HARDENING EM LINUX

CADERNO DE ATIVIDADES

Rua Lauro Müller, 116 sala 1103

22290-906 Rio de Janeiro, RJ

Diretor Geral

Nelson Simões

Diretor de Serviços e Soluções

José Luiz Ribeiro Filho

Escola Superior de Redes

Diretor Adjunto

Leandro Marcos de Oliveira Guimarães

Equipe ESR (em ordem alfabética)

Adriana Pierro, Celia Maciel, Camila Gomes, Edson Kowask, Elimária Barbosa, Evellyn Feitosa, Felipe Arrais. Felipe Nascimento, Lourdes Soncin, Luciana Batista, Márcia Correa, Márcia Rodrigues, Monique Souza, Renato Duarte, Thays Farias, Thyago Alves e Yve Marcial.

Versão 0.1.0

Índice

Sessão 1: Instalação e configurações iniciais

A segurança e o *hardening* de um sistema Linux começa desde o primeiro momento: sua instalação. Mesmo antes de iniciarmos a preparação de uma máquina ou servidor, as considerações sobre segurança devem povoar a mente do administrador de sistemas, visando reduzir a superfície de ataque, facilitar procedimentos de auditoria e garantir que as melhores práticas de configuração serão aplicadas de forma fácil e homogênea em todo o parque computacional.

Com o advento da virtualização, prevalente na maioria das organizações já há mais de dez anos, toma força o conceito de *one service per server*, ou um serviço por servidor. Nesse caso, o objetivo é que tenhamos vários servidores simples, muitas vezes com um único serviço operacional — isso facilita enormemente a administração e diminiu a superfície de ataque de cada servidor, pois haverão poucos programas, bibliotecas e portas abertas a serem atacadas em cada máquina individual. O uso de *templates* é especialmente vantajoso para garantir que essa premissa seja aplicada com sucesso; construindo imagens-base sólidas e regularmente atualizadas e homologadas pela equipe de segurança da organização, é muito mais fácil e conveniente garantir que as VMs derivadas desses *templates* serão seguras.

Por outro lado, com a virtualização tivemos também o surgimento do *virtual machinel sprawl* — um número crescente (e muitas vezes aparentemente incontrolável) de máquinas virtuais sendo criadas no *datacenter*, minando as vantagens da simplicidade e facilidade de configuração apresentadas anteriormente. Processo e controles são fundamentais para garantir que VMs sejam criadas apenas quando necessário, e que possuam um ciclo de vida que considere sua implantação, operação e descontinuação quando não mais relevantes.

O primeiro passo para garantir que as várias máquinas em nosso ambiente estarão seguras é ser criterioso, portanto, com a criação dos *templates* de máquina virtual. Nesta sessão iremos tratar dos aspectos de segurança relevantes na instalação de um sistema Debian Linux a ser usado como *template* para derivação de VMs futuras a serem usadas neste curso, trabalhando aspectos relevantes da instalação de pacotes, gestão de discos e partições e criptografia de dados sensíveis.

1) Criação de máquina virtual no Virtualbox

1. Abra o *Oracle VM Virtualbox*. Para criar uma nova máquina virtual, clique em *New*. Na tela seguinte, você deverá escolher um nome, tipo e versão do sistema operacional a ser instalado na VM. Em *Name*, digite `debian-template`, em *Type* escolha `Linux` e em *Version* selecione `Debian (64-bit)`.

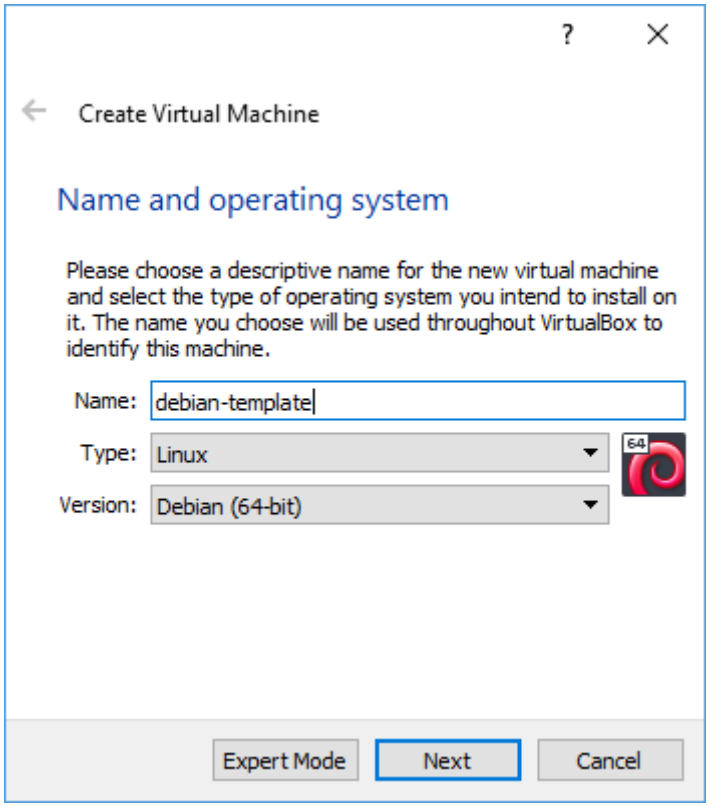

Figura 1. Criação de nova VM, parte 1

Em seguida, clique em *Next*.

2. Na tela seguinte escolheremos a quantidade de memória RAM a ser usada pelo sistema. O Debian Linux é um sistema bastante frugal, com recomendações mínimas de memória da ordem de 512 MB. Como a máquina que estamos instalando será um *template*, é interessante que ela seja bastante enxuta, e que as VMs derivadas cresçam em capacidade de acordo com o *workload* específico de cada aplicação.

   Aponte `768 MB` de RAM, e em seguida clique em *Next*.

3. Agora, iremos definir se iremos criar um novo disco rígido virtual para a VM (ou usar um preexistente), e definir seu tamanho. Nesta primeira tela, mantenha a seleção-padrão *Create a virtual hard disk now*. Clique em *Create*.

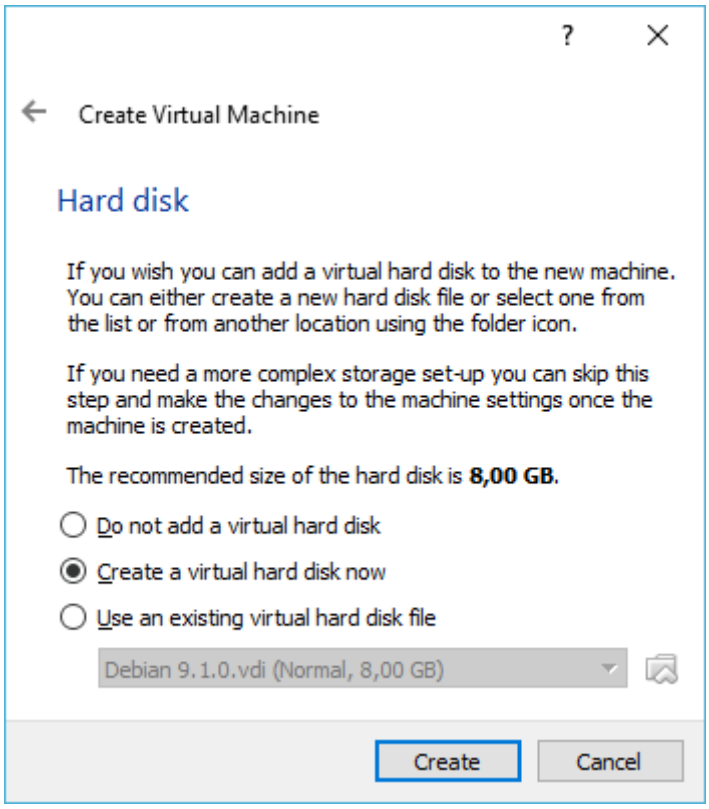

Figura 2. Criação de nova VM, parte 2

Na tela seguinte, de escolha do formato do disco virtual, mantenha a opção-padrão *VDI (VirtualBox Disk Image)*. Em casos específicos em que se deseje interoperabilidade da VM com outros ambientes de virtualização, como VMWare ou Hyper-V, pode ser interessante escolher o formato VMDK. Clique em *Next*.

Agora, iremos selecionar se o espaço disco irá crescer à medida que for usado (*Dynamically allocated*), ou se será completamente alocado quando da sua criação (*Fixed size*). Em ambientes de produção, é geralmente recomendável selecionar a segunda opção, evitando que os dados do disco virtual fiquem fragmentados em pontos diferentes do disco físico, o que pode acarretar lentidão na leitura de dados, especialmente ao usar discos mecânicos. Neste exemplo, mantenha selecionada a opção *Dynamically allocated* e clique em *Next*.

Selecionaremos agora a localização do arquivo de disco virtual e seu tamanho. Não é necessário alterar a primeira opção — já para a segunda, é importante considerar que um *template* de máquina virtual será usado para criar vários tipos diferentes de servidores-alvo. Por esse motivo, é interessante que seu disco seja organizado de forma simples e seja facilmente extensível futuramente: a flexibilidade de adição de novos discos virtuais é especialmente vantajosa, pois permite que criemos uma instalação básica bastante enxuta, e a aumentemos conforme necessário.

Mantenha o valor-padrão de `8 GB` para o tamanho do disco virtual, e clique em *Create*.

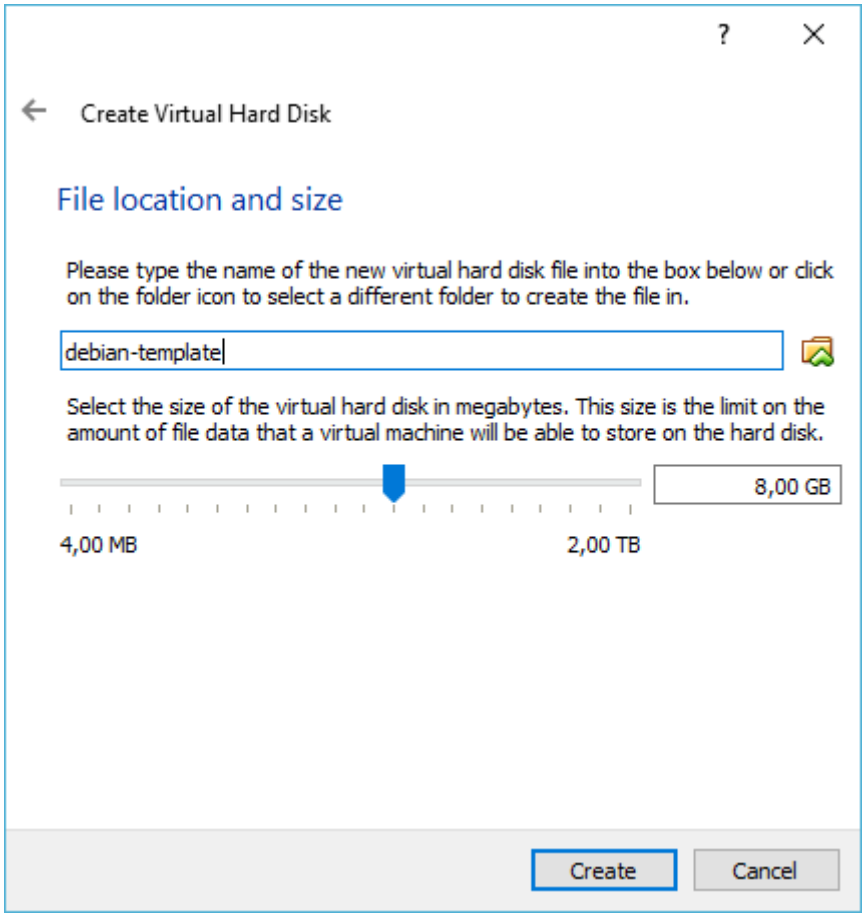

Figura 3. Criação de nova VM, parte 3

4. Será criada uma nova máquina virtual com o nome `debian-template`. Vamos fazer uma rápida pós-configuração antes de iniciar o processo de instalação: clique com o botão direito sobre a VM, e em seguida em *Settings*.

Selecione *Storage* > *Controller: IDE* > *Empty*, e na parte à direita da janela clique no pequeno ícone de um CD em frente à opção *Optical Drive*. Em seguida, clique em *Choose Virtual Optical Disk File...* e navegue pelo sistema de arquivos, selecionando a imagem ISO de instalação do Debian Linux como mostrado abaixo.

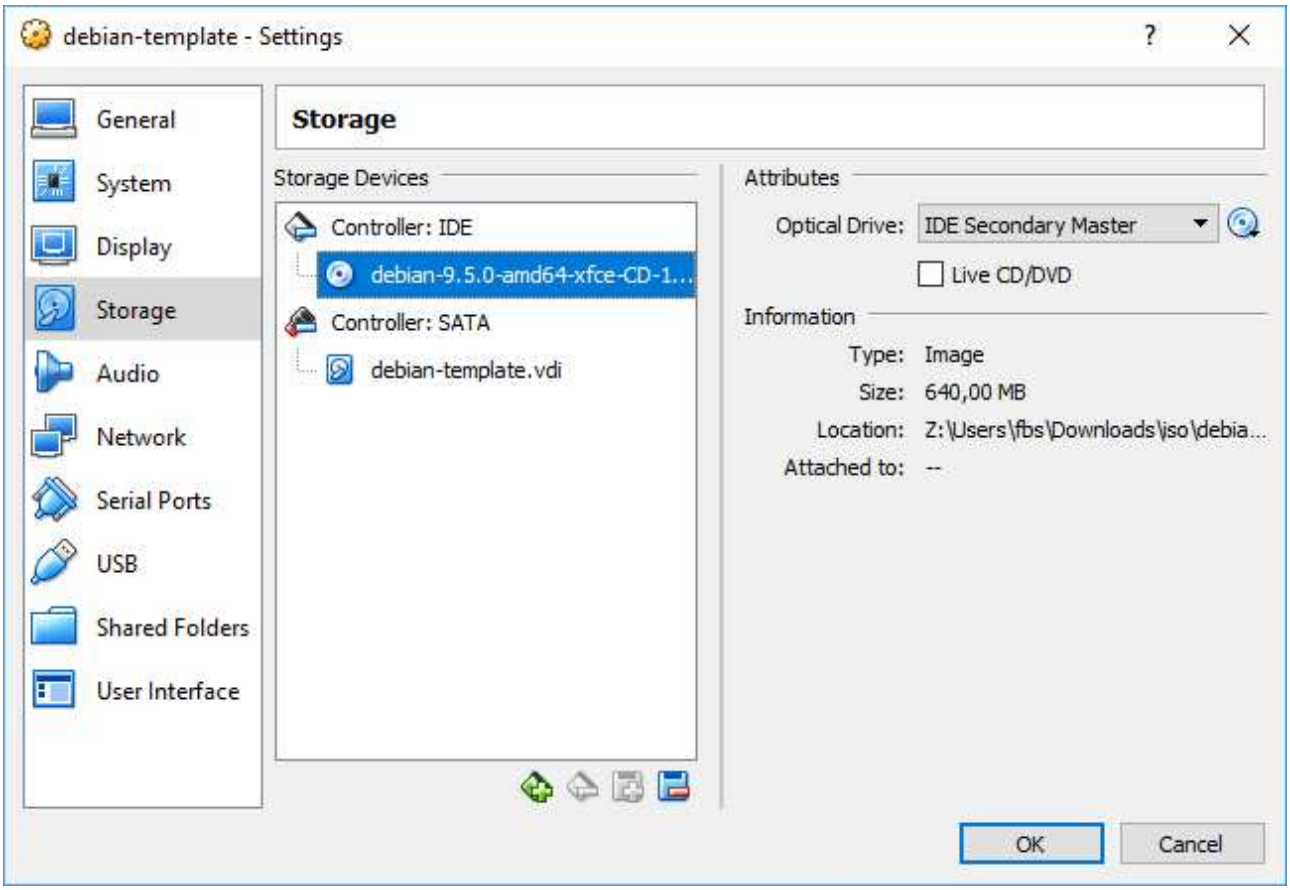

Figura 4. Configuração de nova VM

Em *Audio*, desmarque a caixa *Enable Audio*. Como iremos criar um conjunto de máquinas virtuais que atuarão exclusivamente como servidores, não há necessidade de dispor de áudio nas VMs.

Em *Network* > *Adapter 1* > *Attached to*, altere a conexão de rede da máquina virtual para *Bridged Adapter*. Em *Name*, verifique que a placa de rede física conectada à rede externa está selecionada (isso é especialmente importante em máquinas que possuem múltiplas placas de rede ou interfaces *wireless*). Se desejar, expanda *Advanced* e clique no pequeno círculo azul à direita de *MAC Address* para randomizar um novo endereço físico para a placa de rede da máquina virtual, especialmente útil em casos de conflito de IP.

Em *USB*, marque a caixa *USB 1.1 (OHCI Controller)*. Esta configuração evita que sejam levantados erros ao iniciar a VM caso as extensões do Virtualbox não estejam instaladas na máquina hospedeira.

Finalmente, clique em *OK*.

2) Instalação do Debian Linux

1. Selecione a máquina virtual `debian-template` e clique no botão *Start* para iniciá-la. Apos um curto período, você verá o menu de *boot* do Debian Linux; selecione a opção *Install* para começar o instalador no modo texto.

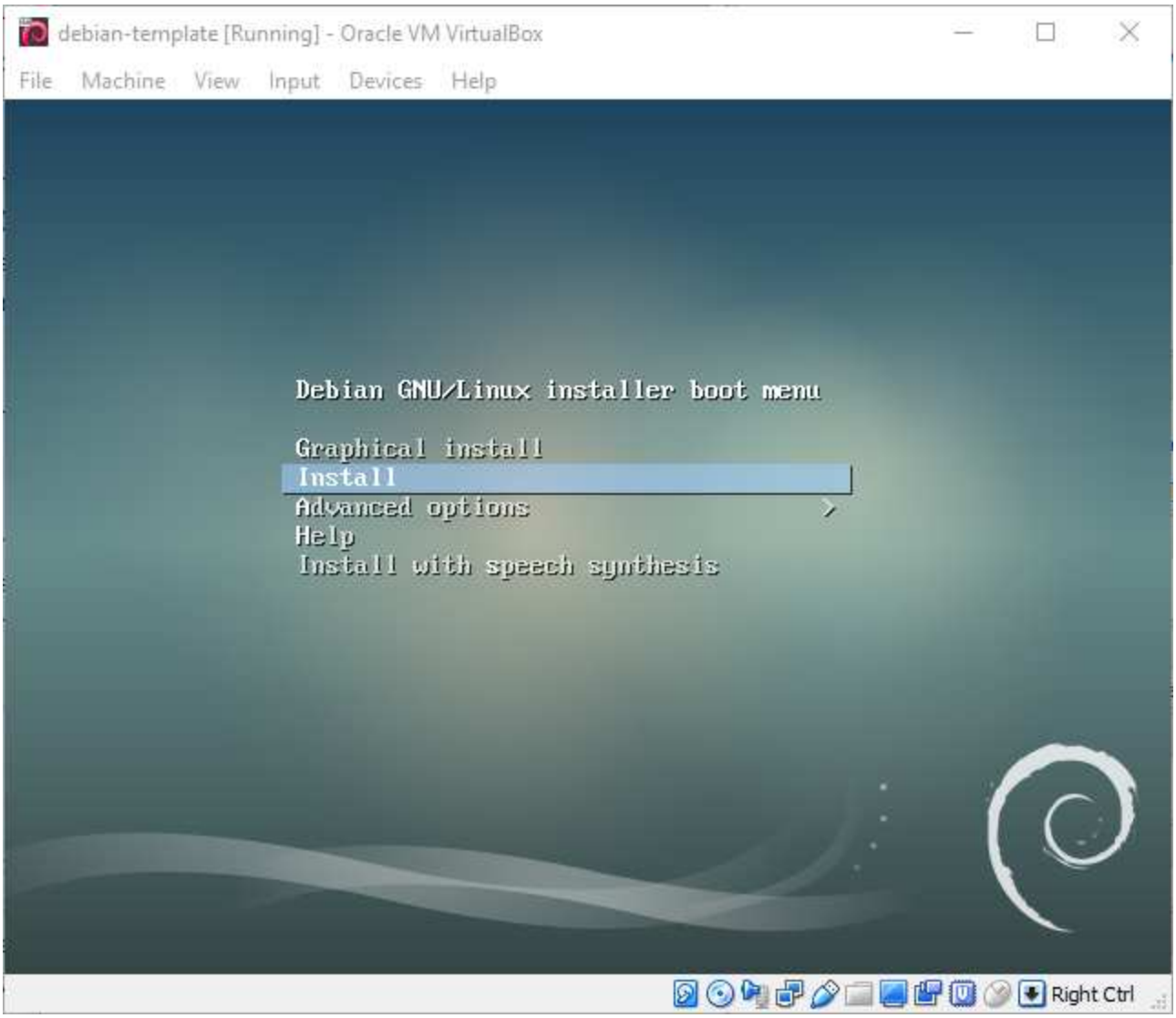

Figura 5. Instalação do Debian Linux, parte 1

2. No passo de seleção de idioma, selecione *Portuguese (Brazil)*. Em localidade, selecione *Brasil*. Para o mapa de teclado a ser usado, provavelmente será o *Português Brasileiro* (verifique se há a tecla `ç` ao lado do caractere `l`).

   Os componentes do instalador serão carregados a seguir.

3. Em seguida, o instalador irá tentar autoconfigurar a rede usando DHCP. Caso esse protocolo não esteja disponível em sua rede local, consulte o instrutor sobre como proceder com a configuração manual das interfaces de rede.

   Configurada a rede, iremos escolher o *hostname* da máquina. Defina o mesmo nome usado para a máquina virtual, `debian-template`.

   Para o nome de domínio da rede, iremos usar a rede local fictícia `intnet` durante o curso.

4. Agora, iremos definir a senha do `root`, o superusuário em sistemas Linux. É bastante recomendado que se defina uma senha especialmente segura, já que esse usuário possui permissões totais sobre o sistema. Por simplicidade e homogeneidade no ambiente de curso, defina a senha como `rnpesr`.

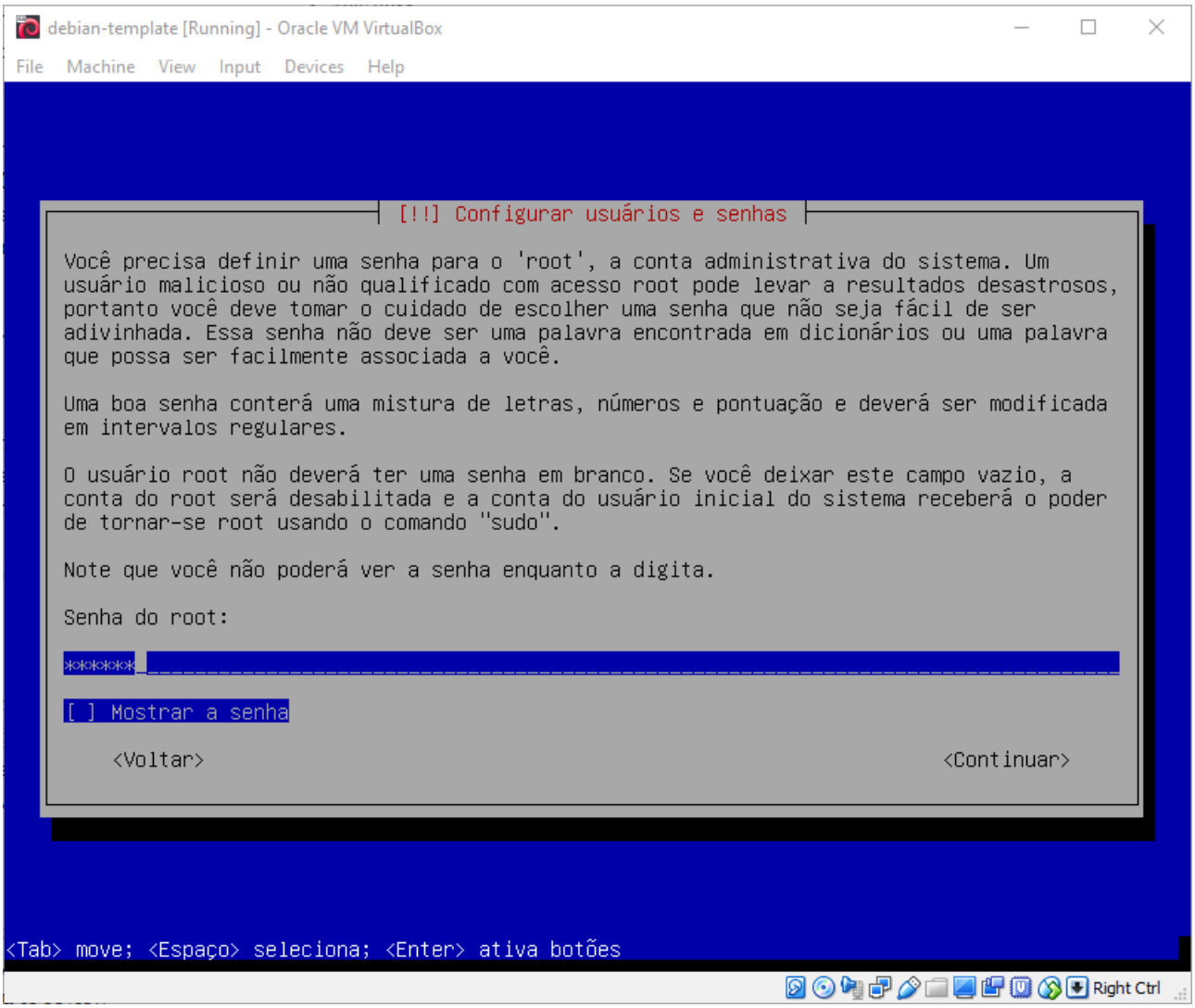

Figura 6. Instalação do Debian Linux, parte 2

Na tela seguinte, confirme a senha.

5. O próximo passo é criar um usuário não-privilegiado para tarefas corriqueiras do sistema. Para o nome completo do usuário, digite `aluno` — este também será o nome de conta do usuário.

   Para a senha, defina de igual forma `rnpesr`.

6. O instalador irá tentar obter a hora via Internet através do protocolo NTP. Em seguida, teremos que escolher um estado para definir o fuso horário do sistema. Escolha o estado em que você está realizando este curso.

7. Agora, faremos o particionamento do disco. Temos quatro opções: particionamento assistido usando o disco inteiro, assistido usando o disco inteiro com LVM, assistido com o disco inteiro e LVM criptografado e particionamento manual. Se tivéssemos mais de um disco virtual conectado à máquina, o instalador ofereceria também a opção de configuração assistida de RAID (*Redundant Array of Independent Disks*).

Mas, o que é LVM?

O *Logical Volume Manager* (LVM) é um sistema de mapeamento de dispositivos do Linux que permite a criação e gestão de volumes lógicos de armazenamento. As utilidades da gestão de armazenamento via volumes lógicos são muitas, destacando-se:

- Criação de volumes lógicos únicos englobando diferentes volumes físicos ou discos físicos inteiros, permitindo redimensionamento dinâmico de volumes.
- Gestão facilitada de grandes quantidades de discos físicos, permitindo que discos sejam adicionados ou substituídos sem *downtime* ou impacto à disponibilidade — especialmente útil quando combinado com hardware que suporta *hot swapping*.
- Em pequenos sistemas (como *desktops* e estações de trabalho) permite que o administrador não tenha que estimar no passo de instalação quão grande uma partição irá se tornar, permitindo redimensionamento dinâmico futuro.
- Criação de backups consistentes através de *snapshots* de volumes lógicos.
- Criptografar múltiplas partições físicas com uma mesma senha.

Em essência, o LVM traz enorme flexibilidade ao administrador de sistemas, resolvendo muitos dos problemas de particionamento que tínhamos no passado. Ele possui alguns conceitos centrais, ilustrados pela imagem a seguir:

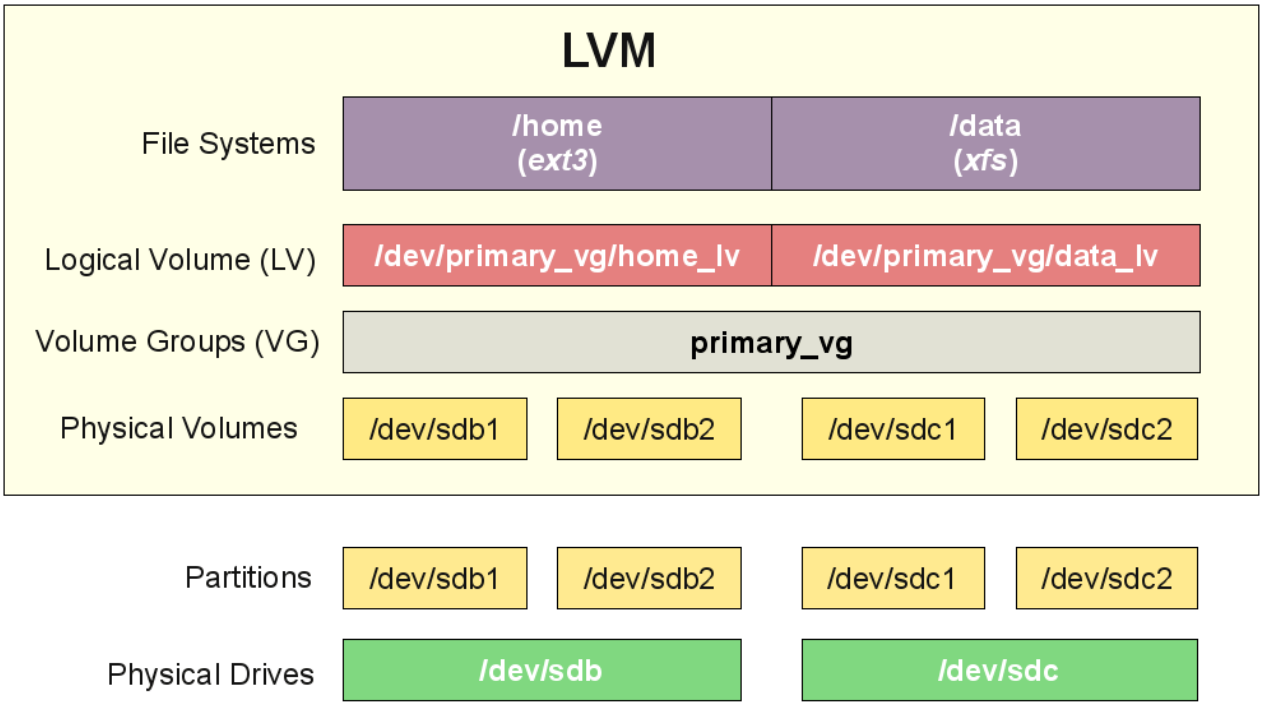

Figura 7. Organização do LVM

Na base do sistema temos os discos físicos conectados à máquina — como `/dev/sdb` ou `/dev/sdc`, por exemplo — que podem ser particionados (em formato MBR ou GPT) em múltiplas partições. Essas partições são denominadas volumes físicos (*Physical Volumes*, ou PVs). Vários PVs podem ser aglutinados para definir um grupo de volumes (*Volume Groups*, ou VGs), que é um agrupamento lógico desses PVs sob um mesmo nome. Pode-se então criar vários volumes lógicos (*Logical Volumes*, ou LVs) dentro desse VG, e finalmente formatar e montar diretórios dentro dos LVs, já no contexto do sistema de arquivos.

A explicação acima é propositalmente sucinta; iremos entrar em maior detalhe com relação ao funcionamento e operação do LVM em atividades subsequentes.

De volta ao instalador, iremos configurar um particionamento manual usando LVM. Por isso, na tela *Método de particionamento*, selecione *Manual*.

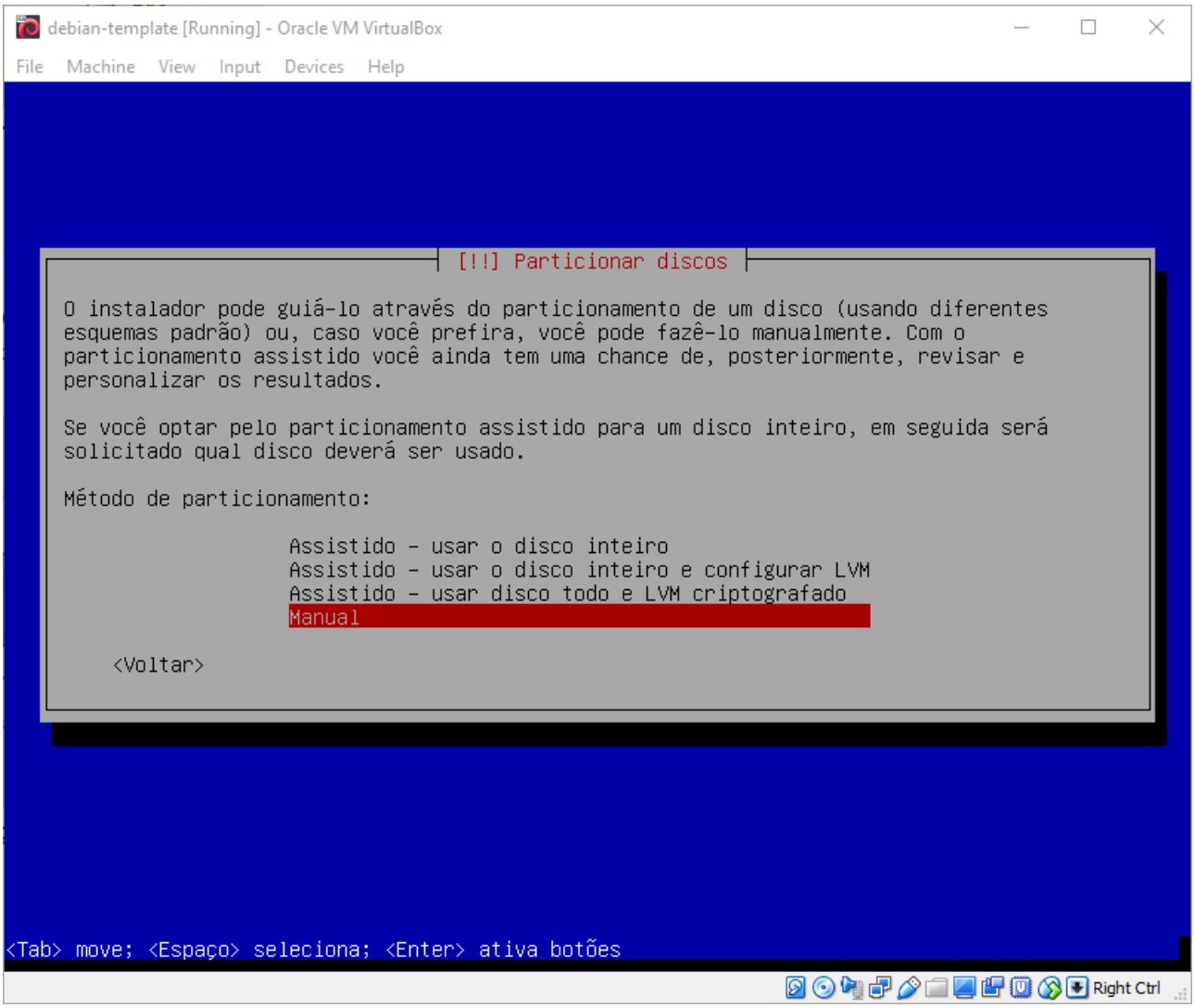

Figura 8. Instalação do Debian Linux, parte 3

8. Na tela seguinte, o primeiro passo é criar uma tabela de partições vazia no disco virtual `/dev/sda`. Coloque o cursor sobre o disco *SCSI1 (0,0,0) (sda) - 8.6 GB ATA VBOX HARDDISK* e pressione `ENTER`. Em seguida, responda *Sim* para a pergunta *Criar nova tabela de partições vazia neste dispositivo?*.
9. Agora, iremos configurar o LVM. Selecione a opção *Configurar o Gerenciador de Volumes Lógicos*, e responda *Sim* para a pergunta *Gravar as mudanças nos discos e configurar LVM?*.

10. Você verá a tela de configuração do LVM, como se segue.

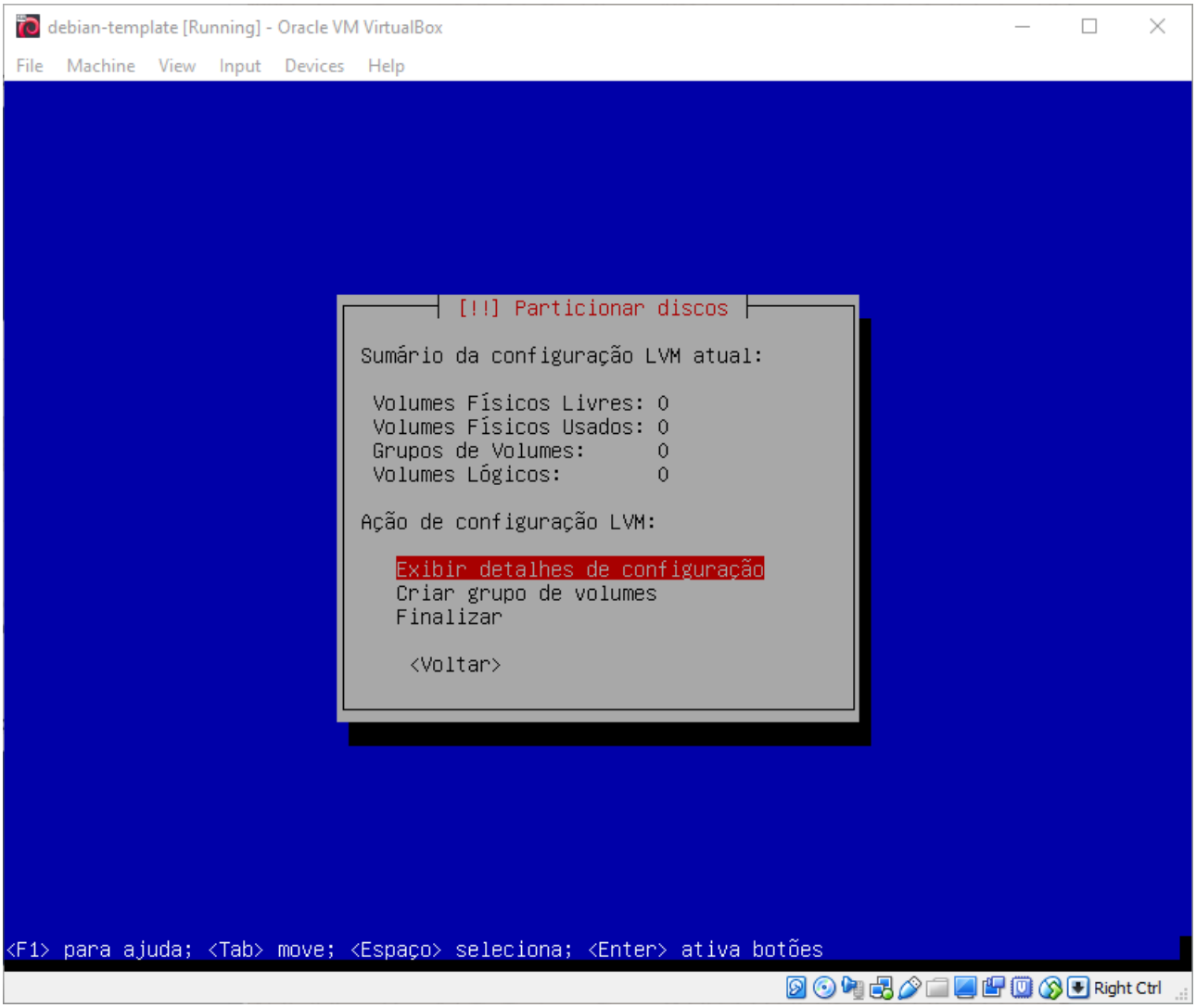

Figura 9. Instalação do Debian Linux, parte 4

A qualquer momento, você pode selecionar a opção *Exibir detalhes de configuração* para verificar o estado atual de configuração do LVM.

O primeiro passo é criar um VG, então escolhar *Criar grupo de volumes*. Para o nome do grupo, digite `vg-base`. Em seguida, marque com a tecla `Espaço` os volmes físicos que integrarão esse VG (no caso, apenas o dispositivo `/dev/sda` está disponível), e finalmente responda *Sim* para a pergunta *Gravar as mudanças nos discos e configurar LVM?*. Ao exibir os detalhes de configuração após este passo, você deverá ver a tela a seguir:

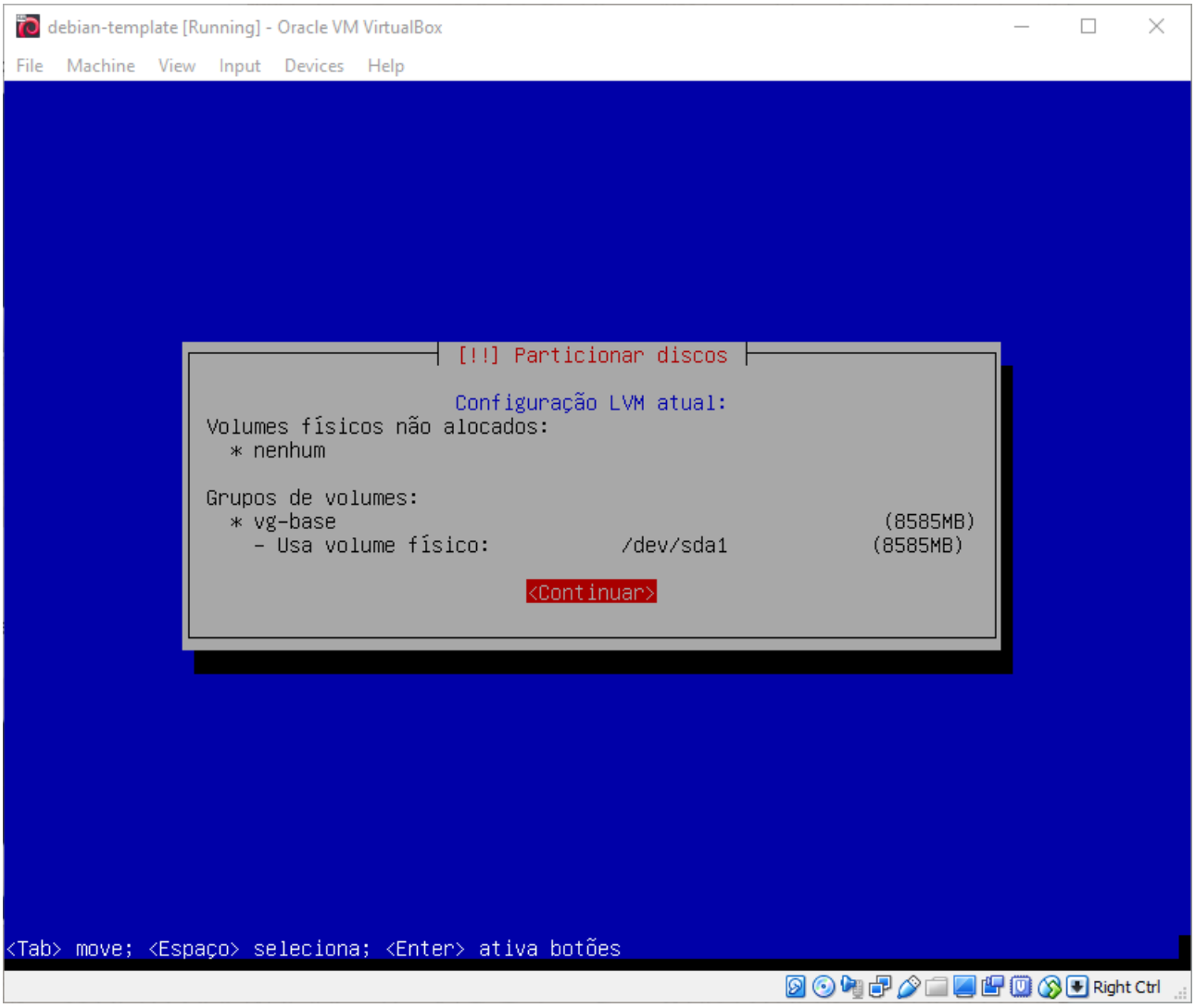

Figura 10. Instalação do Debian Linux, parte 5

11. Todos os volumes físicos (PVs) foram alocados a grupos de volumes (VGs), então agora nos resta criar volumes lógicos (LVs). Com efeito, neste momento é como se estivéssemos particionando um disco no Linux em uma instalação tradicional. Iremos criar três LVs:

    - `lv-boot`: LV que irá armazenar o diretório `/boot` do sistema, com tamanho de 256 MB. É interessante separar o `/boot` para reduzir a complexidade do sistema de arquivos em disco, bem como para aplicar configurações diferenciadas ao *filesystem*, como RAID por software, sistemas de arquivos não-usuais como ZFS, ou caso se deseje criptografar a raiz do sistema.
    - `lv-swap`: LV que irá servidor como área de troca (*swap*) do SO em caso de escassez de memória física. Como idealmente não queremos chegar nesse cenário, alocaremos apenas 256 MB para essa área.
    - `lv-root`: LV quer irá armazenar a raiz do sistema, `/`, com tamanho de 1536 MB. Pode parecer um valor pequeno, mas considere que iremos separar outros sistemas de arquivos posteriormente.

Imediatamente, podem surgir duas perguntas:

1. **Por que não estamos alocando a totalidade do disco?** O LVM nos permite grande flexibilidade, que será demonstrada em atividades subsequentes. Ao alocarmos [256 + 256 + 1536] = 2048 MB em um disco de 8 GB, deixamos (aproximadamente) 6 GB livres que poderão ser alocados de acordo com o tipo específico de uso de cada máquina. Em sentido estrito, a alocação que fizemos acima não é exatamente ideal para um *template*, mas iremos corrigir isso ao demonstrar as funcionalidades do LVM, a seguir.
2. **Por que não criamos LVs para partições como `/tmp`, `/usr` ou `/var`?** De fato, é bastante recomendável separar essas partições em servidores, como veremos a seguir. Iremos criar esses LVs brevemente, após a instalação do SO.

Para criar um LV, selecione *Criar volume lógico*. Em seguida, selecione o grupo de volumes no qual será feita a alocação (no caso, apenas `vg-base` está disponível). Para o nome do primeiro volume lógico, digite `lv-boot`, como delineado acima. Para seu tamanho, defina 256 MB.

Prossiga com a criação dos outros dois LVs, `lv-swap` e `lv-root`, com os tamanhos especificados acima. Ao exibir os detalhes de configuração após este passo, você deverá ver a tela a seguir:

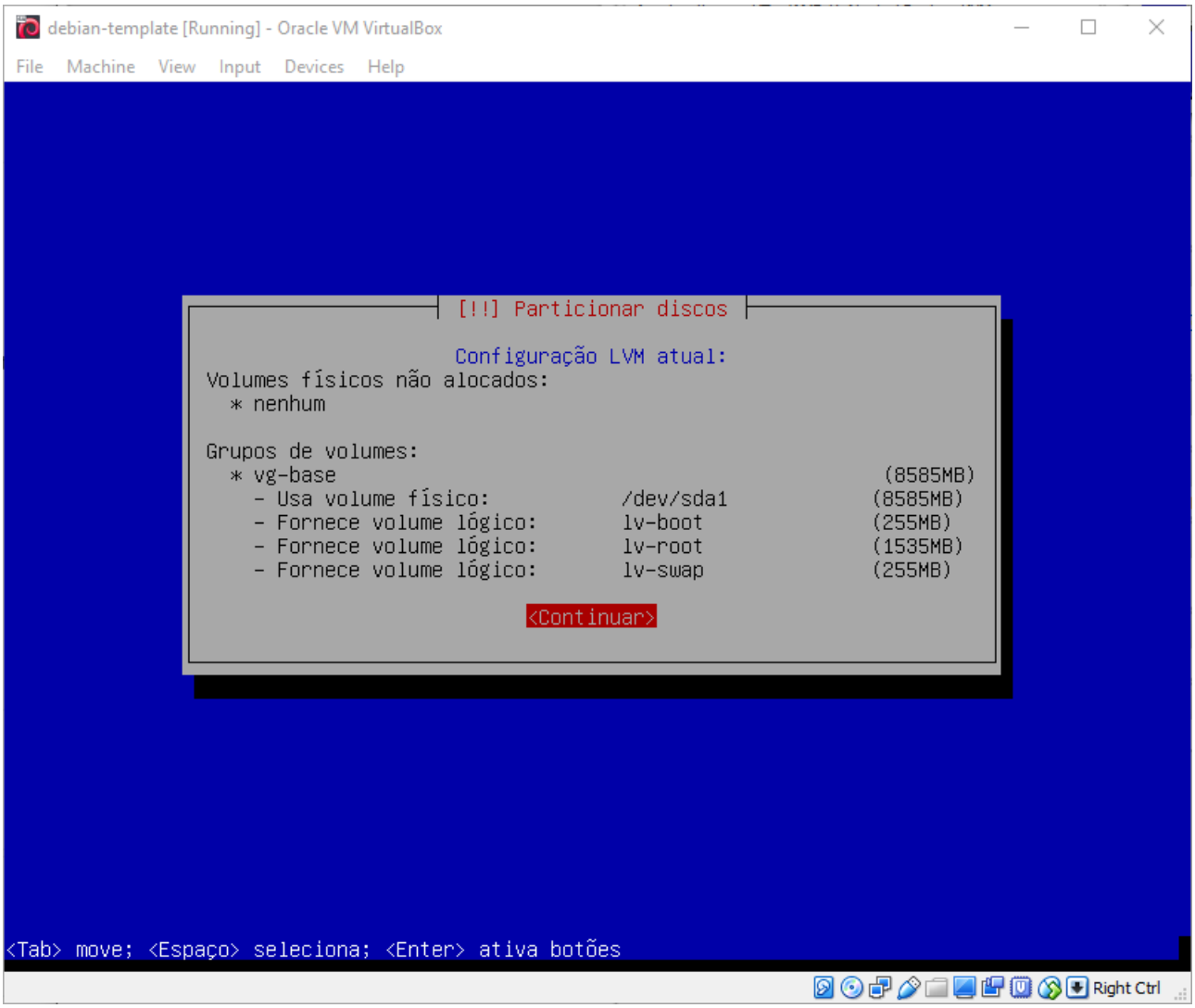

Figura 11. Instalação do Debian Linux, parte 6

Se tudo estiver a contento, selecione *Finalizar*.

