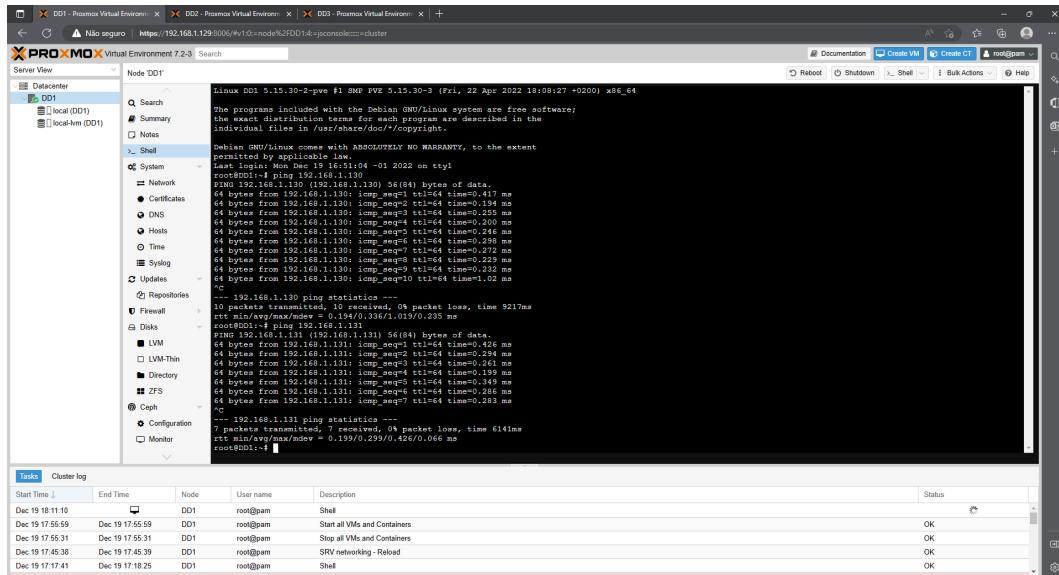


Figura 4.35: Ping para DD2 e DD3

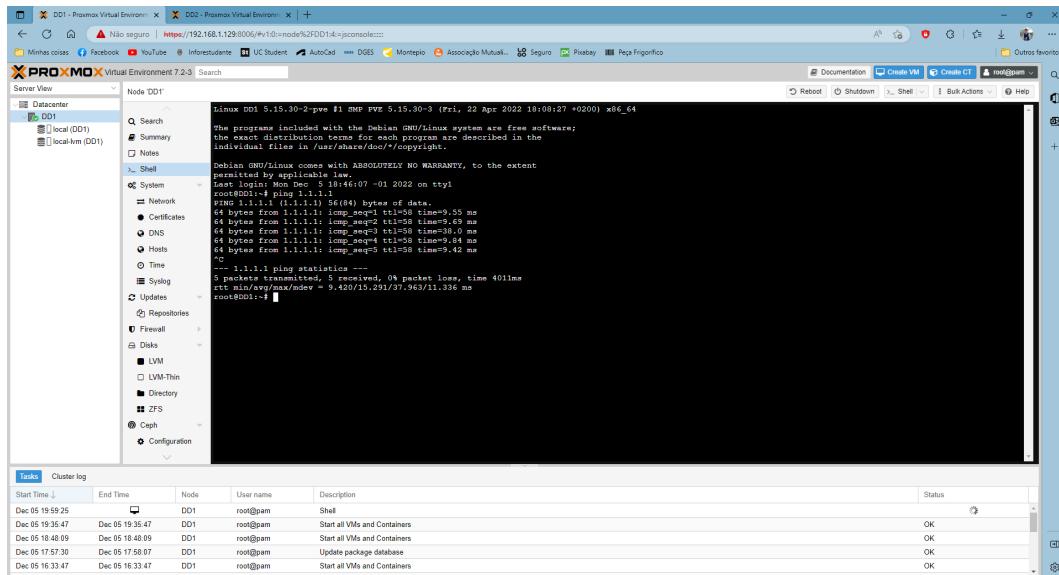
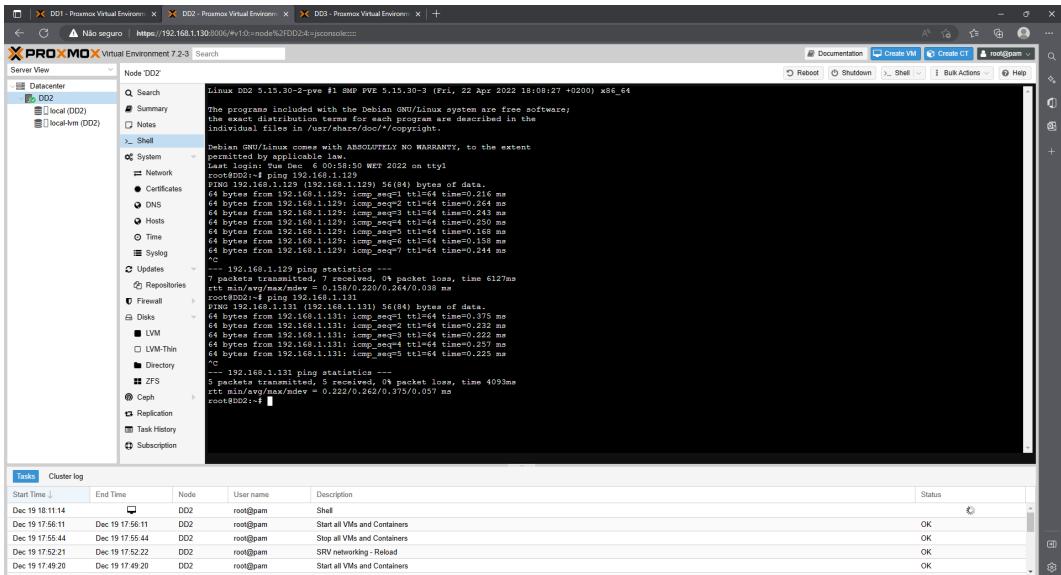


Figura 4.36: Ping para o exterior

DD2: Conseguiu "pingar" o servidor DD1 e DD3.



```

Linux DD2 5.15.30-2-pve #1 SMP PVE 5.15.30-3 (Fri, 22 Apr 2022 18:08:27 +0200) x86_64
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright*.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

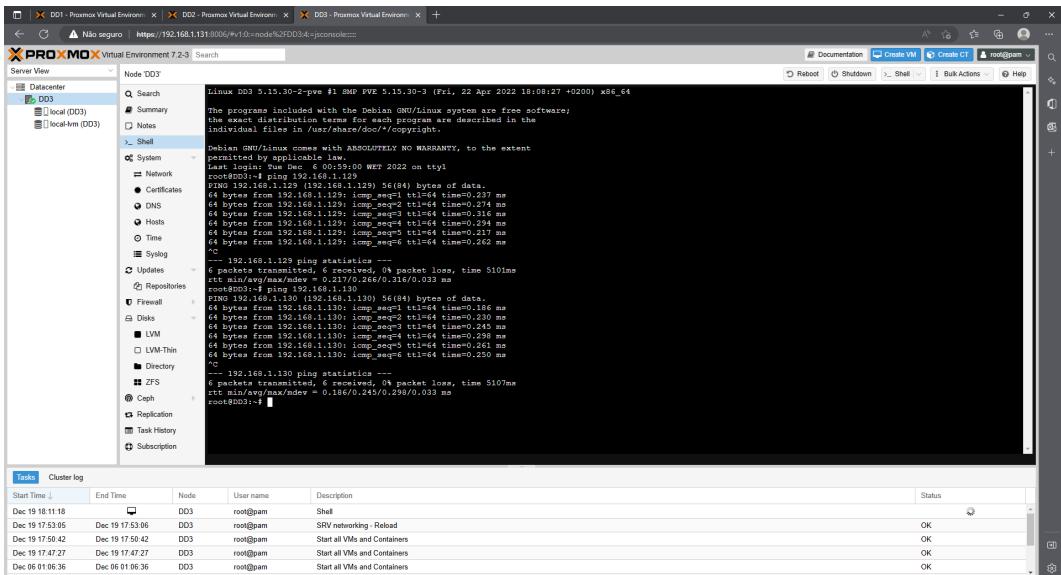
Last login: Tue Dec 6 00:58:50 WST 2022 on tty1
root@DD2:~# ping 192.168.1.129 (192.168.1.129) 56(84) bytes of data.
64 bytes from 192.168.1.129: icmp_seq=1 ttl=64 time=0.216 ms
64 bytes from 192.168.1.129: icmp_seq=2 ttl=64 time=0.223 ms
64 bytes from 192.168.1.129: icmp_seq=3 ttl=64 time=0.243 ms
64 bytes from 192.168.1.129: icmp_seq=4 ttl=64 time=0.250 ms
64 bytes from 192.168.1.129: icmp_seq=5 ttl=64 time=0.257 ms
64 bytes from 192.168.1.129: icmp_seq=6 ttl=64 time=0.258 ms
64 bytes from 192.168.1.129: icmp_seq=7 ttl=64 time=0.244 ms
...
--- 192.168.1.129 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6127ms
rtt min/avg/max/mdev = 0.216/0.220/0.249/0.038 ms
root@DD2:~# ping 192.168.1.131 (192.168.1.131) 56(84) bytes of data.
64 bytes from 192.168.1.131: icmp_seq=1 ttl=64 time=0.237 ms
64 bytes from 192.168.1.131: icmp_seq=2 ttl=64 time=0.222 ms
64 bytes from 192.168.1.131: icmp_seq=3 ttl=64 time=0.222 ms
64 bytes from 192.168.1.131: icmp_seq=4 ttl=64 time=0.257 ms
64 bytes from 192.168.1.131: icmp_seq=5 ttl=64 time=0.225 ms
...
--- 192.168.1.131 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4093ms
rtt min/avg/max/mdev = 0.222/0.226/0.275/0.057 ms
root@DD2:~#

```

Start Time	End Time	Node	User name	Description	Status
Dec 18 11:14		DD2	root@team	Shell	
Dec 19 17:56:11	Dec 19 17:56:11	DD2	root@team	Start all VMs and Containers	OK
Dec 19 17:55:44	Dec 19 17:55:44	DD2	root@team	Stop all VMs and Containers	OK
Dec 19 17:52:21	Dec 19 17:52:22	DD2	root@team	SRV networking - Reload	OK
Dec 19 17:49:20	Dec 19 17:49:20	DD2	root@team	Start all VMs and Containers	OK

Figura 4.37: Ping para DD1 e DD3

DD3: Conseguiu "pingar" o servidor DD1 e DD2.



```

Linux DD3 5.15.30-2-pve #1 SMP PVE 5.15.30-3 (Fri, 22 Apr 2022 18:08:27 +0200) x86_64
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright*.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

Last login: Tue Dec 6 00:59:00 WST 2022 on tty1
root@DD3:~# ping 192.168.1.129 (192.168.1.129) 56(84) bytes of data.
64 bytes from 192.168.1.129: icmp_seq=1 ttl=64 time=0.237 ms
64 bytes from 192.168.1.129: icmp_seq=2 ttl=64 time=0.274 ms
64 bytes from 192.168.1.129: icmp_seq=3 ttl=64 time=0.283 ms
64 bytes from 192.168.1.129: icmp_seq=4 ttl=64 time=0.294 ms
64 bytes from 192.168.1.129: icmp_seq=5 ttl=64 time=0.217 ms
64 bytes from 192.168.1.129: icmp_seq=6 ttl=64 time=0.262 ms
...
--- 192.168.1.129 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 510ms
rtt min/avg/max/mdev = 0.217/0.226/0.293 ms
root@DD3:~# ping 192.168.1.130 (192.168.1.130) 56(84) bytes of data.
64 bytes from 192.168.1.130: icmp_seq=1 ttl=64 time=0.186 ms
64 bytes from 192.168.1.130: icmp_seq=2 ttl=64 time=0.230 ms
64 bytes from 192.168.1.130: icmp_seq=3 ttl=64 time=0.248 ms
64 bytes from 192.168.1.130: icmp_seq=4 ttl=64 time=0.298 ms
64 bytes from 192.168.1.130: icmp_seq=5 ttl=64 time=0.261 ms
64 bytes from 192.168.1.130: icmp_seq=6 ttl=64 time=0.250 ms
...
--- 192.168.1.130 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 510ms
rtt min/avg/max/mdev = 0.186/0.245/0.298/0.033 ms
root@DD3:~#

```

Start Time	End Time	Node	User name	Description	Status
Dec 19 18:11:18		DD3	root@team	Shell	
Dec 19 17:53:05	Dec 19 17:53:06	DD3	root@team	SRV networking - Reload	OK
Dec 19 17:50:42	Dec 19 17:50:42	DD3	root@team	Start all VMs and Containers	OK
Dec 19 17:47:27	Dec 19 17:47:27	DD3	root@team	Start all VMs and Containers	OK
Dec 06 01:06:36	Dec 06 01:06:36	DD3	root@team	Start all VMs and Containers	OK

Figura 4.38: Ping para DD1 e DD3

Por fim, o DD1 ainda conseguiu "pingar" a segunda interface de rede de DD2 (192.168.1.130).

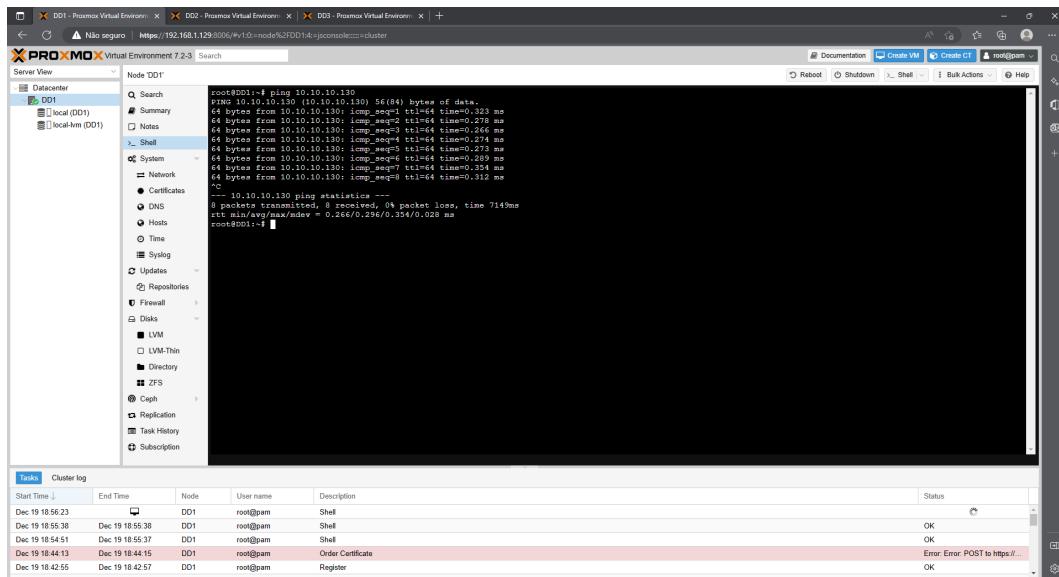


Figura 4.39: Ping para segunda interface de rede DD2

Devido à configuração que foi feita inicialmente do ficheiro "hosts" também testei, no servidor DD2, "pingar" por nome o servidor DD1.

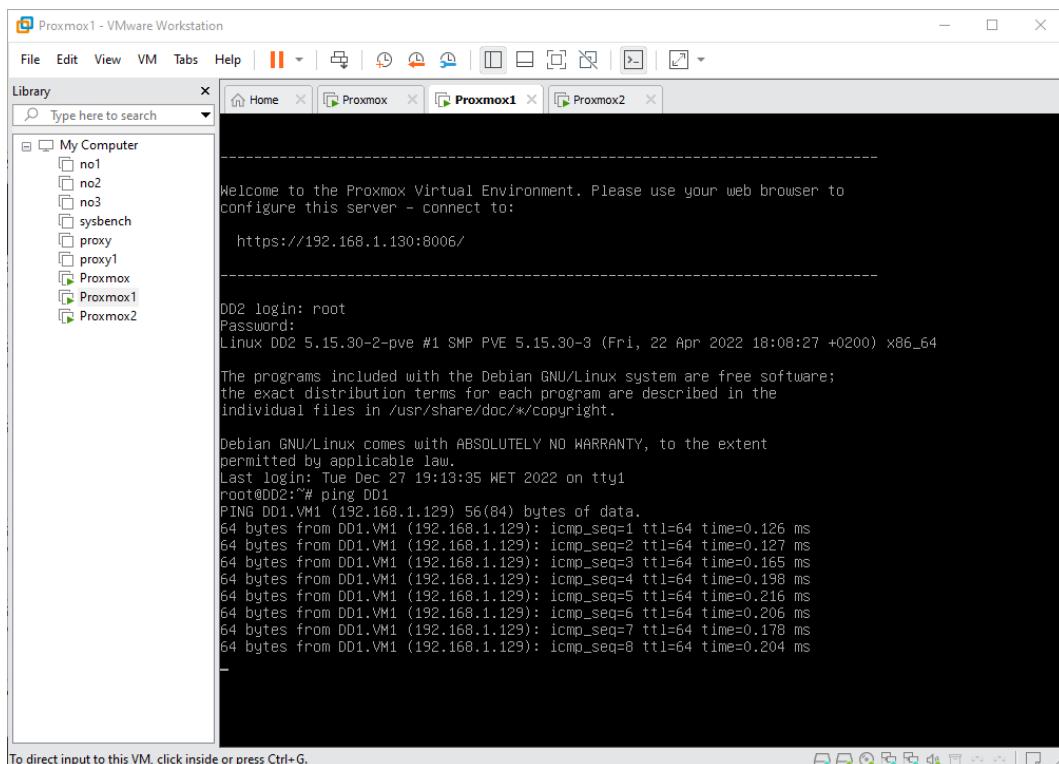


Figura 4.40: Ping para segunda interface de rede DD2

4.5 Criação do Cluster

O cluster, como já referido, é o funcionamento em conjunto de várias VM. Para isso passamos à criação e configuração do cluster em proxmox.

Passo 1: Escolhi criar o cluster no servidor DD1 e os restantes apenas se conectavam ao cluster criado em DD1. O cluster é criado no **Datacenter-Cluster-Create Cluster**.

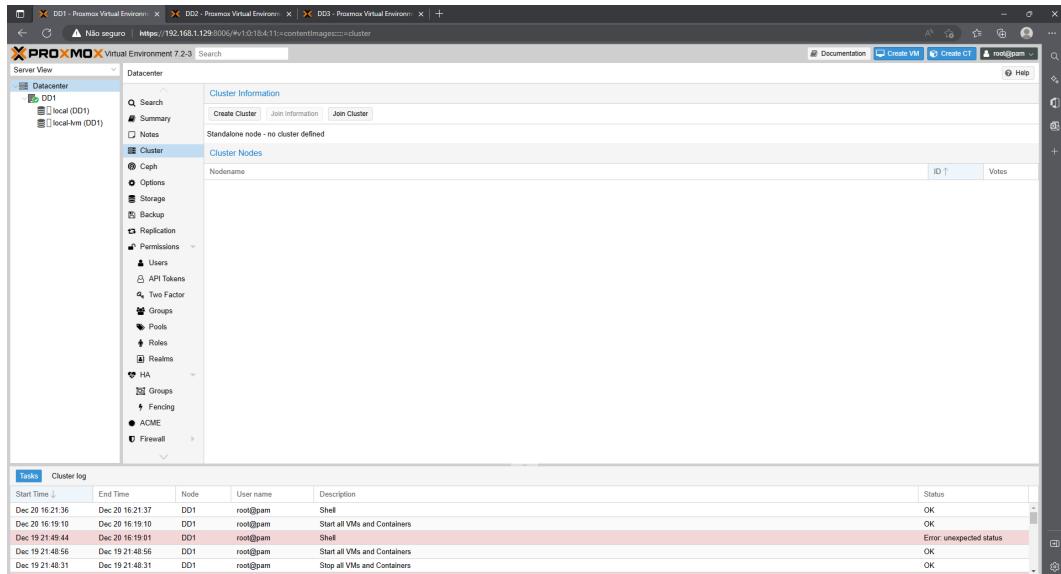


Figura 4.41: Cluster

Passo 2: Nesta passo vamos atribuir um nome ao cluster, no meu caso chamei **clusterTPB**, e atribuir um ou mais IP para criar redundância. Inseri apenas o IP da rede do cluster (**10.10.10.129**).

