



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

## Disponibilidade e Desempenho

### Disponibilidade e Desempenho em Aplicações Web Modernas

Rafael Alves - 2014013189

a21240485@isec.pt

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>1</b>
<b>2</b>	<b>Conceitos de Disponibilidade</b>	<b>2</b>
2.1	Disponibilidade ( <i>Availability</i> ) . . . . .	2
2.2	Single Point of Failure . . . . .	2
<b>3</b>	<b>Replicação de estado em Sistemas Distribuídos</b>	<b>3</b>
3.1	Atomicity Consistency Isolation Durability . . . . .	3
3.1.1	Transações ACID . . . . .	3
3.1.2	Características da ACID . . . . .	3
3.2	Two Phase Commit Protocol . . . . .	4
3.2.1	Arquitetura do Protocolo . . . . .	4
3.3	Base de Dados Distribuídas . . . . .	5
3.3.1	Divisão de Base de Dados Distribuídas . . . . .	5
3.3.2	Vantagens e Desvantagens de Base de Dados Distribuídas . . . . .	5
3.4	Sistema de Gestão de Base de Dados . . . . .	6
3.4.1	Requisitos fundamentais de um SGBD . . . . .	6
3.5	Replicação de dados . . . . .	7
3.6	Conclusão sobre o item (2) do enunciado . . . . .	8
3.6.1	Falhas que podem ocorrer no 2PC . . . . .	8
3.6.2	Desvantagens do 2PC . . . . .	8
<b>4</b>	<b>Arquiteturas de Software Modernas</b>	<b>9</b>
4.1	Definição de Arquitetura de Software . . . . .	9
4.2	Evolução das Aplicações Web . . . . .	9
4.3	Arquiteturas mais Populares . . . . .	10
4.4	Arquitetura de Aplicações Web . . . . .	11
4.4.1	Tipos de Arquitetura de Aplicações Web . . . . .	11
<b>5</b>	<b>Escalabilidade Horizontal e Vertical</b>	<b>12</b>
5.1	Escalabilidade . . . . .	12
5.2	Escalabilidade Horizontal ( <i>Scaling Out</i> ) . . . . .	13
5.2.1	Fatores de Mudança na Escalabilidade Horizontal . . . . .	13
5.2.2	Vantagens da Escalabilidade Horizontal . . . . .	13
5.2.3	Desvantagens da Escalabilidade Horizontal . . . . .	13
5.3	Escalabilidade Vertical ( <i>Scaling Up</i> ) . . . . .	13
5.3.1	Fatores de Mudança na Escalabilidade Vertical . . . . .	13
5.3.2	Vantagens da Escalabilidade Vertical . . . . .	14
5.3.3	Desvantagens da Escalabilidade Vertical . . . . .	14
<b>6</b>	<b>Análise de Sistemas de Gestão de Bases de Dados</b>	<b>15</b>
6.1	MySQL . . . . .	15
6.2	MariaDB . . . . .	15

<b>7 Galera Cluster</b>	<b>16</b>
7.1 O que é um Cluster . . . . .	16
7.2 O que é um Galera Cluster . . . . .	16
7.2.1 Replicação Galera Cluster . . . . .	16
7.2.2 Masters and Slaves . . . . .	17
7.2.3 Replicação Síncrona e Assíncrona . . . . .	18
<b>8 Reverse Proxy e Forward Proxy</b>	<b>19</b>
8.1 Reverse Proxy . . . . .	19
8.2 Forward Proxy . . . . .	19
<b>9 Estudo Experimental Meta 4</b>	<b>21</b>
9.1 Requisitos da Experiência Meta 4 . . . . .	21
9.2 Instalação Ubuntu . . . . .	22
9.3 Portas Firewall . . . . .	31
9.3.1 Portas usadas: . . . . .	31
9.4 Instalação MariaDB + Galera . . . . .	32
9.5 Configurar Galera Cluster . . . . .	35
9.6 Iniciar o Galera Cluster . . . . .	35
9.7 Replicação no Cluster . . . . .	40
9.8 Instalação e Configuração Galera Manager . . . . .	44
9.9 Proxy . . . . .	45
9.10 Instalação Wireshark . . . . .	55
9.11 Experiência . . . . .	56
9.12 Sysbench . . . . .	58
<b>10 Estudo Experimental Meta 5</b>	<b>62</b>
10.1 Requisitos da Experiência Meta 5 . . . . .	62
10.2 Segundo Proxy . . . . .	63
10.2.1 Instalação Keepalived . . . . .	64
10.2.2 Configuração Keepalived Proxy . . . . .	65
10.2.3 Configuração Keepalived Proxy1 . . . . .	68
10.3 Experiência . . . . .	70
10.3.1 Estado atual dos proxies . . . . .	70
10.4 Configuração DSR e Healthcheck Keepalived . . . . .	72
10.4.1 Proxy . . . . .	72
10.4.2 Proxy1 . . . . .	73
10.5 Experiência . . . . .	75
<b>A Anexo</b>	<b>A</b>
A.1 Trabalho desenvolvido da Meta 1 . . . . .	A
A.2 Trabalho desenvolvido da Meta 2 . . . . .	A
A.3 Trabalho desenvolvido da Meta 4 . . . . .	A
A.4 Trabalho desenvolvido da Meta 5 . . . . .	B

# **Lista de Tabelas**

9.1 Tabela endereçamento IP Meta 4 . . . . .	21
10.1 Tabela endereçamento IP Meta 5 . . . . .	62

# Listas de Figuras

2.1	Exemplo de um <i>Single Point of Failure</i> (SPOF) . . . . .	2
3.1	Funcionamento do Protocolo 2PC . . . . .	4
3.2	Arquitetura <i>Sistema de Gestão de Base de Dados</i> (SGBD) . . . . .	6
4.1	Evolução das tecnologias Web . . . . .	10
5.1	Escalabilidade Horizontal e Vertical . . . . .	12
7.1	wsrep API . . . . .	17
7.2	Master-Slave . . . . .	17
7.3	Multi-Master . . . . .	18
8.1	Reverse Proxy . . . . .	19
8.2	Forward Proxy . . . . .	20
9.1	Versão . . . . .	21
9.2	Create a new virtual machine . . . . .	22
9.3	Tipo de configuração . . . . .	22
9.4	Install the Operatin System later . . . . .	23
9.5	Tipo Sistema Operativo . . . . .	23
9.6	Nome e localização da <i>Virtual Machine</i> (VM) . . . . .	24
9.7	Tamanho Disco VM . . . . .	24
9.8	Finalização . . . . .	25
9.9	Inserir <i>International Organization for Standardization</i> (ISO) . . . . .	25
9.10	Inicialização VM . . . . .	26
9.11	Instalar Ubuntu . . . . .	26
9.12	Idioma teclado . . . . .	27
9.13	Atualizações . . . . .	27
9.14	Tipo de instalação . . . . .	28
9.15	Confrmação . . . . .	28
9.16	Localização . . . . .	29
9.17	Informações pessoais . . . . .	29
9.18	Instalação . . . . .	30
9.19	Riniciar o Sistema Operativo . . . . .	30
9.20	Vesão mariadb . . . . .	32
9.21	Status mariaDB . . . . .	33
9.22	Login root Mysql . . . . .	33
9.23	Ficheiro Configuração Galera . . . . .	35
9.24	STOP MariaDB . . . . .	35
9.25	Tamanho do cluster no nó 1 . . . . .	36
9.26	Conectividade entre nós do cluster . . . . .	36
9.27	START Cluster Nô 2 . . . . .	37
9.28	Tamanho do cluster no nó 2 . . . . .	37

9.29 Tamanho do cluster no nó 1 . . . . .	38
9.30 START Cluster nó 3 . . . . .	38
9.31 Tamanho do cluster no nó 3 . . . . .	39
9.32 Tamanho do cluster no nó 1 . . . . .	39
9.33 Base de Dados existentes . . . . .	40
9.34 Criação da Base de dados . . . . .	40
9.35 Listagem <i>Base de Dados</i> (BD) . . . . .	41
9.36 Listagem BD . . . . .	41
9.37 Informações do Cluster . . . . .	42
9.38 Informações do Cluster . . . . .	42
9.39 Informações do Cluster . . . . .	43
9.40 Informações do Cluster . . . . .	43
9.41 Informações do Cluster . . . . .	43
9.42 Erro Galera Manager . . . . .	44
9.43 Bind-Address Nós . . . . .	45
9.44 STATUS ProxySQL . . . . .	46
9.45 User Admin . . . . .	46
9.46 Criação Cliente . . . . .	47
9.47 Utilizador ProxySQL-MariaDB . . . . .	48
9.48 Monitorização ProxySQL . . . . .	49
9.49 Configurações ProxySQL . . . . .	49
9.50 Criação tabela mysql_replication_hostgroup . . . . .	50
9.51 Inserir nós ProxySQL . . . . .	51
9.52 Conexão entre Proxy e Nodes . . . . .	52
9.53 Conexão entre Proxy e nós . . . . .	52
9.54 Veirifcação Base de dados no Proxy . . . . .	54
9.55 Configuring Wireshark-common . . . . .	55
9.56 Wireshark . . . . .	55
9.57 Tamanho do cluster depois da falha . . . . .	56
9.58 Pool Hostgroups . . . . .	56
9.59 Pool Hostgroups . . . . .	57
9.60 Run Teste Sysbench . . . . .	59
9.61 Run Teste Sysbench . . . . .	59
9.62 Query Select . . . . .	60
9.63 Statement SELECT MySQL . . . . .	60
9.64 Query INSERT . . . . .	61
9.65 Statement INSERT MySQL . . . . .	61
10.1 Status ProxySQL Proxy1 . . . . .	63
10.2 Show Databases Proxy1 . . . . .	63
10.3 Instalação Keepalived . . . . .	64
10.4 Keepalived Master Configuração . . . . .	65
10.5 Tráfego VRRP . . . . .	66
10.6 Status Keepalived Master . . . . .	66
10.7 Prioridade Master . . . . .	67
10.8 Keepalived Backup Configuração . . . . .	68
10.9 Status Keepalived Backup . . . . .	69
10.10Proxy1 Master . . . . .	70
10.11Proxy Master . . . . .	70
10.12DSR e Healthcheck Proxy . . . . .	73
10.13DSR e Healthcheck Proxy1 . . . . .	74
10.14IP Virtual . . . . .	74

10.15DBeaver conexão . . . . .	75
10.16DBeaver Erro . . . . .	76

# Acrónimos

**DD** *Disponibilidade e Desempenho*

**EI** *Engenharia Informática*

**SPOF** *Single Point of Failure*

**ACID** *Atomicity Consistency Isolation Durability*

**SGBD** *Sistema de Gestão de Base de Dados*

**2PC** *Two-Phase Commit Protocol*

**3PC** *Three-Phase Commit Protocol*

**BD** *Base de Dados*

**P2P** *Peer-to-Peer*

**MVC** *Model-View Controller*

**SOA** *Service-Oriented Architecture*

**IT** *Tecnologia da Informação*

**CPU** *Central Process Unit*

**CSV** *Comma-Separated Values*

**API** *Application Programming Interface*

**I/O** *Input-Output*

**HA** *High Availability*

**HPC** *High Performance Computing*

**WSREP** *Write-Set Replication*

**SSL** *Secure Sockets Layer*

**VM** *Virtual Machine*

**ISO** *International Organization for Standardization*

**SATA** *Serial Advanced Technology Attachment*

**CD/DVD** *Compact Disk/Digital Versatile Disc*

**PWEB** *Programação Web*

**SO** *Sistema Operativo*

**SSH** *Secure Socket Shell*

**HTTP** *Hyper Text Transfer Protocol*

**SST** *State Snapshot Transfer*

**IST** *Incremental State Transfer*

**IP** *Internet Protocol*

**SQL** *Structured Query Language*

**DSR** *Direct Server Return*

**PC** *Personal Computer*

# 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.

Com este relatório pretendo falar sobre a Disponibilidade e Desempenho em Aplicações Web Modernas e demonstrar com experiências o estudo da Disponibilidade. Para o estudo é preciso utilizar VM e construir um cluster em MariaDB + Galera e utilizar o serviço ProxySQL. Para a realização do estudo é preciso primeiro aprofundar conhecimentos sobre a Arquitetura das Aplicações Web.

# Capítulo 2

## Conceitos de Disponibilidade

### 2.1 Disponibilidade (*Availability*)

Disponibilidade é a probabilidade de tempo de um sistema que está em condições de funcionar e pronto a responder a qualquer pedido de serviço a qualquer momento que lhe for pedido. Além de muitos objetivos a serem cumpridos este, *availability*, deve ser visto como o mais prioritário porque um utilizador/empresa não gosta de ficar com um serviço indisponível.

Numa linguagem mais científica presente na EI é usada a regra dos "9" para quantificar a disponibilidade média. No geral, o esperado numa empresa será atingir a meta dos cem por cento ou então "99,999" por cento na regra dos "9".

Existem fatores que comprometem a disponibilidade de uma rede como por exemplo a falha de energia elétrica, falha ou limites de conexão, quebra de algum componente de hardware ou algum ataque informático.

### 2.2 Single Point of Failure

Um SPOF é uma parte de um sistema que, se falhar, interromperá o funcionamento de todo o sistema. *SPOF's* são indesejáveis em qualquer sistema com o objetivo de alta disponibilidade ou fiabilidade.

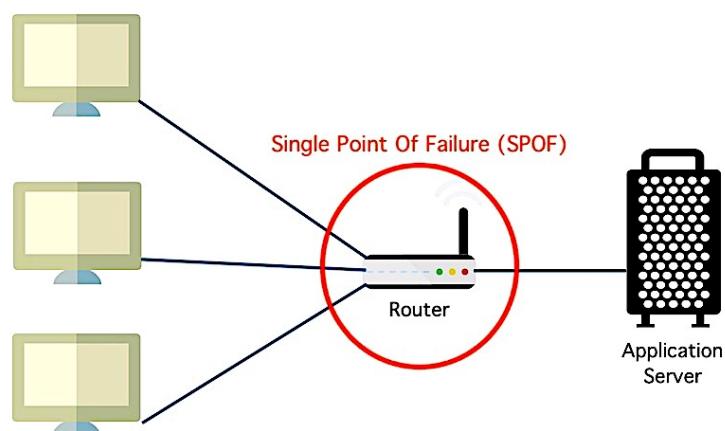


Figura 2.1: Exemplo de um SPOF

# Capítulo 3

## Replicação de estado em Sistemas Distribuídos

### 3.1 Atomicity Consistency Isolation Durability

*Atomicity Consistency Isolation Durability* (ACID) é um conceito em SGBD que identifica um conjunto de propriedades padrão usadas para garantir a fiabilidade de uma determinada base de dados.

#### 3.1.1 Transações ACID

O acesso a diversos itens de dados num sistema distribuído é normalmente acompanhado de transações que têm de preservar as propriedades ACID:

- **A:** Atomicidade
- **C:** Consistência
- **I:** Isolamento
- **D:** Durabilidade

#### 3.1.2 Características da ACID

- *Atomicity* - Numa transação que envolve duas ou mais informações discretas, todas as peças são confirmadas ou nenhuma.
- *Consistency* - Uma transação cria um estado de dados novo e válido ou, se ocorrer alguma falha, retorna todos os dados ao seu estado anterior ao início da transação.
- *Isolation* - Uma transação em processo e ainda não confirmada deve permanecer isolada de qualquer outra operações.
- *Durability* - Os dados comprometidos são salvos pelo sistema de forma que, mesmo em caso de falha e reinicialização do sistema, os dados estejam disponíveis no estado correto.

## 3.2 Two Phase Commit Protocol

O protocolo *Two-Phase Commit Protocol* (2PC) (protocolo de recuperação) divide uma confirmação de BD em duas fases para garantir a correção e a tolerância a falhas num sistema de base de dados distribuído. Assume o modelo *fail-stop* (falha-parar): os *nodes* que falharem, deixam de estar ativos (param) e não provocam mais erros.

### 3.2.1 Arquitetura do Protocolo

Considere um coordenador de transações que coordena as confirmações dos armazenamentos de base de dados. Como o nome sugere, todo o processo é dividido em duas fases:

#### Fase *Prepare*:

1. Após cada armazenamento da base de dados (*slave*) ter concluído a sua transação localmente, ele envia uma mensagem "*DONE*" para o coordenador da transação. Uma vez que o coordenador recebe esta mensagem de todos os *slaves*, ele envia aos *slaves* uma mensagem "*PREPARE*".
2. Cada *slave* responde à mensagem "*PREPARE*" enviando uma mensagem "*READY*" de volta ao coordenador.
3. Se um *slave* responder com uma mensagem "*NOT READY*" ou não responder, então o coordenador envia uma mensagem global "*ABORT*" para todos os outros *slaves*. Apenas ao receber uma confirmação de todos os *slaves* de que a transação foi abortada o coordenador considera a transação inteira abortada.

#### Fase *Commit*:

1. Uma vez que o coordenador da transação recebeu a mensagem "*READY*" de todos os *slaves*, ele envia uma mensagem "*COMMIT*" para todos eles, que contém os detalhes da transação que precisa de ser armazenada na base de dados.
2. Cada *slave* aplica a transação e retorna uma mensagem de confirmação "*DONE*" de volta ao coordenador.
3. O coordenador considera toda a transação concluída assim que receber uma mensagem "*DONE*" de todos os *slaves*.

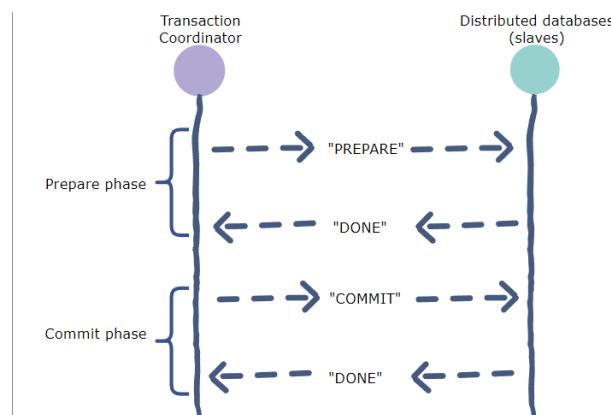


Figura 3.1: Funcionamento do Protocolo 2PC

Para finalizar, o 2PC, está relacionado com os princípios já abordados anteriormente, ACID, porque este protocolo garante a atomicidade, a consistência, o isolamento e a durabilidade de uma transação.

## 3.3 Base de Dados Distribuídas

Um sistema de bases de dados distribuídas consiste em múltiplos servidores, cada um responsável pelos seus dados. Estes dados podem ser acessados utilizando uma rede e, apesar de estarem fisicamente distribuídos, devem apresentar-se ao utilizador logicamente integrados, como se estivessem num único servidor.

### 3.3.1 Divisão de Base de Dados Distribuídas

Podemos dizer que Bases de Dados Distribuídas se encontram divididas em dois sistemas:

1. Sistemas Homogéneos: todos os *nodes* usam o mesmo SGBD mas os dados estão armazenados em várias BD.
2. Sistemas Heterogéneos: cada *node* utiliza um SGBD diferente onde partilham BD pré existentes com mesmo modelo de dados ou modelos de dados diferentes.

### 3.3.2 Vantagens e Desvantagens de Base de Dados Distribuídas

- Vantagens: Uma das vantagens para utilizar Base de Dados Distribuídas é pelo fato, como o nome indica, de ser de natureza distribuída. Ou seja, geograficamente se uma organização tiver várias sedes espalhadas pelo país faz sentido utilizar uma base de dados distribuída, pois vai espelhar a estrutura da organização, vai implicar que cada sede tenha o seu próprio servidor. Resumidamente, existe maior segurança, partilha de dados e autonomia local, desempenho melhorado (em relação às BD), fiabilidade/disponibilidade e é mais vantajoso para representar várias organizações.
- Desvantagem: Ao contrário da vantagem dada, não se justifica ter vários servidores espalhados quando uma organização possui geograficamente as suas delegações/sedesumas ao lado das outras. Neste caso faz mais sentido ter uma solução cliente/servidor. Resumidamente, tem custos de desenvolvimento de Software, maior possibilidade de erros no software, aumento da carga de processamento e aumento da complexidade da coordenação dos *nodes*.

## 3.4 Sistema de Gestão de Base de Dados

Um SGBD é o software que gera o armazenamento, manipulação e pesquisa dos dados existentes na BD, funcionando como uma interface entre as aplicações e os dados necessários para a execução dessas aplicações.

Exemplos SGBD: Microsoft SQL Server, Oracle Server, MariaDB, MySQL, entre outros.

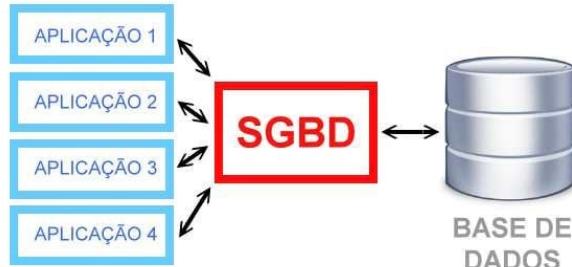


Figura 3.2: Arquitetura SGBD

### 3.4.1 Requisitos fundamentais de um SGBD

- Segurança: proteção da BD contra acessos não autorizados;
- Integridade: validação de operações que coloquem em risco a consistência dos dados;
- Controlo de concorrência nos acessos: coordenação da partilha dos dados pelos vários utilizadores
- Recuperação de falhas: restaurar a integridade da BD depois da ocorrência de uma falha. Mecanismos de recuperação (fundamentalmente baseados em redundância): backups, *transaction logging* (ficheiro *transaction log*, dados para repor as últimas transações).

### 3.5 Replicação de dados

A replicação é a técnica de duplicar dados em vários *nodes* diferentes. Para tornar a replicação segura existe um conjunto de razões para a duplicação dos dados, que passo a citar:

- Prevenção contra falhas: a falha de um *node* que contenha dados importantes ao sistema não compromete, necessariamente, o seu funcionamento;
- Redução da transferência de dados: há uma maior probabilidade de os dados estarem disponíveis localmente ou mais perto do *node* que os requisita. Isto resulta na redução dos custos de transferência;
- Paralelismo: os pedidos à BD podem ser processados por vários *nodes* em paralelo, o que vai existir um maior desempenho.

Como não falamos só de vantagens, neste caso também existe desvantagens e neste cenário de replicação uma das desvantagens é a necessidade de manter uma coerência total do sistema. A coerência total do sistema implica que quando a informação num *node* é atualizada, as suas réplicas também devem ser atualizadas, o que aumenta o *overhead* do sistema (custos de atualização e necessidade de algoritmos complexos para a execução de transações).

Dentro do sistema, conforme as atualizações propagadas, a replicação pode ser **síncrona** (ocorre em tempo real e é preferida para sistemas com objetivos de baixo tempo de recuperação que não podem perder dados), **assíncrona** (replicação lenta e projetado para trabalhar em distâncias e requer menos largura de banda).

Para concluir, quando os dados têm de estar permanentemente atualizados, é preferível recorrer à replicação síncrona. Caso contrário, deve-se utilizar a replicação assíncrona por ter melhores tempos de resposta e até é possível indicar intervalos de tempo para as atualizações.

A replicação traz mais benefícios quando os dados são lidos muitas vezes pelos servidores remotos, mas poucas vezes atualizados, devido à carga adicional que é colocada sobre o sistema para manter as réplicas consistentes.

## 3.6 Conclusão sobre o item (2) do enunciado

O protocolo 2PC assegura que a propriedade da atomicidade da transação é respeitada. Caso exista um *Abort* de um dos *nodes* a transação falha.

O 2PC pode ficar inoperante devido à falha de um dos *nodes* durante o processo. Uma transação inoperante não é desejável porque continua a manter os *locks* dos recursos de que necessita para a sua transação, impedindo que outras transações que precisem desses mesmos recursos os possam obter.

O objetivo das técnicas de tolerâncias de falhas nas Base de Dados Distribuídas é garantir que não há perda/corrupção de dados.

O SGBD é responsável por tudo, guardar os dados no disco rígido, manter em memória os dados mais acedidos, ligar dados e metadados (dados sobre outros dados), disponibilizar uma interface para programas e utilizadores. Isto são algumas das vantagens de um SGBD mas como temos visto também tem as suas desvantagens. Trabalhar com três SGBD implica utilizar mais do que um utilizador e não pode prejudicar nenhum deles. O controlo de acesso irá garantir a integridade dos dados com a possibilidade de configurar níveis de acesso a cada utilizador (segurança).

### 3.6.1 Falhas que podem ocorrer no 2PC

- Falhas nos *nodes*;
- Falhas na rede;
- Falhas de Software;
- Falha do coordenador.

### 3.6.2 Desvantagens do 2PC

Como já foi referido atrás, uma das desvantagens é o protocolo 2PC ser demasiado restritivo, basta que um dos *nodes* falhe ou fique inoperável e os restantes *nodes* efetuam um *rollback*. Uma segunda desvantagem é se existir uma falha após a primeira fase (fase antes do envio das instruções de *commit* ou *abort*, todas as sub-transações ficam bloqueadas (num estafo de espera ativa mantendo os *locks* que possuíam.

Para resolver estes problemas (Desvantagens acima descritas) foi criado o protocolo *Three-Phase Commit Protocol* (3PC). Um protocolo não muito usado devido ao elevado tráfego que necessitava. O 3PC é um protocolo semelhante ao 2PC mas com um passo adicional intermédio *Pre-Commit*, que confirma se todos os *nodes* estão em condições de finalização.

# Capítulo 4

## Arquiteturas de Software Modernas

### 4.1 Definição de Arquitetura de Software

A arquitetura de software possibilita entender as diferenças entre as linguagens, sistemas operativos, ou seja, qualquer componente tecnológico pode ser usado para integrar uma solução de arquitetura de software. É essencial pois otimiza o trabalho dos designers e programadores, permitindo que uma aplicação esteja dentro dos padrões básicos necessários para funcionar de forma assertiva.

A Arquitetura de Software é importante para automatizar novos processos ou melhorar os já existentes, mas para isso envolve:

- Decisões sobre as estruturas do sistema;
- Controlo;
- Protocolos de comunicação, sincronização e acesso a dados;
- Desempenho e outros atributos de qualidade.

Implementar uma arquitetura de software traz diversos benefícios para o sistema. Entre eles, os três principais são:

- Performance;
- Escalabilidade;
- Flexibilidade.

### 4.2 Evolução das Aplicações Web

O desenvolvimento web, a par de outras áreas tecnológicas, está em constante evolução. O que vemos hoje não é o mesmo que víamos à 10 anos atrás e a prova disso são as aplicações que usamos hoje. Como consequência o acesso à informação digital também mudou drasticamente. Agora são os dispositivos móveis a serem a opção preferida para aceder à informação. A grande evolução das aplicações web na minha opinião foi devido ao avanço e à quantidade de dispositivos móveis. Ainda hoje estamos perante mudanças a "toda a hora" por causa dos dispositivos móveis, quer seja a nível de qualidade ou desempenho.

Com a imagem seguinte podemos verificar o avanço das tecnologias em função do tempo. Verificamos que começamos a ter um grande avanço com a tecnologia *Web 2.0* e mais recentemente a recém chegada *Web 3.0*.

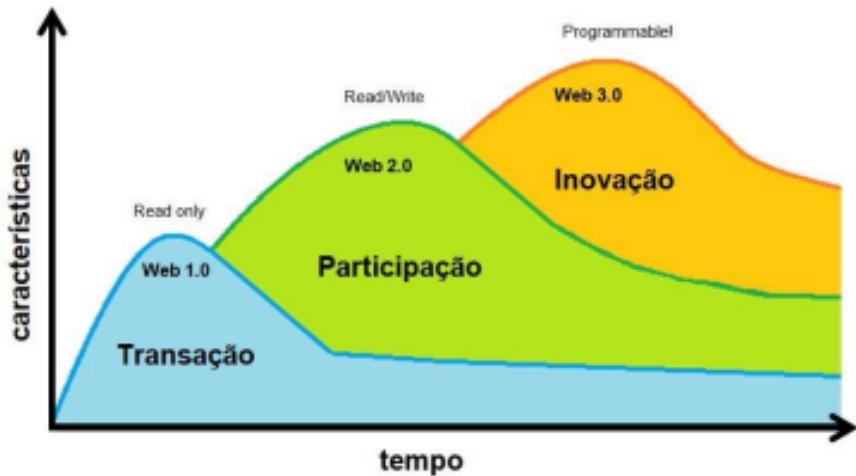


Figura 4.1: Evolução das tecnologias Web

### 4.3 Arquiteturas mais Populares

Alguns exemplos das arquiteturas mais populares:

- Layers (Camadas)
- Client-Server (Cliente-Servidor)
- *Model-View Controller* (MVC)
- *Peer-to-Peer* (P2P)
- Microservices (Micro serviços)

A arquitetura em camadas, como o nome indica, é organizada num conjunto de camadas, oferecendo maior flexibilidade e suporte a portabilidade. A identificação do nível de abstração nem sempre é evidente e perde-se desempenho à medida que o número de camadas cresce. Um exemplo onde é aplicado esta arquitetura é nas aplicações *E-Commerce*.

Na arquitetura Cliente-Servidor o processamento da informação divide-se em módulos e processos distintos, sendo um deles responsável pela manutenção da informação e o outro pela obtenção de dados. Um exemplo do seu uso é no serviço de *e-mail*.

A arquitetura MVC é uma referência para nós (estudantes), uma vez que é uma arquitetura lecionada na disciplina de *Programação Web* (PWEB). O sistema MVC separa o projeto do software em três camadas independentes: o modelo/model (manipulação lógica de dados), a visão/view (interface do usuário) e o controlador/controller (fluxo de aplicação), facilitando a manutenção do código, que pode ser reutilizado em outros projetos.

Passando ao P2P, todos os pares são clientes e servidores, ou seja, cada computador é um provedor de serviços independente de um servidor central. *Torrent*, a aplicação tão conhecida por nós onde usa esta arquiterura.

Em último, e não menos importante, temos a arquitetura Microservices onde se baseia em múltiplos serviços e componentes para desenvolver uma estrutura modular. É o padrão mais utilizado por *devs* e engenheiros(as) de software, por permitir escalabilidade e independência dos módulos, que podem usar diferentes linguagens.

## 4.4 Arquitetura de Aplicações Web

A arquitetura de aplicações Web é o ponto de partida para desenvolver uma aplicação. No entanto, existem vários aspectos de implementação. Por exemplo, como etapa inicial, a escolha de uma linguagem de programação para desenvolver a aplicação.

Existe uma panóplia de linguagens de programação usadas no desenvolvimento de software. Algumas delas utilizadas na criação de determinados tipos de aplicações, como *Swift*, *Java*, para aplicações mobile ou *JavaScript* para desenvolvimento *front-end*.