12. Ainda não acabou! Neste momento, configuramos o LVM — alocamos volumes físicos, criamos um grupo de volumes agrupando esses PVs e finalmente criamos 3 volumes lógicos dentro do VG. Falta informar ao sistema quais serão os pontos de montagem desses LVs, e quais sistemas de arquivos serão usados. Usaremos a seguinte configuração:

    - `lv-boot`: montado sob o diretório `/boot`, formatado em `ext2`.
    - `lv-swap`: área de troca (*swap*).
    - `lv-root`: montado sob o diretório `/`, formatado em `ext4`.

    Para fazer as configurações acima, selecione um dos LVs indicados (por exemplo, deixe o cursor sobre a linha `#1 255.9 MB`, logo abaixo de `lv-boot`), e pressione `ENTER`. Em *Usar como*, escolha *Sistema de arquivos ext2*, e em *Ponto de montagem* selecione `/boot`. Finalmente, selecione *Finalizar a configuração da partição*.

    Prossiga com a configuração dos outros dois LVs, `lv-swap` e `lv-root`, de acordo com as características especificadas acima. Ao exibir os detalhes de configuração após este passo, você deverá ver a tela a seguir:

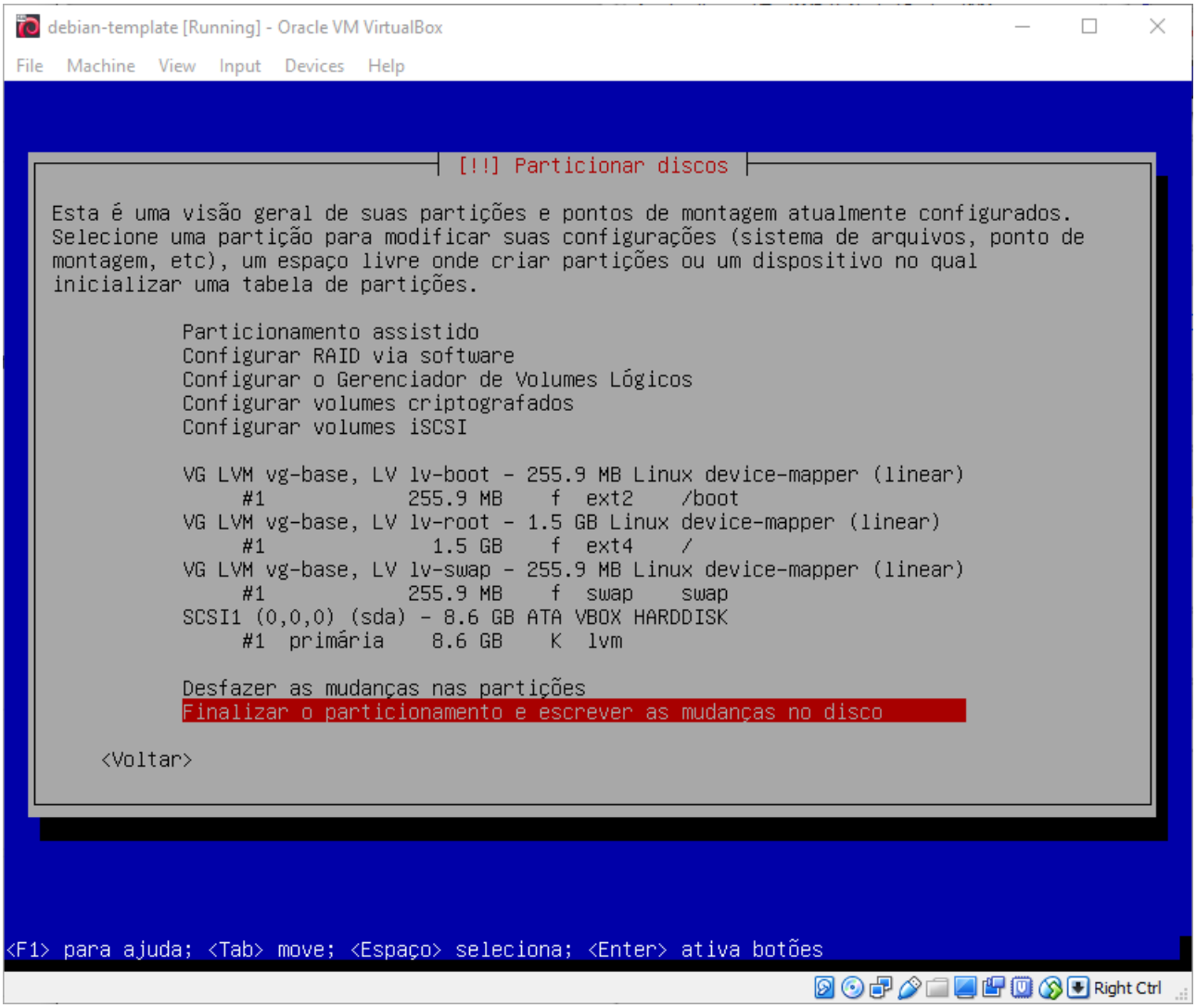

Figura 12. Instalação do Debian Linux, parte 7

Se tudo estiver a contento, selecione *Finalizar o particionamento e escrever as mudanças no disco*. Confirme a pergunta subsequente escolhendo *Sim*. Os discos serão formatados e o

sistema-base do Debian será instalado.

13. O próximo passo será a seleção e instalação de pacotes adicionais. Na pergunta *Selecionar um espelho de rede*, responda *Sim*. Em seguida, selecione *Brasil* como o país do espelho e aponte o servidor `ftp.br.debian.org`. Quanto à informação do proxy HTTP a ser usado, deixe em branco (a menos que o contrário seja indicado pelo seu instrutor).

    O instalador irá fazer o download dos arquivos de índice do repositório de pacotes.

    Após algum tempo, surgirá a pergunta *Participar do concurso de utilização de pacotes* — responda *Não*. Em seguida, o `tasksel` será invocado. Nesta tela podemos escolher quais conjuntos de software iremos instalar no disco.

    Para servidores de rede, em linhas gerais, é usualmente recomendável não selecionar nada além do estritamente necessário nesta tela, e proceder com a instalação manual de pacotes posteriormente; o `tasksel` geralmente instala um conjunto de pacotes superior ao que objetivamos originalmente, aumentando a superfície de ataque e exposição do sistema. Para ambientes *desktop*, é perfeitamente razoável escolher o ambiente gráfico base e uma das opções de gerenciadores de janelas disponíveis.

Mantenha marcadas apenas as caixas *servidor SSH* e *utilitários de sistema padrão*, e selecione *Continuar*.

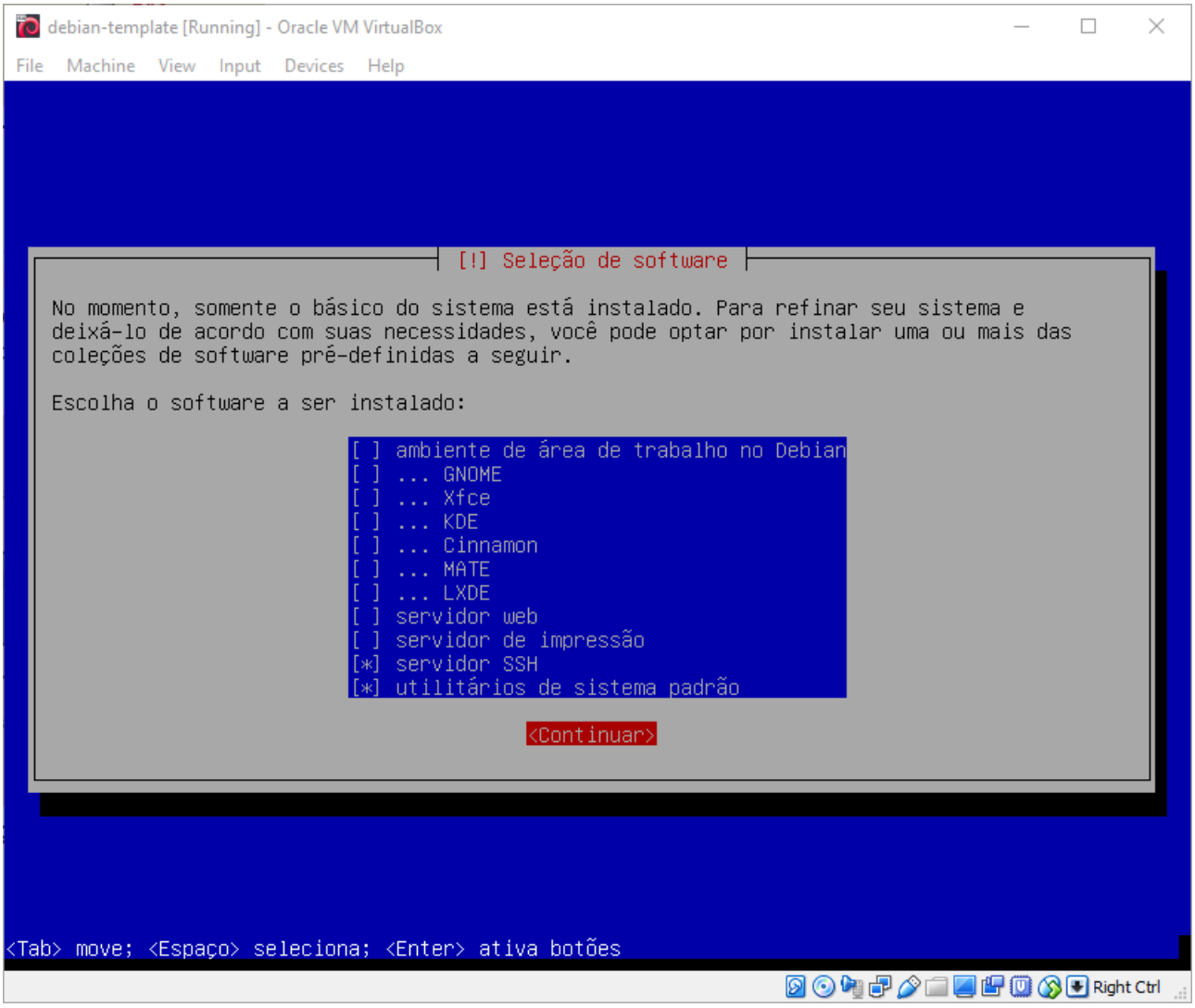

Figura 13. Instalação do Debian Linux, parte 8

O instalador irá fazer o download e instalação dos pacotes selecionados.

14. A última etapa é efetuar a instalação do carregador de inicialização, ou *bootloader*, no sistema. O Debian, assim como a maioria das demais distribuições, utiliza o GRUB (*GRand Unified Bootloader*) como *bootloader* padrão.

    Responda *Sim* para a pergunta *Instalar o carregador de inicialização GRUB no registro mestre de inicialização*, e em seguida selecione o dispositivo de instalação `/dev/sda` (o único disponível).

15. A instalação está concluída. Selecione *Continuar* para reinicializar a VM no novo sistema instalado.

    Após o reboot, você deverá ver a tela de login abaixo:

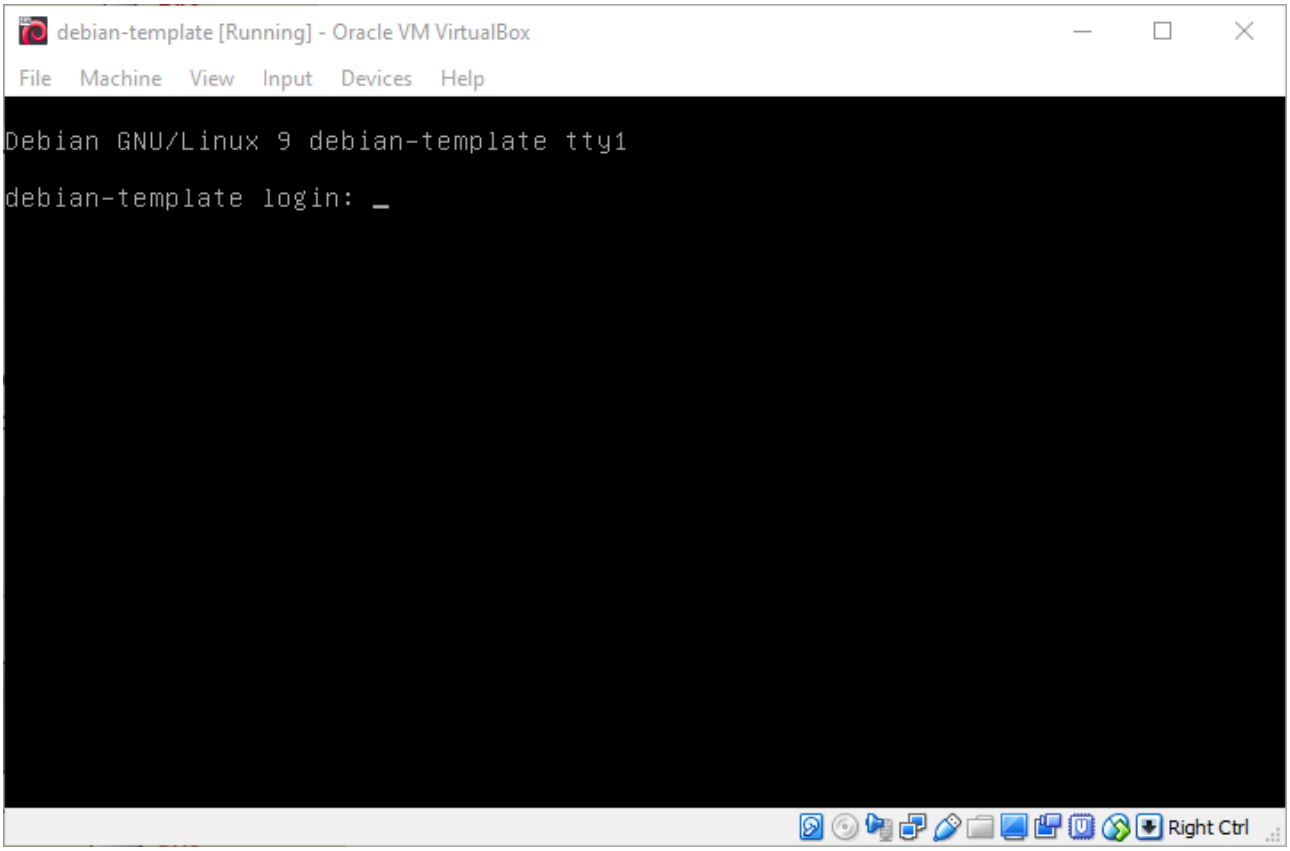

Figura 14. Instalação do Debian Linux, concluída

3) Ajustes pós-instalação

Instalado nosso *template*, iremos continuar sua configuração para torná-lo, de fato, uma imagem-base de boa qualidade para ser usada em derivações de VMs futuras. Além de corrigir a situação dos volumes lógicos (que deixamos incompleta propositalmente durante a instalação), iremos também fazer algumas configurações de base com relação aos repositórios, atualização de pacotes e contas de usuário.

1. Faça login na máquina `debian-template` como usuário `root`, usando a senha `rnpesr`. Imediatamente após o login, você verá uma mensagem parecida com a que se segue:

```
Linux debian-template 4.9.0-7-amd64 #1 SMP Debian 4.9.110-3+deb9u2 (2018-08-13)
x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

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

```
# hostname
debian-template
```

```
# whoami
root
```

Essa mensagem é definida no arquivo `/etc/motd`, conhecida como *message of the day* (ou "mensagem do dia"). É interessante customizar essa mensagem para refletir o ambiente local, avisando o administrador sobre os requisitos legais para operar máquinas da organização, ou informando sobre onde encontrar documentação sobre os servidores. Lembre-se que, já que estamos mexendo no *template*, essa mensagem será copiada para todas as VMs derivadas desta imagem.

Neste curso, vamos deixar o *motd* vazio. Execute:

```
# echo "" > /etc/motd
```

Saia da sessão corrente usando `exit` ou `CTRL + D`, e logue novamente. Note que a mensagem anterior será suprimida.

2. Ao longo do curso, iremos editar vários arquivos de texto em ambiente Linux. Há vários editores de texto disponíveis para a tarefa, como o `vi`, `emacs` ou `nano`. Caso você não esteja familiarizado com um editor de texto, recomendamos o uso do `nano`, que possui uma interface bastante amigável para usuários iniciantes. Para editar um arquivo com o `nano`, basta digitar `nano` seguido do nome do arquivo a editar — não é necessário que o arquivo tenha sido criado previamente:

```
# nano teste
```

Digite livremente a seguir. Use as setas do teclado para navegar no texto, e `DELETE` ou `BACKSPACE` para apagar texto. O `nano` possui alguns atalhos interessantes, como:

- `CTRL + G`: Exibir a ajuda do editor
- `CTRL + X`: Fechar o `buffer` de arquivo atual (que pode ser um texto sendo editado, ou o painel de ajuda), e sair do `nano`. Para salvar o arquivo, digite `Y` (*yes*) ou `S` (*sim*) para confirmar as mudanças ao arquivo, opcionalmente altere o nome do arquivo a ser escrito no disco, e digite `ENTER`.
- `CTRL + O`: Salvar o arquivo no disco sem sair do editor.
- `CTRL + W`: Buscar padrão no texto.
- `CTRL + K`: Cortar uma linha inteira e salvar no `buffer` do editor.
- `CTRL + U`: Colar o `buffer` do editor na posição atual do cursor. Pode ser usado repetidamente.

Para salvar e sair do texto sendo editado, como mencionado acima, utilize `CTRL + X`.

3. Edite o arquivo `/etc/network/interfaces` como se segue, reinicie a rede e verifique o funcionamento:

```
# nano /etc/network/interfaces
(...)
```

```
# cat /etc/network/interfaces
source /etc/network/interfaces.d/*

auto lo enp0s3

iface lo inet loopback

iface enp0s3 inet dhcp
```

```
# systemctl restart networking
```

```
# ip a s | grep '^ *inet '
    inet 127.0.0.1/8 scope host lo
    inet 192.168.29.104/24 brd 192.168.29.255 scope global enp0s3
```

Desabilite a verificação de *hostnames* remotos do `ssh` para agilizar o procedimento de login. Reinicie o *daemon* posteriormente.

```
# sed -i '/UseDNS/s/^#//' /etc/ssh/sshd_config
```

```
# systemctl restart ssh
```

4. Durante as atividades deste curso iremos ter que digitar vários comandos no terminal das VMs, os quais serão mostrados nos cadernos de atividade de cada sessão. Alguns desses comandos serão bastante longos e/ou terão uma sintaxe complicada — nesse caso, o ideal é que tenhamos a possibilidade de copiá-los diretamente do caderno para a console, evitando erros de digitação.

   O protocolo de login remoto SSH é ideal para solucionar essa tarefa. Em ambiente Windows, dois dos métodos mais populares para efetuar logins remotos via SSH são os programas PuTTY (https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html) ou Cygwin (https://cygwin.com/install.html). Vamos, primeiro, visualizar os passos necessários usando o PuTTY.

   Em qualquer caso, o primeiro passo é sempre descobrir qual o endereço IP da máquina remota à qual queremos nos conectar. Para isso digite, na máquina `debian-template`:

```
# ip a s enp0s3 | grep '^ *inet ' | awk '{print $2}' | cut -d'/' -f1
192.168.29.104
```

O uso do **PuTTY**, por se tratar de um programa *standalone* com o objetivo único de efetuar login via SSH, é mais simples. Faça o download do PuTTY em sua máquina física Windows, usando a URL informada acima. Em seguida, apenas abra o programa e digite na caixa *Host Name* o endereço IP da máquina remota descoberto acima. Em seguida, clique em *Open*.

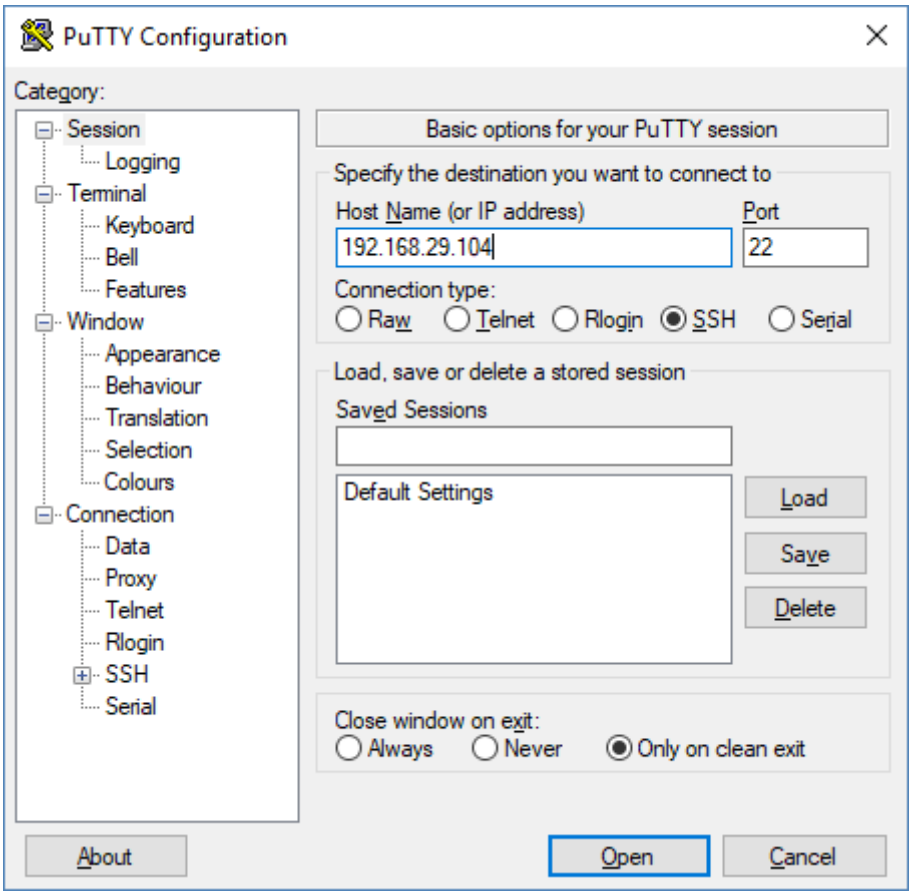

Figura 15. Login via SSH usando o PuTTY, parte 1

Será mostrado um alerta de segurança avisando que a chave do *host* remoto não se encontra na *cache* local, o que pode configurar um risco de segurança. Clique em *Yes* para prosseguir com a tentativa de login.

Em seguida, será solicitado o nome de usuário com o qual efetuar a conexão. Como, por padrão, o *daemon* `sshd` do Debian não permite logins remotos usando o `root`, escolha o usuário `aluno`. Para a senha, informe `rnpesr`. Em caso de sucesso, você verá a tela a seguir:

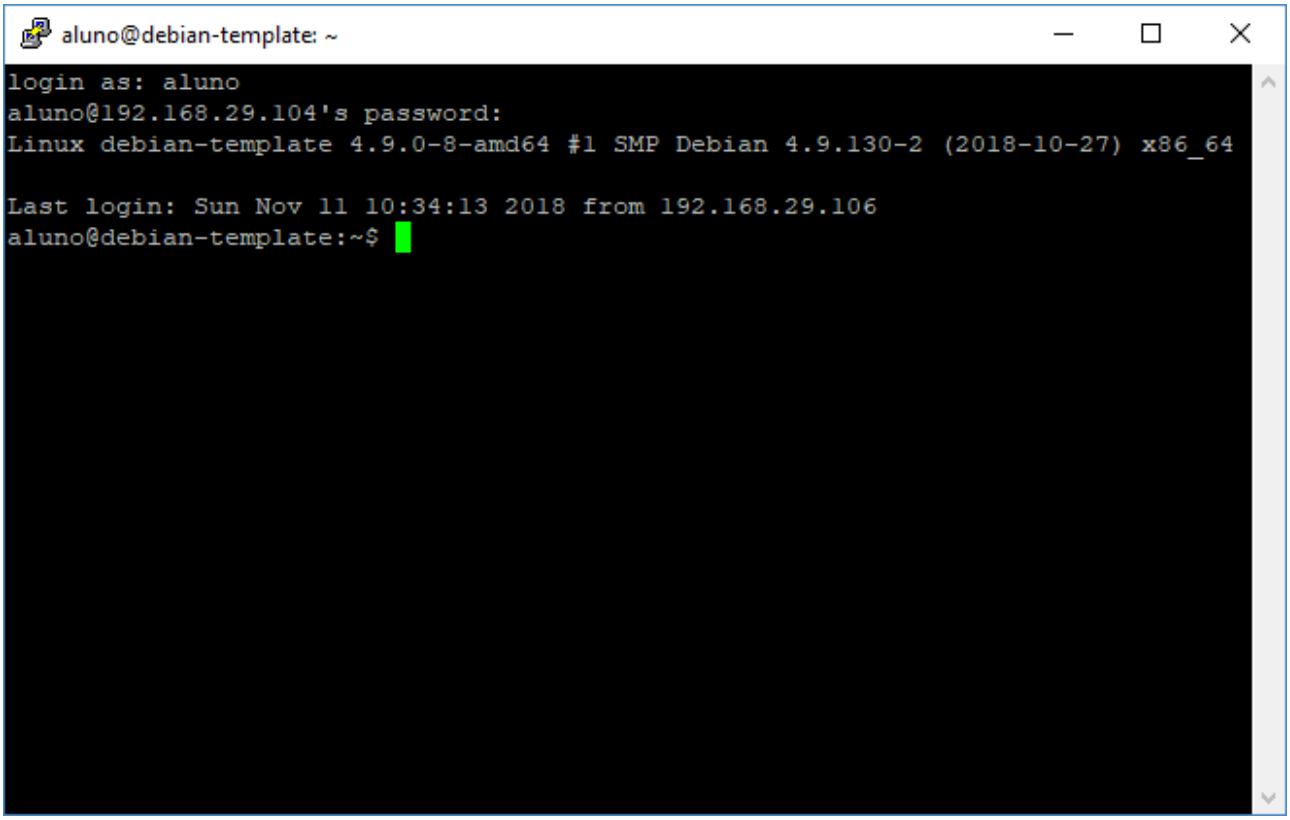

Figura 16. Login via SSH usando o PuTTY, parte 2

Para tornar-se o superusuário `root`, agora, basta executar o comando `su -` e informar a senha correta, como se segue:

```
$ su -
Senha:
```

```
# whoami
root
```

Para copiar/colar comandos no PuTTY, basta selecionar o texto desejado no ambiente da máquina física e digitar `CTRL + C`, e em seguida clicar com o botão direito na janela do PuTTY. O texto selecionado será colado na posição do cursor.

5. O uso do Cygwin é um pouco mais envolvido, já que seu objetivo é mais complexo: prover, em ambiente Windows, funcionalidade equivalente à que temos disponível em uma distribuição Linux. Para começar, faça o download e execute o instalador do Cygwin em sua máquina física Windows.

   A instalação é, em grande parte, bastante similar à de qualquer aplicativo Windows. Na tela inicial, clique em *Next*. Em *Choose a Download Source*, mantenha marcada a caixa *Install from Internet* e clique em *Next*. Em *Select Root Install Directory*, os valores padrão estão apropriados — clique em *Next*. Na tela *Select Local Package Directory*, novamente, mantenha o valor padrão e clique em *Next*.

   Agora, vamos selecionar a fonte de pacotes. Em *Select Your Internet Connection*, a menos que

haja um *proxy* na rede local (informe-se com seu instrutor), mantenha marcada a caixa *Direct Connection* e clique em *Next*. Será feito o download da lista de espelhos disponíveis para o Cygwin. Em *Choose A Download Site*, qualquer espelho irá funcionar, mas evidentemente é desejável que escolhamos um que possua maior velocidade de download — o site http://linorg.usp.br é provavelmente uma boa opção, nesse caso. Clique em *Next*, e o instalador irá baixar a lista de pacotes disponíveis.

Em adição ao sistema-base padrão, é necessário instalar o OpenSSH para efetuar logins remotos. Na caixa de busca *Search*, no topo da tela, digite o termo de busca `openssh`. Expanda a árvore `Net` e clique na palavra *Skip* na linha do pacote `openssh: The OpenSSH server and client programs` — ela irá alterar para a versão a ser instalada, `7.9p1-1` no caso da figura mostrada abaixo:

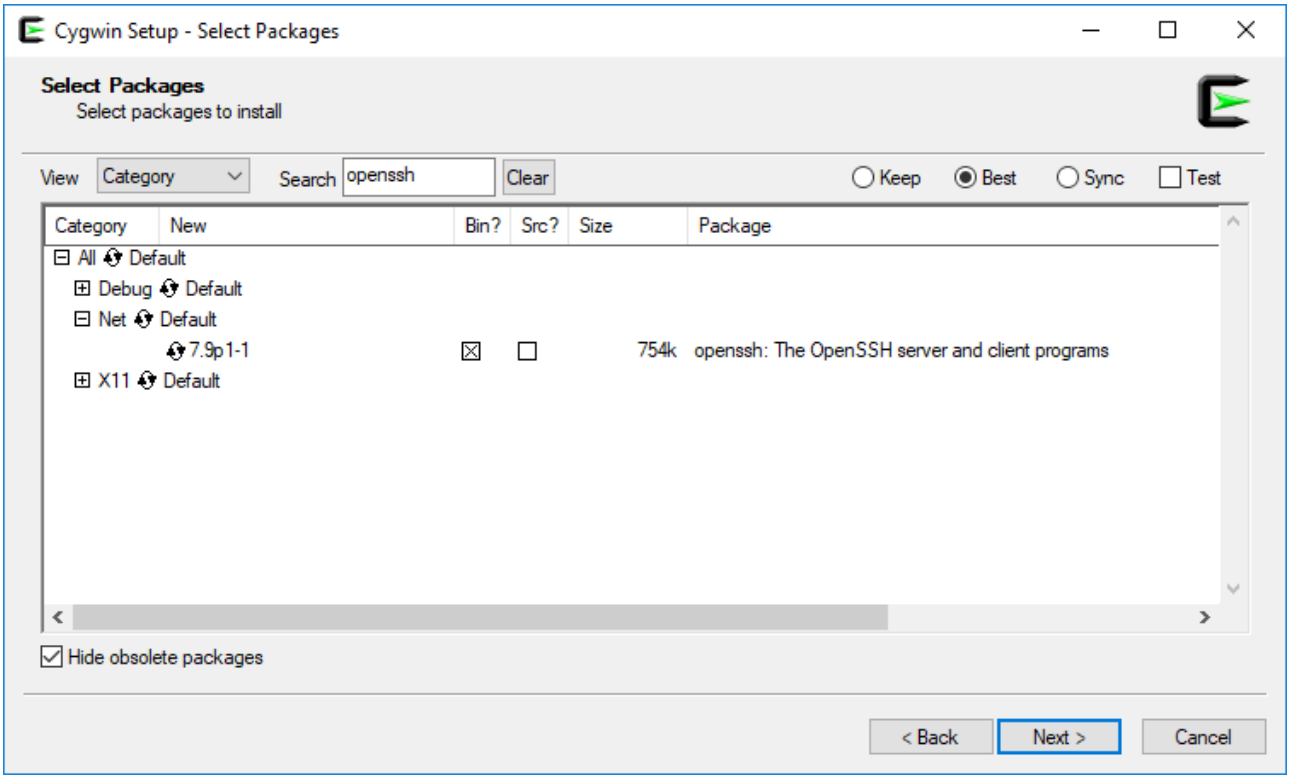

Figura 17. Instalação do OpenSSH no Cygwin

Clique em *Next*. Em *Review and confirm changes*, verifique que o Cygwin irá instalar o OpenSSH e todas as demais dependências do sistema-base Linux, como o *shell* `bash` ou ferramentas como o `grep`, e clique em *Next*. O instalador irá fazer o download e instalação dos pacotes selecionados.

Concluído o processo, procure pelo programa `Cygwin Terminal` no menu iniciar da sua máquina física Windows, e execute-o. Agora, tente fazer login via SSH normalmente, como se estivesse em um *shell* Linux:

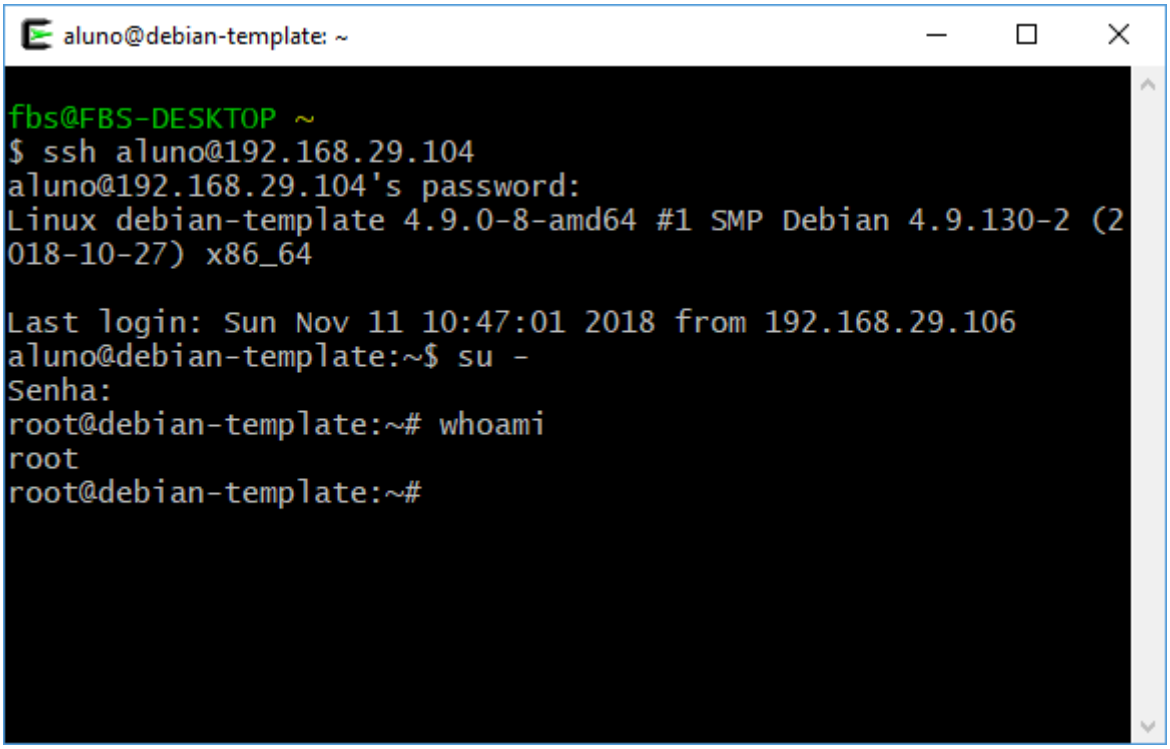

Figura 18. Login via SSH usando o Cygwin

Para copiar/colar comandos no Cygwin, basta selecionar o texto desejado no ambiente da máquina física e digitar `CTRL + C`, e em seguida mudar o foco para a janela do Cygwin e digitar a combinação `SHIFT + Insert`. Para copiar texto a partir da janela do Cygwin, selecione-o e use a combinação de teclas `CTRL + Insert`. Para encontrar os arquivos localizados em sua máquina física, o diretório `/cygdrive/X` pode ser usado para mapear para os discos da máquina local — por exemplo, o diretório `/cygdrive/c` mapeia diretamente para o `C:\` da máquina Windows.

6. A seguir, vamos checar a configuração dos repositórios de pacotes sendo usados pelo sistema. Cheque o conteúdo do arquivo `/etc/apt/sources.list`:

```
# cat /etc/apt/sources.list
#

# deb cdrom:[Debian GNU/Linux 9.5.0 _Stretch_ - Official amd64 xfce-CD Binary-1
20180714-10:25]/ stretch main

deb cdrom:[Debian GNU/Linux 9.5.0 _Stretch_ - Official amd64 xfce-CD Binary-1
20180714-10:25]/ stretch main

deb http://ftp.br.debian.org/debian/ stretch main
deb-src http://ftp.br.debian.org/debian/ stretch main

deb http://security.debian.org/debian-security stretch/updates main
deb-src http://security.debian.org/debian-security stretch/updates main

# stretch-updates, previously known as 'volatile'
deb http://ftp.br.debian.org/debian/ stretch-updates main
deb-src http://ftp.br.debian.org/debian/ stretch-updates main
```

Temos algumas linhas desnecessárias neste arquivo: primeiro, as entradas `deb cdrom` referem-se ao CD de instalação do Debian, que usamos durante a atividade (2) desta sessão; além dessas, as entradas `deb-src` referem-se a pacotes de código-fonte, que podem ser baixados com o comando `apt-get source` e posteriormente compilados pelo administrador. Via de regra, não usaremos quaisquer dessas entradas em um sistema de produção, então elas podem ser removidas com segurança.

Sobre os componentes (ou seções) do repositório, note que apenas a `main` está incluída no arquivo acima. O Debian possui três seções principais:

- `main`: contém apenas software compatível com a DFSG (*Debian Free Software Guidelines*, ou diretivas de software livre do Debian), e não dependem de software fora desta seção para funcionar. Esses são os pacotes considerados parte integrante da distribuição de software do Debian.
- `contrib`: contém pacotes compatíveis com a DFSG, mas que possuem dependências que não estão na seção `main`.
- `non-free`: contém software incompatível com a DFSG.

Em geral, não há grandes restrições para incluir todas as três seções de software detalhadas acima em uma organização. Assim, iremos incluir as duas seções faltantes, `contrib` e `non-free`, no arquivo `/etc/apt/sources.list`. Edite-o usando o `vi` ou o `nano`:

```
# nano /etc/apt/sources.list
(...)
```

Após a edição, seu conteúdo deverá ficar assim:

```
# cat /etc/apt/sources.list
deb http://ftp.br.debian.org/debian/ stretch main contrib non-free
deb http://ftp.br.debian.org/debian/ stretch-updates main contrib non-free
deb http://security.debian.org/debian-security stretch/updates main contrib non-
free
```

7. Atualize a lista de pacotes disponíveis nos repositórios remotos usando o comando:

   ```
   # apt-get update
   ```

   Em seguida, vamos garantir que nosso *template* está plenamente atualizado com:

   ```
   # apt-get dist-upgrade -y
   ```

   Caso o kernel do sistema tenha sido atualizado no processo (pacotes com o nome `linux-image-*`), será necessário reiniciar a máquina para realizar o *boot* com o novo kernel. Faça isso, se for o caso, com o comando `reboot`.

   Após o reinício, logue novamente como o usuário `root`. Para remover os binários dos pacotes recentemente instalados do sistema, execute `apt-get clean`. Se quiser remover as dependências de pacotes que estão instaladas no sistema e já não são mais necessárias, rode `apt-get autoremove`.

   Para remover todos os kernels antigos do sistema (isto é, todos os kernels exceto o que está em execução no momento), você pode executar:

   ```
   # dpkg -l | egrep 'linux-image-[0-9\.-]*-amd64' | awk '{print $2}' | grep -v
   $(uname -r) | xargs apt-get purge -y
   ```

8. Este é um bom momento para instalar pacotes que você acredita serem necessários em todas as máquinas a serem derivadas deste *template*. Pacotes como o `sudo` ou `vim` podem ser boas opções, dependendo das necessidades da sua organização.

   Neste momento, iremos instalar apenas os pacotes listados abaixo. Execute:

   ```
   # apt-get install rsync nfs-common sudo dirmngr
   ```

9. Pode ser interessante desabilitar a combinação de teclas `CTRL + ALT + DEL`; mesmo em um ambiente virtualizado, há casos em que o administrador se confunde e acaba enviando essa combinação de teclas para a console de um servidor aberto, causando seu *reboot* inadvertidamente.

   Note que, por padrão, essa combinação de teclas aponta para o `reboot.target`, um *target* (ou alvo) do `systemd` que é responsável pelo reinício do sistema operacional.

```
# ls -ld /lib/systemd/system/ctrl-alt-del.target
lrwxrwxrwx 1 root root 13 jun 13 17:20 /lib/systemd/system/ctrl-alt-del.target ->
reboot.target
```

Para desabilitar o `CTRL + ALT + DEL`, basta executar:

```
# systemctl mask ctrl-alt-del.target
Created symlink /etc/systemd/system/ctrl-alt-del.target → /dev/null.
```

10. O *template* atual (a máquina `debian-template`) será usada como base para várias VMs futuras, como estabelecido. Ao copiar a máquina, a primeira ação a ser realizada será sempre alterar o *hostname* para o nome da nova máquina — pode ser muito interessante ter um meio para automatizar essa tarefa, como um *shell script*.

    Crie um novo arquivo `/root/scripts/changehost.sh` (crie o diretório `/root/scripts` se este não existir), com o seguinte conteúdo:

```
#!/bin/bash


usage() {
  echo "  Usage: $0 HOSTNAME"
  exit 1
}


# testar parametros
[ -z $1 ] && usage

# testar sintaxe valida
if [[ "$1" =~ [^a-z0-9] ]]; then
  echo "  HOSTNAME must be lowercase alphanumeric: [a-z0-9]*"
  usage
elif [ ${#1} -gt 63 ]; then
  echo "  HOSTNAME must have <63 chars"
  usage
fi

# alterar hostname local
chost="$( hostname -s )"
sed -i "s/${chost}/${1}/g" /etc/hosts
sed -i "s/${chost}/${1}/g" /etc/hostname

invoke-rc.d hostname.sh restart
invoke-rc.d networking force-reload
hostnamectl set-hostname $1

# re-gerar chaves SSH
rm -f /etc/ssh/ssh_host_* 2> /dev/null
dpkg-reconfigure openssh-server &> /dev/null
```

O *script* acima espera receber um único parâmetro: o novo *hostname* a ser configurado para a máquina. Após checar se o parâmetro possui sintaxe válida, o *script* irá alterar o nome da máquina nos arquivos `/etc/hostname` e `/etc/hosts`, reiniciar as interfaces de rede e *daemons* relevantes, e finalmente re-gerar as chaves de host do `ssh` com o novo nome da máquina.

4) Configuração do LVM

1. Vamos prosseguir com a configuração do LVM, que fizemos apenas parcialmente durante a instalação — para relembrar, configuramos três volumes lógicos (LVs), `lv-boot`, `lv-swap` e `lv-root`. Para verificar o estado dos volumes lógicos em um sistema Linux, execute o comando `lvdisplay`:

```
# lvdisplay | grep 'Logical volume\|LV Path\|LV Name\|LV Size'
  --- Logical volume ---
  LV Path                /dev/vg-base/lv-boot
  LV Name                lv-boot
  LV Size                244,00 MiB
  --- Logical volume ---
  LV Path                /dev/vg-base/lv-swap
  LV Name                lv-swap
  LV Size                244,00 MiB
  --- Logical volume ---
  LV Path                /dev/vg-base/lv-root
  LV Name                lv-root
  LV Size                1,43 GiB
```

Para verificar o estado dos grupos de volumes (VGs), execute `vgdisplay`:

```
# vgdisplay
  --- Volume group ---
  VG Name               vg-base
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  4
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                3
  Open LV               3
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               8,00 GiB
  PE Size               4,00 MiB
  Total PE              2047
  Alloc PE / Size       488 / 1,91 GiB
  Free  PE / Size       1559 / 6,09 GiB
  VG UUID               hOXyJN-XA3n-i4RG-4mhJ-rDF5-edi7-xXS5f0
```

E, finalmente, para verificar o estado dos volumes físicos (PVs), execute `pvdisplay`:

```
# pvdisplay
  --- Physical volume ---
  PV Name               /dev/sda1
  VG Name               vg-base
  PV Size               8,00 GiB / not usable 2,00 MiB
  Allocatable           yes
  PE Size               4,00 MiB
  Total PE              2047
  Free PE               1559
  Allocated PE          488
  PV UUID               ZnHHMn-Y37D-6Psd-oHei-K1Qd-wHjY-6gqFZ3
```

2. Vamos criar três novos LVs, com as configurações que se seguem:

   - `lv-tmp`: armazenará o diretório `/tmp`, com tamanho de 512 MB.
   - `lv-var`: armazenará o diretório `/var`, com tamanho de 1536 MB.
   - `lv-usr`: armazenará o diretório `/usr`, ocupando todo o tamanho restante do VG `vg-base`.

   Para criá-los, execute os comandos:

```
# lvcreate -L 512M -n lv-tmp vg-base
Logical volume "lv-tmp" created.
```

```
# lvcreate -L 1536M -n lv-var vg-base
Logical volume "lv-var" created.
```

```
# lvcreate -l 100%FREE -n lv-usr vg-base
  Logical volume "lv-usr" created.
```

   Vamos verificar como ficou a situação dos nossos volumes lógicos:

```
# lvdisplay | grep 'Logical volume\|LV Path\|LV Name\|LV Size'
  --- Logical volume ---
  LV Path                /dev/vg-base/lv-boot
  LV Name                lv-boot
  LV Size                244,00 MiB
  --- Logical volume ---
  LV Path                /dev/vg-base/lv-swap
  LV Name                lv-swap
  LV Size                244,00 MiB
  --- Logical volume ---
  LV Path                /dev/vg-base/lv-root
  LV Name                lv-root
  LV Size                1,43 GiB
  --- Logical volume ---
  LV Path                /dev/vg-base/lv-tmp
  LV Name                lv-tmp
  LV Size                512,00 MiB
  --- Logical volume ---
  LV Path                /dev/vg-base/lv-var
  LV Name                lv-var
  LV Size                1,50 GiB
  --- Logical volume ---
  LV Path                /dev/vg-base/lv-usr
  LV Name                lv-usr
  LV Size                4,09 GiB
```

Naturalmente, o VG está ocupado em sua totalidade, agora:

```
# vgdisplay | grep PE
  PE Size               4,00 MiB
  Total PE              2047
  Alloc PE / Size       2047 / 8,00 GiB
  Free  PE / Size       0 / 0
```

O mesmo pode ser dito para o PV:

```
# pvdisplay | grep PE
  PE Size               4,00 MiB
  Total PE              2047
  Free PE               0
  Allocated PE          2047
```

3. Apesar de termos criado os LVs para os diretórios `/tmp`, `/var` e `/usr`, nosso trabalho ainda não acabou — temos que formatar esses diretórios, copiar o conteúdo dos diretórios atuais para dentro dos novos, configurar a montagem automática via `/etc/fstab` e apagar os diretórios antigos para liberar espaço.

Primeiro, vamos formatar os LVs usando o sistema de arquivos `ext4`:

```
# mkfs.ext4 /dev/mapper/vg--base-lv--tmp
(...)
```

```
# mkfs.ext4 /dev/mapper/vg--base-lv--var
(...)
```

```
# mkfs.ext4 /dev/mapper/vg--base-lv--usr
(...)
```

Apesar de não termos feito nos exemplos acima, este seria um excelente momento para customizar aspectos do sistema de arquivos para adequá-lo aos tipos específicos de arquivos que serão armazenados ali dentro. Por exemplo, pode ser interessante escolher um tamanho de *inode* menor do que o padrão caso se deseje armazenar muitos arquivos pequenos, como é frequentemente o caso em servidores de e-mail, digamos. Para mais informações, consulte a página de manual `man 8 mke2fs`.

Uma curiosidade: o tamanho padrão de *inodes* de novos sistemas de arquivos formatados é definido em `/etc/mke2fs.conf`; este valor pode ser customizado através da *flag* `-I` no comando `mkfs.ext*`.

4. O próximo passo é montar esses sistemas de arquivo e sincronizar o conteúdo dos diretórios atuais (que estão dentro da raiz, `/`) com os novos diretórios. Crie um *shell script*, `/root/scripts/syncdirs.sh`, com o seguinte conteúdo:

```
1 #!/bin/bash
2
3 for d in tmp var usr; do
4   [ -d /mnt/${d} ] || mkdir /mnt/${d}
5   mount /dev/mapper/vg--base-lv--${d} /mnt/${d}
6   rsync -av /${d}/ /mnt/${d}
7   umount /mnt/${d}
8   rmdir /mnt/${d}
9 done
```

O script acima irá iterar sobre os nomes `tmp`, `var` e `usr`, com o nome `DIR`. Para cada um deles, fará os passos a seguir:

1. Criar o diretório `/mnt/DIR`, se não existir.
2. Montar o volume lógico `lv-DIR` dentro do diretório `/mnt/DIR`.
3. Usando o comando `rsync`, copiar o conteúdo do diretório `/DIR` para `/mnt/DIR`.
4. Desmontar o volume lógigo `lv-DIR`.
5. Remover a pasta `/mnt/DIR`, se vazia.

Execute o *script*:

```
# bash ~/scripts/syncdirs.sh

(...)

sent 551,267 bytes  received 2,023 bytes  368,860.00 bytes/sec
total size is 409,038,950  speedup is 739.28
```

5. Vamos configurar a montagem automáticas dos novos volumes lógicos. Edite o arquivo `/etc/fstab` e adicione as linhas a seguir:

```
# nano /etc/fstab
(...)
```

```
# tail -n3 /etc/fstab
/dev/mapper/vg--base-lv--tmp  /tmp  ext4  defaults  0  2
/dev/mapper/vg--base-lv--var  /var  ext4  defaults  0  2
/dev/mapper/vg--base-lv--usr  /usr  ext4  defaults  0  2
```

Note que estamos usando as opções de montagem `defaults`, no exemplo acima. Segundo a página de manual do comando `mount` (que pode ser acessada através do comando `man 8 mount`), essa opção equivale a `rw, suid, dev, exec, auto, nouser, async`.

Considerando o uso das partições acima, pode ser interessante do ponto de vista de *hardening* tornar a montagem um pouco mais restritiva. Considere as seguintes opções:

- `ro`: montar o sistema de arquivos em modo somente-leitura. Inviável para diretórios como `/tmp` ou `/var`, embora possa ser considerado para o `/usr`, por exemplo. O grande inconveniente dessa proteção é que a permissão de escrita terá que ser atribuída manualmente sempre que se quiser escrever no diretório (digamos, durante a instalação de um novo pacote), motivo pelo qual não faremos essa configuração neste curso.
- `nosuid`: não permitir que *bits setuid* ou *setgid* tenham efeito no sistema de arquivos. Antes de colocar em prática, é recomendável escanear o sistema de arquivos por binários desse tipo, como faremos a seguir.
- `nodev`: não interpretar dispositivos especiais de bloco ou caractere nesse sistema de arquivos. Em geral, apenas o diretório `/dev` conterá arquivos dessa natureza.
- `noexec`: não permitir execução direta de quaisquer binários no sistema de arquivos. Não é viável habilitar essa opção para o diretório `/usr`, por motivos óbvios, mas pode ser uma boa opção para o `/tmp`, por exemplo — muitos *exploits* simples de escalada de privilégio tentam escrever e executar binários a partir do `/tmp`, e esta proteção pode dificultar sua ação. Contudo, alguns *scripts* de instalação de pacotes do Debian tentam executar binários diretamente do `/tmp`, e habilitá-lo com `noexec` pode quebrar a instalação desses pacotes — assim, não iremos utilizar essa configuração neste curso.

Antes de prosseguir com a customização das opções de montagem, vamos verificar quais binários possuem o *bit suid* ativo no sistema:

```
# find / -perm -4000 -exec ls {} \; 2> /dev/null
/bin/mount
/bin/ping
/bin/umount
/bin/su
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmcrypt-get-device
/usr/bin/newgrp
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/passwd
/usr/bin/gpasswd
```

Note que temos executáveis nos diretórios `/bin` e `/usr/bin` — assim, não é factível habilitar a opção `nosuid` no diretório `/usr`, neste momento.

De posse do conhecimento acima, vamos editar as opções de montagem dos volumes lógicos no arquivo `/etc/fstab`:

```
# nano /etc/fstab
(...)
```

```
# tail -n3 /etc/fstab
/dev/mapper/vg--base-lv--tmp  /tmp  ext4  defaults,nosuid,nodev         0  2
/dev/mapper/vg--base-lv--var  /var  ext4  defaults,nosuid,nodev         0  2
/dev/mapper/vg--base-lv--usr  /usr  ext4  defaults,nodev                0  2
```

6. O último passo é reiniciar o sistema e verificar se nossas configurações surtiram efeito. Antes disso, vamos registrar o tamanho ocupado por cada um dos diretórios (`tmp`, `var` e `usr`), e comparar com os tamanhos ocupados nos LVs após o *reboot*.