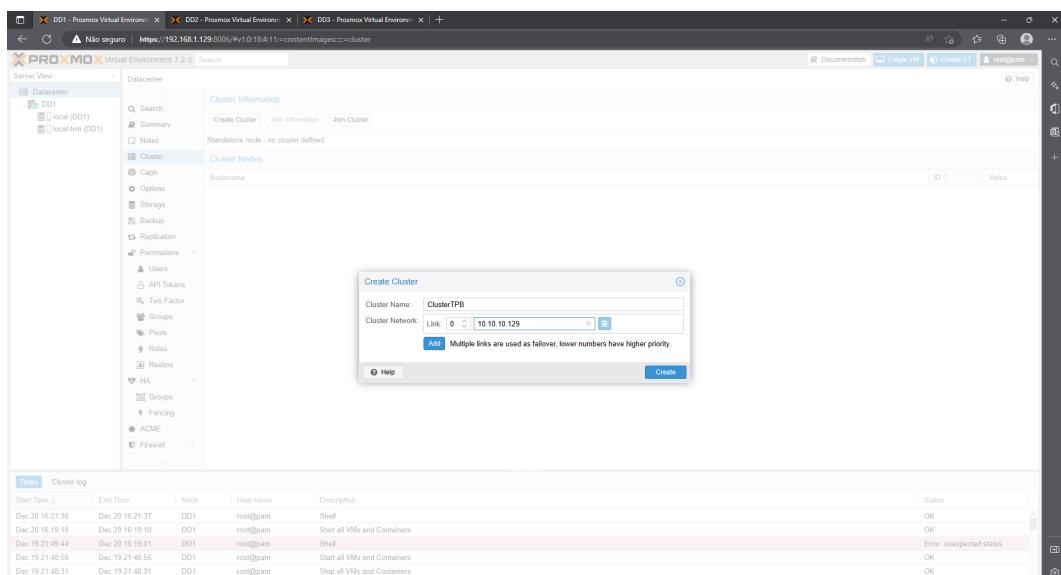


Figura 4.42: Create Cluster

De seguida podemos verificar alguns processos a serem iniciados, automaticamente, pelo proxmox, nomeadamente a configuração do *corosync*. No final quando aparecer a informação de **TASK OK** podemos fechar a janela do *Task Viewer* e o cluster está criado.

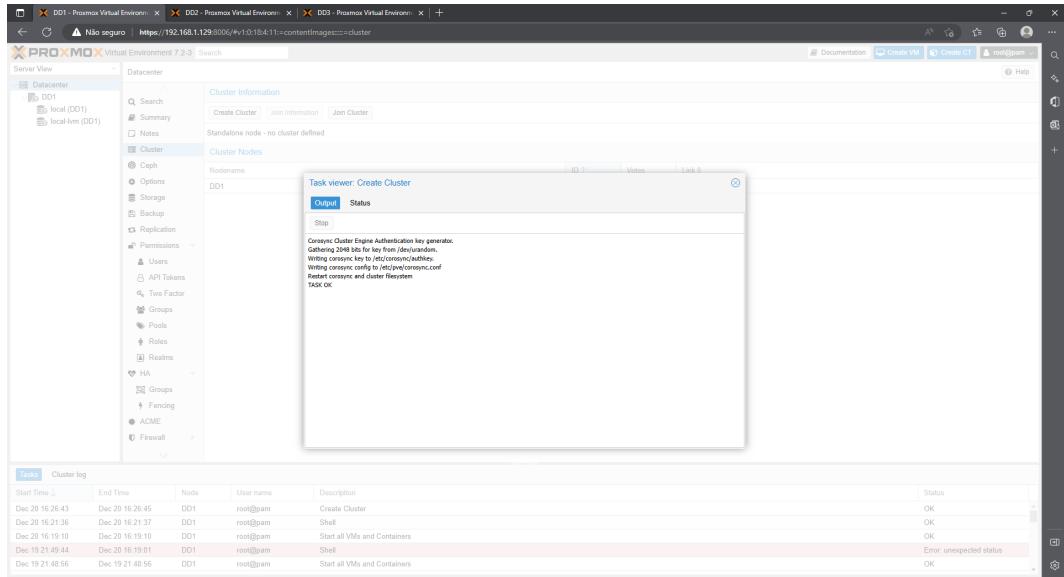


Figura 4.43: Create Cluster Task Viewer

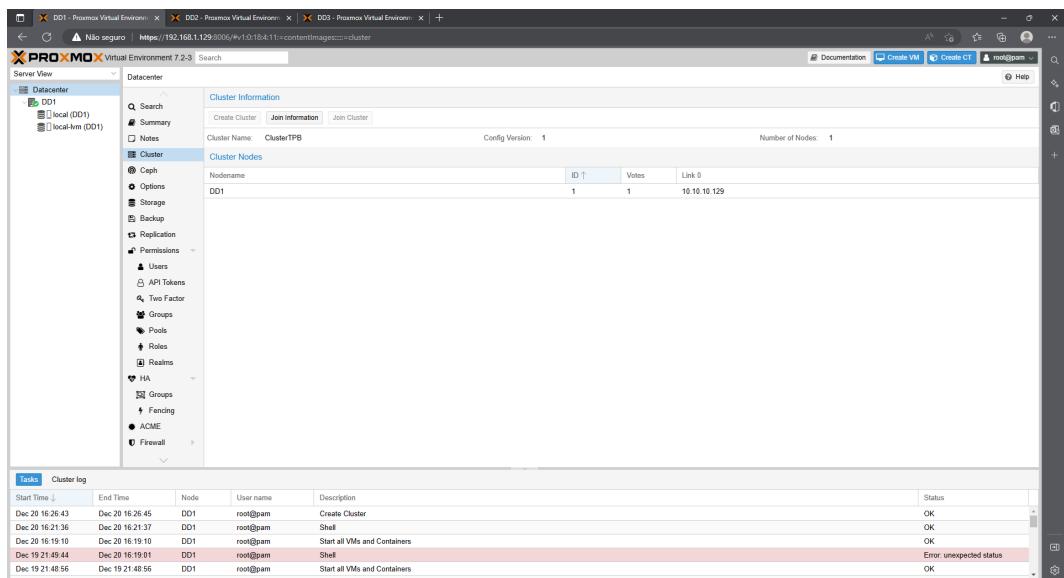


Figura 4.44: Cluster Nodes DD1

Um passo importante antes de adicionar os restantes nós ao cluster é copiar um conjunto de informações que se encontram em **Join Information** do nó criado (DD1). Apenas basta carregar em **Copy Information** e é selecionado e copiado automaticamente toda a informação.

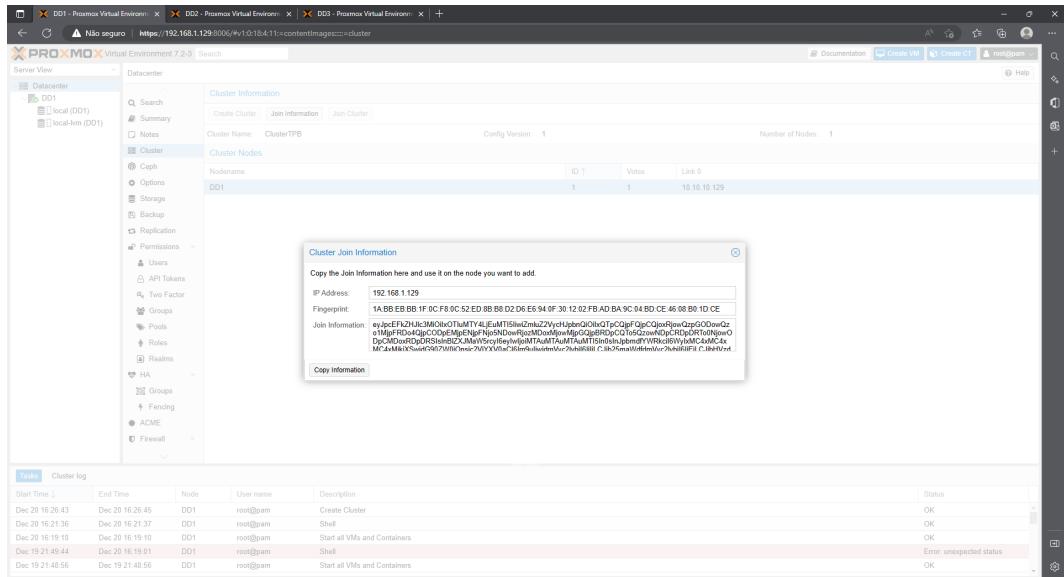


Figura 4.45: Copy Information

Passo 3: No seguinte servidor, DD2, vamos novamente ao **Datacenter-Cluster** e agora não vamos criar um cluster porque já existe um. Vamos adicionar o nó DD2 ao cluster em DD1. Para isso precisamos de ir a **Join Cluster**.

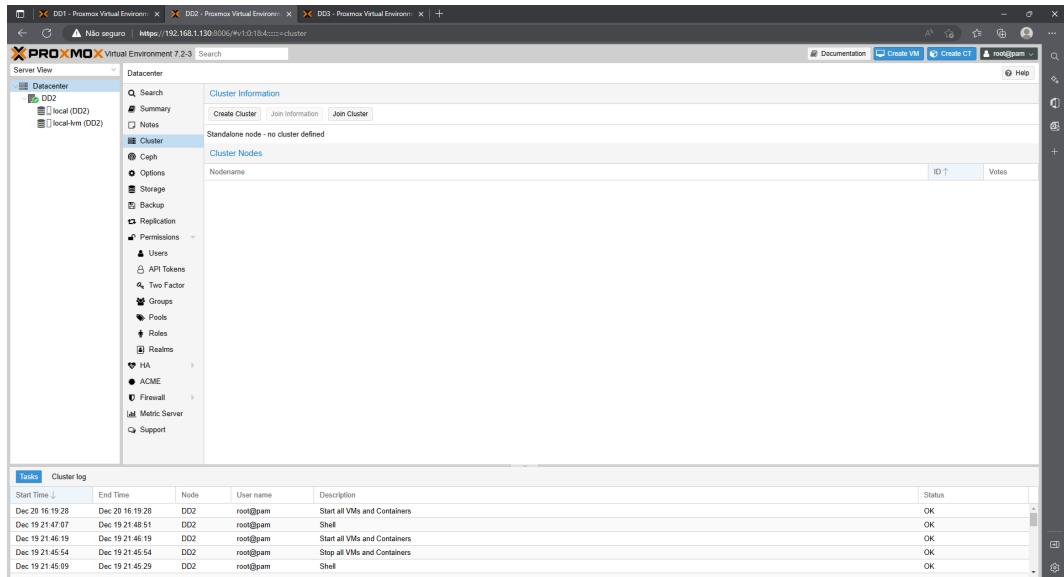


Figura 4.46: Join Cluster

Passo 4: No campo **Information** é onde vai ser colado a informação copiada do cluster em DD1. **Peer Address** é o IP de administração do servidor DD1, 192.168.1.129. **Cluster Network** atribuiu o IP da rede do cluster de DD2, 10.10.10.130. Por último inserir uma **Password**.

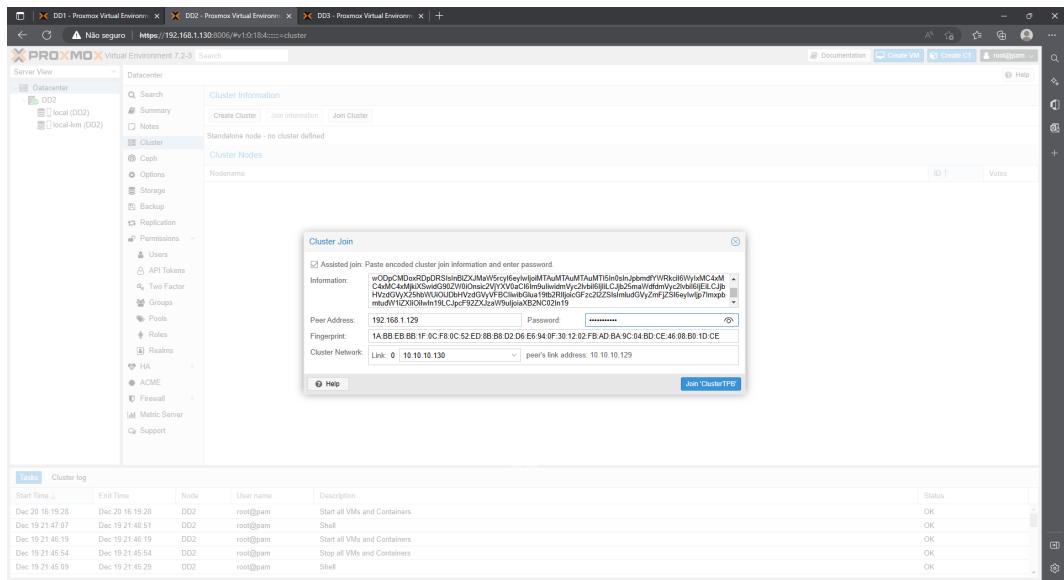


Figura 4.47: Join Cluster

No passo anterior depois de carregar em **Join ClusterTPB** podemos observar que no servidor DD2 já aparecem 2 nós, DD1 e DD2. Em DD1 também ja podemos ver no **Datacenter** os nós do cluster.

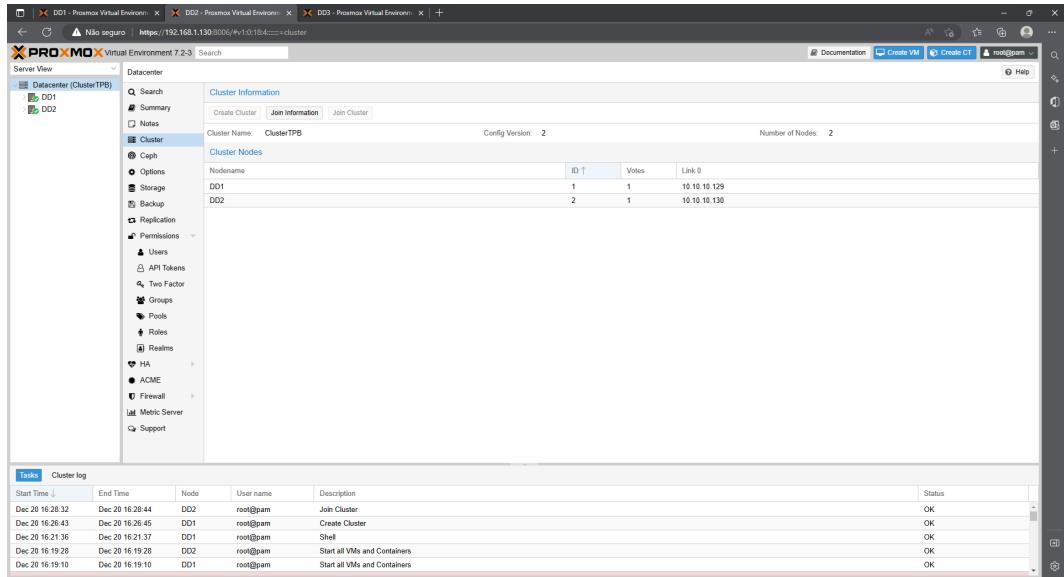


Figura 4.48: Cluster Nodes DD2

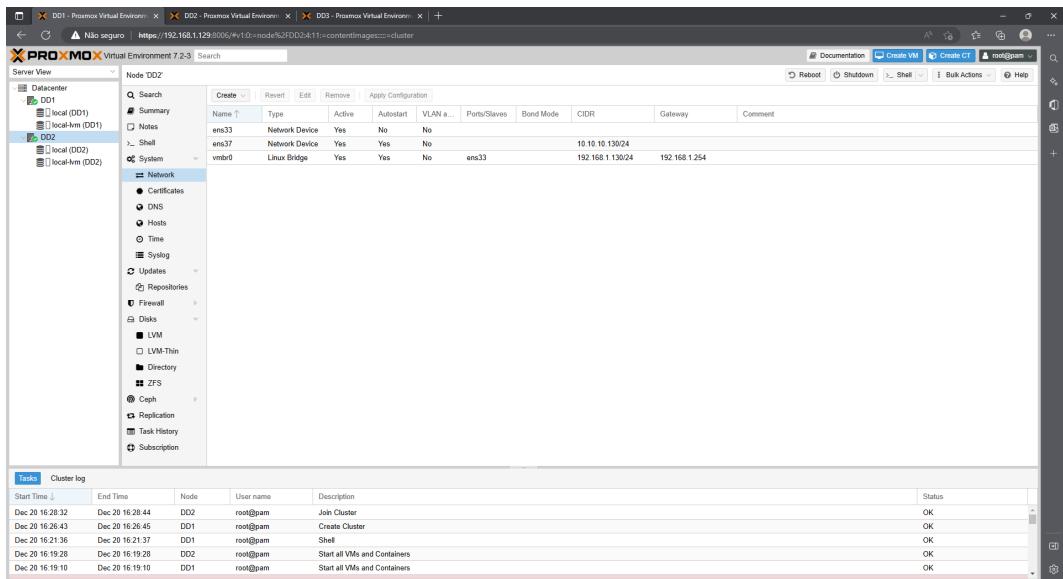


Figura 4.49: Cluster Nodes DD1

Para o cluster ficar completo, no servidor DD3 repetimos o passo 3 e 4 e obtemos o cluster com os 3 nós.