### 4.4.1 Tipos de Arquitetura de Aplicações Web

- *Service-Oriented Architecture* (SOA)
- Microservices (micro serviços)

# Capítulo 5

## Escalabilidade Horizontal e Vertical

### 5.1 Escalabilidade

A escalabilidade é a capacidade de um sistema aumentar ou reduzir rapidamente a potência ou tamanho da infra-estrutura, de armazenamento ou de rede. Com a evolução dos requisitos e exigências dos recursos das aplicações, a escalabilidade da infra-estrutura de armazenamento proporciona um meio de adaptação às exigências dos recursos, otimizando os custos e melhorando a eficiência da equipa de *Tecnologia da Informação* (IT).

Escalabilidade Vertical (*Scaling Up*) e Escalabilidade Horizontal (*Scaling Out*) são métodos chave que as organizações utilizam para acrescentar capacidade às suas infra-estruturas. Para um utilizador final, estes dois conceitos podem parecer a mesma função. No entanto, cada um deles trata de necessidades específicas e resolve questões específicas de capacidade para a infra-estrutura do sistema de diferentes maneiras.

A escalabilidade vertical está a acrescentar mais recursos, como discos rígidos e memória, para aumentar a capacidade computacional dos servidores. Enquanto que a escalabilidade horizontal está a adicionar mais servidores à sua arquitetura para distribuir a carga de trabalho por mais máquinas.

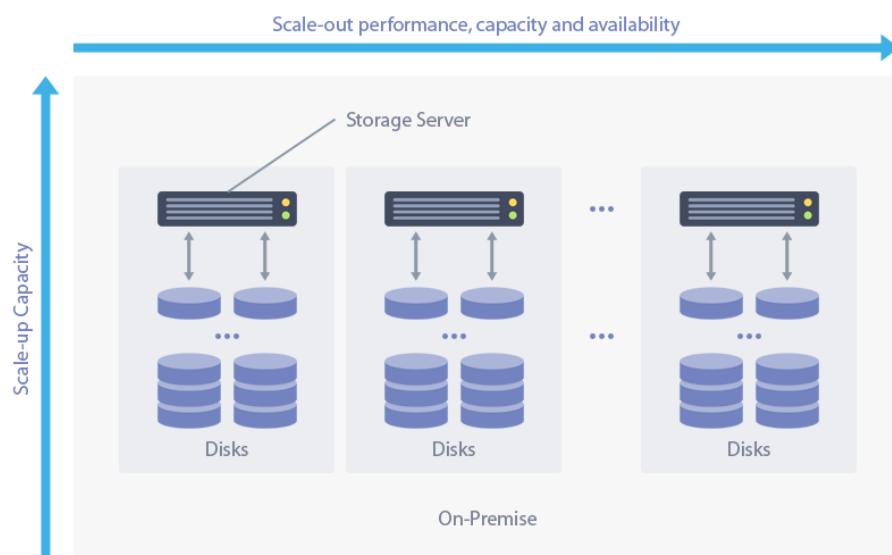


Figura 5.1: Escalabilidade Horizontal e Vertical

## 5.2 Escalabilidade Horizontal (*Scaling Out*)

A infra-estrutura da escalabilidade horizontal substitui o hardware pela funcionalidade, desempenho e capacidade de armazenamento. Esta escalabilidade resolve algumas das limitações da infra-estrutura, uma vez que é geralmente mais eficiente e eficaz. Além disso, a escalabilidade utilizando a nuvem (cloud) assegura que não terá de comprar novo hardware sempre que quiser atualizar o seu sistema.

### 5.2.1 Fatores de Mudança na Escalabilidade Horizontal

- Estratégia de escalabilidade de longo prazo: Os componentes podem ser acrescentados ou removidos, dependendo dos seus objetivos;
- Atualizações flexíveis: A escalabilidade evita as limitações da depreciação da tecnologia;
- Distribuição (*Load Balancing*) do armazenamento.

### 5.2.2 Vantagens da Escalabilidade Horizontal

- Tecnologias mais recentes de servidores;
- Adaptabilidade às mudanças da procura;
- Gestão de custos.

### 5.2.3 Desvantagens da Escalabilidade Horizontal

- Espaço em rack limitado;
- Custos operacionais acrescidos;
- Custos iniciais mais elevados.

## 5.3 Escalabilidade Vertical (*Scaling Up*)

A Escalabilidade Vertical da infra-estrutura visa acrescentar recursos de apoio a uma aplicação para melhorar ou manter um desempenho amplo. Tanto os recursos virtuais como de hardware podem ser aumentados/melhorados. No contexto do hardware, pode ser tão simples como utilizar um disco rígido maior para aumentar a capacidade de armazenamento. O aumento da infra-estrutura é viável até que os componentes individuais sejam impossíveis de aumentar, tornando isto uma solução a curto prazo.

### 5.3.1 Fatores de Mudança na Escalabilidade Vertical

- Impacto no desempenho: Um bom indicador de quando aumentar a escala é quando as suas cargas de trabalho começam a atingir os limites de desempenho, resultando no aumento da latência e desempenho causados pela capacidade de *Input-Output* (I/O) e *Central Process Unit* (CPU);
- Quando a otimização do armazenamento não funciona: Sempre que a eficácia das soluções de otimização do desempenho e da capacidade diminui, é tempo de mudança na escalabilidade vertical.

### **5.3.2 Vantagens da Escalabilidade Vertical**

- Velocidade;
- Simplicidade;
- Relação custo-benefício;
- Consumo de energia limitado;

### **5.3.3 Desvantagens da Escalabilidade Vertical**

- Latência;
- Mão de obra e riscos;
- Equipamento antigo.

# Capítulo 6

## Análise de Sistemas de Gestão de Bases de Dados

### 6.1 MySQL

O MySQL é a BD de código aberto (*Open Source*) mais popular do mundo que utiliza a linguagem *Structured Query Language* (SQL) como interface. De acordo com a DB-Engines, o MySQL é a segunda base de dados mais popular, atrás da Base de Dados Oracle. MySQL é um SGBD relacional que suporta vários motores de armazenamento (InnoDB, *Comma-Separated Values* (CSV), entre outros) e dependendo do seu ambiente de programação, pode ser introduzido SQL diretamente (por exemplo, para gerar relatórios), incorporar instruções SQL em código escrito noutra linguagem, ou utilizar uma *Application Programming Interface* (API) específica da linguagem que esconde a sintaxe SQL.

### 6.2 MariaDB

O MariaDB é um dos mais populares servidores de bases de dados relacionais do mundo. Foi feito pelos criadores do MySQL e também é de código aberto (*Open Source*). A intenção é também manter uma elevada compatibilidade com MySQL, assegurando uma equivalência de bibliotecas e correspondência exata com API e comandos MySQL. Em versões mais recentes o MariaDB inclui o Galera Cluster o que facilita bastante o processo de criação de um cluster usando o SGBD MariaDB.

# Capítulo 7

## Galera Cluster

### 7.1 O que é um Cluster

Um Cluster é um grupo de computadores interligados que trabalham em conjunto para apoiar aplicações e middleware (por exemplo, BD). Num Cluster, cada computador é referido como um "nó" ou *node*. Ao contrário dos computadores de rede, em que cada nó executa uma tarefa diferente, os clusters de computadores atribuem a mesma tarefa a cada nó. Os nós de um cluster estão normalmente ligados entre si através de redes locais de alta velocidade. Cada nó executa a sua própria instância de um sistema operativo.

Os Clusters são frequentemente utilizados para *High Performance Computing* (HPC) e *High Availability* (HA). Se um único componente falhar num Cluster, os outros nós continuam a operar e a fornecer todo o processamento. Os clusters são normalmente dedicados a funções específicas, tais como alta disponibilidade, alto desempenho ou processamento em grande escala.

### 7.2 O que é um Galera Cluster

Galera Cluster é um cluster de BD multi-master baseado em replicação síncrona. Quando o Galera Cluster está em uso, as leituras e gravações da BD podem ser dirigidas a qualquer nó. Qualquer nó individual pode ser perdido sem interrupção nas operações e sem utilizar procedimentos complexos de *failover*.

#### 7.2.1 Replicação Galera Cluster

O Galera Cluster consiste num servidor de base de dados (MySQL ou MariaDB) que utiliza o plugin de Replicação Galera para gerir a replicação. O API do plugin de replicação MySQL foi criada para fornecer toda a informação para uma replicação multi-master e síncrona. Esta API é chamada de *Write-Set Replication* (WSREP) API.

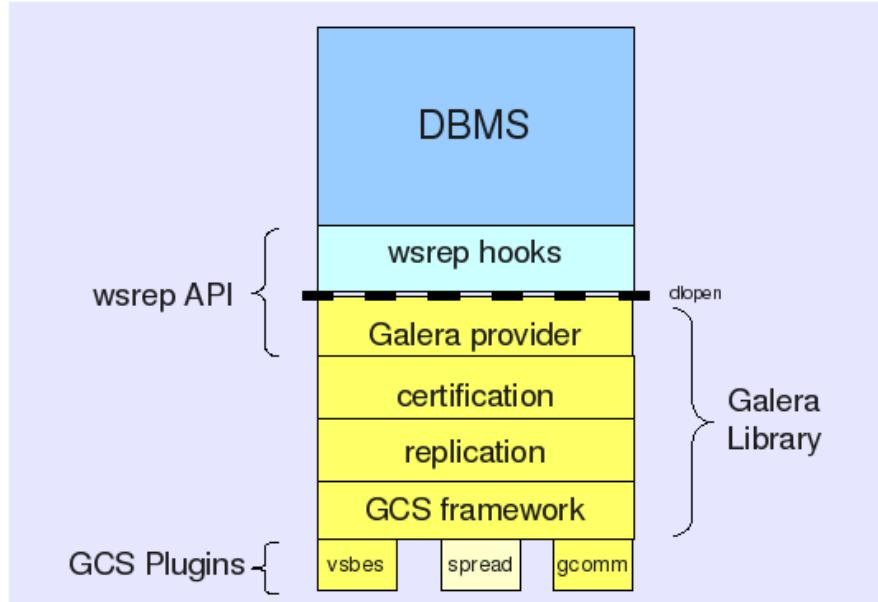


Figura 7.1: wsrep API

Através do WSREP API, o Galera Cluster replica, entre todos os nós, a mesma informação mantendo todos os nós o mais atualizado possível. Mesmo que um nó deixe de ficar operacional os restantes assumem o papel para não comprometer o serviço e quando esse nó voltar a entrar no ativo recebe toda a informação que deixou de receber. A isto se chama uma replicação síncrona.

### 7.2.2 Masters and Slaves

Existem 2 modos de replicação no sistema de gestão de base de dados, *Master-Slave* ou *Multi-Master*. No modo *Master-Slave* o servidor da base de dados principal regista as atualizações dos dados e propaga esses registos através da rede para os *slaves*. Os servidores de base de dados de *slaves* recebem um fluxo de atualizações do *master* e aplicam essas alterações.

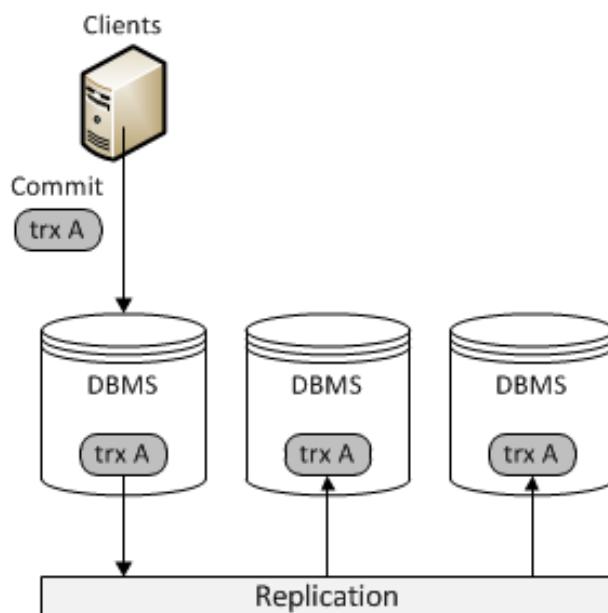


Figura 7.2: Master-Slave

No modo *Multi-Master*, todos os nós de BD funcionam como *master*. É possível submeter atualizações a qualquer nó da BD porque estas se propagam através da rede para todos os nós.

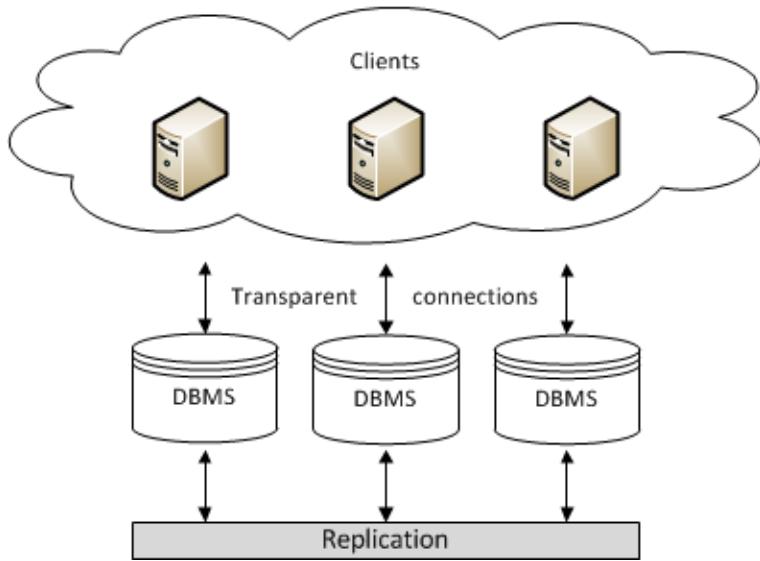


Figura 7.3: Multi-Master

### 7.2.3 Replicação Síncrona e Assíncrona

Replicação síncrona não é nada mais do que todos os nós do cluster receberem a atualização que aconteceu num determinado nó. Ou seja, qualquer que seja a alteração feita num dos nós do cluster, todos os restantes vão receber imediatamente a atualização.

Replicação assíncrona não garante que a replicação chegue a todos os nós, tanto em tempo útil quanto a disponibilidade. A replicação assíncrona pode ser curta ou longa e implica também se um nó *master* falhar, algumas das atualizações podem ser perdidas.

# Capítulo 8

## Reverse Proxy e Forward Proxy

### 8.1 Reverse Proxy

Um servidor *Reverse Proxy* transfere os pedidos dos clientes para esses servidores. Os *Reverse Proxy* são geralmente utilizados para aumentar a proteção, a velocidade e a fiabilidade. Um *Reverse Proxy* recebe o pedido de um cliente, passa-o para outro servidor, e depois reencaminha-o de volta para o cliente, fazendo-o parecer como se fosse o servidor proxy inicial. *Reverse Proxy* asseguram que os utilizadores não chegam diretamente ao servidor de origem, fornecendo assim anonimato a este servidor web.

*Reverse Proxy* podem proteger os servidores web, aumentar o desempenho do website, e ajudar a evitar a sobrecarga. Os *Reverse Proxy* são também utilizados para balanceamento de carga, caching, e encriptação *Secure Sockets Layer* (SSL).

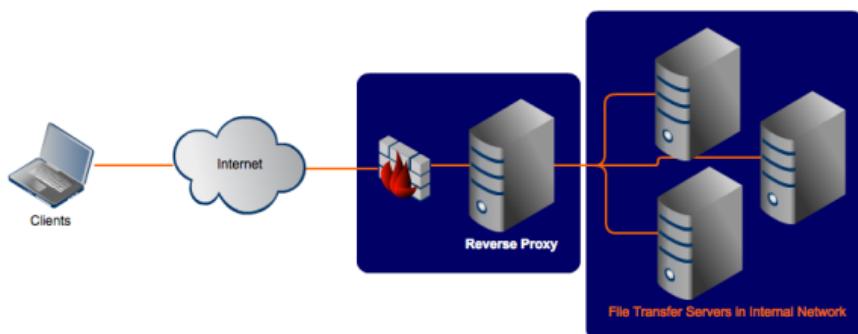


Figura 8.1: Reverse Proxy

### 8.2 Forward Proxy

*Forward Proxy* atua como intermediário entre os utilizadores e os servidores web. Isto significa que o pedido do utilizador passa primeiro pelo *Forward Proxy* e depois chega à página web. Estes são enviados para o servidor proxy, redirecionando-os de volta para o utilizador. O pedido é feito pelo próprio servidor proxy e não pelo utilizador. Uma vez que um *Forward Proxy* pode ser visto como um ponto de acesso e controlo, pode aumentar a segurança dos utilizadores dentro de uma rede privada.

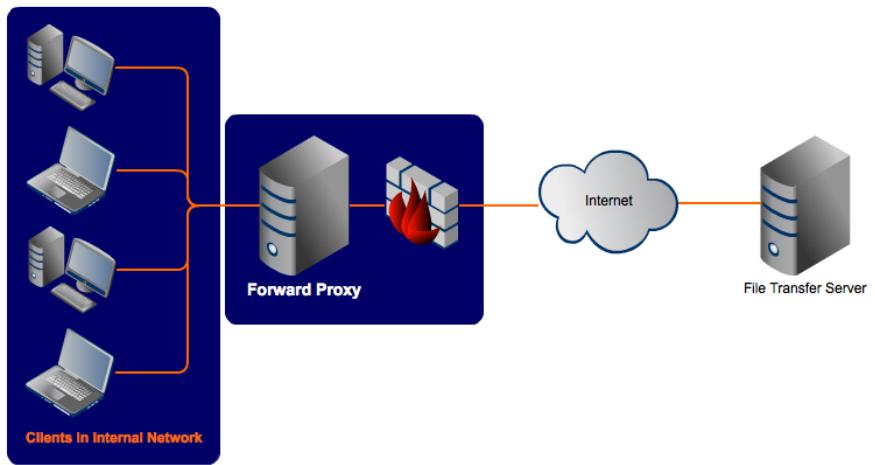


Figura 8.2: Forward Proxy

# Capítulo 9

## Estudo Experimental Meta 4

### 9.1 Requisitos da Experiência Meta 4

Para este estudo experimental foi criado um cluster Multi-Master com recurso ao MariaDB e Galera.

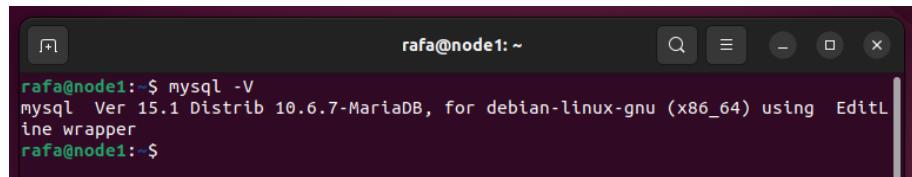
Virtualizador: VMware Workstation Pro 16.2.0 (Licensed)

Sistema Operativo: Linux Ubuntu 22.04: <https://ubuntu.com/download/desktop>

MariaDB: 10.6.7

Mysql: 15.1

Galera: 4



```
rafa@node1:~$ mysql -V
mysql Ver 15.1 Distrib 10.6.7-MariaDB, for debian-linux-gnu (x86_64) using EditLine wrapper
rafa@node1:~$
```

Figura 9.1: Versão

Endereçamento:

Nome	IP
Node 1	192.168.1.118
Node 2	192.168.1.119
Node 3	192.168.1.120
Sysbench	192.168.1.121
Proxy	192.168.1.122

Tabela 9.1: Tabela endereçamento IP Meta 4

## 9.2 Instalação Ubuntu

Nesta secção explico passo a passo a instalação da VM Ubuntu, nó 1. A instalação das outras VM é repetir o mesmo processo.

Passo 1: "Create a new virtual machine".

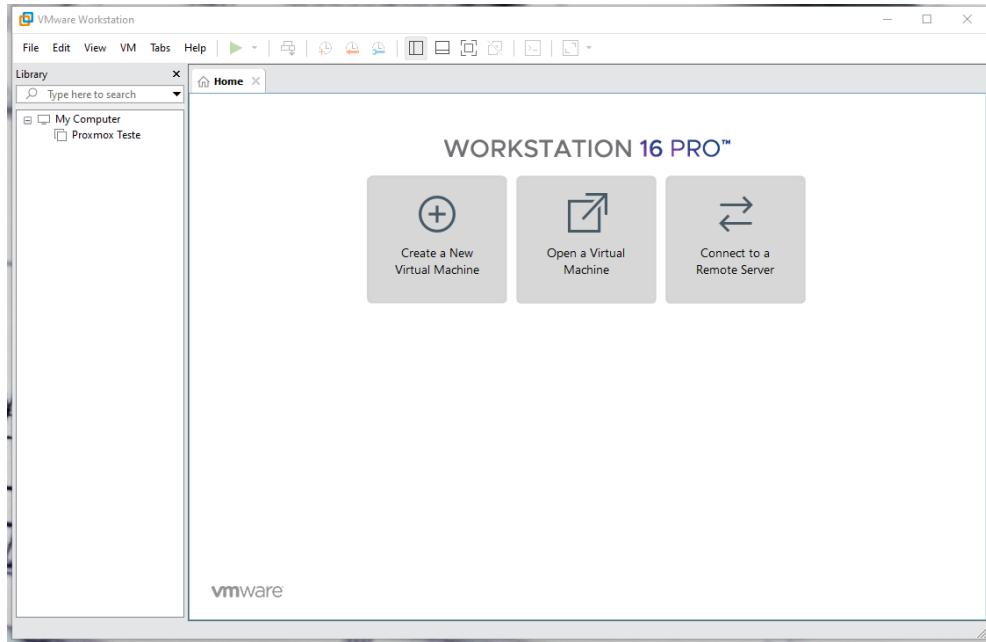


Figura 9.2: Create a new virtual machine

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

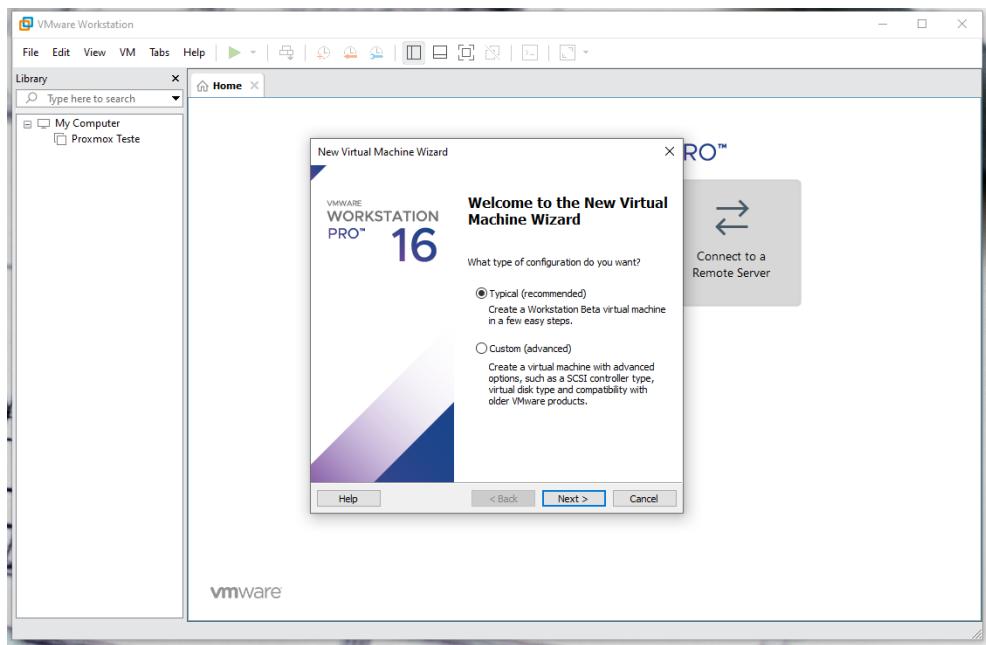


Figura 9.3: Tipo de configuração

Passo 3: Escolher instalar o Sistema Operativo mais tarde.

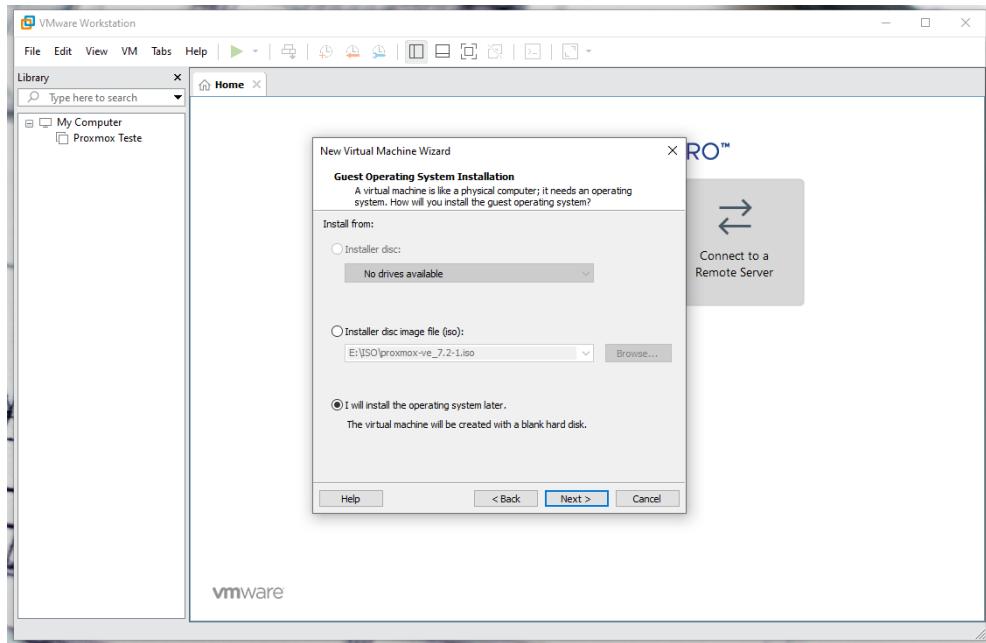


Figura 9.4: Install the Operatin System later

Passo 4: Por pré-definição deixei ficar as opções dadas uma vez que também ia instalar Ubuntu.

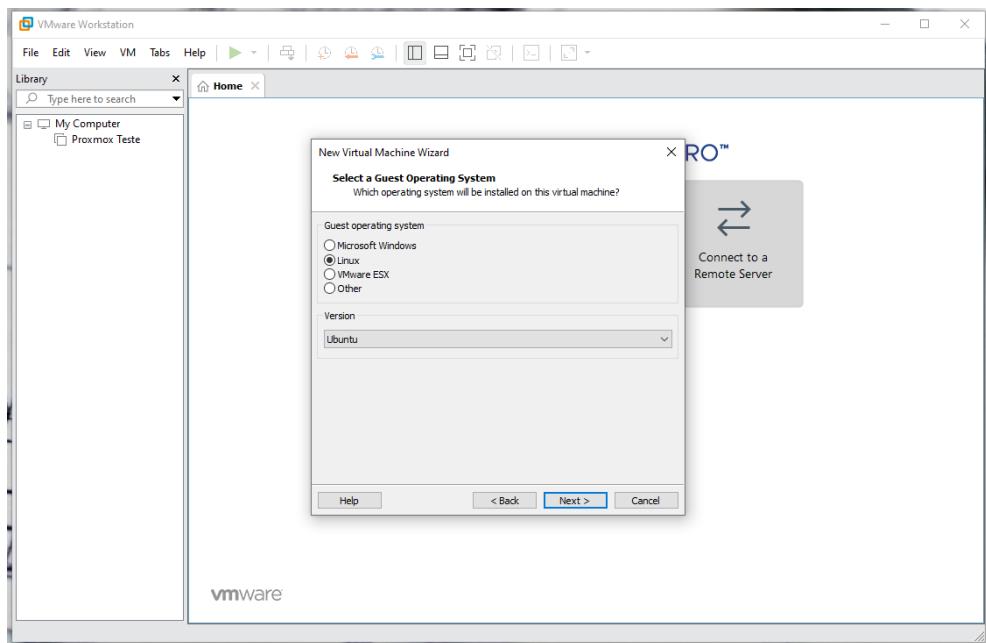


Figura 9.5: Tipo Sistema Operativo

Passo 5: Escolher o nome para a VM e a localização.

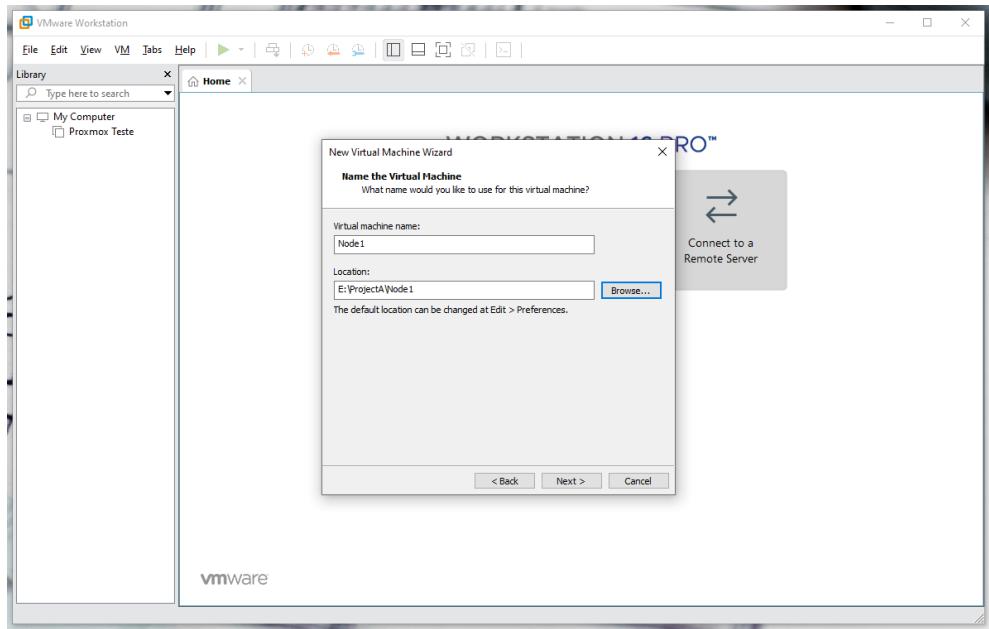


Figura 9.6: Nome e localização da VM

Passo 6: Definir tamanho do disco para a VM.