```
# du -sm /{tmp,var,usr}
1       /tmp
181     /var
463     /usr
```

Perfeito. Reinicie a máquina:

```
# reboot
```

Após o *reboot*, logue como o usuário `root` e verifique que os volumes lógicos estão montados

corretamente:

```
# mount | grep '/tmp\|/var\|/usr'
/dev/mapper/vg--base-lv--tmp on /tmp type ext4
(rw,nosuid,nodev,relatime,data=ordered)
/dev/mapper/vg--base-lv--var on /var type ext4
(rw,nosuid,nodev,relatime,data=ordered)
/dev/mapper/vg--base-lv--usr on /usr type ext4 (rw,nodev,relatime,data=ordered)
```

Verifique, ainda, que o espaço ocupado dentro desses volumes é bastante próximo do tamanho dos diretórios dentro da raiz, `/`:

```
# df -m | sed -n '1p; /\/\(tmp\|var\|usr\)/p'
Sist. Arq.                   Blocos de 1M Usado Disponível Uso% Montado em
/dev/mapper/vg--base-lv--tmp          488     1        452   1% /tmp
/dev/mapper/vg--base-lv--var         1480   186       1203  14% /var
/dev/mapper/vg--base-lv--usr         4059   486       3348  13% /usr
```

7. Faltou alguma coisa? Ah sim! Não apagamos o conteúdo dos diretórios `tmp`, `var` e `usr` de dentro da raiz, `/`. Note como o espaço ocupado ainda é bastante grande neste momento:

```
# df -m | sed -n '1p; /\/$/p'
Sist. Arq.                   Blocos de 1M Usado Disponível Uso% Montado em
/dev/mapper/vg--base-lv--root        1409   900        421  69% /
```

Mas, como apagar esses diretórios? Não podemos simplesmente rodar um comando `rm -rf /usr`, pois estaríamos removendo os arquivos gravados dentro do volume lógico `/dev/mapper/vg—base-lv—usr`, e não dentro da raiz. O que fazer, então?

A opção `bind` do comando `mount` (8) permite remontar um sistema de arquivos em outro ponto da hierarquia de diretórios, tornando seu conteúdo acessível em ambos os lugares. Monte, usando a opção `bind`, a raiz do sistema dentro do diretório `/mnt`:

```
# mount -o bind / /mnt
```

O diretório raiz, `/`, agora está acessível também abaixo de `/mnt`, como podemos observar:

```
# ls /mnt/
bin   dev  home        initrd.img.old  lib64       media  opt   root  sbin  sys
usr  vmlinuz
boot  etc  initrd.img  lib             lost+found  mnt    proc  run   srv   tmp
var  vmlinuz.old
```

O volume lógico `/dev/mapper/vg—base-lv—usr`, no entanto, está montado apenas abaixo do diretório `/usr`, e não abaixo de `/mnt/usr` — em outras palavras, a pasta `/mnt/usr` referencia

diretamente o conjunto de arquivos gravados dentro da raiz do sistema, os quais queremos apagar para liberar espaço.

Faça um teste — crie um arquivo dentro de `/usr` com o nome `teste`. Note que ele está acessível pelo caminho `/usr/teste`, mas não via `/mnt/usr/teste`:

```
# touch /usr/teste
```

```
# ls -ld /usr/teste
-rw-r--r-- 1 root root 0 out 18 15:48 /usr/teste
```

```
# ls -ld /mnt/usr/teste
ls: não foi possível acessar '/mnt/usr/teste': Arquivo ou diretório não encontrado
```

Perfeito! Apague o conteúdo dos diretórios `/mnt/tmp`, `/mnt/var` e `/mnt/usr`, que foram copiados para os volumes lógicos via `rsync` no passo (9) desta atividade e não são mais necessários. Não apague as pastas em si, pois elas são ponto de montagem desses LVs.

```
# for d in tmp var usr; do cd /mnt/${d} ; rm -rf ..?* .[!.]* *; done
```

Para referência, o comando acima irá entrar nas pastas `/mnt/tmp`, `/mnt/var` e `/mnt/usr`, e, em cada uma irá apagar todos os arquivos e diretórios:

- Não-ocultos (`*`)
- Ocultos, cujo primeiro caractere seja `.` e o segundo caractere seja qualquer *exceto* `.`
- Ocultos, iniciados por dois caracteres `.` e seguidos obrigatoriamente por algum outro caractere qualquer

Com efeito, a expressão regular acima irá apagar todo o conteúdo da pasta exceto os *symlinks* especiais `.` e `..`.

Verifique que o espaço ocupado dentro do diretório raiz, `/`, reduziu significativamente:

```
# df -m | sed -n '1p; /\/$/p'
Sist. Arq.                   Blocos de 1M Usado Disponível Uso% Montado em
/dev/mapper/vg--base-lv--root         1409   258       1063  20% /
```

Reinicie a máquina virtual para verificar que suas alterações não causaram nenhum impacto à estabilidade do sistema.

5) Inserção de senha no *bootloader*

Um aspecto que não pode ser esquecido é o *bootloader*, que faz a carga inicial do kernel — se desprotegido, um atacante com acesso físico à máquina pode utilizá-lo para alterar a senha do usuário `root` e ter acesso irrestrito ao sistema, dentre outras possibilidades.

Como vimos durante a instalação do Debian, o *bootloader* em uso pela grande maioria das distribuições Linux atualmente é o GRUB (*GRand Unified Bootloader*). Vamos configurar uma senha de acesso ao GRUB para impedir que um atacante consiga ter acesso indevido ao sistema.

1. Usando o comando `grub-mkpasswd-pbkdf2`, vamos gerar um hash para a senha `rnpesr123`.

```
# echo -e 'rnpesr123\nrnpesr123' | grub-mkpasswd-pbkdf2 | awk '/grub.pbkdf/{print$N
F}'
grub.pbkdf2.sha512.10000.E025151B0DA98A3153BADD61FCDC2A6037A0505699B7C414D046D83438
0AB53D20532441EDAFF9B1E330E8496D2C7799E6EFB43C399CC6567D0AFD8961F70109.50A159E523E8
89A805937F5BB65B4067149D0FDAB0536061015B4345647350A2E09D19580D77D51E58BFDA3432FE241
6AE61F90D7F84D1D834CFC979DCBA8F8D
```

2. Agora, vamos editar o arquivo `/etc/grub.d/40_custom` e inserir o superusuário `admin`, com senha idêntica ao hash gerado no passo anterior.

```
# echo 'set superusers="admin"' >> /etc/grub.d/40_custom
```

```
# ghash="$( echo -e 'rnpesr123\nrnpesr123' | grub-mkpasswd-pbkdf2 | awk
'/grub.pbkdf/{print$NF}' )" ; echo "password_pbkdf2 admin ${ghash}" >>
/etc/grub.d/40_custom ; unset ghash
```

```
# tail -n2 /etc/grub.d/40_custom
set superusers="admin"
password_pbkdf2 admin
grub.pbkdf2.sha512.10000.65E70B724A540A0AE79C2F6BB34AFC4397BB13952D20A9C209DD70A9F3
1FE462D301B733D8B1B308C67908A25B44AB09420CEB306EDAEEB15765905A7DEEB3BF.209A3080C89E
D4C542716B9BE14162E8338DF8E36B68F9E0146BDC8E572CF41585F6BF67C2573AFF2645F0D851A8E9D
5B6AA2E4608E4735E689FA84ECA815C14
```

3. Finalmente, vamos reconfigurar o GRUB com a nova combinação usuário/senha e reiniciar a máquina. Verifique se a configuração está funcionando.

```
# grub-mkconfig -o /boot/grub/grub.cfg
Generating grub configuration file ...
Imagem Linux encontrada: /boot/vmlinuz-4.9.0-8-amd64
Imagem initrd encontrada: /boot/initrd.img-4.9.0-8-amd64
concluído
```

```
# reboot
```

Após o *boot* da máquina, o menu do GRUB nos apresenta a possibilidade de editar a configuração apertando a tecla e:

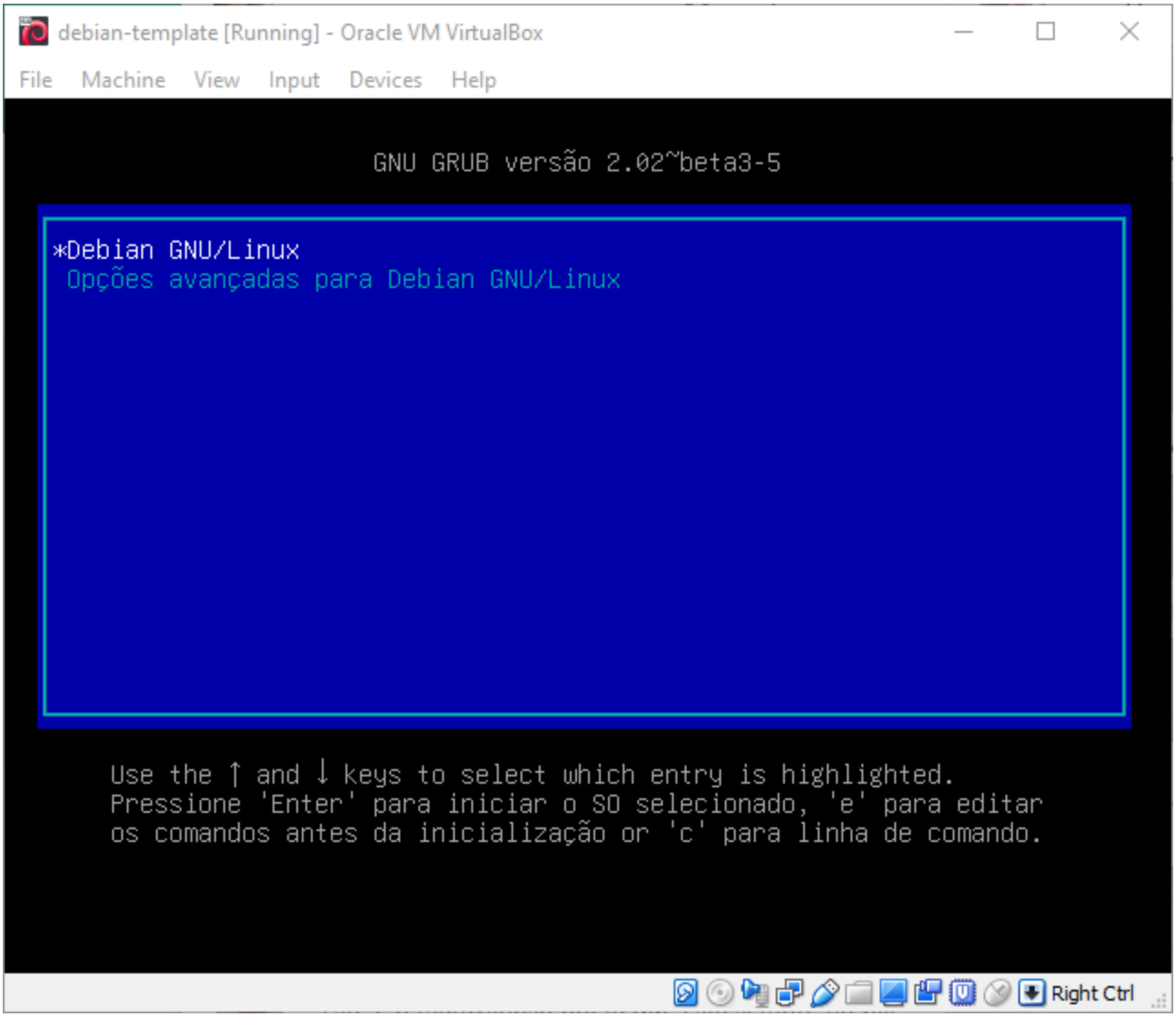

Figura 19. Edição de opções no GRUB

Apertando e sobre a primeira opção, imediatamente o sistema requisita a combinação usuário/senha configurada anteriormente:

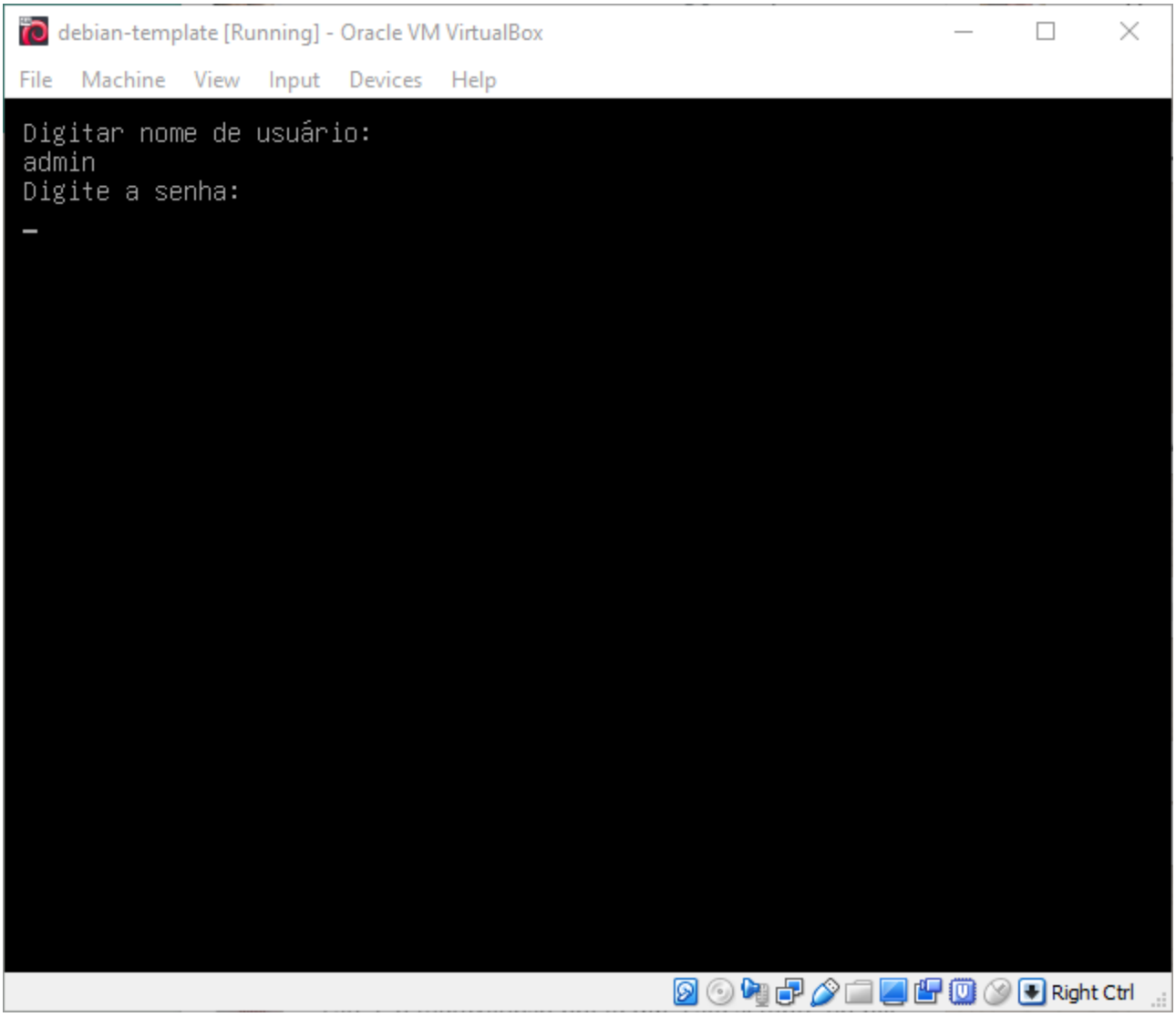

Figura 20. Inserção de usuário/senha no GRUB

Mediante a inserção da combinação correta, o menu de edição de opções de *boot* é mostrado, como se segue.

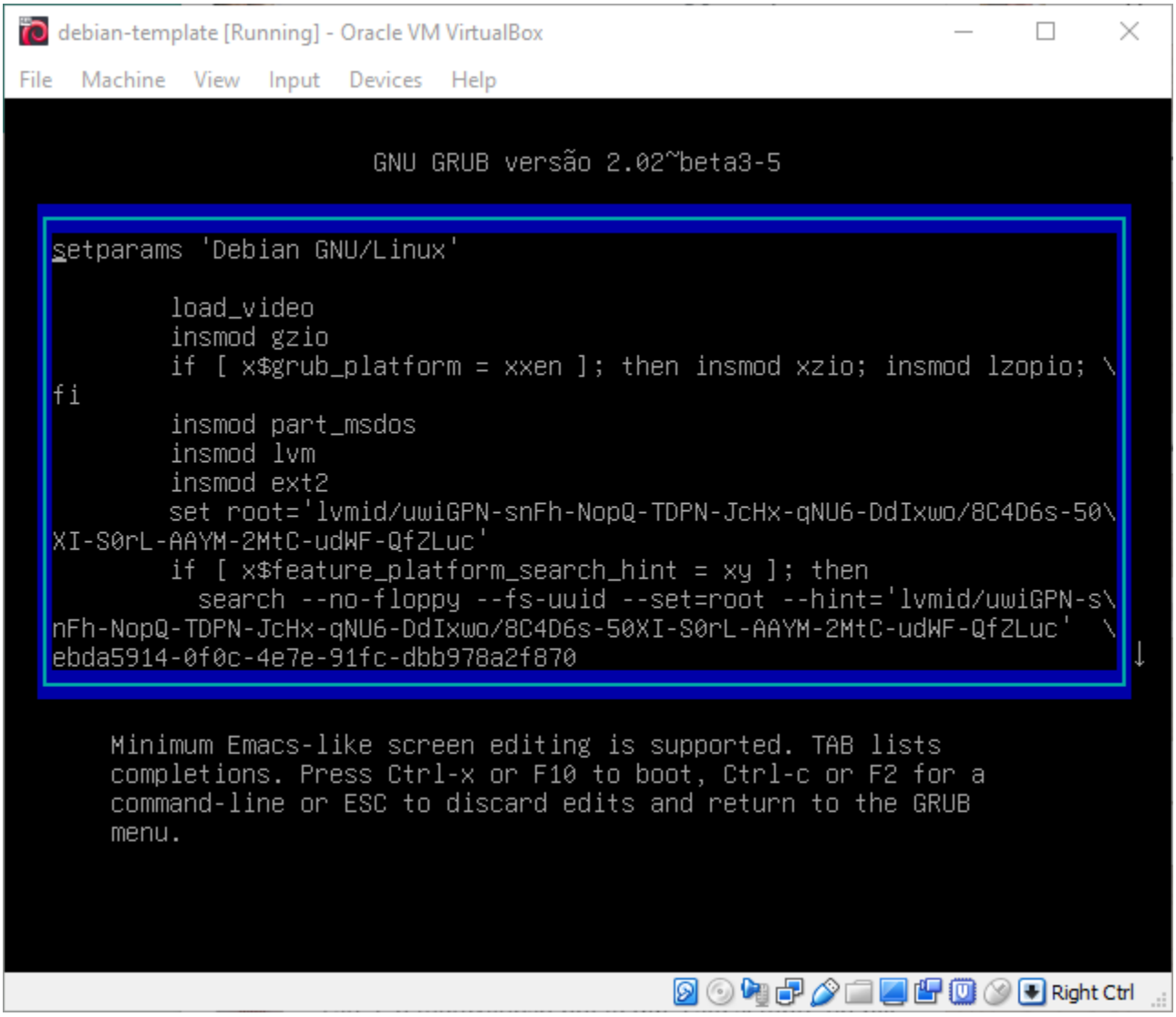

Figura 21. Edição de opções de boot no GRUB

Note ainda que, com esta configuração, o *boot* normal do sistema prossegue apenas se a combinação de usuário/senha correta for inserida no GRUB.

4. Vamos editar a configuração do GRUB para que ele solicite senha **apenas** em caso de edição de entradas do menu, e que o *boot* normal do sistema prossiga sem que haja necessidade de interação.

   Para conseguir o efeito desejado, é necessário editar o arquivo `/etc/grub.d/10_linux`. Na função `linux_entry()`, iremos editar as duas linhas `echo "menuentry (…)`, inserindo a flag `--unrestricted` antes da variável `${CLASS}`.

   Vamos ver um antes/depois para ficar mais claro. Veja como estão as linhas 132-134 do arquivo `/etc/grub.d/10_linux` antes da edição:

Listagem 1. /etc/grub.d/10_linux

```
132       echo "menuentry '$(echo "$title" | grub_quote)' ${CLASS}
\$menuentry_id_option 'gnulinux-$version-$type-$boot_device_id' {" | sed
"s/^/$submenu_indentation/"
133   else
134       echo "menuentry '$(echo "$os" | grub_quote)' ${CLASS}
\$menuentry_id_option 'gnulinux-simple-$boot_device_id' {" | sed
"s/^/$submenu_indentation/"
```

Após a edição, elas devem ficar assim:

Listagem 2. /etc/grub.d/10_linux

```
132       echo "menuentry '$(echo "$title" | grub_quote)' --unrestricted ${CLASS}
\$menuentry_id_option 'gnulinux-$version-$type-$boot_device_id' {" | sed
"s/^/$submenu_indentation/"
133   else
134       echo "menuentry '$(echo "$os" | grub_quote)' --unrestricted ${CLASS}
\$menuentry_id_option 'gnulinux-simple-$boot_device_id' {" | sed
"s/^/$submenu_indentation/"
```

Note a adição da `flag --unrestricted` antes de `${CLASS}` nas linhas 132 e 134.

Refaça a configuração do GRUB, reinicie a máquina e teste o funcionamento.

```
# update-grub
Generating grub configuration file ...
Imagem Linux encontrada: /boot/vmlinuz-4.9.0-8-amd64
Imagem initrd encontrada: /boot/initrd.img-4.9.0-8-amd64
concluído
```

```
# reboot
```

6) Clonando máquinas virtuais

Nosso *template*, para todos os efeitos, está preparado, atualizado e com configurações básicas de segurança aplicadas. Assim sendo, podemos utilizá-lo como base para a criação de novas máquinas virtuais durante este curso, começando a partir de agora.

Contudo, o Oracle VM Virtualbox não suporta o conceito de *templates* "a rigor", da mesma forma como interpretado em outras soluções de virtualização (como VMWare e Hyper-V). Para emular esse conceito de *templates*, sempre que necessário iremos clonar a máquina virtual `debian-template` e renomear a VM-clone, garantindo que o endereço físico (MAC) da placa de rede seja randomizado para evitar conflitos de IP.

1. Desligue a máquina `debian-template`:

```
# halt -p
```

2. Na janela principal do Virtualbox, clique com o botão direito na máquina `debian-template` e selecione a opção *Clone...*.

   Na tela seguinte, indique qual o nome da nova máquina virtual: para este exemplo, defina o nome `lvm-test`. Mantenha a caixa *Reinitialize the MAC address of all network cards* marcada.

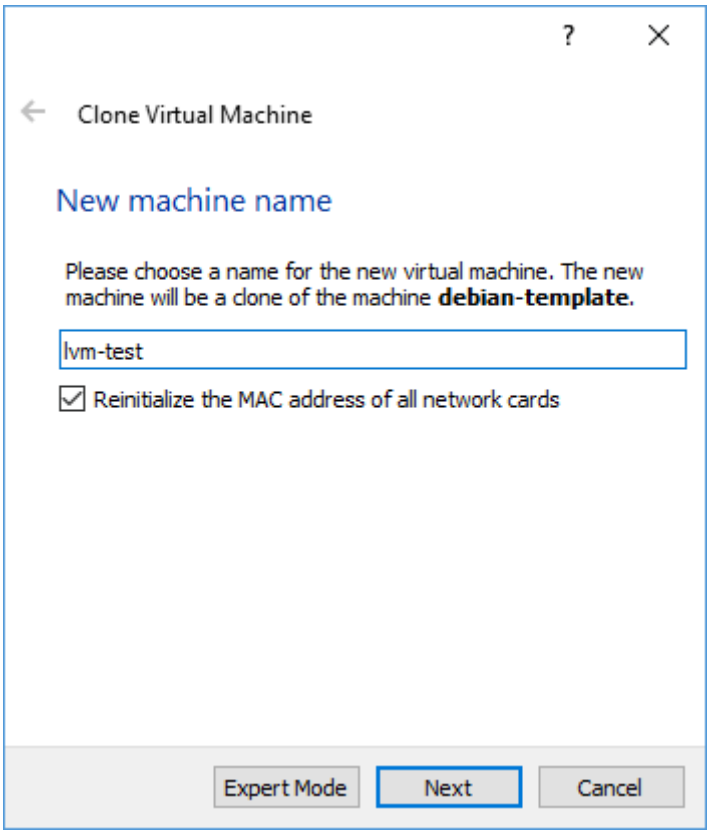

Figura 22. Clonagem de máquinas virtuais no Virtualbox

   Clique em *Next*.

3. Na janela seguinte, você pode escolher se deseja fazer um clone completo (*Full clone*), ou um clone ligado (*Linked clone*). A diferença entre ambos é que no caso do clone completo é criada uma cópia separada do disco da VM original, ao passo que no clone ligado faz-se apenas um *snapshot* desse disco.

   Mantenha *Full clone* marcado e clique em *Clone*.

7) Operações avançadas com LVM

Imagine que a máquina que acabamos de criar, `lvm-test`, será usada para dois propósitos: 1) atuar como um servidor de e-mail, armazenando as mensagens dos usuários da organização sob a pasta `/var/mail` e 2) armazenar dados sensíveis da organização, que devem ser acessados apenas por um número muito restrito de usuários, sob a pasta `/crypt`.

Vamos lidar com a situação (1), primeiramente.

1. Opere com a máquina recém-clonada, `lvm-test`. Na janela principal do Virtualbox, clique com o botão direito sobre a VM e depois em *Settings*.

   Em *Storage* > *Controller: SATA*, clique no ícone com um pequeno HD com um sinal de `+`, com a legenda *Adds hard disk* para adicionar um novo disco à VM. Depois, clique em *Create new disk*.

   Para o formato, escolha *VDI* e clique em *Next*. Mantenha a caixa *Dynamically allocated* marcada e clique em *Next*.

   Para o nome do disco, digite `lvm-pv2` e mantenha o tamanho em 8 GB. Finalmente, clique em *Create*.

2. Repita o passo (1), adicionando um terceiro disco à VM. Desta vez, nomeie o disco como `lvm-crypt` e mantenha seu tamanho em 8 GB.

   Ao final do processo, sua VM deverá estar com a seguinte configuração:

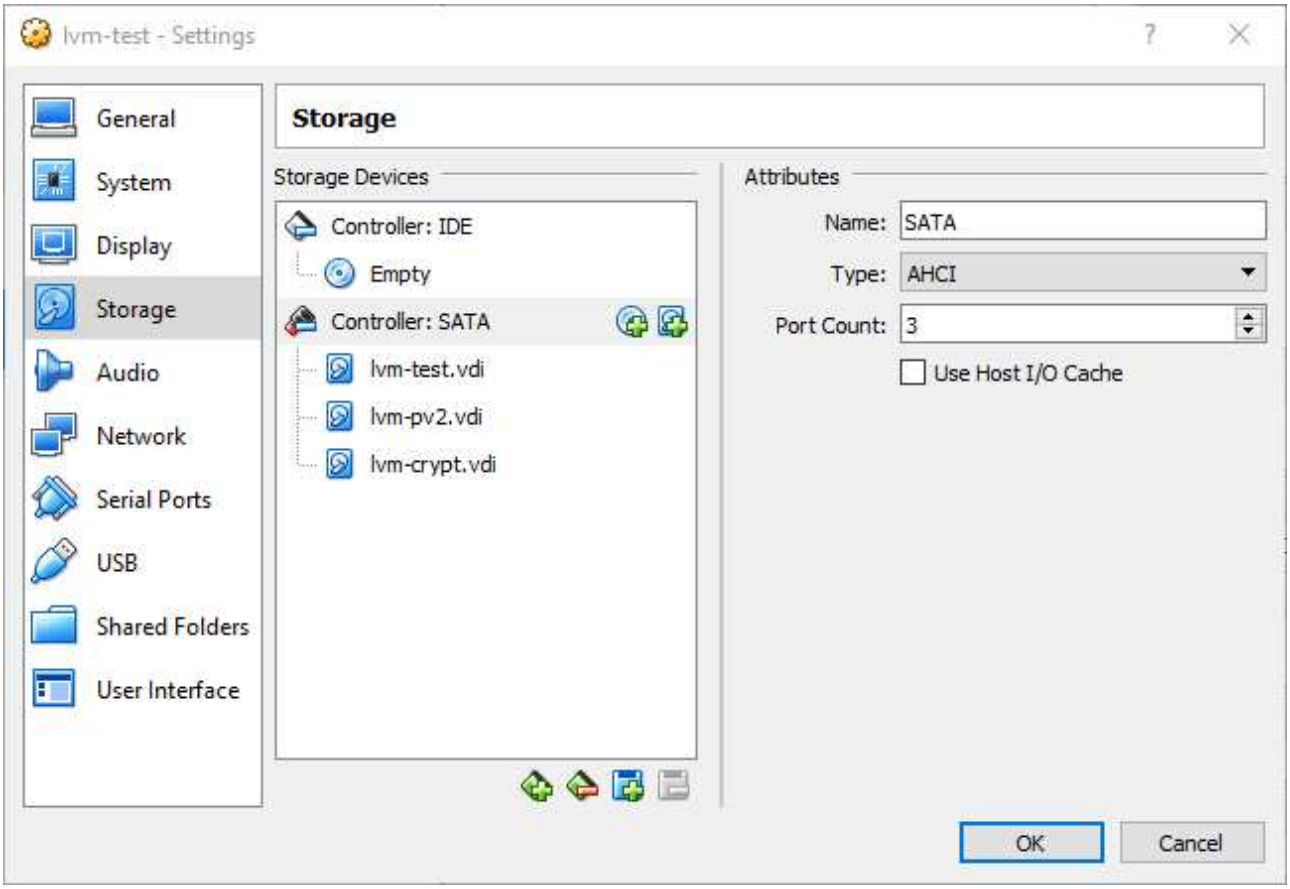

Figura 23. Discos adicionados à máquina lvm-test

   Clique em *OK*, e ligue a máquina `lvm-test`.

3. Usando o script `/root/scripts/changehost.sh` que criamos anteriormente, renomeie a máquina:

```
# hostname
debian-template
```

```
# bash ~/scripts/changehost.sh lvm-test
```

```
# hostname
lvm-test
```

4. Prosseguindo com o tratamento do requisito (1), abordado no enunciado desta atividade, note que o espaço disponível no `/var` atualmente é bastante exíguo:

   ```
   # df -h | sed -n '1p; /\/var$/p'
   Sist. Arq.                    Tam. Usado Disp. Uso% Montado em
   /dev/mapper/vg--base-lv--var   1,5G  185M  1,2G  14% /var
   ```

   Iremos utilizar o primeiro disco adicionado à VM, `lvm-pv2`, para aumentar o tamanho disponível para o diretório `/var`. Queremos, em ordem:

   1. Identificar sob qual nome o disco `lvm-pv2` foi identificado pelo sistema Linux
   2. Adicionar a totalidade do disco `lvm-pv2` ao grupo de volumes `vg-base`
   3. Estender o volume lógico `lv-var` para usar o espaço extra disponível no VG `vg-base`
   4. Estender o sistema de arquivos do dispositivo `/dev/mapper/vg—base-lv—var` para utilizar o espaço extra disponível no LV `lv-var`
   5. Verificar o aumento do espaço disponível

5. Primeiramente, temos que detectar sob qual nome foi adicionado o disco virtual `lvm-pv2`. O comando `dmesg` nos mostra que três discos foram detectados durante o *boot*:

   ```
   # dmesg | grep 'Attached SCSI disk'
   [    1.924154] sd 2:0:0:0: [sdc] Attached SCSI disk
   [    1.924182] sd 1:0:0:0: [sdb] Attached SCSI disk
   [    1.936628] sd 0:0:0:0: [sda] Attached SCSI disk
   ```

   Desses, sabemos que o dispositivo `/dev/sda` é o disco original que criamos durante a instalação do sistema, já que ele se encontra formatado e em uso pelo LVM:

```
# pvdisplay
  --- Physical volume ---
  PV Name               /dev/sda1
  VG Name               vg-base
  PV Size               8,00 GiB / not usable 2,00 MiB
  Allocatable           yes (but full)
  PE Size               4,00 MiB
  Total PE              2047
  Free PE               0
  Allocated PE          2047
  PV UUID               ZnHHMn-Y37D-6Psd-oHei-K1Qd-wHjY-6gqFZ3
```

Restam, então, os discos `/dev/sdb` e `/dev/sdc`. Em tese, poderíamos usar a diferença de tamanho entre os discos para intuir qual deles é o volume `lvm-pv2`, e qual é o `lvm-crypt`. Contudo, ambos possuem o mesmo tamanho, 8 GB. O que fazer, então?

Para determinar com precisão essa informação, na janela principal do Virtualbox acesse *File* > *Virtual Media Manager*. Na aba *Hard disks*, selecione o disco `lvm-pv2.vdi` e acesse a aba *Information*, como mostrado abaixo:

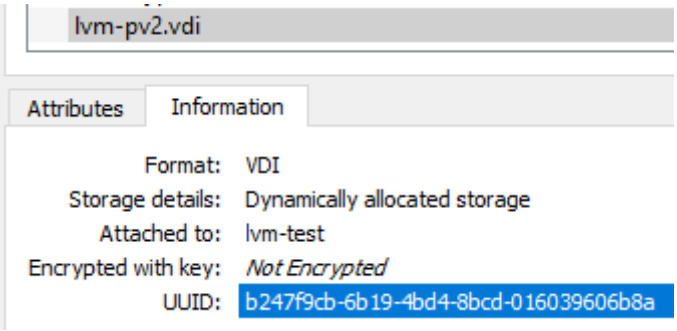

Figura 24. Identificando o UUID de um disco no Virtualbox

Note as *strings* inicial e final do campo *UUID* (*Universally Unique Identifier*), `b247f9cb` e `016039606b8a` respectivamente no exemplo acima.

De volta à máquina `lvm-test`, execute o comando:

```
# hdparm -i /dev/sdb | grep 'SerialNo' | cut -d',' -f3
 SerialNo=VBb247f9cb-8a6b6039
```

Excluindo-se as letras `VB` do *serial* acima, note que a *string* `b247f9cb` é idêntica à que visualizamos no Virtualbox. De igual modo, a *string* `8a6b6039` é uma reversão dois-a-dois do final da *string* identificada no *Virtual Media Manager* do Virtualbox, anteriormente. Portanto, podemos afirmar com segurança que o disco `/dev/sdb` é o volume virtual `lvm-pv2`.

Faça o teste com o volume `lvm-crypt` e o disco `/dev/sdc`. As *strings* identificadoras do campo *UUID* são compatíveis?

6. Identificado o disco `/dev/sdb` como nosso alvo, e como desejamos adicionar um disco inteiro ao VG `vg-base`, o primeiro passo é editar a tabela de partições do disco corretamente. Para tanto,

basta criar uma partição primária ocupando a totalidade do disco, e identificá-la como `Linux LVM`.

```
# fdisk /dev/sdb

Bem-vindo ao fdisk (util-linux 2.29.2).
As alterações permanecerão apenas na memória, até que você decida gravá-las.
Tenha cuidado antes de usar o comando de gravação.
```

```
Comando (m para ajuda): o
Criado um novo rótulo de disco DOS com o identificador de disco 0xe7d643f2.
```

```
Comando (m para ajuda): n
Tipo da partição
   p   primária (0 primárias, 0 estendidas, 4 livre)
   e   estendida (recipiente para partições lógicas)
Selecione (padrão p):

Usando resposta padrão p.
Número da partição (1-4, padrão 1):
Primeiro setor (2048-16777215, padrão 2048):
Último setor, +setores ou +tamanho{K,M,G,T,P} (2048-16777215, padrão 16777215):

Criada uma nova partição 1 do tipo "Linux" e de tamanho 8 GiB.
```

```
Comando (m para ajuda): t
Selecionou a partição 1
Tipo de partição (digite L para listar todos os tipos): 8e
O tipo da partição "Linux" foi alterado para "Linux LVM".
```

```
Comando (m para ajuda): w
A tabela de partição foi alterada.
Chamando ioctl() para reler tabela de partição.
Sincronizando discos.
```

Em ordem, executamos `fdisk /dev/sdb` para editar a tabela de partições do dispositivo e então:

- `o` para criar uma tabela de partições vazia
- `n` para criar uma nova partição
- `ENTER` para aceitar o tipo padrão de partição (primária)
- `ENTER` para aceitar o número padrão de partição (número `1`)
- `ENTER` para aceitar o primeiro setor disponível no disco (2048)

- `ENTER` para aceitar o último setor disponível no disco (16777215), maximizando o tamanho da partição
- `t` para alterar o identificador da partição
- `8e` para identificar a partição como `Linux LVM`
- `w` para gravar as alterações realizadas e sair do programa

Para visualizar o estado do disco, podemos usar `fdisk -l`:

```
# fdisk -l /dev/sdb
Disco /dev/sdb: 8 GiB, 8589934592 bytes, 16777216 setores
Unidades: setor de 1 * 512 = 512 bytes
Tamanho de setor (lógico/físico): 512 bytes / 512 bytes
Tamanho E/S (mínimo/ótimo): 512 bytes / 512 bytes
Tipo de rótulo do disco: dos
Identificador do disco: 0xe7d643f2

Dispositivo Inicializar Início      Fim  Setores Tamanho Id Tipo
/dev/sdb1                 2048 16777215 16775168      8G 8e Linux LVM
```

A seguir, usaremos o comando `pvcreate` para inicializar a partição e prepará-la para o LVM:

```
# pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created.
```

```
# pvdisplay /dev/sdb1
  "/dev/sdb1" is a new physical volume of "8,00 GiB"
  --- NEW Physical volume ---
  PV Name               /dev/sdb1
  VG Name
  PV Size               8,00 GiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               FdZmMJ-ZFdQ-vi0z-lc5r-9Zy2-zfbR-1uJbuj
```

O comando `pvscan` é uma opção interessante para mostrar o estado dos volumes físicos do sistema:

```
# pvscan
  PV /dev/sda1   VG vg-base         lvm2 [8,00 GiB / 0    free]
  PV /dev/sdb1                      lvm2 [8,00 GiB]
  Total: 2 [16,00 GiB] / in use: 1 [8,00 GiB] / in no VG: 1 [8,00 GiB]
```

7. Agora sim, podemos adicionar o novo volume físico `/dev/sdb1` ao grupo de volumes `vg-base`. Note seu espaço atual:

```
# vgdisplay | grep 'VG Name\|VG Size'
  VG Name               vg-base
  VG Size               8,00 GiB
```

   Expanda o VG:

```
# vgextend vg-base /dev/sdb1
  Volume group "vg-base" successfully extended
```

   Verificando o estado do VG `vg-base`, note que seu tamanho saltou para 16 GB, como esperado:

```
# vgdisplay | grep 'VG Name\|VG Size'
  VG Name               vg-base
  VG Size               15,99 GiB
```

8. A seguir, iremos estender o volume lógico `lv-var` para utilizar a totalidade do novo espaço adicionado ao VG. Note seu espaço atual:

```
# lvdisplay /dev/vg-base/lv-var | grep Size
  LV Size                1,50 GiB
```

   Façamos o procedimento de expansão:

```
# lvextend -l +100%FREE /dev/vg-base/lv-var
  Size of logical volume vg-base/lv-var changed from 1,50 GiB (384 extents) to 9,50
GiB (2431 extents).
  Logical volume vg-base/lv-var successfully resized.
```

   E em seguida, chequemos o novo tamanho disponível:

```
# lvdisplay /dev/vg-base/lv-var | grep Size
  LV Size                9,50 GiB
```

9. O passo final é redimensionar o sistema de arquivos. Como queremos simplesmente ocupar a totalidade do volume lógico, e kernels mais modernos do Linux suportam *on-line resizing* (i.e. redimensionamento sem necessidade de desmontar o sistema de arquivos), basta executar:

```
# resize2fs /dev/mapper/vg--base-lv--var
resize2fs 1.43.4 (31-Jan-2017)
Filesystem at /dev/mapper/vg--base-lv--var is mounted on /var; on-line resizing
required
old_desc_blocks = 1, new_desc_blocks = 2
The filesystem on /dev/mapper/vg--base-lv--var is now 2489344 (4k) blocks long.
```

Note, imediatamente, que o espaço disponível para o `/var` aumenta significativamente:

```
# df -h | sed -n '1p; /\/var$/p'
Sist. Arq.                    Tam. Usado Disp. Uso% Montado em
/dev/mapper/vg--base-lv--var   9,4G  188M  8,8G   3% /var
```

E assim, concluímos nossa seção sobre o LVM. É um sistema muito poderoso, que oferece grande flexibilidade na gestão de armazenamento no Linux. Imagine: qual teria sido a dificuldade em estender o espaço disponível para o `/var` em um sistema sem o uso do LVM?

Outros aspectos avançados do LVM, como gestão de volumes *striped* (concatenados) e *mirrored* (espelhados), provisionamento dinâmico de espaço e *snapshots* não foram trabalhados aqui. Convidamos o aluno a investigar essas capacidades, e testar suas funcionalidades em ambiente de laboratório. A documentação do Red Hat Enterprise Linux sobre o LVM é um excelente recurso para começar: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/logical_volume_manager_administration/lv_overview

8) Criptografia de partições

Retomando o requisito (2) apresentado na atividade (7), foi dito que se desejava: armazenar dados sensíveis da organização, que devem ser acessados apenas por um número muito restrito de usuários, sob a pasta `/crypt`.

Vamos, agora, implementar esse requisito na máquina `lvm-test`.

1. A solução que iremos implantar para garantir o requisito é criptografar a partição. Para tanto, precisaremos instalar os seguintes pacotes:

```
# apt-get install cryptsetup -y
```

2. Prepare o disco `/dev/sdc` da mesma forma que fizemos no passo (6) da atividade anterior. Ao final do processo, a saída do comando `pvdisplay /dev/sdc1` deve ser como mostrada abaixo:

```
# pvdisplay /dev/sdc1
  "/dev/sdc1" is a new physical volume of "8,00 GiB"
  --- NEW Physical volume ---
  PV Name               /dev/sdc1
  VG Name
  PV Size               8,00 GiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               JKRAlt-0sgK-d0np-3N3J-JRyE-YDbG-0cdZee
```

3. Vamos criar um novo VG para armazenar dados sensíveis. Execute:

```
# vgcreate vg-crypt /dev/sdc1
  Volume group "vg-crypt" successfully created
```

4. O próximo passo é criar um volume lógico:

```
# lvcreate -l +100%FREE -n lv-crypt vg-crypt
  Logical volume "lv-crypt" created.
```

   Verifique que o LV foi criado corretamente:

```
# lvdisplay /dev/vg-crypt/lv-crypt
  --- Logical volume ---
  LV Path                /dev/vg-crypt/lv-crypt
  LV Name                lv-crypt
  VG Name                vg-crypt
  LV UUID                mMOiAc-Q7Yo-hgjN-rb2S-mzfr-youw-PnHGRC
  LV Write Access        read/write
  LV Creation host, time lvm-test, 2018-10-19 11:12:42 -0300
  LV Status              available
  # open                 0
  LV Size                8,00 GiB
  Current LE             2047
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           254:6
```

5. Agora, vamos criptografar esse LV — o comando `cryptsetup` pode ser usado para este fim. Iremos criptografar o disco no formato LUKS (*Linux Unified Key Setup*), o padrão para criptografia de armazenamento no Linux. Diferentemente de outras soluções de criptografia, o

LUKS armazena todas as informações de configuração no cabeçalho da partição, permitindo ao usuário migrar seus dados de forma fácil entre diferentes distribuições Linux ou mesmo outros sistemas operacionais.

A cifra padrão de criptografia do LUKS pode ser visualizada com o comando:

```
# cryptsetup --help | grep 'LUKS1:'
        LUKS1: aes-xts-plain64, Chave: 256 bits, Hash de cabeçalho LUKS: sha256,
RNG: /dev/urandom
```

Para criptografar a partição, execute:

```
# cryptsetup luksFormat /dev/mapper/vg--crypt-lv--crypt

WARNING!
========
Isto vai sobrescrever dados em /dev/mapper/vg--crypt-lv--crypt permanentemente.

Are you sure? (Type uppercase yes): YES
Digite a senha:
Verificar senha:
```

Digite *YES* em letras maiúsculas para confirmar sobrescrita dos dados. Escolha uma senha forte para proteger os dados. Neste laboratório, recomendamos a senha `rnpesr123`, por conveniência.

6. Como saber que uma partição está criptografada? E, de fato, como saber o tipo de uma partição qualquer? O comando `lsblk` é especialmente útil nesse cenário:

```
# lsblk --fs
NAME                    FSTYPE      LABEL UUID
MOUNTPOINT
sda
└──sda1                 LVM2_member       n3dXDL-gCma-P258-pl31-Ow9A-Spcp-2mjsCG
   ├──vg--base-lv--boot ext2              ebda5914-0f0c-4e7e-91fc-dbb978a2f870
/boot
   ├──vg--base-lv--swap swap              fcf0e5dd-d744-4acc-aff4-65aa2219fde2
[SWAP]
   ├──vg--base-lv--root ext4              bfebe607-ef84-4ac8-8ce0-54dded08ae9e
/
   ├──vg--base-lv--tmp  ext4              3db27de5-69ae-4db7-baba-2de5a4576942
/tmp
   ├──vg--base-lv--var  ext4              2a13673c-4266-4683-a548-d66b0c1078b1
/var
   └──vg--base-lv--usr  ext4              5621572f-4786-4cb6-8826-2dae623b3e5d
/usr
sdb
└──sdb1                 LVM2_member       FdZmMJ-ZFdQ-vi0z-lc5r-9Zy2-zfbR-1uJbuj
   └──vg--base-lv--var  ext4              2a13673c-4266-4683-a548-d66b0c1078b1
/var
sdc
└──sdc1                 LVM2_member       JKRAlt-0sgK-d0np-3N3J-JRyE-YDbG-0cdZee
   └──vg--crypt-lv--crypt crypto_LUKS     2f10b9c6-aab3-4038-b74c-6bab83e658fe
sr0
```

Note que o LV que acabamos de criar e criptografar, `lv-crypt`, possui o tipo de sistema de arquivos `crypto_LUKS`. Note, ainda, que ele não está montado:

```
# mount | grep 'lv--crypt'
```

7. O próximo passo é formatar a partição. Contudo, para acessar partições LUKS, temos primeiro que mapeá-la sob um nome — execute:

```
# cryptsetup luksOpen /dev/mapper/vg--crypt-lv--crypt seg10-crypt
Digite a senha para /dev/mapper/vg--crypt-lv--crypt:
```

Para verificar os nomes mapeados, novamente podemos usar o `lsblk`:

```
# lsblk /dev/sdc
NAME                  MAJ:MIN RM SIZE RO TYPE  MOUNTPOINT
sdc                     8:32   0   8G  0 disk
└──sdc1                   8:33   0   8G  0 part
   └──vg--crypt-lv--crypt 254:6    0   8G  0 lvm
     └──seg10-crypt       254:7    0   8G  0 crypt
```

Com o dispositivo `/dev/mapper/vg—crypt-lv—crypt` mapeado para `/dev/mapper/seg10-crypt`, podemos, agora sim, formatar a partição:

```
# mkfs.ext4 /dev/mapper/seg10-crypt
mke2fs 1.43.4 (31-Jan-2017)
Creating filesystem with 2095616 4k blocks and 524288 inodes
Filesystem UUID: 5740608c-8609-4d95-ae9b-73a016765610
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

E, finalmente, montá-la:

```
# mount /dev/mapper/seg10-crypt /mnt/
```

```
# mount | grep '/mnt'
/dev/mapper/seg10-crypt on /mnt type ext4 (rw,relatime,data=ordered)
```

8. Após gravar dados na partição criptografada, temos que fazer o caminho oposto. Primeiro, desmontá-la:

```
# umount /mnt
```

E, em seguida, remover o mapeamento por nome da partição LUKS:

```
# cryptsetup luksClose seg10-crypt
```

9. Concluídas nossas atividades de exemplo com o LVM e criptografia de partições, desligue a máquina `lvm-test`:

```
# halt -p
```

Como não utilizaremos mais esta máquina no decorrer do curso, vamos removê-la. Na janela principal do Virtualbox, clique com o botão direito sobre a VM `lvm-test` e selecione *Remove*. Na nova janela, clique em *Delete all files* para remover todos os discos associados à máquina.