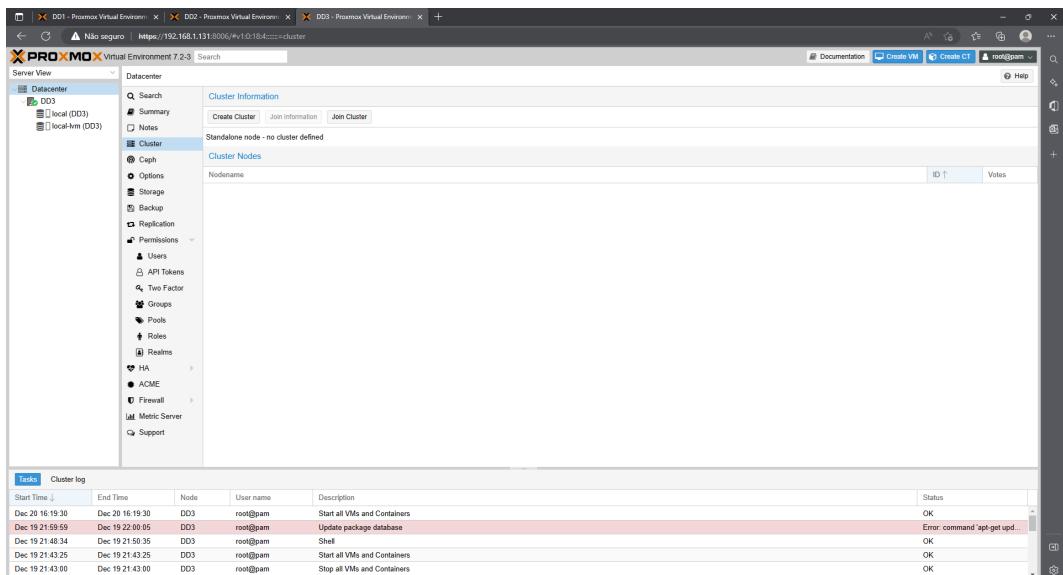


Figura 4.50: Cluster DD3

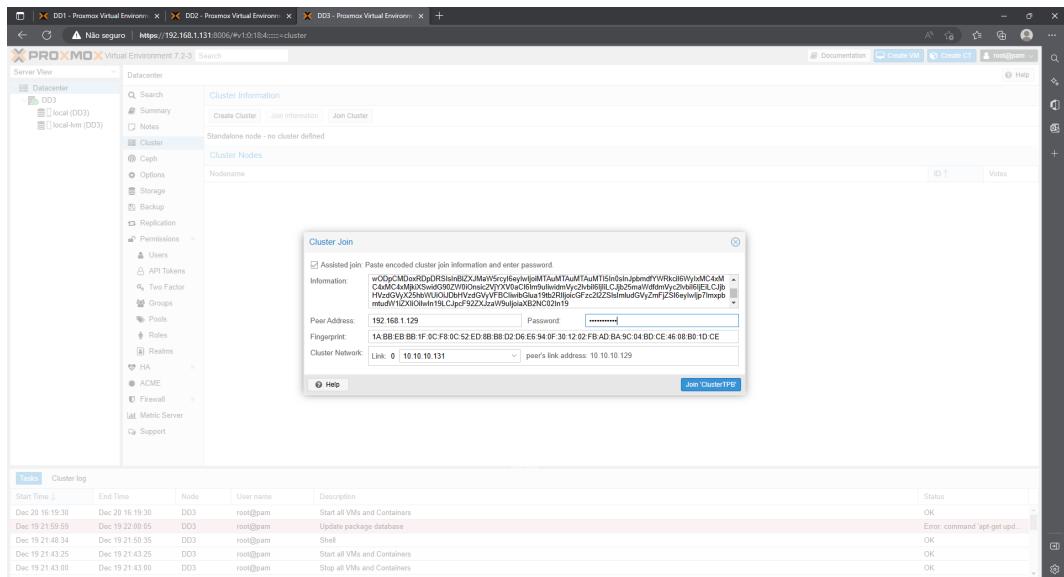


Figura 4.51: Cluster Join

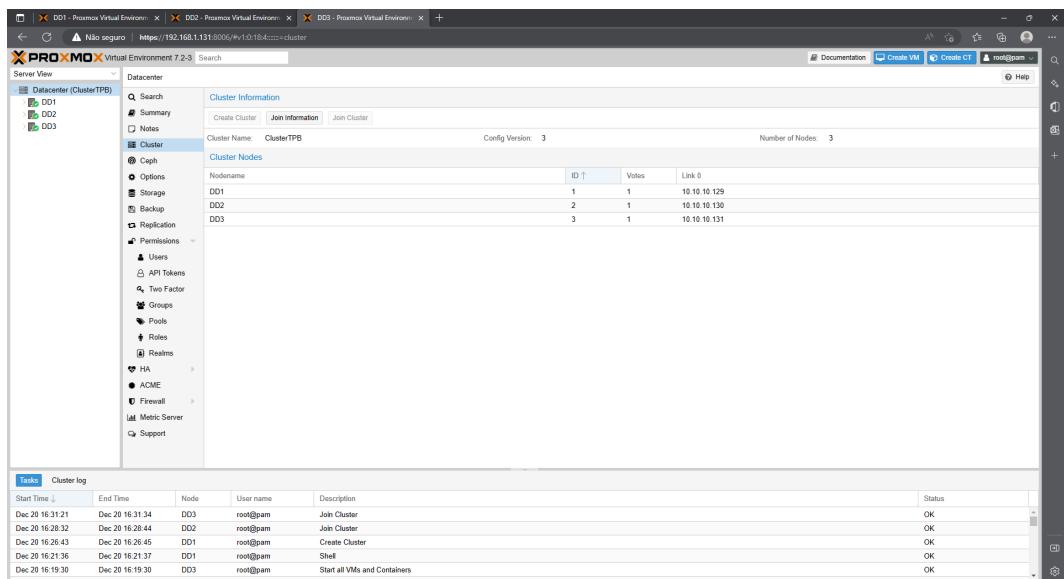


Figura 4.52: Cluster Nodes DD3

Em DD1 também já podemos ver o cluster completo.

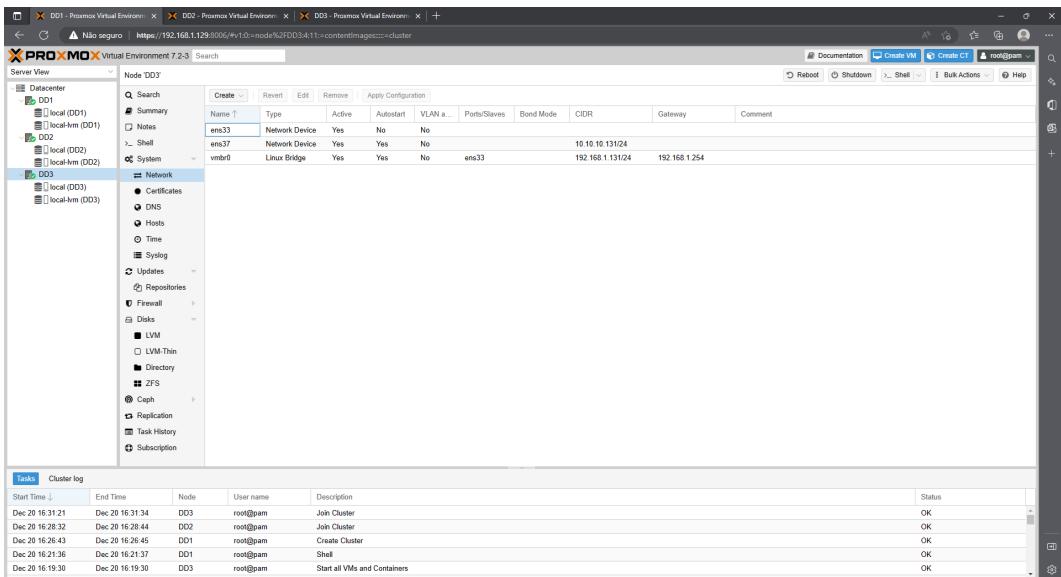


Figura 4.53: Cluster DD1

A partir de agora pode-se trabalhar apenas numa interface web de um servidor à escolha, eu usei sempre o DD1, 192.168.1.129. Como o cluster já está completo a replicação já está a funcionar entre os nós, por isso o que for feito num nó será replicado entre os restantes.

Por fim, podemos observar a configuração do ficheiro *corosync*:

```
GNU nano 5.4                               /etc/pve/corosync.conf
logging {
    debug: off
    to_syslog: yes
}

nodegrid {
    node {
        name: DD1
        nodeid: 1
        quorum_votes: 1
        ring0_addr: 10.10.10.129
    }
    node {
        name: DD2
        nodeid: 2
        quorum_votes: 1
        ring0_addr: 10.10.10.130
    }
    node {
        name: DD3
        nodeid: 3
        quorum_votes: 1
        ring0_addr: 10.10.10.131
    }
}

quorum {
    provider: corosync_votequorum
}

totem {
    cluster_name: ClusterTPB
    config_version: 3
}
```

Figura 4.54: Configuração Corosync

```

GNU nano 5.4
    ring0_addr: 10.10.10.129
}
node {
    name: DD2
    nodeid: 2
    quorum_votes: 1
    ring0_addr: 10.10.10.130
}
node {
    name: DD3
    nodeid: 3
    quorum_votes: 1
    ring0_addr: 10.10.10.131
}
}

quorum {
    provider: corosync_votequorum
}

totem {
    cluster_name: ClusterTPB
    config_version: 3
    interface {
        linknumber: 0
    }
    ip_version: ipv4-6
    link_mode: passive
    secauth: on
    version: 2
}

```

^G Help ^O Write Out ^W Where Is ^K Cut ^U Paste ^I Execute ^C Location M-U Undo
^X Exit ^R Read File ^L Replace ^J Justify ^P Go To Line M-E Redo

Figura 4.55: Configuração Corosync

Para observar algumas informações acerca do cluster, nodes, resources do Datacenter entre outros, podemos observar em **Datacenter-Summary**.

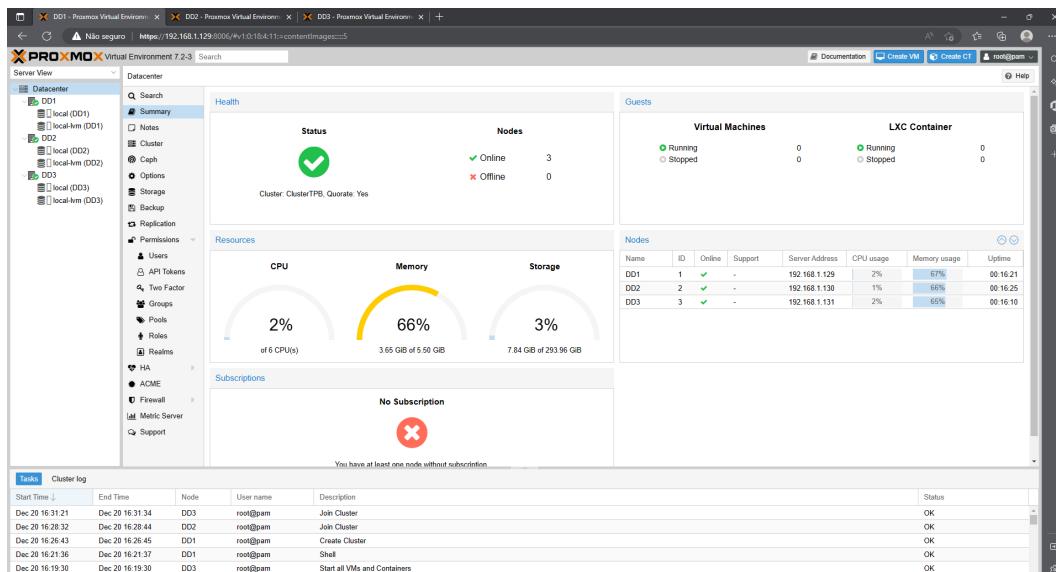


Figura 4.56: Datacenter Summary

4.6 Ceph

Para criar o sistema de armazenamento distribuído passamos para a instalação e configuração do *Ceph*.

Passo 1: Instalação do *Ceph* em cada "nó".

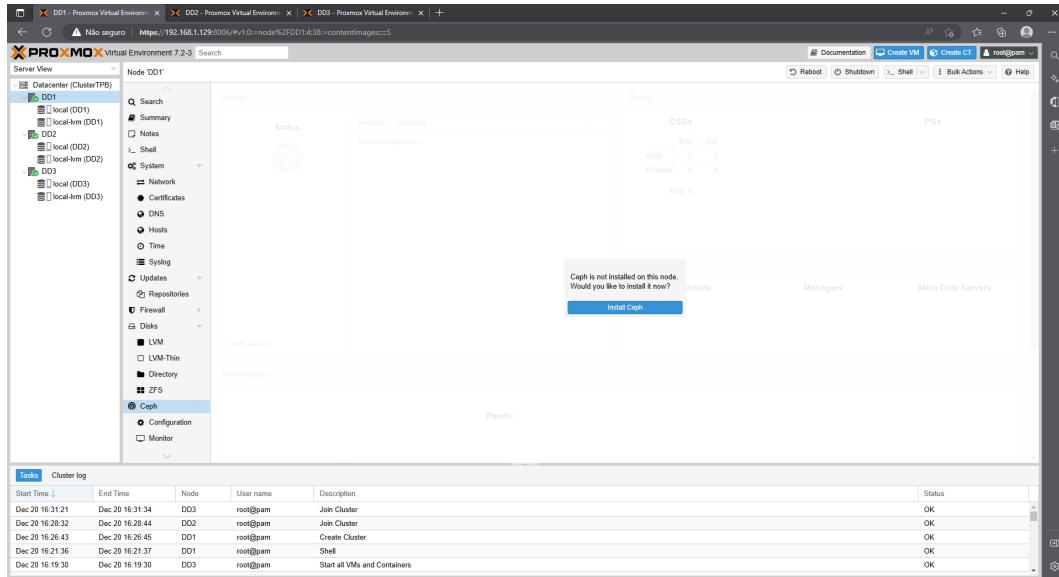


Figura 4.57: Instalação Ceph

Passo 2: A versão escolhida foi 16.2 (pacific). Existiam outras.

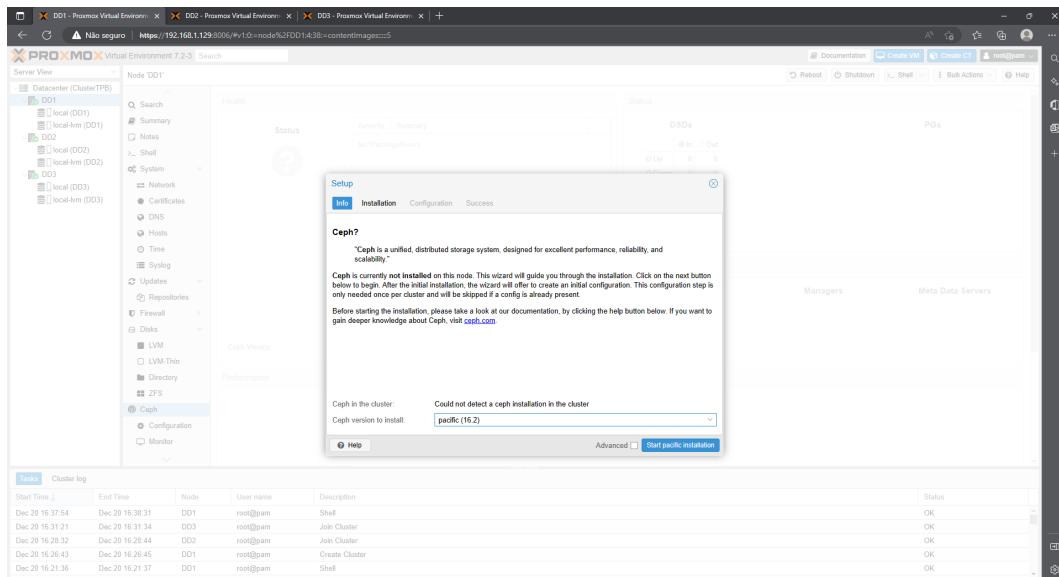


Figura 4.58: Versão do Ceph

Passo 3: Durante o processo de instalação vai ser preciso confirmar a continuação.

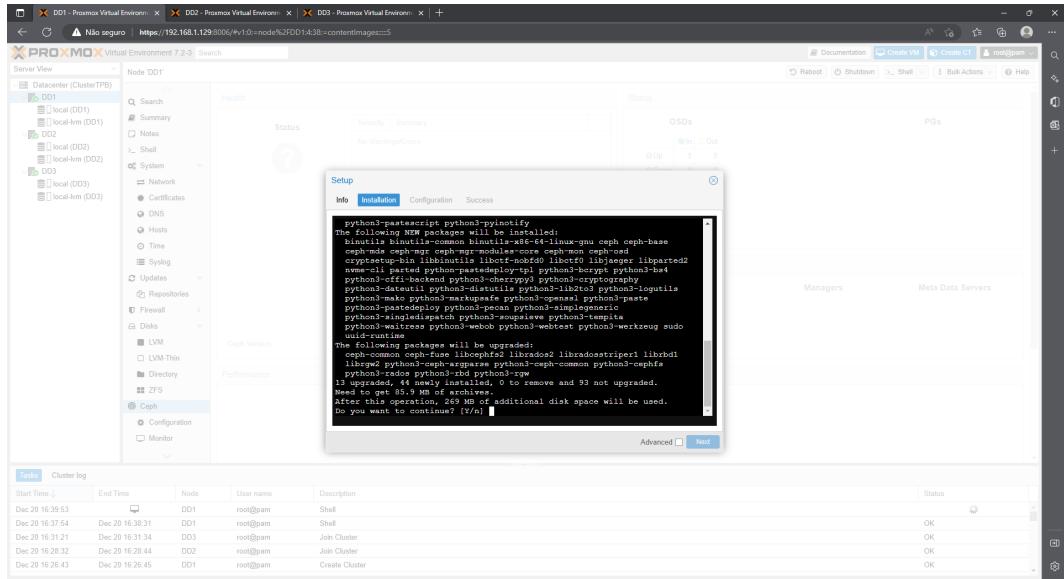


Figura 4.59: Continuação da instalação

Passo 4: Configuração do *Ceph*. Em **Public Network** fica a rede que foi criada para separar o *Cluster* da rede de "Administração": **10.10.10.129/24** e **Cluster Network** a rede da "Administração": **192.168.1.129/24**. Para terminar, no Monitor a escolha é **DD1**.

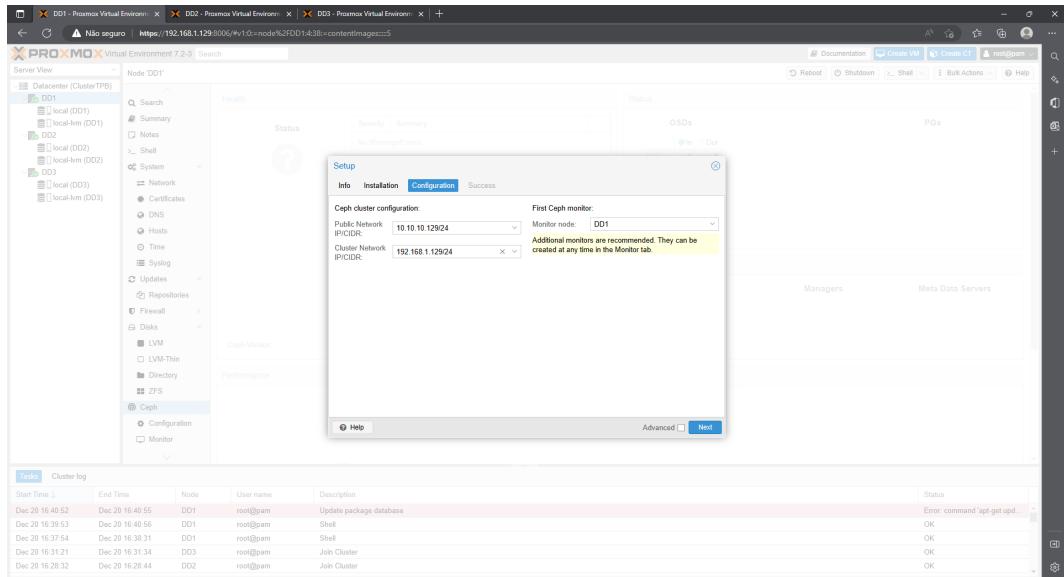


Figura 4.60: Configuração do *Ceph*

Por último, a instalação e configuração fica finalizada.