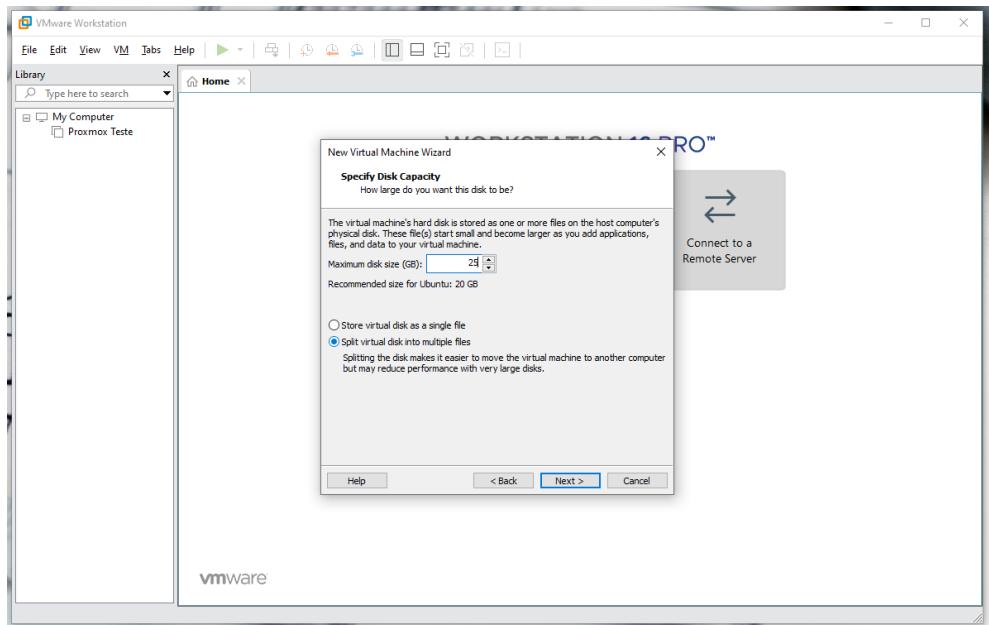


Figura 9.7: Tamanho Disco VM

## Passo 7: Resumo da configuração e finalização.

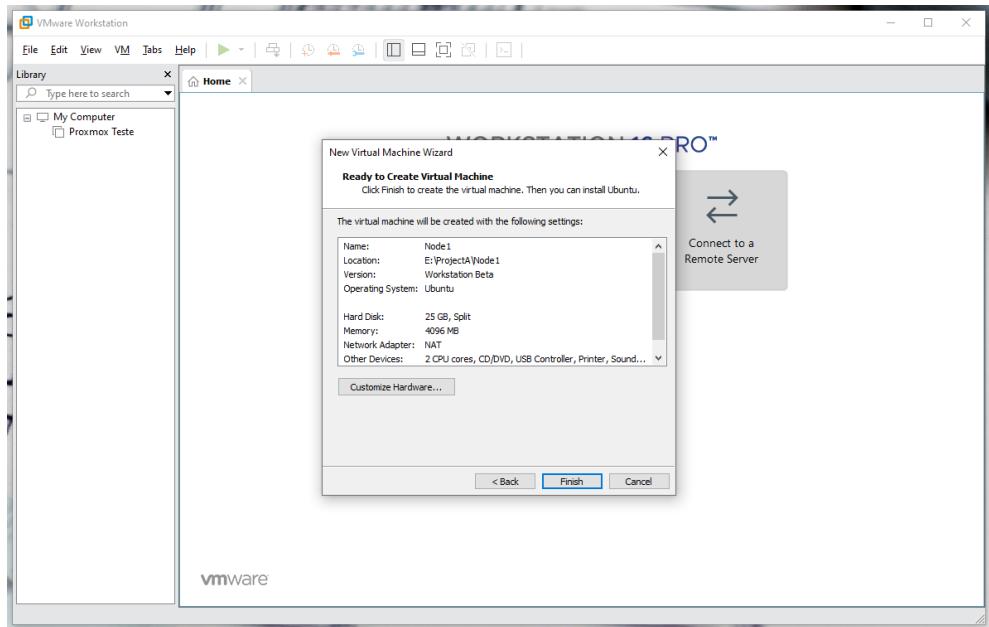


Figura 9.8: Finalização

Passo 8: Neste passo vamos às propriedades da VM inserir a ISO do Ubuntu em "*Compact Disk/Digital Versatile Disc (CD/DVD) (Serial Advanced Technology Attachment (SATA))*".

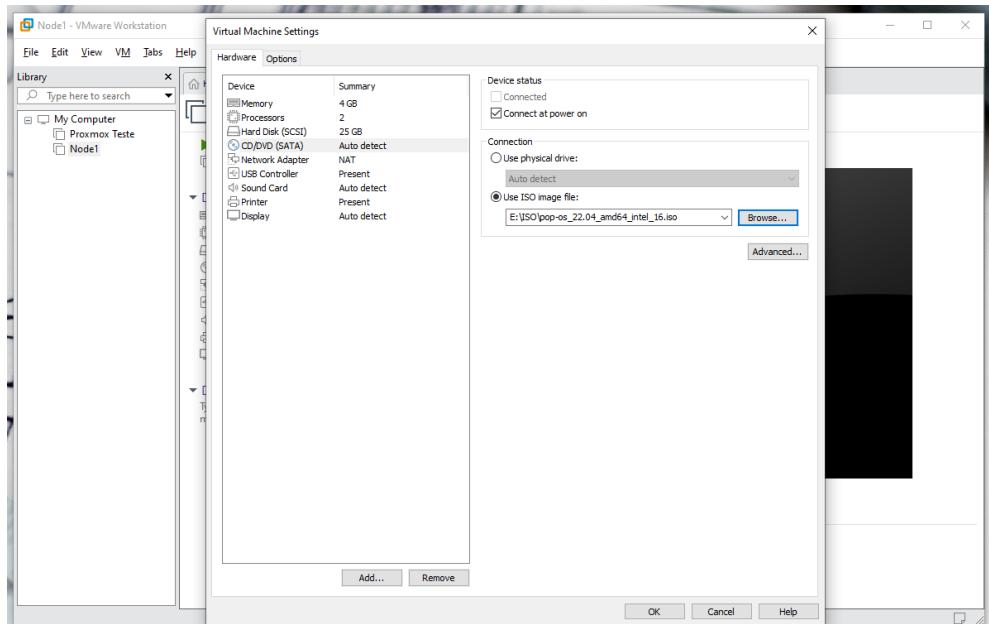
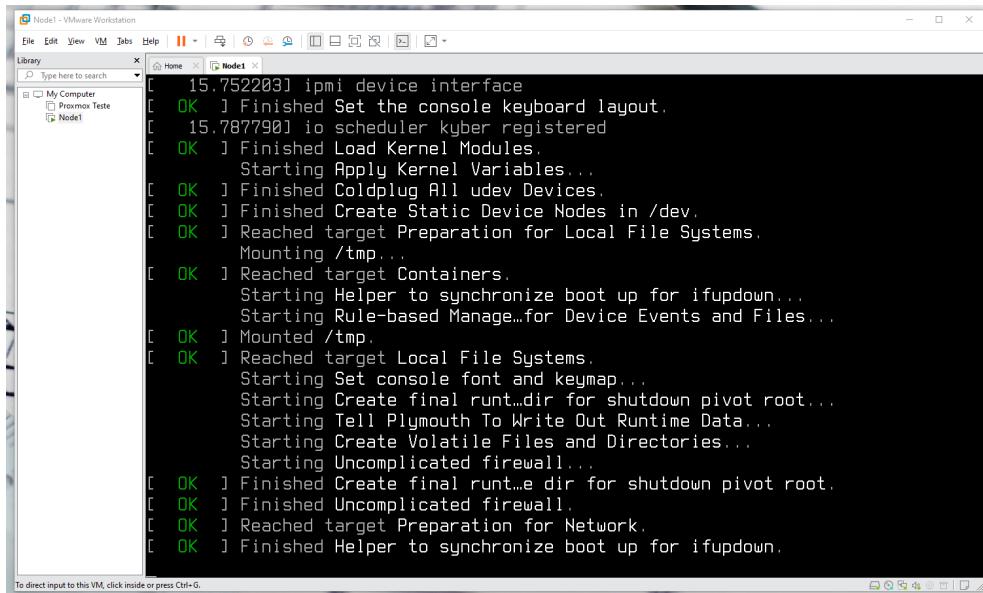


Figura 9.9: Inserir ISO

Passo 9: Depois de definir a ISO no passo anterior, é hora de dar "START" a VM para iniciar as instalação do Ubuntu.



```
[ 15.752203] ipmi device interface
[ OK ] Finished Set the console keyboard layout.
[ 15.787790] io scheduler kuber registered
[ OK ] Finished Load Kernel Modules.
          Starting Apply Kernel Variables...
[ OK ] Finished Coldplug All udev Devices.
[ OK ] Finished Create Static Device Nodes in /dev.
[ OK ] Reached target Preparation for Local File Systems.
          Mounting /tmp...
[ OK ] Reached target Containers.
          Starting Helper to synchronize boot up for ifupdown...
          Starting Rule-based Manage...for Device Events and Files...
[ OK ] Mounted /tmp.
[ OK ] Reached target Local File Systems.
          Starting Set console font and keymap...
          Starting Create final runt...dir for shutdown pivot root...
          Starting Tell Plymouth To Write Out Runtime Data...
          Starting Create Volatile Files and Directories...
          Starting Uncomplicated firewall...
[ OK ] Finished Create final runt...e dir for shutdown pivot root.
[ OK ] Finished Uncomplicated firewall.
[ OK ] Reached target Preparation for Network.
[ OK ] Finished Helper to synchronize boot up for ifupdown.
```

Figura 9.10: Inicialização VM

Passo 10: "Instalar o Ubuntu".



Figura 9.11: Instalar Ubuntu

Passo 11: Idioma teclado.

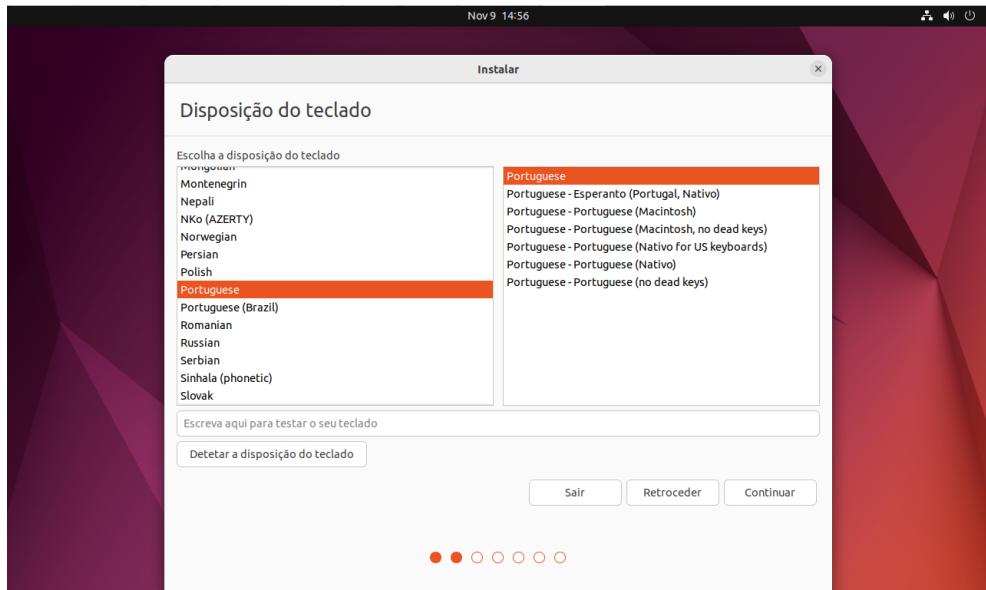


Figura 9.12: Idioma teclado

Passo 12: Atualizações (por defeito não alterei nada).

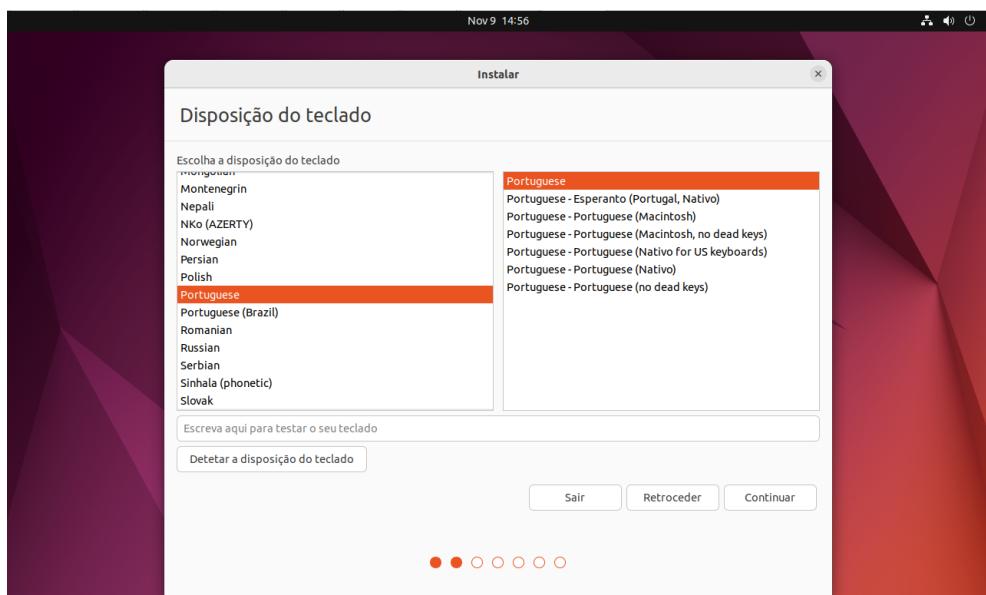


Figura 9.13: Atualizações

Passo 13: Escolher o tipo de instalação (optei por "Apagar disco e instalar o Ubuntu").

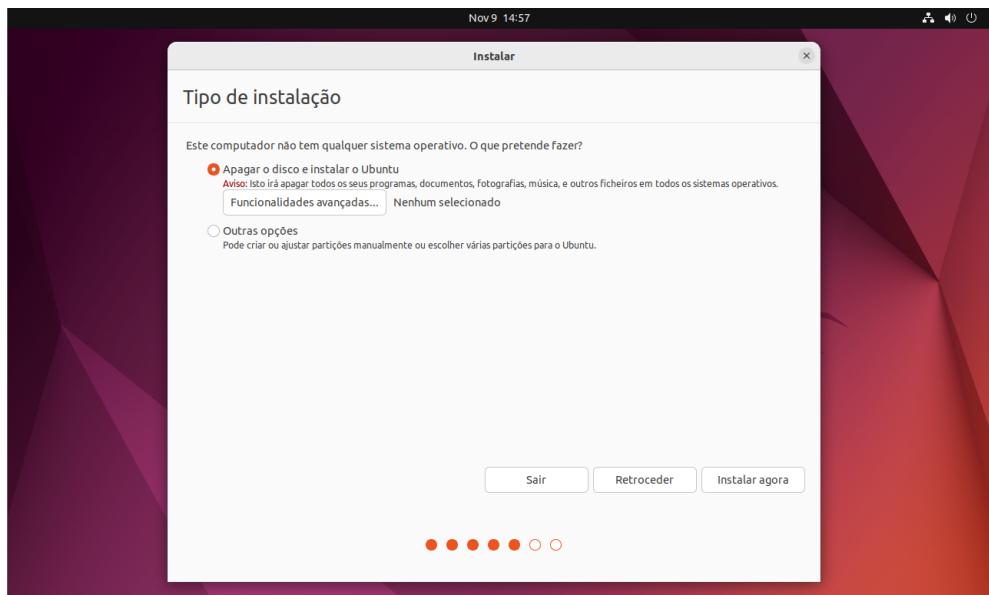


Figura 9.14: Tipo de instalação

Passo 14: Conformação das alterações dos discos, "Continuar".

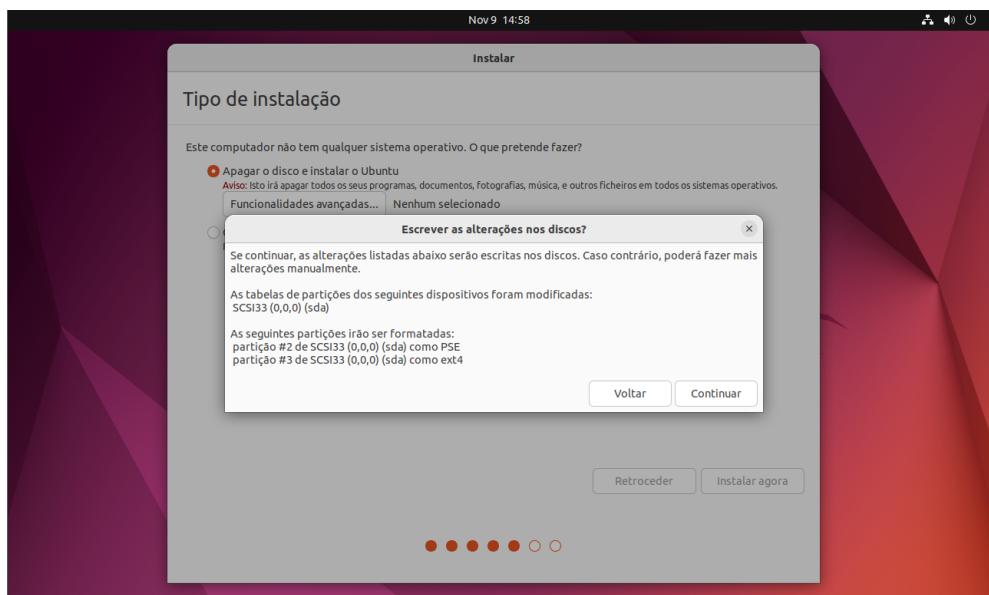


Figura 9.15: Confrmação

Passo 15: Escolha da localização.

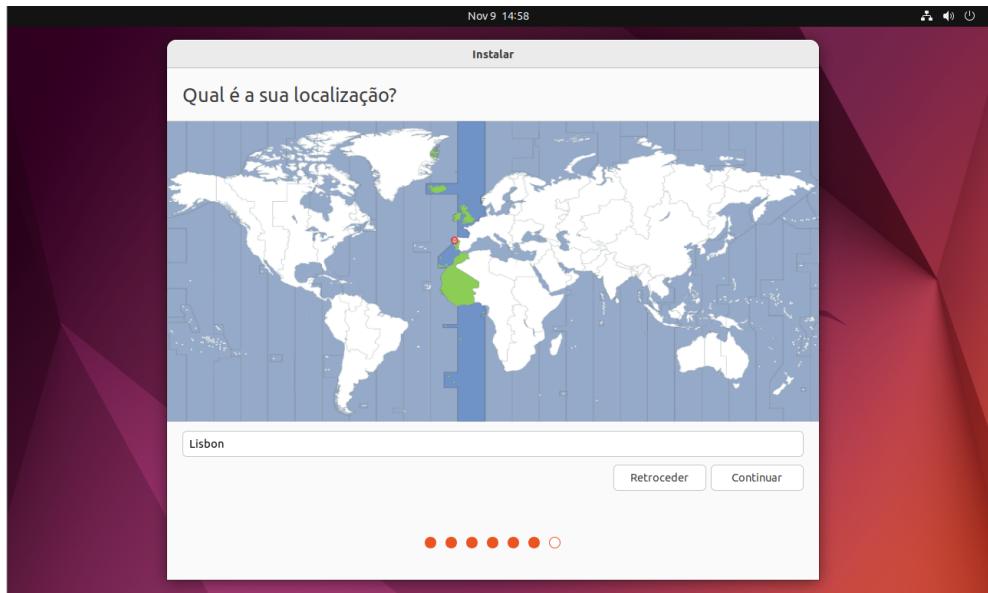


Figura 9.16: Localização

Passo 16: Processo de definir o nome, o nome da maquina, o nome de utilizador e password.

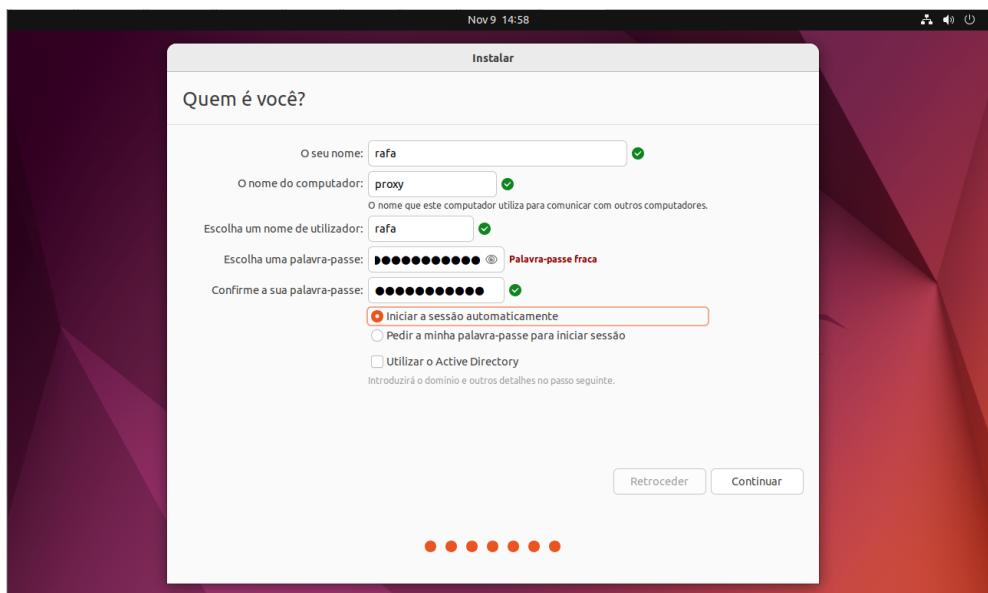


Figura 9.17: Informações pessoais

## Passo 17: Instalação do Ubuntu.

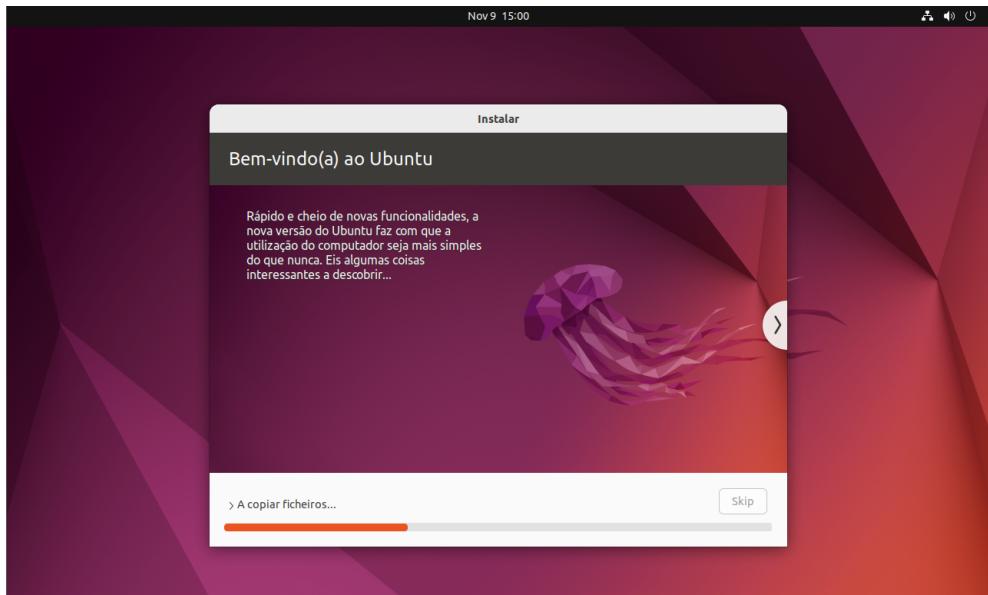


Figura 9.18: Instalação

Passo 18: Finalizada a instalação apenas basta reiniciar e entrar no *Sistema Operativo* (SO).

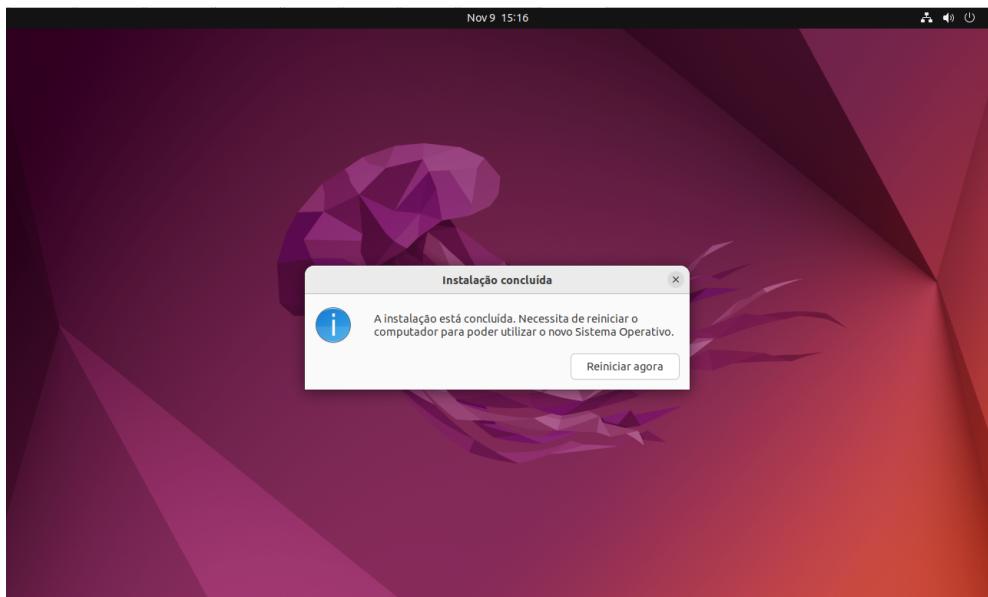


Figura 9.19: Reiniciar o Sistema Operativo

NOTA: nas propriedades da VM, em CD/DVD (SATA), desmarquei a opção de "Use ISO file" para "Use physical drive", para a VM não arrancar através da ISO e sim pelo sistema instalado no disco.

## 9.3 Portas Firewall

Numa primeira fase do projeto fui verificando os vários processos para a implementação desta Meta 4 e percebi que iria precisar de implementar regras para a firewall das VM para os vários serviços. E por isso inicialmente começo por adicionar algumas das portas necessárias:

### 9.3.1 Portas usadas:

- 3306: Serviço MySQL;
- 4567: Tráfego de Replicação do Galera Cluster;
- 4568: *Incremental State Transfer* (IST);
- 4444: *State Snapshot Transfer* (SST);
- 80: *Hyper Text Transfer Protocol* (HTTP);
- 22: *Secure Socket Shell* (SSH);

Podia adicionar ao arquivo `/etc/sysconfig/iptables` a configuração das `iptables`, mas optei por configurar no terminal da VM:

```
sudo ufw enable

sudo ufw allow 3306,4567,4568,4444,80,22,6033/tcp

sudo ufw allow 4567/udp

sudo ufw reload
```

## 9.4 Instalação MariaDB + Galera

NOTA IMPORTANTE: Todos os passos seguintes devem ser replicados pelas restantes VM, Node2 e Node3.

Para este estudo experimental optei por instalar o MariaDB uma vez que nas versões recentes já inclui o galera. Para começar, atualiza-se os repositórios locais do Ubuntu através do terminal:

```
sudo apt update
```

De seguida instalei o MariaDB server:

```
sudo apt install mariadb-server
```

Depois de concluída a instalação do MariaDB server podemos verificar qual as versões instaladas:

```
mariadb --version
```

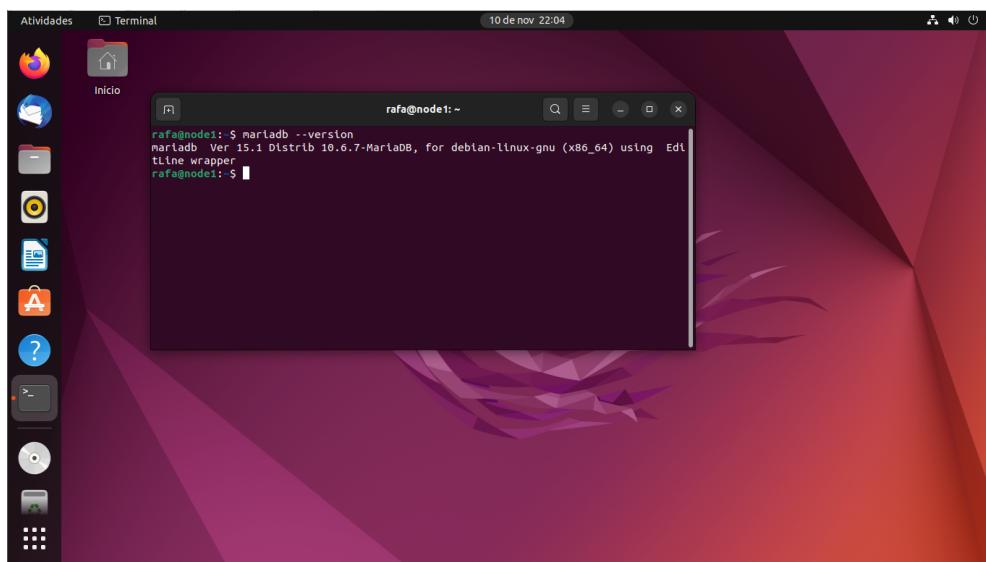


Figura 9.20: Vesão mariadb

De seguida iniciar o serviço mariadb e configurar a inicialização automática quando se liga a VM:

```
sudo systemctl start mariadb
```

```
sudo systemctl enable mariadb
```

Verificar o estado "status" do mariadb:

```
sudo systemctl status mariadb
```

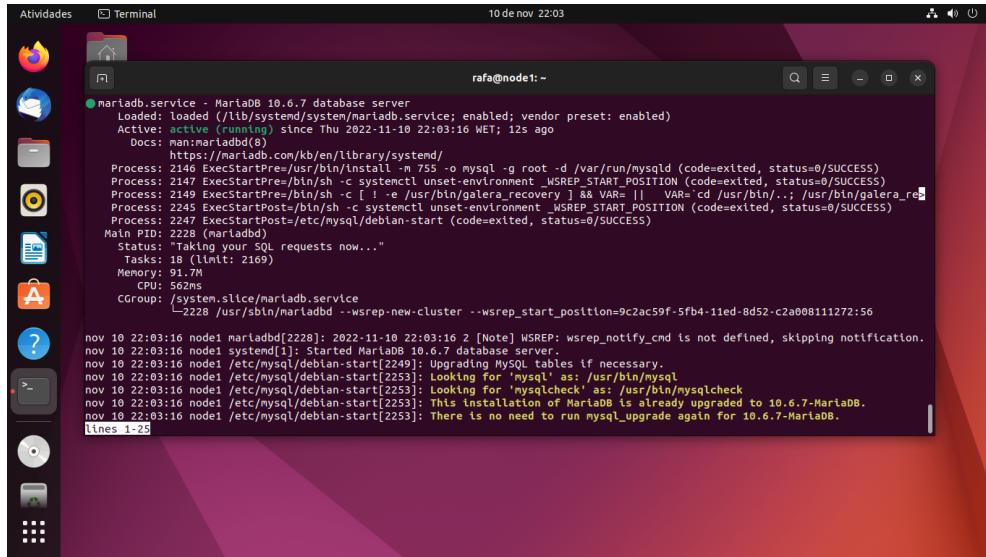


Figura 9.21: Status mariaDB

Próximo passo é garantir a segurança mínima do serviço e para isso foi usado o seguinte comando:

```
sudo mysql_secure_installation
```

Durante a nova configuração apenas é preciso aceitar todas as opções digitando, YES, e introduzir uma password.