Sessão 2: Firewall e DNS

O ponto de partida para a segurança da rede em qualquer organização é, certamente, o *firewall*. Utilizando-se do princípio de segurança do *chokepoint*, ou gargalo, tem como premissa atuar como um ponto focal no controle do tráfego de rede, permitindo aos administradores configurar acessos e proibições de acordo com as políticas organizacionais. Conceitualmente, o firewall consiste em um filtro de pacotes e tem sua atuação restrita até a camada 4 (transporte) do modelo OSI, ficando a tarefa de filtragem em nível de aplicação para outros programas e *appliances* de controle da rede, como IDSs (*Intrusion Detection System*), IPSs (*Intrusion Protection System*), *proxies* ou WAFs (*Web Application Firewalls*).

Juntamente com o firewall, o serviço de resolução de nomes (DNS, ou *Domain Name System*) também é peça-chave na segurança da rede. Provendo um serviço essencial — tradução de endereços de rede para nomes e vice-versa — o serviço DNS deve ser cuidadosamente configurado pelos administradores para evitar ataques de *spoofing* e prover redundância em caso de falhas inesperadas. Os servidores DNS se dividem em primário, secundário e recursivo; destes, os dois primeiros são responsáveis por responder por um domínio de rede (ditos **autoritativos**, portanto), e o último apenas funciona como uma fonte de consulta e *cache* para os clientes da rede.

Neste curso, faremos uma configuração extremamente restritiva do firewall de nosso *datacenter* simulado, e a cada novo serviço implementado iremos retornar à sua configuração para abrir as portas e protocolos adequados. Já no caso do DNS, configuraremos um servidor primário/secundário autoritativo com suporte a DNSSEC para a rede interna `intnet.`, restringindo todas as máquinas a realizarem suas consultas exclusivamente nesses servidores. Iremos, ainda, configurar um servidor DNS recursivo em um *daemon* distinto exclusivamente para consultas e *cache* de resultados, visando o aumento da segurança do sistema de resolução de nomes.

1) Topologia desta sessão

A figura abaixo mostra a topologia de rede que será utilizada nesta sessão, com as máquinas relevantes em destaque.

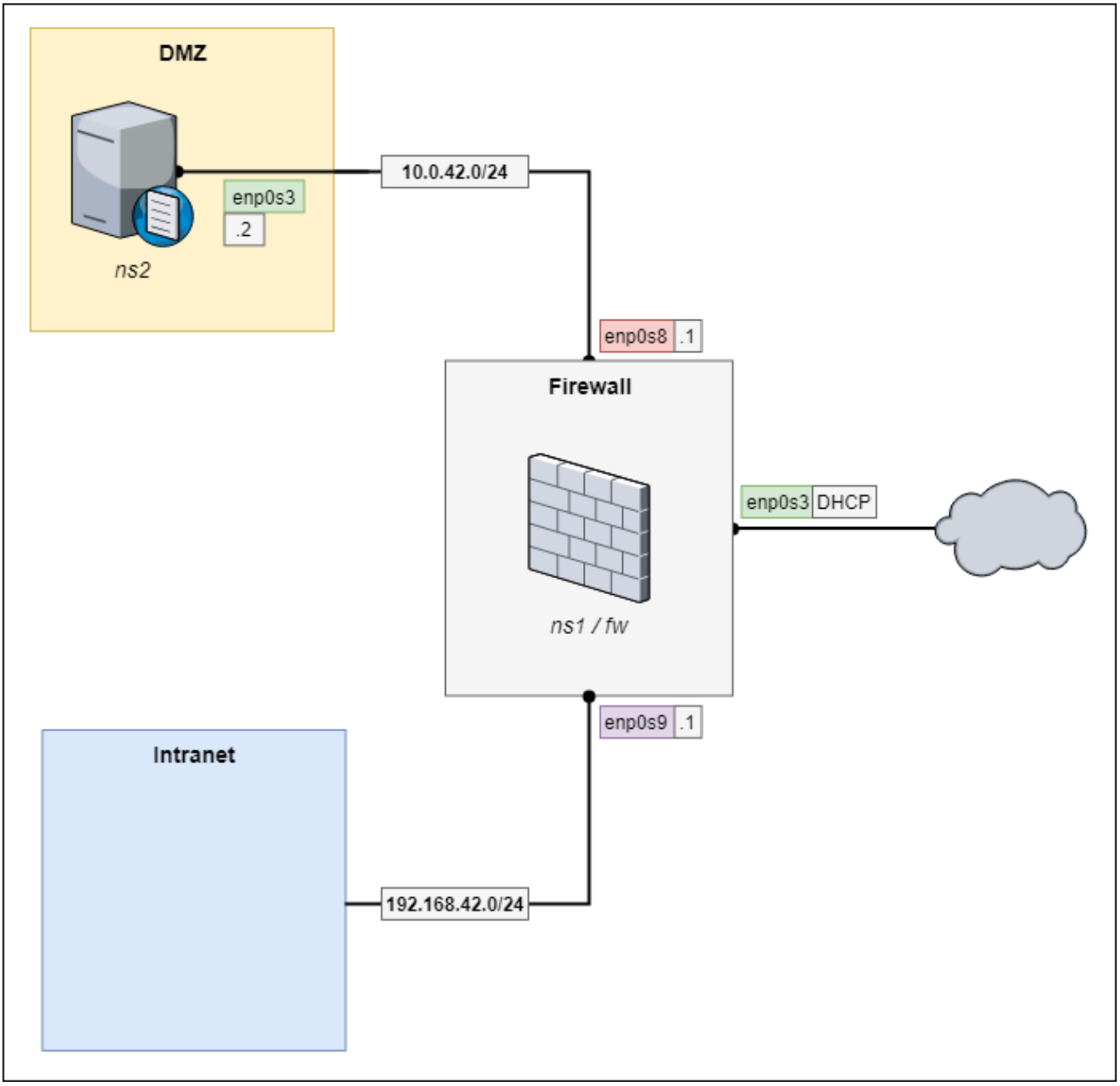

Figura 25. Topologia de rede desta sessão

Teremos apenas duas máquinas, por enquanto:

- `ns1`, atuando como firewall de rede e DNS primário. Endereços IP via DHCP (interface *bridge*), 10.0.42.1/24 (interface DMZ) e 192.168.42.1/24 (interface Intranet).
- `ns2`, atuando como DNS secundário e localizada na DMZ (*Demilitarized Zone*, ou zona desmilitarizada). Endereço IP 10.0.42.2/24.

1. Antes de começarmos, precisamos configurar corretamente as redes virtuais no Virtualbox. Acesse o menu *File* > *Host Network Manager* e crie as seguintes redes:

Tabela 1. Redes host-only no Virtualbox

Rede	Endereço IPv4	Máscara de rede	Servidor DHCP
Virtualbox Host-Only Ethernet Adapter #2	10.0.42.254	255.255.255.0	Desabilitado

Rede	Endereço IPv4	Máscara de rede	Servidor DHCP
Virtualbox Host-Only Ethernet Adapter #3	192.168.42.254	255.255.255.0	Desabilitado

Visualmente, sua janela deve ficar parecida com o seguinte:

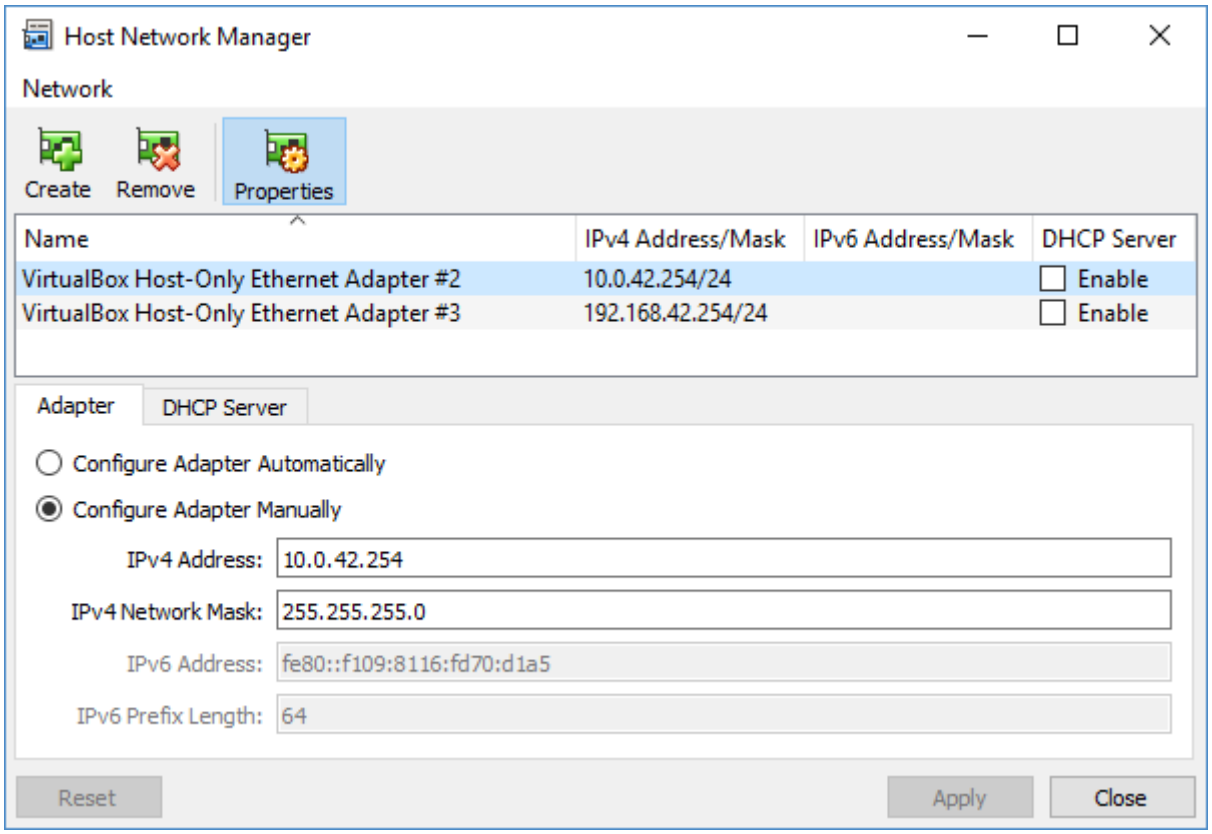

Figura 26. Redes host-only no Virtualbox

É possível que os números específicos das redes (`#2` para a DMZ e `#3` para a Intranet, na imagem acima) não fiquem exatamente iguais aos exemplificados. Nesse caso, faça uma anotação indicando qual número de rede *host-only* corresponde a cada uma das redes configuradas. Verifique, ainda, que o servidor DHCP interno do Virtualbox está desabilitado em ambas as redes.

2. As configurações de rede realizadas internamente em cada máquina virtual foram apresentados de forma sucinta na topologia desta sessão. Iremos detalhar as configurações logo abaixo:

Tabela 2. Configurações de rede de cada VM

VM Nome	Interface	Modo	Endereço	Gateway
ns1	enp0s3	DHCP	Automático	Automático
	enp0s8	Estático	10.0.42.1/24	n/a
	enp0s9	Estático	192.168.42.1/24	n/a
ns2	enp0s3	Estático	10.0.42.2/24	10.0.42.1

A partir do Debian 9, a nomenclatura padrão de interfaces de rede foi alterada. Ao invés de denotarmos as interfaces como `eth0`, `eth1` ou `eth2`, o `systemd/udev` utiliza, a partir da versão v197, um método de nomenclatura de interfaces usando *biosdevnames*, como documentado oficialmente em https://www.freedesktop.org/wiki/Software/systemd/PredictableNetworkInterfaceNames/ . Com efeito, esse novo sistema suporta cinco meios de nomeação de interfaces de rede:

1. Nomes incorporando números de índice providos pelo firmware/BIOS de dispositivos *on-board* (p.ex.: `eno1`)
2. Nomes incorporando números de índice providos pelo firmware/BIOS de encaixes *hotplug* PCI Express (p.ex.: `ens1`)
3. Nomes incorporando localização física/geográfica do conector do hardware (p.ex.: `enp2s0`)
4. Nomes incorporando o endereço MAC da interface (p.ex.: `enx78e7d1ea46da`)
5. Nomes clássicos, usando nomenclatura não-previsível nativa do kernel (p.ex.: `eth0`)

Todas as VMs utilizadas neste curso serão derivadas da máquina `debian-template`, configurada com o Debian 9. A equivalência da nova nomenclatura de interfaces de rede, se comparada com a antiga, ficaria assim:

Tabela 3. Nomenclatura de interfaces de máquinas Debian 9

Interface antiga	Interface nova
eth0	enp0s3
eth1	enp0s8
eth2	enp0s9

Observe, por exemplo, como é feita a detecção de interfaces durante o *boot* da máquina `ns1`, que criaremos a seguir:

```
# dmesg | grep 'renamed from'
[    1.658908] e1000 0000:00:09.0 enp0s9: renamed from eth2
[    1.659807] e1000 0000:00:08.0 enp0s8: renamed from eth1
[    1.660711] e1000 0000:00:03.0 enp0s3: renamed from eth0
```

2) Criação da VM de firewall e DNS primário

Iremos agora criar a primeira máquina virtual efetiva de nosso *datacenter* simulado, a máquina `ns1`. Essa máquina atuará como um firewall de borda e DNS primário da rede, como configuraremos a seguir. Por se tratar de um firewall, é necessário que ela possua ao menos duas (ou, em nosso caso específico, três) interfaces de rede interconectando redes distintas.

1. Clone a máquina `debian-template` seguindo os mesmos passos da atividade (6) da sessão 1. Para o nome da máquina, escolha `ns1`.
2. Após a clonagem, na janela principal do Virtualbox, clique com o botão direito sobre a VM `ns1` e depois em *Settings*.

   Em *Network* > *Adapter 1* > *Attached to*, mantenha escolhida a opção *Bridged Adapter*, já que esta será a interface de conexão externa da máquina.

   Em *Adapter 2*, marque a caixa *Enable Network Adapter* e em *Attached to* selecione *Host-only Adapter*. O nome da rede *host-only* deve ser o mesmo alocado para a **DMZ**, como indicado na atividade (1) desta sessão. Seguindo o exemplo mostrado na figura da atividade, a rede escolhida seria portanto a `Virtualbox Host-Only Ethernet Adapter #2`.

   Em *Adapter 3*, marque a caixa *Enable Network Adapter* e em *Attached to* selecione *Host-only Adapter*. O nome da rede *host-only* deve ser o mesmo alocado para a **Intranet**, como indicado na atividade (1) desta sessão. Seguindo o exemplo da figura, escolheríamos então `Virtualbox Host-Only Ethernet Adapter #3`.

   Clique em *OK*, e ligue a máquina `ns1`.

3. Após o *boot*, faça login como o usuário `root`. Primeiro, vamos configurar a rede: edite o arquivo `/etc/network/interfaces` como se segue:

```
# nano /etc/network/interfaces
(...)
```

```
# cat /etc/network/interfaces
source /etc/network/interfaces.d/*

auto lo enp0s3 enp0s8 enp0s9

iface lo inet loopback

iface enp0s3 inet dhcp

iface enp0s8 inet static
address 10.0.42.1/24

iface enp0s9 inet static
address 192.168.42.1/24
```

Para garantir que nenhum endereço IP antigo, primário, se mantenha alocado às interfaces, execute o comando `flush` e em seguida reinicie a rede do sistema:

```
# ip addr flush label 'enp0s*' ; systemctl restart networking
```

Verifique que as interfaces estão com os endereços corretamente alocados:

```
# ip addr show label 'enp0s*' | grep 'inet ' | awk '{print $2,$NF}'
192.168.29.104/24 enp0s3
10.0.42.1/24 enp0s8
192.168.42.1/24 enp0s9
```

4. Usando o script `/root/scripts/changehost.sh` que criamos anteriormente, renomeie a máquina:

```
# hostname
debian-template
```

```
# bash ~/scripts/changehost.sh ns1
```

```
# hostname
ns1
```

3) Configuração inicial do firewall

Para garantir a segurança da rede iremos configurar o firewall de forma extremamente restritiva, como se segue:

a. Tráfego oriundo do firewall (*chain* OUTPUT) será permitido.
b. Todo o tráfego na interface *loopback* será permitido.
c. Serão permitidos pacotes destinados ao firewall (*chain* INPUT) ou passando pelo firewall (*chain* FORWARD) cujo estado seja relacionado ou estabelecido.
d. Serão permitidos pacotes ICMP oriundos das redes DMZ e Intranet com destino ao firewall *FWGW1-G*.
e. Será permitida gerência via `ssh` do firewall a partir de máquinas da Intranet.
f. Será autorizado o tráfego na Internet das máquinas da DMZ e Intranet **exclusivamente** nas portas TCP/80 e TCP/443.
g. Todos os demais acessos serão bloqueados.

À medida que novas regras forem necessárias, nas sessões seguintes, iremos criar regras de exceção pontualmente durante a execução das atividades, explicando os motivos das liberações. Vamos,

ponto a ponto, realizar as configurações explicitadas acima:

1. Primeiro, como em qualquer firewall de rede, devemos habilitar o repasse de pacotes entre interfaces. Para isso, edite o arquivo `/etc/sysctl.conf` e descomente a linha com a diretiva `net.ipv4.ip_forward`, como se segue:

   ```
   # sed -i '/net.ipv4.ip_forward/s/^#//' /etc/sysctl.conf
   ```

   Processe as alterações no arquivo com:

   ```
   # sysctl -p
   net.ipv4.ip_forward = 1
   ```

   Observe que esse arquivo é lido durante o *boot* do sistema, o que garante que nossa configuração perdurará mesmo após reiniciarmos a máquina.

2. Agora, vamos retomar as diretivas informadas no início desta atividade: para a requisição (a) não precisamos fazer nada, já que a política da *chain* OUTPUT encontra-se em ACCEPT:

   ```
   # iptables -L OUTPUT
   Chain OUTPUT (policy ACCEPT)
   target     prot opt source               destination
   ```

3. A diretiva (b) pode ser atendida com a regra que se segue:

   ```
   # iptables -A INPUT -i lo -j ACCEPT
   ```

   Já tratamos do caso da *chain* OUTPUT, e não faz sentido falarmos em tráfego na interface *loopback* na *chain* FORWARD.

4. Para a diretiva (c) devemos usar uma regra de estados nas *chains* INPUT e FORWARD, da seguinte forma:

   ```
   # iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
   ```

   ```
   # iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
   ```

5. Como a diretiva (d) diz especificamente de pacotes "com destino ao firewall", fica claro que devemos adicionar uma regra à *chain* INPUT. Note ainda que a diretiva é precisa em especificar que **apenas** o tráfego ICMP das redes DMZ e Intranet deve ser autorizado, e nenhum outro. Finalmente, não é especificado qual tipo de pacote ICMP será aceito, o que nos permite deduzir que todos serão aceitos.

Para inserir uma regra que inclua ambas as redes de origem, basta separá-las com vírgula, como se segue:

```
# iptables -A INPUT -s 10.0.42.0/24,192.168.42.0/24 -p icmp -m icmp --icmp-type any
-j ACCEPT
```

6. A diretiva (e) é clara ao especificar que a gerência será via `ssh` (portanto, na porta 22 do protocolo TCP), a ser feita no firewall (ou seja, *chain* INPUT), e apenas a partir de máquinas da Intranet. Podemos atender a esse requisito com a seguinte regra:

   ```
   # iptables -A INPUT -s 192.168.42.0/24 -p tcp -m tcp --dport 22 -j ACCEPT
   ```

7. A diretiva (f) diz que o tráfego na Internet das máquinas da DMZ e Intranet deve ser autorizado apenas nas portas TCP/80 e TCP/443. O requisito de IP de origem é bastante claro, mas o de destino não — de fato, máquinas na Internet podem ter, a princípio, qualquer endereço IP. Faz sentido, então, indicarmos a interface de saída dos pacotes, `enp0s3`.

   Outro aspecto a ser observado é que os pacotes desta vez não se destinam ao firewall, mas sim passam por ele para atingir máquinas na Internet, indicando que a regra deve ser inserida na *chain* FORWARD.

   Podemos ainda usar o módulo `multiport` para evitar a digitação de duas regras similares. Então, execute:

   ```
   # iptables -A FORWARD -s 10.0.42.0/24,192.168.42.0/24 -o enp0s3 -p tcp -m multiport
   --dports 80,443 -j ACCEPT
   ```

   Há ainda que se considerar a necessidade de realizar a tradução dos endereços de saída, pois não é possível que as máquinas da DMZ/Intranet naveguem com seus IPs em faixas privadas. Temos que criar uma regra de SNAT para permitir a navegação — levando em conta que o endereço da interface `enp0s3` é dinâmico, faz sentido usar o alvo MASQUERADE, nesse caso.

   Um adendo final: podemos fazer uma regra tão restritiva quanto a que fizemos na *chain* FORWARD acima, especificando também o protocolo e porta em que o SNAT será realizado. Tenha em mente apenas que, em caso de adição de exceções futuras, será necessário adicionar o protocolo/porta de exceção em **ambas** as *chains*, `filter/FORWARD` e `nat/POSTROUTING`.

   A regra fica assim:

   ```
   # iptables -t nat -A POSTROUTING -s 10.0.42.0/24,192.168.42.0/24 -o enp0s3 -p tcp
   -m multiport --dports 80,443 -j MASQUERADE
   ```

8. Atender a diretiva (g) final é bastante fácil: basta alterar a política das *chains* INPUT e FORWARD para DROP:

```
# iptables -P INPUT DROP
```

```
# iptables -P FORWARD DROP
```

9. Verifique a configuração final do firewall, comparando com os requisitos iniciais. Consulte primeiro a tabela *filter*:

```
# iptables -L -vn
Chain INPUT (policy DROP 189 packets, 52988 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
  982 67024 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
state RELATED,ESTABLISHED
    0     0 ACCEPT     icmp --  *      *       10.0.42.0/24         0.0.0.0/0
icmptype 255
    0     0 ACCEPT     icmp --  *      *       192.168.42.0/24      0.0.0.0/0
icmptype 255
    0     0 ACCEPT     tcp  --  *      *       192.168.42.0/24      0.0.0.0/0
tcp dpt:22

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
state RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  *      enp0s3  10.0.42.0/24         0.0.0.0/0
multiport dports 80,443
    0     0 ACCEPT     tcp  --  *      enp0s3  192.168.42.0/24      0.0.0.0/0
multiport dports 80,443

Chain OUTPUT (policy ACCEPT 16 packets, 1264 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

E, depois, a tabela *nat*:

```
# iptables -L -vn -t nat
Chain PREROUTING (policy ACCEPT 10 packets, 1485 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain INPUT (policy ACCEPT 6 packets, 1012 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 MASQUERADE  tcp  --  *      enp0s3  10.0.42.0/24         0.0.0.0/0
multiport dports 80,443
    0     0 MASQUERADE  tcp  --  *      enp0s3  192.168.42.0/24      0.0.0.0/0
multiport dports 80,443
```

10. As configurações realizadas até aqui estão todas em memória — em caso de *reboot* da máquina `ns1`, elas serão perdidas. Para gravar as regras e integrá-las ao sistema de *init* do SO, o pacote `iptables-persistent` é uma excelente opção. Instale-o com:

```
# apt-get install iptables-persistent
```

Na instalação do pacote, quando perguntado, responda:

Tabela 4. Configurações do iptables-persistent

Pergunta	Resposta
Salvar as regras IPv4 atuais?	Sim
Salvar as regras IPv6 atuais?	Sim

As regras IPv4 e IPv6 serão gravadas nos arquivos `/etc/iptables/rules.v4` e `/etc/iptables/rules.v6`, respectivamente. Em caso de alterações futuras nas configurações do firewall, é possível gravá-las de forma fácil com o comando:

```
# /etc/init.d/netfilter-persistent save
[....] Saving netfilter rules...run-parts: executing /usr/share/netfilter-
persistent/plugins.d/15-ip4tables save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
done.
```

Se quiser limpar todas as regras de firewall e começar do zero, execute:

```
# /etc/init.d/netfilter-persistent flush
[....] Flushing netfilter rules...run-parts: executing /usr/share/netfilter-
persistent/plugins.d/15-ip4tables flush
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables flush
done.
```

```
# iptables -L -vn
Chain INPUT (policy ACCEPT 13 packets, 1558 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 11 packets, 1068 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Se cometer qualquer erro durante o processo de configuração ou simplesmente quiser recarregar o conjunto de regras gravado em `/etc/iptables/rules.v4`, execute:

```
# /etc/init.d/netfilter-persistent reload
[....] Loading netfilter rules...run-parts: executing /usr/share/netfilter-
persistent/plugins.d/15-ip4tables start
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables start
done.
```

```
# iptables -L -vn
Chain INPUT (policy DROP 70 packets, 19740 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
  111  5861 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
state RELATED,ESTABLISHED
    0     0 ACCEPT     icmp --  *      *       10.0.42.0/24         0.0.0.0/0
icmptype 255
    0     0 ACCEPT     icmp --  *      *       192.168.42.0/24      0.0.0.0/0
icmptype 255
    0     0 ACCEPT     tcp  --  *      *       192.168.42.0/24      0.0.0.0/0
tcp dpt:22

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
state RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  *      enp0s3  10.0.42.0/24         0.0.0.0/0
multiport dports 80,443
    0     0 ACCEPT     tcp  --  *      enp0s3  192.168.42.0/24      0.0.0.0/0
multiport dports 80,443

Chain OUTPUT (policy ACCEPT 120 packets, 12617 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

4) Configuração do servidor DNS primário

Iremos agora configurar a máquina `ns1` como o servidor DNS primário da rede; para tanto, usaremos os programas NSD e Unbound.

O NSD (*Name Server Daemon*) é um servidor DNS *open-source* desenvolvido pela NLNet de Amsterdã em parceria com a RIPE NCC, o registro regional de Internet da Europa, oeste da Ásia e ex-países da União Soviética. Ele foi projetado para atuar exclusivamente como um servidor DNS autoritativo, não implementando portanto funções de *cache* recursiva como sua contraparte mais famosa, o BIND. A ideia por trás disso é aumentar a variabilidade de implementações DNS disponíveis, aumentando a resiliência do sistema DNS contra falhas de software e *exploits*.

O Unbound, por outro lado, é um *resolver* DNS com funções de validação, *cache* e recursividade também implementado pela NLNet Labs. Como se pode observar, então, ele faz as funções complementares às do NSD. Por sua percepção como sendo um software mais moderno, enxuto e seguro que seus concorrentes, o Unbound foi adotado como o *resolver* padrão do sistema-base dos sistemas FreeBSD e OpenBSD em 2014.

1. O primeiro passo, naturalmente, é instalar os pacotes dos programas mencionados:

```
# apt-get install nsd unbound dnsutils
```

2. Vamos começar configurando o **NSD**. Primeiro, pare o *daemon*:

```
# systemctl stop nsd
```

O programa pode ser controlado no terminal usando o comando `nsd-control` — para tanto, é necessário gerar as chaves TLS de controle com o comando:

```
# nsd-control-setup
setup in directory /etc/nsd
nsd_server.key exists
nsd_control.key exists
create nsd_server.pem (self signed certificate)
create nsd_control.pem (signed client certificate)
Signature ok
subject=CN = nsd-control
Getting CA Private Key
Setup success. Certificates created.
```

Verifique a criação das chaves com o comando:

```
# ls -1 /etc/nsd | grep '.key\|.pem'
nsd_control.key
nsd_control.pem
nsd_server.key
nsd_server.pem
```

3. Faça um backup do arquivo de configuração original `/etc/nsd/nsd.conf`:

```
# mv /etc/nsd/nsd.conf /etc/nsd/nsd.conf.orig
```

Em seguida, crie o arquivo novo `/etc/nsd/nsd.conf` com o seguinte conteúdo:

```
server:
  ip-address: 127.0.0.1
  ip-address: 10.0.42.1
  do-ip4: yes
  port: 8053
  username: nsd
  zonesdir: "/etc/nsd/zones"

  logfile: "/var/log/nsd.log"
  pidfile: "/run/nsd/nsd.pid"
  hide-version: yes
  version: "intnet DNS"
  identity: "unidentified server"

remote-control:
  control-enable: yes
  control-interface: 127.0.0.1
  control-port: 8952
  server-key-file: "/etc/nsd/nsd_server.key"
  server-cert-file: "/etc/nsd/nsd_server.pem"
  control-key-file: "/etc/nsd/nsd_control.key"
  control-cert-file: "/etc/nsd/nsd_control.pem"

key:
  name: "inkey"
  algorithm: sha512
  secret: "TSIGKEY"

pattern:
  name: "inslave"
  notify: 10.0.42.2@8053 inkey
  provide-xfr: 10.0.42.2 inkey

zone:
  name: "intnet"
  include-pattern: "inslave"
  zonefile: "intnet.zone"

zone:
  name: "42.0.10.in-addr.arpa"
  include-pattern: "inslave"
  zonefile: "10.0.42.zone"
```

Abaixo, destacamos e explicamos algumas das diretivas mais relevantes do arquivo acima:

- `server/ip-address`: define os endereços de rede nos quais o NSD irá escutar — atenderemos requisições de *localhost* e do servidor DNS secundário, que será instalado na DMZ a seguir. O controle de quais *hosts* estarão autorizados a consultar o NSD será feito via diretivas `notify` e `provide-xfr`, bem como restrição via regra de firewall.

- `server/port`: define a porta em que o NSD irá escutar; como iremos instalar o Unbound na mesma máquina (escutando na porta 53/UDP), escolhemos a porta alternativa 8053 para evitar conflitos de *bind*.
- `server/username`: usuário com o qual o NSD irá operar. É possível aplicar um controle adicional além da redução de privilégios para um usuário comum, o uso de *chroot*, que não faremos nesta atividade.
- `server/zonesdir`: diretório em que serão armazenados os arquivos de zonas do NSD.
- `server/hide-version`, `server/version` e `server/identity`: os três controles objetivam mascarar o software rodando na máquina local, dificultando a tarefa de *banner grabbing* e identificação de *exploits* por parte de eventuais atacantes.
- `remote-control/control-enable` e `remote-control/interface`: define que o NSD poderá ser controlado "remotamente", mas logo depois restringe o IP de escuta para 127.0.0.1, efetivamente autorizando conexão de controle exclusivamente a partir de *localhost*.
- `key/secret`: define uma chave secreta que servidores primário e secundário deverão possuir em comum para que a transferência de zona seja feita com sucesso. Faremos a geração dessa chave e sua substituição de forma automática no arquivo a seguir.
- `pattern`: define um conjunto de diretivas que serão inseridas em diversas zonas a seguir (efetivamente, como uma *macro*). No caso, estamos indicando que o servidor 10.0.42.2 será notificado de alterações de zona (`notify`), e transferências de zona partir desse endereço serão autorizadas (`provide-xfr`).
- `zone/intnet`: define um arquivo de zona direta para o qual o servidor NSD será autoritativo, o nome de domínio `intnet`. A sintaxe do arquivo é idêntica à usada no software BIND.
- `zone/42.0.10-in-addr.arpa`: define um arquivo de zona reversa para o qual o servidor NSD será autoritativo, a faixa de endereços `10.0.42.0/24`. A sintaxe do arquivo é idêntica à usada no software BIND.

A transferência de zonas entre servidor primário e secundário é assegurada, além dos controles de interface de escuta e regras de firewall, por uma chave secreta TSIG (*Transaction SIGnature*) em formato Base64. Para inseri-la no arquivo use o comando:

```
# tsigkey_t=$( dd if=/dev/urandom of=/dev/stdout count=1 bs=32 2> /dev/null| base64
); sed -i "s|TSIGKEY|$tsigkey_t|" /etc/nsd/nsd.conf ; unset tsigkey_t
```

Verifique a inserção da chave com:

```
# grep 'secret:' /etc/nsd/nsd.conf
  secret: "i7IoB5VDHVOCW9wvuOGQuFLNu8hzfAblVAbCD1SbPL4="
```

Lembre-se que precisaremos dessa mesma chave quando estivermos configurando o servidor secundário.

4. Crie o diretório de zonas `/etc/nsd/zones`:

```
# mkdir /etc/nsd/zones
```

Agora, crie o arquivo novo `/etc/nsd/zones/intnet.zone`, com as configuração de zona direta para o domínio `intnet.`, com o seguinte conteúdo:

```
$TTL 86400 ; (1 day)
$ORIGIN intnet.

@       IN   SOA   ns1.intnet.  admin.intnet. (
                   2018111000   ;serial (YYYYMMDDnn)
                   14400        ;refresh (4 hours)
                   1800         ;retry (30 minutes)
                   1209600      ;expire (2 weeks)
                   3600         ;negative cache TTL (1 hour)
                   )

@       IN   NS                 ns1.intnet.
@       IN   NS                 ns2.intnet.

@       IN   MX    10           mx1.intnet.
@       IN   MX    20           mx2.intnet.

ns1     IN    A                 10.0.42.1
ns2     IN    A                 10.0.42.2

mx1     IN    A                 10.0.42.91
mx2     IN    A                 10.0.42.92

fw      IN    CNAME             ns1
```

A sintaxe é exatamente a mesma utilizada para um arquivo de zonas do BIND. Note que:

- Definimos a máquina `ns1.intnet.` como o servidor autoritativo para o domínio `intnet.`, com email de contato `admin@intnet`.
- O serial é `2018111000` — este é um número que deve identificar a versão do arquivo de zonas, e incrementado a cada atualização. O formato escolhido para esse *serial* foi `ANO-MES-DIA-VERSAO`, de forma que versões do arquivo em datas futuras terão sempre um valor superior ao de versões antigas.
- Os servidores DNS responsáveis pelo domínio são `ns1.intnet` e `ns2.intnet.`.
- Os servidores de e-mail (MX — *mail exchange*) do domínio são `mx1.intnet.` e `mx2.intnet.`. Ambos são servidores fictícios, que não implantaremos neste curso, e mostrados aqui apenas para demonstrar a sintaxe desse tipo de entrada no arquivo de zonas.
- Configuram-se registros A (*address*) para as máquinas mencionadas.
- Configura-se um registro CNAME (*canonical name*) para a máquina `ns1.intnet.`, `fw.intnet.`. Com isso, os usuário poderão referir-se a essa máquina também pelo seu "apelido".

5. Vamos para o arquivo de zona reversa. Crie o arquivo novo `/etc/nsd/zones/10.0.42.zone` com o seguinte conteúdo:

```
$TTL 86400 ; (1 day)
$ORIGIN 42.0.10.in-addr.arpa.

@       IN   SOA   ns1.intnet.  admin.intnet. (
                   2018111000   ;serial (YYYYMMDDnn)
                   14400        ;refresh (4 hours)
                   1800         ;retry (30 minutes)
                   1209600      ;expire (2 weeks)
                   3600         ;negative cache TTL (1 hour)
                   )

@       IN   NS                 ns1.intnet.
@       IN   NS                 ns2.intnet.

@       IN   MX    10           mx1.intnet.
@       IN   MX    20           mx2.intnet.

1       IN   PTR                ns1.intnet.
2       IN   PTR                ns2.intnet.

91      IN   PTR                mx1.intnet.
92      IN   PTR                mx2.intnet.
```

Nada muito diferente neste arquivo em relação ao anterior — note apenas que os registros inseridos aqui são do tipo PTR (*pointer*), fazendo o mapeamento reverso entre endereços IP e nomes de domínio. Note, ainda, que o *ORIGIN* do arquivo é definido como `42.0.10.in-addr.arpa.`, já que somos o servidor autoritativo para a faixa 10.0.42.0/24.

6. Vamos verificar se a sintaxe dos arquivos de configuração não possui erros:

```
# nsd-checkconf /etc/nsd/nsd.conf
```

Agora, basta reiniciar o serviço e verificar sua correta operação:

```
# nsd-control start
```

```
# tail /var/log/nsd.log
[2018-11-13 17:17:26.763] nsd[2012]: notice: nsd starting (NSD 4.1.14)
[2018-11-13 17:17:26.780] nsd[2014]: notice: nsd started (NSD 4.1.14), pid 2013
```

```
# ss -tunlp | grep 8053
udp    UNCONN     0      0      10.0.42.1:8053                  *:*
users:(("nsd",pid=648,fd=5),("nsd",pid=647,fd=5),("nsd",pid=646,fd=5))
udp    UNCONN     0      0      127.0.0.1:8053                  *:*
users:(("nsd",pid=648,fd=4),("nsd",pid=647,fd=4),("nsd",pid=646,fd=4))
tcp    LISTEN     0      128    10.0.42.1:8053                  *:*
users:(("nsd",pid=648,fd=7),("nsd",pid=647,fd=7),("nsd",pid=646,fd=7))
tcp    LISTEN     0      128    127.0.0.1:8053                  *:*
users:(("nsd",pid=648,fd=6),("nsd",pid=647,fd=6),("nsd",pid=646,fd=6))
```

7. Vamos colocar o servidor à prova: teste a resolução direta de nomes usando a ferramenta `dig`. Lembre-se que o NSD está operando na porta 8053/UDP, e não na porta padrão:

```
# dig @127.0.0.1 -p 8053 fw.intnet +noadditional +noquestion

; <<>> DiG 9.10.3-P4-Debian <<>> @127.0.0.1 -p 8053 fw.intnet +noadditional
+noquestion
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64874
;; flags: qr aa rd; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 2
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; ANSWER SECTION:
fw.intnet.              86400   IN      CNAME   ns1.intnet.
ns1.intnet.             86400   IN      A       10.0.42.1

;; AUTHORITY SECTION:
intnet.                 86400   IN      NS      ns1.intnet.
intnet.                 86400   IN      NS      ns2.intnet.

;; Query time: 0 msec
;; SERVER: 127.0.0.1#8053(127.0.0.1)
;; WHEN: Mon Nov 12 09:19:30 -02 2018
;; MSG SIZE  rcvd: 120
```

Excelente! Todas as seções de resposta parecem corretas, tanto ANSWER quando AUTHORITY. Vamos verificar a resolução reversa:

```
# dig @127.0.0.1 -p 8053 -x 10.0.42.2 +noadditional +noquestion

; <<>> DiG 9.10.3-P4-Debian <<>> @127.0.0.1 -p 8053 -x 10.0.42.2 +noadditional
+noquestion
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4272
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; ANSWER SECTION:
2.42.0.10.in-addr.arpa. 86400   IN      PTR     ns2.intnet.

;; AUTHORITY SECTION:
42.0.10.in-addr.arpa.   86400   IN      NS      ns1.intnet.
42.0.10.in-addr.arpa.   86400   IN      NS      ns2.intnet.

;; Query time: 0 msec
;; SERVER: 127.0.0.1#8053(127.0.0.1)
;; WHEN: Mon Nov 12 09:19:47 -02 2018
;; MSG SIZE  rcvd: 107
```

8. Agora que o NSD está configurado, vamos implementar o **Unbound**. Primeiro, pare o *daemon* se estiver rodando:

```
# systemctl stop unbound
```

Assim como no caso do NSD, o Unbound pode ser controlado via `unbound-control` — gere as chaves de controle:

```
# unbound-control-setup
setup in directory /etc/unbound
unbound_server.key exists
unbound_control.key exists
create unbound_server.pem (self signed certificate)
create unbound_control.pem (signed client certificate)
Signature ok
subject=CN = unbound-control
Getting CA Private Key
Setup success. Certificates created.
```

Faça um backup do arquivo de configuração original `/etc/unbound/unbound.conf`:

```
# mv /etc/unbound/unbound.conf /etc/unbound/unbound.conf.orig
```

Em seguida, crie o arquivo novo `/etc/unbound/unbound.conf` com o seguinte conteúdo:

```
server:
  interface: 127.0.0.1
  interface: 10.0.42.1
  interface: 192.168.42.1
  port: 53

  access-control: 127.0.0.0/8 allow
  access-control: 10.0.42.0/24 allow
  access-control: 192.168.42.0/24 allow

  cache-min-ttl: 300
  cache-max-ttl: 14400

  local-zone: "intnet" nodefault
  domain-insecure: "intnet"

  local-zone: "10.in-addr.arpa." nodefault
  domain-insecure: "10.in-addr.arpa."

  verbosity: 1
  prefetch: yes
  hide-version: yes
  hide-identity: yes
  use-caps-for-id: yes
  rrset-roundrobin: yes
  minimal-responses: yes
  do-not-query-localhost: no

stub-zone:
  name: "intnet"
  stub-addr: 127.0.0.1@8053

stub-zone:
  name: "42.0.10.in-addr.arpa."
  stub-addr: 127.0.0.1@8053

forward-zone:
  name: "."
  forward-addr: 8.8.8.8
  forward-addr: 8.8.4.4

include: "/etc/unbound/unbound.conf.d/*.conf"
```

Abaixo, destacamos e explicamos algumas das diretivas mais relevantes do arquivo acima:

- `server/interface`: define os endereços de rede nos quais o Unbound irá escutar — atenderemos requisições de *localhost* e das sub-redes DMZ e Intranet.
- `server/port`: o Unbound irá escutar na porta padrão, 53/UDP.
- `server/access-control`: define que as sub-redes declaradas (*localhost*, DMZ e Intranet, novamente) terão permissão para utilizar os serviços de resolução de nomes.
- `server/local-zone` e `server/domain-insecure`: define opções para as zonais locais `intnet.` e `42.0.10.in-addr.arpa.`, e que a cadeia de confiança DNSSEC não será verificada para esses domínios.
- `server/hide-version` e `server/hide-identity`: controles que objetivam mascarar o software rodando na máquina local, dificultando a tarefa de *banner grabbing* e identificação de *exploits* por parte de eventuais atacantes.
- `stub-zone` e `stub-zone/stub-addr`: declara zonas autoritativas locais que devem ser consultadas para os domínios especificados, para as quais o registro público DNS não será usado. Usaremos o servidor NSD rodando em *localhost*, consultando a porta 8053/UDP.
- `forward-zone` e `forward-zone/forward-addr`: define servidores nos quais o Unbound irá buscar recursão caso não consiga resolver um nome. Como está sendo declarado o nome de zona `"."`, todas as pesquisas serão redirecionadas para os servidores indicados em `forward-addr`.

9. Vamos verificar o funcionamento do *daemon*. Inicie o Unbound usando o `unbound-control`:

```
# unbound-control start
```

Verifique seu funcionamento usando o `ss`:

```
# ss -unlp | grep :53
UNCONN     0      0      192.168.42.1:53                  *:*
users:(("unbound",pid=2084,fd=7))
UNCONN     0      0      10.0.42.1:53                     *:*
users:(("unbound",pid=2084,fd=5))
UNCONN     0      0      127.0.0.1:53                     *:*
users:(("unbound",pid=2084,fd=3))
```

Reconfigure o DNS *system-wide*, no arquivo `/etc/resolv.conf`.

```
# nano /etc/resolv.conf
(...)
```

```
# cat /etc/resolv.conf
domain intnet.
search intnet.
nameserver 10.0.42.1
nameserver 10.0.42.2
```

Devido ao fato de estarmos usando DHCP na interface de saída (`enp0s3`), o arquivo `/etc/resolv.conf` poderá ser sobrescrito sempre que as interfaces de rede forem reiniciadas, ou após o *reboot* da máquina. Para evitar isso, marque o arquivo como imutável:

```
# chattr +i /etc/resolv.conf
```

10. Feito! Vamos testar a resolução de domínios internos `unbound` com a ferramenta `dig`.

```
# dig fw.intnet +noquestion +noadditional

; <<>> DiG 9.10.3-P4-Debian <<>> fw.intnet +noquestion +noadditional
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 63613
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; ANSWER SECTION:
fw.intnet.              86400   IN      CNAME   ns1.intnet.
ns1.intnet.             86400   IN      A       10.0.42.1

;; Query time: 0 msec
;; SERVER: 10.0.42.1#53(10.0.42.1)
;; WHEN: Mon Nov 12 09:39:55 -02 2018
;; MSG SIZE  rcvd: 72
```

E quanto a nomes de domínio externos, usando recursão? Vejamos:

```
# dig openbsd.org +noquestion +noadditional

; <<>> DiG 9.10.3-P4-Debian <<>> openbsd.org +noquestion +noadditional
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5694
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; ANSWER SECTION:
openbsd.org.            21599   IN      A       129.128.5.194

;; Query time: 420 msec
;; SERVER: 10.0.42.1#53(10.0.42.1)
;; WHEN: Mon Nov 12 09:40:33 -02 2018
;; MSG SIZE  rcvd: 56
```

11. Ainda não acabou! Lembre-se que nossas regras de firewall configuradas no começo desta sessão estão extremamente restritivas, então temos que criar exceções para que a consulta de nomes funcione. Vamos ver os tipos de acesso que precisamos liberar:

    a. O NSD será consultado apenas pelo Unbound em *localhost* (não é necessário criar regras neste caso), e pelo servidor DNS secundário em 10.0.42.2. Como nesse caso será feita transferência de zonas entre os servidores, é necessário liberar tráfego nos protocolos TCP e UDP na porta 8053.

    b. O Unbound local será consultado por todas as máquinas da DMZ e Intranet, na porta 53/UDP.

    c. Os clientes da Intranet devem conseguir consultar o Unbound rodando no servidor DNS secundário, porta 53/UDP. Não é necessário criar uma regra para a DMZ, neste caso, pois as máquinas estão na mesma sub-rede e portanto seu tráfego não passa pelo firewall (na *chain* FORWARD).

    Vamos ao requisito (a):

    ```
    # iptables -A INPUT -s 10.0.42.2/32 -p tcp -m tcp --dport 8053 -j ACCEPT
    ```

    ```
    # iptables -A INPUT -s 10.0.42.2/32 -p udp -m udp --dport 8053 -j ACCEPT
    ```

    Agora, o requisito (b):

    ```
    # iptables -A INPUT -s 10.0.42.0/24,192.168.42.0/24 -p udp -m udp --dport 53 -j
    ACCEPT
    ```

    E, finalmente, o critério (c) estabelecido inicialmente:

    ```
    # iptables -A FORWARD -s 192.168.42.0/24 -d 10.0.42.2/32 -p udp -m udp --dport 53
    -j ACCEPT
    ```

    Salve as regras na configuração do firewall local:

    ```
    # /etc/init.d/netfilter-persistent save
    [....] Saving netfilter rules...run-parts: executing /usr/share/netfilter-
    persistent/plugins.d/15-ip4tables save
    run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
    done.
    ```

5) Configuração do DNSSEC

Vamos agora configurar o DNSSEC em nosso servidor autoritativo, de forma a produzir consultas assinadas digitalmente. Evidentemente, como estamos configurando um domínio fictício não

iremos conseguir propagar a assinatura pela cadeia de confiança raiz, mas esta atividade servirá como um exemplo caso você queira fazê-lo em sua organização posteriormente.

1. Primeiro, vamos instalar as ferramentas de suporte para gestão/geração de chaves DNSSEC:

```
# apt-get install ldnsutils haveged
```

2. Vamos criar as chaves de assinatura de zona (ZSK, ou *Zone Signing Key*) e chave (KSK, ou *Key Signing Key*). O KSK é um par de chaves público/privada usado para gerar uma assinatura digital para o ZSK. O ZSK, por sua vez, é um par de chaves público/privada para gerar uma assinatura digital conhecida como RRSIG (*Resource Record Signature*) para cada um dos RRSETs (*Resource Record Sets*) da zona. O website da Cloudflare (https://www.cloudflare.com/dns/dnssec/how-dnssec-works/) possui uma excelente explicação sobre o sistema DNSSEC que pode ser usada para consulta.

   Primeiro, entre no diretório `/etc/nsd`:

```
# cd /etc/nsd/
```

   Agora, vamos gerar o ZSK/KSK e armazenar seus nomes em uma variável temporária. Os algoritmos escolhidos estão de acordo com as melhores práticas recomendadas na RFC 6944 (https://tools.ietf.org/html/rfc6944).

```
# export ZSK=$( ldns-keygen -a RSASHA512 -b 2048 intnet )
```

```
# export KSK=$( ldns-keygen -k -a RSASHA512 -b 2048 intnet )
```

   Podemos remover o registro DS (*Delegation of Signing*) auto-gerado, já que iremos regerá-lo futuramente sob demanda.