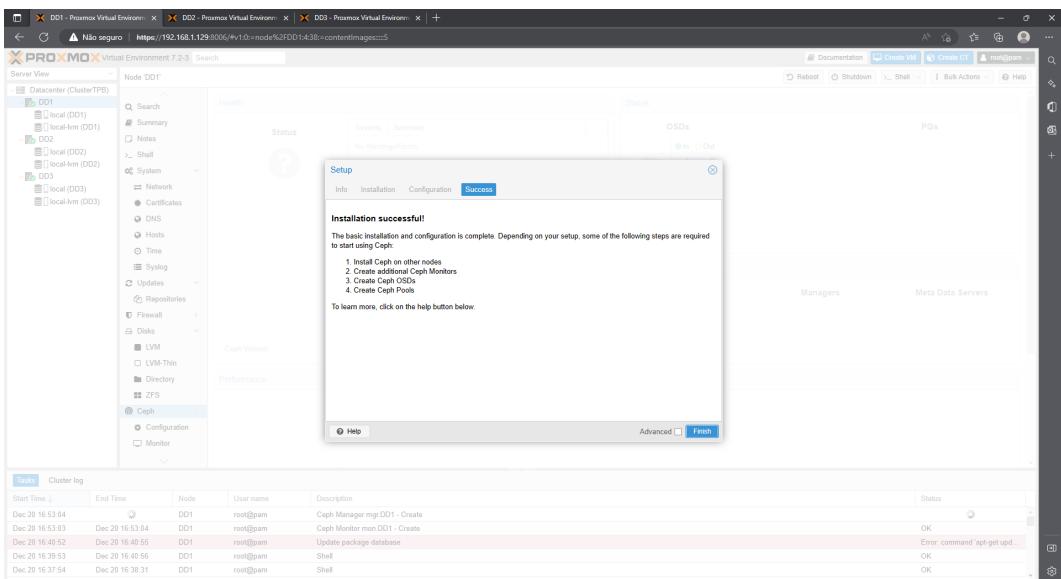


Figura 4.61: Término da instalação

Podemos verificar que no "nó" DD1 foi criado o *Monitor*.

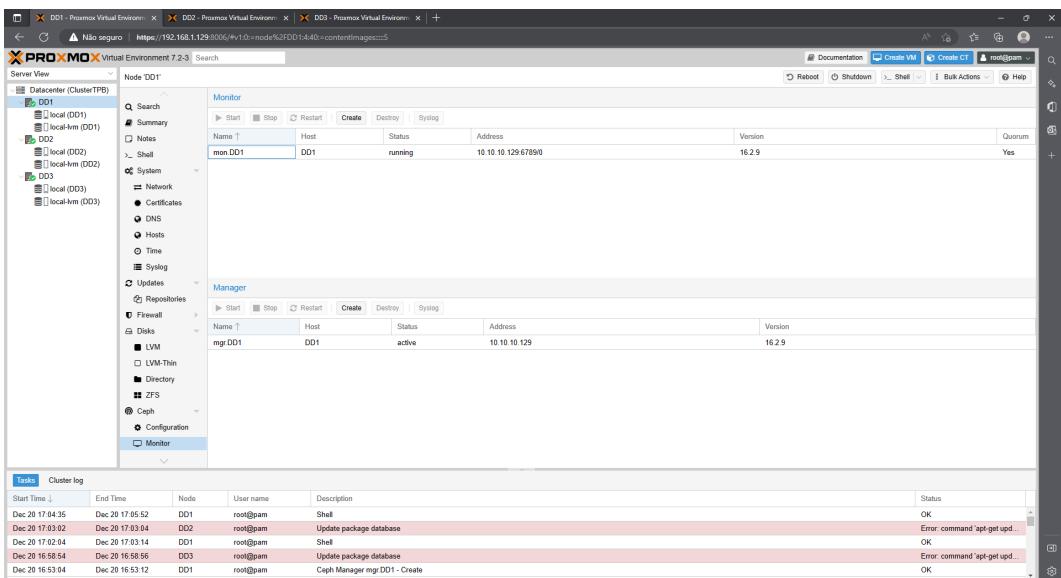


Figura 4.62: Monitor DD1

Nos restantes "nós" temos de repetir os passos anteriores para cada um deles ficar com o *Ceph*. Existe uma diferença na configuração e que não foi preciso configurar porque automaticamente é preenchida.

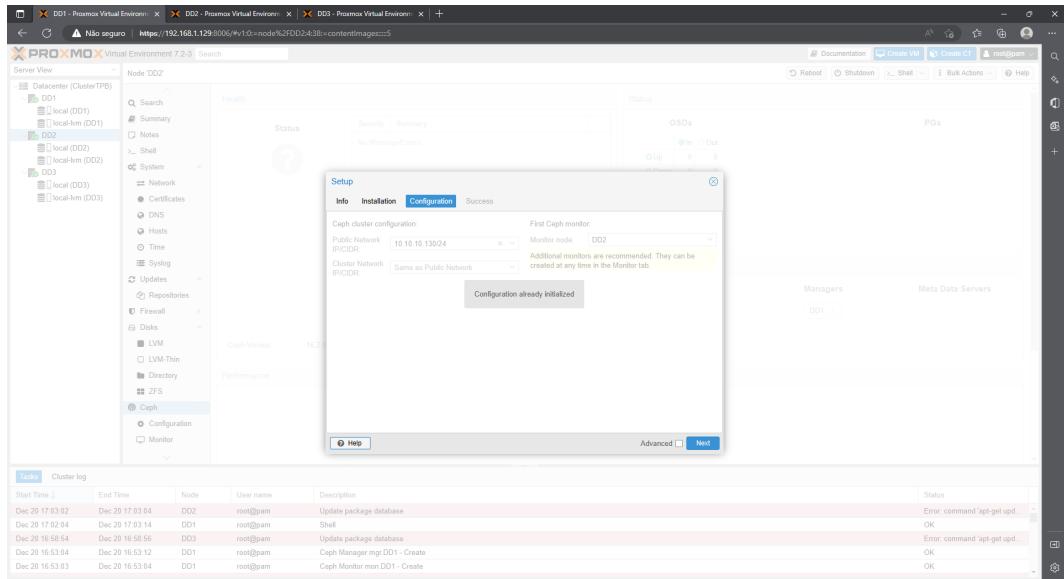


Figura 4.63: Configuração nos restantes nós

No "nó" DD2 podemos verificar que já existem os dois *Monitor*.

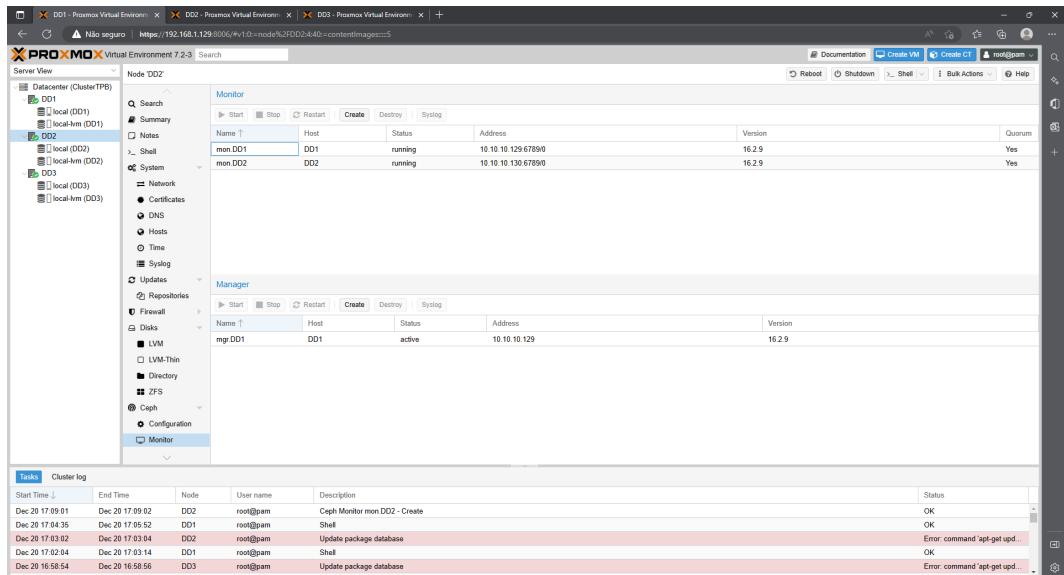


Figura 4.64: Monitor DD2

Caso algum monitor não seja criado automaticamente com a instalação do *Ceph*, é preciso adicionar manualmente. Para isso precisamos de ir a: **Ceph-Monitor-Create**. No meu caso foi o DD3.

The screenshot shows the Proxmox VE interface with three nodes (DD1, DD2, DD3) listed in the sidebar. The 'Monitor' section is selected in the main pane. A 'Create: Monitor' dialog box is open, showing 'Host: DD3'. Below it, a table lists existing monitors: mon DD1 (DD1, running, 10.10.10.129, 16.2.9, Quorum Yes), mon DD2 (DD2, running, 10.10.10.130, 16.2.9, Quorum Yes). A log table at the bottom shows tasks like 'Ceph Monitor mon DD2 - Create' and 'Update package database'.

Figura 4.65: Create Monitor DD3 Manualmente

Por último, podemos veirifcar que todos os "nós" possuem os vários *Monitor*.

The screenshot shows the Proxmox VE interface with three nodes (DD1, DD2, DD3) listed in the sidebar. The 'Monitor' section is selected in the main pane. A table lists monitors: mon DD1 (DD1, running, 10.10.10.129, 16.2.9, Quorum Yes), mon DD2 (DD2, running, 10.10.10.130, 16.2.9, Quorum Yes), mon DD3 (DD3, running, 10.10.10.131, 16.2.9, Quorum Yes). A log table at the bottom shows tasks like 'Ceph Monitor mon DD3 - Create' and 'Ceph Monitor mon DD2 - Create'.

Figura 4.66: Monitor do Cluster

Podemos verificar o estado (*Health*) do *Ceph*. Neste momento vai indicar um *WARNING* porque ainda não existem *Ceph Object Storage Daemons* (OSD) criados.

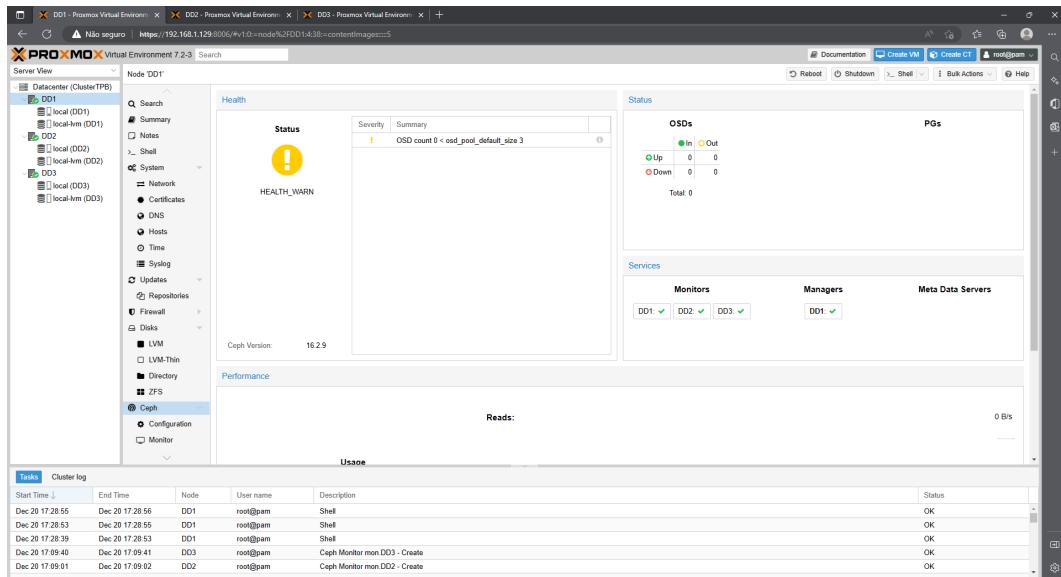


Figura 4.67: Configuração do *Ceph*

4.6.1 OSD

Cada OSD faz a gestão de armazenamento de cada "nó" e para isso foi preciso criar de seguida um novo "disco" para associar a cada OSD. Disco esse que vai ser replicado pelo *Cluster*.

Passo 1: Adicionar mais um *Hard Disk* a cada servidor no *VMWare*.

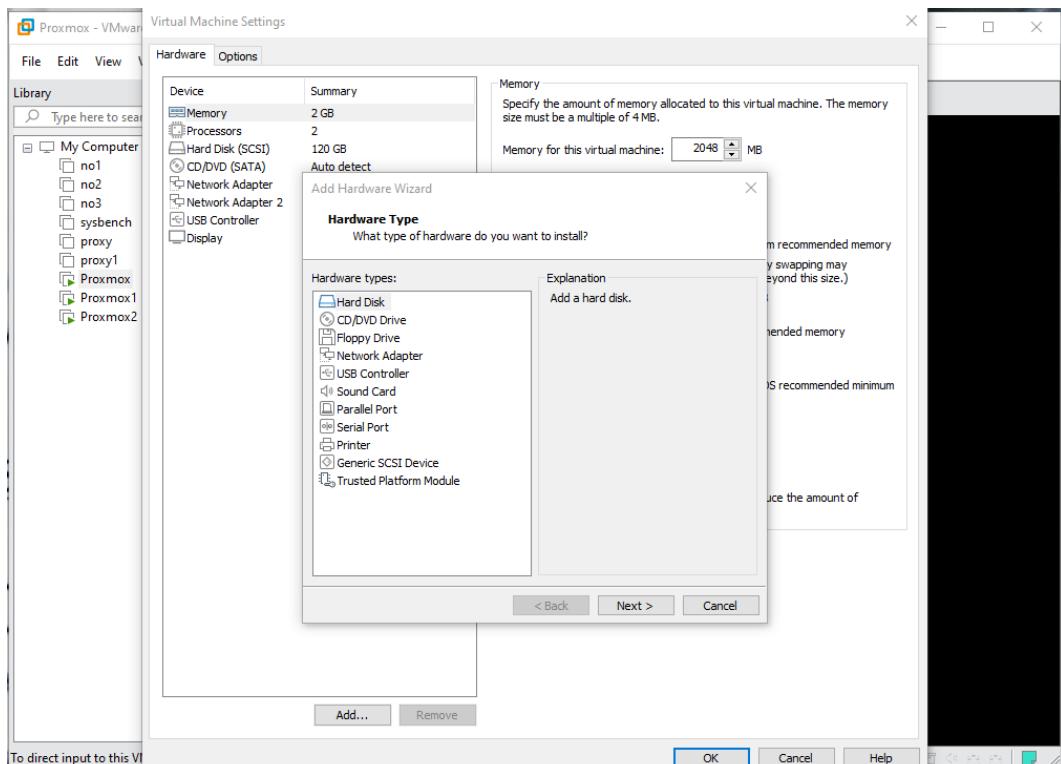


Figura 4.68: Adicionar Hard Disk

Passo 2: Escolha do tipo de Disco: *Small Computer System Interface* (SCSI).

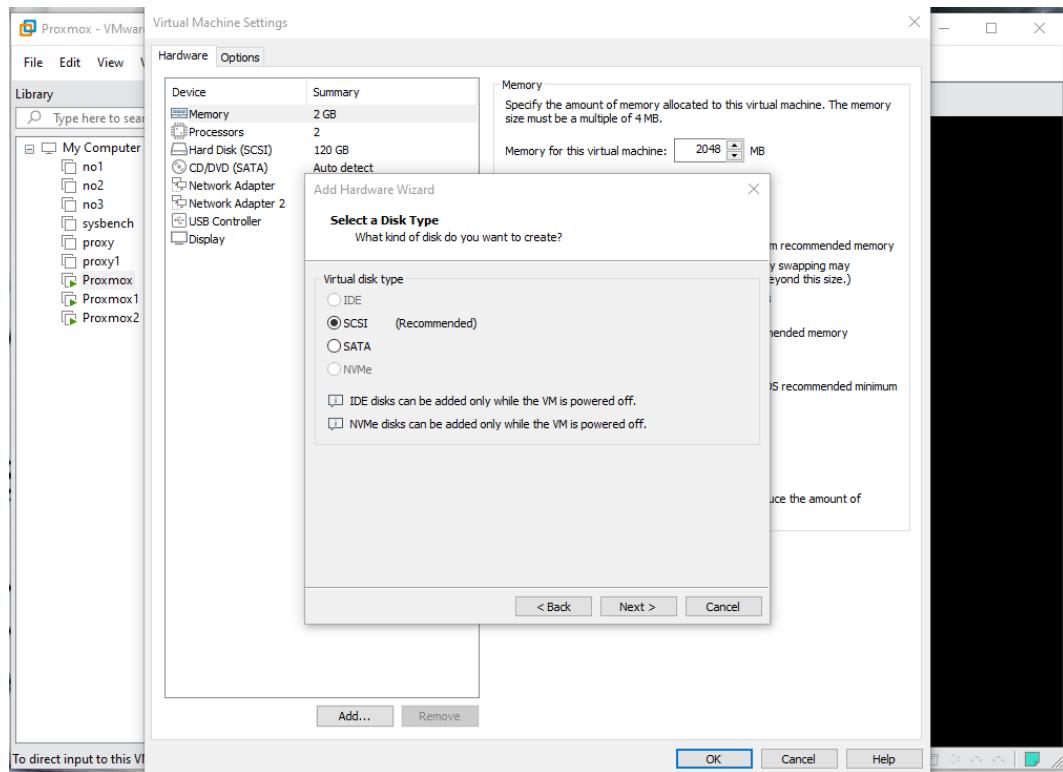


Figura 4.69: Escolha de Tipo de Disco

Passo 3: Por defeito escolhi **Create a new virtual disk**.

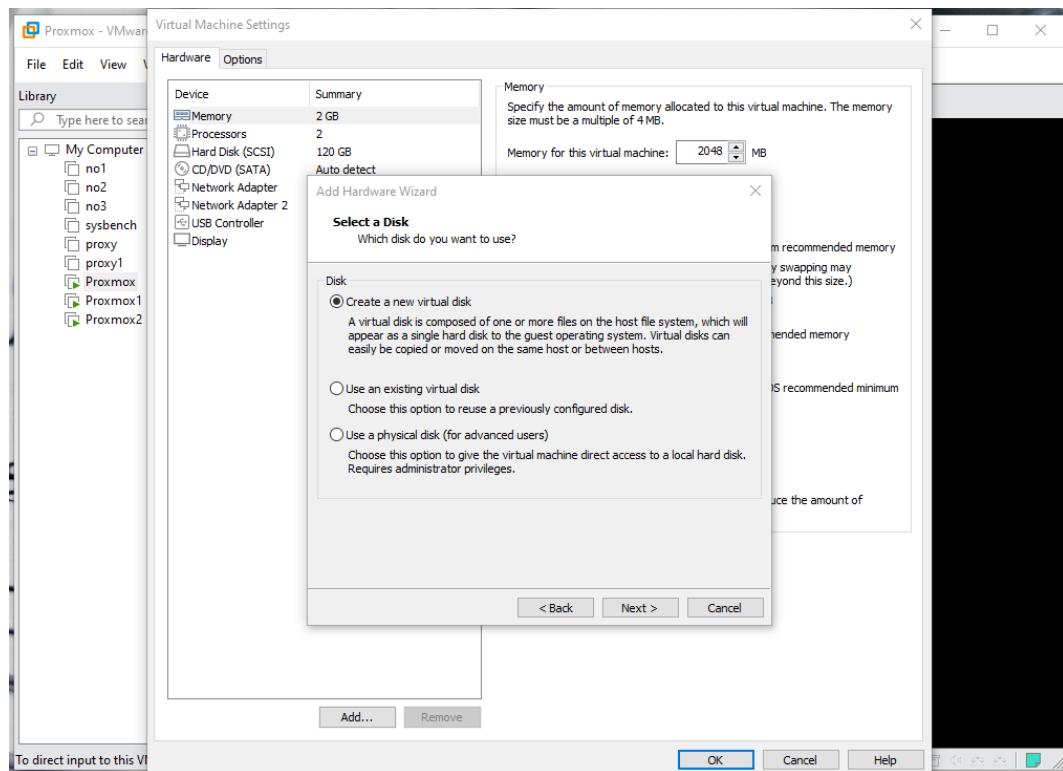


Figura 4.70: Create New virtual Disk

Passo 4: Capacidade do disco: 80Gb.