De seguida, criamos uma nova conta de utilizador no servidor de BD com autenticação e posteriormente atribuir privilégios administrativos a esse utilizador. Para isso entramos da seguinte forma:

```
sudo mariadb -u root -p
```

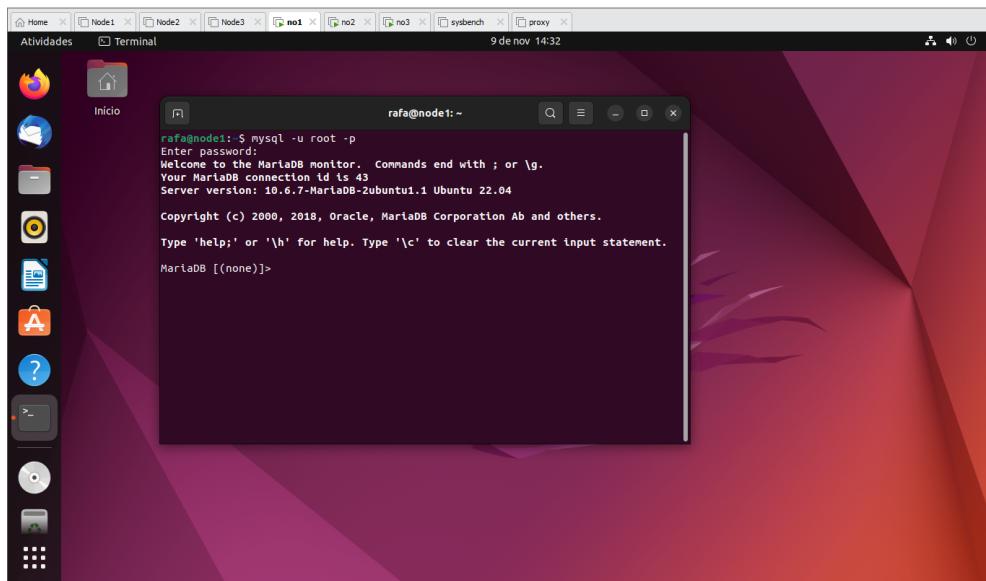


Figura 9.22: Login root Mysql

Criar um utilizador com privilégios altos:

```
CREATE USER 'admin_user'@'localhost' IDENTIFIED BY 'secret_password';
```

NOTA: "admin\_user" será o nome do utilizador, e 'secret\_password' a password.

Privilégios:

```
GRANT ALL PRIVILEGES ON *.* TO 'admin_user'@'localhost';
```

NOTA: O \*.\* garante que o utilizador tem permissões para qualquer alteração ou inserção na base de dados.

Aplicar as alterações e de seguida sair.

```
FLUSH PRIVILEGES;
```

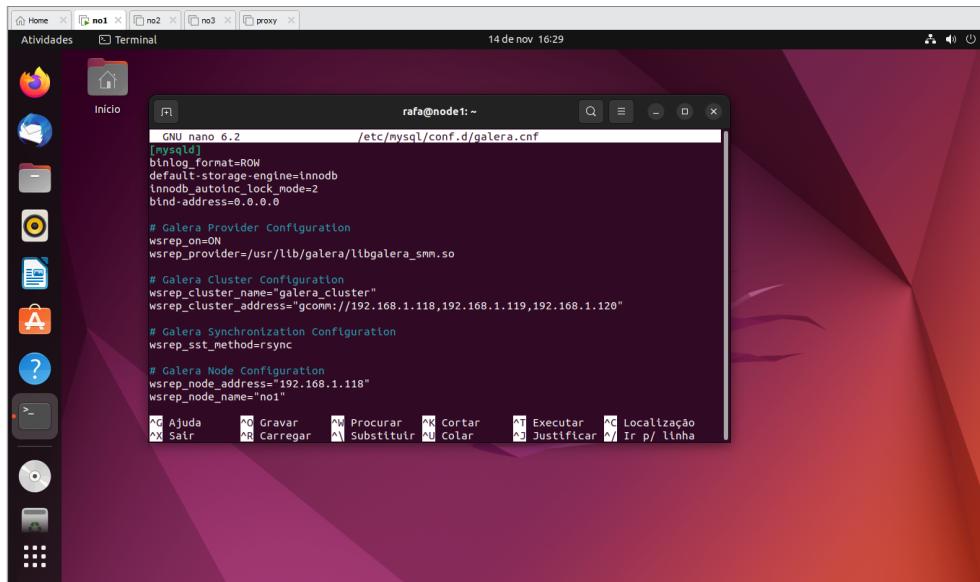
```
EXIT;
```

## 9.5 Configurar Galera Cluster

Para que cada nó possa comunicar entre si precisamos de criar um arquivo de configuração Galera (repetir a criação do ficheiro galera nos vários nós):

```
sudo nano /etc/mysql/conf.d/galera.cnf
```

Basta criar a seguinte configuração:



```
GNU nano 6.2 /etc/mysql/conf.d/galera.cnf
[mysql]
innodb_format=ROW
default_storage_engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

# Galera Cluster Configuration
wsrep_cluster_name='galera_cluster'
wsrep_cluster_address='gcomm://192.168.1.118,192.168.1.119,192.168.1.120'

# Galera Synchronization Configuration
wsrep_sst_method=rsync

# Galera Node Configuration
wsrep_node_address='192.168.1.118'
wsrep_node_name='no1'
```

Figura 9.23: Ficheiro Configuração Galera

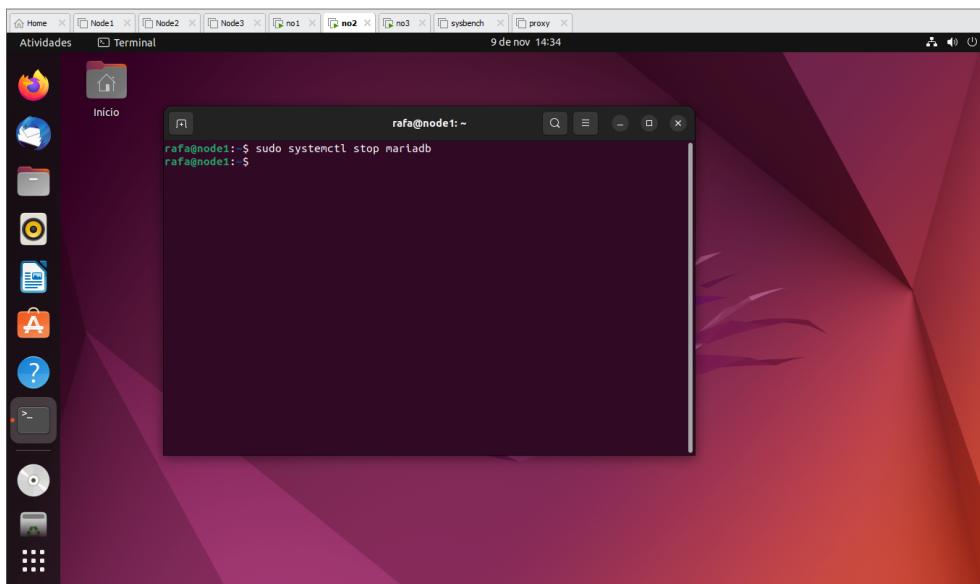
Nos restantes nós apenas se altera o IP do "wsrep\_node\_address" e "wsrep\_node\_name".

## 9.6 Iniciar o Galera Cluster

Depois de concluir o passo anterior os nós estão preparados para comunicar uns com os outros.

De seguida, precisamos de interromper o serviço MariaDB em TODOS os nós.

```
sudo systemctl stop mariadb
```



```
rafa@node1: ~
rafa@node1: $ sudo systemctl stop mariadb
rafa@node1: $
```

Figura 9.24: STOP MariaDB

No primeiro nó, (de salientar que este comando é APENAS inserido no primeiro nó), o cluster MariaDB Galera é iniciado com o seguinte comando:

```
galera_new_cluster
```

Agora, para verificar o tamanho do cluster entra-se como *root* e utiliza-se:

```
mysql -u root -p  
show status like 'wsrep_cluster_size';
```

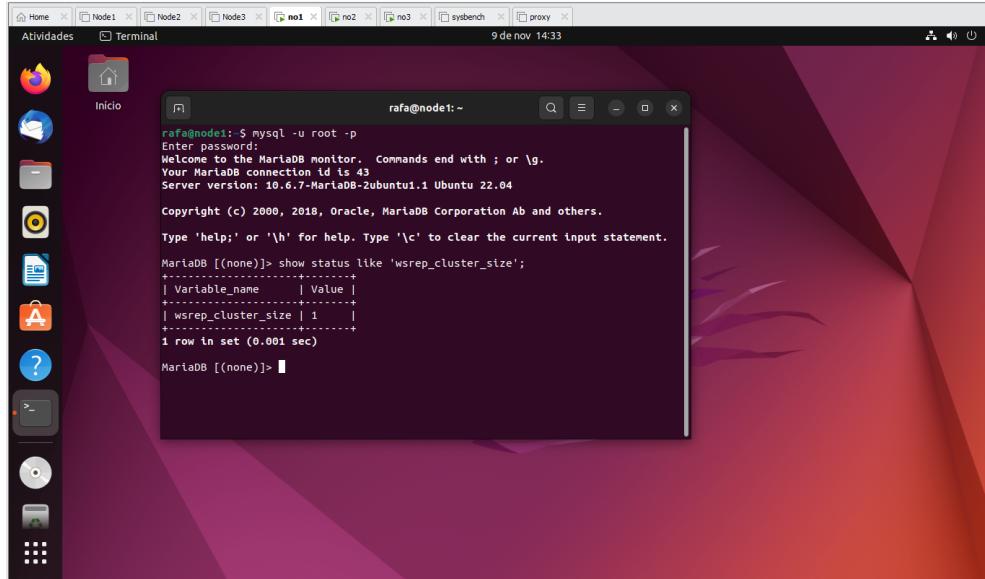


Figura 9.25: Tamanho do cluster no nó 1

Ainda podemos verificar se o nó 1 tem conectividade com os restantes nós:

```
SHOW GLOBAL STATUS LIKE 'wsrep_connected';
```

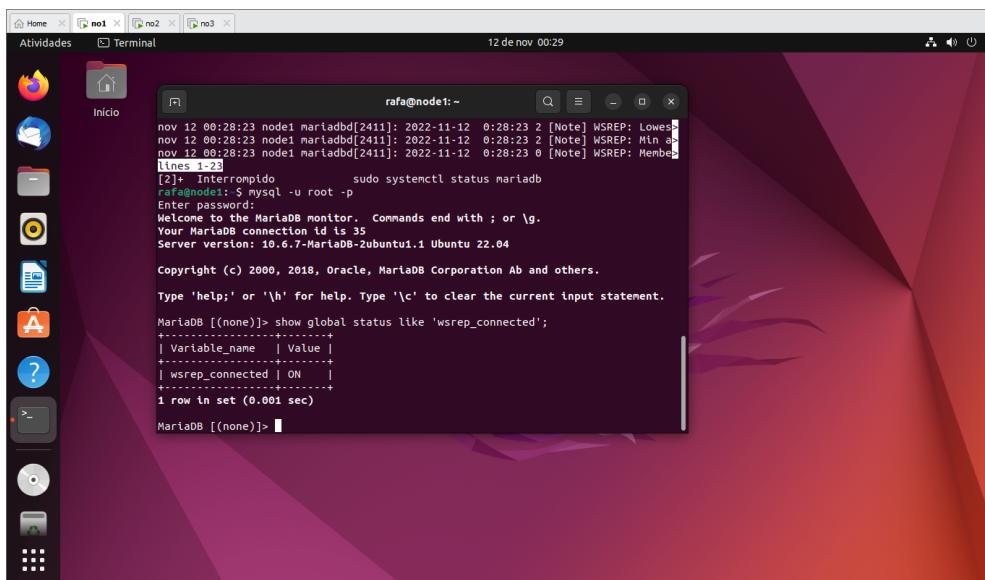


Figura 9.26: Conectividade entre nós do cluster

No segundo nó, inicia-se o serviço MariaDB:

```
systemctl start mariadb
```

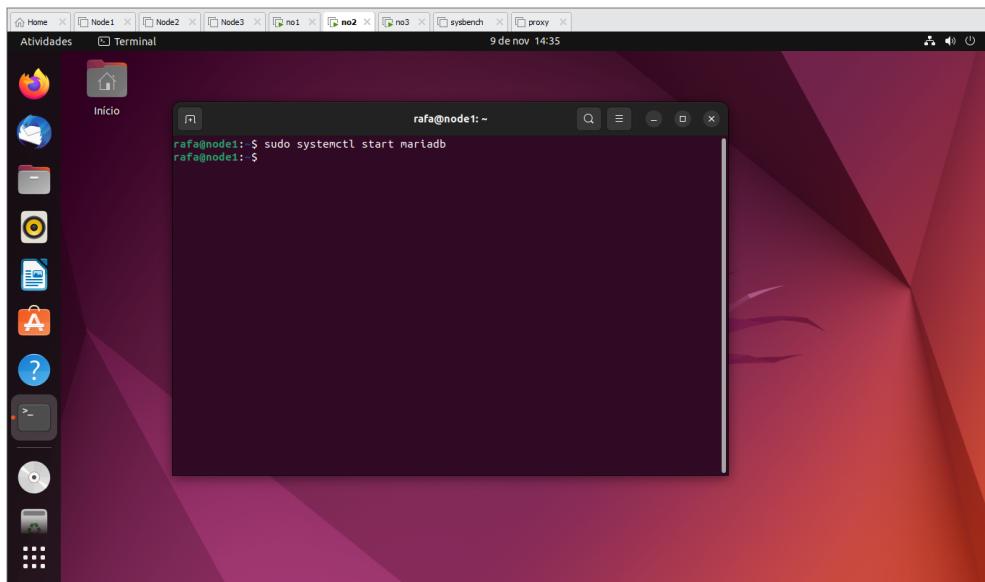


Figura 9.27: START Cluster N o 2

Repeite-se o procedimento no segundo n o para verificar o tamanho do cluster:

```
mysql -u root -p  
show status like 'wsrep_cluster_size';
```

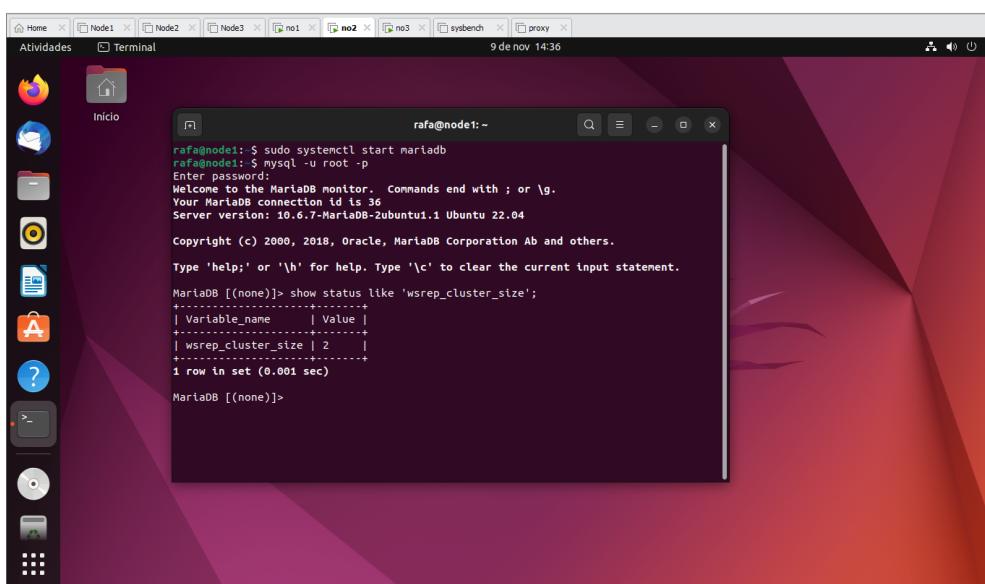


Figura 9.28: Tamanho do cluster no n o 2

Entretanto se verificar-mos o tamanho do cluster no nó 1, verificamos que o nó 1 e nó 2 já pertencem ao mesmo cluster galera:

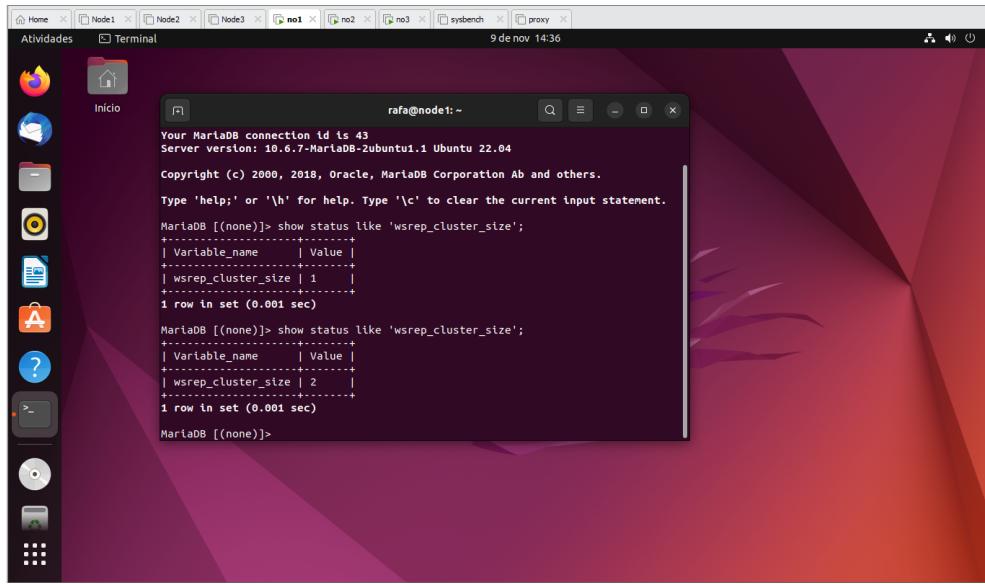


Figura 9.29: Tamanho do cluster no nó 1

No terceiro nó, inicia-se o serviço MariaDB:

```
sudo systemctl start mariadb
```

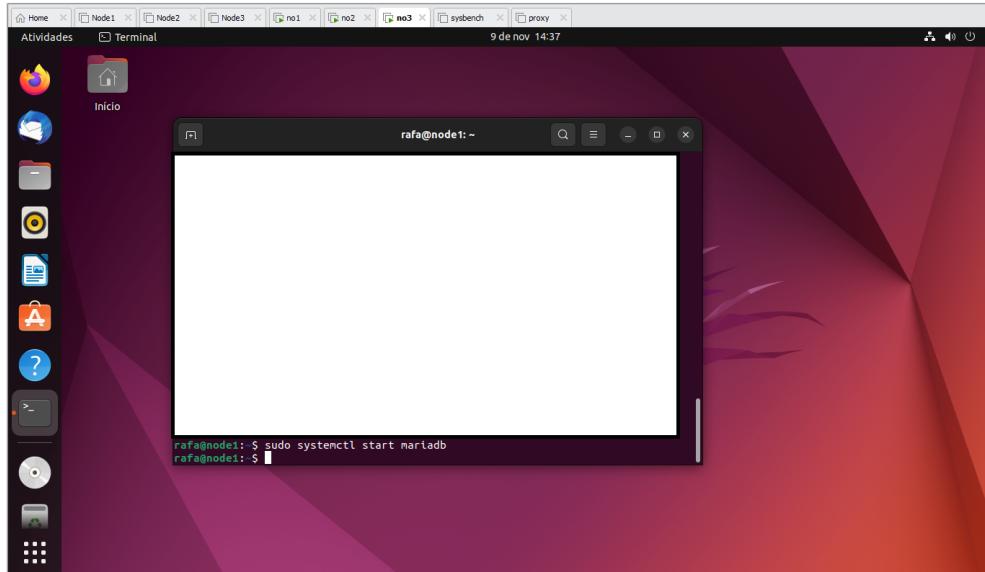
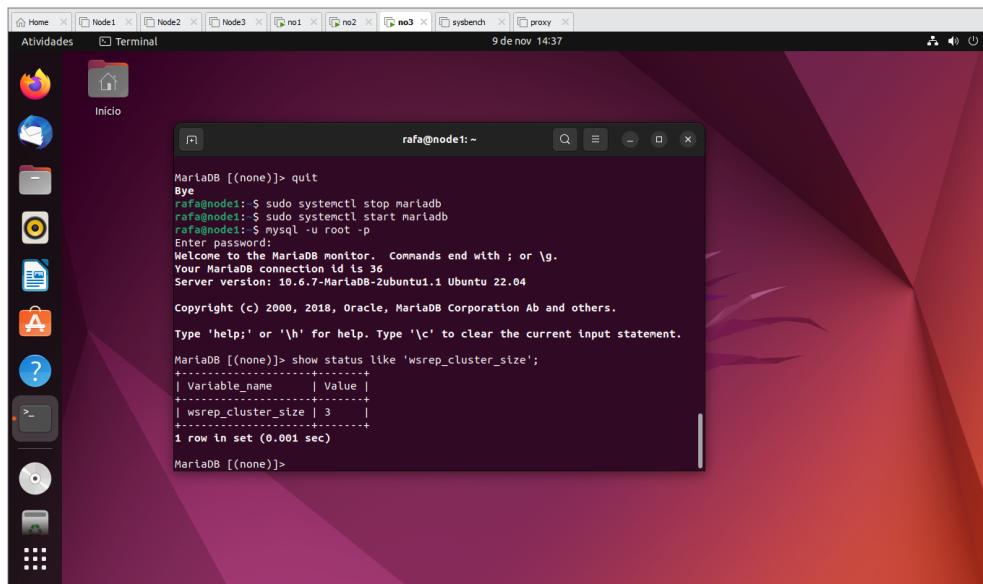


Figura 9.30: START Cluster nº 3

Repeite-se o procedimento no segundo nó para verificar o tamanho do cluster:

```
mysql -u root -p  
show status like 'wsrep_cluster_size';
```

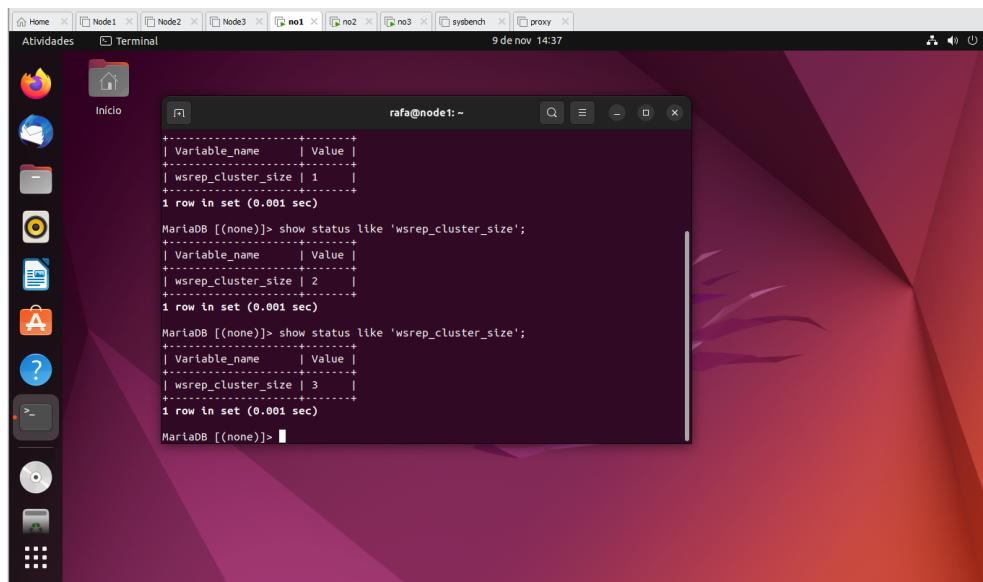


A screenshot of a Linux desktop environment showing a terminal window titled "rafa@node1: ~". The terminal displays the following MySQL session:

```
MariaDB [(none)]> quit  
Bye  
rafa@node1: $ sudo systemctl stop mariadb  
rafa@node1: $ sudo systemctl start mariadb  
rafa@node1: $ mysql -u root -p  
Enter password:  
Welcome to the MariaDB monitor. Commands end with ; or \g.  
Your MariaDB connection id is 36  
Server version: 10.6.7-MariaDB-2ubuntu1.1 Ubuntu 22.04  
  
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
MariaDB [(none)]> show status like 'wsrep_cluster_size';  
+-----+-----+  
| Variable_name | Value |  
+-----+-----+  
| wsrep_cluster_size | 3 |  
+-----+-----+  
1 row in set (0.001 sec)  
  
MariaDB [(none)]>
```

Figura 9.31: Tamanho do cluster no nó 3

Entretanto se se verificar novamente o tamanho do cluster no nó 1 verificamos que todos os nós já pertencem ao mesmo cluster galera:



A screenshot of a Linux desktop environment showing a terminal window titled "rafa@node1: ~". The terminal displays the following MySQL session:

```
+-----+-----+  
| Variable_name | Value |  
+-----+-----+  
| wsrep_cluster_size | 1 |  
+-----+-----+  
1 row in set (0.001 sec)  
  
MariaDB [(none)]> show status like 'wsrep_cluster_size';  
+-----+-----+  
| Variable_name | Value |  
+-----+-----+  
| wsrep_cluster_size | 2 |  
+-----+-----+  
1 row in set (0.001 sec)  
  
MariaDB [(none)]> show status like 'wsrep_cluster_size';  
+-----+-----+  
| Variable_name | Value |  
+-----+-----+  
| wsrep_cluster_size | 3 |  
+-----+-----+  
1 row in set (0.001 sec)  
  
MariaDB [(none)]>
```

Figura 9.32: Tamanho do cluster no nó 1

## 9.7 Replicação no Cluster

Para testar a replicação com o cluster concluído (3 nós), criei uma base de dados no nó 1 para verificar se a mesma base de dados era replicada pelos restantes nós.