```
# rm Kintnet.*.ds
```

   Vejamos como ficaram as chaves ZSK/KSK para o domínio:

```
# ls -1 Kintnet.*
Kintnet.+010+14936.key
Kintnet.+010+14936.private
Kintnet.+010+42867.key
Kintnet.+010+42867.private
```

   Para identificar qual dessas chaves é a ZSK e qual é a KSK, consulte as variáveis do *shell*:

```
# echo $ZSK
Kintnet.+010+14936
```

```
# echo $KSK
Kintnet.+010+42867
```

3. Agora, vamos assinar a zona `intnet.zone` com o comando `ldns-signzone`:

```
# ldns-signzone -n -p -s $(head -n 1000 /dev/urandom | sha1sum | cut -b 1-16)
/etc/nsd/zones/intnet.zone $ZSK $KSK
```

O que ocorreu? Verifique o conteúdo do diretório `/etc/nsd/zones`:

```
# ls -1 /etc/nsd/zones
10.0.42.zone
intnet.zone
intnet.zone.signed
```

Vamos configurar o NSD para usar a zona assinada. Para isso, basta substituir o caminho de busca do arquivo de zona direta `/etc/nsd/zones/intnet.zone` por sua contraparte assinada:

```
# sed -i 's/intnet\.zone/intnet\.zone\.signed/' /etc/nsd/nsd.conf
```

```
# grep -B3 intnet.zone.signed /etc/nsd/nsd.conf
zone:
  name: "intnet"
  include-pattern: "inslave"
  zonefile: "intnet.zone.signed"
```

Para informar ao NSD que o arquivo de zonas mudou, basta usar o programa `nsd-control` como mostrado a seguir:

```
# nsd-control reconfig
reconfig start, read /etc/nsd/nsd.conf
ok
```

```
# nsd-control reload intnet
ok
```

4. Vamos testar se nossa configuração surtiu efeito. Primeiro, pesquise pelos registros DNSKEY do domínio no NSD em *localhost*, verificando as chaves ZSK e KSK:

```
# dig -p 8053 @localhost DNSKEY intnet. +multiline +norec +noquestion

; <<>> DiG 9.10.3-P4-Debian <<>> -p 8053 @localhost DNSKEY intnet. +multiline
+norec +noquestion
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18768
;; flags: qr aa; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; ANSWER SECTION:
intnet.                 86400 IN DNSKEY 256 3 10 (
                                AwEAAerseiw1VUd7L90iBA4IgztUXpkw/k/6kKIHx5Qm
                                Cm1HvtLTT/dSFk9JOHVBoQ2Rpk0LQBc+WZsnqz6dQZJn
                                b9FkGzsAh39BnP8gOA1d+SfaGGkceEteQfgi3rKz5dnk
                                EWgOonWk2vGOJ5oPDvcy4aUQqfTACY9Tk0qeWZT7enfy
                                gdrg4AjaTKBdJ4kruBj2z3FKXBolDYO3HTPmosKe94Xr
                                muiLQa3uHFvFvu6k/X0mH1kCS+lb91R/OF60mG40iW4/
                                z8aLNSKPUuwJNddvZ0I14F7SwN7YGaVpoY11DAsxmbS5
                                1lGxp6Fc0fWQpAfT4WG9/mbyHj29kZnP/Oxktsk=
                                ) ; ZSK; alg = RSASHA512; key id = 14936
intnet.                 86400 IN DNSKEY 257 3 10 (
                                AwEAAdIo8BoUEY6ab3YBbs24FEj8Vt9aRByAFyKJk0sH
                                BiNIy9I1vXZIGB+5wlIKPzT4ZVd3oYO3MDExxDYZ1HVS
                                D1magCRBzLF2oeTCWY3kLz/9ls23RV9r5IMY68aPHMg0
                                tDBJ8IZTC9SbO+os3L9jazwpoL22Akl34YN6iIC6kXpQ
                                gN3TtNorzD8DIHkBvJ9NrpKYJjzt0g8oTyg0La3xTNcO
                                6q51Q4eVBnlcXbbJ6gquSFzAnoJ9qzq5JUnbvAB9I4yv
                                Spo3RGCX3lZFmsDvsDfnwkXShyeEPdGTr6m/mbqPrceW
                                R53QLOWnZq1KrczPwTlKjgtzwg+F7ol8YDdQOwU=
                                ) ; KSK; alg = RSASHA512; key id = 42867

;; Query time: 0 msec
;; SERVER: 127.0.0.1#8053(127.0.0.1)
;; WHEN: Mon Nov 12 11:13:56 -02 2018
;; MSG SIZE  rcvd: 587
```

Perfeito! Aparentemente, nossas chaves foram registradas e estão sendo utilizadas com sucesso. O Unbound, no entanto, ainda possui algumas entradas na *cache* desatualizadas, prévias à configuração do DNSSEC, que foram mantidas pelas consultas anteriores. Para limpá-las, basta usar o comando `unbound-control flush_zone`:

```
# unbound-control flush_zone intnet.
ok removed 4 rrsets, 2 messages and 0 key entries
```

Agora sim, faça uma consulta via Unbound usando DNSSEC:

```
# dig fw.intnet +dnssec +multiline +noquestion

; <<>> DiG 9.10.3-P4-Debian <<>> fw.intnet +dnssec +multiline +noquestion
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 47390
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; ANSWER SECTION:
fw.intnet.              86355 IN CNAME ns1.intnet.
fw.intnet.              86355 IN RRSIG CNAME 10 2 86400 (
                                20181210130559 20181112130559 14936 intnet.
                                u2DBcjAp91qYSYvspczvWiW8QuIX24aAoFTzfM8hJdUQ
                                3VcjekyJeQQfNPIM4+rE9ZdTywADS4nKCFl/XqkHwFST
                                kitVm9So17I4GBJ+sgpKkpoKEwh6IfbUB563Mmfj9jIQ
                                SvuR0dP0ot9QV+DXWCFLD02tvXTOBSltBT9Nogx1h926
                                5lkSZ0F+5x0Ss1Law5Rpn8ZKyAAwCYS0UNJIqB8XzIqb
                                eSblG5Q4Si7Qwrccv7Ll6DscBbh1gjmnCAYCE63XF3SN
                                s0N9SgTgfzibgVaVGw+S1SoVYjvxDVd5ZW/tpQxhx4kY
                                Wt/c+541nj7xmFT58/dnoo8t6Meo/JCKgw== )
ns1.intnet.             86297 IN A 10.0.42.1
ns1.intnet.             86297 IN RRSIG A 10 2 86400 (
                                20181210130559 20181112130559 14936 intnet.
                                bmwQPD9b/GwUX2ZvpP6baOzyEXNW/ghcc9rqzZSWwUYn
                                VfX+UqRwzon/2lxJfYO1uJh0v8noO2S5+zb7saroDsCi
                                9/Oaukw+iAvwrZh6VO1dHDr6WjAlS4hi9MxmfJ94NwD7
                                txB5WLG39KM0V/EpH+v4L6ser4v+oYFSer7a6EsZxCcD
                                nfciKtN5L/QUWVMacduxpRM16+MI83bEf/ulC82xIThf
                                2cVVdsDA4xfZvAOcYS/IXUauYXaj9fkE2TlVoyIC1UIM
                                XE6CSbulo2iWc+V+lj5tkP9kctmqit3rfegWXJlQ4es3
                                4fsxR64+Weqre/iTMpVxVss/TLus7g7OiA== )

;; Query time: 0 msec
;; SERVER: 10.0.42.1#53(10.0.42.1)
;; WHEN: Mon Nov 12 11:17:12 -02 2018
;; MSG SIZE  rcvd: 660
```

Caso fosse desejável exportar a configuração DNS para um *registrar* hierarquicamente superior (fechando a cadeia de verificação DNS), pode-se gerar os registros DS das chaves com o comando abaixo.

```
# ldns-key2ds -n -1 /etc/nsd/zones/intnet.zone.signed && ldns-key2ds -n -2
/etc/nsd/zones/intnet.zone.signed
intnet. 86400   IN      DS      42867 10 1 5b8c2c011bd011618f7e339667c075587e65ab05
intnet. 86400   IN      DS      42867 10 2
424bcf7d7c09df8be1520c4230680b5c33a63598b4c6185c7884592ad3bce63b
```

6) Automatizando assinatura DNSSEC após alterações

1. É claro, seria muito inconveniente ter que realizar todos os passos que fizemos na atividade (5) a cada alteração no servidor DNS. Vamos automatizar a assinatura de zonas e reinício dos *daemons* relevantes com um *script shell*. Crie o arquivo novo `/root/scripts/signzone-intnet.sh` com o seguinte conteúdo:

```
#!/bin/bash

ZSK="ZSKSTUB"
KSK="KSKSTUB"
ZONE_FILE="/etc/nsd/zones/intnet.zone"

cd /etc/nsd

ldns-signzone -n \
  -p \
  -s $(head -n 1000 /dev/random | sha1sum | cut -b 1-16) \
  $ZONE_FILE \
  $ZSK \
  $KSK

nsd-control reconfig
nsd-control reload intnet
nsd-control reload 42.0.10.in-addr.arpa
unbound-control flush_zone intnet.
```

   Note que é necessário substituir as variáveis `ZSKSTUB` e `KSKSTUB` com os valores da variáveis `$ZSK` e `$KSK` no seu *shell* corrente, como se segue:

```
# sed -i "s/ZSKSTUB/${ZSK}/" /root/scripts/signzone-intnet.sh
```

```
# sed -i "s/KSKSTUB/${KSK}/" /root/scripts/signzone-intnet.sh
```

2. Teste a assinatura de zonas usando o *script*:

```
# bash scripts/signzone-intnet.sh
reconfig start, read /etc/nsd/nsd.conf
ok
ok
ok
ok removed 4 rrsets, 2 messages and 0 key entries
```

Ao atualizar arquivos de zona, não se esqueça que além de adicionar registros A, CNAME ou PTR conforme necessário, é também **imprescindível** incrementar o valor do *serial* do arquivo, próximo do topo no registro SOA. Se isso não for feito, o NSD não irá detectar que o arquivo de zona foi atualizado e suas modificações não serão propagadas.

7) Reconfiguração da VM *debian-template*

Agora que temos um firewall e servidores DNS na rede é interessante que alteremos a configuração padrão da VM `debian-template` para que as novas máquinas, clonadas a partir dela, já estejam corretamente ajustadas para operar em nosso *datacenter* simulado.

1. Ligue a máquina `debian-template` e acesse o terminal como o usuário `root`.

```
# hostname ; whoami
debian-template
root
```

2. Com a máquina `debian-template` ligada, acesse na janela principal do Virtualbox o menu *Settings* > *Network* > *Adapter 1* > *Attached to* e altere a opção para *Host-only Adapter*. O nome da rede *host-only* escolhido deve ser o mesmo alocado para a interface de rede da máquina virtual `ns1` que está conectada à DMZ. Seguindo o exemplo mostrado no início desta sessão, portanto, a rede escolhida seria a `Virtualbox Host-Only Ethernet Adapter #2`.
3. De volta ao terminal da máquina `debian-template`, vamos configurar a rede: edite o arquivo `/etc/network/interfaces` como se segue:

```
# nano /etc/network/interfaces
(...)
```

```
# cat /etc/network/interfaces
source /etc/network/interfaces.d/*

auto lo enp0s3

iface lo inet loopback

iface enp0s3 inet static
address 10.0.42.253/24
gateway 10.0.42.1
```

Para garantir que nenhum endereço IP antigo, primário, se mantenha alocado às interfaces, execute o comando `flush` e em seguida reinicie a rede do sistema:

```
# ip addr flush label 'enp0s*' ; systemctl restart networking
```

Verifique que as interfaces estão com os endereços corretamente alocados:

```
# ip addr show label 'enp0s*' | grep 'inet ' | awk '{print $2,$NF}'
10.0.42.253/24 enp0s3
```

4. Agora, configure a resolução de nomes no arquivo `/etc/resolv.conf`:

```
# nano /etc/resolv.conf
(...)
```

```
# cat /etc/resolv.conf
domain intnet.
search intnet.
nameserver 10.0.42.1
nameserver 10.0.42.2
```

5. Vamos alterar o *script* `/root/scripts/changehost.sh` para que possamos configurar, de uma vez só, informações como endereço IP e *hostname* na máquina clonada. Apague todo o conteúdo do *script* original e substitua-o pelo seguinte:

```
#!/bin/bash


# exibir uso do script e sair
function usage() {
  echo "  Usage: $0 -h HOSTNAME -i IPADDR -g GATEWAY"
  echo "  Netmask is assumed as /24."
  exit 1
}


# testar sintaxe valida de HOSTNAME
function valid_host() {
  if [[ "$nhost" =~ [^a-z0-9] ]]; then
    echo "  [*] HOSTNAME must be lowercase alphanumeric: [a-z0-9]*"
    usage
  elif [ ${#nhost} -gt 63 ]; then
    echo "  [*] HOSTNAME must have <63 chars"
    usage
  fi
}


```

```
# testar sintaxe valida de IPADDR/GATEWAY
function valid_ip() {
  local  ip=$1
  local  stat=1

  if [[ $ip =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
    OIFS=$IFS
    IFS='.'
    ip=($ip)
    IFS=$OIFS
    [[ ${ip[0]} -le 255 && \
       ${ip[1]} -le 255 && \
       ${ip[2]} -le 255 && \
       ${ip[3]} -le 255 ]]
        stat=$?
  fi

  if [ $stat -ne 0 ] ; then
    echo "  [*] Invalid syntax for $2"
    usage
  fi
}


while getopts ":g:h:i:" opt; do
  case "$opt" in
    g)
      ngw=${OPTARG}
      ;;
    h)
      nhost=${OPTARG}
      ;;
    i)
      nip=${OPTARG}
      ;;
    *)
      usage
      ;;
  esac
done

# testar se parametros foram informados
[ -z $ngw ]   && { echo "  [*] No gateway?"; usage; }
[ -z $nhost ] && { echo "  [*] No hostname?"; usage; }
[ -z $nip ]   && { echo "  [*] No ipaddr?"; usage; }

# testar sintaxe de parametros
valid_ip $nip "IPADDR"
valid_ip $ngw "GATEWAY"
valid_host $nhost

```

```
# alterar endereco ip/gateway
iff="/etc/network/interfaces"
cip="$( egrep '^address ' $iff | awk -F'[ /]' '{print $2}' )"
cgw="$( egrep '^gateway ' $iff | awk '{print $NF}' )"
sed -i "s|${cip}|${nip}|g" $iff
sed -i "s|${cgw}|${ngw}|g" $iff
ip addr flush label 'enp0s*'

# alterar hostname local
chost="$( hostname -s )"
sed -i "s/${chost}/${nhost}/g" /etc/hosts
sed -i "s/${chost}/${nhost}/g" /etc/hostname

invoke-rc.d hostname.sh restart
invoke-rc.d networking restart
hostnamectl set-hostname $nhost

# re-gerar chaves SSH
rm -f /etc/ssh/ssh_host_* 2> /dev/null
dpkg-reconfigure openssh-server &> /dev/null
```

Em relação ao *script* original, algumas diferenças: adicionou-se um *loop* para leitura de opções a partir da linha de comando (*hostname*, endereço IP e *gateway* a serem utilizados pela nova máquina), verificação de sintaxe dessas opções e efetiva alteração do endereço IP e *gateway* no arquivo `/etc/network/interfaces`.

Feito isso, desligue a máquina `debian-template`.

8) Criação da VM de DNS secundário

Vamos agora criar a segunda máquina de nosso *datacenter* simulado, a máquina `ns2`. Ela atuará como um servidor DNS secundário sob o endereço IP 10.0.42.2/24, dentre outras funções que serão configuradas em sessões posteriores.

1. Clone a máquina `debian-template` seguindo os mesmos passos da atividade (6) da sessão 1. Para o nome da máquina, escolha `ns2`.
2. Ligue a máquina `ns2` e, após o *boot*, faça login como o usuário `root`. Usando o script `/root/scripts/changehost.sh`, efetue a configuração automática da máquina:

```
# ip addr show label 'enp0s*' | grep 'inet ' | awk '{print $2,$NF}' ; hostname ;
whoami
10.0.42.253/24 enp0s3
debian-template
root
```

```
# bash ~/scripts/changehost.sh -h ns2 -i 10.0.42.2 -g 10.0.42.1
```

```
# ip addr show label 'enp0s*' | grep 'inet ' | awk '{print $2,$NF}' ; hostname ;
whoami
10.0.42.2/24 enp0s3
ns2
root
```

9) Configuração do DNS secundário

A configuração do NSD como um servidor DNS secundário é bastante similar à do servidor primário, com algumas poucas diferenças na seção `pattern` do arquivo original. Já a configuração do Unbound, por outro lado, é absolutamente idêntica. Vamos ao trabalho:

1. Instale os pacotes relevantes:

   ```
   # apt-get install nsd unbound dnsutils
   ```

   Durante a pós-configuração do Unbound o sistema pode demorar um pouco a retornar para o *shell* — seja paciente, em alguns segundos a instalação será concluída.

2. Primeiro, o **NSD**. Queremos copiar o arquivo `/etc/nsd/nsd.conf` da máquina `ns1` e fazer as alterações necessárias — no entanto, o acesso SSH à essa máquina pela DMZ está proibido pelas configurações de firewall que fizemos no começo desta sessão. Assim sendo, vamos fazer a cópia no sentido oposto: da máquina `ns1` para a máquina `ns2`.

   Logue como `root` na máquina `ns1` e copie o arquivo usando o `scp`, como se segue:

   ```
   # hostname ; whoami
   ns1
   root
   ```

   ```
   # scp /etc/nsd/nsd.conf aluno@ns2:~
   The authenticity of host 'ns2 (10.0.42.2)' can't be established.
   ECDSA key fingerprint is SHA256:jxd7SPFgwNSsaMS7ApIEMpdAmxEnWeJ83s/K4h2XV5o.
   Are you sure you want to continue connecting (yes/no)? yes
   Warning: Permanently added 'ns2,10.0.42.2' (ECDSA) to the list of known hosts.
   aluno@ns2's password:
   nsd.conf
   100%  909     2.5MB/s   00:00
   ```

   Perfeito, o arquivo foi copiado para `/home/aluno/nsd.conf` na máquina `ns2` — note que não fizemos o login remoto com o usuário `root` pois esse acesso não é permitido pela política padrão do *daemon* `sshd` no Debian.

3. De volta à máquina `ns2` como o usuário `root`, pare o NSD e gere as chaves TLS de controle:

```
# hostname ; whoami
ns2
root
```

```
# systemctl stop nsd
```

```
# nsd-control-setup
setup in directory /etc/nsd
nsd_server.key exists
nsd_control.key exists
create nsd_server.pem (self signed certificate)
create nsd_control.pem (signed client certificate)
Signature ok
subject=CN = nsd-control
Getting CA Private Key
Setup success. Certificates created.
```

Vamos fazer o backup do arquivo de configuração original:

```
# mv /etc/nsd/nsd.conf /etc/nsd/nsd.conf.orig
```

Copie o arquivo `/home/aluno/nsd.conf` para a pasta `/etc/nsd` e ajuste suas permissões:

```
# cp /home/aluno/nsd.conf /etc/nsd
```

```
# chown root. /etc/nsd/nsd.conf
```

```
# ls -ld /etc/nsd/nsd.conf
-rw-r--r-- 1 root root 909 nov 12 12:14 /etc/nsd/nsd.conf
```

Excelente. Temos agora que editar o arquivo com as configurações do servidor secundário, o que faremos através do uso do `sed` nos comandos a seguir:

```
# sed -i 's/^\( *ip-address:\) 10\.0\.42\.1/\1 10\.0\.42\.2/' /etc/nsd/nsd.conf
```

```
# sed -i '/^ *zonesdir:/d' /etc/nsd/nsd.conf
```

```
# sed -i 's/\"inslave\"/\"inmaster\"/' /etc/nsd/nsd.conf
```

```
# sed -i 's/^\( *\)notify:[0-9@. ]*\(.*\)/\1allow-notify: 10\.0\.42\.1 \2/'
/etc/nsd/nsd.conf
```

```
# sed -i 's/^\( *\)provide-xfr:[0-9. ]*\(.*\)/\1request-xfr: AXFR 10\.0\.42\.1@8053
\2/' /etc/nsd/nsd.conf
```

O que esses comandos fizeram, afinal? O comando `diff` pode ser usado para comparar arquivos, evidenciando as diferenças entre eles:

```
# diff -u /home/aluno/nsd.conf /etc/nsd/nsd.conf
--- /home/aluno/nsd.conf        2018-11-12 20:51:45.040000000 -0200
+++ /etc/nsd/nsd.conf   2018-11-12 20:55:34.156000000 -0200
@@ -1,10 +1,9 @@
 server:
   ip-address: 127.0.0.1
-  ip-address: 10.0.42.1
+  ip-address: 10.0.42.2
   do-ip4: yes
   port: 8053
   username: nsd
-  zonesdir: "/etc/nsd/zones"

   logfile: "/var/log/nsd.log"
   pidfile: "/run/nsd/nsd.pid"
@@ -27,17 +26,17 @@
   secret: "i7IoB5VDHVOCW9wvuOGQuFLNu8hzfAblVAbCD1SbPL4="

 pattern:
-  name: "inslave"
-  notify: 10.0.42.2@8053 inkey
-  provide-xfr: 10.0.42.2 inkey
+  name: "inmaster"
+  allow-notify: 10.0.42.1 inkey
+  request-xfr: AXFR 10.0.42.1@8053 inkey

 zone:
   name: "intnet"
-  include-pattern: "inslave"
+  include-pattern: "inmaster"
   zonefile: "intnet.zone.signed"

 zone:
   name: "42.0.10.in-addr.arpa"
-  include-pattern: "inslave"
+  include-pattern: "inmaster"
   zonefile: "10.0.42.zone"
```

Não se esqueça de remover o arquivo `/home/aluno/nsd.conf`:

```
# rm /home/aluno/nsd.conf
```

4. Inicie o *daemon* do NSD usando o `nsd-control`:

```
# nsd-control start
```

Note que não foi necessário criar o diretório de zonas no servidor secundário, `/etc/nsd/zones`, já que as zonas transferidas ficam mantidas no diretório `/var/lib/nsd`:

```
# ls -1 /var/lib/nsd
nsd.db
xfrd.state
```

Vamos testar? Tente efetuar uma resolução direta de nomes no servidor local usando a ferramenta `dig`, também testando o funcionamento do DNSSEC:

```
# dig @127.0.0.1 -p 8053 ns1.intnet +dnssec +noadditional +noquestion +multiline
+noauthority

; <<>> DiG 9.10.3-P4-Debian <<>> @127.0.0.1 -p 8053 ns1.intnet +dnssec
+noadditional +noquestion +multiline +noauthority
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29660
;; flags: qr aa rd; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 3
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; ANSWER SECTION:
ns1.intnet.             86400 IN A 10.0.42.1
ns1.intnet.             86400 IN RRSIG A 10 2 86400 (
                              20181210223558 20181112223558 14936 intnet.
                              TdK1Ivm1Ykdk1MBaMQSqNHANV14FQbfdBCgiOznMc7wT
                              vqZPa2TBMTe2GYvcjbWd2WzH4o88pOAzwmCxMnnAxGJ4
                              7IglUx4GlQKTG/hx/5vxpFYgczJLAKpt/HEMfrInYVyP
                              qvCFpCyUudgA7kV8w05+BvBiK4IQAWZCiH8GcC3ERFoC
                              B/ZNE/f8YnMyi2sNqdN8Mhtkgz2vMZnACiLrifnV7hlI
                              3uvqn0fQetnDWLF/xSeHoAE4WMZ6KwMgutUT1H9lQwOg
                              Ci5PemGKnOn2Va4ARzvQeTzBdLHZlVi04X25NHIeKZTj
                              8Sx1vRnBwPfT8qlyI1oaOozuiLSyrII3Mw== )

;; Query time: 0 msec
;; SERVER: 127.0.0.1#8053(127.0.0.1)
;; WHEN: Mon Nov 12 20:58:05 -02 2018
;; MSG SIZE  rcvd: 985
```

Excelente. E quanto à resolução reversa?

```
# dig @127.0.0.1 -p 8053 -x 10.0.42.2 +noadditional +noquestion

; <<>> DiG 9.10.3-P4-Debian <<>> @127.0.0.1 -p 8053 -x 10.0.42.2 +noadditional
+noquestion
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44932
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; ANSWER SECTION:
2.42.0.10.in-addr.arpa. 86400   IN      PTR     ns2.intnet.

;; AUTHORITY SECTION:
42.0.10.in-addr.arpa.   86400   IN      NS      ns1.intnet.
42.0.10.in-addr.arpa.   86400   IN      NS      ns2.intnet.

;; Query time: 0 msec
;; SERVER: 127.0.0.1#8053(127.0.0.1)
;; WHEN: Mon Nov 12 20:58:17 -02 2018
;; MSG SIZE  rcvd: 107
```

5. Agora, vamos configurar o **Unbound**. Como feito anteriormente, logue como `root` na máquina `ns1` e copie desta vez o arquivo `/etc/unbound/unbound.conf` usando o `scp`, como se segue:

```
# hostname ; whoami
ns1
root
```

```
# scp /etc/unbound/unbound.conf aluno@ns2:~
aluno@ns2's password:
unbound.conf
100%  820     2.4MB/s   00:00
```

   O arquivo foi copiado para `/home/aluno/unbound.conf` na máquina `ns2`, como objetivado.

6. De volta a `ns2` como o usuário `root`, pare o *daemon* do Unbound e configure as chaves de controle:

```
# hostname ; whoami
ns2
root
```

```
# systemctl stop unbound
```

```
# unbound-control-setup
setup in directory /etc/unbound
unbound_server.key exists
unbound_control.key exists
create unbound_server.pem (self signed certificate)
create unbound_control.pem (signed client certificate)
Signature ok
subject=CN = unbound-control
Getting CA Private Key
Setup success. Certificates created.
```

Faça o backup do arquivo de configuração original:

```
# mv /etc/unbound/unbound.conf /etc/unbound/unbound.conf.orig
```

Copie o arquivo `/home/aluno/unbound.conf` para a pasta `/etc/unbound` e ajuste suas permissões:

```
# cp /home/aluno/unbound.conf /etc/unbound
```

```
# chown root. /etc/unbound/unbound.conf
```

```
# ls -ld /etc/unbound/unbound.conf
-rw-r--r-- 1 root root 820 nov 12 22:15 /etc/unbound/unbound.conf
```

Edite o arquivo com as configurações do servidor secundário via `sed`, como se segue:

```
# sed -i 's/^\( *interface: 10\.0\.42\.\)1/\12/' /etc/unbound/unbound.conf
```

```
# sed -i '/^ *interface: 192.*/d' /etc/unbound/unbound.conf
```

Use o comando `diff` para verificar as diferenças entre o arquivo original e as alterações feitas com o `sed`:

```
# diff -u /home/aluno/unbound.conf /etc/unbound/unbound.conf
--- /home/aluno/unbound.conf    2018-11-12 22:13:47.276000000 -0200
+++ /etc/unbound/unbound.conf   2018-11-12 22:24:48.564000000 -0200
@@ -1,7 +1,6 @@
 server:
   interface: 127.0.0.1
-  interface: 10.0.42.1
-  interface: 192.168.42.1
+  interface: 10.0.42.2
   port: 53

   access-control: 127.0.0.0/8 allow
```

Remova o arquivo de configuração do Unbound no diretório `/home/aluno`:

```
# rm /home/aluno/unbound.conf
```

7. Vamos proceder aos testes. Inicie o Unbound usando o comando `unbound-control`:

```
# unbound-control start
```

Teste uma resolução direta qualquer no servidor Unbound local, solicitando DNSSEC:

```
# dig @127.0.0.1 mx1.intnet +dnssec +multiline +noquestion +noadditional

; <<>> DiG 9.10.3-P4-Debian <<>> @127.0.0.1 mx1.intnet +dnssec +multiline
+noquestion +noadditional
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53007
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; ANSWER SECTION:
mx1.intnet.             86388 IN A 10.0.42.91
mx1.intnet.             86388 IN RRSIG A 10 2 86400 (
                                20181210223558 20181112223558 14936 intnet.
                                YPluG8EXq3WwlKIlMpeXc+C1C3AZhC+P1ej+6I5ax9Aw
                                xQ+Zp6lwMXz64e7nPN2mjNa1PcAVuUNc8gE77U5dohhI
                                Al++b4+1mPHLzN6EY4UuSsWqkE5NacSaUngmhV52Mrwj
                                cwXtmJgJm9vhSTLKnUSYhSwFWZlC2FxI96kTQhtzJuDd
                                KI7SVpYsNZPGiKwj5r9F8Wr/sKpCc0PgVnseewyPmNGY
                                5pJEV3Ox6XgzQk0qjatoO4IxSj3CqX6ghU+MykcU2L+n
                                56LfqqrV+R6TzzVst9/bFC7UgSXjF4v9eDKvPc86mcGn
                                J9giG3w+07D2+jv7Ap6RfQWnx1zBy5CWFA== )

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Mon Nov 12 22:45:11 -02 2018
;; MSG SIZE  rcvd: 349
```

Sessão 3: Autenticação centralizada

Retomando o cenário apresentado na introdução da primeira sessão, num ambiente em que a virtualização é usada em larga escala, teremos diversas máquinas virtuais operando cada qual com seus serviços alocados. Imagine, hipoteticamente, que temos centenas de VMs dentro do *datacenter*. Vários desafios podem surgir à mente, mas tente responder a seguinte pergunta: com relação à gestão de contas, o que fazer quando um novo colaborar é integrado à equipe? Ou, por outro lado, quando um funcionário é desligado da empresa?

Ora, se temos centenas de VMs, é fácil supor que teremos que logar nas diferentes máquinas que novo colaborador (ou o antigo) deverá acessar e criar uma conta de usuário para ele — além disso, adicioná-lo a grupos e editar permissões relevantes. Fazer esse procedimento inúmeras vezes é claramente um processo que pode gerar erros de configuração, então poderia-se pensar em automatizá-lo, digamos, via *shell scripts*. Uma boa solução, mas não a ideal neste caso.

Sistemas de autenticação centralizados, como NIS (*Network Information Service*) ou LDAP (*Lightweight Directory Access Protocol*) são excelentes ferramentas para facilitar a gestão em cenários como o apresentado — adicionando o usuário em um único ponto, é possível distribuir essa configuração para dezenas, ou centenas, de máquinas de forma instantânea. O gerenciamento de grupos no sistema centralizado também permite atribuir permissionamento de forma fácil, ou removê-lo quando necessário.

Nesta sessão, iremos configurar um sistema de autenticação centralizado para o nosso laboratório usando LDAP, no qual gerenciaremos usuários e grupos, e faremos a integração desse sistema de autenticação com o Linux através do PAM (*Pluggable Authentication Modules*). Em lugar de fazer o controle de senhas dos usuários diretamente via `/etc/shadow` ou no LDAP, criaremos um sistema de autoridade certificadora (*Certificate Authority*) para o SSH, com o qual os usuários farão login nos servidores usando chaves assimétricas assinadas pela CA. Finalmente, para controlar ataques de força-bruta contra os servidores, usaremos o programa Fail2Ban para realizar o bloqueio automático de atacantes no firewall de host das máquinas.

1) Topologia desta sessão

A figura abaixo mostra a topologia de rede que será utilizada nesta sessão, com as máquinas relevantes em destaque.

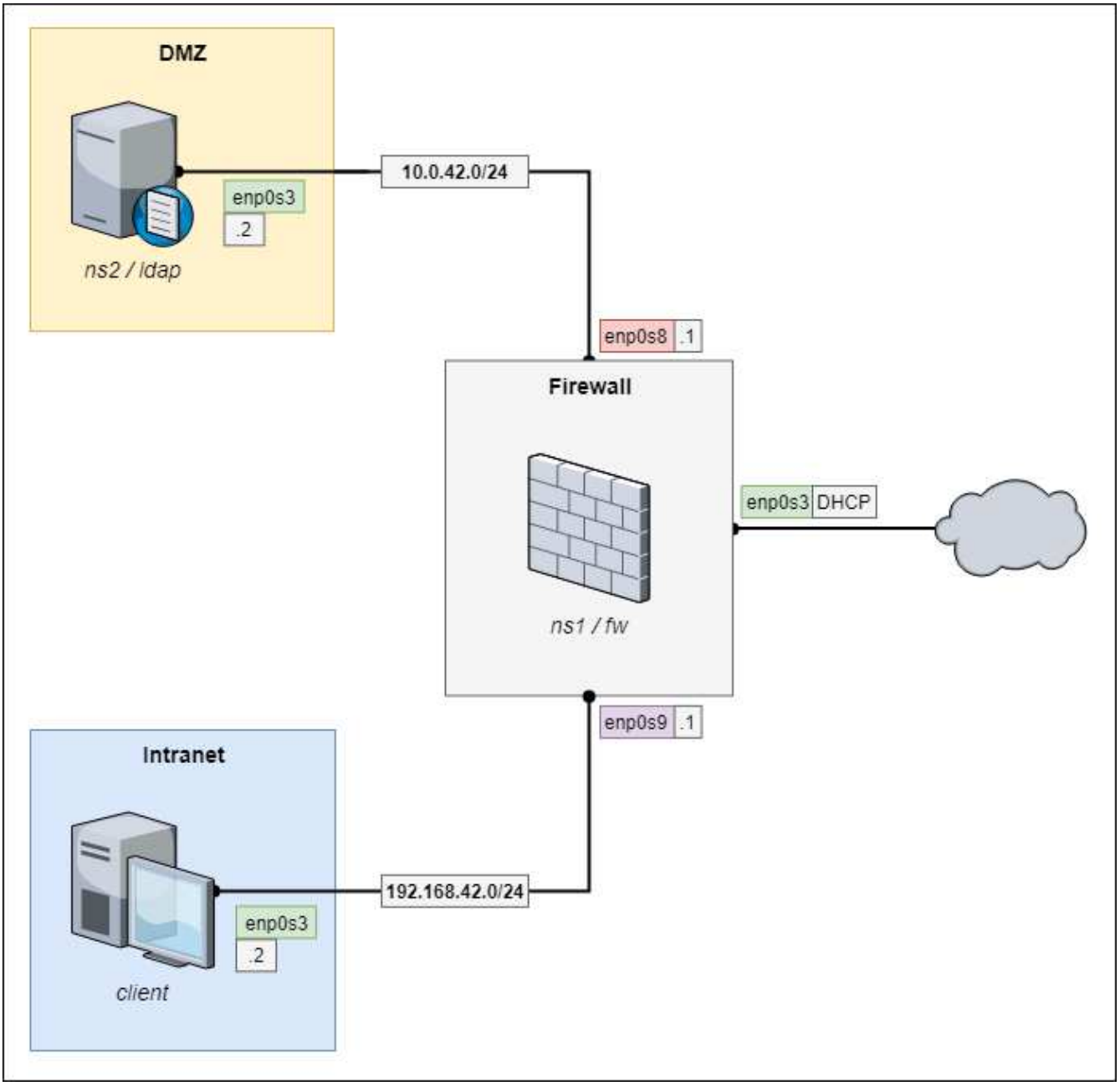

Figura 27. Topologia de rede desta sessão

Teremos três máquinas:

- `ns1`, com *alias* `firewall`, atuando como firewall de rede e DNS primário. Endereços IP via DHCP (interface *bridge*), 10.0.42.1/24 (interface DMZ) e 192.168.42.1/24 (interface Intranet).
- `ns2`, com *alias* `ns2`, localizada na DMZ e atuando como DNS secundário, servidor LDAP de autenticação centralizada e repositório de chaves SSH-CA. Endereço IP 10.0.42.2/24.
- `client`, localizada na Intranet e usada a partir de agora como ponto de partida para logins remotos nos diferentes servidores do *datacenter* simulado. Endereço IP 192.168.42.2/24.

1. Observe que a topologia acima prevê a criação de um novo *alias* de rede para a máquina `ns2`, o nome `ns2`. Antes de começar, vamos ajustar nossa configuração DNS para refletir essa situação: acesse a máquina `ns1` como o usuário `root`:

```
# hostname ; whoami
ns1
root
```

Edite o arquivo de zonas `/etc/nsd/zones/intnet.zone`, inserindo uma entrada CNAME para a máquina `ns2`, como se segue. **Não se esqueça** de incrementar o valor do serial no topo do arquivo!

```
# nano /etc/nsd/zones/intnet.zone
(...)
```

```
# grep '^ldap ' /etc/nsd/zones/intnet.zone
ldap    IN    CNAME             ns2
```

Assine o arquivo de zonas usando o *script* criado na sessão anterior:

```
# bash /root/scripts/signzone-intnet.sh
reconfig start, read /etc/nsd/nsd.conf
ok
ok
ok
ok removed 0 rrsets, 0 messages and 0 key entries
```

Verifique que a entrada foi criada com sucesso, usando o comando `dig`:

```
# dig @127.0.0.1 ldap.intnet +noall +answer

; <<>> DiG 9.10.3-P4-Debian <<>> @127.0.0.1 ldap.intnet +noall +answer
; (1 server found)
;; global options: +cmd
ldap.intnet.            86138   IN      CNAME   ns2.intnet.
ns2.intnet.             86138   IN      A       10.0.42.2
```

Certifique-se que a transferência de zonas entre o servidor DNS primário e secundário está funcionando corretamente — repita a consulta, desta vez no servidor `ns2`:

```
# dig @10.0.42.2 ldap.intnet +noquestion

; <<>> DiG 9.10.3-P4-Debian <<>> @10.0.42.2 ldap.intnet +noquestion
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 48391
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; ANSWER SECTION:
ldap.intnet.            86207   IN      CNAME   ns2.intnet.
ns2.intnet.             86207   IN      A       10.0.42.2

;; Query time: 0 msec
;; SERVER: 10.0.42.2#53(10.0.42.2)
;; WHEN: Tue Nov 13 18:28:16 -02 2018
;; MSG SIZE  rcvd: 74
```

2) Configuração do servidor LDAP

1. O servidor LDAP, como explicitado na topologia desta sessão, será a máquina `ns2`. Acesse-a como o usuário `root`:

   ```
   # hostname ; whoami
   ns2
   root
   ```

   Vamos, primeiramente, instalar o servidor LDAP através dos pacotes:

   ```
   # apt-get install slapd ldap-utils ldapscripts
   ```

   Durante a instalação, será solicitada uma senha administrativa para o LDAP, que iremos redefinir em breve. Informe `rnpesr`.

2. Agora, vamos reconfigurar o servidor LDAP para definir alguns parâmetros que não foram questionados durante a instalação via `apt-get`. Execute:

   ```
   # dpkg-reconfigure -plow slapd
   ```

   Informe as seguintes opções:

   Tabela 5. Configurações do pacote slapd

Pergunta	Opção
Omitir a configuração do servidor OpenLDAP?	Não
Nome de domínio DNS	intnet
Nome da organização	seg10
Senha do administrador	rnpesr (repetir)
Backend da base de dados a ser usado	MDB
Você deseja que a base de dados seja removida quando o pacote slapd for expurgado ("purged")	Não
Move a base de dados antiga	Sim

Durante a configuração, será criada uma base LDAP (semi) vazia, apenas com a raiz `dc=intnet` e o usuário administrativo `cn=admin,dc=intnet`. Iremos popular esta base brevemente.

3. Agora, vamos instalar o `nslcd`, um *daemon* LDAP local para resolução de diretivas de autenticação. Execute:

```
# apt-get install nslcd
```

Durante a instalação do pacote, informe as seguintes opções:

Tabela 6. Configurações do pacote nslcd

Pergunta	Opção
URI do servidor LDAP	ldapi:///
Base de buscas do servidor LDAP	dc=intnet
Serviços de nome para configurar	passwd, group, shadow

Note que informamos que a localização do servidor LDAP é local, já que ele está instalado na máquina corrente. Além disso, iremos configurar as bases de contas (`passwd`), grupos (`group`) e senhas (`shadow`) junto ao LDAP.

4. Vamos configurar as opções padrão dos binários de linha de comando do `ldap-utils` (como os comandos `ldapsearch` e `ldapadd`, por exemplo). Edite manualmente ou use o `sed` para editar as linhas apropriadas do arquivo `/etc/ldap/ldap.conf`:

```
# sed -i 's/^#\(BASE\).*/\1 dc=intnet/' /etc/ldap/ldap.conf
```

```
# sed -i 's/^#\(URI\).*/\1 ldapi\:\/\/\//' /etc/ldap/ldap.conf
```

Verifique que os valores editados estão corretos:

```
# grep -v '^#' /etc/ldap/ldap.conf | sed '/^$/d'
BASE dc=intnet
URI ldapi:///
TLS_CACERT      /etc/ssl/certs/ca-certificates.crt
```

5. A seguir, vamos configurar o `ldapscripts`, um conjunto de ferramentas auxiliares que facilitam enormemente a configuração e uso de bases LDAP via linha de comando. Primeiro, edite manualmente ou use o `sed` para ajustar o arquivo `/etc/ldapscripts/ldapscripts.conf`, informando o *bind DN* do usuário administrativo na base LDAP, como se segue:

   ```
   # sed -i 's/^\(BINDDN=\).*/\1\"cn=admin,dc=intnet\"/'
   /etc/ldapscripts/ldapscripts.conf
   ```

   Depois, informe a senha do usuário administrativo no arquivo `/etc/ldapscripts/ldapscripts.passwd`. Execute:

   ```
   # echo -n "rnpesr" > /etc/ldapscripts/ldapscripts.passwd
   ```

   Como a senha do usuário administrativo do LDAP está embutida em texto claro nesse arquivo, é fundamental garantir que suas permissões estão suficientemente estritas. Verifique:

   ```
   # ls -ld /etc/ldapscripts/ldapscripts.passwd
   -rw-r----- 1 root root 6 out 29 09:16 /etc/ldapscripts/ldapscripts.passwd
   ```

6. Vamos inicializar a base LDAP usando o `ldapscripts`. Execute:

   ```
   # ldapinit -s
   ```

7. Será que funcionou? Consulte a base LDAP sem autenticação, e liste seu conteúdo:

```
# ldapsearch -x -LLL
dn: dc=intnet
objectClass: top
objectClass: dcObject
objectClass: organization
o: seg10
dc: intnet

dn: cn=admin,dc=intnet
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator

dn: ou=People,dc=intnet
objectClass: top
objectClass: organizationalUnit
ou: People

dn: ou=Groups,dc=intnet
objectClass: top
objectClass: organizationalUnit
ou: Groups

dn: ou=Hosts,dc=intnet
objectClass: top
objectClass: organizationalUnit
ou: Hosts

dn: ou=Idmap,dc=intnet
objectClass: organizationalUnit
ou: Idmap
```

Perfeito! Temos configurada a raiz `dc=intnet` e usuário administrativo `cn=admin,dc=intnet` como anteriormente, e o comando `ldapinit` se encarregou de criar DNs para armazenar usuários, grupos, *hosts* e mapeamentos de identidade na base LDAP. Podemos prosseguir.

3) Habilitando logs do LDAP

Por padrão, o *daemon* `slapd` envia logs para a *facility* `local4` do sistema. Porém, após a instalação via `apt-get`, seu *log level* (nível de criticidade dos eventos registrados) é configurado para `none` — ou seja, nenhum evento é registrado. Evidentemente, essa situação não é interessante ao configurar o servidor, pois será muito difícil identificar as fontes de problemas se não tivermos nenhum log para consultar. Vamos corrigir isso.

1. Primeiro, crie um diretório específico para o armazenamento de LDIFs. LDIFs são uma sigla para LDAP *Data Interchange Format*, arquivos de texto plano que podem ser usados para representar informações em uma base LDAP. Algo equivalente a arquivos *dump* de bases SQL,

em comparação livre.

```
# mkdir /root/ldif
```

2. Crie neste diretório um LDIF de nome `/root/ldif/slapdlog.ldif`, com o seguinte conteúdo:

```
1 dn: cn=config
2 changeType: modify
3 replace: olcLogLevel
4 olcLogLevel: stats
```

O LDIF acima irá alterar o valor do parâmetro `olcLogLevel`, que define o *log level* do `slapd`, para `stats`. Para aplicar essa configuração, execute:

```
# ldapmodify -Y external -H ldapi:/// -f /root/ldif/slapdlog.ldif
```

3. O próximo passo é informar ao `rsyslog` que as mensagens enviadas para a *facility* `local4` devem ser enviadas para um arquivo específico. Crie o arquivo novo `/etc/rsyslog.d/slapd.conf` com o seguinte conteúdo:

```
1 $template slapdtmpl,"[%$DAY%-%$MONTH%-%$YEAR% %timegenerated:12:19:date-rfc3339%]
%app-name% %syslogseverity-text% %msg%\n"
2 local4.*    /var/log/slapd.log;slapdtmpl
```

4. Em seguida, reinicie ambos os *daemons* do `rsyslog` e do `slapd`:

```
# systemctl restart rsyslog.service
```

```
# systemctl restart slapd.service
```

5. Verifique que os registros de eventos do `slapd` estão sendo, de fato, enviados para o arquivo `/var/log/slapd.log`:

```
tail /var/log/slapd.log
[13-11-2018 18:35:18] slapd debug  daemon: shutdown requested and initiated.
[13-11-2018 18:35:18] slapd debug  slapd shutdown: waiting for 0 operations/tasks
to finish
[13-11-2018 18:35:18] slapd debug  slapd stopped.
[13-11-2018 18:35:18] slapd debug  @(#) $OpenLDAP: slapd  (May 23 2018 04:25:19)
$#012#011Debian OpenLDAP Maintainers <pkg-openldap-devel@lists.alioth.debian.org>
[13-11-2018 18:35:18] slapd debug  slapd starting
```

6. A rotação de logs deve ser habilitada, caso contrário os arquivos de log poderão ficar excessivamente grandes. Crie o arquivo novo `/etc/logrotate.d/slapd`, com o seguinte conteúdo:

```
/var/log/slapd.log {
  missingok
  notifempty
  compress
  daily
  rotate 30
  sharedscripts
  postrotate
    systemctl restart rsyslog.service
  endscript
}
```

Como o `logrotate` é invocado via `cron`, não é necessário reiniciar nenhum serviço neste caso.

4) Edição de índices e permissões no LDAP

1. Para maior performance durante as consultas ao LDAP, é recomendável aumentar os parâmetros de indexação de alguns atributos. Crie o arquivo `/root/ldif/olcDbIndex.ldif`, com o seguinte conteúdo:

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
replace: olcDbIndex
olcDbIndex: objectClass eq
olcDbIndex: cn pres,sub,eq
olcDbIndex: sn pres,sub,eq
olcDbIndex: uid pres,sub,eq
olcDbIndex: displayName pres,sub,eq
olcDbIndex: default sub
olcDbIndex: uidNumber eq
olcDbIndex: gidNumber eq
olcDbIndex: mail,givenName eq,subinitial
olcDbIndex: dc eq
```

O LDIF acima irá alterar o valor do parâmetro `olcDbIndex`, que define quais parâmetros serão indexados pelo `slapd` e quais tipos de busca serão suportados. Para aplicar essa configuração, execute:

```
# ldapmodify -Y EXTERNAL -H ldapi:/// -f /root/ldif/olcDbIndex.ldif
```

2. É interessante permitir que usuários possam alterar seus parâmetros `loginShell` e entrada `gecos` (informações de conta do usuário) via comandos `chsh` e `chfn`. Para fazer isso, crie o arquivo novo `/root/ldif/olcAccess.ldif` com o seguinte conteúdo:

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
add: olcAccess
olcAccess: {1}to attrs=loginShell,gecos
  by dn="cn=admin,dc=intnet" write
  by self write
  by * read
```

Para aplicar a configuração, execute:

```
# ldapmodify -Y EXTERNAL -H ldapi:/// -f /root/ldif/olcAccess.ldif
```

5) Adição de grupos e usuários no LDAP

Agora sim, vamos começar a criar usuários e grupos em nossa base LDAP. Imagine que iremos criar um grupo de nome `sysadm`, no qual estarão todos os administradores de sistema que têm permissão para acessar servidores não-críticos. Nesse grupo, iremos criar o usuário `luke`, com senha `seg10luke`. Como proceder?

1. Primeiro, crie o grupo usando o comando `ldapaddgroup`:

   ```
   # ldapaddgroup sysadm
   Successfully added group sysadm to LDAP
   ```

   Verifique que o grupo foi corretamente criado usando o `ldapsearch`:

   ```
   # ldapsearch -x -LLL 'cn=sysadm'
   dn: cn=sysadm,ou=Groups,dc=intnet
   objectClass: posixGroup
   cn: sysadm
   gidNumber: 10000
   description: Group account
   ```

2. A seguir, crie o usuário `luke`, informando seu grupo primário como `sysadm`:

   ```
   # ldapadduser luke sysadm
   Successfully added user luke to LDAP
   Successfully set password for user luke
   ```

   Novamente, consulte o `ldapsearch` para checar se o usuário foi criado com sucesso:

```
# ldapsearch -x -LLL 'cn=luke'
dn: uid=luke,ou=People,dc=intnet
objectClass: account
objectClass: posixAccount
cn: luke
uid: luke
uidNumber: 10000
gidNumber: 10000
homeDirectory: /home/luke
loginShell: /bin/bash
gecos: luke
description: User account
```

O comando `ldapid` também é uma boa opção neste cenário, fornecendo saída bastante similar à do comando `id`:

```
# ldapid luke
uid=10000(luke) gid=10000(sysadm) groups=10000(sysadm),10000(sysadm)
```

3. Vamos configurar a senha do usuário `luke` como `seg10luke`:

```
# ldapsetpasswd luke
Changing password for user uid=luke,ou=People,dc=intnet
New Password:
Retype New Password:
Successfully set password for user uid=luke,ou=People,dc=intnet
```

Para verificar, cheque que o campo `userPassword` do usuário está preenchido com um *hash* de senha. Para o comando `ldapsearch` funcionar, temos que informar um *bind DN* administrativo e senha correspondente, já que este atributo não é legível sem autenticação:

```
# ldapsearch -x -LLL -D 'cn=admin,dc=intnet' -W 'cn=luke' userPassword
Enter LDAP Password:
dn: uid=luke,ou=People,dc=intnet
userPassword:: e1NTSEF9NHdUSWZRcUhGR0o5VU5jNS9tVnhoaGJzNFVvNkFzMmE=
```

O comando `ldapfinger` também é uma boa opção para consultar as informações do usuário (e seu *hash* de senha), já que faz parte do `ldapscripts` e utiliza as credenciais administrativas que configuramos anteriormente:

```
# ldapfinger luke
dn: uid=luke,ou=People,dc=intnet
objectClass: account
objectClass: posixAccount
cn: luke
uid: luke
uidNumber: 10000
gidNumber: 10000
homeDirectory: /home/luke
loginShell: /bin/bash
gecos: luke
description: User account
userPassword:: e1NTSEF9NEU0aFI2N3lCNGplUHdXRHNHVEZYZTBEYm5YdGxDTUg=
```

4. Apesar de termos informado o grupo `sysadm` como grupo primário do usuário `luke`, do ponto de vista do grupo esse usuário ainda não o integra. O comando `ldapaddusertogroup` faz esse trabalho:

   ```
   # ldapaddusertogroup luke sysadm
   Successfully added user luke to group cn=sysadm,ou=Groups,dc=intnet
   ```

   Para verificar o pertencimento, use o comando `ldapsearch`, consultando os atributos `memberUid`:

   ```
   # ldapsearch -x -LLL 'cn=sysadm'
   dn: cn=sysadm,ou=Groups,dc=intnet
   objectClass: posixGroup
   cn: sysadm
   gidNumber: 10000
   description: Group account
   memberUid: luke
   ```

   O `ldapgid`, parte da suíte `ldapscripts`, também é uma alternativa interessante:

   ```
   # ldapgid sysadm
   gid=10000(sysadm) users(primary)=10000(luke) users(secondary)=10000(luke)
   ```

5. Como já mencionado algumas vezes, o `ldapscripts` oferece um conjunto de programas que facilitam bastante as tarefas corriqueiras de manipulação da base LDAP via linha de comando. Apesar de termos trabalhado um bom número desses programas, listamos abaixo alguns outros que não foram utilizados. Consulte suas páginas de manual para mais informações sobre como operá-los:

   - Deleção de entradas:
     - `/usr/sbin/ldapdeletegroup`
     - `/usr/sbin/ldapdeleteuser`

    - `/usr/sbin/ldapdeleteuserfromgroup`
  - Modificação de entradas:
    - `/usr/sbin/ldapmodifygroup`
    - `/usr/sbin/ldapmodifyuser`
  - Renomear entradas:
    - `/usr/sbin/ldaprenamegroup`
    - `/usr/sbin/ldaprenameuser`
  - Configurar grupo primário de um usuário:
    - `/usr/sbin/ldapsetprimarygroup`

6) Integração e teste do sistema de autenticação com LDAP

Agora, vamos integrar o sistema de autenticação do Linux com a base LDAP que configuramos anteriormente usando o PAM (*Pluggable Authentication Modules*). Felizmente, muito do trabalho de configuração já foi feito automaticamente pelos *scripts* de instalação do `apt-get` quando instalamos o pacote `nslcd`. Faltam apenas alguns poucos passos.

1. Quando um usuário do LDAP efetua login em uma máquina pela primeira vez, seu diretório *home* não existe, naturalmente. Vamos configurar o PAM para criar esse diretório automaticamente. Crie o arquivo novo `/usr/share/pam-configs/mkhomedir` com o seguinte conteúdo:

   ```
   Name: Create home directory during login
   Default: yes
   Priority: 900
   Session-Type: Additional
   Session:
           required        pam_mkhomedir.so umask=0022 skel=/etc/skel
   ```

   Em seguida, execute:

   ```
   # pam-auth-update
   ```

   Durante a configuração do PAM, na pergunta "Perfis PAM para habilitar", mantenha todas as caixas marcadas e selecione OK.

   Verifique que a configuração surtiu efeito pesquisando pelo termo `mkhomedir` nos arquivos de configuração do PAM, em `/etc/pam.d`:

```
# grep -ri mkhomedir /etc/pam.d
/etc/pam.d/common-session:session       required        pam_mkhomedir.so umask=0022
skel=/etc/skel
/etc/pam.d/common-session-noninteractive:session        required
pam_mkhomedir.so umask=0022 skel=/etc/skel
```

2. Reinicie os *daemons* `nslcd` e `nscd` para que a *cache* de usuários, grupos e senhas do LDAP seja atualizada:

```
# systemctl restart nslcd.service
```

```
# systemctl restart nscd.service
```

3. Será que funcionou? Pesquise a lista de usuários do sistema e busque pelo usuário recém-criado `luke`:

```
# getent passwd | grep luke
luke:*:10000:10000:luke:/home/luke:/bin/bash
```

Excelente! E quando ao grupo `sysadm`?