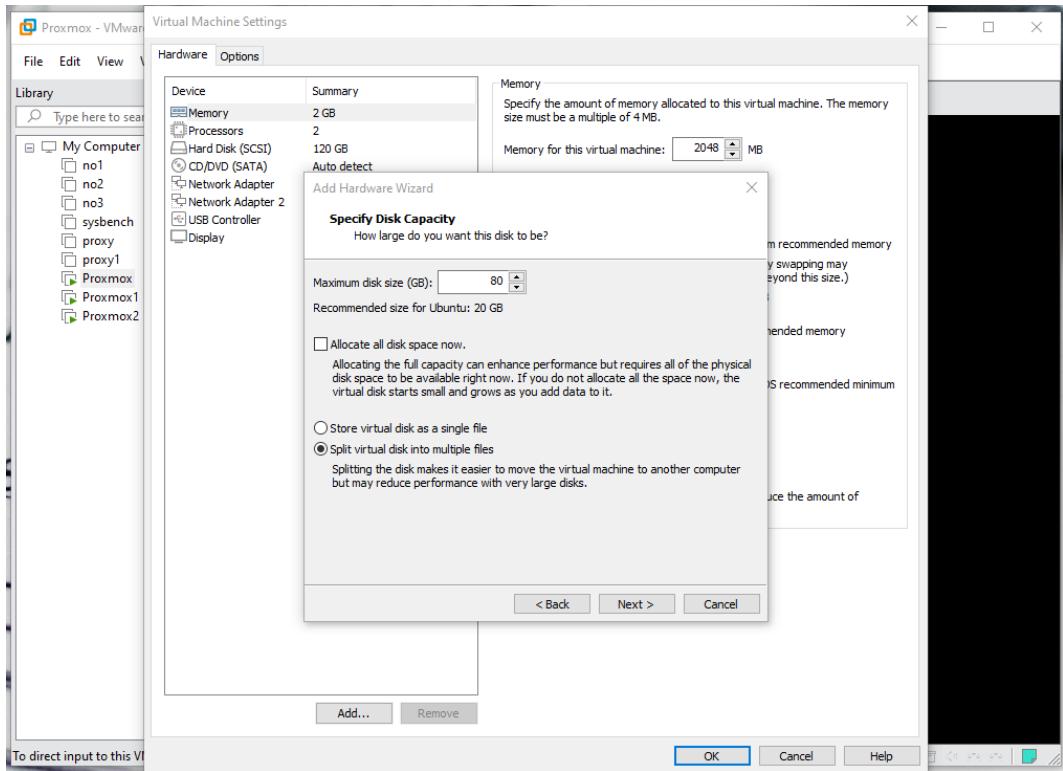


Figura 4.71: Capacidade do disco

Passo 5: Por defeito optei por deixar o nome do ficheiro *vmdk* e finalizar.

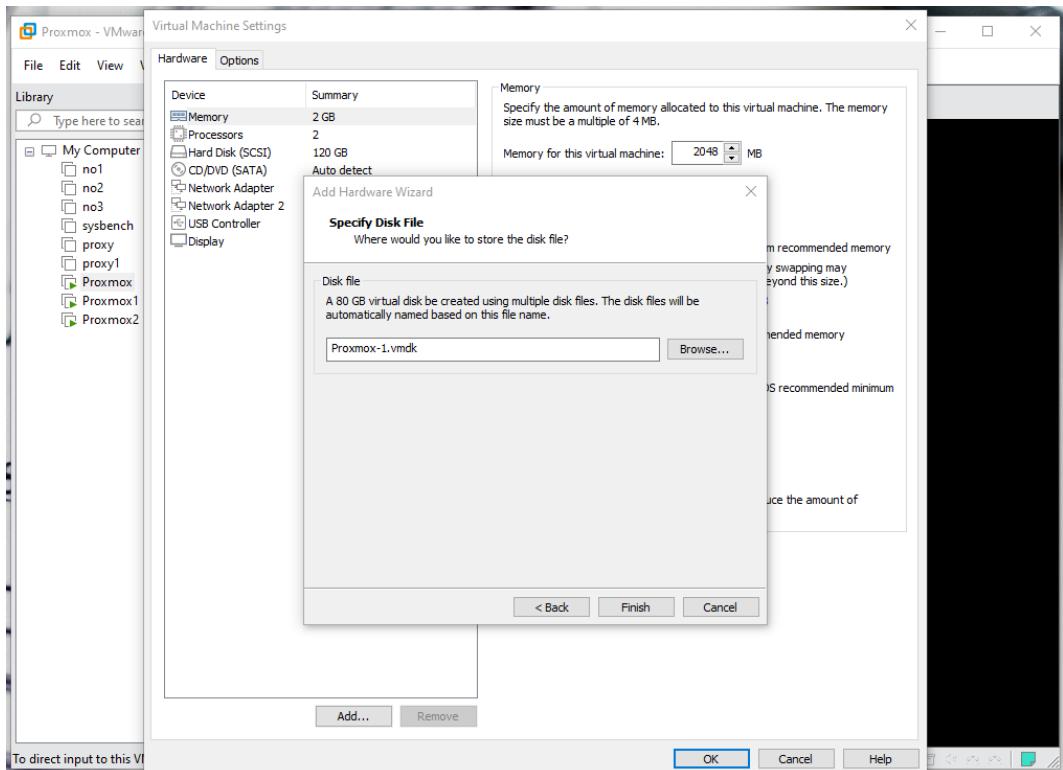


Figura 4.72: Término da criação do novo disco

Nos restantes servidores é só repetir os passos anteriores para adicionar um novo disco ao servidor.

Passo 6: Depois de adicionar um novo disco aos servidores, criei então o OSD com o devido disco.

Figura 4.73: Create OSD

Figura 4.74: Escolha do disco

Em DD1 e DD2 já podemos verificar o OSD criado com o disco de 80Gb agregado.

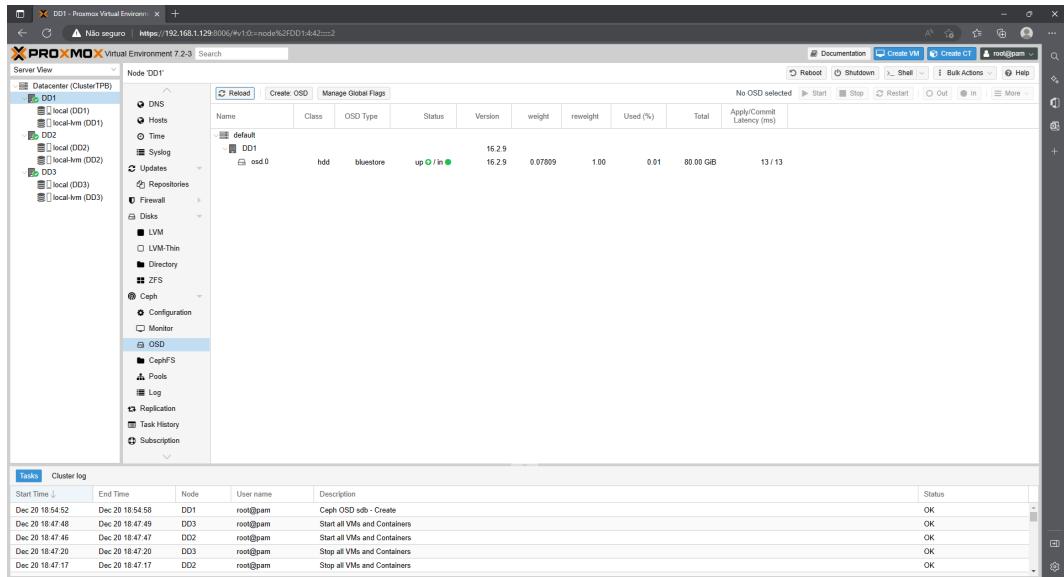


Figura 4.75: OSD DD1

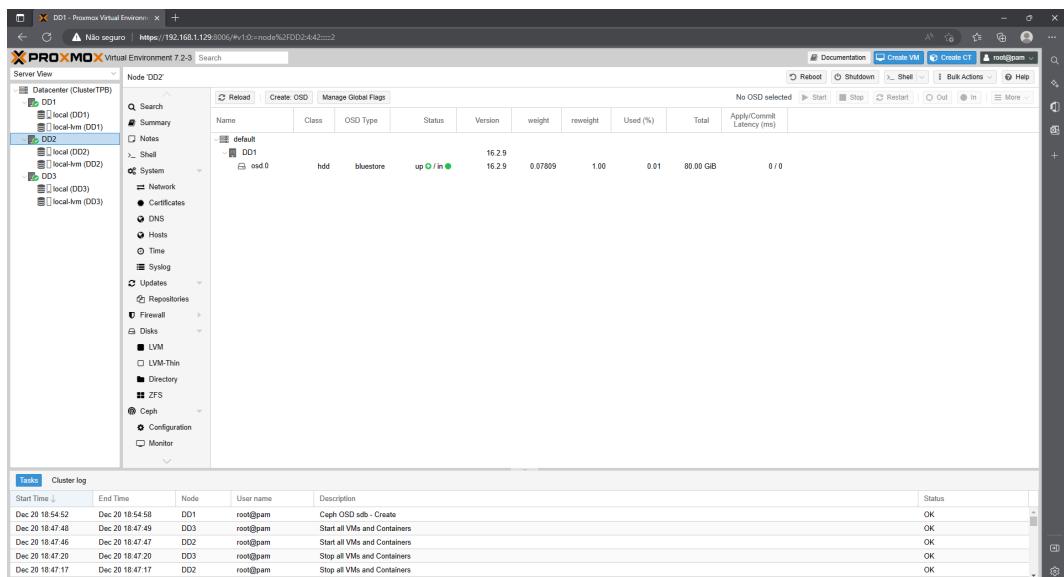


Figura 4.76: OSD DD2

Para obter todos os OSD no *Cluster* apenas foi preciso repetir o passo 6. No final podemos observar que em DD1 e nos restantes "nós" existem os três OSD.

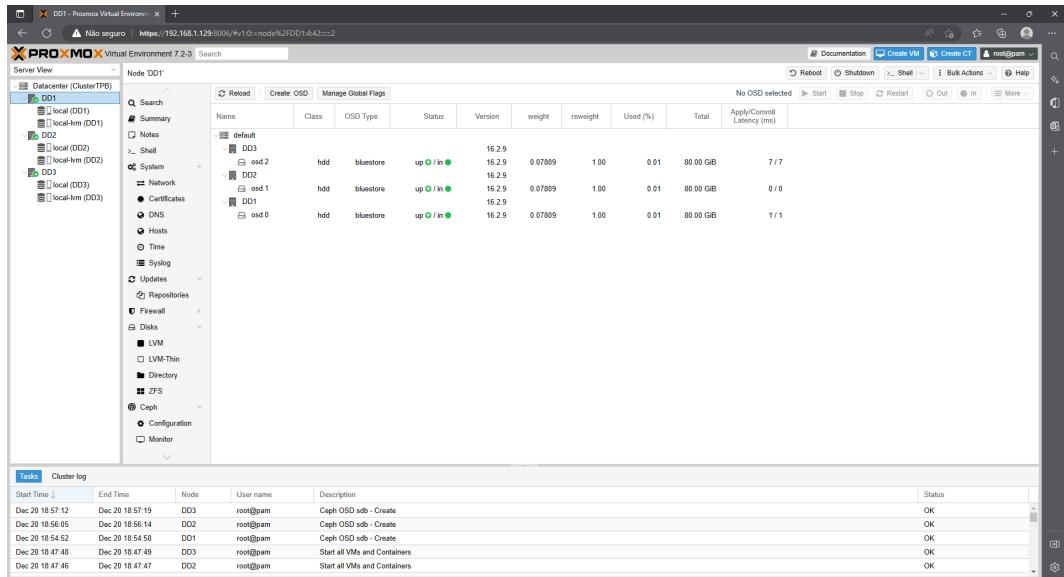


Figura 4.77: OSD Cluster

Depois de ter adicionado um novo disco aos servidores e ter criado os OSD podemos confirmar que o *Ceph* está com boa "saúde".

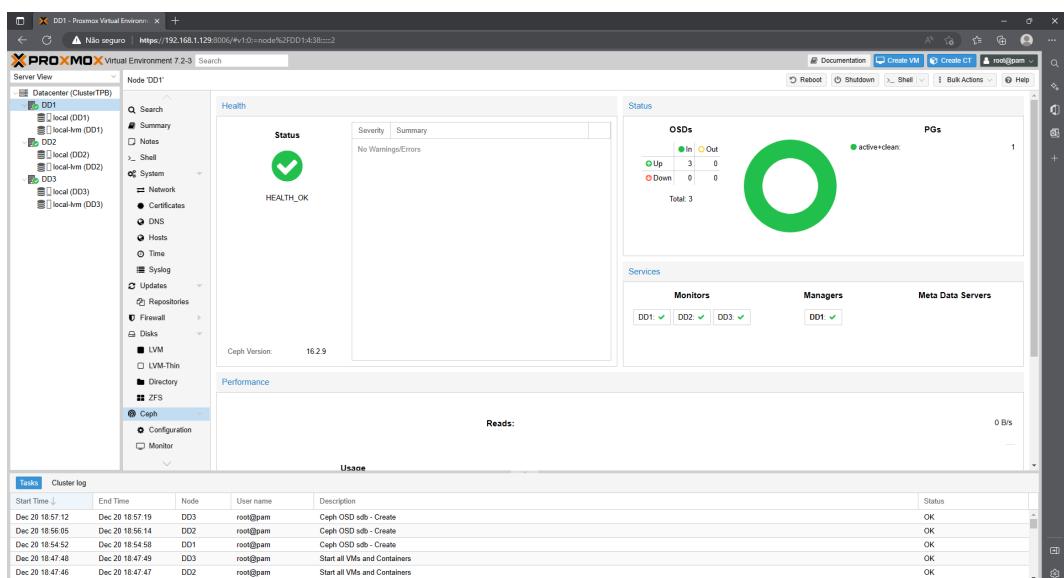


Figura 4.78: Health Ceph

4.6.2 Pools

Pools são partições lógicas usadas para armazenar objetos. Objetos esses que ficam armazenados no núcleo do *Ceph*, *Reliable Autonomic Distributed Object Store* (RADOS). Para criar a *pool* de armazenamento apenas é preciso criar uma vez. Neste caso criei em DD1.

Passo 1: Em **DD1-Ceph-Pools-Create**, criei a *pool* indicando apenas um nome.

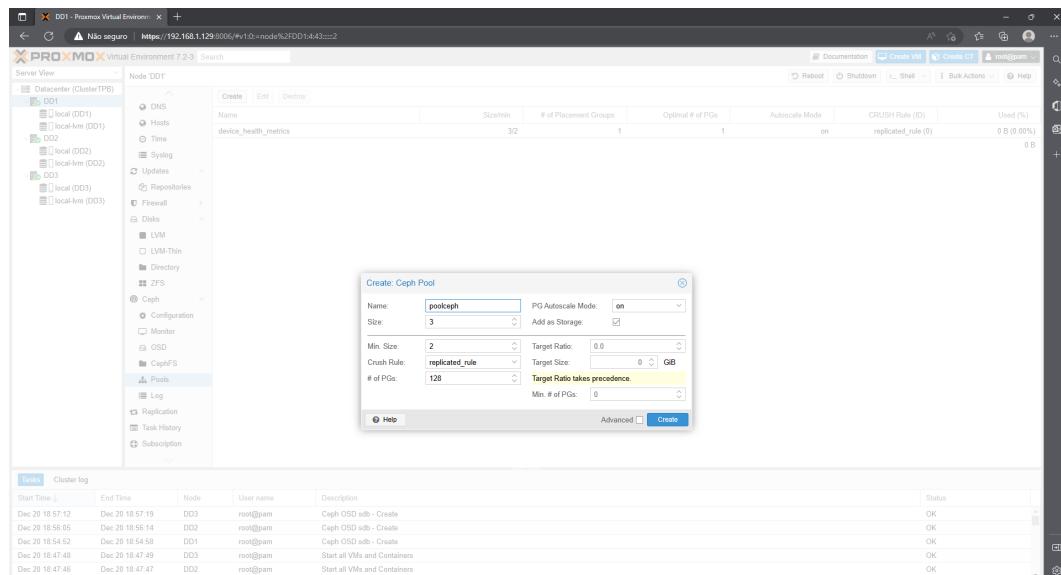


Figura 4.79: Pool Create

Como no *Cluster* existem menos de 5 OSD, os *Placement Group* (PG) ficam definidos com o número de 128.

Depois de ter criado a *pool* de armazenamento podemos verificar que foi criada e replicada no *cluster*.

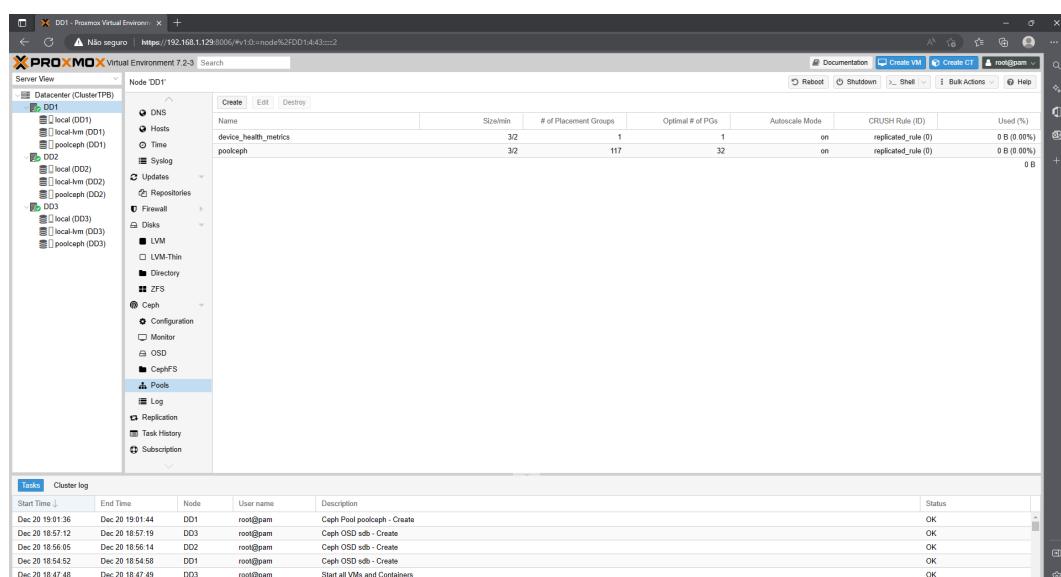


Figura 4.80: Pool Create

De seguida, ao verificar a *Health* do *Ceph* podemos observar os PG a 33.