Comecei por verificar as bases de dados existentes com o comando:

```
SHOW DATABASES;
```

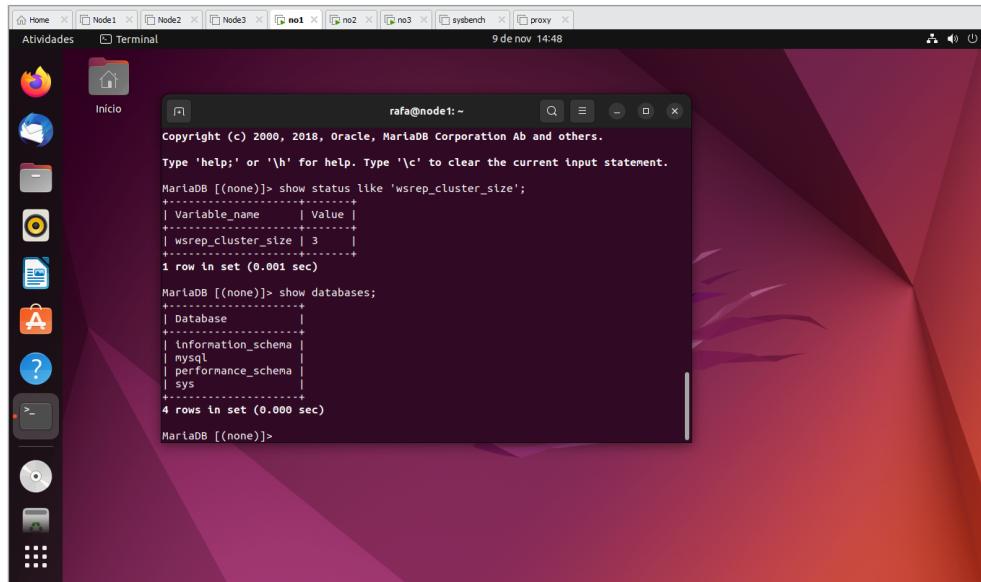


Figura 9.33: Base de Dados existentes

De seguida, criei a base de dados e pude confirmar novamente com o comando *SHOW DATABASES*; que foi criada:

```
CREATE DATABASE dbtest1;
```

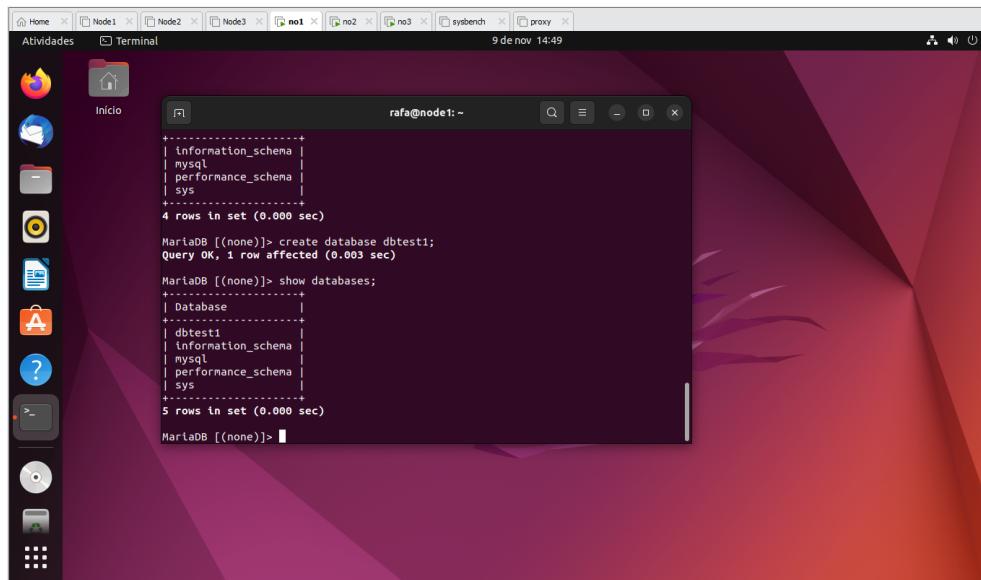
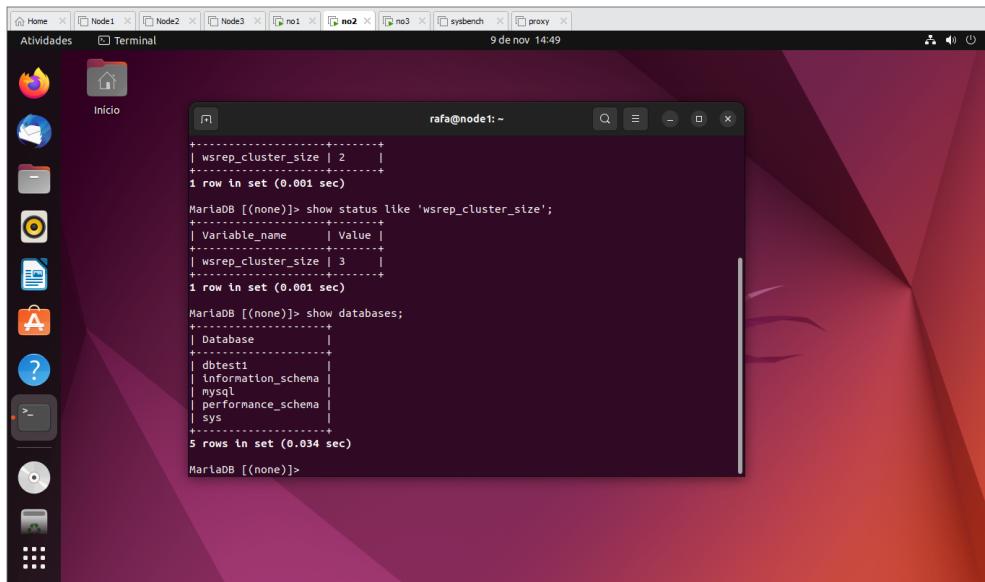


Figura 9.34: Criação da Base de dados

Passando para o nó 2 e 3 para verificar se a BD criada já existe no cluster basta fazer *login* no MariaDB como *root* e verificar com os comandos acima já mencionados:

```
SHOW DATABASES;
```



A screenshot of a Linux desktop environment showing a terminal window titled "rafa@node1: ~". The terminal displays the following MySQL command output:

```
+ wsrep_cluster_size | 2 |
| wsrep_cluster_size | 3 |
1 row in set (0.001 sec)

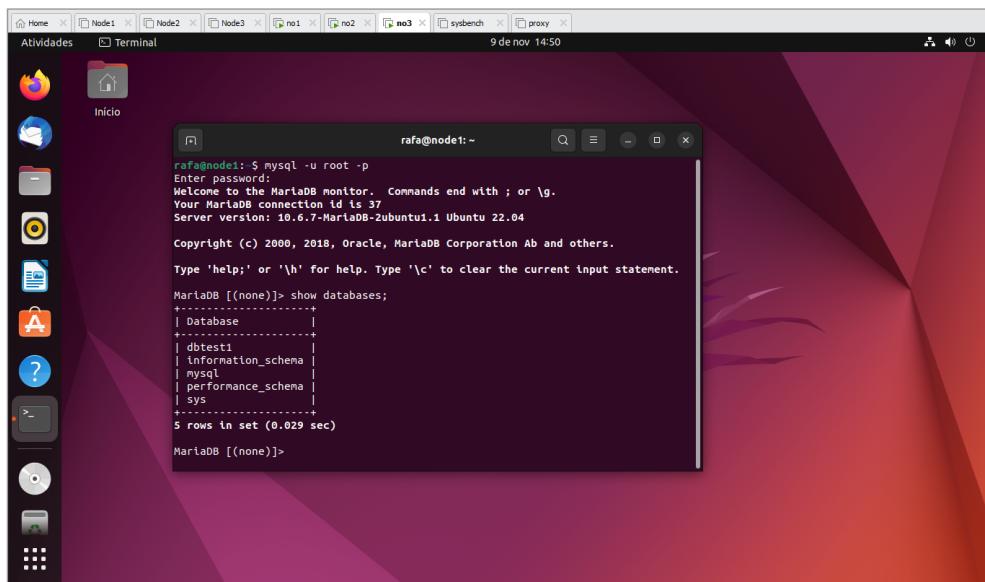
MariaDB [(none)]> show status like 'wsrep_cluster_size';
+ Variable_name      | value |
| wsrep_cluster_size | 3 |
1 row in set (0.001 sec)

MariaDB [(none)]> show databases;
+ Database           |
| dbtest1            |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
5 rows in set (0.034 sec)

MariaDB [(none)]>
```

Figura 9.35: Listagem BD

Podemos verificar que a BD *dbtest1* inserida no nó 1 foi replicada de imediato para o nó 2 e para o 3:



A screenshot of a Linux desktop environment showing a terminal window titled "rafa@node1: ~". The terminal displays the following MySQL command output:

```
rafa@node1: $ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 37
Server version: 10.6.7-MariaDB-2ubuntu1.1 Ubuntu 22.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

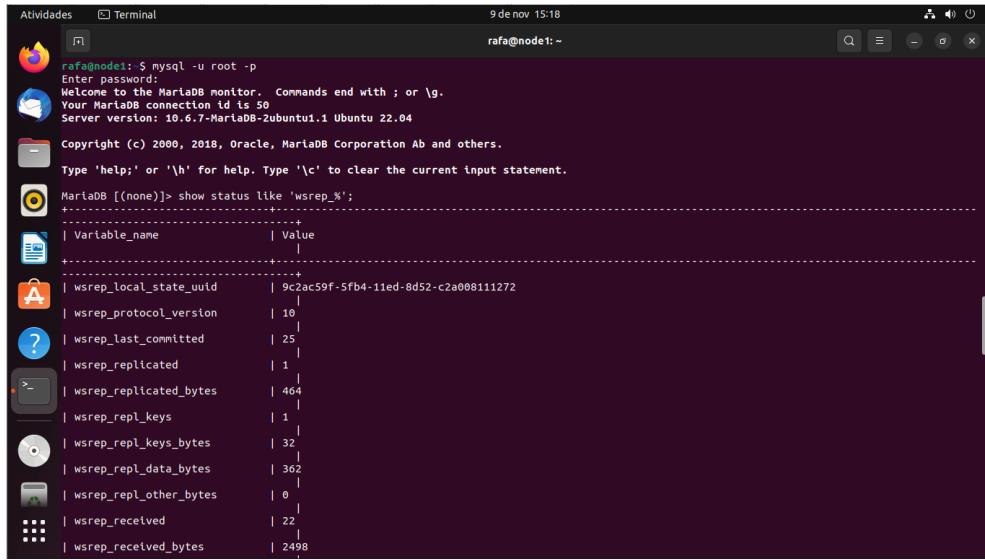
MariaDB [(none)]> show databases;
+ Database           |
| dbtest1            |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
5 rows in set (0.029 sec)

MariaDB [(none)]>
```

Figura 9.36: Listagem BD

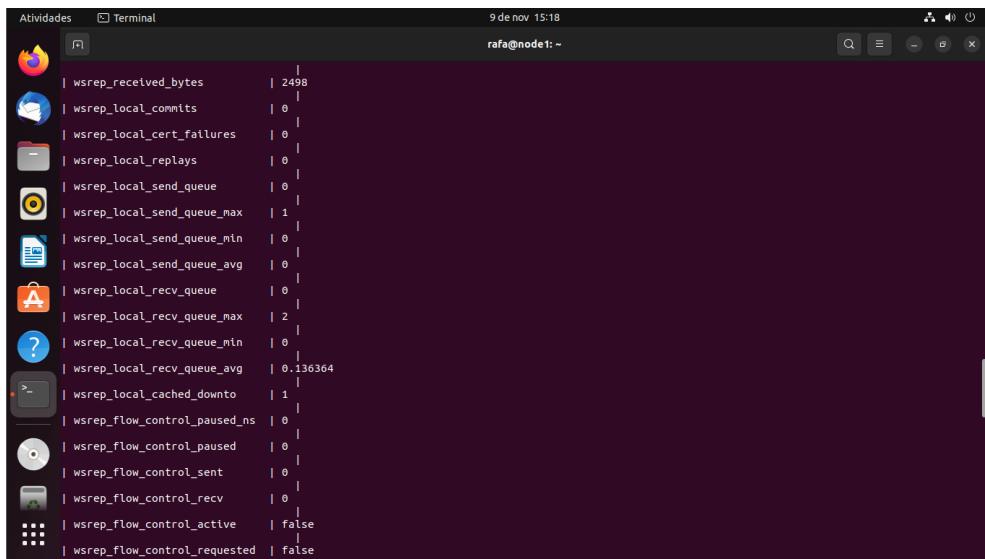
Por fim, ainda pude verificar as várias informações/definições gravadas no cluster:

```
SHOW STATUS LIKE 'WSREP_\%';
```



```
Atividades Terminal 9 de nov 15:18 rafa@node1:~  
rafa@node1: $ mysql -u root -p  
Enter password:  
Welcome to the MariaDB monitor. Commands end with ; or \g.  
Your MariaDB connection id is 50  
Server version: 10.6.7-MariaDB-2ubuntu1.1 Ubuntu 22.04  
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
MariaDB [(none)]> show status like 'wsrep_\%';  
+-----+-----+  
| Variable_name | Value |  
+-----+-----+  
| wsrep_local_state_uuid | 9c2ac59f-5fb4-11ed-8d52-c2a008111272 |  
| wsrep_protocol_version | 10 |  
| wsrep_last_committed | 25 |  
| wsrep_replicated | 1 |  
| wsrep_replicated_bytes | 464 |  
| wsrep_repl_keys | 1 |  
| wsrep_repl_keys_bytes | 32 |  
| wsrep_repl_data_bytes | 362 |  
| wsrep_repl_other_bytes | 0 |  
| wsrep_received | 22 |  
| wsrep_received_bytes | 2498 |
```

Figura 9.37: Informações do Cluster



```
Atividades Terminal 9 de nov 15:18 rafa@node1:~  
rafa@node1: $ mysql -u root -p  
Enter password:  
Welcome to the MariaDB monitor. Commands end with ; or \g.  
Your MariaDB connection id is 50  
Server version: 10.6.7-MariaDB-2ubuntu1.1 Ubuntu 22.04  
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
MariaDB [(none)]> show status like 'wsrep_\%';  
+-----+-----+  
| Variable_name | Value |  
+-----+-----+  
| wsrep_received_bytes | 2498 |  
| wsrep_local_commits | 0 |  
| wsrep_local_cert_failures | 0 |  
| wsrep_local_replays | 0 |  
| wsrep_local_send_queue | 0 |  
| wsrep_local_send_queue_max | 1 |  
| wsrep_local_send_queue_min | 0 |  
| wsrep_local_send_queue_avg | 0 |  
| wsrep_local_recv_queue | 0 |  
| wsrep_local_recv_queue_max | 2 |  
| wsrep_local_recv_queue_min | 0 |  
| wsrep_local_recv_queue_avg | 0.136364 |  
| wsrep_local_cached_downto | 1 |  
| wsrep_flow_control_paused_ns | 0 |  
| wsrep_flow_control_paused | 0 |  
| wsrep_flow_control_sent | 0 |  
| wsrep_flow_control_recv | 0 |  
| wsrep_flow_control_active | false |  
| wsrep_flow_control_requested | false |
```

Figura 9.38: Informações do Cluster

```

Atividades Terminal 9 de nov 15:19
rafa@node1:~ 

| wsrep_flow_control_requested | false
| wsrep_cert_deps_distance | 1
| wsrep_apply_oooe | 0
| wsrep_apply_oool | 0
| wsrep_apply_window | 1
| wsrep_apply_waits | 0
| wsrep_commit_ooee | 0
| wsrep_commit_oool | 0
| wsrep_commit_window | 1
| wsrep_local_state | 4
| wsrep_local_state_comment | Synced
| wsrep_cert_index_size | 1
| wsrep_causal_reads | 0
| wsrep_cert_interval | 11
| wsrep_open_transactions | 0
| wsrep_open_connections | 0
| wsrep_incoming_addresses | AUTO,AUTO,AUTO
| wsrep_cluster_weight | 3
| wsrep_desync_count | 0

```

Figura 9.39: Informações do Cluster

```

Atividades Terminal 9 de nov 15:19
rafa@node1:~ 

| wsrep_desync_count | 0
| wsrep_evs_delayed | 
| wsrep_evs_evict_list | 
| wsrep_evs_repl_latency | 0/0/0/0/0
| wsrep_evs_state | OPERATIONAL
| wsrep_gcomm_uuid | 6e8e4549-602f-11ed-b2cf-f6e5ab9652b
| wsrep_gmcast_segment | 0
| wsrep_applier_thread_count | 1
| wsrep_cluster_capabilities | 
| wsrep_cluster_conf_id | 10
| wsrep_cluster_size | 3
| wsrep_cluster_state_uuid | 9c2ac59f-5fb4-11ed-8d52-c2a008111272
| wsrep_cluster_status | Primary
| wsrep_connected | ON
| wsrep_local_bf_aborts | 0
| wsrep_local_index | 0
| wsrep_provider_capabilities | :MULTI_MASTER:CERTIFICATION:PARALLEL_APPLYING:TRX_REPLAY:ISOLATION:PAUSE:CAUSAL_READS:INCREMENTAL_WRITESET:UNORDERED:PREORDERED:STREAMING:NBO:
| wsrep_provider_name | Galera
| wsrep_provider_vendor | Codership Oy <info@codership.com>

```

Figura 9.40: Informações do Cluster

```

Atividades Terminal 9 de nov 15:19
rafa@node1:~ 

| wsrep_applier_thread_count | 1
| wsrep_cluster_capabilities | 
| wsrep_cluster_conf_id | 10
| wsrep_cluster_size | 3
| wsrep_cluster_state_uuid | 9c2ac59f-5fb4-11ed-8d52-c2a008111272
| wsrep_cluster_status | Primary
| wsrep_connected | ON
| wsrep_local_bf_aborts | 0
| wsrep_local_index | 0
| wsrep_provider_capabilities | :MULTI_MASTER:CERTIFICATION:PARALLEL_APPLYING:TRX_REPLAY:ISOLATION:PAUSE:CAUSAL_READS:INCREMENTAL_WRITESET:UNORDERED:PREORDERED:STREAMING:NBO:
| wsrep_provider_name | Galera
| wsrep_provider_vendor | Codership Oy <info@codership.com>
| wsrep_provider_version | 4.9(rcece3baz)
| wsrep_ready | ON
| wsrep_rollbacker_thread_count | 1
| wsrep_thread_count | 2
+-----+
69 rows in set (0.001 sec)

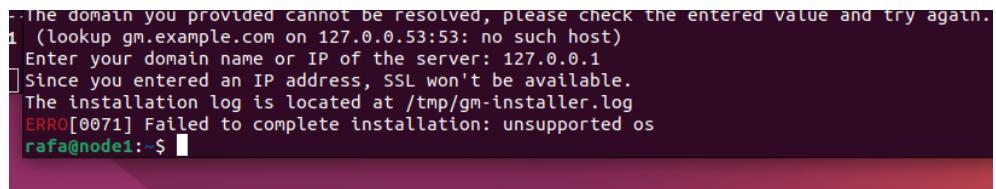
MariaDB [(none)]> 

```

Figura 9.41: Informações do Cluster

## 9.8 Instalação e Configuração Galera Manager

Para esta componente da meta 4, não consegui explorar devido ao fato de estar a usar o Ubuntu 22.04 e ser incompatível com o Galera Manager. Como prova fica anexado na figura abaixo.



```
The domain you provided cannot be resolved, please check the entered value and try again.  
1 (lookup gm.example.com on 127.0.0.53:53: no such host)  
Enter your domain name or IP of the server: 127.0.0.1  
Since you entered an IP address, SSL won't be available.  
The installation log is located at /tmp/gm-installer.log  
ERRO[0071] Failed to complete installation: unsupported os  
rafa@node1:~$
```

Figura 9.42: Erro Galera Manager

## 9.9 Proxy

Para executar este novo tema denominado como Proxy, foi adicionado uma nova VM com as mesmas características dos nós acima referidos e procedi à instalação do ProxySQL.

Antes de começar a avançar com o proxySQL como já sabia que iria ser preciso definir o *bind-address* de cada nó com o *Internet Protocol* (IP) atribuído pela placa de rede comecei por fazer essa atribuição:

```
sudo nano /etc/mysql/mariadb.conf.d/50-server.cnf
```

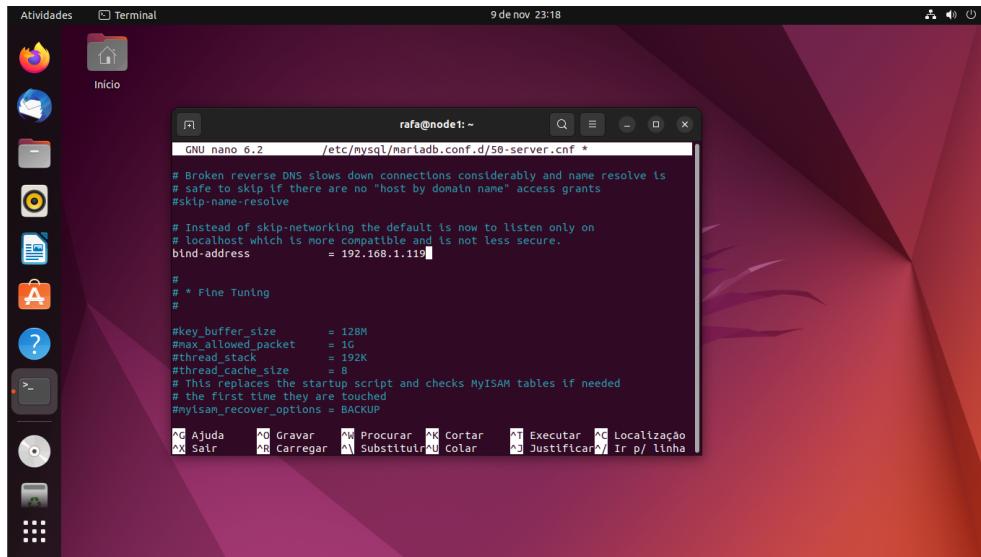


Figura 9.43: Bind-Address Nós

Passo 1: Instalação ProxySQL

```
wget https://github.com/sysown/proxysql/releases/download/v2.4.2/proxysql_2.4.2-ubuntu20_amd64.deb
```

```
dpkg -i proxysql_2.4.2-ubuntu20_amd64.deb
```

```
sudo apt install mysql-client
```

Iniciar o serviço ProxySQL:

```
sudo systemctl start proxysql
```

Habilitar o serviço ProxySQL no sistema:

```
sudo systemctl enable --now proxysql
```

Verificar o estado do serviço ProxySQL:

```
sudo systemctl status proxysql
```

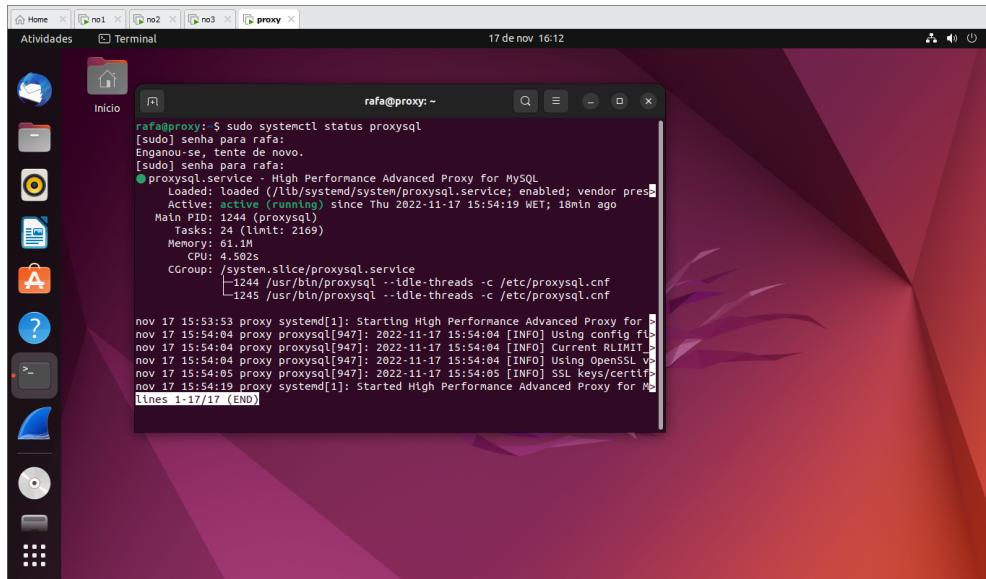


Figura 9.44: STATUS ProxySQL

## Passo 2: Configurar o ProxySQL

Para me conectar a área administrativa do ProxySQL foi preciso criar um cliente com a respetiva password:

```
mysql -u admin -psenhasegura -h 127.0.0.1 -P6032 --prompt='Admin> '
```

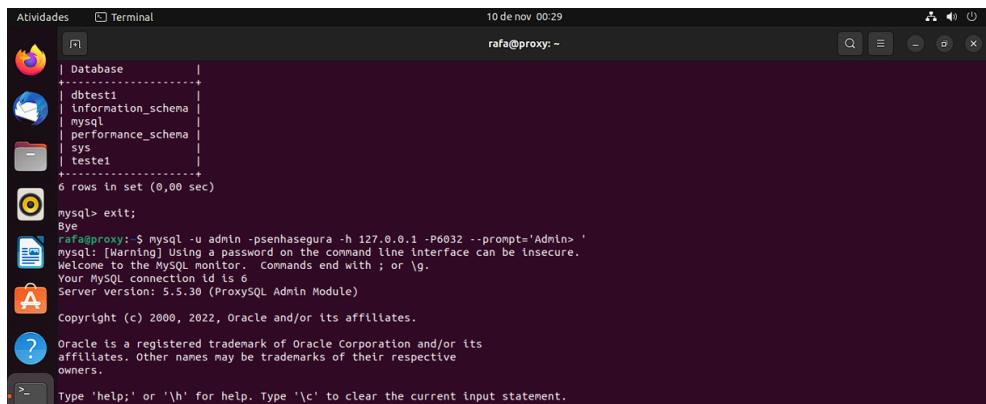


Figura 9.45: User Admin

Depois de criar o cliente, por questões de segurança, alteramos a palavra-passe original:

```
UPDATE global_variables SET variable_value='admin:senhasegura'
```

```
WHERE variable_name='admin-admin_credentials';
```

A consulta acima usada foi gravada apenas em memória e para isso precisamos de copiar a configuração para o disco executando os seguintes comandos:

```
LOAD ADMIN VARIABLES TO RUNTIME;
```

```
SAVE ADMIN VARIABLES TO DISK;
```

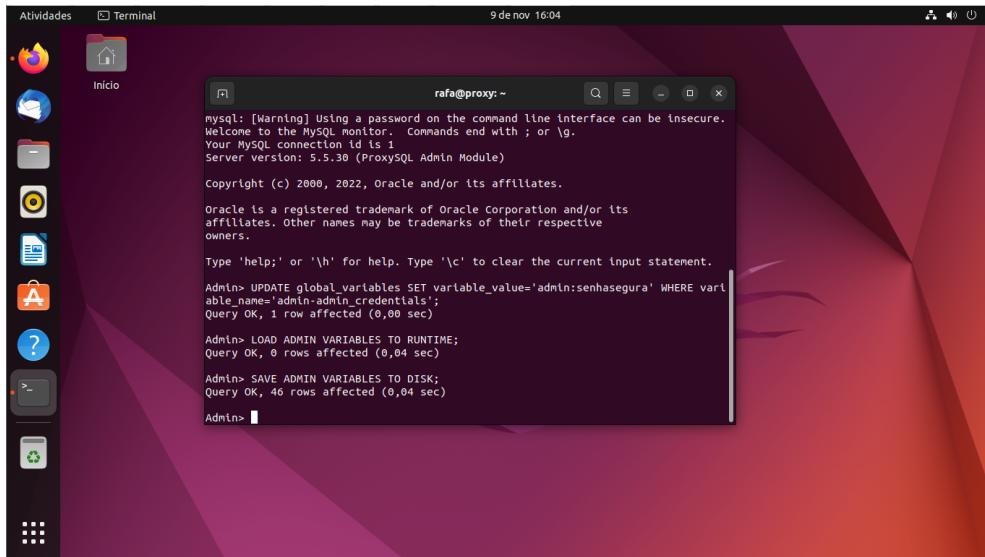


Figura 9.46: Criação Cliente

A configuração ProxySQL consiste em três camadas:

- Memory (memória);
- Disk (disco);
- Runtime (tempo de execução).

### Passo 3: Configurar Monitorização Galera Cluster

O ProxySQL precisa de comunicar com os nós do cluster para conhecer o seu estado. Isto significa que o ProxySQL tem de se ligar aos nós através de um utilizador.

Para isso foi criado um utilizador num dos nós e esse utilizador será replicado automaticamente através do cluster.

Nó 1:

```
CREATE USER 'monitor'@'%' IDENTIFIED BY 'senhasegura';  
flush privileges;
```

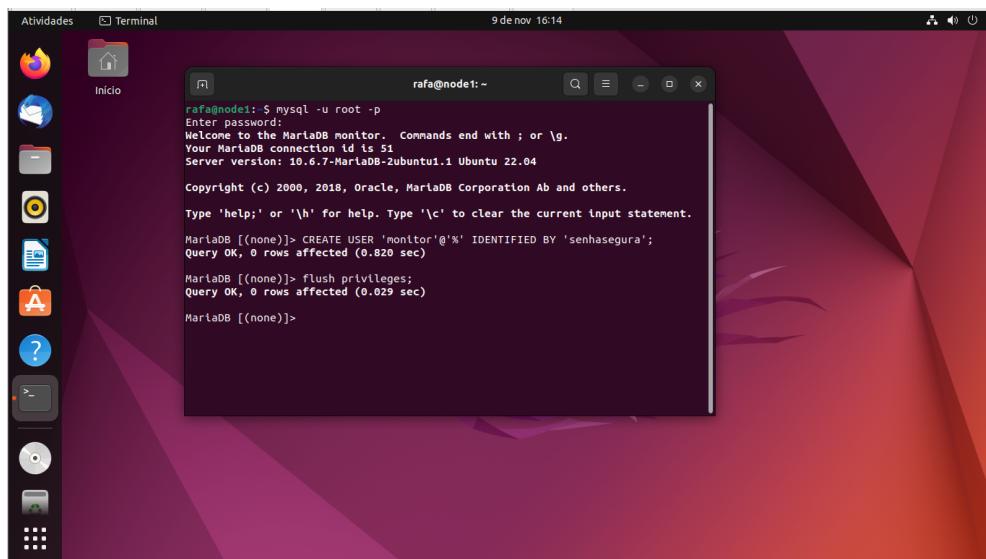


Figura 9.47: Utilizador ProxySQL-MariaDB

### Passo 4: Configurar Monitorização ProxySQL

Passamos à fase da criação do administrador do ProxySQL para monitorizar os nós.

NOTA: As credenciais do utilizador que configurámos no passo anterior têm de coincidir.

```
UPDATE global_variables SET variable_value='monitor'  
WHERE variable_name='mysql-monitor_username';
```

```
UPDATE global_variables SET variable_value='senhasegura'  
WHERE variable_name='mysql-monitor_password';
```

Definir parâmetros de intervalos na monitorização:

```
UPDATE global_variables SET variable_value='2000' WHERE variable_name
IN ('mysql-monitor_connect_interval','mysql-monitor_ping_interval',
'mysql-monitor_read_only_interval');
```

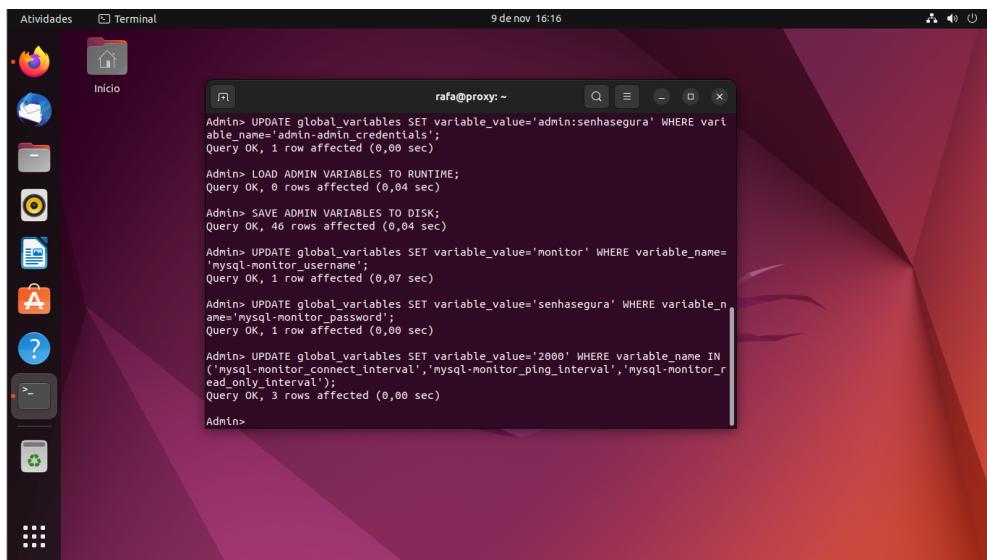
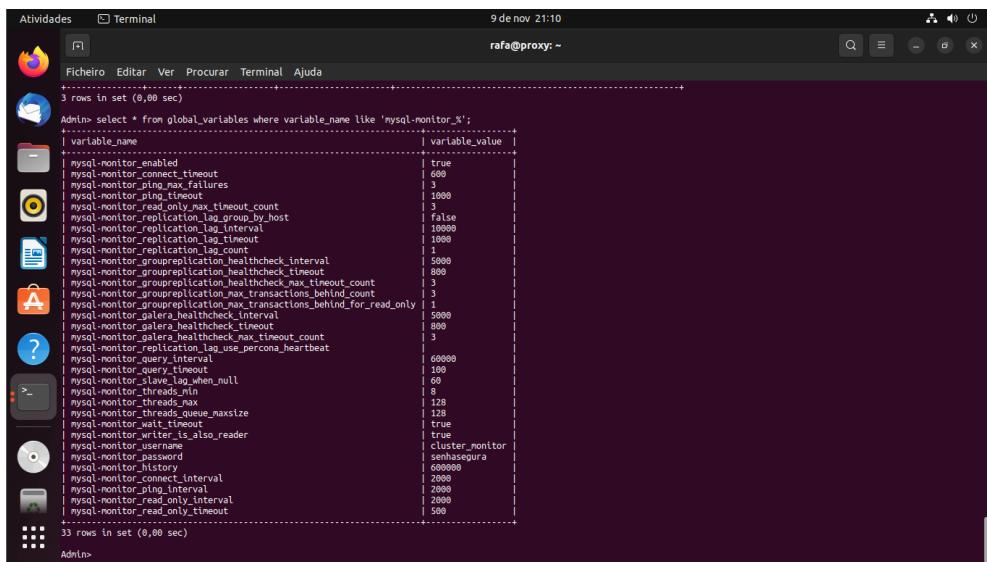


Figura 9.48: Monitorização ProxySQL

Confirmar as alterações feitas:

```
SELECT * FROM global_variables WHERE variable_name LIKE 'mysql-monitor_%';
```



variable_name	variable_value
mysql-monitor_enabled	true
mysql-monitor_connect_timeout	600
mysql-monitor_ping_max_failures	3
mysql-monitor_ping_timeout	1000
mysql-monitor_read_only_max_timeout_count	3
mysql-monitor_replication_lag_group_by_host	false
mysql-monitor_replication_lag_interval	10000
mysql-monitor_replication_lag_max_timeout	1000
mysql-monitor_replication_lag_count	1
mysql-monitor_groupreplication_healthcheck_interval	5000
mysql-monitor_groupreplication_healthcheck_timeout	800
mysql-monitor_groupreplication_healthcheck_max_timeout_count	3
mysql-monitor_groupreplication_transactions_behind_count	3
mysql-monitor_groupreplication_max_transactions_behind_for_read_only	1
mysql-monitor_galera_healthcheck_interval	5000
mysql-monitor_galera_healthcheck_timeout	800
mysql-monitor_galera_healthcheck_max_timeout_count	3
mysql-monitor_replication_lag_use_percona_heartbeat	
mysql-monitor_query_interval	60000
mysql-monitor_query_timeout	100
mysql-monitor_query_when_null	10
mysql-monitor_threads_min	8
mysql-monitor_threads_max	128
mysql-monitor_threads_queue_maxsize	128
mysql-monitor_wait_timeout	true
mysql-monitor_user_is_also_reader	cluster_monitor
mysql-monitor_username	senhassegura
mysql-monitor_password	600000
mysql-monitor_history	
mysql-monitor_connecting_interval	2000
mysql-monitor_ping_interval	2000
mysql-monitor_read_only_interval	2000
mysql-monitor_read_only_timeout	500

Figura 9.49: Configurações ProxySQL

Guardar as alterações realizadas:

```
LOAD MYSQL VARIABLES TO RUNTIME;
```

```
SAVE MYSQL VARIABLES TO DISK;
```

## Passo 5: Adicionar nós ao Backend ProxySQL

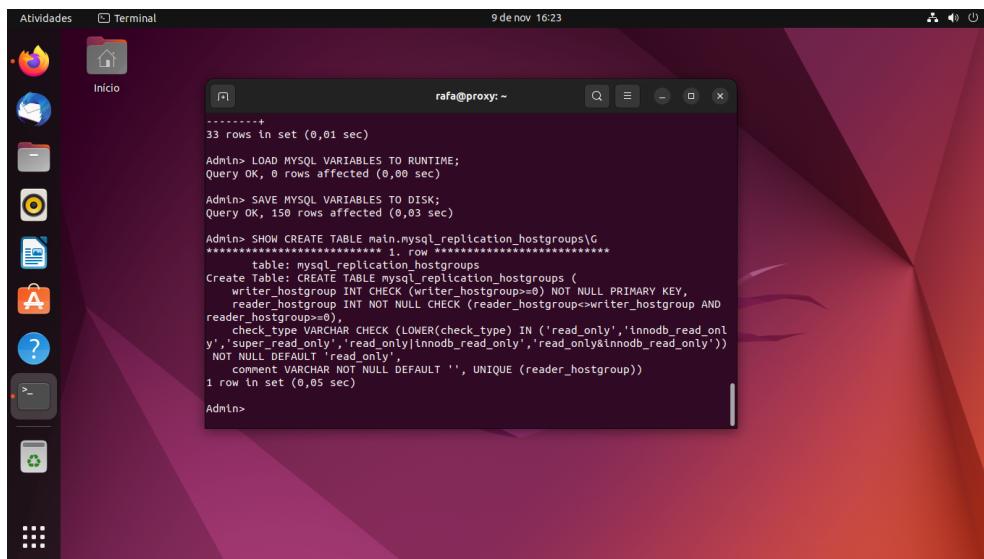
O passo seguinte é adicionar os três nós que existem no nosso Galera Cluster. O ProxySQL usa *hostgroups* para os nós de backend. Um *hostgroup* é um conjunto de nós identificados por um número positivo, 1 ou 2. O objetivo de usar *hostgroups* é auxiliar as consultas do ProxySQL aos diferentes *host groups*, utilizando o encaminhamento de consultas ProxySQL.