```
# getent group | grep sysadm
sysadm:*:10000:luke
```

Finalmente, consulte se o usuário `luke` é reconhecido como membro de `sysadm` pelo sistema:

```
# groups luke
luke : sysadm
```

4. Vamos testar: tente logar via `ssh` na máquina local usando o usuário `luke`:

```
# ssh luke@localhost
The authenticity of host 'localhost (::1)' can't be established.
ECDSA key fingerprint is SHA256:mI2LM9Nxt5ltC7/VnlEgB+JBdqFbKxUj1FhhvLijcv4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
luke@localhost's password:
Creating directory '/home/luke'.
```

Perfeito. Note que o diretório `/home/luke` foi criado automaticamente, como esperado. Faça as verificações pós-login de costume:

```
$ whoami
luke
```

```
$ pwd
/home/luke
```

```
$ id
uid=10000(luke) gid=10000(sysadm) grupos=10000(sysadm)
```

5. Falta testar se o usuário consegue alterar sua senha diretamente via console, sem necessidade de edição direta à base LDAP. Primeiro, verifique o *hash* da senha atual — note que iremos usar o próprio DN do usuário `luke` como *bind DN* durante a conexão LDAP, motivo pelo qual deve-se informar a senha desse usuário, e não do administrador:

   ```
   $ ldapsearch -x -LLL -D 'uid=luke,ou=People,dc=intnet' -W 'uid=luke' userPassword
   Enter LDAP Password:
   dn: uid=luke,ou=People,dc=intnet
   userPassword:: e1NTSEF9K29BcE55S3AwRU9XM0sreWVPeFNoZUJjdFhBbVJyVEg=
   ```

   Use o comando `passwd` para alterar a senha para um outro valor qualquer:

   ```
   $ passwd
   (current) LDAP Password:
   Nova senha:
   Redigite a nova senha:
   passwd: senha atualizada com sucesso
   ```

   Verifique novamente o *hash* da senha do usuário `luke` e compare os dois valores:

   ```
   $ ldapsearch -x -LLL -D 'uid=luke,ou=People,dc=intnet' -W 'uid=luke' userPassword
   Enter LDAP Password:
   dn: uid=luke,ou=People,dc=intnet
   userPassword:: e1NTSEF9b0REb2lVbWgvR0swMm9qaGJoZWJ2ZGtNYzFKTE1kazk=
   ```

   Perfeito, os *hashes* são diferentes; ou seja, a alteração de senha via `passwd` funcionou normalmente. Retorne a senha do usuário `luke` ao valor anterior, para evitar confusões no futuro.

7) Configurando uma autoridade certificadora (CA) para o SSH

Até o momento, instalamos e configuramos uma base LDAP, e testamos sua integração com os sistemas de autenticação do sistema usando o `ssh`. Porém, a todo momento, tivemos que digitar as senhas dos usuários para nos autenticar — será que podemos fazer isso de uma forma mais segura?

Nesta atividade iremos configurar uma autoridade certificadora para o `ssh`, com a qual assinaremos pares de chaves de *hosts* e de usuários. De posse dessas chaves, não será mais necessário informar senhas durante o login, tornando o processo significativamente mais seguro (desde que se tome cuidado para não perder a chave privada, como veremos).

1. Primeiro, verifique que você está logado como usuário `root` na máquina `ns2`:

   ```
   # hostname ; whoami
   ns2
   root
   ```

2. Vamos adicionar um novo usuário à base LDAP, `sshca`, que será responsável por realizar os processos de assinaturas de chaves de *hosts* e usuários. Esse usuário irá pertencer ao grupo `setup`, um grupo especial para atividades de configuração de sistemas. Sua senha será `seg10sshca`.

   Vamos por partes. Primeiro, crie o grupo:

   ```
   # ldapaddgroup setup
   Successfully added group setup to LDAP
   ```

   Em seguida, o usuário:

   ```
   # ldapadduser sshca setup
   Successfully added user sshca to LDAP
   Successfully set password for user sshca
   ```

   Adicione o usuário ao grupo:

   ```
   # ldapaddusertogroup sshca setup
   Successfully added user sshca to group cn=setup,ou=Groups,dc=intnet
   ```

   E, finalmente, configure a senha do usuário `sshca`:

```
# ldapsetpasswd sshca
Changing password for user uid=sshca,ou=People,dc=intnet
New Password:
Retype New Password:
Successfully set password for user uid=sshca,ou=People,dc=intnet
```

3. Faça login com o usuário recém-criado no sistema local:

```
# ssh sshca@localhost
sshca@localhost's password:
Creating directory '/home/sshca'.
```

```
$ whoami
sshca
```

```
$ pwd
/home/sshca
```

4. Vamos criar dois pares de chaves para a CA (*Certificate Authority*, ou autoridade certificadora) do `ssh`: uma para assinar chaves de *hosts*, denominada `server_ca`, e outra para assinar chaves de usuários, denominada `user_ca`. Iremos criar chaves RSA de 4096 bits, e devemos escolher uma senha bastante segura para a chave privada — já que, com ela, pode-se assinar quaisquer chaves `ssh` que autorizarão máquinas a se passarem por membros do nosso *datacenter* e usuários a logarem em qualquer servidor integrado.

   Como estamos em um ambiente de laboratório, não iremos utilizar senhas para as chaves privadas de modo a agilizar a execução das atividades.

   Para criar a chave de assinatura de *hosts*, `server_ca`, execute:

```
$ ssh-keygen -f server_ca -t rsa -b 4096 -N ''
Generating public/private rsa key pair.
Your identification has been saved in server_ca.
Your public key has been saved in server_ca.pub.
The key fingerprint is:
SHA256:op2g5J5DN/1pPy8RUHxnAdh51NHP/2Pmyo3Vq5Qlvjo sshca@ns2
The key's randomart image is:
+---[RSA 4096]----+
|         o.o.+o+o|
|        . o + + o|
|         . . + ..|
|          .     o|
|  . ... S  . . ..|
| o..o+.o  . . + o|
| .o...o. . . + .o|
| ...    + oEo =+o|
|  o.   . ..=+**+.|
+----[SHA256]-----+
```

Verifique que o par de chaves foi criado com sucesso:

```
$ ls
server_ca  server_ca.pub
```

Para criar a chave de assinatura de usuários, `user_ca`, execute:

```
$ ssh-keygen -f user_ca -t rsa -b 4096 -N ''
Generating public/private rsa key pair.
Your identification has been saved in user_ca.
Your public key has been saved in user_ca.pub.
The key fingerprint is:
SHA256:T6dFFMQiJnbGCLg0rj1tzaT4mfqagBn80YQaBGuWpuM sshca@ns2
The key's randomart image is:
+---[RSA 4096]----+
|o. ... o   o+.   |
|..= . + * ...    |
|.O + o = . ..    |
|* = o .    .     |
|+= + *  S . o    |
|++= = o  o +     |
|+E = o    o      |
| . .+            |
|  ++.            |
+----[SHA256]-----+
```

Cheque que todos os quatro arquivos de chaves pública/privada foram criados:

```
$ ls
server_ca  server_ca.pub  user_ca  user_ca.pub
```

5. Dada a grande sensibilidade das chaves privadas das CAs nesse sistema de autenticação que estamos configurando, é fundamental garantir também que, além de terem uma senha forte configurada, suas permissões estejam corretamente ajustadas.

   Verifique as permissões das chaves usando o comando `ls`:

```
$ ls -l *_ca*
-rw------- 1 sshca setup 1766 out 28 08:39 server_ca
-rw-r--r-- 1 sshca setup  392 out 28 08:39 server_ca.pub
-rw------- 1 sshca setup 1766 out 28 08:40 user_ca
-rw-r--r-- 1 sshca setup  392 out 28 08:40 user_ca.pub
```

8) Configurando a SSH-CA no servidor LDAP

1. Vamos configurar o servidor LDAP para interoperar com a CA `ssh` que configuramos na atividade anterior. Volte ao usuário `root` na máquina `ns2`:

```
# hostname ; whoami
ns2
root
```

   Crie um novo arquivo, `/root/scripts/sshsign.sh` com o seguinte conteúdo:

```
#!/bin/bash

CA_USER="sshca"
CA_ADDR="10.0.42.2"
CA_USER_PASS="seg10sshca"
SSH_OPTS="-o PreferredAuthentications=password -o PubkeyAuthentication=no"


function cleanup() {
  sed -i '/^HostCertificate/d' /etc/ssh/sshd_config
  sed -i '/^TrustedUserCAKeys/d' /etc/ssh/sshd_config

  rm -f ~/.ssh/known_hosts       \
    /etc/ssh/ssh_host_*      \
    /etc/ssh/ssh_known_hosts \
    /etc/ssh/user_ca.pub 2> /dev/null
  dpkg-reconfigure openssh-server &> /dev/null

  systemctl restart sshd.service
```

```
}


# resetar sistema para estado conhecido
cleanup

# escanear chave do host ssh-ca
[ -d ~/.ssh ] || { mkdir ~/.ssh; chmod 700 ~/.ssh; }
if ! ssh-keygen -F ${CA_ADDR} 2>/dev/null 1>/dev/null; then
  ssh-keyscan -t rsa -T 10 ${CA_ADDR} 2> /dev/null >> ~/.ssh/known_hosts
fi

# iterar em todas as pubkeys SSH
for pkeypath in /etc/ssh/ssh_host_*.pub; do
  pkeyname="$( echo "${pkeypath}" | awk -F'/' '{print $NF}' )"
  certname="$( echo "${pkeyname}" | sed 's/\(\.pub$\)/-cert\1 /' )"
  echo "Signing ${pkeyname} key..."

  # copiar pubkey
  sshpass -p "${CA_USER_PASS}" \
    scp ${SSH_OPTS} ${pkeypath} ${CA_USER}@${CA_ADDR}:~

  # assinar pubkey, validade [-5 min -> 3 anos]
  identity="$(hostname --fqdn)"
  principals="$(hostname),$(hostname --fqdn),$(hostname -I | tr ' ' ',' | sed
's/,$//')"
  sshpass -p "${CA_USER_PASS}" \
    ssh ${SSH_OPTS} ${CA_USER}@${CA_ADDR} \
      ssh-keygen -s server_ca \
      -I "${identity}" \
      -n "${principals}" \
      -V -5m:+1095d \
      -h \
      ${pkeyname} 2> /dev/null

  # copiar pubkey assinada de volta
  sshpass -p "${CA_USER_PASS}" \
    scp ${SSH_OPTS} ${CA_USER}@${CA_ADDR}:${certname} /etc/ssh/

  # remover temporarios do diretorio remoto
  sshpass -p "${CA_USER_PASS}" \
    ssh ${SSH_OPTS} ${CA_USER}@${CA_ADDR} \
      rm ${pkeyname} ${certname}

  # remover pubkey RSA antiga e configurar ssh para apresentar pubkey assinada
  rm -f ${pkeypath} 2> /dev/null
  echo "HostCertificate /etc/ssh/${certname}" >> /etc/ssh/sshd_config
done

# copiar pubkey da server_ca e configurar reconhecimento de chaves de host
assinadas
```

```
69 echo "Configuring host key trust..."
70 echo "@cert-authority * $(sshpass -p "${CA_USER_PASS}" ssh ${SSH_OPTS} ${
CA_USER}@${CA_ADDR} cat server_ca.pub)" > /etc/ssh/ssh_known_hosts
71
72 # copiar pubkey da user_ca e configurar reconhecimento de chaves de usuario
assinadas
73 echo "Configuring user key trust..."
74 sshpass -p "${CA_USER_PASS}" \
75   scp ${SSH_OPTS} ${CA_USER}@${CA_ADDR}:~/user_ca.pub /etc/ssh/
76 echo "TrustedUserCAKeys /etc/ssh/user_ca.pub" >> /etc/ssh/sshd_config
77
78 echo "All done!"
79 systemctl restart sshd.service
```

Você deve estar se perguntando: o que esse *script* faz? Vamos ver:

1. (Linhas 3-6) Definimos o usuário `sshca` e o IP `10.0.42.2` (o servidor `ns2` local no qual estamos logados no momento) como a origem das chaves da CA que utilizaremos a seguir. Configuramos a senha do usuário `sshca` de forma *hardcoded* para agilizar o processo de execução do *script* e informamos que os logins SSH serão feitos sempre via senha, não chaves assimétricas.
2. (Linhas 9-20) Função que reseta a situação do SSH no servidor para um estado conhecido, em caso de falha ou encerramento abrupto do usuário durante a execução do *script*.
3. (Linhas 27-30) Escaneamos a chave do servidor `10.0.42.2` e armazenamos no arquivo `~/.ssh/known_hosts`, evitando que o usuário tenha que confirmar que confia no *host* remoto antes de logar.
4. (Linhas 33-35) Iteramos sobre todas as chaves públicas no diretório `/etc/ssh` (RSA, ECDSA e ED25519), extraindo *strings* para usar à frente.
5. (Linhas 39-40) Copiamos a chave pública do *host* sendo processadao pelo *loop* para o servidor da CA.
6. (Linhas 43-52) Assinamos a chave pública do *host* sendo processadao pelo *loop* usando a chave `server_ca`, com validade de 3 anos.
7. (Linhas 55-56) Copiamos a chave pública assinada sendo processadao pelo *loop* de volta para a máquina local.
8. (Linhas 59-61) Removemos chaves pública não-assinada e assinada sendo processadao pelo *loop* da pasta do usuário `sshca` no servidor `10.0.42.2`, para evitar confusão em assinaturas futuras.
9. (Linhas 64-65) Removemos a chave pública sendo processadao pelo *loop* não-assinada e mantemos apenas a assinada, configurando o servidor `ssh` para apresentá-la para clientes.
10. (Linha 70) Configuramos a chave `server_ca.pub` como uma CA confiável para *hosts* remotos; a partir deste momento, quaisquer logins de cliente `ssh` da máquina local para *hosts* assinados não terão que confirmar relação de confiança antes de prosseguir.
11. (Linhas 74-76) Copiamos a chave `user_ca.pub` e a configuramos como uma CA confiável para usuários; a partir deste momento, qualquer usuário que apresente uma chave assinada pela CA terá seu login autorizado sem necessitar digitação de senha.

12. (Linha 79) Reiniciamos o servidor `ssh` para aplicar as configurações realizadas.

2. Vamos instalar o `sshpass`, dependência para que o *script* acima funcione corretamente.

```
# apt-get install sshpass
```

3. Vamos configurar a máquina `ns2` para operar com a CA do `ssh`: execute o *script* criado no passo (1).

```
# bash ~/scripts/sshsign.sh
Signing ssh_host_ecdsa_key.pub key...
Signing ssh_host_ed25519_key.pub key...
Signing ssh_host_rsa_key.pub key...
Configuring host key trust...
Configuring user key trust...
All done!
```

Note que não foi necessário passar quaisquer parâmetros de linha de comando para o *script*. Ele autodetecta o *hostname* e endereços da máquina local e os configura como *principals* (conjunto de endereços/*hostnames* válidos para uma determinada chave; linha 44 do *script*).

4. Vamos verificar que o *script* fez seu trabalho corretamente. Cheque as linhas finais do arquivo `/etc/ssh/sshd_config`:

```
# tail -n4 /etc/ssh/sshd_config
HostCertificate /etc/ssh/ssh_host_ecdsa_key-cert.pub
HostCertificate /etc/ssh/ssh_host_ed25519_key-cert.pub
HostCertificate /etc/ssh/ssh_host_rsa_key-cert.pub
TrustedUserCAKeys /etc/ssh/user_ca.pub
```

Verifique que apenas chaves públicas assinadas existem no diretório `/etc/ssh`:

```
# ls -1 /etc/ssh/ssh_host_*.pub
/etc/ssh/ssh_host_ecdsa_key-cert.pub
/etc/ssh/ssh_host_ed25519_key-cert.pub
/etc/ssh/ssh_host_rsa_key-cert.pub
```

Compare o conteúdo dos arquivos de chave pública `/home/sshca/server_ca.pub` e de chave da CA confiável `/etc/ssh/ssh_known_hosts` — eles devem ser iguais:

```
# cat /etc/ssh/ssh_known_hosts
@cert-authority * ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQC+ZP47AOiRPcakZiLm+szvxWF5HKNa+BauaYXt5A6XjuUUJIYXXdT
dzrq+d4rx379jxk5E+RHvPls1Y+5Cvn2/EpBgx20TXoT9+O0Nn7pemHd7qkDb2JqPzoHzjViG033L8UociI
tit13UVMlvB2+miW4RPMHhLjJpJT2BoWzvkBtJdukO/SA+3EYpIZAj8kyFfkp9+2A/A+OCREWoh5EkcjREA
QRay+pc/j3sWDYrymu65OamPwGAe3Ih9/TmflUiwdHaNfJMY0BqpJsdFyUJeS36asjRhLtZltfLNTyY1gOq
3XglLZb76YaQWOoIhzwqQ2WNSt72cY7s8p97loX5LTBSjNw6Kq0rOpKY3XmkyP2A5+L+CFxRxpl0obOkqiZ
qK/oT4cFE72EBaSg0XfFej19rd/AqqFSEHpbK4GjZzBAEID932DrgcR6ud74k1TDemQgWZ5dge0wEKMfeNV
zUXhAPuxeIyQ6E+DxuayHIrUKZ7T36ZVz5N56TXnr+iaaejqpJfuSKZxC8YT9ZmRiHeZ+edze+R6QPiV76Q
UIQqzocO+OlWPHUZ2zVINV76DZARqxzZuKWW7JzA1+kTJ3VW3zGaOs53n36lfThb39ULHplsJUPfUGiun8c
K7onzIbkRibbVu/kmrNG8nr2Z4GHMGpQ6KvYyGZNFIwioGMu6Q== sshca@ns2
```

```
# cat /home/sshca/server_ca.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQC+ZP47AOiRPcakZiLm+szvxWF5HKNa+BauaYXt5A6XjuUUJIYXXdT
dzrq+d4rx379jxk5E+RHvPls1Y+5Cvn2/EpBgx20TXoT9+O0Nn7pemHd7qkDb2JqPzoHzjViG033L8UociI
tit13UVMlvB2+miW4RPMHhLjJpJT2BoWzvkBtJdukO/SA+3EYpIZAj8kyFfkp9+2A/A+OCREWoh5EkcjREA
QRay+pc/j3sWDYrymu65OamPwGAe3Ih9/TmflUiwdHaNfJMY0BqpJsdFyUJeS36asjRhLtZltfLNTyY1gOq
3XglLZb76YaQWOoIhzwqQ2WNSt72cY7s8p97loX5LTBSjNw6Kq0rOpKY3XmkyP2A5+L+CFxRxpl0obOkqiZ
qK/oT4cFE72EBaSg0XfFej19rd/AqqFSEHpbK4GjZzBAEID932DrgcR6ud74k1TDemQgWZ5dge0wEKMfeNV
zUXhAPuxeIyQ6E+DxuayHIrUKZ7T36ZVz5N56TXnr+iaaejqpJfuSKZxC8YT9ZmRiHeZ+edze+R6QPiV76Q
UIQqzocO+OlWPHUZ2zVINV76DZARqxzZuKWW7JzA1+kTJ3VW3zGaOs53n36lfThb39ULHplsJUPfUGiun8c
K7onzIbkRibbVu/kmrNG8nr2Z4GHMGpQ6KvYyGZNFIwioGMu6Q== sshca@ns2
```

Observer, finalmente, que os arquivos `/home/sshca/user_ca.pub` e `/etc/ssh/user_ca.pub` também devem ser idênticos:

```
# diff /home/sshca/user_ca.pub /etc/ssh/user_ca.pub
```

Note que tanto no caso de assinatura de chaves de *host* como de chaves de usuário, futuramente, estamos fazendo o acesso no sentido **máquina remota → servidor da CA**, que não é o ideal. Em produção, o correto seria tornar o acesso ao servidor da CA o mais controlado possível, e fazer as assinaturas de chaves apenas de forma local, ou com acessos no sentido **servidor da CA → máquina remota**.

Outro aspecto relevante de segurança que sacrificamos em nosso cenário para agilizar a execução das atividades foi a configuração das chaves privadas de assinatura de *hosts* e usuários sem senha: em produção, é altamente recomendado que essas chaves sejam criadas com senhas fortes, para mitigar a possibilidade de assinatura indevida de chaves em caso de vazamento das mesmas.

9) Automatizando a assinatura de chaves SSH de usuários

1. Vamos agora fazer a segunda "perna" da configuração da CA `ssh` — assinar chaves de usuários. Na linha da atividade anterior, iremos usar um *script* para assinar chaves de usuários; porém, como o cenário de chaves de usuário é mais flexível que o de máquinas, iremos estabelecer algumas premissas para o funcionamento do *script*:

   1. Deve-se estar logado com o mesmo nome do usuário com o qual se deseja assinar a chave.
   2. Deve-se ter conectividade com o servidor `ns2`, no endereço `10.0.42.2`.
   3. O nome de chave será fixo (`~/.ssh/id_rsa`).

   Devido à primeira limitação, é conveniente que o *script* esteja localizado dentro da pasta do usuário — e, como se sabe, ao criar novos usuários o conteúdo da pasta `/etc/skel` é copiado para dentro de seu *home*. Assim, crie a pasta `/etc/skel/scripts`:

   ```
   # mkdir /etc/skel/scripts
   ```

   Dentro dela, crie o arquivo novo, `/etc/skel/scripts/sshsign_user.sh`, com o conteúdo que se segue:

   ```
    1 #!/bin/bash
    2
    3 CA_USER="sshca"
    4 CA_ADDR="10.0.42.2"
    5 CA_USER_PASS="seg10sshca"
    6 SSH_OPTS="-o PreferredAuthentications=password -o PubkeyAuthentication=no"
    7
    8 # testar se chave ja foi assinada
    9 if [ -f ~/.ssh/id_rsa-cert.pub ]; then
   10   echo "Key already signed, bailing."
   11   exit 1
   12 fi
   13
   14 # escanear chave do host ssh-ca, se necessario
   15 rm -f ~/.ssh/known_hosts 2> /dev/null
   16 [ -d ~/.ssh ] || { mkdir ~/.ssh; chmod 700 ~/.ssh; }
   17 if ! ssh-keygen -F ${CA_ADDR} 2>/dev/null 1>/dev/null; then
   18   ssh-keyscan -t rsa -T 10 ${CA_ADDR} 2> /dev/null >> ~/.ssh/known_hosts
   19 fi
   20
   21 # gerar par de chaves RSA, se inexistentes
   22 [ -f ~/.ssh/id_rsa.pub ] || ssh-keygen -f ~/.ssh/id_rsa -t rsa -b 4096 -N '' &>
      /dev/null
   23
   24 # copiar pubkey RSA
   25 sshpass -p "${CA_USER_PASS}" \
   ```

```
26   scp ${SSH_OPTS} ~/.ssh/id_rsa.pub ${CA_USER}@${CA_ADDR}:~
27
28 # assinar pubkey RSA, validade [-5 min -> 1 ano]
29 echo "Signing ~/.ssh/id_rsa.pub key..."
30 user="$( whoami )"
31 sshpass -p "${CA_USER_PASS}" \
32   ssh ${SSH_OPTS} ${CA_USER}@${CA_ADDR} \
33     ssh-keygen -s user_ca \
34     -I ${user} \
35     -n ${user} \
36     -V -5m:+1095d \
37     id_rsa.pub 2> /dev/null
38
39 # copiar pubkey assinada de volta
40 sshpass -p "${CA_USER_PASS}" \
41   scp ${SSH_OPTS} ${CA_USER}@${CA_ADDR}:~/id_rsa-cert.pub ~/.ssh/
42
43 # remover temporarios do diretorio remoto
44 sshpass -p "${CA_USER_PASS}" \
45   ssh ${SSH_OPTS} ${CA_USER}@${CA_ADDR} \
46     rm id_rsa.pub id_rsa-cert.pub
47
48 # copiar pubkey da server_ca e configurar reconhecimento de chaves de host
assinadas
49 echo "@cert-authority * $(sshpass -p "${CA_USER_PASS}" ssh ${SSH_OPTS} ${
CA_USER}@${CA_ADDR} cat server_ca.pub)" > ~/.ssh/known_hosts
50
51 # remover pubkey RSA antiga
52 rm -f ~/.ssh/id_rsa.pub 2> /dev/null
53
54 echo "All done!"
```

Como o *script* guarda grandes semelhanças com o anterior, iremos destacar apenas os pontos de divergência:

1. (Linhas 9-12) Testamos se a chave RSA do usuário já foi assinada anteriormente; se positivo, o programa se encerra.
2. (Linha 22) Caso o usuário não possua um par de chaves criado, cria-se automaticamente um par RSA de 4096 bits, sem senha.
3. (Linhas 30-37) A chave usada para assinar a chave pública do usuário desta vez é a `user_ca`. A validade é ajustada para um ano.
4. (Linha 49) De forma similar ao que foi feito anteriormente, copia-se a chave `server_ca.pub` como uma CA confiável para *hosts* remotos; a partir deste momento, logins de cliente `ssh` deste usuário para *hosts* assinados não terão que confirmar relação de confiança antes de prosseguir.

2. Vamos testar o funcionamento do *script* com o usuário `luke`. Remova o diretório dele para garantir que o conteúdo do `/etc/skel` será copiado no próximo login:

```
# rm -rf /home/luke
```

Agora, logue como o usuário `luke` usando os comandos `su` ou `ssh`:

```
$ whoami ; pwd
luke
/home/luke
```

```
$ ls ~/scripts
sshsign_user.sh
```

3. Execute o *script*:

```
$ bash ~/scripts/sshsign_user.sh
Signing ~/.ssh/id_rsa.pub key...
All done!
```

Note que, novamente, não foi necessário passar quaisquer parâmetros de linha de comando para o *script*. Ele utiliza o *username* do usuário informado pelo comando `whoami` como *principal* da chave.

4. Vamos verificar o funcionamento do *script*. Verifique que os arquivos de chave foram criados:

```
$ ls ~/.ssh/
id_rsa  id_rsa-cert.pub  known_hosts
```

Cheque o conteúdo do arquivo `~/.ssh/known_hosts`, que deve conter informações da CA `ssh` para chaves de *host* assinadas:

```
$ cat ~/.ssh/known_hosts
@cert-authority * ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQC+ZP47AOiRPcakZiLm+szvxWF5HKNa+BauaYXt5A6XjuUUJIYXXdT
dzrq+d4rx379jxk5E+RHvPls1Y+5Cvn2/EpBgx20TXoT9+O0Nn7pemHd7qkDb2JqPzoHzjViG033L8UociI
tit13UVMlvB2+miW4RPMHhLjJpJT2BoWzvkBtJdukO/SA+3EYpIZAj8kyFfkp9+2A/A+OCREWoh5EkcjREA
QRay+pc/j3sWDYrymu65OamPwGAe3Ih9/TmflUiwdHaNfJMY0BqpJsdFyUJeS36asjRhLtZltfLNTyY1gOq
3XglLZb76YaQWOoIhzwqQ2WNSt72cY7s8p97loX5LTBSjNw6Kq0rOpKY3XmkyP2A5+L+CFxRxpl0obOkqiZ
qK/oT4cFE72EBaSg0XfFej19rd/AqqFSEHpbK4GjZzBAEID932DrgcR6ud74k1TDemQgWZ5dge0wEKMfeNV
zUXhAPuxeIyQ6E+DxuayHIrUKZ7T36ZVz5N56TXnr+iaaejqpJfuSKZxC8YT9ZmRiHeZ+edze+R6QPiV76Q
UIQqzocO+OlWPHUZ2zVINV76DZARqxzZuKWW7JzA1+kTJ3VW3zGaOs53n36lfThb39ULHplsJUPfUGiunhi
story8cK7onzIbkRibbVu/kmrNG8nr2Z4GHMGpQ6KvYyGZNFIwioGMu6Q== sshca@ns2
```

5. Agora sim, vamos testar. Tente logar na máquina local usando o endereço IP ou *hostname* (o endereço especial *localhost* não é registrado como um *principal* válido na chave de *host*).

```
$ ssh luke@ns2
Linux ns2 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

Last login: Tue Nov 13 18:40:31 2018 from ::1
luke@ns2:~$
```

Perfeito! Não tivemos que digitar a senha do usuário ou confirmar a relação de confiança com o servidor, como esperado.

6. Todos as características do sistema que queríamos testar estão funcionais. Claro que devemos levar em consideração que todos os testes foram feitos dentro da máquina `ns2` — não testamos o funcionamento de contas de usuário via LDAP na rede, ou a autenticação de login remoto usando a CA do `ssh`.

   Retorne ao usuário `root` na máquina `ns2` e remova o diretório do usuário `luke`. Vamos prosseguir com a configuração do *template* e de um cliente Linux para testar as funcionalidades.

   ```
   # hostname ; whoami
   ns2
   root
   ```

   ```
   # rm -rf /home/luke
   ```

10) Configurando o template para funcionar com LDAP/SSH-CA

1. Como mencionado anteriormente, iremos configurar a VM `debian-template` para funcionar com os sistemas de autenticação do LDAP e SSH-CA. Assim, todas as máquinas que forem derivadas futuramente desse *template* estarão automaticamente integradas com o sistema de autenticação do nosso *datacenter* hipotético.

   Ligue a VM `debian-template` e faça login como o usuário `root`:

   ```
   # hostname ; whoami
   debian-template
   root
   ```

2. Crie o arquivo novo `/root/scripts/sshsign.sh`, e cole dentro dele o conteúdo do *script* que discutimos no passo (1) da atividade (8).

   ```
   # nano /root/scripts/sshsign.sh
   (...)
   ```

```
# ls /root/scripts/
changehost.sh  sshsign.sh  syncdirs.sh
```

3. Agora, crie o diretório `/etc/skel/scripts`, e dentro dele crie o arquivo novo `/etc/skel/scripts/sshsign_user.sh`, com conteúdo idêntico ao do *script* que foi apresentado no passo (1) da atividade (9).

```
# mkdir /etc/skel/scripts
```

```
# nano /etc/skel/scripts/sshsign_user.sh
(...)
```

```
# ls /etc/skel/scripts/
sshsign_user.sh
```

4. Instale as dependências para funcionamento dos *scripts* criados anteriormente e também para integração do sistema de autenticação PAM com o LDAP.

```
# apt-get install sshpass nslcd
```

Durante a instalação do pacote `nslcd`, informe as seguintes opções:

Tabela 7. Configurações do pacote nslcd

Pergunta	**Opção**
URI do servidor LDAP	ldap://ldap.intnet/
Base de buscas do servidor LDAP	dc=intnet
Serviços de nome para configurar	passwd, group, shadow

5. Assim como configurado antes, informe ao PAM que diretórios *home* inexistentes de usuários do LDAP devem ser criados automaticamente. Crie o arquivo novo `/usr/share/pam-configs/mkhomedir`, e cole dentro dele o conteúdo do arquivo discutido durante o passo (1) da atividade (6).

```
# nano /usr/share/pam-configs/mkhomedir
(...)
```

```
# pam-auth-update
```

Durante a configuração do PAM, na pergunta "Perfis PAM para habilitar", mantenha todas as caixas marcadas e selecione OK.

6. Finalmente, vamos alterar o *script* `/root/scripts/changehost.sh`, criado durante a sessão (1) deste curso, para invocar automaticamente o *script* `/root/scripts/sshsign.sh` ao final e reiniciar os *daemons* do `nslcd` e `nscd`. Altere o conteúdo deste arquivo para:

```
#!/bin/bash


# exibir uso do script e sair
function usage() {
  echo "  Usage: $0 -h HOSTNAME -i IPADDR -g GATEWAY"
  echo "  Netmask is assumed as /24."
  exit 1
}


# testar sintaxe valida de HOSTNAME
function valid_host() {
  if [[ "$nhost" =~ [^a-z0-9] ]]; then
    echo "  [*] HOSTNAME must be lowercase alphanumeric: [a-z0-9]*"
    usage
  elif [ ${#nhost} -gt 63 ]; then
    echo "  [*] HOSTNAME must have <63 chars"
    usage
  fi
}


# testar sintaxe valida de IPADDR/GATEWAY
function valid_ip() {
  local  ip=$1
  local  stat=1

  if [[ $ip =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
    OIFS=$IFS
    IFS='.'
    ip=($ip)
    IFS=$OIFS
    [[ ${ip[0]} -le 255 && \
       ${ip[1]} -le 255 && \
       ${ip[2]} -le 255 && \
       ${ip[3]} -le 255 ]]
        stat=$?
  fi

  if [ $stat -ne 0 ] ; then
    echo "  [*] Invalid syntax for $2"
    usage
  fi
}

```

```
47
48 while getopts ":g:h:i:" opt; do
49   case "$opt" in
50     g)
51       ngw=${OPTARG}
52       ;;
53     h)
54       nhost=${OPTARG}
55       ;;
56     i)
57       nip=${OPTARG}
58       ;;
59     *)
60       usage
61       ;;
62   esac
63 done
64
65 # testar se parametros foram informados
66 [ -z $ngw ]   && { echo "  [*] No gateway?"; usage; }
67 [ -z $nhost ] && { echo "  [*] No hostname?"; usage; }
68 [ -z $nip ]   && { echo "  [*] No ipaddr?"; usage; }
69
70 # testar sintaxe de parametros
71 valid_ip $nip "IPADDR"
72 valid_ip $ngw "GATEWAY"
73 valid_host $nhost
74
75 # alterar endereco ip/gateway
76 iff="/etc/network/interfaces"
77 cip="$( egrep '^address ' $iff | awk -F'[ /]' '{print $2}' )"
78 cgw="$( egrep '^gateway ' $iff | awk '{print $NF}' )"
79 sed -i "s|${cip}|${nip}|g" $iff
80 sed -i "s|${cgw}|${ngw}|g" $iff
81 ip addr flush label 'enp0s*'
82
83 # alterar hostname local
84 chost="$( hostname -s )"
85 sed -i "s/${chost}/${nhost}/g" /etc/hosts
86 sed -i "s/${chost}/${nhost}/g" /etc/hostname
87
88 invoke-rc.d hostname.sh restart
89 invoke-rc.d networking restart
90 hostnamectl set-hostname $nhost
91
92 # assinar chaves SSH
93 bash /root/scripts/sshsign.sh
94
95 # reiniciar sistema de autenticacao LDAP
96 systemctl restart nslcd.service
97 systemctl restart nscd.service
```

```
98
99 for table in `ls -1 /var/cache/nscd` ; do
100   nscd --invalidate $table
101 done
```

Ao invocar este *script* após a criação de uma nova VM, iremos não apenas alterar seu endereço IP, *gateway* e *hostname* mas também integrá-la com o sistema de autenticação remota do LDAP e SSH-CA em um único comando, como veremos a seguir.

Findo este passo, desligue a máquina `debian-template`.

11) Ajuste das regras de firewall

1. Observando os acessos feitos pelo *script* em `/root/scripts/sshsign.sh`, note que a máquina efetuando as assinaturas de chave necessita conseguir alcançar a máquina `ns2` pela rede, e logar via SSH com o usuário `sshca`. Mais além, as autenticações via rede são feitas junto ao servidor OpenLDAP, operando com o *daemon* `slapd`. Nenhum desses requisitos foi previsto em nossas regras de firewall originais, então temos que ajustá-las para permitir que máquinas da DMZ e Intranet consigam realizar esses acessos.

   Vamos listar os requerimentos de regras:

   a. Máquinas na DMZ devem conseguir acessar a máquina `ns2` nas portas 22/TCP (SSH) e 389/TCP (LDAP). Como essas máquinas estão todas na mesma sub-rede, os acessos não passam pelo firewall e nenhuma configuração adicional se faz necessária.

   b. Máquinas na Intranet devem conseguir acessar a máquina `ns2` nas portas 22/TCP (SSH) e 389/TCP (LDAP). Como esse tráfego passa **através do** firewall, devemos inserir regras na *chain* FORWARD.

   Logue como `root` na máquina `ns1`:

   ```
   # hostname ; whoami
   ns1
   root
   ```

   Basta criar uma regra para atender o requisito (b), como se segue:

   ```
   # iptables -A FORWARD -s 192.168.42.0/24 -d 10.0.42.2/32 -p tcp -m multiport
   --dports 22,389 -j ACCEPT
   ```

   Salve as regras na configuração do firewall local:

```
# /etc/init.d/netfilter-persistent save
[....] Saving netfilter rules...run-parts: executing /usr/share/netfilter-
persistent/plugins.d/15-ip4tables save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
done.
```

12) Configurando um cliente Linux

Nesta atividade iremos criar uma máquina cliente Linux para utilizarmos como ponto de partida para os logins `ssh` nos diferentes servidores que configuraremos durante este curso. O nome da máquina será `client`, alocada para a Intranet com o endereço IP 192.168.42.2/24. Iremos integrá-la com os sistemas de autenticação do LDAP e SSH-CA, e testar login remoto na máquina `ns2`.

1. Clone a máquina `debian-template` seguindo os mesmos passos da atividade (6) da sessão 1. Para o nome da máquina, escolha `client`.
2. Após a clonagem, na janela principal do Virtualbox, clique com o botão direito sobre a VM `client` e depois em *Settings*.

   Em *Network* > *Adapter 1* > *Attached to* > *Host-only Adapter*, altere o nome da rede *host-only* para o mesmo da interface da VM `ns1` que está alocada à Intranet. Seguindo o exemplo mostrado no início da sessão 2, a rede escolhida seria a `Virtualbox Host-Only Ethernet Adapter #3`.

   Clique em *OK*, e ligue a máquina `client`.

3. Logue como o usuário `root` — o primeiro login poderá demorar um pouco até que a tentativa de autenticação via rede no servidor LDAP, neste momento inatingível, incorra em *timeout*. Feito o login, use o script `/root/scripts/changehost.sh` para configurar a máquina de forma automática:

   ```
   # hostname ; whoami
   debian-template
   root
   ```

   ```
   # bash ~/scripts/changehost.sh -h client -i 192.168.42.2 -g 192.168.42.1
   Signing ssh_host_ecdsa_key.pub key...
   Signing ssh_host_ed25519_key.pub key...
   Signing ssh_host_rsa_key.pub key...
   Configuring host key trust...
   Configuring user key trust...
   All done!
   ```

   ```
   # hostname
   client
   ```

4. Se tudo tiver funcionado corretamente, a máquina `client` já estará integrada aos sistemas de

autenticação LDAP e SSH-CA. Faça login como o usuário `luke` usando os comandos `su` ou `ssh`:

```
$ hostname ; whoami ; pwd
client
luke
/home/luke
```

5. Vamos criar um par de chaves para esse usuário e assiná-las. Como o conteúdo do diretório *home* foi copiado diretamente do `/etc/skel`, temos à disposição o *script* para essa tarefa na pasta `~/scripts/sshsign_user.sh`. Execute-o:

```
$ bash ~/scripts/sshsign_user.sh
Signing ~/.ssh/id_rsa.pub key...
All done!
```

6. Vamos testar? Tente logar usando o usuário `luke` na máquina `ns2`:

```
$ ssh luke@ns2
Creating directory '/home/luke'.
Linux ns2 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

Last login: Tue Nov 13 21:05:01 2018 from 127.0.0.1
luke@ns2:~$
```

```
$ hostname ; whoami ; pwd
ns2
luke
/home/luke
```

Excelente! Conseguimos logar sem ter que confirmar a relação de confiança com o servidor e sem digitar senha, como esperado.

13) Configurando o firewall para funcionar com LDAP/SSH-CA

Suponha, agora, que queremos integrar o firewall no sistema de autenticação LDAP/SSH-CA, mas com um maior nível de restrição. Sendo uma máquina crítica, não podemos permitir que qualquer usuário faça login nessa máquina, sendo necessário implementar controles mais estritos.

O primeiro passo, naturalmente, é fazer a integração com os sistemas de autenticação. Vamos fazer isso:

1. Logue como usuário `root` na máquina `ns1`:

```
# hostname ; whoami
ns1
root
```

2. Instale as dependências para funcionamento dos *scripts* e integração do sistema de autenticação.

```
# apt-get install sshpass nslcd
```

Novamente, durante a instalação do pacote `nslcd`, informe as seguintes opções:

Tabela 8. Configurações do pacote nslcd

Pergunta	Opção
URI do servidor LDAP	ldap://ldap.intnet/
Base de buscas do servidor LDAP	dc=intnet
Serviços de nome para configurar	passwd, group, shadow

3. Configure a criação automática de diretórios, com o arquivo novo `/usr/share/pam-configs/mkhomedir`. Para maior conveniência, você pode copiar o arquivo diretamente da máquina `ns2` para o local apropriado usando o comando `scp`, como se segue:

```
# scp -o StrictHostKeyChecking=no aluno@ns2:/usr/share/pam-configs/mkhomedir
/usr/share/pam-configs/
Warning: Permanently added 'ns2,10.0.42.2' (ECDSA) to the list of known hosts.
aluno@ns2's password:
mkhomedir
100%  170   449.4KB/s   00:00
```

```
# pam-auth-update
```

Durante a configuração do PAM, na pergunta "Perfis PAM para habilitar", mantenha todas as caixas marcadas e selecione OK.

4. Crie o arquivo novo `/root/scripts/sshsign.sh` e cole o conteúdo do *script* de assinatura de chaves de *host* que utilizamos anteriormente:

```
# nano /root/scripts/sshsign.sh
(...)
```

```
# ls /root/scripts/
changehost.sh  signzone-intnet.sh  sshsign.sh  syncdirs.sh
```

Execute-o:

```
# bash ~/scripts/sshsign.sh
Signing ssh_host_ecdsa_key.pub key...
Signing ssh_host_ed25519_key.pub key...
Signing ssh_host_rsa_key.pub key...
Configuring host key trust...
Configuring user key trust...
All done!
```

14) Restringindo login por grupos e usuários

Agora sim, com a integração concluída, imagine o seguinte cenário: não queremos que usuários do grupo `sysadm`, do qual faz parte o usuário `luke`, possam logar no firewall. Esse permissão será dada apenas a membros do grupo `fwadm`, que criaremos a seguir. Um desses usuários é o colaborador `han`, cuja senha será `seg10han`. Como configurar esse tipo de restrição?

1. Primeiro, vamos criar o grupo e usuário. Logue na máquina `ns2` como usuário `root`:

   ```
   # hostname ; whoami
   ns2
   root
   ```

2. Crie o grupo:

   ```
   # ldapaddgroup fwadm
   Successfully added group fwadm to LDAP
   ```

   Usuário:

   ```
   # ldapadduser han fwadm
   Successfully added user han to LDAP
   Successfully set password for user han
   ```

   Adicione o usuário ao grupo:

   ```
   # ldapaddusertogroup han fwadm
   Successfully added user han to group cn=fwadm,ou=Groups,dc=intnet
   ```

   E, finalmente, configure a senha do usuário `han`:

```
# ldapsetpasswd han
Changing password for user uid=han,ou=People,dc=intnet
New Password:
Retype New Password:
Successfully set password for user uid=han,ou=People,dc=intnet
```

3. De volta ao firewall, como usuário `root`:

```
# hostname ; whoami
ns1
root
```

Edite o arquivo `/etc/nslcd.conf`, configurando a opção `pam_authz_search`. Essa opção permite que sejam definidos filtros de busca para o `nslcd`, através dos quais podemos restringir que usuários e/ou grupos podem logar na máquina local. No caso, queremos que apenas membros do grupo `fwadm` possam logar, portanto adicionamos a seguinte linha ao final do arquivo:

```
# echo 'pam_authz_search
(&(objectClass=posixGroup)(cn=fwadm)(memberUid=$username))' >> /etc/nslcd.conf
```

Podemos customizar o filtro acima para incluir apenas usuários específicos, ou mesmo DNs que possuam um atributo qualquer (por exemplo, e-mails com um determinado sufixo). Tome sempre cuidado para não filtrar todos os usuários disponíveis no LDAP acidentalmente — é importante, nesses casos, sempre manter uma conta local (como `aluno` ou `root`, no nosso caso específico) com acesso ao sistema.

Reinicie os serviços do `nslcd` e `nscd`:

```
# systemctl restart nslcd.service
```

```
# systemctl restart nscd.service
```

Verifique que o usuário `han` é visto como membro do grupo `fwadm`:

```
# groups han
han : fwadm
```

Ocasionalmente, reiniciar o `nscd` não é suficiente para que ele detecte novas alterações na base de usuários/grupos do LDAP. Nesse caso, podemos invalidar as *caches* das tabelas do `nscd` com o comando:

```
# nscd --invalidate TABLE
```

As tabelas disponíveis podem ser consultadas na página de manual do `nscd`, ou vistas diretamente dentro da pasta `/var/cache/nscd`:

```
# ls -1 /var/cache/nscd/
group
hosts
netgroup
passwd
services
```

Para invalidar todas as *caches* do `nscd`, podemos executar por exemplo:

```
# for table in `ls -1 /var/cache/nscd` ; do nscd --invalidate $table ; done
```

4. Vamos testar a efetividade do controle aplicado. Na máquina `client`, faça login como o usuário `luke`:

```
$ hostname ; whoami
client
luke
```

Tente logar via `ssh` na máquina `ns1`:

```
$ ssh luke@ns1
LDAP authorisation check failed
Authentication failed.
```

Como o usuário `luke` não pertence ao grupo `fwadm`, o acesso é negado. Observando o log de *debug* do `nslcd`, podemos ver que a pesquisa com o filtro aplicado anteriormente não retorna resultados:

```
nslcd: [95f874] <authz="luke"> DEBUG: trying pam_authz_search
"(&(objectClass=posixGroup)(cn=fwadm)(memberUid=luke))"
nslcd: [95f874] <authz="luke"> DEBUG: myldap_search(base="dc=intnet",
filter="(&(objectClass=posixGroup)(cn=fwadm)(memberUid=luke))")
nslcd: [95f874] <authz="luke"> DEBUG: ldap_result(): end of results (0 total)
nslcd: [95f874] <authz="luke"> pam_authz_search
"(&(objectClass=posixGroup)(cn=fwadm)(memberUid=luke))" found no matches
```

5. Vamos fazer o mesmo procedimento com o usuário `han`. Logue-se como `han` na máquina `client`:

```
$ hostname ; whoami
client
han
```

Como é a primeira vez que estamos usando este usuário, gere um par de chaves assinadas para ele:

```
$ bash ~/scripts/sshsign_user.sh
Signing ~/.ssh/id_rsa.pub key...
All done!
```

Tente logar via `ssh` na máquina `ns1`:

```
$ ssh han@ns1
Creating directory '/home/han'.
Linux ns1 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

han@ns1:~$
```

```
$ hostname ; whoami ; pwd
ns1
han
/home/han
```

Observando o log de *debug* do `nslcd`, podemos ver que a pesquisa com o filtro aplicado anteriormente retorna como resultado o grupo `cn=fwadm,ou=Groups,dc=intnet`:

```
nslcd: [138641] <authz="han"> DEBUG: trying pam_authz_search
"(&(objectClass=posixGroup)(cn=fwadm)(memberUid=han))"
nslcd: [138641] <authz="han"> DEBUG: myldap_search(base="dc=intnet",
filter="(&(objectClass=posixGroup)(cn=fwadm)(memberUid=han))")
nslcd: [138641] <authz="han"> DEBUG: ldap_result(): cn=fwadm,ou=Groups,dc=intnet
nslcd: [138641] <authz="han"> DEBUG: pam_authz_search found
"cn=fwadm,ou=Groups,dc=intnet"
```

15) Restringindo logins SSH apenas via chaves assimétricas

Apesar de o controle que aplicamos na atividade anterior ser interessante, ainda não resolvemos o problema completamente. Como é possível tentar login na máquina `ns1` usando senha, é possível que um atacante tente login por força-bruta, adivinhando a senha do usuário `han`, até conseguir. Vamos resolver isso.

1. Primeiro, vamos constatar o problema. Logue na máquina `client` como o usuário `han`:

```
$ hostname ; whoami
client
han
```

Para evitar que o cliente `ssh` use nossa chave assinada, passe as opções abaixo para o comando. Em seguida, digite a senha correta do usuário `han`:

```
$ ssh -o PreferredAuthentications=keyboard-interactive,password -o
PubkeyAuthentication=no han@ns1
han@ns1's password:
Linux ns1 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

Last login: Wed Nov 14 00:26:13 2018 from 192.168.42.2
han@ns1:~$
```

```
$ hostname ; whoami
ns1
han
```

Note que conseguimos fazer o login usando senha normalmente, sem usar a chave assinada pela CA.

2. Logue como `root` na máquina `ns1`:

```
# hostname ; whoami
ns1
root
```

Iremos aplicar o controle sobre a opção `PasswordAuthentication` do `sshd`, desativando-o. Assim, não será mais possível logar via senha, apenas via chaves assimétricas. Execute o comando abaixo:

```
# sed -i 's/^#\(PasswordAuthentication\).*/\1 no/' /etc/ssh/sshd_config
```

E reinicie o `sshd`:

```
# systemctl restart sshd.service
```

3. De volta à máquina `client`, como `han`:

```
$ hostname ; whoami
client
han
```

Tente novamente logar usando senha:

```
$ ssh -o PreferredAuthentications=keyboard-interactive,password -o
PubkeyAuthentication=no han@ns1
Permission denied (publickey).
```

Note que a permissão foi negada, pois apenas o método `publickey` é aceito para autenticação. Remova as opções do `ssh` e tente novamente, desta vez usando chaves:

```
$ ssh han@ns1
Linux ns1 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

Last login: Wed Nov 14 00:27:39 2018 from 192.168.42.2
han@ns1:~$
```

```
$ hostname ; whoami
ns1
han
```

16) Bloqueando tentativas de brute force contra o SSH

Não podemos aplicar o mesmo tipo de controle que fizemos na máquina `ns1` (o firewall da rede) no servidor `ns2` (o servidor LDAP) — nosso *script* de assinatura de chaves de *host* e de usuário utiliza login via senha com o usuário `sshca` para operar. A alteração dos *scripts* para usarem chaves e a correspondente restrição a login usando senhas teria implicações muito negativas na segurança do sistema, neste momento. Vamos implementar um controle diferente, então: proteção contra ataques de força-bruta usando a ferramenta Fail2ban.

O Fail2ban opera através da análise de eventos de log (normalmente registrados no diretório `/var/log`) e seu processamento através de expressões regulares. Caso um evento "case" (ou seja, ocorra um *match*) com uma expressão regular configurada, o Fail2ban irá adicionar uma unidade ao contador de violações de um determinado *host* remoto. Se esse *host* ultrapassar o número de violações configuradas em um dado período, o Fail2ban irá então tomar alguma ação configurada pelo administrador (registrar um evento nos logs, enviar um e-mail para o administrador ou até mesmo bloquear de forma automática o *host* no firewall local).

Várias expressões regulares já vêm pré-configuradas no Fail2ban, para as ferramentas mais populares (como o `sshd`, o servidor web Apache ou o servidor SMTP Postfix). Caso se deseje configurar expressões regulares para ferramentas customizadas, também é possível fazê-lo.

1. Logue como o usuário `root` na máquina `ns2`:

```
# hostname ; whoami
ns2
root
```

2. Instale a ferramenta `fail2ban`:

```
# apt-get install fail2ban
```

3. Note que, por padrão, apenas a *jail* `sshd` vem habilitada no Debian. Não teremos que fazer qualquer alteração nesse sentido, já que é justamente o serviço `ssh` que queremos proteger.