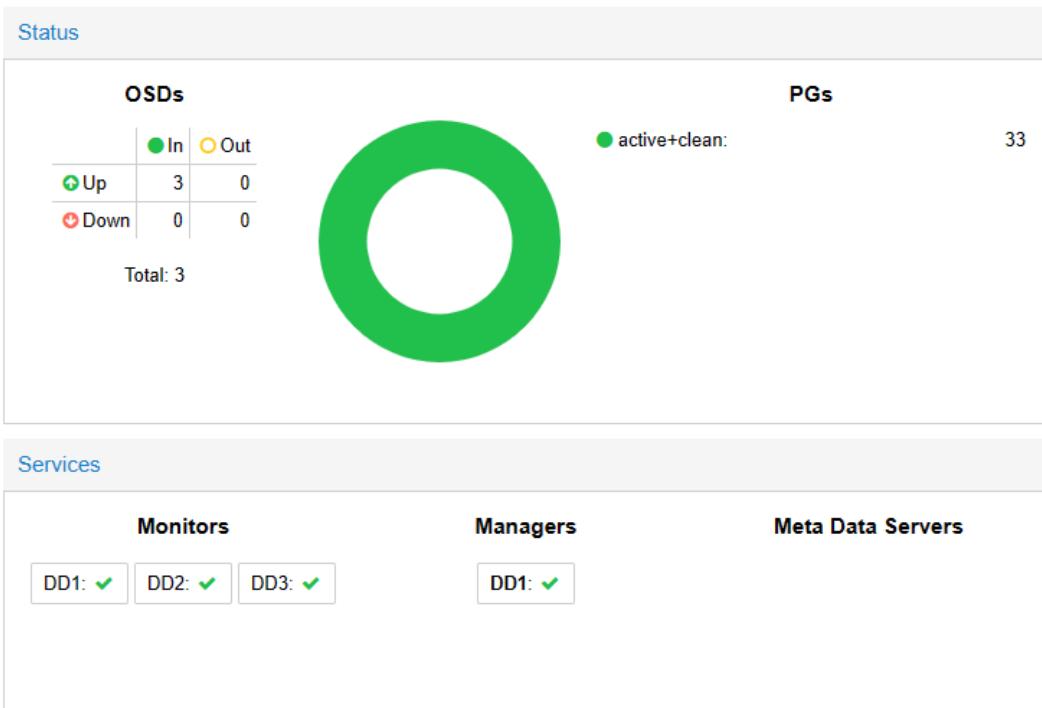


Figura 4.81: Health Ceph

4.7 Criação Máquina Virtual

Para testar a migração precisamos de criar uma máquina virtual/container. De seguida segue o registo do passo a passo da criação da máquina virtual.

Passo 1: *Upload* do ficheiro *ISO*. Neste caso foi usado o Ubuntu Server 22.04.

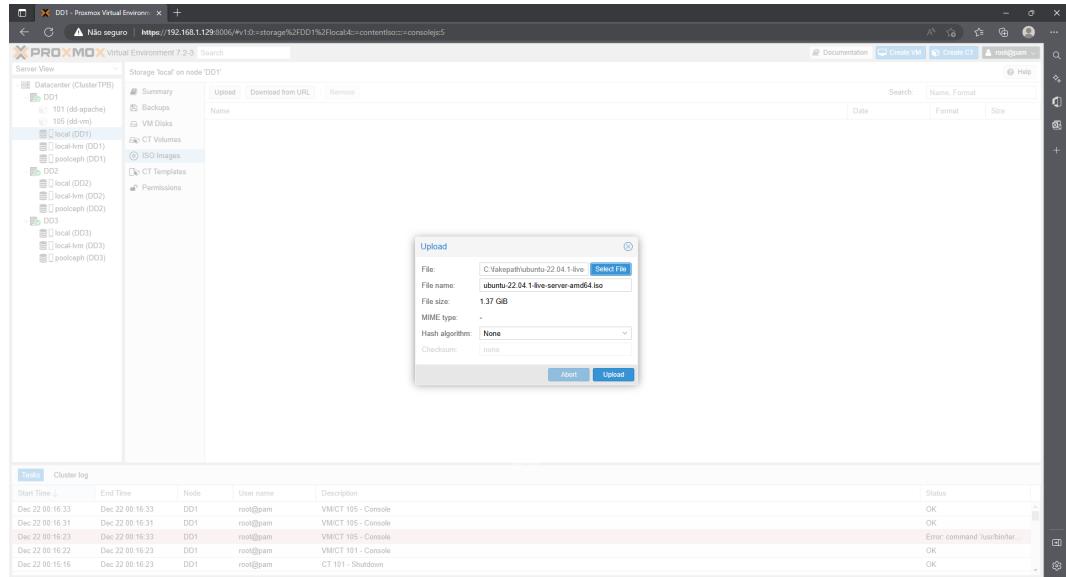


Figura 4.82: Upload ISO

Depois da importação finalizada é esperado o "TASK OK".

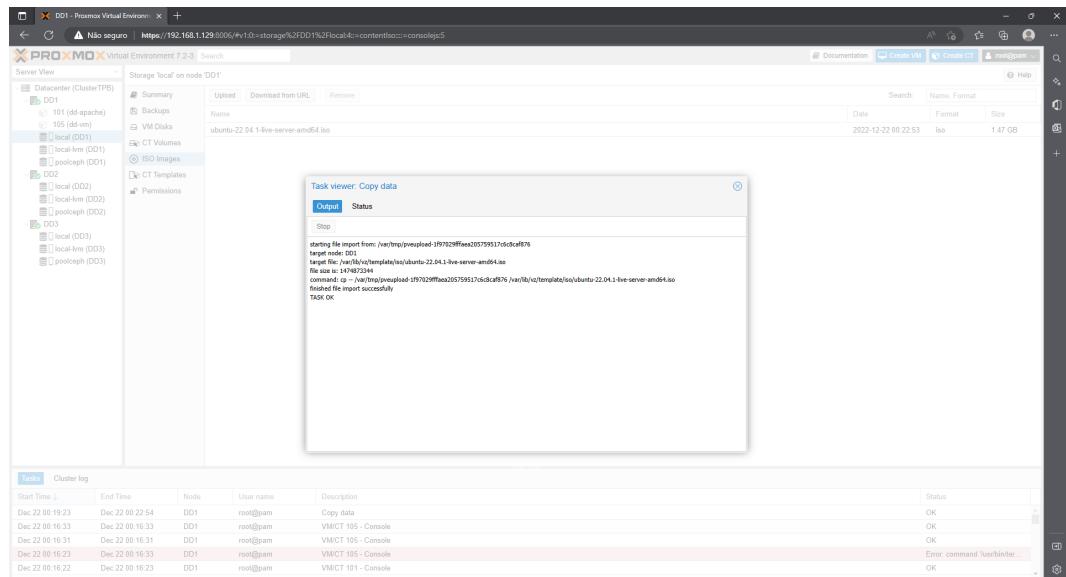


Figura 4.83: Task Ok Upload

Passo 2: Depois de ter o ficheiro *ISO* no servidor podemos criar a máquina virtual. **DD1-Create VM.** Em General:

Node: "nó" onde colocar a VM;

VM ID: um ID para a VM;

Name: Nome da VM.

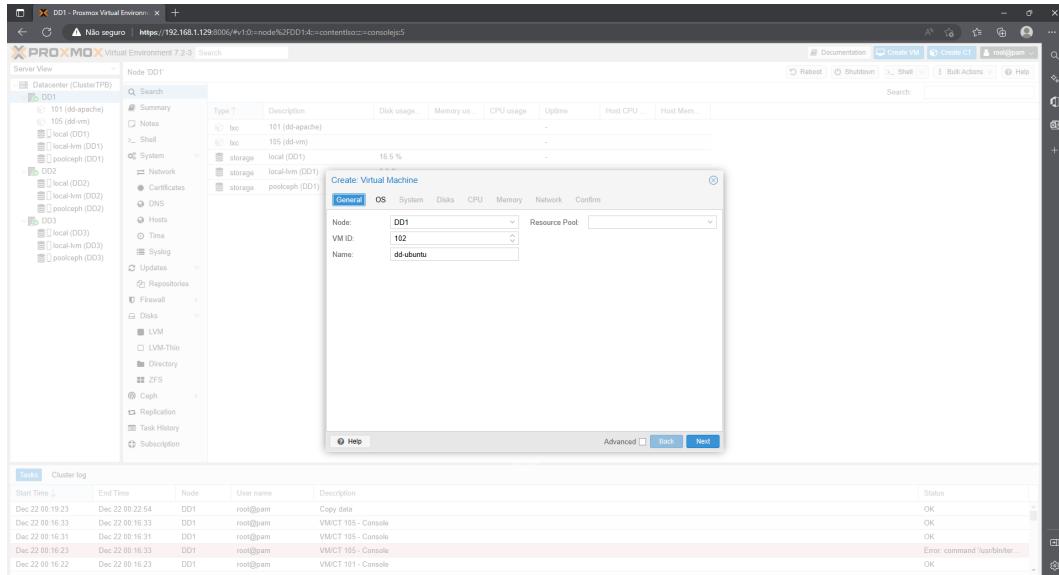


Figura 4.84: Create Virtual Machine-General

Passo 3: Em OS:

Storage: local;

ISO image: escolher o ficheiro *ISO* que foi importado para o servidor.

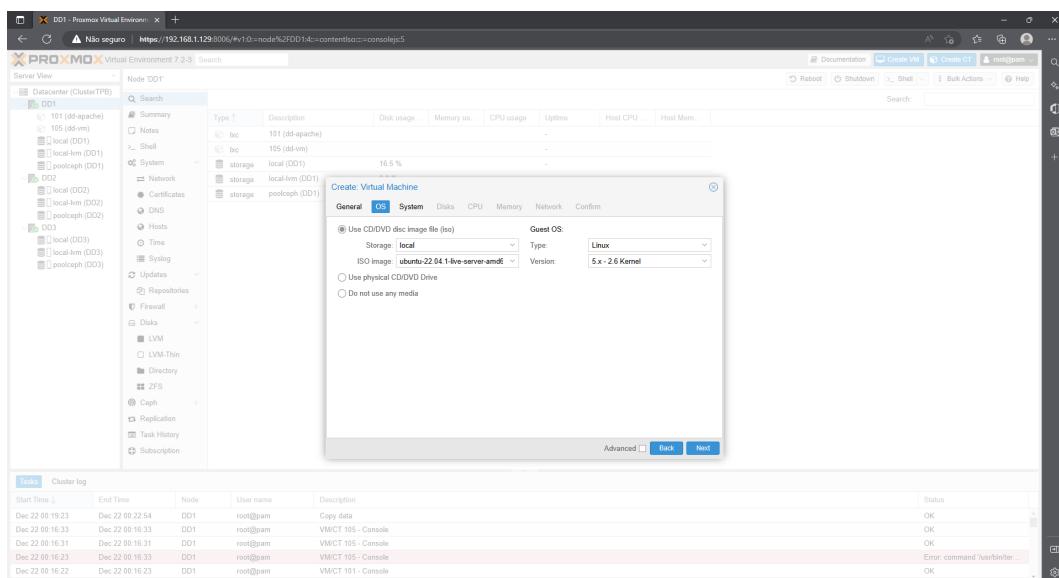


Figura 4.85: Create Virtual Machine-OS

Passo 4: Em Disks:
Storage: poolceph, que foi a *pool* criada anteriormente.

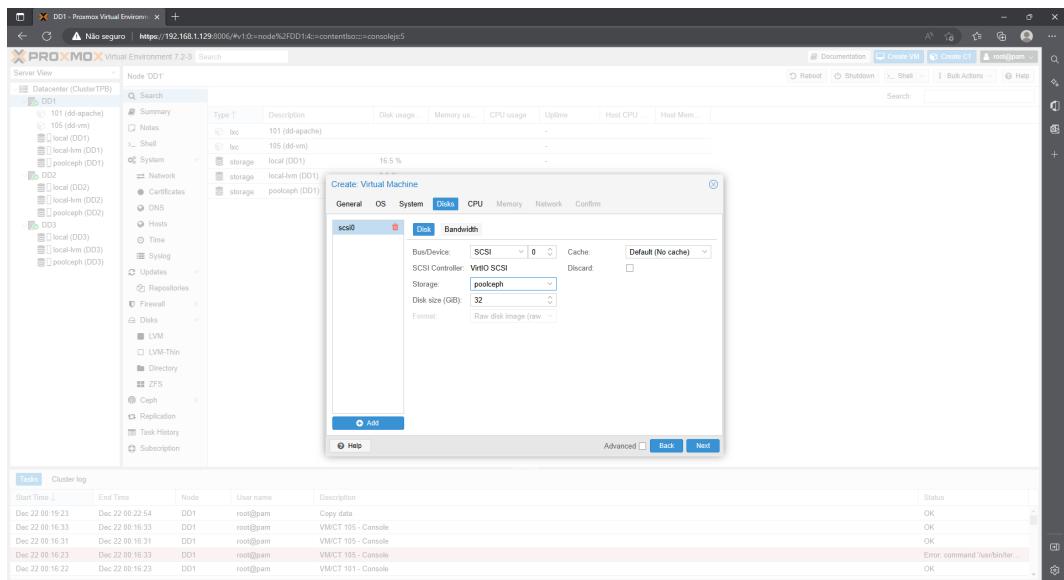


Figura 4.86: Create Virtual Machine-Disks

Passo 5: Em CPU:
Cores: 2.

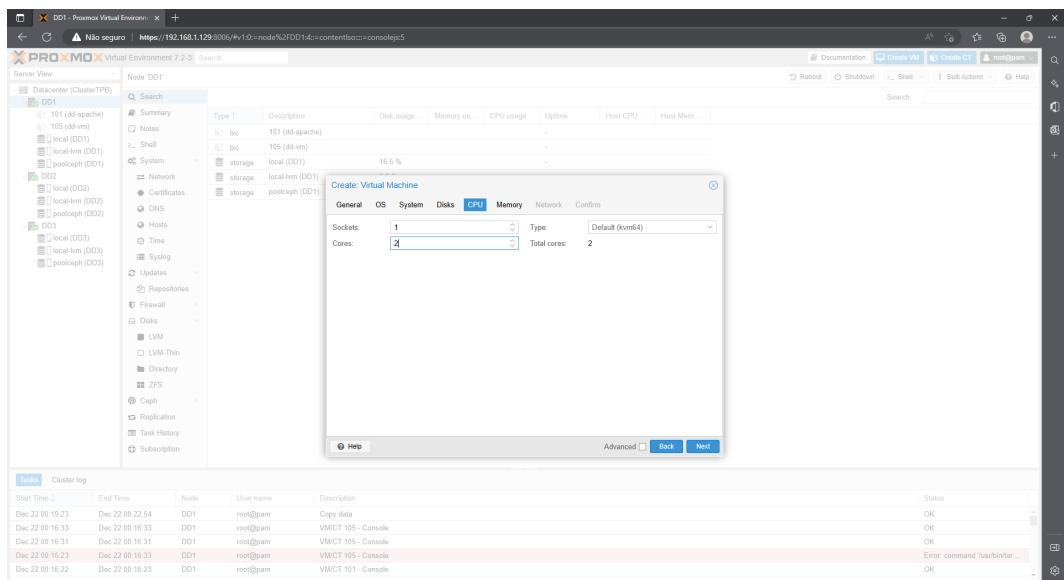


Figura 4.87: Create Virtual Machine-CPU

Passo 6: Em Memory:
Memory (MiB): 512.

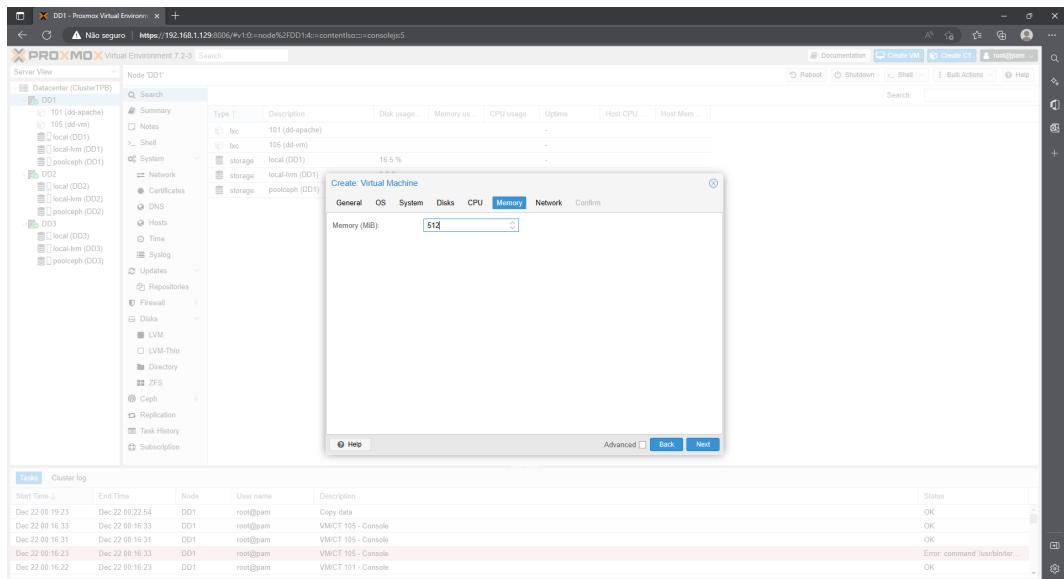


Figura 4.88: Create Virtual Machine-Memory

Passo 6: Em Network:
Bridge: vmbr0.

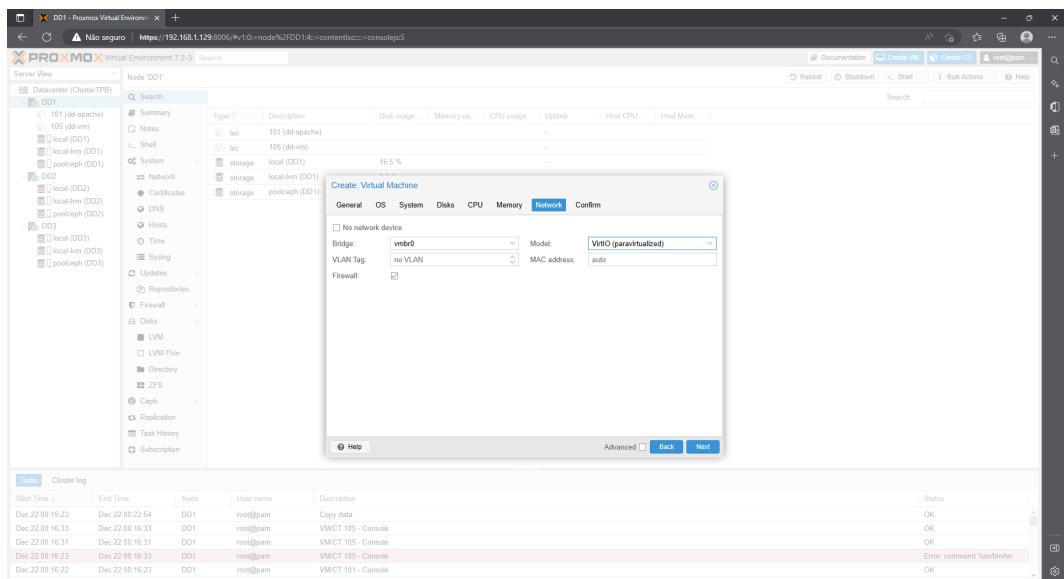


Figura 4.89: Create Virtual Machine-Network

Terminada a configuração da criação da máquina virtual, esta está logo pronta no "nó" para ser iniciada. Na consola da máquina virtual temos acesso ao SO.