O ProxySQL tem os seguintes *hostgroups* lógicos:

- Escritores - nós que podem escrever/alterar dados (Nó Master), também são leitores.
- Leitores - nós que só podem aceitar consultas de leitura (Nó Slave)

Próximo passo, configurar a tabela ‘*mysql\_replication\_hostgroup*’ na base de dados e especificar os *hostgroups*:

```
SHOW CREATE TABLE main.mysql_replication_hostgroups\G
```



```
Atividades Terminal 9 de nov 16:23
Inicio
rafa@proxy: ~
-----
33 rows in set (0,01 sec)

Admin> LOAD MYSQL VARIABLES TO RUNTIME;
Query OK, 0 rows affected (0,00 sec)

Admin> SAVE MYSQL VARIABLES TO DISK;
Query OK, 150 rows affected (0,03 sec)

Admin> SHOW CREATE TABLE main.mysql_replication_hostgroups\G
-----
 1 row in set (0,01 sec)

table: mysql_replication_hostgroups
Create Table: CREATE TABLE mysql_replication_hostgroups (
    writer_hostgroup INT CHECK (writer_hostgroup>=0) NOT NULL PRIMARY KEY,
    reader_hostgroup INT NOT NULL CHECK (reader_hostgroup>writer_hostgroup AND
reader_hostgroup>=0),
    check_type VARCHAR CHECK (LOWER(check_type) IN ('read only','innodb_read_only',
'read_only','read_only&innodb_read_only','read_only&innodb_read_only'))
NOT NULL DEFAULT 'read_only',
    comment VARCHAR NOT NULL DEFAULT '' , UNIQUE (reader_hostgroup)
)
1 row in set (0,05 sec)

Admin>
```

Figura 9.50: Criação tabela mysql\_replication\_hostgroup

```
INSERT INTO main.mysql_replication_hostgroups (writer_hostgroup,
reader_hostgroup,comment) VALUES (1,2,'galera_cluster');
```

De seguida inserir os nós no proxySQL:

```
INSERT INTO main.mysql_servers(hostgroup_id,hostname,port)
VALUES (1,'192.168.1.118',3306);

INSERT INTO main.mysql_servers(hostgroup_id,hostname,port)
VALUES (1,'192.168.1.119',3306);

INSERT INTO main.mysql_servers(hostgroup_id,hostname,port)
VALUES (1,'192.168.1.120',3306);
```

Guardar as configurações em disco:

```
LOAD MYSQL VARIABLES TO RUNTIME;
```

```
SAVE MYSQL VARIABLES TO DISK;
```

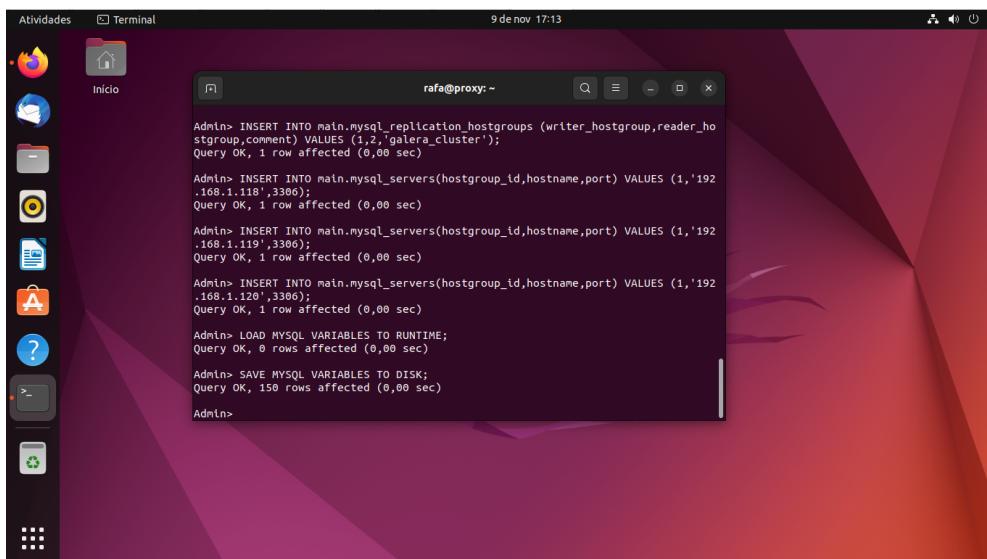


Figura 9.51: Inserir nós ProxySQL

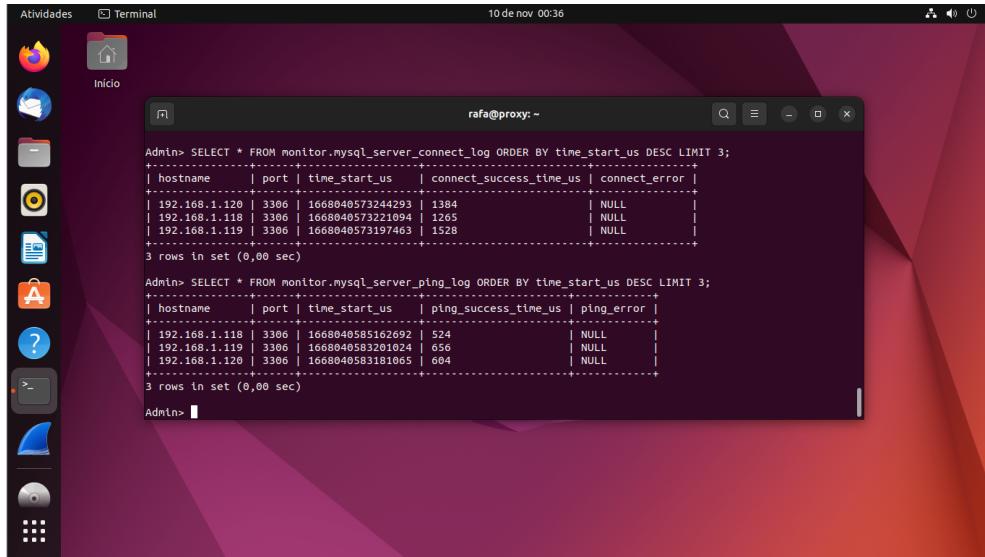
Carregar as configurações para o *runtime*:

```
LOAD MYSQL SERVERS TO RUNTIME;
```

Confirmação de que os servidores se conectaram corretamente (conexão e ping):

```
SELECT * FROM monitor.mysql_server_connect_log ORDER BY time_start_us DESC LIMIT 3;
```

```
SELECT * FROM monitor.mysql_server_ping_log ORDER BY time_start_us DESC LIMIT 3;
```



A screenshot of a Linux desktop environment showing a terminal window titled "Terminal". The terminal window displays two MySQL queries and their results. The first query is:

```
Admin> SELECT * FROM monitor.mysql_server_connect_log ORDER BY time_start_us DESC LIMIT 3;
```

The results show three rows of data with columns: hostname, port, time\_start\_us, connect\_success\_time\_us, and connect\_error. The data is as follows:

hostname	port	time_start_us	connect_success_time_us	connect_error
192.168.1.120	3306	1668040573244293	1384	NULL
192.168.1.118	3306	1668040573221094	1265	NULL
192.168.1.119	3306	1668040573197463	1528	NULL

The second query is:

```
Admin> SELECT * FROM monitor.mysql_server_ping_log ORDER BY time_start_us DESC LIMIT 3;
```

The results show three rows of data with columns: hostname, port, time\_start\_us, ping\_success\_time\_us, and ping\_error. The data is as follows:

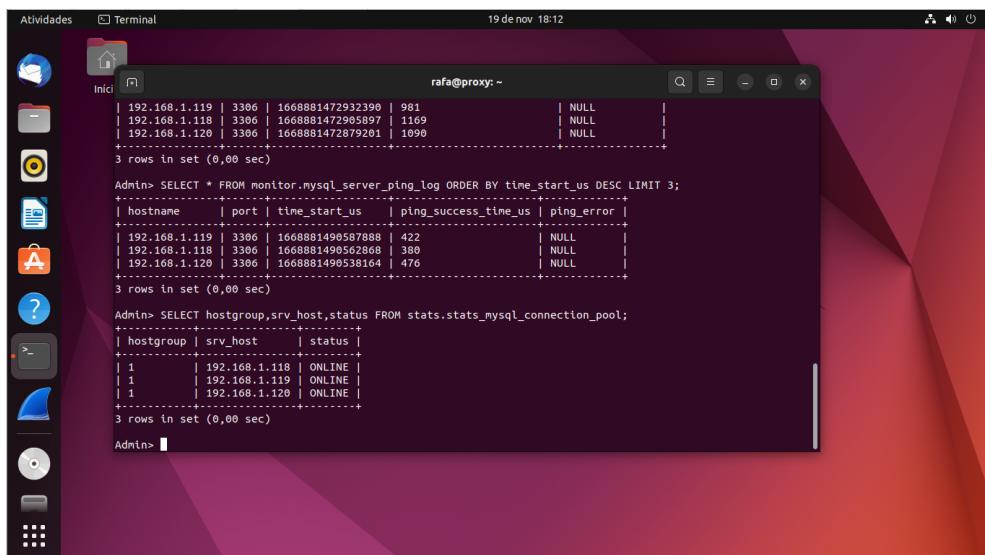
hostname	port	time_start_us	ping_success_time_us	ping_error
192.168.1.118	3306	1668040585162692	524	NULL
192.168.1.119	3306	1668040583201024	656	NULL
192.168.1.120	3306	1668040583181065	664	NULL

Both queries were executed in less than 0.00 seconds.

Figura 9.52: Conexão entre Proxy e Nodes

Por fim, ainda pude verificar no Proxy o estado dos nós do cluster.

```
SELECT hostgroup,srv_host,status FROM stats.stats_mysql_connection_pool;
```



A screenshot of a Linux desktop environment showing a terminal window titled "Terminal". The terminal window displays three MySQL queries and their results. The first query is:

```
Admin> SELECT hostgroup,srv_host,status FROM stats.stats_mysql_connection_pool;
```

The results show three rows of data with columns: hostgroup, srv\_host, and status. The data is as follows:

hostgroup	srv_host	status
1	192.168.1.118	ONLINE
1	192.168.1.119	ONLINE
1	192.168.1.120	ONLINE

The second query is:

```
Admin> SELECT * FROM monitor.mysql_server_connect_log ORDER BY time_start_us DESC LIMIT 3;
```

The third query is:

```
Admin> SELECT * FROM monitor.mysql_server_ping_log ORDER BY time_start_us DESC LIMIT 3;
```

Figura 9.53: Conexão entre Proxy e nós

## Passo 6: Criação de utilizador MySQL

Neste passo vamos criar um utilizador MySQL que se irá ligar ao cluster através do ProxySQL. Para isso num dos nós é criado o utilizador que será replicado pelos restantes.

```
create user 'rafa1'@'%' identified by 'senhasegura';
```

Garantir que o utilizador tem privilégios para aceder a várias funções:

```
grant all privileges on testdb.* to 'rafa1'@'%' with grant option;  
flush privileges;
```

## Passo 7: Adicionar o utilizador MySQL ao ProxySQL

O registo deste utilizador é acrescentado na tabela *mysql\_users* da base de dados do ProxySQL:

```
SHOW CREATE TABLE mysql_users\G
```

Depois de criar a tabela, é inserido numa query o nome de utilizador, palavra-passe e *hostgroup*:

```
INSERT INTO mysql_users(username,password,default_hostgroup)  
VALUES ('rafa1','senhasegura',1);
```

Para terminar, falta guardar as alterações:

```
LOAD MYSQL USERS TO RUNTIME;
```

```
SAVE MYSQL USERS TO DISK;
```

Para confirmar, podemos verificar se o utilizador foi adicionado:

```
SELECT * FROM mysql_users;
```

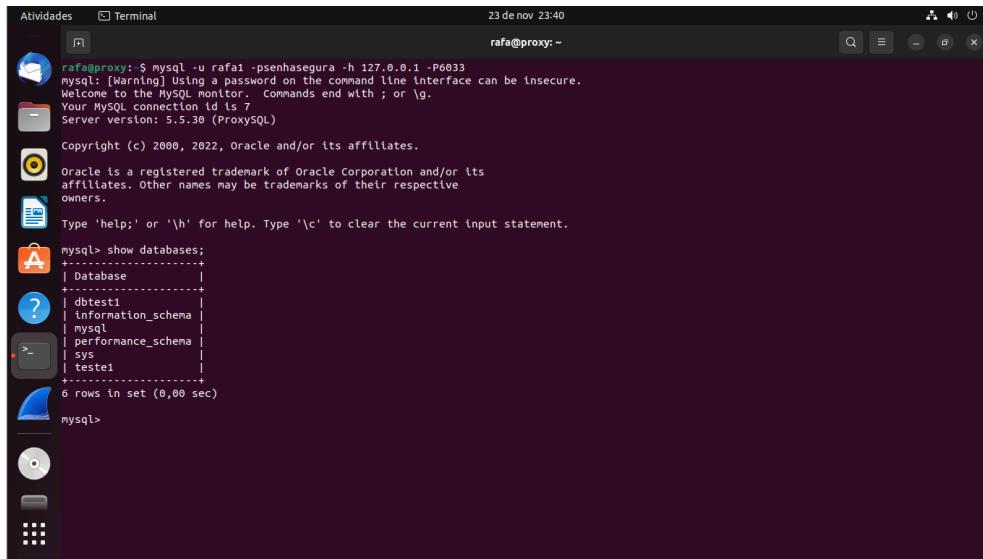
Passo 8: Conexão do utilizador MySQL criado no passo anterior no ProxySQL.

Assim, podemos fazer query ás base de dados, testando a ligação e permissões criadas para o acesso ao cluster pelo ProxySQL.

```
mysql -urafa1 -h 127.0.0.1 -P6033 -p
```

Depois de fazer login no ProxySQL com o utilizador MySQL podemos ver, por exemplo, as bases de dados existentes no cluster:

```
show databases;
```



```
rafa@proxy: ~
rafa@proxy: $ mysql -u rafal -psenhasegura -h 127.0.0.1 -P6033
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.5.30 (ProxySQL)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| dbtest1   |
| information_schema |
| mysql     |
| performance_schema |
| sys       |
| teste1   |
+-----+
6 rows in set (0,00 sec)

mysql>
```

Figura 9.54: Verificação Base de dados no Proxy

## 9.10 Instalação Wireshark

```
sudo add-apt-repository ppa:wireshark-dev/stable  
sudo apt update  
sudo apt install wireshark
```

Durante a instalação vamos ser notificados com uma configuração onde se prentende que seja escolhido a opção SIM ou NÃO, para utilizadores do SO sem permissões poderem capturar pacotes, eu optei por escolher SIM.

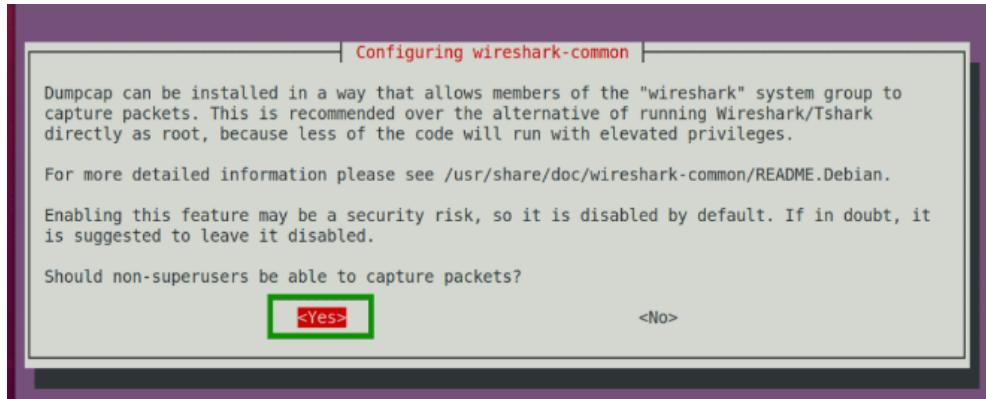


Figura 9.55: Configuring Wireshark-common

```
sudo usermod -aG wireshark $(whoami)
```

Em algumas experiências anteriores ao abrir o Wireshark pelo atalho gerado, existia um *bug* onde não aparecia na tela inicial do Wireshark as várias interfaces de rede do "computador" e para evitar esse *bug* eu inicie a partir daí o Wireshark pelo terminal usando:

```
sudo wireshark
```

Depois de aberto o Wireshark na tela inicial escolhe-se a interface de rede, no meu caso é a ens33, e começa a captura de pacotes.

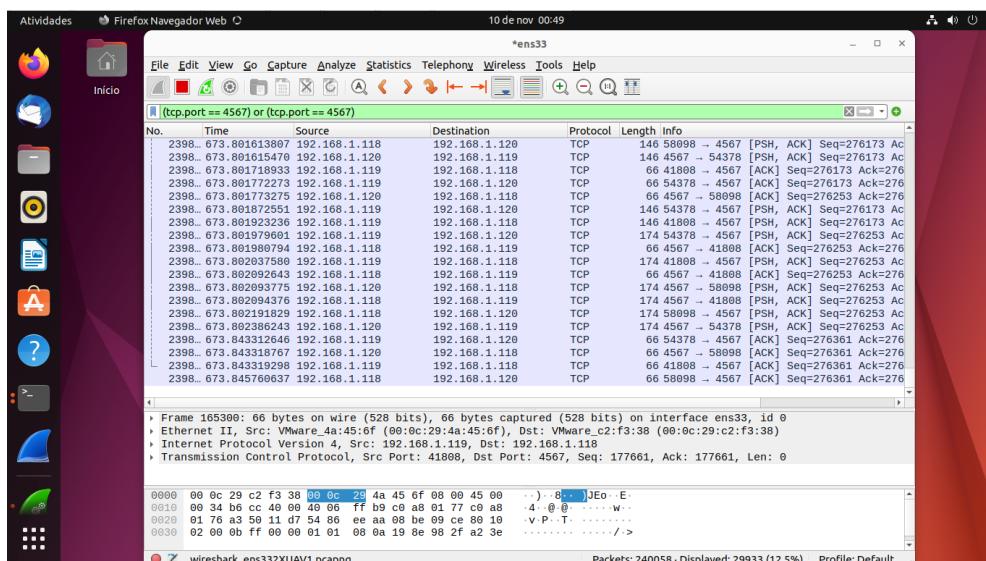
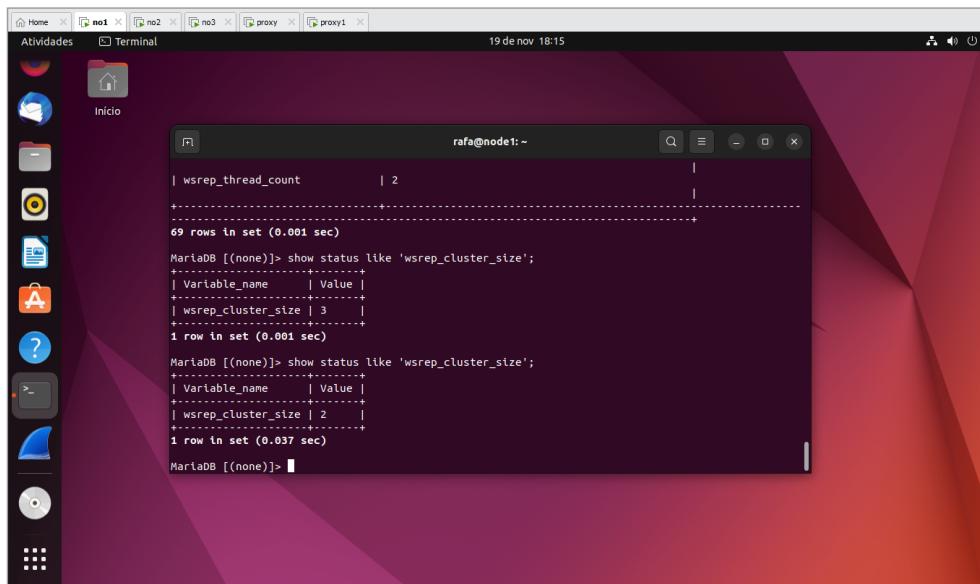


Figura 9.56: Wireshark

## 9.11 Experiência

Para esta experiência simulei a falha de um nó do cluster. Apenas desliguei a interface de rede na VM 3 (nó 3), ou podia também desligar o serviço mariaDB com o comando `sudo systemctl stop mariadb`.

Depois de desligar a interface de rede no nó 3 seria de esperar que este nó saísse do cluster:



```
| wsrep_thread_count | 2 |
+-----+-----+
69 rows in set (0.001 sec)

MariaDB [(none)]> show status like 'wsrep_cluster_size';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| wsrep_cluster_size | 3 |
+-----+-----+
1 row in set (0.001 sec)

MariaDB [(none)]> show status like 'wsrep_cluster_size';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| wsrep_cluster_size | 2 |
+-----+-----+
1 row in set (0.037 sec)

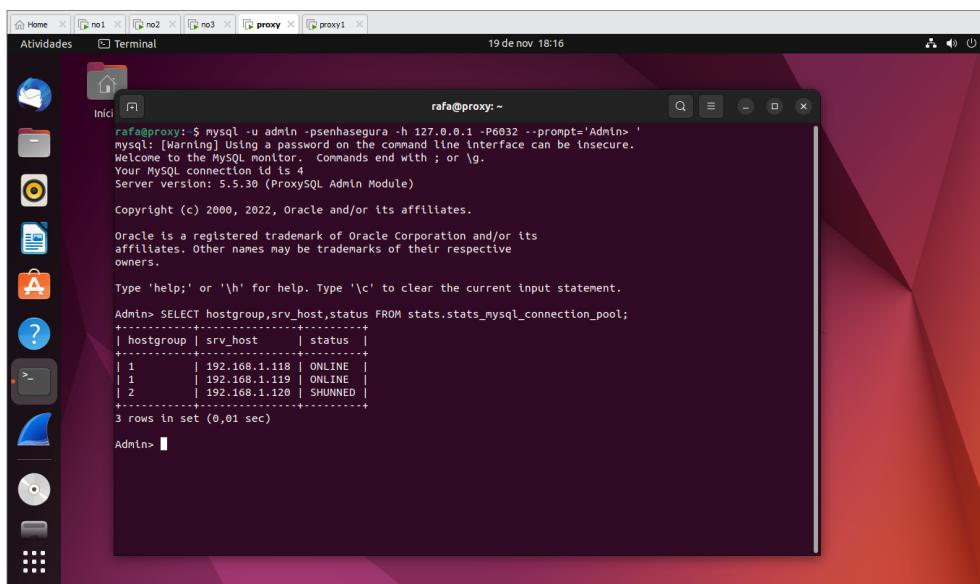
MariaDB [(none)]>
```

Figura 9.57: Tamanho do cluster depois da falha

Como podemos ver, eu tinha verificado antes de desligar a interface de rede do nó 3 o tamanho do cluster e depois do nó trés ficar inativo o tamanho do cluster passou para dois, como esperado.

Passando para a VM do ProxySQL fui verificar a pool dos *hostgroup* e verifiquei que o nó trés ficou *SHUNNED*:

```
SELECT hostgroup,srv_host,status FROM stats.stats_mysql_connection_pool;
```



```
rafa@proxy:~$ mysql -u admin -psenhasegura -h 127.0.0.1 -P6032 --prompt='Admin>' 
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.5.30 (ProxySQL Admin Module)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

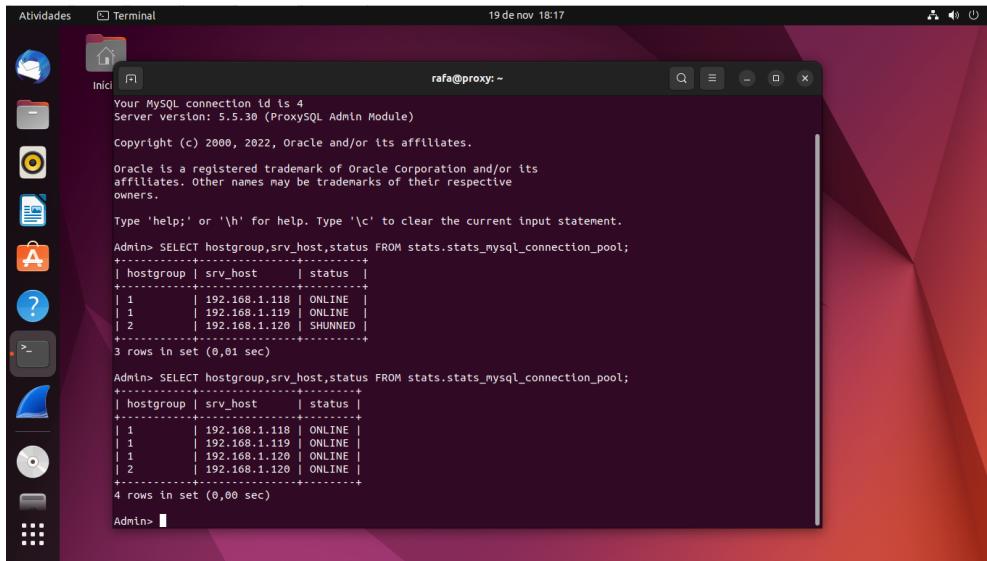
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

Admin> SELECT hostgroup,srv_host,status FROM stats.stats_mysql_connection_pool;
+-----+-----+-----+
| hostgroup | srv_host | status |
+-----+-----+-----+
| 1         | 192.168.1.118 | ONLINE |
| 1         | 192.168.1.119 | ONLINE |
| 2         | 192.168.1.120 | SHUNNED |
+-----+-----+-----+
3 rows in set (0,01 sec)

Admin>
```

Figura 9.58: Pool Hostgroups

Ativei novamente a interface de rede da VM do nó três e o cluster voltou a ficar com o tamanho de três e o nó três da pool do ProxySQL voltou ao estado inicial.



The screenshot shows a terminal window on a Linux desktop environment. The title bar reads "Atividades Terminal" and the date "19 de nov 18:17". The user is connected as "rafa@proxy: ~". The terminal displays the following MySQL command output:

```
Your MySQL connection id is 4
Server version: 5.5.30 (ProxySQL Admin Module)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

Admin> SELECT hostgroup,srv_host,status FROM stats.stats_mysql_connection_pool;
+-----+-----+-----+
| hostgroup | srv_host | status |
+-----+-----+-----+
| 1         | 192.168.1.118 | ONLINE |
| 1         | 192.168.1.119 | ONLINE |
| 2         | 192.168.1.120 | SHUNNED |
+-----+-----+-----+
3 rows in set (0,01 sec)

Admin> SELECT hostgroup,srv_host,status FROM stats.stats_mysql_connection_pool;
+-----+-----+-----+
| hostgroup | srv_host | status |
+-----+-----+-----+
| 1         | 192.168.1.118 | ONLINE |
| 1         | 192.168.1.119 | ONLINE |
| 1         | 192.168.1.120 | ONLINE |
| 2         | 192.168.1.120 | ONLINE |
+-----+-----+-----+
4 rows in set (0,00 sec)

Admin>
```

Figura 9.59: Pool Hostgroups

## 9.12 Sysbench

Para dar inicio ao tema do Sysbench comecei por adicionar uma nova VM e o seu IP como está descrito na tabela de encaminhamento no inicio do capitulo é 192.168.1.121. De seguida registo detalhadamente toda a configuração e os testes efetuados.