```
# cat /etc/fail2ban/jail.d/defaults-debian.conf
[sshd]
enabled = true
```

4. As opções padrão do Fail2ban ficam configuradas no arquivo `/etc/fail2ban/jail.conf`, seção `[DEFAULT]`. Em particular, temos interesse nas seguintes configurações:

   - `findtime`: Intervalo em que o Fail2ban irá registrar violações de *hosts* remotos.
   - `maxretry`: Número de violações máximo permitido dentro do período `findtime` definido acima. Caso este valor seja ultrapassado, o Fail2ban irá tomar a ação configurada pelo administrador.
   - `bantime`: Período em que o *host* remoto será afetado pela ação configurada. Caso esta ação seja, por exemplo, um bloqueio no firewall local, o *host* ficará banido pelo tempo especificado aqui.

   Os valores padrão para as variáveis acima são os que se seguem:

```
# cat /etc/fail2ban/jail.conf | sed -n -e '/^\[DEFAULT\]/,/^\[/p' | grep
'^maxretry\|^bantime\|^findtime'
bantime  = 600
findtime  = 600
maxretry = 5
```

5. Vamos configurar o seguinte cenário: caso um atacante seja detectado pelo Fail2ban com mais de 3 violações (`maxretry`) num período de dez minutos (`findtime`), então iremos bani-lo via regra no firewall local (ação `iptables-multiport`) por dez minutos (`bantime`). Como o `findtime` e o `bantime` padrão estão corretos, iremos apenas configurar as duas outras variáveis, como se segue:

```
# echo "maxretry = 3" >> /etc/fail2ban/jail.d/defaults-debian.conf
```

```
# echo "banaction = iptables-multiport" >> /etc/fail2ban/jail.d/defaults-
debian.conf
```

O arquivo `/etc/fail2ban/jail.d/defaults-debian.conf` ficou assim, portanto:

```
# cat /etc/fail2ban/jail.d/defaults-debian.conf
[sshd]
enabled = true
maxretry = 3
banaction = iptables-multiport
```

6. Uma outra configuração necessária é comentar uma linha do arquivo `/etc/fail2ban/filter.d/sshd.conf` que contém uma expressão regular para detectar entradas no seguinte formato no arquivo `/var/log/auth.log`:

```
Oct 30 12:03:47 ldap sshd[6677]: pam_unix(sshd:auth): authentication failure;
logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.42.2  user=sshca
```

Em sistemas com autenticação LDAP, como é o nosso caso, a linha acima é inserida em tentativas de login mesmo em caso de sucesso, como reportado em https://github.com/fail2ban/fail2ban/issues/106 . Para corrigir esse falso positivo, basta executar:

```
# sed -i 's/^\(.*pam_unix.*\)/#\1/' /etc/fail2ban/filter.d/sshd.conf
```

7. Reinicie o Fail2ban para aplicar as configurações que realizamos:

```
# systemctl restart fail2ban.service
```

8. O Fail2ban criará novas *chains* no firewall para inserção de regras de banimento, quando adequado. Observe que o firewall está, até este momento, sem regras de BLOCK ou REJECT:

```
# iptables -L -vn
Chain INPUT (policy ACCEPT 37 packets, 5021 bytes)
 pkts bytes target     prot opt in     out     source               destination
   26  1820 f2b-sshd   tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
multiport dports 22

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 13 packets, 1152 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain f2b-sshd (1 references)
 pkts bytes target     prot opt in     out     source               destination
   26  1820 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

Agora, vamos fazer uma simulação de ataque ao `sshd`. Monitore o arquivo `/var/log/fail2ban.log`:

```
# tail -n5 -f /var/log/fail2ban.log
2018-11-14 00:31:44,246 fail2ban.filter         [2099]: INFO    Set findtime = 600
2018-11-14 00:31:44,246 fail2ban.filter         [2099]: INFO    Set jail log file
encoding to UTF-8
2018-11-14 00:31:44,246 fail2ban.filter         [2099]: INFO    Set maxlines = 10
2018-11-14 00:31:44,277 fail2ban.server         [2099]: INFO    Jail sshd is not a
JournalFilter instance
2018-11-14 00:31:44,281 fail2ban.jail           [2099]: INFO    Jail 'sshd' started
```

9. Logue na máquina `client` como o usuário `han`, por exemplo:

```
$ hostname ; whoami
client
han
```

Para disparar o filtro do Fail2ban na máquina `ns2`, não poderemos usar o login via chaves assimétricas, que obterá sucesso. Faça login usando senha como mostrado no comando a seguir; digite senhas incorretas para ativar a detecção do Fail2ban:

```
$ ssh -o PreferredAuthentications=keyboard-interactive,password -o
PubkeyAuthentication=no han@ns2
han@ns2's password:
Permission denied, please try again.
han@ns2's password:
Permission denied, please try again.
han@ns2's password:
Permission denied (publickey,password).
```

Tente logar novamente:

```
$ ssh -o PreferredAuthentications=keyboard-interactive,password -o
PubkeyAuthentication=no han@ns2
ssh: connect to host ns2 port 22: Connection refused
```

A máquina foi bloqueada, como esperado.

10. De volta à máquina `ns2`, como o usuário `root`:

```
# hostname ; whoami
ns2
root
```

Note que os eventos de senha incorreta foram registrados pelo Fail2ban, bem como o banimento:

```
2018-11-14 00:34:01,944 fail2ban.filter         [2099]: INFO    [sshd] Found
192.168.42.2
2018-11-14 00:34:04,830 fail2ban.filter         [2099]: INFO    [sshd] Found
192.168.42.2
2018-11-14 00:34:08,081 fail2ban.filter         [2099]: INFO    [sshd] Found
192.168.42.2
2018-11-14 00:34:08,534 fail2ban.actions        [2099]: NOTICE  [sshd] Ban
192.168.42.2
```

Observe que a regra de REJECT foi inserida automaticamente pelo Fail2ban no firewall local:

```
# iptables -L f2b-sshd -vn
Chain f2b-sshd (1 references)
 pkts bytes target     prot opt in     out     source               destination
    1    60 REJECT     all  --  *      *       192.168.42.2         0.0.0.0/0
reject-with icmp-port-unreachable
  612 70528 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

Para remover o banimento de um endereço IP antes que o tempo total do `bantime` tenha transcorrido, é possível usar o comando `fail2ban-client`, como mostrado a seguir:

```
# fail2ban-client set sshd unbanip 192.168.42.2
192.168.42.2
```

Note que a regra de firewall é apagada, como esperado:

```
# iptables -L f2b-sshd -vn
Chain f2b-sshd (1 references)
 pkts bytes target     prot opt in     out     source               destination
  395 25992 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

11. De volta à máquina `client`, como `han`, podemos tentar o login via senha novamente — desta vez, digite a senha correta:

```
$ hostname ; whoami
client
han
```

```
$ ssh -o PreferredAuthentications=keyboard-interactive,password -o
PubkeyAuthentication=no han@ns2
han@ns2's password:
Creating directory '/home/han'.
Linux ns2 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

han@ns2:~$
```

```
$ hostname ; whoami
ns2
han
```

Sessão 4: Controles de segurança

Estando configurado nosso sistema de autenticação centralizado, quais seriam os próximos passos para realizar o *hardening* do ambiente? Nesta sessão, iremos tratar de algumas configurações mais simples, num escopo particular, mas que somadas tornarão o *datacenter* muito mais resiliente contra ataques, além de mais funcional. Iremos verificar se as senhas escolhidas pelos usuários são de fato seguras, implementar *quotas* de disco em um servidor de arquivos Linux, permitir controle mais granular de permissões de arquivos através de ACLs (*Access Control Lists*) e realizar um controle refinado de autorizações administrativas usando o comando `sudo`.

Vamos ao trabalho?

1) Topologia desta sessão

A figura abaixo mostra a topologia de rede que será utilizada nesta sessão, com as máquinas relevantes em destaque.

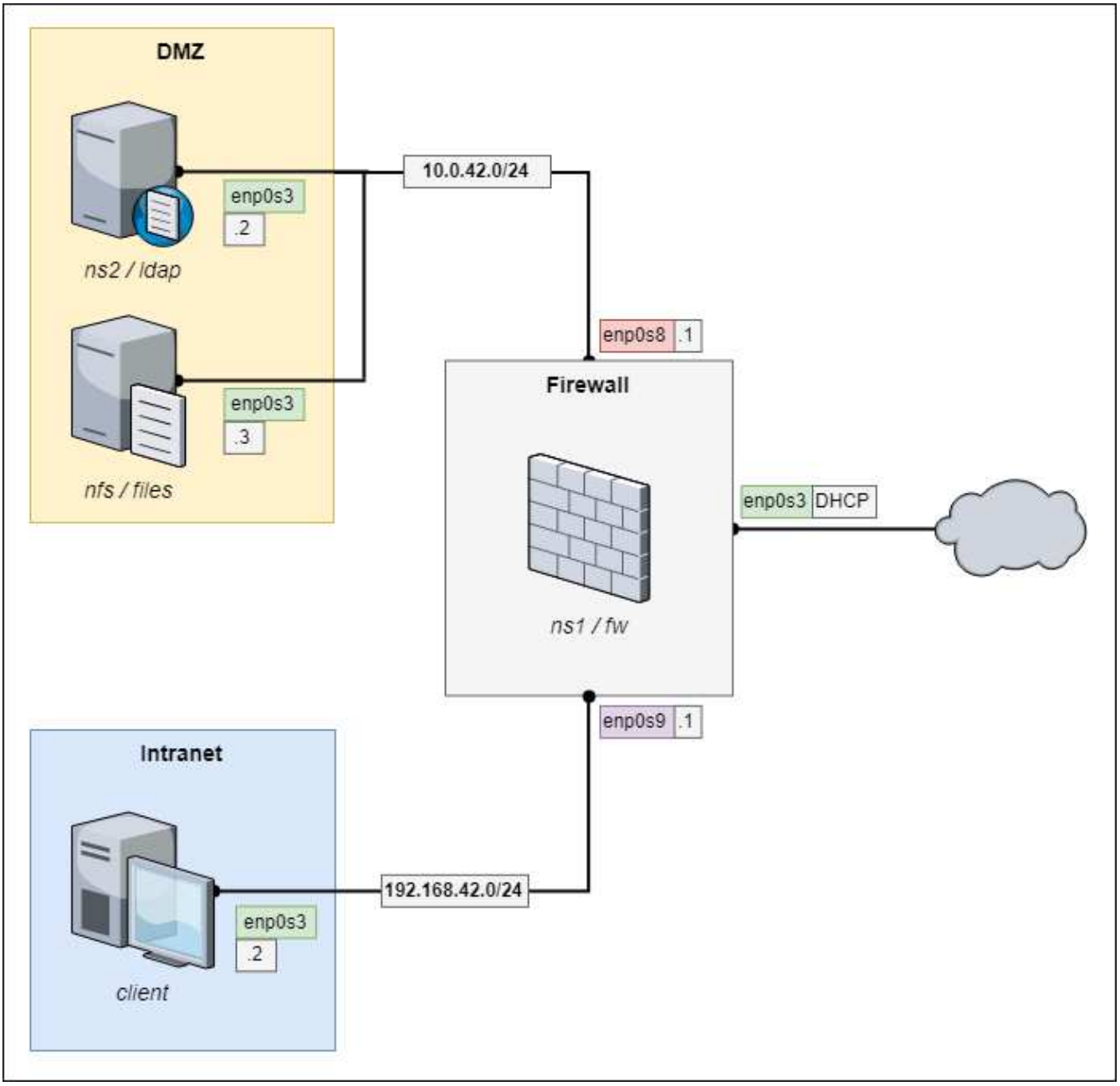

Figura 28. Topologia de rede desta sessão

Teremos agora quatro máquinas:

- `ns1`, com *alias* `firewall`, atuando como firewall de rede e DNS primário. Endereços IP via DHCP (interface *bridge*), 10.0.42.1/24 (interface DMZ) e 192.168.42.1/24 (interface Intranet).
- `ns2`, com *alias* `ns2`, localizada na DMZ e atuando como DNS secundário, servidor LDAP de autenticação centralizada e repositório de chaves SSH-CA. Endereço IP 10.0.42.2/24.
- `nfs`, com *alias* `files`, localizada na DMZ e atuando como servidor de arquivos NFS. Endereço IP 10.0.42.3/24.
- `client`, localizada na Intranet e usada como ponto de partida para logins remotos nos diferentes servidores do *datacenter* simulado. Endereço IP 192.168.42.2/24.

1. Na topologia há a previsão de criação de dois novos registros DNS, um mapeamento direto para o nome `nfs` e um *alias* dessa mesma máquina para o nome `files`. Vamos ajustar o DNS: acesse a máquina `ns1` como o usuário `root`:

```
# hostname ; whoami
ns1
root
```

Edite o arquivo de zonas `/etc/nsd/zones/intnet.zone`, inserindo uma entrada A e outra CNAME para a máquina `nfs`, como se segue. **Não se esqueça** de incrementar o valor do serial no topo do arquivo!

```
# nano /etc/nsd/zones/intnet.zone
(...)
```

```
# grep nfs /etc/nsd/zones/intnet.zone
nfs     IN    A                10.0.42.3
files   IN    CNAME            nfs
```

Desta vez também será necessário alterar o arquivo de mapeamento reverso `/etc/nsd/zones/10.0.42.zone` com um registro PTR para a nova máquina. Novamente, lembre-se de incrementar o **serial** neste arquivo também.

```
# nano /etc/nsd/zones/10.0.42.zone
(...)
```

```
# grep nfs /etc/nsd/zones/10.0.42.zone
3       IN   PTR              nfs.intnet.
```

Assine o arquivo de zonas usando o *script* criado anteriormente:

```
# bash /root/scripts/signzone-intnet.sh
reconfig start, read /etc/nsd/nsd.conf
ok
ok
ok
ok removed 0 rrsets, 0 messages and 0 key entries
```

Verifique que as entradas foram criadas com sucesso, usando o comando `dig`:

```
# dig nfs.intnet +short
10.0.42.3
```

```
# dig files.intnet +short
nfs.intnet.
10.0.42.3
```

```
# dig -x 10.0.42.3 +short
nfs.intnet.
```

2) Requisitos de senha na base LDAP

Uma preocupação frequente dos analistas de segurança é quanto às senhas dos usuários: será que elas tem um tamanho apropriado, não utilizam palavras constantes em *wordlists*, contém caracteres especiais? Apesar de termos configurado o acesso aos nossos servidores usando chaves assimétricas via SSH-CA (e, no caso da máquina `ns1`, aplicado restrição de acesso exclusivamente via chaves), não é interessante que nos despreocupemos totalmente da segurança de senhas dos usuários — afinal, os logins na máquina `ns2` ainda podem usar senhas, por exemplo.

Podemos utilizar o *policy overlay* do `slapd` (documentação em https://www.openldap.org/doc/admin24/overlays.html ou `man 5 slapo-ppolicy`) para implementar alguns controles no diretório LDAP para exigir aspectos mínimos de qualidade das senhas dos usuários, tais como:

- `pwdInHistory`: Histórico de senhas, mantém uma lista de senhas passadas que impede que o usuário as repita. O número de senhas mantidas em histórico é configurável.
- `pwdMaxAge`: Tempo máximo de validade da senha.
- `pwdMinAge`: Tempo mínimo de validade da senha, para evitar que o usuário circule pelo histórico rapidamente e apague o registro de uma senha que queira repetir.
- `pwdMinLength`: Tamanho mínimo da senha do usuário, em caracteres.
- `pwdMaxFailure`: Número máximo de tentativas de *bind* com senha incorreta antes que a conta do usuário seja travada.
- `pwdCheckQuality`: Define uma função externa para checagem de qualidade da senha do usuário — esta é uma extensão não-padrão da política de senhas do diretório LDAP, e não iremos configurá-la. O website http://ltb-project.org/wiki/documentation/openldap-ppolicy-check-password disponibiliza um software customizado que pode ser usado para implementar esse tipo de política.

1. Faça login como `root` na máquina `ns2`:

   ```
   # hostname ; whoami
   ns2
   root
   ```

   Para habilitar esses controles em nossa base LDAP, o primeiro passo é carregar o arquivo LDIF do *schema* com as informações de políticas de senhas:

```
# ldapadd -Y external -H ldapi:/// -f /etc/ldap/schema/ppolicy.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
adding new entry "cn=ppolicy,cn=schema,cn=config"
```

2. Em seguida, iremos adicionar o módulo `/usr/lib/ldap/ppolicy.la` à lista de módulos carregados pelo `slapd` em seu início. Crie o arquivo novo `/root/ldif/olcModuleLoad.ldif` com o seguinte conteúdo:

```
1 dn: cn=module{0},cn=config
2 changetype: modify
3 add: olcModuleLoad
4 olcModuleLoad: ppolicy.la
```

   Para aplicar as modificações desse LDIF à base LDAP, basta executar:

```
# ldapmodify -Y EXTERNAL -H ldapi:/// -f ~/ldif/olcModuleLoad.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
modifying entry "cn=module{0},cn=config"
```

3. A próxima etapa é configurar o *overlay* de políticas de senhas para controlar os atributos `userPassword` de nossa base `cn=intnet`. Crie o arquivo novo `/root/ldif/olcOverlayPpolicy.ldif` com o seguinte conteúdo:

```
1 dn: olcOverlay=ppolicy,olcDatabase={1}mdb,cn=config
2 objectClass: olcOverlayConfig
3 objectClass: olcPPolicyConfig
4 olcOverlay: ppolicy
5 olcPPolicyDefault: cn=passwordDefault,ou=Policies,dc=intnet
6 olcPPolicyHashCleartext: FALSE
7 olcPPolicyUseLockout: FALSE
8 olcPPolicyForwardUpdates: FALSE
```

   Note que estamos indicando que o *overlay* `ppolicy` será aplicado sobre a base `{1}mdb`, que é exatamente a base com raiz em `dc=intnet`, como podemos confirmar através do comando:

```
# ldapsearch -Y external -H ldapi:/// -LLL -b 'cn=config'
'(&(objectClass=olcDatabaseConfig)(olcSuffix=dc=intnet))' dn 2> /dev/null
dn: olcDatabase={1}mdb,cn=config
```

   Caso estivéssemos fazendo esta configuração em um ambiente que possua várias bases LDAP

carregadas dentro de um mesmo *daemon* `slapd`, seria necessário determinar o número da base MDB e editar o arquivo mostrado anteriormente.

Para aplicar as modificações desse LDIF à base LDAP, execute:

```
# ldapadd -Y EXTERNAL -H ldapi:/// -f ~/ldif/olcOverlayPpolicy.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
adding new entry "olcOverlay=ppolicy,olcDatabase={1}mdb,cn=config"
```

4. Agora, vamos definir a política de senhas da base `dc=intnet`. Crie o arquivo novo `/root/ldif/passwordDefault.ldif` com o seguinte conteúdo:

```
dn: ou=Policies,dc=intnet
ou: Policies
objectClass: organizationalUnit

dn: cn=passwordDefault,ou=Policies,dc=intnet
objectClass: pwdPolicy
objectClass: person
objectClass: top
cn: passwordDefault
sn: passwordDefault
pwdAttribute: userPassword
pwdCheckQuality: 2
pwdMinAge: 0
pwdMaxAge: 2592000
pwdMinLength: 8
pwdInHistory: 5
pwdMaxFailure: 3
pwdFailureCountInterval: 0
pwdLockout: TRUE
pwdLockoutDuration: 0
pwdAllowUserChange: TRUE
pwdExpireWarning: 0
pwdGraceAuthNLimit: 0
pwdMustChange: FALSE
pwdSafeModify: FALSE
```

Estamos, em ordem:

- Criando uma entrada `ou=Policies,dc=intnet` para armazenar políticas da base `dc=intnet`.
- Dentro desta OU, criando o CN `cn=passwordDefault,ou=Policies,dc=intnet` que define a política de senhas da base. Configurações mais relevantes:
  - `pwdAttribute` define o atributo que será verificado, que armazena senhas de usuários.
  - `pwdCheckQuality` ativa a checagem de qualidade de senhas; como não estamos

habilitando nenhum módulo externo, apenas a checagem de comprimento será aplicada.

- `pwdMinAge` define o tempo mínimo de validade de senhas; como queremos testar o histórico de senhas, explicado a seguir, não iremos ativar essa opção.
- `pwdMaxAge` define o tempo máximo de validade da senha, em segundos; ajustamos esse valor para 30 dias.
- `pwdMinLength` define o tamanho mínimo de senha, 8 caracteres.
- `pwdInHistory` define que iremos guardar o *hash* das 5 senhas mais recentes de cada usuário, que não poderão repeti-las.
- `pwdMaxFailure` define que usuários que errarem a senha consecutivamente mais de 3 vezes terão suas contas bloqueadas.

Para aplicar o LDIF à base LDAP temos que nos autenticar na raiz `dc=intnet`, como se segue:

```
# ldapadd -D 'cn=admin,dc=intnet' -W -f ~/ldif/passwordDefault.ldif
Enter LDAP Password:
adding new entry "ou=Policies,dc=intnet"

adding new entry "cn=passwordDefault,ou=Policies,dc=intnet"
```

5. Reinicie o `slapd` para aplicar as configurações:

```
# systemctl restart slapd.service
```

6. Vamos testar nossos controles — logue na máquina `client` como o usuário `luke`:

```
$ hostname ; whoami
client
luke
```

Tente alterar a senha do usuário para uma *string* menor que o tamanho exigido, como `marte` por exemplo:

```
$ passwd
(current) LDAP Password:
Nova senha:
Redigite a nova senha:
password change failed: Password fails quality checking policy
passwd : Erro de manipulação de token de autenticação
passwd: senha inalterada
```

O `slapd` nos informa que a senha não atende os requisitos mínimos de qualidade, nesse caso, o tamanho da senha.

7. Altere a senha para um valor aceitável, como `seg10luke2`, por exemplo:

```
$ passwd
(current) LDAP Password:
Nova senha:
Redigite a nova senha:
passwd: senha atualizada com sucesso
```

Agora, tente alterar a senha para um valor já usado anteriormente, como `seg10luke`:

```
$ passwd
(current) LDAP Password:
Nova senha:
Redigite a nova senha:
password change failed: Password is in history of old passwords
passwd : Erro de manipulação de token de autenticação
passwd: senha inalterada
```

Somos informados que a senha consta do histórico de senhas antigas. Como o LDAP implementa isso? Acesse a máquina `ns2` como usuário `root` e pesquise pelo campo `pwdHistory` do usuário `luke`:

```
# hostname ; whoami
ns2
root
```

```
# ldapsearch -LLL -D 'cn=admin,dc=intnet' -W 'uid=luke' pwdHistory
Enter LDAP Password:
dn: uid=luke,ou=People,dc=intnet
pwdHistory: 20181114030204Z#1.3.6.1.4.1.1466.115.121.1.40#38#{SSHA}bI0fgkHR6MZ
 tAavv1bUYx9PuPucZhDyY
```

Ao informarmos uma nova senha, o `slapd` compara o seu hash com um dos *hashes* guardados no histórico do usuário (nesse caso, `luke`); se encontrada, a senha é rejeitada.

8. Vamos testar o *lockout* de contas. Como teremos que fazer logins propositalmente incorretos, ainda como o usuário `root` na máquina `ns2`, pare o serviço Fail2ban para evitar que sejamos bloqueados pelo firewall durante o teste:

```
# systemctl stop fail2ban
```

De volta à máquina `client` como `luke`, tente logar via SSH na máquina `ns2` usando senha e erre propositalmente a combinação por 3 vezes consecutivas:

```
$ hostname ; whoami
client
luke
```

```
$ ssh -o PreferredAuthentications=keyboard-interactive,password -o
PubkeyAuthentication=no luke@ns2
luke@ns2's password:
Permission denied, please try again.
luke@ns2's password:
Permission denied, please try again.
luke@ns2's password:
Permission denied (publickey,password).
```

Agora, tente logar com a senha correta — note que seu acesso será negado:

```
$ ssh -o PreferredAuthentications=keyboard-interactive,password -o
PubkeyAuthentication=no luke@ns2
luke@ns2's password:
Permission denied, please try again.
```

De volta à máquina `ns2` como o usuário `root`, vamos verificar o que aconteceu:

```
# hostname ; whoami
ns2
root
```

Execute o comando `ldapsearch` abaixo para listar todos os usuários bloqueados na base `dc=intnet`:

```
# ldapsearch -LLL -D 'cn=admin,dc=intnet' -W 'pwdAccountLockedTime=*'
pwdAccountLockedTime
Enter LDAP Password:
dn: uid=luke,ou=People,dc=intnet
pwdAccountLockedTime: 20181114030659Z
```

Como esperado, `luke` está bloqueado. Para desbloquear um usuário específico crie um arquivo LDIF novo, `/root/ldif/unlockUser.ldif` com o seguinte conteúdo:

```
1 dn: uid=luke,ou=People,dc=intnet
2 changetype: modify
3 delete: pwdAccountLockedTime
```

Aplique as alterações do LDIF à base com:

```
# ldapmodify -D 'cn=admin,dc=intnet' -W -f ~/ldif/unlockUser.ldif
Enter LDAP Password:
modifying entry "uid=luke,ou=People,dc=intnet"
```

De volta à máquina `client` como `luke`, tente logar novamente com a senha correta:

```
$ hostname ; whoami
client
luke
```

```
$ ssh -o PreferredAuthentications=keyboard-interactive,password -o
PubkeyAuthentication=no luke@ns2
luke@ns2's password:
Linux ns2 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

Last login: Wed Nov 14 00:15:11 2018 from 192.168.42.2
luke@ns2:~$
```

```
$ hostname ; whoami
ns2
luke
```

Perfeito, nossos controles funcionaram como esperado. Na máquina `ns2`, como `root`, não se esqueça de reiniciar o Fail2ban:

```
# hostname ; whoami
ns2
root
```

```
# systemctl start fail2ban
```

3) Busca de senhas fracas

Simplesmente configurar um tamanho mínimo de senha, como fizemos na atividade anterior, não é garantia que os usuários escolherão senhas seguras para suas contas. Por exemplo, um usuário pode definir `12345678` como sua senha — essa *string* está dentro do tamanho mínimo exigido mas não pode, nem de perto, ser considerada uma senha segura. O que fazer?

Podemos submeter os *hashes* de senha dos usuários a testes de segurança, como ataques de força-bruta — em que testamos combinações de caracteres exaustivamente para descobrir a senha — ou de dicionário — em que usamos uma base de senhas previamente preechida, conhecida como *wordlist*, e verificamos se a senha do usuário se encontra nessa lista. Devido ao fato de as senhas do

LDAP serem armazenadas por padrão em formado SSHA (SHA-1 com *salt*), ataques do tipo *rainbow table* — em que comparamos o *hash* da senha do usuário com uma base de *hashes* previamente computados, buscando por similaridades — não são viáveis. Podemos verificar o *hash* utilizado para armazenar a senha do usuário `luke`, por exemplo, usando o comando:

```
# ldapsearch -x -LLL -D 'cn=admin,dc=intnet' -W 'uid=luke' userPassword | grep
'^userPassword::' | awk '{print $NF}' | base64 --decode
Enter LDAP Password:
{SSHA}JK1/uM/9bmoWM/IzW1uIBM4b1Q4UEWd8
```

A ferramenta que iremos utilizar para realizar os ataques de dicionário e força-bruta mencionados anteriormente será o `hashcat` (https://hashcat.net/hashcat/). Uma das ferramentas mais rápidas para quebra de senhas disponíveis, é um programa *open source* multiplataforma que se utiliza da CPU ou GPUs (placas gráficas) de uma máquina para acelerar o processo de ataque sensivelmente, especialmente quando comparada com ferramentas mais tradicionais como o `john`.

Até a versão v3.00, o `hashcat` era dividido em duas versões, uma voltada para CPUs e outra para GPUs (esta, implementada via OpenCL ou CUDA). Com o lançamento da versão v3.00, as duas versões foram unificadas em uma única ferramenta, requerendo a biblioteca OpenCL (https://www.khronos.org/opencl/) como dependência.

É boa prática de segurança que instalemos apenas o estritamente necessário em servidores, a fim de reduzir a superfície de ataque disponível em uma eventual invasão. Por esse motivo, instalaremos o `hashcat` e as demais bibliotecas necessárias na máquina `client`, que é menos crítica que os servidores `ns2` e `ns1`.

1. Acesse a máquina `client` como o usuário `root`, e instale o `hashcat` e suas dependências:

   ```
   # hostname ; whoami
   client
   root
   ```

   ```
   # apt-get install --no-install-recommends hashcat libhwloc-dev ocl-icd-dev ocl-icd-
   opencl-dev pocl-opencl-icd
   ```

2. Agora, acesse como o usuário `luke`, em seu diretório *home*.

   ```
   $ hostname ; whoami ; pwd
   client
   luke
   /home/luke
   ```

   Altere a senha do usuário `luke` para um valor propositalmente inseguro, como `password`:

```
$ passwd
(current) LDAP Password:
Nova senha:
Redigite a nova senha:
passwd: senha atualizada com sucesso
```

O primeiro passo para testarmos a segurança das senhas dos usuários é obter seus *hashes*. Vamos fazer isso, de forma remota, usando um *script* shell mostrado a seguir. Crie o arquivo novo `/home/luke/scripts/gethashes.sh` com o seguinte conteúdo:

```
1  #!/bin/bash
2
3  TMPFILE="$( mktemp )"
4  OUTFILE="${HOME}/hashes.txt"
5
6  rm -f ${OUTFILE}
7  touch ${OUTFILE}
8
9  ldapsearch -x                    \
10   -LLL                           \
11   -H ldap://10.0.42.2            \
12   -D 'cn=admin,dc=intnet'        \
13   -w 'rnpesr'                    \
14   -b 'dc=intnet'                 \
15   'userPassword=*'               \
16   cn userPassword                \
17   | grep '^cn:\|^userPassword::' \
18   | awk '{print $NF}'            \
19   | sed 'N;s/\n/ /'              \
20   | tr ' ' ':' > ${TMPFILE}
21
22 while read l; do
23   luser="$( echo ${l} | cut -d':' -f1 )"
24   lhash="$( echo ${l} | cut -d':' -f2 )"
25
26   echo "${luser}:$( echo ${lhash} | base64 --decode )" >> ${OUTFILE}
27 done < ${TMPFILE}
28
29 rm -f ${TMPFILE}
```

O que esse *script* faz? Vamos ver:

1. (Linhas 3-4) Criamos um arquivo temporário com o comando `mktemp`, e definimos o arquivo de saída como `~/hashes.txt`.
2. (Linhas 6-7) Se existente, removemos o arquivo de saída e criamos um novo, vazio.
3. (Linhas 9-20) Executamos um comando `ldapsearch` remoto na máquina `ns2`, executando o *bind* como o usuário `cn=admin,dc=intnet` e senha informada diretamente na linha de

comando. Buscamos todos os DNs que possuem o campo `userPassword` não-vazio, e filtramos apenas os campos `cn` e `userPassword` na saída. Finalmente, fazemos uma junção de linhas duas-a-duas usando os comandos `awk`, `sed` e inserimos um separador usando o `tr`. Essa saída é escrita no arquivo temporário criado anteriormente.

4. (Linhas 22-27) Processamos o arquivo temporário linha-a-linha. Em cada linha, extraímos o campo 1 (`cn` do usuário) e o campo 2 (`userPassword`). O campo `userPassword` está codificado em base64, então usamos `base64 --decode` para traduzir esse campo, e escrevemos o *output* em ordem no arquivo de saída.
5. (Linha 29) O arquivo temporário é removido.

3. Vamos testar o funcionamento do *script*:

```
$ bash ~/scripts/gethashes.sh
```

```
$ cat hashes.txt
admin:{SSHA}NzQZTz7ufOxNM3PYy7cp+zV6p7bKFNcy
luke:{SSHA}46Qe8Ny+QQgDsbPcps2MODqUHGtdLX41
sshca:{SSHA}+JTtQ5+XEi+sJ4sPmWK3lZXrIHSpbcbn
han:{SSHA}BE6cC89vaJQtB/g9yEJTt0O8HCtRabel
```

4. Vamos, primeiramente, executar um ataque de dicionário. Um ataque de dicionário, como mencionado anteriormente, é quando obtermos um arquivo com um conjunto de senhas em texto claro, calculamos seus *hashes* usando os valores de *salt* conhecidos, e comparamos os resultados com os *hashes* dos usuários.

   No caso do algorito SSHA implementado no OpenLDAP, para extrair o *salt* devemos decodificar o *hash* original em base64 uma vez, remover o prefixo `{SSHA}`, decodificar o *hash* resultante em base64 novamente, e extrair os últimos 4 bytes; esses 4 bytes são o *salt* da senha codificada. Para ilustrar esse conceito, o *script* Perl a seguir pode ser usado para fazer a extração — crie o arquivo novo `/home/luke/scripts/getsalt.pl` com o seguinte conteúdo:

```
#!/usr/bin/perl -w

my $hash=$ARGV[0];
# The hash is encoded as base64 twice:
use MIME::Base64;
$hash = decode_base64($hash);
$hash=~s/{SSHA}//;
$hash = decode_base64($hash);

# The salt length is four (the last four bytes).
$salt = substr($hash, -4);

# Split the salt into an array.
my @bytes = split(//,$salt);

# Convert each byte from binary to a human readable hexadecimal number.
foreach my $byte (@bytes) {
$byte = uc(unpack "H*", $byte);
print "$byte";
}
```

Vamos recuperar o *hash* de senha do usuário `luke`:

```
$ ldapsearch -x -LLL -H ldap://ldap.intnet/ -D 'cn=admin,dc=intnet' -b 'dc=intnet'
-w 'rnpesr' 'uid=luke' userPassword | grep '^userPassword::' | awk '{print $NF}'
e1NTSEF9anFkZC80MW1BT3dFYVpkMFpvWUZzYk1xQTJyVlVMVlU=
```

Executando o *script* `getsalt.pl`, podemos extrair o *salt* da senha. Note que o valor de saída está em hexadecimal.

```
$ perl ~/scripts/getsalt.pl e1NTSEF9anFkZC80MW1BT3dFYVpkMFpvWUZzYk1xQTJyVlVMVlU= ;
echo
D550B554
```

5. De volta ao ataque de dicionário, vamos executá-lo usando o `hashcat`. Primeiro, temos que descobrir a qual código o *hash* SSHA do LDAP corresponde:

```
$ hashcat --help | grep SSHA
    111 | nsldaps, SSHA-1(Base64), Netscape LDAP SSHA      | HTTP, SMTP, LDAP
Server
   1711 | SSHA-512(Base64), LDAP {SSHA512}                 | HTTP, SMTP, LDAP
Server
  10300 | SAP CODVN H (PWDSALTEDHASH) iSSHA-1               | Enterprise Application
Software (EAS)
```

O código é, então, `111`. Quanto ao tipo de ataque:

```
$ hashcat --help | grep 'Attack Modes' -A8
- [ Attack Modes ] -

  # | Mode
 ===+======
  0 | Straight
  1 | Combination
  3 | Brute-force
  6 | Hybrid Wordlist + Mask
  7 | Hybrid Mask + Wordlist
```

O ataque de dicionário, também conhecido como *straight mode* (https://hashcat.net/wiki/doku.php?id=dictionary_attack), possui código `0`.

Falta apenas obter uma *wordlist* apropriada para executar o ataque. Procurando por termos como *"wordlist"*, *"password"* ou *"common"* no Google, é possível encontrar uma infinidade de páginas web dedicadas ao assunto, como por exemplo https://github.com/danielmiessler/SecLists/tree/master/Passwords . Iremos usar uma *wordlist* que alegadamente contém as 10 milhões de senhas mais comuns, que pode ser baixada na URL anteriormente mencionada ou solicitada ao instrutor. Note que, para um arquivo que contém apenas texto puro, seu tamanho é impressionante:

```
$ wget -q https://github.com/danielmiessler/SecLists/raw/master/Passwords/Common-
Credentials/10-million-password-list-top-1000000.txt
```

```
$ du -sh 10-million-password-list-top-1000000.txt
8,2M    10-million-password-list-top-1000000.txt
```

Tudo pronto! Vamos executar o ataque:

```
$ hashcat --force --hash-type 111 --attack-mode 0 --username hashes.txt 10-million-
password-list-top-1000000.txt
hashcat (v3.30) starting...

(...)

Session..........: hashcat
Status...........: Exhausted
Hash.Type........: SSHA-1(Base64), nsldaps, Netscape LDAP SSHA
Hash.Target......: hashes.txt
Time.Started.....: Wed Nov 14 01:16:49 2018 (1 sec)
Time.Estimated...: Wed Nov 14 01:16:50 2018 (0 secs)
Input.Base.......: File (10-million-password-list-top-1000000.txt)
Input.Queue......: 1/1 (100.00%)
Speed.Dev.#1.....:  2591.6 kH/s (0.36ms)
Recovered........: 1/4 (25.00%) Digests, 1/4 (25.00%) Salts
Progress.........: 3999996/3999996 (100.00%)
Rejected.........: 36/3999996 (0.00%)
Restore.Point....: 999999/999999 (100.00%)
Candidates.#1....: vjq445 -> vjht008
HWMon.Dev.#1.....: N/A

Started: Wed Nov 14 01:16:45 2018
Stopped: Wed Nov 14 01:16:51 2018
```

Na máquina usada como exemplo (a velocidade pode variar de acordo com a velocidade da CPU/GPU disponível), o ataque aos quatro *hashes* disponíveis usando 10 milhões de senhas demorou... 9 segundos. Como visualizado em `Speed.Dev.#1`, a velocidade de tentativas foi de 2591 kilo-*hashes* por segundo ou, em outras palavras, 2591000 *hashes* por segundo. Foi descoberto um *digest*, que podemos visualizar emitindo o mesmo comando com a *flag* `--show`:

```
$ hashcat --force --hash-type 111 --attack-mode 0 --username hashes.txt 10-million-
password-list-top-1000000.txt --show
luke:{SSHA}jqdd/41mAOwEaZd0ZoYFsbMqA2rVULVU:password
```

Excelente! Como era de se esperar, a senha fraca `password` do usuário `luke` foi descoberta usando o ataque de dicionário.

6. Mas, e as demais senhas? O usuário `han` e `sshca` possuem senhas relativamente mais complexas, mas sabemos que a senha do usuário `admin` é simples, `rnpesr`. Como essa *string* não consta do arquivo com 10 milhões de senhas usado no ataque anterior, ela não foi descoberta, no entanto.

   Vamos executar um ataque de força-bruta contra essa senha. Para isso, alteraremos o modo de ataque do `hashcat` para `3`, e definiremos uma máscara igual a `?l?l?l?l?l?l` — senhas de até seis caracteres, apenas com caracteres de `[a-z]` minúsculos. Para aprender mais sobre a sintaxe de máscaras suportadas pelo `hashcat`, consulte sua página de manual ou https://hashcat.net/wiki/doku.php?id=mask_attack .

Ao trabalho:

```
$ hashcat --force --hash-type 111 --attack-mode 3 --username hashes.txt
?l?l?l?l?l?l
hashcat (v3.30) starting...

(...)

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit => s
```

Após a inicialização, a linha acima será mostrada. Podemos apertar os atalhos destacados entre colchetes para instruir o `hashcat` com ações durante o ataque. Apertando `s`, visualizamos o estado atual do ataque:

```
Session..........: hashcat
Status...........: Running
Hash.Type........: SSHA-1(Base64), nsldaps, Netscape LDAP SSHA
Hash.Target......: hashes.txt
Time.Started.....: Wed Nov 14 01:18:02 2018 (7 secs)
Time.Estimated...: Wed Nov 14 01:18:34 2018 (25 secs)
Input.Mask.......: ?l?l?l?l?l?l [6]
Input.Queue......: 1/1 (100.00%)
Speed.Dev.#1.....: 27956.2 kH/s (6.13ms)
Recovered........: 1/4 (25.00%) Digests, 1/4 (25.00%) Salts
Progress.........: 268409856/1235663104 (21.72%)
Rejected.........: 0/268409856 (0.00%)
Restore.Point....: 99072/456976 (21.68%)
Candidates.#1....: saufph -> xqoxjk
HWMon.Dev.#1.....: N/A
```

Observando a linha `Progress`, notamos que o ataque está 22,83% concluído. Aguardamos.

```
{SSHA}AZv4/v1spKXTHw5GOK1y+vQhPrnb7CzA:rnpesr
```

Após algum tempo, a linha acima é mostrada na tela. O `hashcat` conseguiu quebrar a senha do usuário `admin`, descobrindo-a como sendo `rnpesr`. Aguardamos a conclusão do processo.

```
Session..........: hashcat
Status...........: Exhausted
Hash.Type........: SSHA-1(Base64), nsldaps, Netscape LDAP SSHA
Hash.Target......: hashes.txt
Time.Started.....: Wed Nov 14 01:18:02 2018 (30 secs)
Time.Estimated...: Wed Nov 14 01:18:32 2018 (0 secs)
Input.Mask.......: ?l?l?l?l?l?l [6]
Input.Queue......: 1/1 (100.00%)
Speed.Dev.#1.....: 27362.0 kH/s (6.14ms)
Recovered........: 2/4 (50.00%) Digests, 2/4 (50.00%) Salts
Progress.........: 1235663104/1235663104 (100.00%)
Rejected.........: 0/1235663104 (0.00%)
Restore.Point....: 456976/456976 (100.00%)
Candidates.#1....: sacxqg -> xqqfqg
HWMon.Dev.#1.....: N/A

Started: Wed Nov 14 01:18:00 2018
Stopped: Wed Nov 14 01:18:33 2018
```

Depois de 33 segundos, o ataque encontra-se 100% concluído. Um novo *digest* foi descoberto, como podemos visualizar com a *flag* `--show`:

```
$ hashcat --force --hash-type 111 --attack-mode 3 --username hashes.txt
?l?l?l?l?l?l --show
luke:{SSHA}jqdd/41mAOwEaZd0ZoYFsbMqA2rVULVU:password
admin:{SSHA}AZv4/v1spKXTHw5GOK1y+vQhPrnb7CzA:rnpesr
```

O `hashcat` reporta não somente a senha descoberta do usuário `admin`, bem como a senha do usuário `luke` descoberta na execução anterior. Isso ocorre porque o `hashcat` mantém o histórico de ataques realizados no diretório `~/.hashcat`:

```
$ ls -1 ~/.hashcat/
hashcat.dictstat
hashcat.potfile
kernels
sessions
```

E assim, concluímos nossa busca por senhas fracas, via ataques de dicionário e força-bruta. O próximo passo, naturalmente, seria alterar o valor de senha dos usuários para um valor novo (bloqueando seu acesso), informá-los da nova senha e comunicar que devem alterar sua senha para uma combinação segura assim que possível.

4) Criação da VM do servidor de arquivos NFS

Iremos implementar, agora, um servidor de arquivos simples para que os colaboradores da Intranet possam compartilhar arquivos e ter uma opção de backup emergencial para suas estações

de trabalho. Como o ambiente que estamos simulando é inteiramente baseado em Linux, não há a necessidade de configurar uma solução interoperável com outros sistemas operacionais, como o Samba. Por isso, utilizaremos o NFS (*Network File System*), que é significativamente mais fácil de ser implementado.

1. O primeiro passo, assim como fizemos antes, é clonar a máquina `debian-template` e criar uma nova, que chamaremos de `nfs`. Essa máquina estará conectada a uma única rede *host-only*, com o mesmo nome que foi alocado para a interface de rede da máquina virtual `ns1`, configurada durante a sessão 2, que está conectada à DMZ. O IP da máquina será 10.0.42.3/24.

   Concluída a clonagem, ligue a máquina e faça login como o usuário `root`. Em seguida, use o script `/root/scripts/changehost.sh` para fazer a configuração automática:

```
# hostname ; whoami
debian-template
root
```

```
# bash ~/scripts/changehost.sh -h nfs -i 10.0.42.3 -g 10.0.42.1
Signing ssh_host_ecdsa_key.pub key...
Signing ssh_host_ed25519_key.pub key...
Signing ssh_host_rsa_key.pub key...
Configuring host key trust...
Configuring user key trust...
All done!
```

```
$ ip addr show label 'enp0s*' | grep 'inet ' | awk '{print $2,$NF}' ; hostname ;
whoami
10.0.42.3/24 enp0s3
nfs
root
```

5) Ajuste das regras de firewall

1. Se você tentar acessar a máquina `nfs` recém-criada usando o SSH, a partir da máquina `client`, rapidamente perceberá que não é possível fazer o acesso:

```
$ hostname ; whoami
client
luke
```

```
$ ssh nfs
ssh: connect to host nfs port 22: Connection timed out
```

Qual a razão disso? Após breve reflexão, fica claro que o problema reside no firewall. Vamos revisá-lo.

2. Logue na máquina `ns1` como o usuário `root`, como de costume. Tendo em vista que o acesso que desejamos fazer **passa pelo** firewall, devemos checar a *chain* FORWARD:

```
# hostname ; whoami
ns1
root
```

```
# iptables -L FORWARD -vn
Chain FORWARD (policy DROP 171 packets, 12318 bytes)
 pkts bytes target     prot opt in     out     source               destination
31432   74M ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
state RELATED,ESTABLISHED
    3   180 ACCEPT     tcp  --  *      enp0s3  10.0.42.0/24         0.0.0.0/0
multiport dports 80,443
    3   180 ACCEPT     tcp  --  *      enp0s3  192.168.42.0/24      0.0.0.0/0
multiport dports 80,443
    2   148 ACCEPT     udp  --  *      *       192.168.42.0/24      10.0.42.2
udp dpt:53
   68  4080 ACCEPT     tcp  --  *      *       192.168.42.0/24      10.0.42.2
multiport dports 22,389
```

Observe que liberamos o acesso SSH a partir da Intranet apenas para a máquina `ns2`, e nenhuma outra. Considerando que além da máquina `nfs` criaremos ainda outros servidores em nosso *datacenter* simulado, essa regra obviamente está inadequada. Vamos removê-la:

```
# iptables -D FORWARD -s 192.168.42.0/24 -d 10.0.42.2/32 -p tcp -m multiport
--dports 22,389 -j ACCEPT
```

Em seu lugar, vamos criar duas regras: uma que reautoriza o acesso da Intranet ao servidor LDAP (que estava embutido na regra antiga), e outra que autoriza a Intranet a realizar SSH para qualquer máquina da DMZ:

```
# iptables -A FORWARD -s 192.168.42.0/24 -d 10.0.42.2/32 -p tcp -m tcp --dport 389
-j ACCEPT
```

```
# iptables -A FORWARD -s 192.168.42.0/24 -d 10.0.42.0/24 -p tcp -m tcp --dport 22
-j ACCEPT
```

3. Ainda temos que tratar do acesso da Intranet ao serviço NFS, que configuraremos na atividade a seguir. Para nossa sorte, a configuração do firewall para o NFS versão 4 (que utilizaremos neste curso) é significativamente mais fácil que as versões anteriores, bastando liberar o tráfego dos

clientes para a porta 2049/TCP do servidor remoto (em nosso caso, a máquina `nfs`). A regra a seguir irá atender este requisito:

```
# iptables -A FORWARD -s 192.168.42.0/24 -d 10.0.42.3/24 -p tcp -m tcp --dport 2049
-j ACCEPT
```

4. Finalmente, salve as regras de firewall atuais:

```
# /etc/init.d/netfilter-persistent save
[....] Saving netfilter rules...run-parts: executing /usr/share/netfilter-
persistent/plugins.d/15-ip4tables save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
done.
```

6) Configuração do servidor de arquivos NFS e quotas de disco

1. Queremos usar o servidor NFS para armazenar arquivos de usuários da Intranet, para compartilhamento e backup. Isso imediatamente suscita uma grande preocupação: imagine um cenário em que usuários querem armazenar muitos arquivos no servidor (de forma acidental ou maliciosa) — isso pode atrapalhar o uso de outros colaboradores, ou até mesmo chegar a encher a partição raiz (`/`) do sistema, causando instabilidade. Para evitar esse cenário, vamos criar uma partição dedicada para arquivos de usuário no diretório `/home` do servidor `nfs`, e também aplicar *quotas* de disco aos usuários.