O IP atribuído à máquina virtual foi: **192.168.1.128**

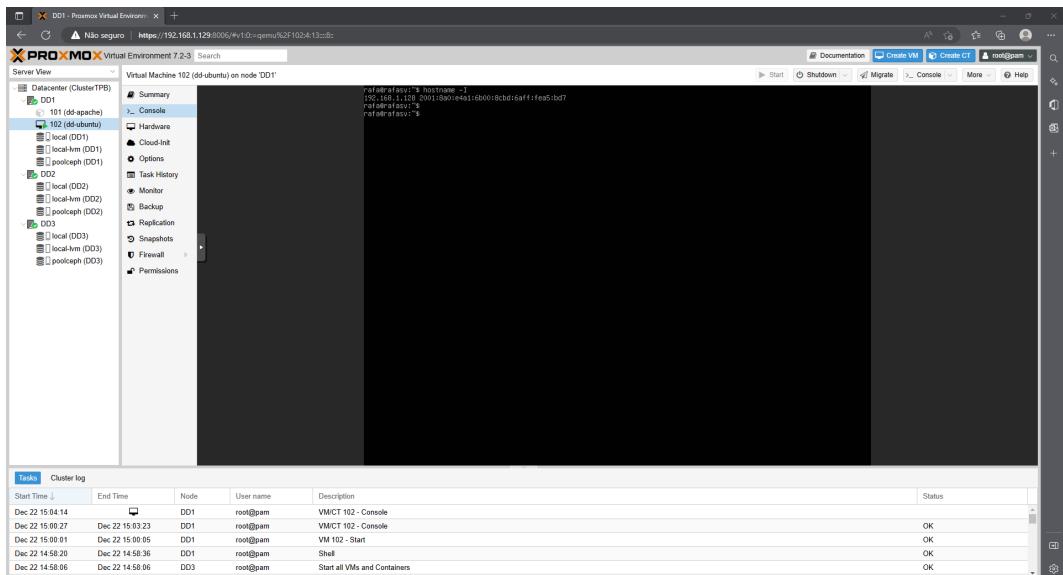


Figura 4.90: Ifconfig Ubuntu Server

Existe conectividade com a máquina virtual:

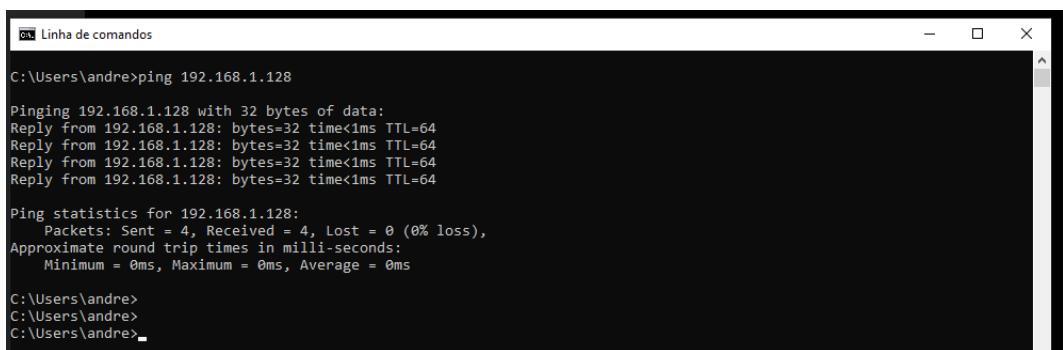


Figura 4.91: Conectividade com a VM

4.8 Experiência 1 - Migration Ceph

O Cluster ainda não está totalmente configurado mas consegue-se fazer uma experiência simples com a migração da VM já criada recorrendo à opção "*Migrate*" existente no Proxmox para a VM.

Para esta experiência fiz um *ping* contínuo do *Personal Computer* (PC) host para a máquina virtual:

Figura 4.92: Ping VM

Voltando a aceder à consola do Ubuntu Server no proxmox e recorrendo a opção **Migrate**, efetuei a migração da VM de DD1 para DD2:

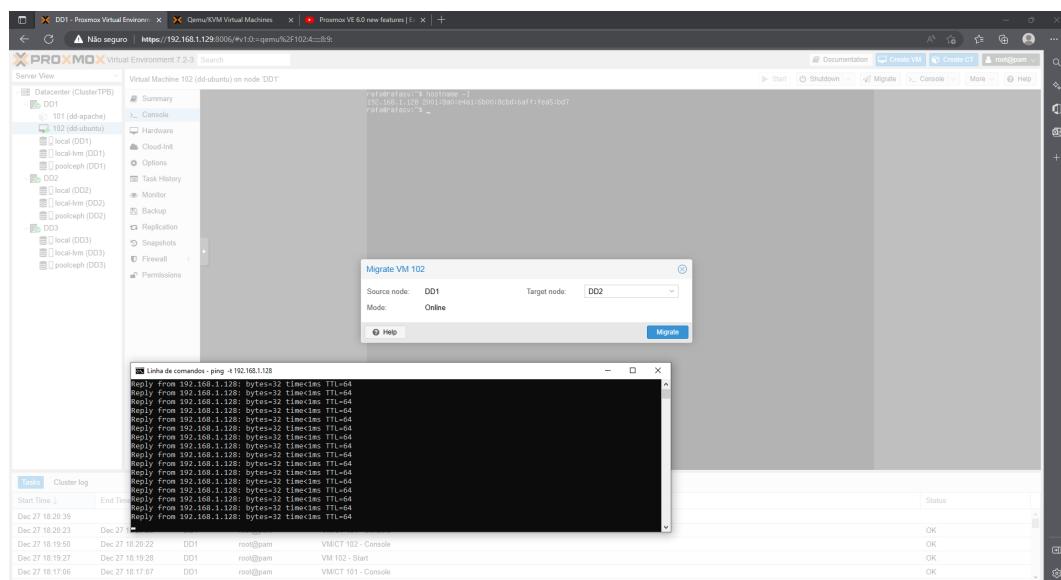


Figura 4.93: Migrate VM

Pode-se observar na *Task View* o processo de migração a decorrer e ao mesmo tempo o *Ping* a continuar em execução:

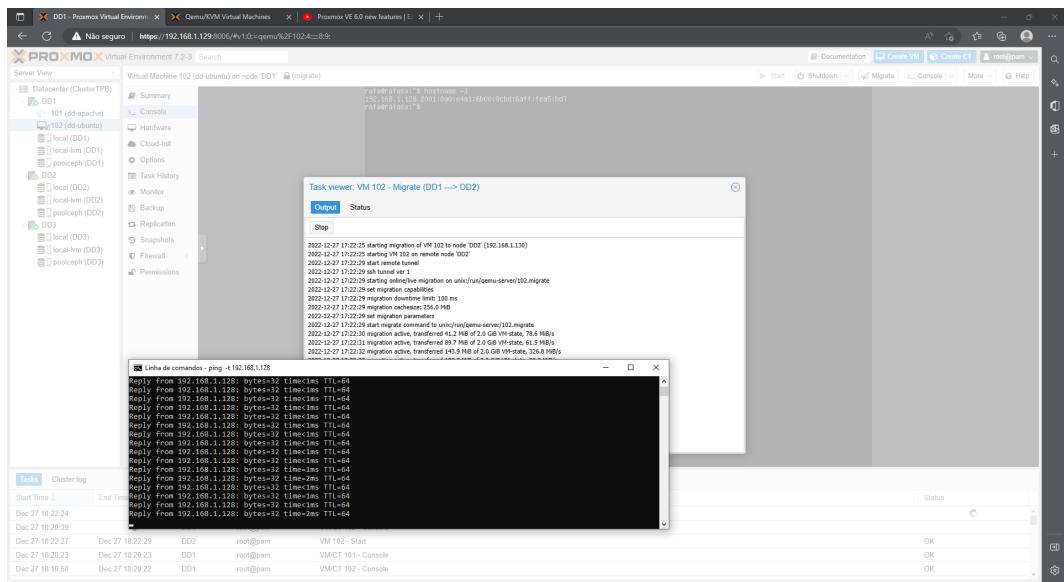


Figura 4.94: Migração em curso

Para terminar, pode-se observar que não houve, em momento algum, nenhuma interrupção no *Ping* e a VM foi migrada para DD2 com sucesso.

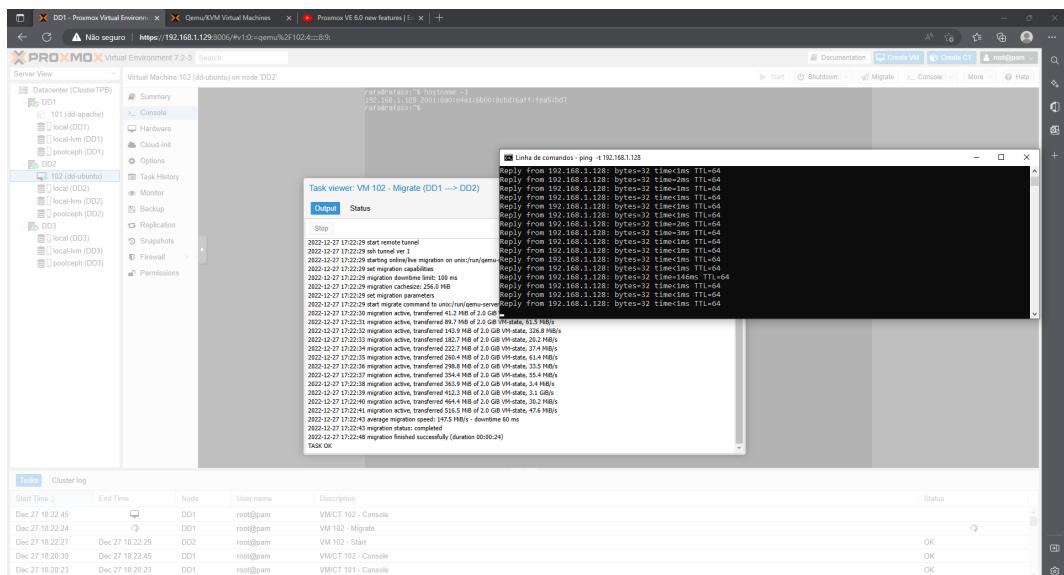


Figura 4.95: Ping

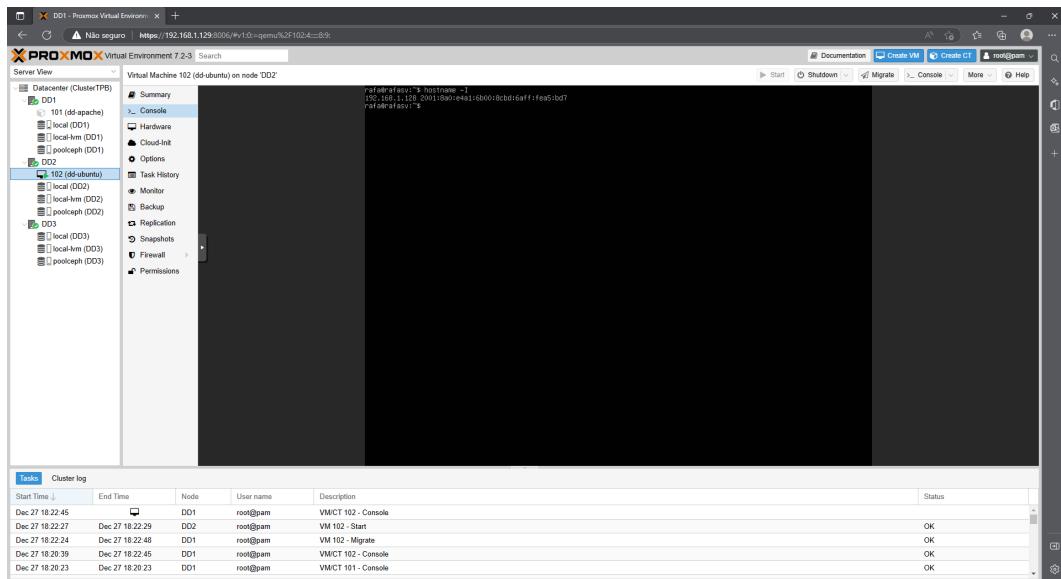


Figura 4.96: Migração concluída

4.9 HA

HA refere-se às características que são concebidas para assegurar que os dados permanecem acessíveis mesmo em caso de falha de um ou mais nós. Isto é conseguido através da utilização de múltiplas réplicas de dados, para que se uma réplica ficar indisponível, as outras ainda possam ser acedidas. Para evitar o *failover* automático o HA assegura que as VM são restaurados noutros nós.

Desta forma, e primeiro que tudo, é preciso criar um grupo HA onde especificamos quais os nós pertencentes e a sua prioridade. Neste caso chamei ao grupo HA: **groupHA** e a prioridade: **DD1:5, DD2:15 e DD3:10**.

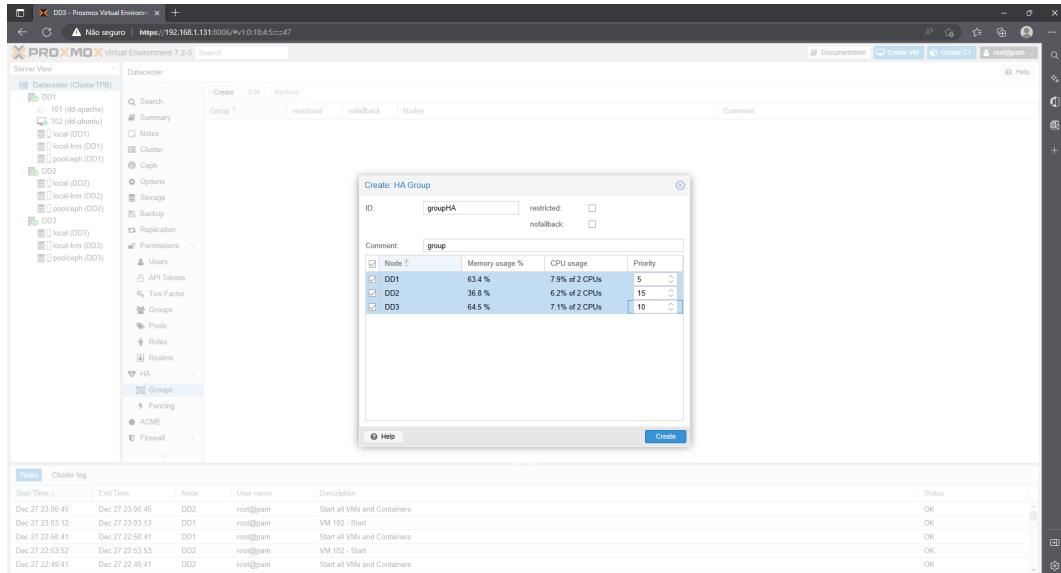


Figura 4.97: Group HA

Próximo passo é associar qual ou quais as máquinas (VM) pertencentes ao HA. Além da VM que tinha criado juntei um *container* Ubuntu Server 22.10. O HA é configurado em **Datacenter-HA-ADD**.

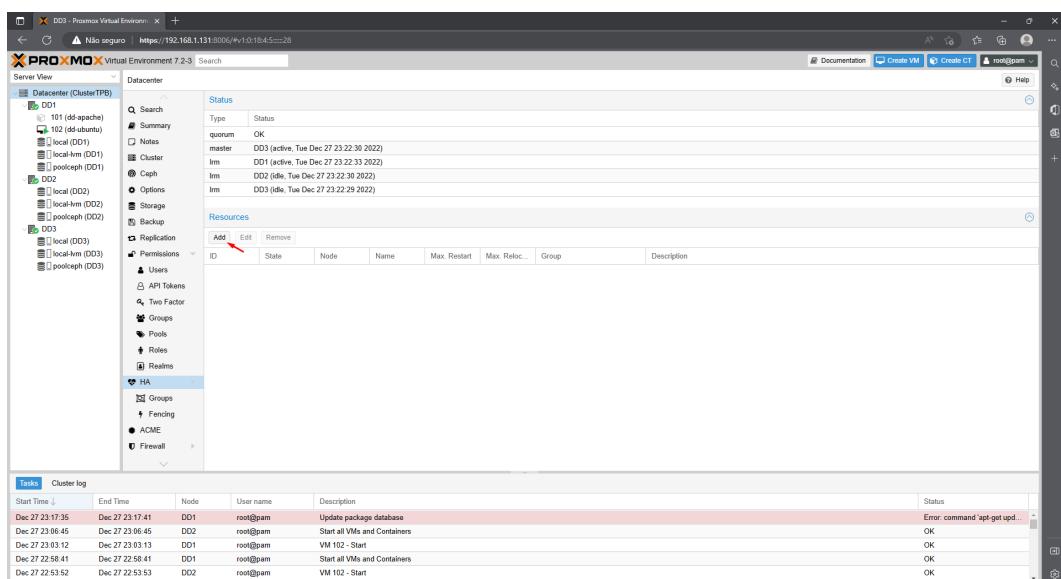


Figura 4.98: Add HA

Adicionei a VM 102 e o *container* 101 ao **groupHA**. Tanto a VM como o *container* estavam em DD1 e assim que foram adicionadas ao **groupHA**, migraram automaticamente para DD2 devido à sua prioridade ser a mais elevada.