Passo 1: Instalação Sysbench

```
curl -s https://packagecloud.io/install/repositories/akopytov/sysbench  
/script.deb.sh | sudo bash  
  
sudo apt-get install sysbench
```

Passo 2: Criação Base de Dados no Cluster

Para este passo evitava de ter criado uma nova base de dados uma vez que já existia no cluster uma denominada "dbtest1". Então, no nó 1 comecei por criar a base de dados que serviu de teste. Como a replicação está ativa, como já expliquei anteriormente, ao criar num nó a base de dados, essa será replicada por todo o cluster.

```
create database sbtest1
```

Passo 3: Diretoria

A execução do teste foi feita na diretoria:

```
/usr/share/sysbench/
```

Passo 4: Execução do teste

Para a execução do teste existem duas fases, uma onde será indicado a fase de *prepare* e a segunda que será o *run*. Para o comando *prepare* e *run* utilizei o utilizador rafal1 com a respetiva password, o host do proxy, e a base de dados sbtest1.

```
sysbench oltp_read_write --db-driver=mysql  
--table-size=100 --mysql-user=rafa1 --mysql-password=senhasegura  
--mysql-port=6033 --mysql-host=192.168.1.122 --mysql-db=sbtest1 --threads=4  
prepare --debug
```

```
sysbench oltp_read_write --db-driver=mysql
--table-size=100 --mysql-user=rafa1 --mysql-password=senhasegura
--mysql-port=6033 --mysql-host=192.168.1.122 --mysql-db=sbtest1 --threads=4
run --debug
```

```
Atividades Terminal 24 de nov 02:30
rafa@sysbench:/usr/share/sysbench
rafa@sysbench: /usr/share/sysbench$ sysbench oltp_read_write --db-driver=mysql --table-size=100 --mysql-user=rafa1 --mysql-password=senhasegura
--mysql-port=6033 --mysql-host=192.168.1.122 --mysql-db=sbtest1 --threads=4 run --debug
sysbench 1.0.20 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 4
Debug mode enabled.

Initializing random number generator from current time

Initializing worker threads..

DEBUG: Worker thread (#0) started
DEBUG: Worker thread (#1) started
DEBUG: Worker thread (#2) started
DEBUG: Worker thread (#3) started
DEBUG: Worker thread (#0) initialized
DEBUG: Worker thread (#2) initialized
DEBUG: Worker thread (#1) initialized
DEBUG: Worker thread (#3) initialized

Threads started!

DEBUG: Ignoring error 1213 Deadlock found when trying to get lock; try restarting transaction,
(last message repeated 78 times)
Time limit exceeded, exiting...
DEBUG: Ignoring error 1213 Deadlock found when trying to get lock; try restarting transaction,
Time limit exceeded, exiting...
(last message repeated 2 times)
Done.

SQL statistics:
queries performed:
    read:          3108
    write:         708
    other:        436
    total:        4252
transactions:
    queries:      142  (14.11 per sec.)
    ignored errors: 4252 (422.51 per sec.)
    reconnects:   80  (7.95 per sec.)
    total time:   10.0017s
    total number of events: 142

General statistics:
    total time:   10.0017s
    total number of events: 142

Latency (ms):
    min:           24.78
    avg:          282.62
    max:          4354.75
    95th percentile: 893.56
    sum:          40132.68

Threads fairness:
    events (avg/stddev): 35.5000/13.67
    execution time (avg/stddev): 10.0332/0.02

DEBUG: Verbose per-thread statistics:
DEBUG:     thread # 0: min: 0.0441s avg: 0.2648s max: 1.5133s events: 38
DEBUG:             total time taken by event execution: 10.0608s
DEBUG:     thread # 1: min: 0.0519s avg: 0.7144s max: 4.3547s events: 14
DEBUG:             total time taken by event execution: 10.0017s
DEBUG:     thread # 2: min: 0.0248s avg: 0.1929s max: 0.9229s events: 52
DEBUG:             total time taken by event execution: 10.0285s
DEBUG:     thread # 3: min: 0.0441s avg: 0.2643s max: 1.4464s events: 38
DEBUG:             total time taken by event execution: 10.0417s
```

Figura 9.60: Run Teste Sysbench

```
Atividades Terminal 24 de nov 02:30
rafa@sysbench:/usr/share/sysbench
rafa@sysbench: /usr/share/sysbench$ sysbench oltp_read_write --db-driver=mysql --table-size=100 --mysql-user=rafa1 --mysql-password=senhasegura
--mysql-port=6033 --mysql-host=192.168.1.122 --mysql-db=sbtest1 --threads=4 run --debug
sysbench 1.0.20 (using system LuaJIT 2.1.0-beta3)

SQL statistics:
queries performed:
    read:          3108
    write:         708
    other:        436
    total:        4252
transactions:
    queries:      142  (14.11 per sec.)
    ignored errors: 4252 (422.51 per sec.)
    reconnects:   80  (7.95 per sec.)
    total time:   10.0017s
    total number of events: 142

General statistics:
    total time:   10.0017s
    total number of events: 142

Latency (ms):
    min:           24.78
    avg:          282.62
    max:          4354.75
    95th percentile: 893.56
    sum:          40132.68

Threads fairness:
    events (avg/stddev): 35.5000/13.67
    execution time (avg/stddev): 10.0332/0.02

DEBUG: Verbose per-thread statistics:
DEBUG:     thread # 0: min: 0.0441s avg: 0.2648s max: 1.5133s events: 38
DEBUG:             total time taken by event execution: 10.0608s
DEBUG:     thread # 1: min: 0.0519s avg: 0.7144s max: 4.3547s events: 14
DEBUG:             total time taken by event execution: 10.0017s
DEBUG:     thread # 2: min: 0.0248s avg: 0.1929s max: 0.9229s events: 52
DEBUG:             total time taken by event execution: 10.0285s
DEBUG:     thread # 3: min: 0.0441s avg: 0.2643s max: 1.4464s events: 38
DEBUG:             total time taken by event execution: 10.0417s
```

Figura 9.61: Run Teste Sysbench

Na máquina do proxy no Wireshark ainda fiz uma pequena captura de pacotes para poder verificar os *SELECT* e *INSERT* que o teste fez.

Podemos verificar todos as capturas do *SELECT* durante o teste, usando o comando no Wireshark *mysql.query contains "SELECT"*:

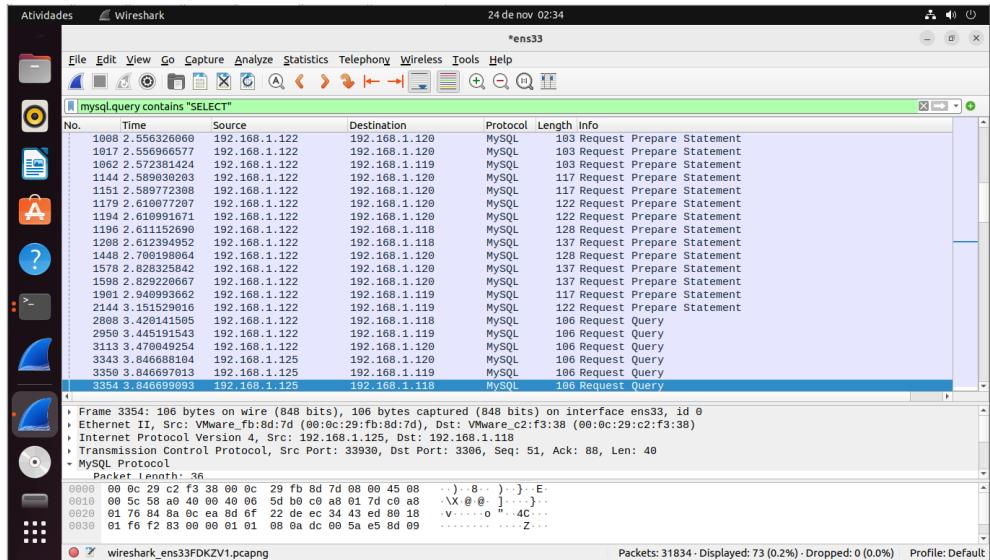


Figura 9.62: Query Select

Analizando um pacote verifiquei o Statement do MySQL e o seu conteúdo. Neste pacote a máquina de destino foi o 192.168.1.118 (Nó 1):

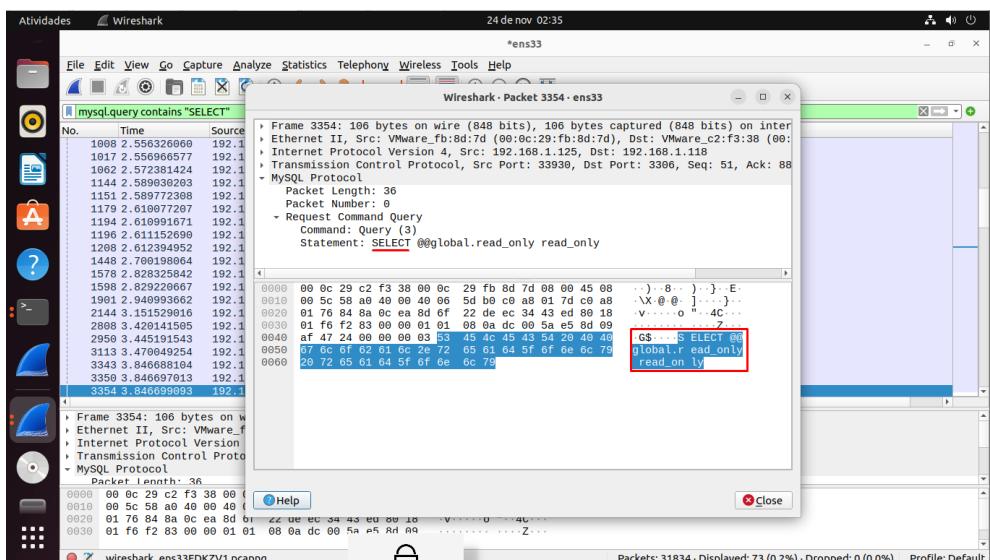


Figura 9.63: Statement SELECT MySQL

Por fim ainda verifiquei as capturas do *INSERT* durante o teste, usando o comando no Wireshark *mysql.query contains "INSERT"*:

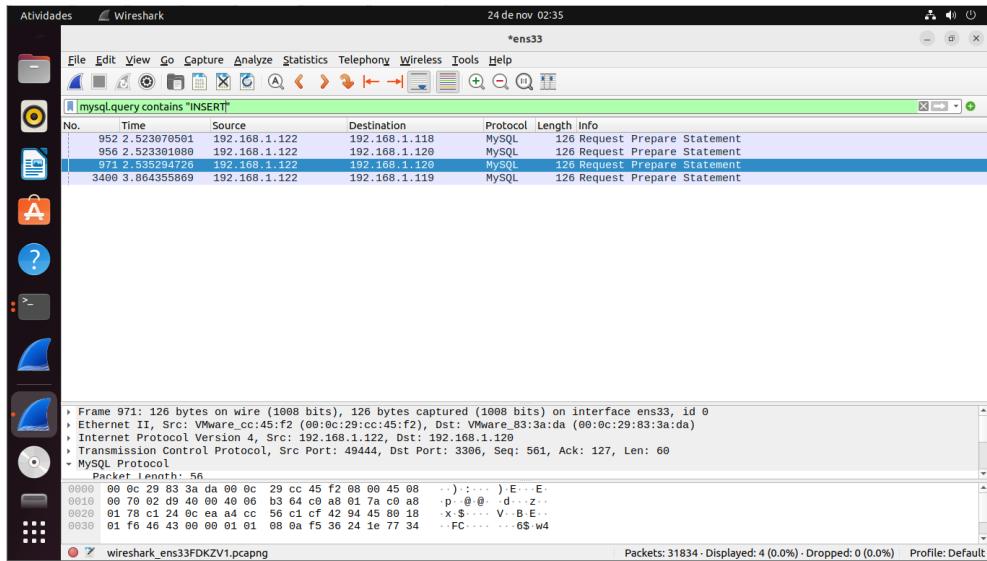


Figura 9.64: Query INSERT

Analizando um dos pacotes verifiquei o Statement do MySQL e o seu conteúdo. Neste pacote a máquina de destino foi o 192.168.1.119 (Nó 2):

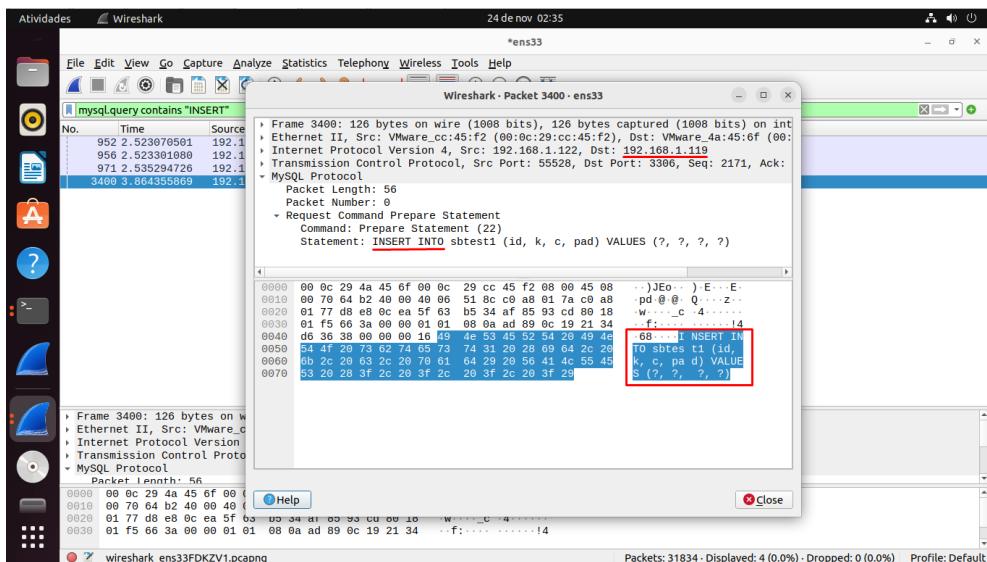


Figura 9.65: Statement INSERT MySQL

# Capítulo 10

## Estudo Experimental Meta 5

### 10.1 Requisitos da Experiência Meta 5

A elaboração da Meta 4 consistia em adicionar um servidor Proxy ao Cluster. Neste momento com a Meta 4 concluída verificamos que existe um ponto critico chamado de SPOF. Este SPOF não é nada mais do que existir apenas um servidor Proxy o que quer dizer que se acontecer algum problema no servidor Proxy o sistema fica comprometido.

A meta 5 consiste em eliminar o SPOF criado na meta 4 e para isso vamos adicionar mais um servidor Proxy e instalar e proceder a devida configuração do Keepalived.

Endereçamento Meta 5:

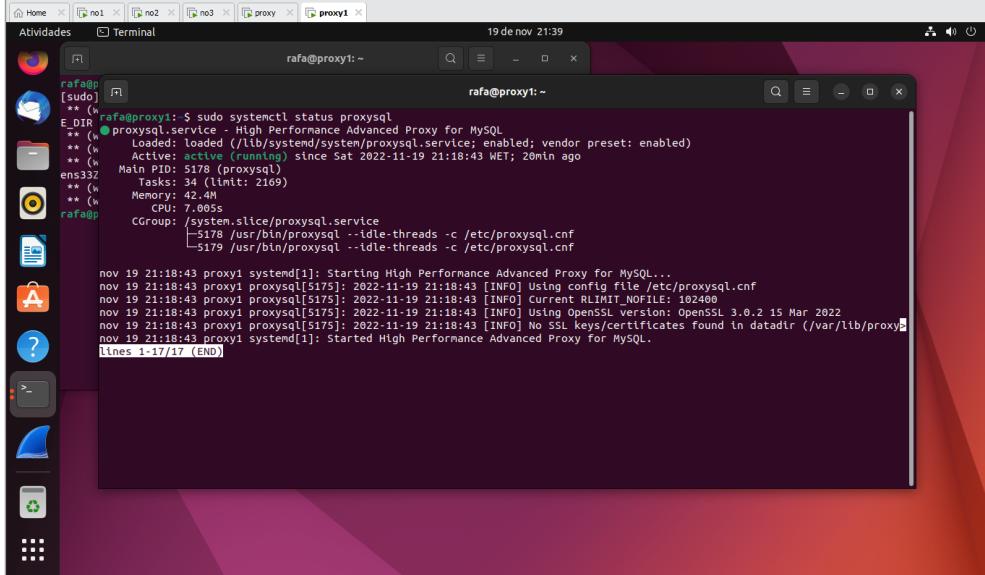
Nome	IP
Node 1	192.168.1.118
Node 2	192.168.1.119
Node 3	192.168.1.120
Sysbench	192.168.1.121
Proxy	192.168.1.122
Proxy1	192.168.1.125

Tabela 10.1: Tabela endereçamento IP Meta 5

## 10.2 Segundo Proxy

Para adicionar um segundo Proxy comecei por criar mais uma VM (proxy1) com as mesmas configurações já mencionadas na meta 4.

Antes de proceder á instalação do Keepalived foi preciso repetir o processo de criação do ProxySQL neste Proxy1. Para isso bastou seguir os passos do Capítulo 9.9 Proxy. No final verifiquei o *status* do ProxySQL:



```
[sudo] EDIR rafa@proxy1:~
```

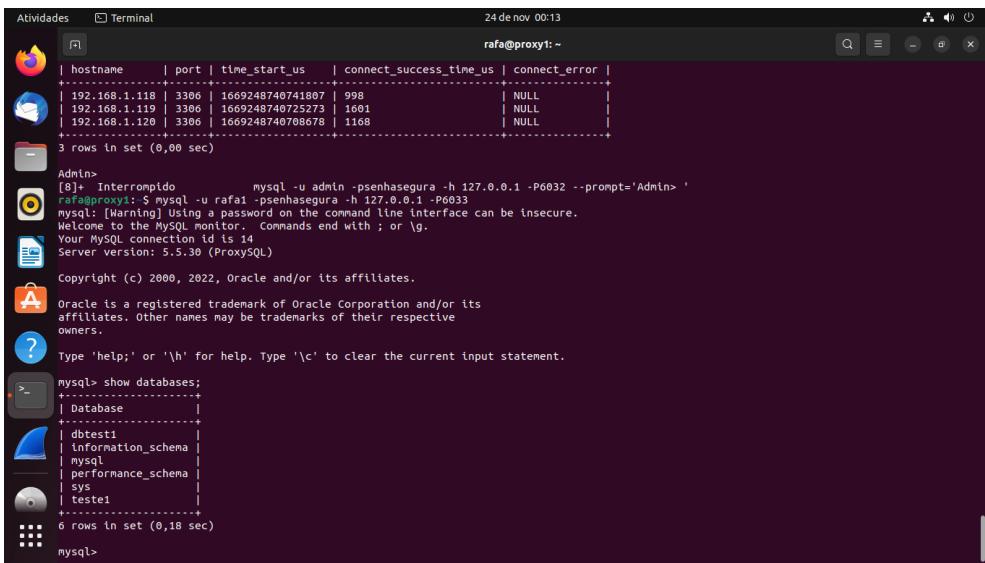
```
** (w proxysql.service - High Performance Advanced Proxy for MySQL
** (w Loaded: loaded (/lib/systemd/system/proxysql.service; enabled; vendor preset: enabled)
** (w Active: active (running) since Sat 2022-11-19 21:18:43 WET; 20min ago
** (w Main PID: 34 (proxysql)
   Tasks: 34 (limit: 2109)
** (w Memory: 42.4M
** (w CPU: 7.005s
rafa@p CGroupl /system.slice/proxysql.service
      |-5178 /usr/bin/proxysql --idle-threads -c /etc/proxysql.cnf
      +--5179 /usr/bin/proxysql --idle-threads -c /etc/proxysql.cnf

nov 19 21:18:43 proxy1 systemd[1]: Starting High Performance Advanced Proxy for MySQL...
nov 19 21:18:43 proxy1 proxysql[5175]: 2022-11-19 21:18:43 [INFO] Using config file /etc/proxysql.cnf
nov 19 21:18:43 proxy1 proxysql[5175]: 2022-11-19 21:18:43 [INFO] Current RLIMIT_NOFILE: 102400
nov 19 21:18:43 proxy1 proxysql[5175]: 2022-11-19 21:18:43 [INFO] Using OpenSSL version: OpenSSL 3.0.2 15 Mar 2022
nov 19 21:18:43 proxy1 proxysql[5175]: 2022-11-19 21:18:43 [INFO] No SSL keys/certificates found in datadir /var/lib/proxy1
nov 19 21:18:43 proxy1 systemd[1]: Started High Performance Advanced Proxy for MySQL.

lines 1-17 (END)
```

Figura 10.1: Status ProxySQL Proxy1

Com a autenticação do utilizador MySQL criado, verifiquei também se neste segundo proxy tinha acesso ás bases de dados existentes no cluster:



```
rafa@proxy1:~
```

hostname	port	time_start_us	connect_success_time_us	connect_error
192.168.1.118	3306	1669248740741807	998	NULL
192.168.1.119	3306	1669248740725273	1601	NULL
192.168.1.120	3306	1669248740708678	1168	NULL

```
[8]+  Interrompido  mysql -u admin -pseñasegura -h 127.0.0.1 -P6032 --prompt='Admin> '
rafa@proxy1:~$ mysql -u rafai -pseñasegura -h 127.0.0.1 -P6033
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \q.
Your MySQL connection id is 14
Server version: 5.5.30 (ProxySQL)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| dbtest1  |
| information_schema |
| mysql    |
| performance_schema |
| sys     |
| teste1  |
+-----+
6 rows in set (0,18 sec)

mysql>
```

Figura 10.2: Show Databases Proxy1

Concluída toda a configuração necessária do ProxySQL, estavam reunidas todas as condições para avançar com o Keepalived.

### 10.2.1 Instalação Keepalived

A instalação do Keepalived foi feita nos dois proxies e para isso verificamos a seguinte instalação:

```
apt-get install keepalived
```

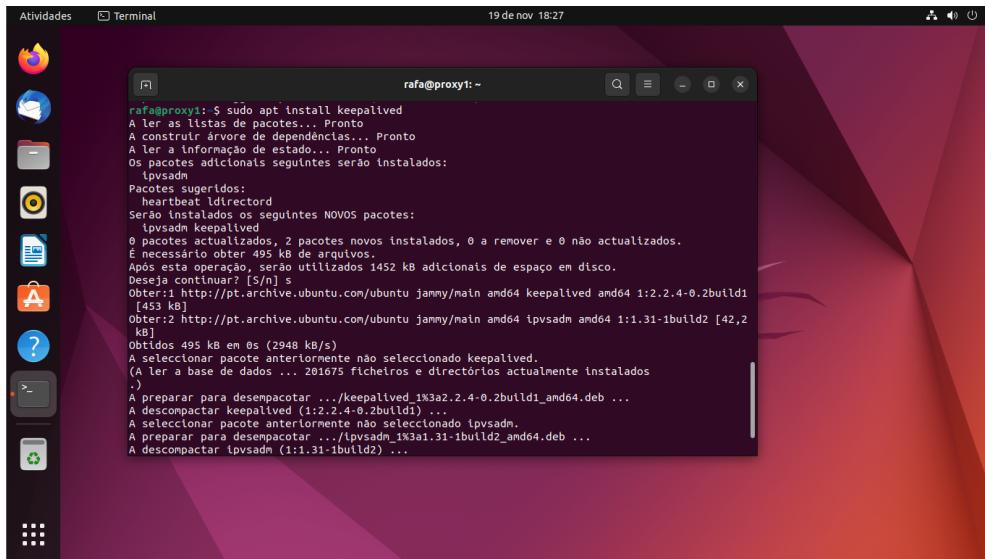


Figura 10.3: Instalação Keepalived

## 10.2.2 Configuração Keepalived Proxy

No ficheiro de configuração do Keepalived, `/etc/keepalived/keepalived.conf`, vamos dar inicio á configuração com o intuito de verificar o *MASTER* e *BACKUP* entre os dois proxies.

```
vrrp_instance V1_1{
    interface ens33
    state MASTER
    virtual_router_id 40
    priority 101
    authentication {
        auth_type PASS
        auth_pass senhasegura
    }
    virtual_ipaddress {
        192.168.1.200/24
    }
}
```

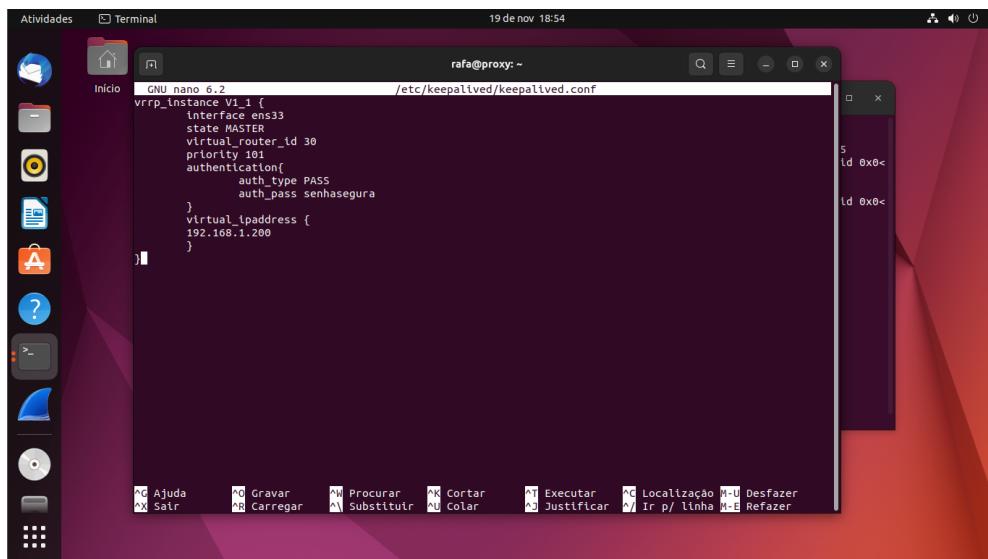


Figura 10.4: Keepalived Master Configuração

Depois da configuração convém reiniciar o serviço Keepalived:

```
sudo service keepalived restart
```

No Wireshark podemos verificar que já existe tráfego VRRP:

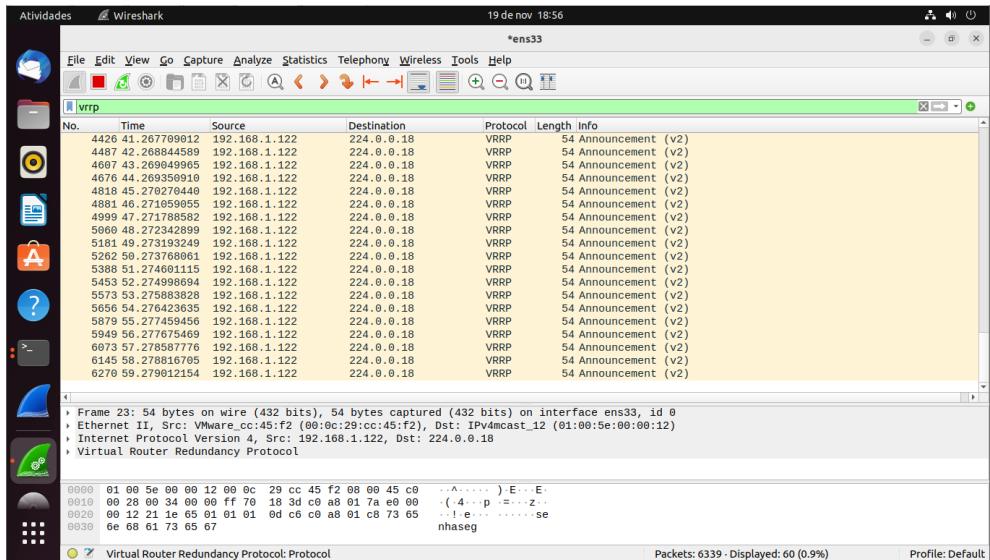


Figura 10.5: Tráfego VRRP

NOTA: Como o Keepalived de segundo a segundo envia um pacote e é MASTER, só verificamos tráfego do proxy, 192.168.1.122.

De seguida verificamos o *status* do Keepalived:

```
sudo service keepalived status
```

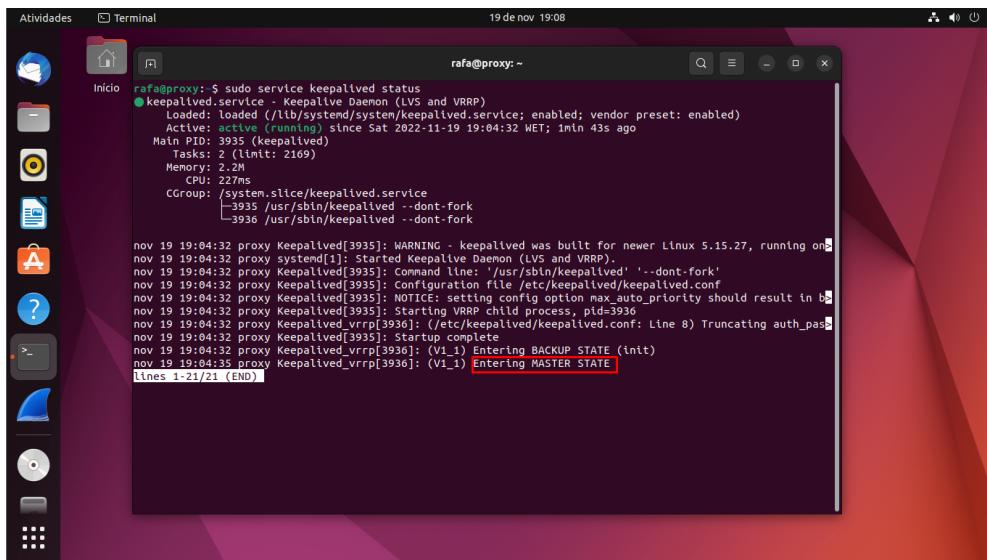


Figura 10.6: Status Keepalived Master

Verificamos que o Proxy (192.168.1.122) é o MASTER.

Ao verificar um pacote VRRP do Wireshark confirmei a prioridade do MASTER:

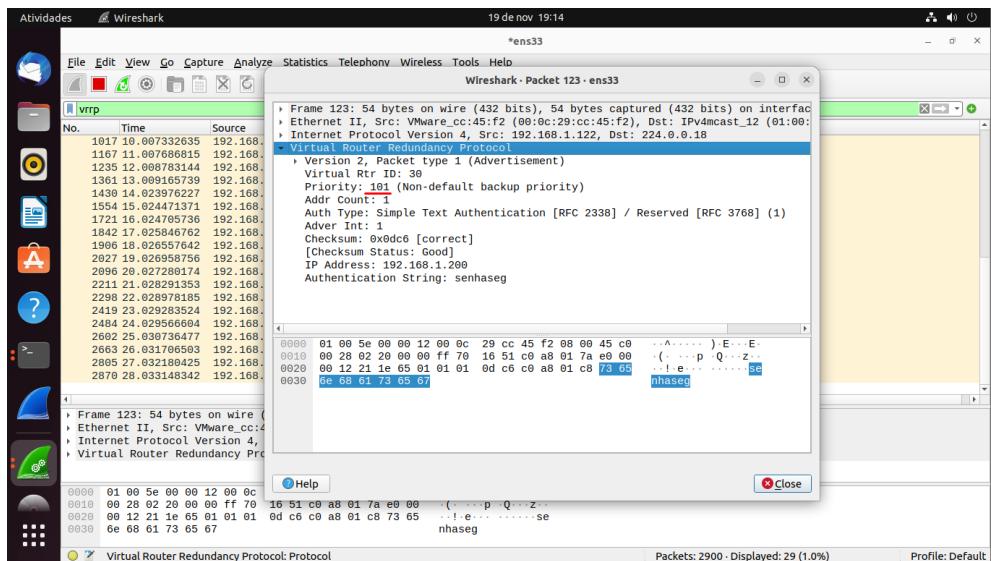


Figura 10.7: Prioridade Master

### 10.2.3 Configuração Keepalived Proxy1

Repetindo o passo anterior da configuração do ficheiro do Keepalived:

```
vrrp_instance V1_1{  
    interface ens33  
    state BACKUP  
    virtual_router_id 40  
    priority 100  
    authentication {  
        auth_type PASS  
        auth_pass senhasegura  
    }  
    virtual_ipaddress {  
        192.168.1.200/24  
    }  
}
```

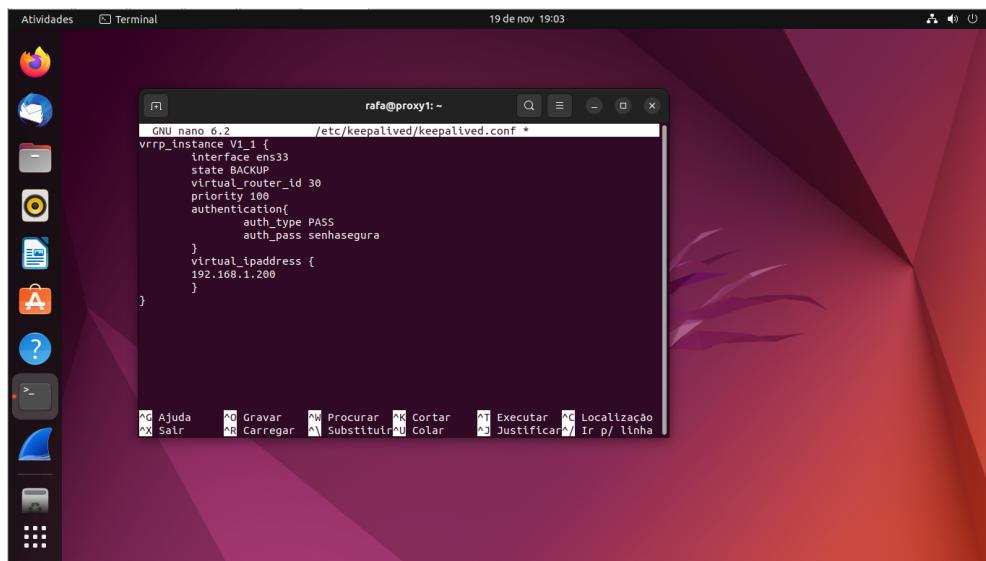


Figura 10.8: Keepalived Backup Configuração

Depois da configuração convém reiniciar o serviço Keepalived:

```
sudo service keepalived restart
```

De seguida verificamos o *status* do Keepalived:

```
sudo service keepalived status
```

Figura 10.9: Status Keepalived Backup

Verificamos que o Proxy1 192.168.1.125 é o BACKUP.

## 10.3 Experiência

O intuito desta experiência foi presenciar a troca de MASTER e BACKUP entre os dois proxies.

### 10.3.1 Estado atual dos proxies

- proxy (192.168.1.122): MASTER
  - proxy1 (192.168.1.125): BACKUP

No proxy, ao desabilitar a interface *ens33* com o comando *ip link set ens3 down* verifiquei o STATUS, *sudo service keepalived status*, do proxy1 e passou a ser MASTER.

```
File Edit View Go Capture Analyze Statistics Telephone Wireless Tools Help
[...]
rafa@proxy:~ nov 28 00:24:04 proxy1 Keepalived_vrrp[6282]: (/etc/keepalived/keepalived.conf: Line 8) Truncating auth_pass to 8 characters nov 28 00:24:04 proxy1 Keepalived[6281]: Startup complete nov 28 00:24:04 proxy1 Keepalived_vrrp[6282]: (V1_1) Entering BACKUP STATE (init)
[lines 1-21/21 (END)]
[34s] Interruptionpi sudo service keepalived status
rafa@proxy:~ [sudo] password: Sudo service keepalived status
[...]
kepalived.service - Keepalived Daemon (LVS and VRRP)
   Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun Nov 28 00:24:04 WET; 6min ago
       Main PID: 6281 (keepalived)
          Tasks: 2 (limit: 2169)
            Memory: 0M
              CPU: 0.556s
            CGroup: /system.slice/keepalived.service
                    ├─6282 /usr/sbin/keepalived --dont-fork
                    ├─6282 /usr/sbin/keepalived --dont-fork
[...]
nov 28 00:29:30 proxy1 Keepalived_vrrp[6282]: (V1_1) Entering MASTER STATE
nov 28 00:29:32 proxy1 Keepalived_vrrp[6282]: (V1_1) Master received advert from 192.168.1.122 with higher priority 101, ou
nov 28 00:29:32 proxy1 Keepalived_vrrp[6282]: (V1_1) Entering BACKUP STATE
nov 20 00:29:51 proxy1 Keepalived_vrrp[6282]: (V1_1) Entering MASTER STATE
nov 20 00:29:52 proxy1 Keepalived_vrrp[6282]: (V1_1) Master received advert from 192.168.1.122 with higher priority 101, ou
nov 20 00:29:52 proxy1 Keepalived_vrrp[6282]: (V1_1) Entering BACKUP STATE
nov 20 00:29:53 proxy1 Keepalived_vrrp[6282]: (V1_1) Master received advert from 192.168.1.122 with higher priority 101, ou
nov 20 00:29:59 proxy1 Keepalived_vrrp[6282]: (V1_1) Entering BACKUP STATE
nov 20 00:30:37 proxy1 Keepalived_vrrp[6282]: (V1_1) Entering MASTER STATE
[lines 1-21/21 (END)]
```

Figura 10.10: Proxy1 Master

Ao mesmo tempo com o Wireshark a capturar tráfego verifiquei o momento em que o proxy 192.168.1.122 deixa de ser MASTER.

De seguida, ao habilitar a interface no proxy 192.168.1.122, *ip link set ens3 up*, verificamos o seguinte:

```
rafa@proxy:~$ sudo wtre
[sudo] senha para rafa:
sudo: wtre: comando não encontrado

rafa@proxy:~$ sudo service keepalived status
[sudo] senha para rafa:
* [wired] keepalived.service - Keepalived Daemon (LVS and VRPP)
  * [loaded] loaded (/lib/systemd/system/keepalived.service; enabled; vendor preset: enabled)
    Main PID: 5511 (keepalived)
      Active: active (running) since Sun 2022-11-20 00:22:47 WET; 11min ago
        Tasks: 2 (limit: 2169)
       Memory: 2.6M
          CPU: 1.006s
         CGroup: /system.slice/keepalived.service
                   ├─5511 /usr/sbin/keepalived --dont-fork
                   ├─5512 /usr/sbin/keepalived --dont-fork

nov 28 00:22:47 proxy Keepalived[5511]: Startup complete
nov 28 00:22:47 proxy systemd[1]: keepalived.service: Got notification message from PID 5512, but
nov 28 00:22:47 proxy systemd[1]: Started Keepalived Daemon (LVS and VRPP).
nov 28 00:22:47 proxy Keepalived_vrrp[5512]: (V1_1) Entering BACKUP STATE (init)
nov 28 00:22:50 proxy Keepalived_vrrp[5512]: (V1_1) Entering MASTER STATE
nov 28 00:30:35 proxy Keepalived_vrrp[5512]: Neighbors reports ens33 down
nov 28 00:30:35 proxy Keepalived_vrrp[5512]: (V1_1) Entering FAULT STATE
nov 28 00:33:43 proxy Keepalived_vrrp[5512]: Neighbors reports ens33 up
nov 28 00:33:45 proxy Keepalived_vrrp[5512]: (V1_1) Entering BACKUP STATE
nov 28 00:33:48 proxy Keepalived_vrrp[5512]: (V1_1) Entering MASTER STATE
lines 1-21/21 (END)
```

Figura 10.11: Proxy Master

Verificou-se que, assim que a interface *ens33* ficou *up*, o proxy 192.168.1.122 voltou a MASTER por ter prioridade 101 e o proxy1 192.168.1.125 voltou a ser BACKUP.

## 10.4 Configuração DSR e Healthcheck Keepalived

### 10.4.1 Proxy

Para a configuração do *Direct Server Return* (DSR) ou *Direct Routing* e *Healthcheck* do Keepalived no proxy 192.168.1.122, foi feita a seguinte configuração:

```
vrrp_instance V1_1{
    interface ens33
    state MASTER
    virtual_router_id 40
    priority 101
    authentication {
        auth_type PASS
        auth_pass senhasegura
    }
    virtual_ipaddress {
        192.168.1.200/24
    }
}

virtual_server 192.168.1.200 6033 {
    delay_loop 10
    lb_algo rr
    lb_kind DR
    protocol TCP

    real_server 192.168.1.122 6033 {
        weight 1
        TCP_CHECK {
            connect_timeout 10
            connect_port    6033
        }
    }
    real_server 192.168.1.125 6033 {
        weight 1
        TCP_CHECK {
            connect_timeout 10
            connect_port    6033
        }
    }
}
```

Na configuração, o parâmetro *weight* significa o peso relativo do servidor para fazer o balanceamento de carga. O *lb\_kind* seleciona um método de encaminhamento (NAT, DR, TUN). O *lb\_algo* especifica o tipo de algoritmo utilizado para a disponibilidade, neste caso foi usado *rr*, *Round-Robin*.

Depois de guardar a configuração e dar *restart* ao keepalived, *sudo service keepalived restart*, verifiquei o *status* do mesmo e verifiquei que o *helthchecker* já estava ativado com o *IP Virtual*:

```

Atividades Terminal 24 de nov 00:26
rafa@proxy: ~
rafa@proxy: $ sudo service keepalived status
● keepalived.service - Keepalive Daemon (LVS and VRRP)
   Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2022-11-24 00:22:03 WET; 4min 2s ago
     Main PID: 12020 (keepalived)
       Tasks: 3 (limit: 2169)
         Memory: 3.6M
        CPU: 303ms
      CGroup: /system.slice/keepalived.service
           ├─12020 /usr/sbin/keepalived --dont-fork
           ├─12028 /usr/sbin/keepalived --dont-fork
           └─12029 /usr/sbin/keepalived --dont-fork

nov 24 00:22:03 proxy Keepalived vrrp[12029]: (/etc/keepalived/keepalived.conf: Line 8) Truncating auth_pass to 8 characters
nov 24 00:22:03 proxy Keepalived[12020]: Startup complete
nov 24 00:22:04 proxy Keepalived vrrp[12029]: (V1_1) Entering BACKUP STATE (init)
nov 24 00:22:04 proxy Keepalived healthcheckers[12028]: Gained quorum 1+0=1 <= 2 for VS [192.168.1.200]:tcp:6033
nov 24 00:22:04 proxy Keepalived healthcheckers[12028]: Activating healthchecker for service [192.168.1.122]:tcp:6033 for VS [192.168.1.200]
nov 24 00:22:04 proxy Keepalived healthcheckers[12028]: Activating healthchecker for service [192.168.1.125]:tcp:6033 for VS [192.168.1.200]
nov 24 00:22:07 proxy Keepalived vrrp[12029]: (V1_1) Entering MASTER STATE
nov 24 00:22:12 proxy Keepalived_healthcheckers[12028]: TCP connection to [192.168.1.125]:tcp:6033 success.
nov 24 00:22:14 proxy Keepalived_healthcheckers[12028]: TCP connection to [192.168.1.122]:tcp:6033 success.
lines 1-22/22 (END)

```

Figura 10.12: DSR e Healthcheck Proxy

#### 10.4.2 Proxy1

Para a configuração do DSR ou *Direct Routing* e Healthcheck do Keepalived no proxy1 192.168.1.125, foi feita a seguinte configuração:

```

vrrp_instance V1_1{
    interface ens33
    state BAKCUP
    virtual_router_id 40
    priority 100
    authentication {
        auth_type PASS
        auth_pass senhasegura
    }
    virtual_ipaddress {
        192.168.1.200/24
    }
}

virtual_server 192.168.1.200 6033 {
    delay_loop 10
    lb_algo rr
    lb_kind DR
    protocol TCP

    real_server 192.168.1.122 6033 {
        weight 1
        TCP_CHECK {
            connect_timeout 10
            connect_port    6033
        }
    }
}

```

```

}
real_server 192.168.1.125 6033 {
    weight 1
    TCP_CHECK {
        connect_timeout 10
        connect_port      6033
    }
}

```

De seguida guarda-se a configuração e faz-se o *restart* ao keepalived, *sudo service keepalived restart*, verifiquei novamente o mesmo acontecimento que foi mostrado na configuração do MASTER (proxy):

```

Atividades Terminal 24 de nov 00:26
rafa@proxy1: ~
rafa@proxy1: $ sudo service keepalived status
● keepalived.service - Keepalive Daemon (LVS and VRRP)
   Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2022-11-24 00:22:17 WET; 3min 56s ago
     Main PID: 3901 (keepalived)
       Tasks: 3 (limit: 2169)
         Memory: 4.6M
            CPU: 348ms
          CGroup: /system.slice/keepalived.service
                  ├─3901 /usr/sbin/keepalived --dont-fork
                  ├─3902 /usr/sbin/keepalived --dont-fork
                  ├─3903 /usr/sbin/keepalived --dont-fork

nov 24 00:22:17 proxy1 Keepalived[3901]: Startup complete
nov 24 00:22:17 proxy1 systemd[1]: keepalived.service: Got notification message from PID 3903, but reception only permitted for main PID 3901
nov 24 00:22:17 proxy1 systemd[1]: Started Keepalive Daemon (LVS and VRRP).
nov 24 00:22:18 proxy1 Keepalived_vrrp[3903]: (V1_1) Entering BACKUP STATE (init)
nov 24 00:22:18 proxy1 Keepalived_healthcheckers[3902]: Gained quorum 1>=1 <= 2 for VS [192.168.1.200]:tcp:6033
nov 24 00:22:18 proxy1 Keepalived_healthcheckers[3902]: Activating healthchecker for service [192.168.1.122]:tcp:6033 for VS [192.168.1.200]
nov 24 00:22:18 proxy1 Keepalived_healthcheckers[3902]: Activating healthchecker for service [192.168.1.125]:tcp:6033 for VS [192.168.1.200]
nov 24 00:22:23 proxy1 Keepalived_healthcheckers[3902]: Activating BFD healthchecker
nov 24 00:22:29 proxy1 Keepalived_healthcheckers[3902]: TCP connection to [192.168.1.122]:tcp:6033 success.
nov 24 00:22:29 proxy1 Keepalived_healthcheckers[3902]: TCP connection to [192.168.1.125]:tcp:6033 success.
lines 1-22/22 (END)

```

Figura 10.13: DSR e Healthcheck Proxy1

Depois de verificar que o endereço IP Virtual, 192.168.1.200, já se encontrava ativo verifiquei se estava associado á minha interface de rede:

```
hostname -I
```

```

Atividades Terminal 24 de nov 01:15
rafa@proxy1: ~
rafa@proxy1: $ hostname -I
192.168.1.125 192.168.1.200 2001:8a0:e4a1:6b00:17fa:75f5:d27e:490 2001:8a0:e4a1:6b00:2601:ebd9:da10:ca38
rafa@proxy1: $ 

```

Figura 10.14: IP Virtual

Verifica-se que o IP Virtual já se encontra associado á minha interface, assim como no proxy também se verifica.

## 10.5 Experiência

Feita a configuração do Keepalived nos dois proxies tentei fazer a experiência de aceder à base de dados através do IP Virtual. Para isso instalei o DBeaver no *Personal Computer* (PC) Host.

Depois do DBeaver estar instalado criei uma conexão com a BD MariaDB usando o IP Virtual e o IP do proxy 192.168.1.12:

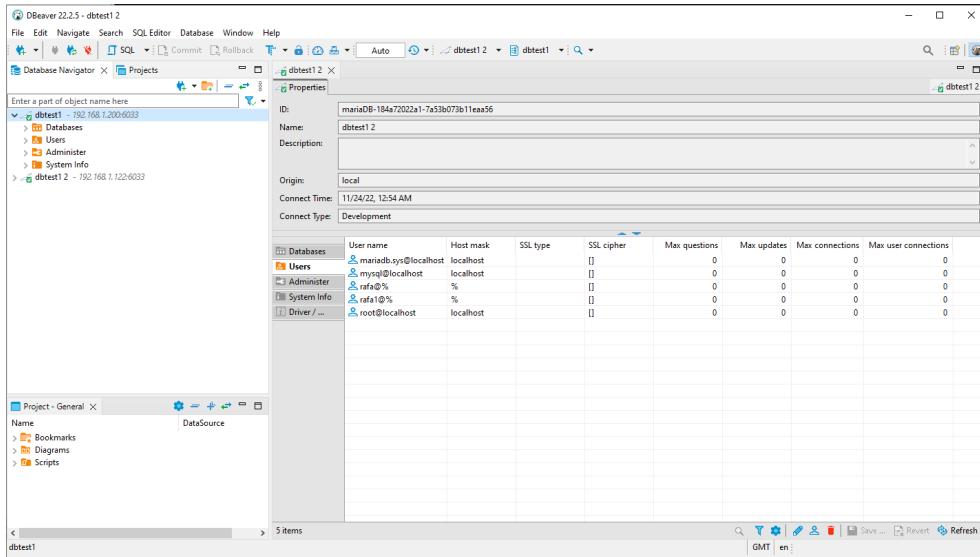


Figura 10.15: DBeaver conexão

A conexão acima, com os dois proxies ligados, funcionou. Conseguia verificar as BD existentes, os utilizadores MySQL que criei entre outras coisas.

Outra experiência que realizei foi aceder à BD através do IP Virtual mas com um proxy desligado. Para isso coloquei a VM do proxy, 192.168.1.122. O proxy1, 192.168.1.125 entrou em modo *MASTER* e consegui novamente no DBeaver aceder a BD através do IP Virtual, 192.168.1.200 com sucesso.

Para esta última experiência não bastava desligar apenas o serviço Keepalived. Ao parar apenas o serviço Keepalived no proxy, 192.168.1.122, no DBeaver não conseguia estabelecer ligação à BD com o IP Virtual.

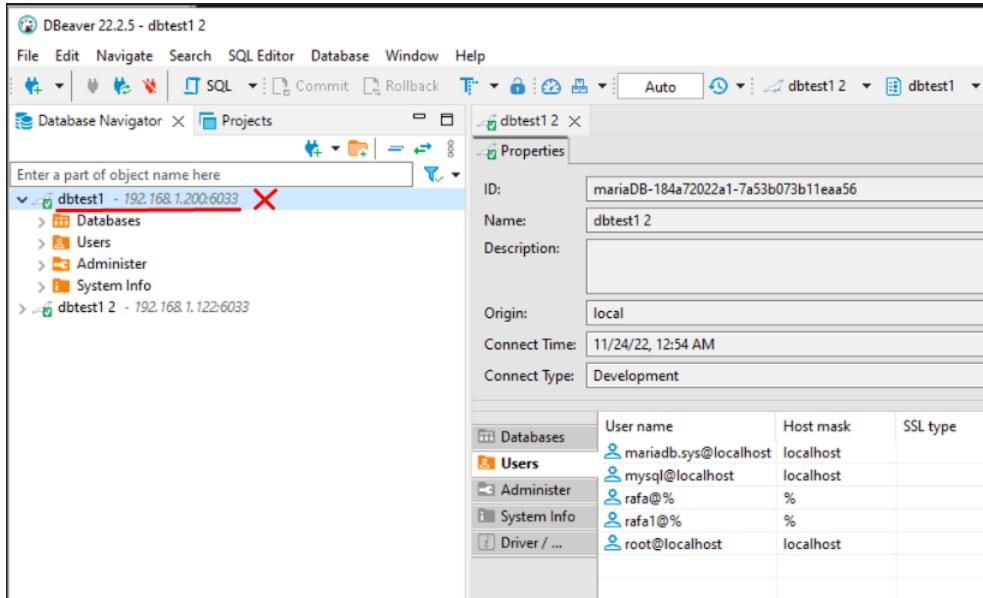


Figura 10.16: DBeaver Erro

Realizei várias experiências para compreender este problema e só consegui estabelecer ligação à BD com o IP Virtual colocando a VM do proxy em *pause* ou desligar.

# Bibliografia

- [1] *ACID (atomicity, consistency, isolation, and durability)*. URL: <https://www.techtarget.com/searchdatamanagement/definition/ACID> (acedido em 27/09/2022).
- [2] *Arquitetura de Software*. URL: [https://pt.wikipedia.org/wiki/Arquitetura\\_de\\_software](https://pt.wikipedia.org/wiki/Arquitetura_de_software) (acedido em 13/10/2022).
- [3] *Arquitetura de Software*. URL: <https://www.devmedia.com.br/arquitetura-de-software-desenvolvimento-orientado-para-arquitetura/8033> (acedido em 13/10/2022).
- [4] *Atomicity Consistency Isolation Durability*. URL: <https://www.techopedia.com/definition/23949/atomicity-consistency-isolation-durability-acid-database-management-system> (acedido em 27/09/2022).
- [5] *Base de Dados Distribuídas*. URL: <https://www.linkedin.com/pulse/bancos-de-dados-distribu%C3%ADdos-bdd-ronie-ramos-de-oliveira/?originalSubdomain=pt> (acedido em 11/10/2022).
- [6] *Cluster*. URL: <https://galeracluster.com/library/documentation/overview.html> (acedido em 08/11/2022).
- [7] *Clusters LVS + Keepalived en Linux*. URL: <https://www.estrellateyarde.org/discover/virtualizacion/clusters/clusters-ha-con-lvs/cluster-lvs-keepalived-en-linux> (acedido em 19/11/2022).
- [8] *Data Replication*. URL: <https://www.astera.com/pt/type/blog/data-replication/> (acedido em 11/10/2022).
- [9] *DBeaver*. URL: <https://dbeaver.io/> (acedido em 23/11/2022).
- [10] *Diferenças entre Escalabilidade Vertical e Horizontal*. URL: <https://pt.stackoverflow.com/questions/160142/qual-a-diferen%C3%A7a-entre-escalonamento-vertical-e-horizontal> (acedido em 09/11/2022).
- [11] *Escalabilidade Vertical e Horizontal*. URL: <https://waldyrfelix.com.br/escalabilidade-vertical-vs-escalabilidade-horizontal-6a3981783477> (acedido em 09/11/2022).
- [12] *Failover Keepalived*. URL: <https://www.redhat.com/sysadmin/advanced-keepalived> (acedido em 19/11/2022).
- [13] *Forward and Reverse Proxy*. URL: <https://www.jscape.com/blog/forward-proxy-vs-reverse-proxy> (acedido em 08/11/2022).
- [14] *Healthcheck*. URL: <https://www.linode.com/docs/guides/keepalived-with-bgp-failover/> (acedido em 19/11/2022).
- [15] *Instalação Keepalived*. URL: <https://www.questioncomputer.com/how-to-install-and-lab-keepalived-on-ubuntu-20-04-and-rocky-linux-8-5/> (acedido em 20/11/2022).

- [16] *Instalação MariaDB*. URL: <https://computingforgeeks.com/install-mariadb-galera-cluster-on-ubuntu-with-proxysql/> (acedido em 08/11/2022).
- [17] *Instalação MariaDB e galera*. URL: <https://www.howtoforge.com/how-to-setup-mariadb-galera-cluster-on-ubuntu-20-04/> (acedido em 10/11/2022).
- [18] *Instalação Sysbench*. URL: <https://github.com/akopytov/sysbench> (acedido em 08/11/2022).
- [19] *Instalação Wireshark*. URL: [https://linuxhint.com/install\\_wireshark\\_ubuntu/](https://linuxhint.com/install_wireshark_ubuntu/) (acedido em 07/11/2022).
- [20] *Keepalived and high availability: Advanced topics*. URL: <https://www.redhat.com/sysadmin/advanced-keepalived> (acedido em 19/11/2022).
- [21] *Keepalived configuration synopsis*. URL: <https://keepalived.readthedocs.io/en/latest/configurationSynopsis.html> (acedido em 19/11/2022).
- [22] *Keepalived Direct Routing Configuration*. URL: [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/7/html/load\\_balancer\\_administration/s1-initial-setup-conf-dr-vsa](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/load_balancer_administration/s1-initial-setup-conf-dr-vsa) (acedido em 19/11/2022).
- [23] *MariaDB*. URL: [https://hub.docker.com/\\_/mariadb](https://hub.docker.com/_/mariadb) (acedido em 08/11/2022).
- [24] *Master and Slave*. URL: <https://mariadb.com/kb/en/about-galera-replication/> (acedido em 08/11/2022).
- [25] *MySQL*. URL: <https://www.oracle.com/mysql/what-is-mysql/> (acedido em 09/11/2022).
- [26] *O que é uma Base de Dados Distribuídas*. URL: <https://www.devmedia.com.br/o-que-e-um-banco-de-dados-distribuido/24762> (acedido em 12/10/2022).
- [27] José Luís Pereira. *Tecnologia de Bases de Dados (2º Edição)*. FCA, 1998.
- [28] *Portas Firewall*. URL: <https://galeracluster.com/library/documentation/firewall-settings.html> (acedido em 17/11/2022).
- [29] *ProxySQL Configuration*. URL: <https://proxysql.com/documentation/proxysql-configuration/> (acedido em 19/11/2022).
- [30] *Replicação Galera Cluster*. URL: <https://galeracluster.com/library/documentation/tech-desc-introduction.html> (acedido em 08/11/2022).
- [31] *Replicação síncrona e assíncrona*. URL: <https://mariadb.com/kb/en/about-galera-replication/> (acedido em 08/11/2022).
- [32] *Sistema de Gestão de Base de Dados*. URL: [https://pt.wikipedia.org/wiki/Sistema\\_de\\_gerenciamento\\_de\\_banco\\_de\\_dados](https://pt.wikipedia.org/wiki/Sistema_de_gerenciamento_de_banco_de_dados) (acedido em 12/10/2022).
- [33] *Two Phase Commit Protocol (Distributed Transaction Management)*. URL: <https://www.geeksforgeeks.org/two-phase-commit-protocol-distributed-transaction-management/> (acedido em 27/09/2022).
- [34] *Two-Phase Commit Protocol*. URL: <https://www.educative.io/answers/what-is-the-two-phase-commit-protocol> (acedido em 28/09/2022).
- [35] *Two-Phase Commit Protocol*. URL: <https://sauravomar01.medium.com/2pc-two-phase-commit-protocol-4ba47e441cdf> (acedido em 28/09/2022).

- [36] *Virtual Servers Defenition*. URL: [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/7/html/load\\_balancer\\_administration/ch-initial-setup-vsa#s3-initial-setup-conf-file-virt-VSA](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/load_balancer_administration/ch-initial-setup-vsa#s3-initial-setup-conf-file-virt-VSA) (ace-dido em 19/11/2022).

# Apêndice A

## Anexo

### A.1 Trabalho desenvolvido da Meta 1

O trabalho desenvolvido para a Meta 1 foi uma pesquisa simples para o tema proposto pelo professor Luís Santos.

A nível de arquitetura do trabalho em Latex foi o básico. Eu já tinha trabalhado com Latex no ano letivo anterior (muito pouco) e fiquei a gostar mas soube a pouco. No verão de 2021-2022 decidi aprender mais sobre Latex e criei o meu próprio repositório GitHub, *First Contact With Latex*. Fiquei a conhecer melhor os *packages*, a implementação e como se tratava os ficheiros para os capítulos, a bibliografia, entre outros. Contudo para esta meta e trabalho inicial já possuía noções básicas da implementação e funcionamento de um trabalho em Latex. Foi enviado com alguns erros e foi feito mais trabalho de pesquisa para mitigar os erros para a meta seguinte.

### A.2 Trabalho desenvolvido da Meta 2

Para a Meta 2 foi preciso uma pesquisa mais profunda, exigiu um estudo profundo para perceber com detalhe vários assuntos da proposta para esta meta.

As implementações que efetuei foi: criação de ficheiros individuais para cada capítulo, implementação da bibliografia, introdução e melhorar alguns erros como páginas em branco, folhas com erros de formatação.

Para a próxima meta tenciono explorar e desenvolver melhor o último capítulo, Arquitetura de Aplicações Web e acrescentar a este trabalho a página de Acrónimos.

Apesar de todo o trabalho exigente em Latex, tem sido motivador pesquisar cada vez mais para conseguir entregar a tempo e horas cada meta.

### A.3 Trabalho desenvolvido da Meta 4

Esta meta exigiu um grande trabalho com uma carga horária que ocupou bastante tempo da semana. Apesar do grande trabalho que foi preciso eu já tinha trabalhado com cluster em linux e já tinha conhecimento da experiência que ia ter. Valeu a pena mais uma semana de aprendizagem e esforço para completar esta meta! Entrego o relatório mas está por finalizar, falta quase para eu conseguir o que quero entregar. Na próxima meta irá completo.

## A.4 Trabalho desenvolvido da Meta 5

Para concluir esta meta foi preciso consolidar muitos conhecimentos práticos para a implementação das experiências. Exigiu bastantes horas para a implementações do estudo com as máquinas virtuais e ir compreendendo os vários erros em alguns serviços. Neta meta final apesar do esforço para a implementar, entregue também a meta 4 concluída. Com mais algum tempo gostava de ter explorado por exemplo o Galera Cluster e o Cluster Control. Tenho noção que existe tópicos incompletos e outros que devia ter implementado mas foi um Projeto A muito exigente e conjugar com o resto das cadeiras não é fácil.