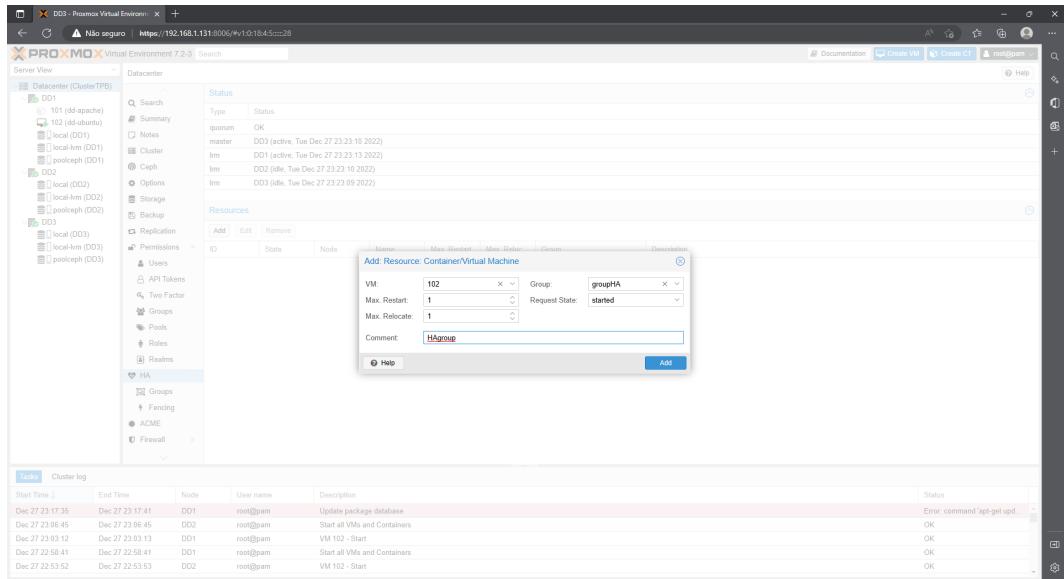


Figura 4.99: Resources HA

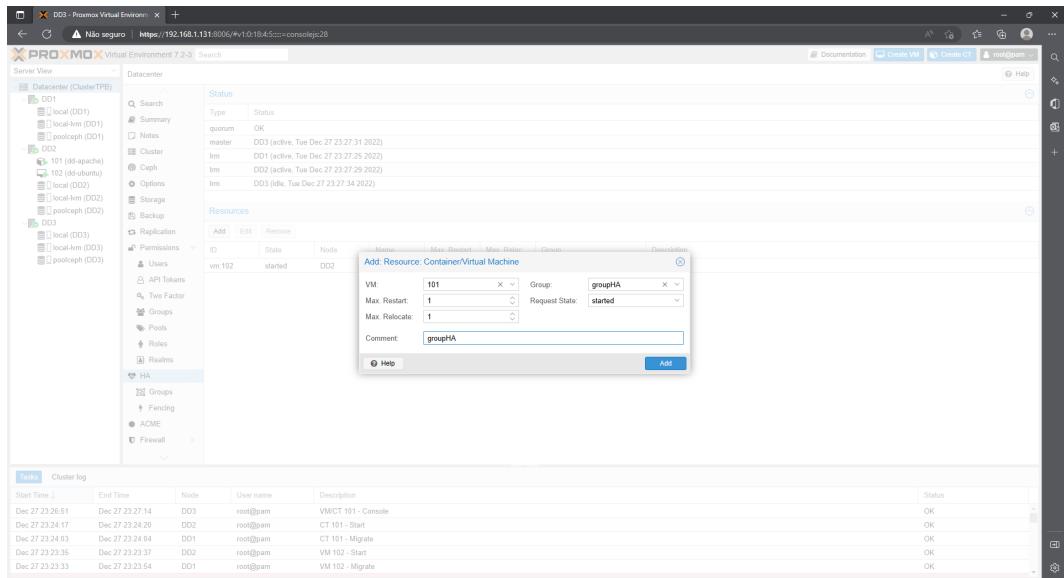


Figura 4.100: Resources HA

Por fim podemos verificar o Grupo HA com a VM e o *container*.

ID	State	Node	Name	Max. Restart	Max. Reloc...	Group	Description
vm102	started	DD2	dd-ubuntu	1	1	groupHA	HGroup
ct101	started	DD2	dd-apache	1	1	groupHA	groupHA

Figura 4.101: Resources HA

4.10 Experiência 2 - High Availability/Automatic Failover

Para tornar este projeto mais interessante e antes de passar para a experiência 2, implementei um Website muito simples e uma ferramenta de monitorização (Uptime Kuma) onde adicionei à Dashboard os vários servidores.

Utilizei um servidor HTTP Apache para o Website (*HyperText Markup Language* (HTML) e *Cascading Style Sheet* (CSS)) e *docker* para o Uptime Kuma. O projeto Uptime Kuma pode ser consultado em Uptime Kuma

A implementação destas duas funcionalidades foram feitas no *container* com o IP 192.168.1.140.

Website da experiência:

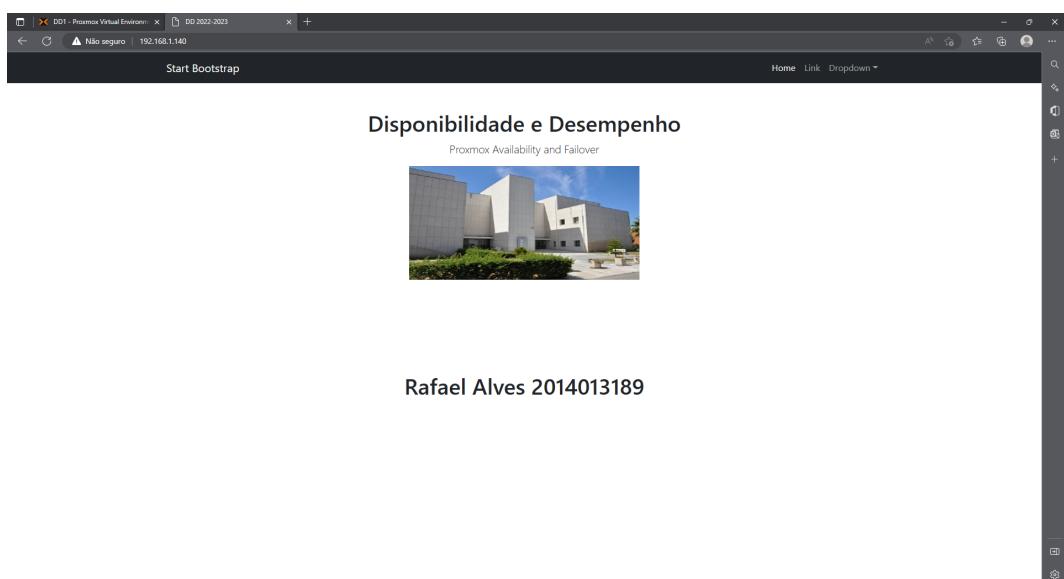


Figura 4.102: Website

Uptime Kuma:

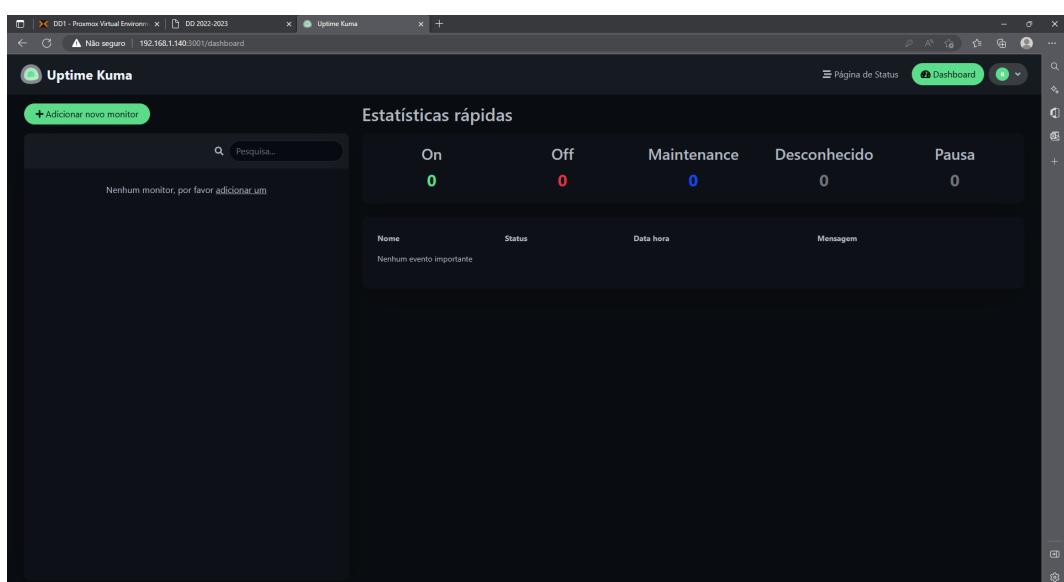


Figura 4.103: Dashboard Uptime Kuma

Com esta experiência quis mostrar não só a disponibilidade entre o *Cluster* mas também a disponibilidade dos serviços contidos nas VM/*container*.

Para dar início à experiência 2 a VM e o *container* encontravam-se em DD2 (prioridade 15) e do PC Host fiz um *ping* contínuo para o *container* 192.168.1.140.

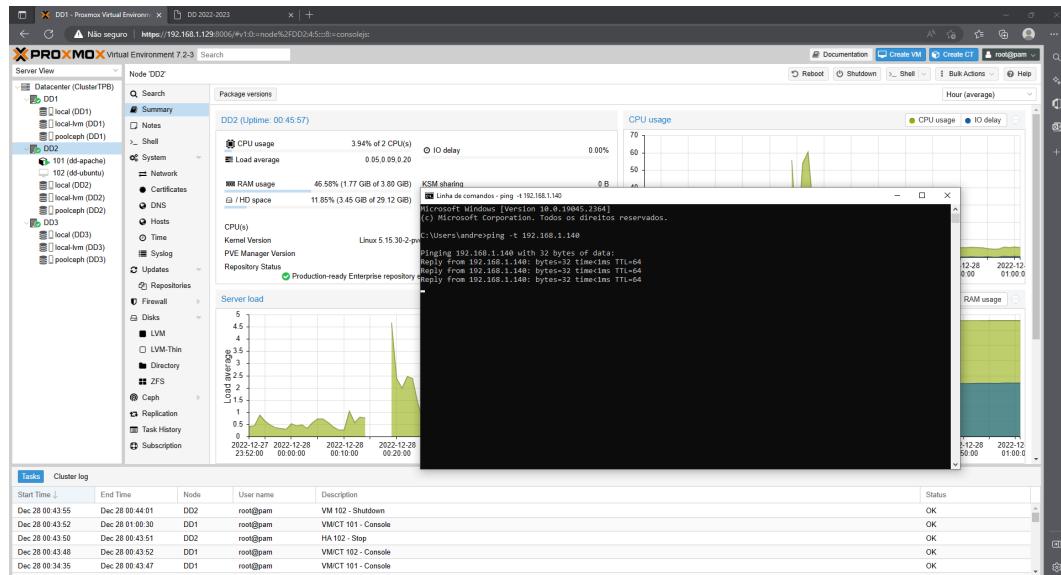


Figura 4.104: Ping para Container

De seguida a falha injetada no *Cluster* foi desligar o servidor proxmox DD2. Passado segundos foi possível ver as máquinas a serem migradas para DD3 (prioridade 10 enquanto DD1 tinha prioridade 5) e a continuidade do ping. O ping pode-se observar que houve uns segundos que dizia *Destination Host Unreachable*. Foi muito rápido e normal devido à migração de nó.

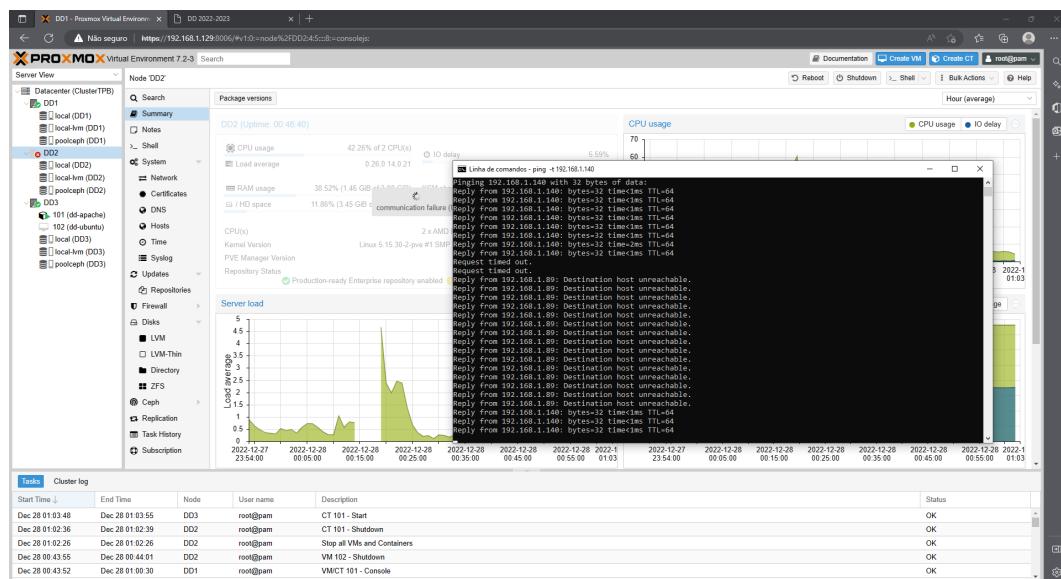


Figura 4.105: Migração

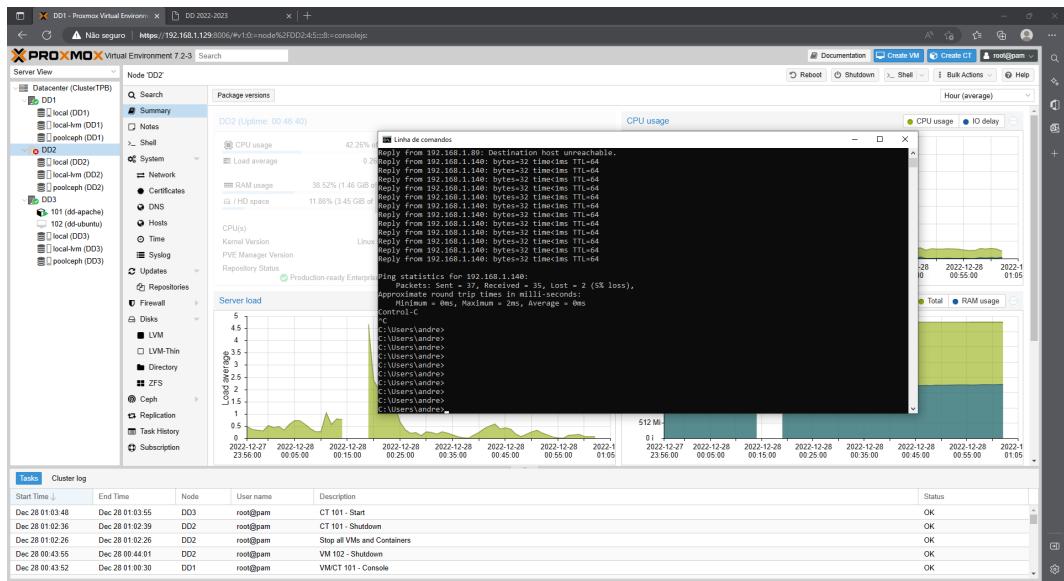


Figura 4.106: Migração Concluída

No final do ping pode-se observar que apenas se perderam 2 pacotes.

Durante a migração, que foi em poucos segundos, em momento algum o Website ficou *Down*. Manteve-se sempre ativo.



Figura 4.107: Website online

Ao injetar a falha no *Cluster* fui informado por *email* acerca do *Overall* do cluster.

FENCE: Try to fence node 'DD2'



Proxmox VE <root@DD3.VM3>

Ontem, 22:24

root@DD3.VM3 ˅

Inbox

The node 'DD2' failed and needs manual intervention.

The PVE HA manager tries to fence it and recover the configured HA resources to a healthy node if possible.

Current fence status: FENCE

Try to fence node 'DD2'

Overall Cluster status:

```
{  
    "manager_status": {  
        "master_node": "DD3",  
        "node_status": {  
            "DD1": "online",  
            "DD2": "unknown",  
            "DD3": "online"  
        },  
        "service_status": {  
            "ct:101": {  
                "node": "DD2",  
                "running": 1,  
                "state": "started",  
                "uid": "frzkNI3+fzX5WdpIXWopcg"  
            },  
            "vm:102": {  
                "node": "DD2",  
                "running": 1  
            }  
        }  
    }  
}
```

Figura 4.108: Overall Cluster

```

"vm:102": {
    "node": "DD2",
    "running": 1,
    "state": "started",
    "uid": "DhUq/N8vYeHCzmi2+xuSBA"
}
},
"timestamp": 1672186605
},
"node_status": {
    "DD1": "online",
    "DD2": "fence",
    "DD3": "online"
}
}

```

Figura 4.109: Overall Cluster

Depois de configurar a Dashboard, Uptime Kuma, com os servidores pretendidos, esperei um tempo para tudo ficar normalizado e só depois injetei a falha no *Cluster* em DD2. O resultado final foi o seguinte e podemos ver a disponibilidade de cada servidor:

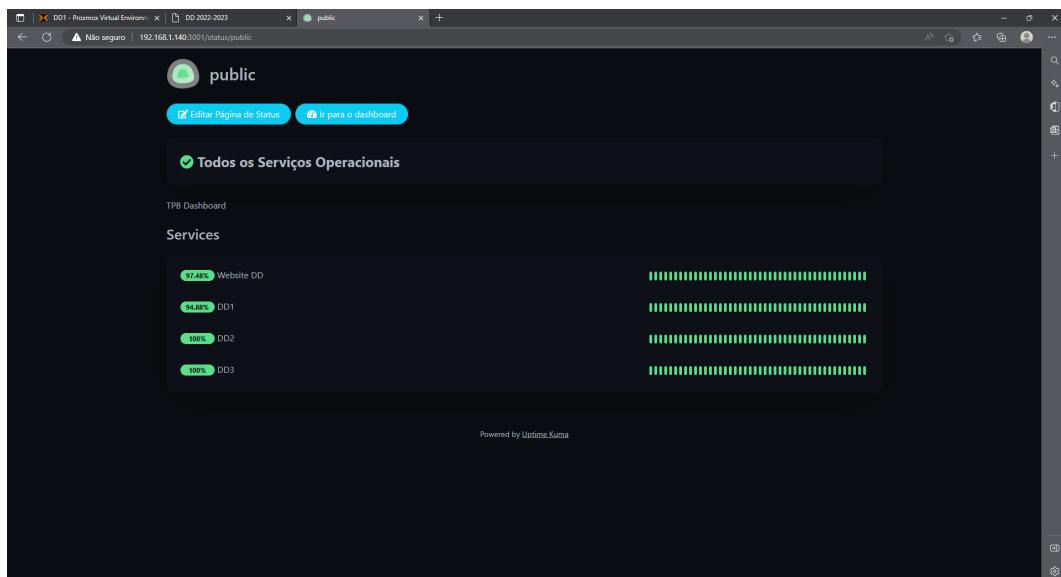


Figura 4.110: Dashboard Uptime Kuma

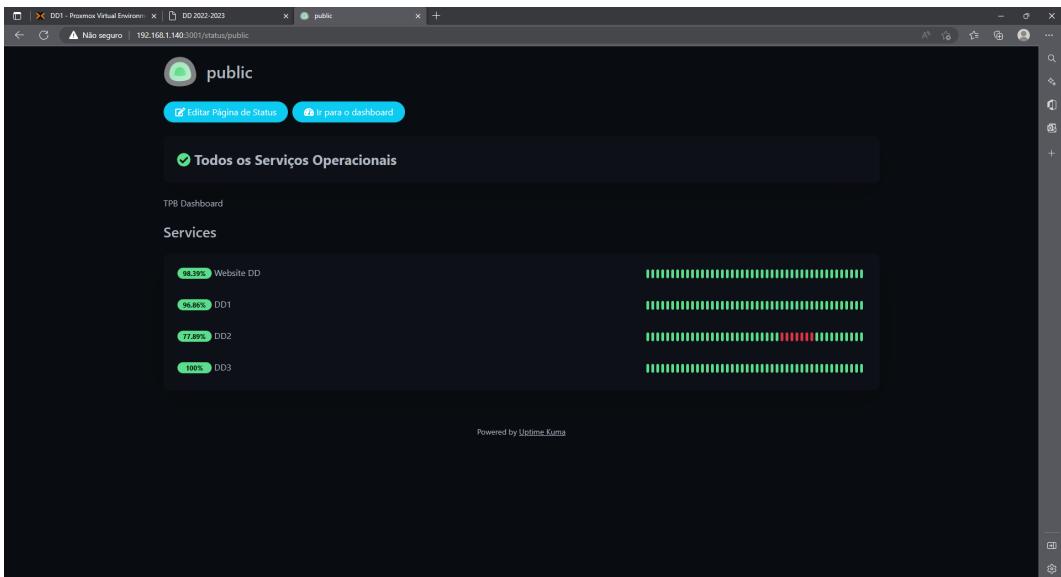


Figura 4.111: Dashboard Uptime Kuma