   Desligue a máquina `nfs` e adicione a ela um novo disco de 10 GB, usando a interface do Virtualbox. A seguir, formate o disco e adicione-o ao sistema LVM, criando um novo *volume group* `vg-home` com um único volume lógico `lv-home`, de forma análoga ao que fizemos na atividades (7) e (8) da sessão (1). Finalmente, formate esse volume em `ext4` e ative sua montagem automática durante o *boot* da máquina `nfs` no diretório `/home` com as opções `defaults,nosuid,nodev`.

   Vamos ao trabalho. Após desligar a VM e adicionar o disco de 10 GB, acessamos a máquina `nfs` como o usuário `root`:

```
# hostname ; whoami
nfs
root
```

   O próximo passo é descobrir sob qual nome o disco foi detectado. Nesse caso, temos a vantagem de saber que o tamanho do disco novo, 10 GB, é diferente do disco preexistente.

```
# dmesg | grep 'GiB'
[    1.585018] sd 1:0:0:0: [sdb] 20971520 512-byte logical blocks: (10.7 GB/10.0
GiB)
[    1.585187] sd 0:0:0:0: [sda] 16777216 512-byte logical blocks: (8.59 GB/8.00
GiB)
```

Evidentemente, o disco `/dev/sdb` é o que acabamos de adicionar, portanto. Vamos formatá-lo:

```
# fdisk /dev/sdb

Bem-vindo ao fdisk (util-linux 2.29.2).
As alterações permanecerão apenas na memória, até que você decida gravá-las.
Tenha cuidado antes de usar o comando de gravação.

A unidade não contém uma tabela de partição conhecida.
Criado um novo rótulo de disco DOS com o identificador de disco 0x3bc30929.
```

```
Comando (m para ajuda): o
Criado um novo rótulo de disco DOS com o identificador de disco 0x8a16601c.
```

```
Comando (m para ajuda): n
Tipo da partição
   p   primária (0 primárias, 0 estendidas, 4 livre)
   e   estendida (recipiente para partições lógicas)

Selecione (padrão p):

Usando resposta padrão p.
Número da partição (1-4, padrão 1):
Primeiro setor (2048-20971519, padrão 2048):
Último setor, +setores ou +tamanho{K,M,G,T,P} (2048-20971519, padrão 20971519):

Criada uma nova partição 1 do tipo "Linux" e de tamanho 10 GiB.
```

```
Comando (m para ajuda): t
Selecionou a partição 1
Tipo de partição (digite L para listar todos os tipos): 8e
O tipo da partição "Linux" foi alterado para "Linux LVM".
```

```
Comando (m para ajuda): w
A tabela de partição foi alterada.
Chamando ioctl() para reler tabela de partição.
Sincronizando discos.
```

Agora, vamos criar o volume físico:

```
# pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created.
```

O grupo de volumes:

```
# vgcreate vg-home /dev/sdb1
  Volume group "vg-home" successfully created
```

E, finalmente, o volume lógico:

```
# lvcreate -l +100%FREE -n lv-home vg-home
  Logical volume "lv-home" created.
```

Vamos, agora, formatar o sistema de arquivos do LV:

```
# mkfs.ext4 /dev/mapper/vg--home-lv--home
mke2fs 1.43.4 (31-Jan-2017)
Creating filesystem with 2620416 4k blocks and 655360 inodes
Filesystem UUID: be99743c-7ca0-4144-8548-d7aab33a878b
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

A seguir, sincronizar os arquivos do diretório `/home` atual com o LV recém-criado:

```
# mount /dev/mapper/vg--home-lv--home /mnt/
```

```
# rsync -av /home/ /mnt/
sending incremental file list
./
aluno/
aluno/.bash_history
aluno/.bash_logout
aluno/.bashrc
aluno/.profile
aluno/.vimrc
luke/
luke/.bash_history
luke/.bash_logout
luke/.bashrc
luke/.profile
luke/scripts/
luke/scripts/sshsign_user.sh

sent 11,669 bytes  received 225 bytes  23,788.00 bytes/sec
total size is 10,787  speedup is 0.91
```

```
# umount /mnt
```

E, finalmente, configurar a montagem no `/etc/fstab` e montar o LV:

```
# nano /etc/fstab
(...)
```

```
# tail -n1 /etc/fstab
/dev/mapper/vg--home-lv--home /home ext4  defaults,nosuid,nodev         0  2
```

```
# mount -a
```

```
# df -h | sed -n '1p; /\/home/p'
Sist. Arq.                     Tam. Usado Disp. Uso% Montado em
/dev/mapper/vg--home-lv--home  9,8G   37M  9,3G   1% /home
```

2. Agora, vamos instalar os pacotes para habilitar o compartilhamento de arquivos via NFS e *quotas* de disco. Como `root`, instale os pacotes `nfs-kernel-server`, `quota` e `quotatool`:

```
# apt-get install nfs-kernel-server quota quotatool
```

3. Vamos configurar primeiro o sistema de *quotas*. Edite a entrada do diretório `/home` no arquivo

`/etc/fstab` e adicione as opções `usrquota,grpquota`, que ativam suporte a *quotas* por usuário e por grupo no sistema de arquivos.

```
# nano /etc/fstab
(...)
```

```
# tail -n1 /etc/fstab
/dev/mapper/vg--home-lv--home /home ext4  defaults,nosuid,nodev,usrquota,grpquota
0  2
```

Em seguida, remonte o sistema de arquivos e verifique se o sistema de *quotas* foi habilitado na partição.

```
# mount -o remount /home
```

```
# mount | grep '/home'
/dev/mapper/vg--home-lv--home on /home type ext4
(rw,nosuid,nodev,relatime,quota,usrquota,grpquota,data=ordered)
```

Crie os arquivos de configuração de *quotas* usando o comando `quotacheck`:

```
# quotacheck -ugc /home/
```

Pronto, o sistema de *quotas* está configurado. Iremos editar *quotas* de usuário e testar seu funcionamento mais à frente, após a configuração do NFS.

4. O próximo passo é configurar o serviço NFS — lembre-se que queremos disponibilizar compartilhamentos para arquivos de usuário, no diretório `/home` do sistema. Insira a linha a seguir no arquivo `/etc/exports`:

```
# echo '/home 192.168.42.0/24(rw,async,no_subtree_check,root_squash)' >>
/etc/exports
```

A linha acima irá configurar a exportação o diretório `/home` para todas as máquinas da Intranet (faixa 192.168.42.0/24), em modo leitura-escrita, assíncrono (i.e. escritas ao disco do servidor remoto não precisam ter sido efetivadas para que o cliente receba confirmação), sem checagem de sub-árvores de montagem (consultar página de manual com `man 5 exports`) e desabilitando o mapeamento de UID do `root` da máquina remota no servidor local.

Para exportar o diretório, basta executar:

```
# exportfs -a
```

Finalmente, para visualizar quais diretórios estão sendo exportados, execute:

```
# showmount -e
Export list for nfs:
/home 192.168.42.0/24
```

5. Vamos testar nosso sistema de compartilhamento de arquivos via NFS e *quotas* de disco. Acesse a máquina `client` como o usuário `root`, crie um diretório `/remote` para ser o ponto de montagem NFS e monte o diretório compartilhado.

```
# hostname ; whoami
client
root
```

```
# mkdir /remote
```

```
# mount -t nfs 10.0.42.3:/home /remote
```

```
# ls -l /remote/
total 40
drwxr-xr-x 2 aluno aluno   4096 nov 13 17:01 aluno
-rw------- 1 root  root    8192 nov 14 01:49 aquota.group
-rw------- 1 root  root    8192 nov 14 01:49 aquota.user
drwx------ 2 root  root   16384 nov 14 01:46 lost+found
drwxr-xr-x 3 luke  sysadm  4096 nov 14 01:41 luke
```

6. Um dos principais problemas em sistemas de compartilhamento de arquivos em ambientes Unix é o mapeamento de UIDs e GIDs — como garantir que os usuários de múltiplas máquinas remotas possuam os mesmos identificadores que os usuários existentes no servidor de arquivos? Felizmente, nosso sistema centralizado de autenticação usando LDAP resolve esse problema de forma transparente: todos os usuários possuem um valor de UID e GID consistente em todo o *datacenter*, já que as contas são gerenciadas em um ponto único.

   Senão, vejamos: como o usuário `luke`, tente criar um arquivo novo dentro do ponto de montagem `/remote/luke`:

```
$ hostname ; whoami
client
luke
```

```
$ echo test > /remote/luke/file
```

```
$ cat /remote/luke/file
test
```

Funcionou perfeitamente! Uma vez que os valores de UID e GID do usuário `luke` são consistentes entre a máquina `client` e o servidor de arquivos `nfs`, não temos problemas de permissão.

7. E quanto ao usuário `root`? Será que o `root` local da máquina `client` possui acesso irrestrito aos arquivos compartilhados?

```
$ su -
Senha:
root@client:~#
```

```
# hostname ; whoami
client
root
```

```
# echo test > /remote/file
-su: /remote/file: Permissão negada
```

```
# rm /remote/aquota.user
rm: não foi possível remover '/remote/aquota.user': Permissão negada
```

Devido à utilização da opção `root_squash` no compartilhamento configurado via arquivo `/etc/exports` da máquina `nfs`, o mapeamento de UID do usuário `root` em máquinas remotas é desativado, efetivamente impedindo-o de alterar quaisquer arquivos.

8. Vamos testar o sistema de *quotas*. Na máquina `nfs`, como o usuário `root`, edite as *quotas* do usuário `luke` usando o comando `edquota`:

```
# edquota -u luke
```

O comando `edquota` irá invocar um editor (indicado pela variável de ambiente `$EDITOR`) para que as *quotas* sejam ajustadas. Vamos editar os campos *soft* e *hard* da seção *block* do arquivo, ajustando limites de 100 MB e 200 MB, respectivamente — note que os valores devem ser informados em kilobytes. Pode-se, opcionalmente, também setar um limite para *inodes* que o usuário pode criar.

```
Disk quotas for user luke (uid 10000):
  Filesystem                   blocks       soft       hard     inodes     soft
hard
  /dev/mapper/vg--home-lv--home    32     100000     200000          8        0
0
```

Para verificar as *quotas* de um sistema de arquivos, use o comando `repquota`:

```
# repquota -u /home
*** Report for user quotas on device /dev/mapper/vg--home-lv--home
Block grace time: 7days; Inode grace time: 7days
                        Block limits                File limits
User            used    soft    hard  grace    used  soft  hard  grace
----------------------------------------------------------------------
root      --      20       0       0              2     0     0
aluno     --      24       0       0              6     0     0
luke      --      32  100000  200000              8     0     0
```

Para ativar as *quotas* em uma partição, utilize o comando `quotaon`:

```
# quotaon -ug /home
```

9. Acesse a máquina `client` como o usuário `luke`. Vamos tentar extrapolar o limite estabelecido pela *quota* no passo anterior.

```
$ hostname ; whoami
client
luke
```

O kernel do SO é um arquivo interessante a ser usado para esse teste, já que possui um tamanho razoável. Vamos copiá-lo sucessivas vezes para o diretório `/remote/luke` e verificar o que acontece:

```
$ du -sh /boot/vmlinuz-4.9.0-8-amd64
4,1M    /boot/vmlinuz-4.9.0-8-amd64
```

```
$ for i in {1..100}; do cp /boot/vmlinuz-4.9.0-8-amd64 /remote/luke/vmlinuz-$i;
done
cp: falha ao fechar '/remote/luke/vmlinuz-49': Disk quota exceeded
cp: falha ao fechar '/remote/luke/vmlinuz-50': Disk quota exceeded
cp: falha ao fechar '/remote/luke/vmlinuz-51': Disk quota exceeded
(...)
cp: falha ao fechar '/remote/luke/vmlinuz-98': Disk quota exceeded
cp: falha ao fechar '/remote/luke/vmlinuz-99': Disk quota exceeded
cp: falha ao fechar '/remote/luke/vmlinuz-100': Disk quota exceeded
```

Note que após 48 cópias de arquivo, o sistema reporta a *quota* de disco como excedida, e o usuário não pode mais escrever na partição. De fato, checando o estado da *quota* de disco com o comando `repquota` na máquina `nfs`, temos que:

```
# hostname ; whoami
nfs
root
```

```
# repquota -u /home/
*** Report for user quotas on device /dev/mapper/vg--home-lv--home
Block grace time: 7days; Inode grace time: 7days
                        Block limits                File limits
User            used    soft    hard  grace    used  soft  hard  grace
----------------------------------------------------------------------
root      --      20       0       0              2     0     0
aluno     --      24       0       0              6     0     0
luke      +-  200000  100000  200000  6days     108     0     0
```

Temos, portanto, que nosso esquema de *quotas* está funcionando como esperado. Não se esqueça de apagar os diversos arquivos `vmlinuz*` que criamos, para liberar espaço no disco novamente:

```
# rm /home/luke/vmlinuz-*
```

Observe que apenas os usuários `aluno` e `luke` possuem pastas no diretório `/home` compartilhado pela máquina `nfs`. Isso se deve ao fato de que apenas esses usuários haviam feito acesso local à máquina `nfs` até aquele momento — lembre-se que o arquivo de configuração `/usr/share/pam-configs/mkhomedir` que aplicamos ao PAM cria diretórios *home* apenas quando o usuário faz acesso à máquina pela primeira vez. Como consequência, o usuário `han`, para citar um exemplo, não possui uma pasta no servidor de arquivos.

Em produção, seria interessante que a pasta compartilhada do usuário fosse criada assim que este fosse adicionado à base LDAP, juntamente com o comando `ldapadduser`, por exemplo. Um *script* shell seria ideal para resolver essa situação. Claro, é possível que nem todos os novos usuários criados na base LDAP devam ter uma pasta nesse servidor, o que pode complicar sua configuração.

7) Uso de ACLs localmente

Imagine a seguinte situação, agora: o usuário `luke` quer criar um arquivo novo, sigiloso, e dar permissão para que `han` possa visualizá-lo. Ora, com as permissões padrão disponíveis em um sistema Linux, quais são nossas opções?

Sabemos que, ao criar o arquivo, o usuário-dono será `luke` e o grupo-dono será `sysadm`. Se `luke` altera o grupo-dono do arquivo para `fwadm` e `chmod` de `640`, apesar de a permissão objetivada para `han` ser garantida, todos os outros membros do grupo `fwadm` também poderão visualizar o arquivo, que não é o que queremos. Se garante-se a permissão de `644`, não só `han` como qualquer outro usuário pode visualizar o arquivo. Finalmente, a alternativa final que seria adicionar `han` ao grupo `sysadm` pode não ser desejável ou aceitável do ponto de vista administrativo. O que fazer?

O uso de ACLs (*Access Control Lists*) é especialmente adequado para esse tipo de situação, quando precisamos configurar permissões de arquivos e diretórios de forma granular. Com o uso de ACLs, é possível definir permissões customizadas para usuários e grupos diferentes dos donos do arquivo/diretório original, solucionando problemas de permissionamento para os quais o sistema tradicional de permissões Unix é inadequado.

1. Acesse a máquina `nfs` como o usuário `root`. Para consultar e ajustar ACLs localmente, basta instalar o pacote `acl`:

```
# hostname ; whoami
nfs
root
```

```
# apt-get install acl
```

2. Vamos testar o funcionamento de ACLs localmente, usando os usuários `luke` e `han`. Acesse a máquina `nfs` como o usuário `luke`:

```
$ hostname ; whoami ; pwd
nfs
luke
/home/luke
```

Agora, crie o arquivo novo `~/teste`, com qualquer conteúdo. Em seguida, consulte suas ACLs atuais.

```
$ echo oi > teste
```

```
$ getfacl teste
# file: teste
# owner: luke
# group: sysadm
user::rw-
group::r--
other::r--
```

3. Imaginemos que o arquivo criado na atividade anterior é especialmente sigiloso, devendo ser visualizado apenas pelo usuário `han` e seu dono, `luke`. Primeiro, retire as permissões do grupo e de outros:

```
$ chmod 600 ~/teste
```

Em seguida, use ACLs para dar permissão de leitura a `han`:

```
$ setfacl -m u:han:r ~/teste
```

Verifique as permissões Unix tradicionais — observe que ao final da coluna de permissionamento do `ls` vemos o caractere `+`, que indica que o arquivo possui permissões estendidas na forma de ACLs.

```
$ ls -ld /home/luke/teste
-rw-r-----+ 1 luke sysadm 3 nov  1 18:27 /home/luke/teste
```

Consulte novamente as ACLs do arquivo, verificando que a configuração desejada foi aplicada.

```
$ getfacl teste
# file: teste
# owner: luke
# group: sysadm
user::rw-
user:han:r--
group::---
mask::r--
other::---
```

4. Terá funcionado? Vamos ver. Como o usuário `aluno`, tente visualizar o conteúdo do arquivo `/home/luke/teste`:

```
$ whoami
aluno
```

```
$ cat /home/luke/teste
cat: /home/luke/teste: Permissão negada
```

E como `han`? Vamos ver:

```
$ whoami
han
```

```
$ cat /home/luke/teste
oi
```

5. Como `luke`, vamos remover a ACL de leitura do usuário `han` e testar:

```
$ whoami
luke
```

```
$ setfacl -x u:han ~/teste
```

```
$ su - han
Senha:
```

```
$ whoami
han
```

```
$ cat /home/luke/teste
cat: /home/luke/teste: Permissão negada
```

Perfeito! Lembre-se que também podemos configurar ACLs para grupos através do caractere `g`, o que não foi testado nesta atividade.

8) Uso de ACLs via NFS

A atividade anterior, apesar de interessante, é pouco prática quando consideramos nossa configuração atual: se ACLs podem apenas ser manipuladas localmente mas estamos mantendo nossos arquivos compartilhados via rede com NFS, então toda vez que um usuário quiser alterar ACLs ele terá que fazer um acesso local à máquina `nfs`? Não é razoável fazermos isso. De fato, tente fazer a alteração de ACLs a partir da máquina `client` como o usuário `luke` — lembre-se: assim como fizemos na máquina `nfs`, será necessário instalar o pacote `acl` antes de realizar este teste:

```
$ hostname ; whoami
client
luke
```

```
$ setfacl -m u:han:rw /remote/luke/teste
setfacl: /remote/luke/teste: Operação não suportada
```

Com efeito, ACLs POSIX não são suportadas diretamente via `setfacl` em *mounts* NFS.

Por outro lado, *mounts* NFS versão 4 possuem suporte a ACLs — de fato, a um conjunto de permissões ainda mais granulares e expressivas que as ACLs POSIX padrão. Mas primeiro, temos que responder à pergunta: nosso compartilhamento atual está em qual versão? Vamos ver:

```
$ mount | grep '/home' | grep -o 'vers=[0-9\.]*'
vers=4.2
```

Excelente, estamos usando a versão 4.2, o que deve ser suficiente. Os comandos para visualização e edição de ACLs NFSv4 não são os mesmos que utilizamos até agora, no entanto — vamos instalá-los.

1. Acesse a máquina `client` como o usuário `root` e instale o pacote `nfs4-acl-tools`:

```
# hostname ; whoami
client
root
```

```
# apt-get install nfs4-acl-tools
```

2. Agora sim, vamos testar o funcionamento de ACLs com a pasta compartilhada via NFS. Acesse como o usuário `luke`; para tornar o uso corriqueiro dessa pasta compartilhada mais conveniente, crie um link simbólico com o nome `remote` em seu diretório *home*.

```
$ hostname ; whoami ; pwd
client
luke
/home/luke
```

```
$ ln -s /remote/luke/ ~/remote
```

3. Consulte as ACLs NFSv4 do arquivo criado na atividade anterior:

```
$ nfs4_getfacl ~/remote/teste
A::OWNER@:rwatTcCy
A::GROUP@:tcy
A::EVERYONE@:tcy
```

O formato de representação de permissões NFSv4 é bastante diferente do que estamos acostumados — muitas opções e controles adicionais são suportados. Nesta atividade iremos trabalhar apenas com as permissões mais usuais, `rwx`, mas a página de manual `man 5 nfs4_acl` possui uma documentação bastante completa sobre as possibilidades de uso desse sistema. Em especial, a seção *ACE PERMISSIONS* é recomendada para entender o formado do *output* acima.

Como um exemplo, vamos analisar em detalhe a ACE (*Access Control Entry*) `A::OWNER@:rwatTcCy`:

- `A`: tipo da ACE; pode ser `A` (*allow*), `D` (*deny*), `U` (*audit*, usada para configurar log de acessos) e `L` (*alarm*, para gerar alarmes de sistema em caso de acesso).
- `::`: o segundo campo, neste caso vazio, define as *flags* da ACE; pode ser utilizado para indicar ACEs aplicáveis a grupos, configurações de herança da ACE para diretórios e arquivos-filho, ou *flags* administrativas para controlar eventos de log e alarme.
- `OWNER@`: define o *principal* ao qual se aplica a ACE corrente; pode ser um usuário, grupo ou uma de três ACEs especiais, `OWNER@`, `GROUP@` e `EVERYONE@`, funcionalmente equivalentes às suas contrapartes POSIX.
- `rwatTcCy`: permissões definidas pela ACE; no caso, temos definidas:
  - `r`: permissão de leitura para arquivos, ou listagem de diretórios.
  - `w`: permissão de escrita para arquivos, ou criação de novos arquivos em diretórios.
  - `a`: *append* de dados em arquivos (escrever ao final), ou criar novos subdiretórios em diretórios.
  - `t`: ler atributos do arquivo/diretório.
  - `T`: escrever atributos do arquivo/diretório.
  - `c`: ler ACLs NFSv4 do arquivo/diretório.

- `C`: escrever ACLs NFSv4 do arquivo/diretório.
- `y`: autorizar clientes a usar I/O síncrono com o servidor.

4. Vamos configurar uma ACL NFSv4 de leitura do arquivo para o usuário `han`, assim como fizemos anteriormente.

```
$ nfs4_setfacl -a A::han@intnet:rtcy ~/remote/teste
```

Vamos ver como ficaram as ACEs do arquivo:

```
$ nfs4_getfacl ~/remote/teste
A::OWNER@:rwatTcCy
A::10002:rtcy
A::GROUP@:tcy
A::EVERYONE@:tcy
```

Note que o nome de usuário `han@intnet` foi mapeado para o UID `10002` — que é consistente entre todas as máquinas do *datacenter* graças à integração com o LDAP que fizemos na sessão 3. Verifique a correspondência do UID:

```
$ getent passwd han
han:*:10002:10002:han:/home/han:/bin/bash
```

5. Vamos testar? Acesse como o usuário `aluno` e tente exibir o conteúdo do arquivo `/remote/luke/teste`:

```
$ su - aluno
Senha:
```

```
$ whoami
aluno
```

```
$ cat /remote/luke/teste
cat: /remote/luke/teste: Permissão negada
```

Agora, teste com o usuário `han`:

```
$ su - han
Senha:
```

```
$ whoami
han
```

```
$ cat /remote/luke/teste
oi
```

Excelente, tudo funcionando a contento.

6. Como remover uma ACL NFSv4? É simples:

```
$ nfs4_setfacl -x A::han@intnet:rtcy ~/remote/teste
```

```
$ nfs4_getfacl ~/remote/teste
A::OWNER@:rwatTcCy
A::10002:rtcy
A::GROUP@:tcy
A::EVERYONE@:tcy
```

Ué, não funcionou. Para deletar ACEs, temos que especificá-las **exatamente** no mesmo formato da linha reportada pelo comando `nfs4_getfacl`, ou usando o índice numérico da regra. Vamos tentar novamente:

```
$ nfs4_setfacl -x A::10002:rtcy ~/remote/teste
```

```
$ nfs4_getfacl ~/remote/teste
A::OWNER@:rwatTcCy
A::GROUP@:tcy
A::EVERYONE@:tcy
```

Agora sim, perfeito. Vamos verificar que a remoção da ACE surtiu efeito:

```
$ su - han
Senha:
```

```
$ whoami
han
```

```
$ cat /remote/luke/teste
cat: /remote/luke/teste: Permissão negada
```

9) Controle granular de permissões via sudo

Para todas as ações privilegiadas que precisamos tomar até aqui, sempre usamos o comando `su` para nos tornarmos o usuário `root`, e então efetuamos a instalação de pacotes, adição de usuários ou criação de arquivos de configuração. Mas, como fica essa situação em um ambiente de *datacenter* como o que estamos simulando? Seria interessante passar a senha do usuário `root` para os usuários `luke` e `han` (e outros que viermos a criar), permitindo que tomem quaisquer ações nas máquinas?

O `sudo` (*Super User DO*) é um comando que permite que usuários comuns obtenham privilégios de outro usuário, em geral o `root`, para executar tarefas específicas dentro do sistema de maneira segura e controlável pelo administrador. Assim, podemos delimitar que um determinado usuário ou grupo pode executar apenas um pequeno conjunto de comandos dentro de um servidor específico. Como o `sudo` é compatível com *hostnames* e endereços IP, é possível utilizar o mesmo arquivo em todas as máquinas do parque, facilitando tremendamente o esforço de configuração.

Para ilustrar esse cenário, vamos solucionar dois exemplos hipotéticos:

- A colaboradora `leia` acaba de se juntar à equipe de `han`, o grupo `fwadm` em nosso sistema LDAP. Imagine que ela ficará responsável por editar regras no firewall de borda, a máquina `ns1`. Mas, por estar começando agora na empresa, `han` quer restringir o conjunto de comandos que `leia` pode executar na máquina, liberando apenas a edição do firewall via `iptables`. Sua senha será `seg10leia`. Nas demais máquinas (`ns2` e `nfs`) `leia` não deve ter qualquer acesso especial, apenas como um usuário regular.
- O colaborador `chewie` foi contratado para auxiliar na manutenção da base LDAP da empresa. Para desempenhar suas tarefas, iremos colocá-lo em um novo grupo `ldapadm`. Os membros desse grupo devem ter acesso aos principais comandos de edição do LDAP (criação, modificação e deleção de usuários e grupos) na máquina `ns2`. Sua senha será `seg10chewie`. Nas demais máquinas (`ns1` e `nfs`) `chewie` não deve ter qualquer acesso especial, apenas como um usuário regular.
- Os usuários atuais, `luke` e `han`, terão permissão para executar qualquer comando como o usuário `root`, em qualquer máquina.
- Observe que temos controles alheios ao `sudo` já aplicados que irão restringir o acesso de certos usuários — por exemplo, apenas membros do grupo `fwadm` conseguem fazer login na máquina `ns1` devido à configuração do `nslcd` que realizamos na atividade (13) da sessão (3).

Vamos solucionar esses problemas?

1. Primeiro, devemos criar os usuários e grupos — vamos começar com `leia`. Como `root`, na máquina `ns2`:

```
# hostname ; whoami
ns2
root
```

```
# ldapadduser leia fwadm
Successfully added user leia to LDAP
Successfully set password for user leia
```

```
# ldapaddusertogroup leia fwadm
Successfully added user leia to group cn=fwadm,ou=Groups,dc=intnet
```

```
# ldapsetpasswd leia
Changing password for user uid=leia,ou=People,dc=intnet
New Password:
Retype New Password:
Successfully set password for user uid=leia,ou=People,dc=intnet
```

Agora, `chewie`. Lembre-se que no caso dele temos também que adicionar um novo grupo, `ldapadm`:

```
# ldapaddgroup ldapadm
Successfully added group ldapadm to LDAP
```

```
# ldapadduser chewie ldapadm
Successfully added user chewie to LDAP
Successfully set password for user chewie
```

```
# ldapaddusertogroup chewie ldapadm
Successfully added user chewie to group cn=ldapadm,ou=Groups,dc=intnet
```

```
# ldapsetpasswd chewie
Changing password for user uid=chewie,ou=People,dc=intnet
New Password:
Retype New Password:
Successfully set password for user uid=chewie,ou=People,dc=intnet
```

Fácil, não é mesmo?

2. Como já instalamos o `sudo` na máquina `debian-template` na sessão 1, o comando deve estar diponível no `$PATH`:

```
# which sudo
/usr/bin/sudo
```

Vamos testar? Edite o arquivo `/etc/sudoers` e autorize o usuário `luke` a usar o comando

`/bin/grep` como o usuário `root`. Edite o arquivo com:

```
# visudo
(...)
```

Insira a linha `luke ALL=/bin/grep` abaixo da entrada do usuário `root` na seção *User privilege specification*, como se segue:

```
# grep -A2 'User privilege specification' /etc/sudoers
# User privilege specification
root    ALL=(ALL:ALL) ALL
luke    ALL=(root)    /bin/grep
```

Como o usuário `luke`, tente usar o comando `grep` com o `sudo` para visualizar um arquivo restrito, como o `/etc/shadow`:

```
# su - luke
```

```
$ whoami
luke
```

```
$ sudo grep root /etc/shadow

Presumimos que você recebeu as instruções de sempre do administrador
de sistema local. Basicamente, resume-se a estas três coisas:

    #1) Respeite a privacidade dos outros.
    #2) Pense antes de digitar.
    #3) Com grandes poderes vêm grandes responsabilidades.

[sudo] senha para luke:
root:$6$s7Gt1cd.$UXQf67CVYxR7HP..h2wvhOx4nOtBT7do28R1uChYdMpZc.uLi43OKdtentrWD2zSTK
v9EyB7Bdqcpwr6nAlNo.:17848:0:99999:7:::
```

Agora, tente executar um comando não-autorizado, como o `cat`:

```
$ sudo cat /etc/shadow
Sinto muito, usuário luke não tem permissão para executar "/bin/cat /etc/shadow"
como root em ns2.intnet.
```

Perfeito, nosso teste inicial funcionou com sucesso. Remova a linha referente ao usuário `luke` no arquivo `/etc/sudoers`, e vamos prosseguir.

3. Queremos controlar os comandos utilizados nos servidores do *datacenter*, que até o momento são as máquinas `ns1`, `ns2` e `nfs`.

   Tecnicamente, seria possível configurar o `sudo` em cada um dos servidores — uma vez que as regras para a usuária `leia` são específica para a máquina `ns1` e as do usuário `chewie` se aplicam à máquina `ns2` — mas não faremos isso. Imagine que, ao invés de três máquinas, nosso *datacenter* tivesse centenas de VMs: seria factível controlar as regras de `sudo` localmente em cada um dos servidores? É evidente que não.

   Temos algumas opções para configurar o `sudo` de forma coordenada entre múltiplos servidores. A gestão de configuração de servidores, usando ferramentas como o Puppet, Chef ou Ansible (que veremos na sessão seguinte) é um dos métodos mais modernos e eficientes de atingir esse objetivo.

   Assim sendo, ao invés de desenvolver uma solução inadequada para o problema de configuração do `sudo` neste momento, iremos fazer uma breve pausa nessa tarefa. Na próxima sessão, aprenderemos as bases do Ansible e, com isso, a solução do problema ficará bem mais fácil. Nos vemos em breve!

Sessão 5: Gestão de configuração

Algo que já deve ter ficado claro, a este ponto, é que a configuração manual de múltiplas máquinas é um processo tedioso, demorado, e ainda oferece o risco de configuração incorreta por parte do administrador durante o processo. Fica ainda mais evidente a dimensão do problema quando observamos que nosso *datacenter* simulado possui até o momento apenas três servidores — imagine o tamanho do problema se tivéssemos dezenas ou centenas de VMs!

As ferramentas de gestão de configuração oferecem uma alternativa: através delas, podemos ter um processo de implementação sistemática de mudanças em um (ou vários) sistemas de forma a manter sua integridade. A ideia básica por trás dessas ferramentas é a gestão de **estados desejados**: criamos artefatos de configuração que definem um estado-alvo para um grupo de sistemas, e o sistema de gestão de configuração se encarrega de checar se esse estado está atendido — caso positivo, nada precisa ser feito; e, caso negativo, as alterações de configuração previstas nos artefatos de configuração serão aplicadas nos sistemas.

Outro conceito central é o **comportamento idempotente** desse tipo de ferramenta: mesmo com aplicações sucessivas dos artefatos de configuração, o sistema se encarrega de verificar se o estado-alvo está atendido, não reaplicando mudanças desnecessárias. Podemos citar também como vantagens desses sistemas de gestão a existência de *frameworks* de automação que facilitam o trabalho do administrador, uso de *templates* para aproveitar e customizar configuração entre diferentes grupos de máquinas, e grande extensibilidade de capacidades através de módulos e *plugins* de terceiros.

Nesta sessão iremos trabalhar com o Ansible, uma ferramenta de gestão de configuração *open-source* criada por Michael DeHaan em 2012 e atualmente mantida pela Red Hat. O Ansible possibilita também a automatização de provisionamento de software e *deployment* de aplicações, conectando-se via SSH ou PowerShell aos servidores-alvo em um arquitetura que dispensa a instalação de agentes (*agentless*). Usando o Ansible, iremos solucionar a gestão centralizada do arquivo `/etc/sudoers`, problema que encontramos no final da sessão anterior. Finalmente, iremos usar a ferramenta de controle de versão Git para gerenciar as mudanças que faremos nos conjuntos de *scripts* e artefatos do Ansible ao longo das sessões.

1) Topologia desta sessão

A figura abaixo mostra a topologia de rede que será utilizada nesta sessão, com as máquinas relevantes em destaque.

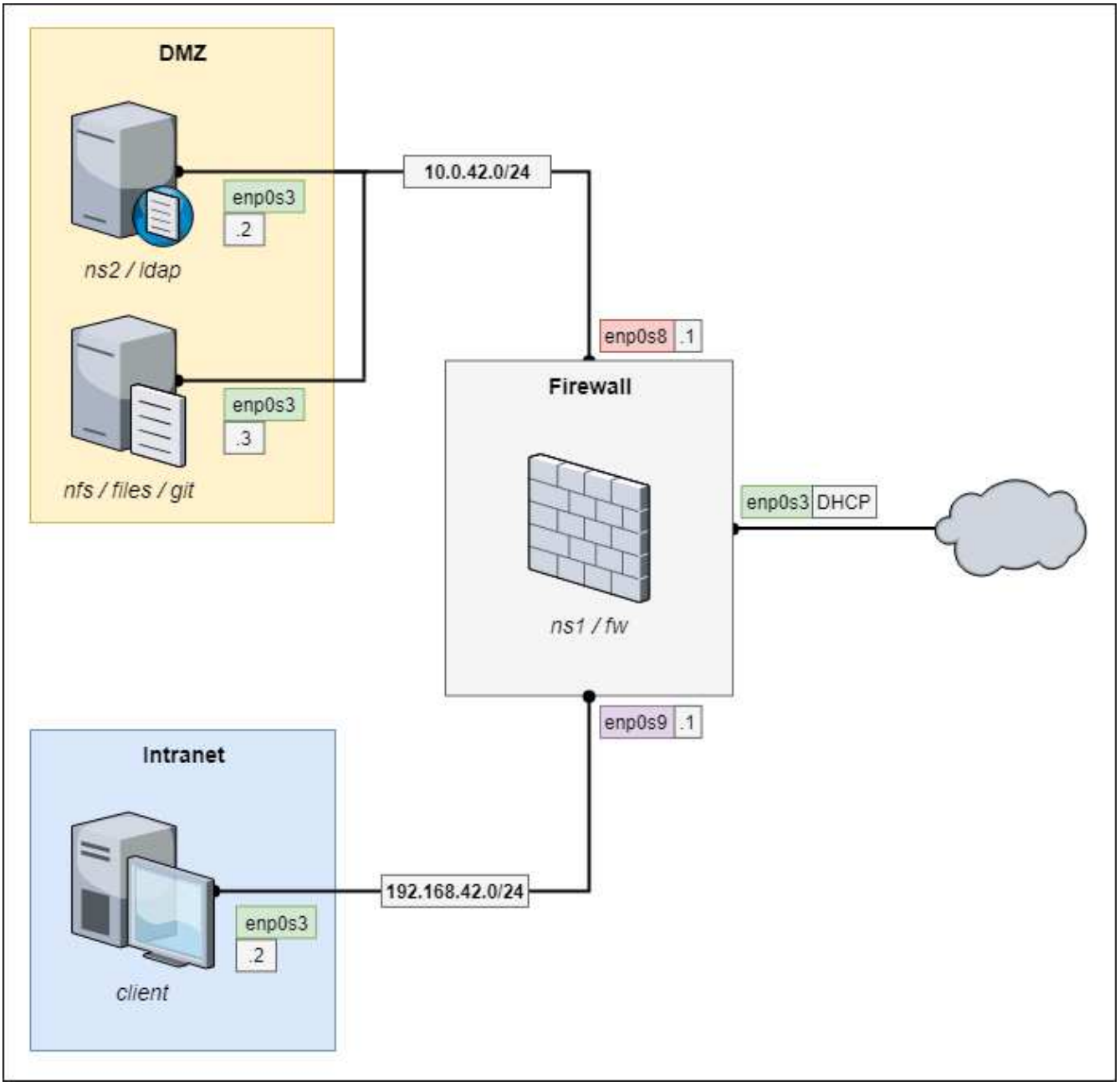

Figura 29. Topologia de rede desta sessão

1. Praticamente não há mudanças em relação à topologia da sessão anterior — a única diferença é que a máquina `nfs` possuirá um novo *alias*, `git`, já que atuará como servidor da solução de controle de versão Git. Vamos ajustar o DNS: acesse a máquina `ns1` como o usuário `root`:

   ```
   # hostname ; whoami
   ns1
   root
   ```

   Edite o arquivo de zonas `/etc/nsd/zones/intnet.zone`, inserindo uma entrada CNAME para a máquina `nfs`, como se segue. **Não se esqueça** de incrementar o valor do serial no topo do arquivo!

```
# nano /etc/nsd/zones/intnet.zone
(...)
```

```
# grep git /etc/nsd/zones/intnet.zone
git     IN    CNAME             nfs
```

Assine o arquivo de zonas usando o *script* criado anteriormente:

```
# bash /root/scripts/signzone-intnet.sh
reconfig start, read /etc/nsd/nsd.conf
ok
ok
ok
ok removed 0 rrsets, 0 messages and 0 key entries
```

Verifique a criação das entradas usando o comando `dig`:

```
# dig git.intnet +short
nfs.intnet.
10.0.42.3
```

2) Instalação e configuração inicial do Ansible

Um dos aspectos mais interessantes do Ansible é sua simplicidade — dependendo apenas do interpretador Python para operar (o qual já vem instalado no sistema-base do Debian), sua instalação é bastante simples. Outro fator relevante: como o sistema dispensa a instalação de agentes, não é necessário criar um servidor dedicado para usar o Ansible, de forma que iremos usar a máquina `client` como um conveniente ponto de partida para os logins remotos.

Vamos, no entanto, criar um usuário específico para o Ansible no servidor LDAP, gerenciando suas permissões de forma granular via `sudo`.

1. Acesse a máquina `ns2` como o usuário `root` e crie um usuário para o Ansible, membro dos grupos `setup` (primário) e `fwadm`, com senha `seg10ansible`:

```
# hostname ; whoami
ns2
root
```

```
# ldapadduser ansible setup
Successfully added user ansible to LDAP
Successfully set password for user ansible
```

```
# ldapaddusertogroup ansible setup
Successfully added user ansible to group cn=setup,ou=Groups,dc=intnet
```

```
# ldapaddusertogroup ansible fwadm
Successfully added user ansible to group cn=fwadm,ou=Groups,dc=intnet
```

```
# ldapsetpasswd ansible
Changing password for user uid=ansible,ou=People,dc=intnet
New Password:
Retype New Password:
Successfully set password for user uid=ansible,ou=People,dc=intnet
```

2. Agora, acesse a máquina `client` como o usuário `root` e instale o Ansible seguindo os passos abaixo. Todas as instruções a seguir referenciam o passo-a-passo da documentação oficial do Ansible, acessível em https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#latest-releases-via-apt-debian .

```
# hostname ; whoami
client
root
```

Primeiro, adicione o repositório de pacotes do Ansible à lista de fontes do `apt-get`:

```
# echo "deb http://ppa.launchpad.net/ansible/ansible/ubuntu trusty main" >
/etc/apt/sources.list.d/ansible.list
```

Em seguida, adicione à lista de chaves confiáveis para instalação de pacotes a *pubkey* dos mantenedores do pacote do Ansible:

```
# apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys
93C4A3FD7BB9C367
Executing: /tmp/apt-key-gpghome.UOSyzarqGQ/gpg.1.sh --keyserver
hkp://keyserver.ubuntu.com:80 --recv-keys 93C4A3FD7BB9C367
gpg: key 93C4A3FD7BB9C367: public key "Launchpad PPA for Ansible, Inc." imported
gpg: Total number processed: 1
gpg:               imported: 1
```

Agora, basta atualizar a lista de pacotes disponíveis nos repositórios remotos e instalar o Ansible usando o `apt-get`:

```
# apt-get update
```

```
# apt-get install ansible
```

3) Execução de comandos simples

1. Começaremos utilizando as funções mais básicas do Ansible. Antes de mais nada, no entanto, precisamos configurar o acesso para os diversos servidores do *datacenter* simulado usando o sistema de autenticação LDAP/SSH-CA.

   Acesse a máquina `client` como o usuário `ansible`:

   ```
   $ hostname ; whoami ; pwd
   client
   ansible
   /home/ansible
   ```

   Agora, use o script `~/scripts/sshsign_user.sh` para assine um par de chaves e conseguir logar nos servidores:

   ```
   $ bash ~/scripts/sshsign_user.sh
   Signing ~/.ssh/id_rsa.pub key...
   All done!
   ```

2. Como estabelecido, o Ansible consegue trabalhar com múltiplos sistemas em uma infraestrutura ao mesmo tempo — para isso, ele seleciona conjuntos de máquinas em seu **inventário**. O inventário é um arquivo texto em formato INI ou YAML que lista as máquinas e grupos a serem gerenciados.

   Crie o arquivo novo `/home/ansible/hosts` com o seguinte conteúdo:

   ```
   $ nano ~/hosts
   (...)
   ```

   ```
   $ cat hosts
   [srv]
   ns1
   ns2
   nfs
   ```

   No arquivo acima criamos um único grupo, `srv`, contendo todos os servidores do *datacenter* simulado criados até aqui.

3. Vamos testar? Execute um comando simples em todas as máquinas gerenciadas pelo Ansible:

```
$ ansible -i ~/hosts srv -m shell -a 'hostname --fqdn ; whoami'
ns2 | CHANGED | rc=0 >>
ns2.intnet
ansible

ns1 | CHANGED | rc=0 >>
ns1.intnet
ansible

nfs | CHANGED | rc=0 >>
nfs.intnet
ansible
```

O que aconteceu? Vejamos:

- Com `ansible`, invocamos o Ansible para executar uma única tarefa, com parâmetros passados diretamente via linha de comando.
- Depois, `-i ~/hosts` define o arquivo de inventário a ser usado, `/home/ansible/hosts`.
- O grupo `srv` é indicado a seguir, falando para o Ansible qual dos grupos disponíveis no arquivo de inventário será o alvo dos comandos que se seguem.
- Com `-m shell` carregamos o módulo *shell* do Ansible, que permite execução de comandos diretamente pelo interpretador `/bin/sh` (ou outro à sua escolha) nas máquinas remotas.
- Finalmente, `-a 'hostname --fqdn ; whoami'` indica quais comandos o *shell* sendo executado em cada uma das máquinas-membros do grupo `srv` irá operar.

Analise a saída, agora: note que o Ansible loga em cada uma das máquinas do grupo `srv` (`ns2`, `ns1` e `nfs`) e executa os comandos indicados, mostrando claramente o *hostname* local e usuário logado. Observe, ainda, que a ordem das máquinas escrita no arquivo `/home/ansible/hosts` não foi respeitada — o Ansible inicia conexões nos servidores de forma paralela, e as respostas podem vir fora de ordem.

Um último adendo: mencionamos o uso do módulo *shell* no comando acima, mas quais outros módulos existem? Há muitos outros — mesmo. Confira a lista completa na página https://docs.ansible.com/ansible/latest/modules/modules_by_category.html .

4. A próxima pergunta que pode surgir neste momento é: Ok, conseguimos executar um comando remoto com um usuário não-privilegiado. E se quisermos executar como o `root`? Para isso, podemos executar a escalação de privilégios usando o `become`. Veja:

```
$ ansible -i ~/hosts srv --become --become-user=root --become-method=su --ask
-become-pass -m shell -a 'hostname --fqdn ; head -n1 /etc/shadow'
SU password:
ns1 | CHANGED | rc=0 >>
ns1.intnet
root:$6$s7Gt1cd.$UXQf67CVYxR7HP..h2wvhOx4nOtBT7do28R1uChYdMpZc.uLi43OKdtentrWD2zSTK
v9EyB7Bdqcpwr6nAlNo.:17848:0:99999:7:::

ns2 | CHANGED | rc=0 >>
ns2.intnet
root:$6$s7Gt1cd.$UXQf67CVYxR7HP..h2wvhOx4nOtBT7do28R1uChYdMpZc.uLi43OKdtentrWD2zSTK
v9EyB7Bdqcpwr6nAlNo.:17848:0:99999:7:::

nfs | CHANGED | rc=0 >>
nfs.intnet
root:$6$s7Gt1cd.$UXQf67CVYxR7HP..h2wvhOx4nOtBT7do28R1uChYdMpZc.uLi43OKdtentrWD2zSTK
v9EyB7Bdqcpwr6nAlNo.:17848:0:99999:7:::
```

Quais as diferenças para o comando anterior?

- `--become` indica que desejamos executar comandos como outro usuário — esta opção não implica qual usuário, método ou senha serão usados, mas apenas que haverá a alteração de usuário efetivo.
- Em seguida, `--become-user=root` informa que o usuário a se tornar nas máquinas remotas será, de fato, o `root`.
- A opção `--become-method=su` configura o método de escalada de privilégio usado. Como ainda não configuramos o `sudo` nas máquinas remotas, iremos usar o `su`.
- `--ask-become-pass` solicita ao Ansible que pergunte a senha de escalação de privilégio antes de executar os comandos, evitando que tenhamos que digitá-la diretamente no terminal.
- Por fim, `'hostname --fqdn ; head -n1 /etc/shadow'` visa identificar a máquina remota e depois executar um comando que apenas o `root` conseguiria fazer.

Excelente! Estamos conseguindo controlar remotamente todos os servidores com sucesso. Mas e se quisermos fazer mais que apenas rodar comandos simples?

4) Uso de roles no Ansible

Caso queiramos fazer mais que executar poucos comandos num *shell* remoto, o uso de *roles* (ou papéis) é fundamental para organizar tarefas complexas, cópia remota de arquivos e gerência de dependências entre serviços. Vamos usar *roles* no Ansible para solucionar o problema de gestão centralizada do arquivo `/etc/sudoers`, apresentado no final da sessão anterior.

1. Crie um diretório para armazenar as *roles* que serão criadas, `/home/ansible/roles`. Em seguida, entre nesse diretório.

```
$ mkdir ~/roles
```

```
$ cd ~/roles/
```

2. O comando `ansible-galaxy` nos permite realizar uma série de operações relacionadas a *roles*, desde a criação de simples papéis locais até a busca e importação de *roles* criadas por outros usuários do Ansible e acessíveis na Internet pelo website https://galaxy.ansible.com/ . Recomendamos ao aluno que pesquise por tarefas usuais nesse site — provavelmente já existe uma *role* para solucionar esse problema!

   Para manter a simplicidade das atividades executadas no curso, iremos criar *roles* manualmente. Use o `ansible-galaxy` para criar uma estrutura de diretórios padrão para a *role* `sudoers`:

   ```
   $ ansible-galaxy init --init-path ~/roles/ sudoers
   - sudoers was created successfully
   ```

   O que foi criado? Vejamos:

   ```
   $ ls -R ~/roles/sudoers/
   /home/ansible/roles/sudoers/:
   defaults  files  handlers  meta  README.md  tasks  templates  tests  vars

   /home/ansible/roles/sudoers/defaults:
   main.yml

   /home/ansible/roles/sudoers/files:

   /home/ansible/roles/sudoers/handlers:
   main.yml

   /home/ansible/roles/sudoers/meta:
   main.yml

   /home/ansible/roles/sudoers/tasks:
   main.yml

   /home/ansible/roles/sudoers/templates:

   /home/ansible/roles/sudoers/tests:
   inventory  test.yml

   /home/ansible/roles/sudoers/vars:
   main.yml
   ```

   Cada um dos diretórios criados possui uma função específica, a saber:

- `defaults`: contém variáveis-padrão a serem usadas na *role*.
- `files`: contém arquivos que serão copiados através desta *role*.
- `handlers`: contém funções especiais denominadas *handlers*. Essas funções são bastante parecida com *tasks*, mas são invocadas de forma indireta e servem para automatizar tarefas recorrentes, como o *reload* de *daemons* de serviços.
- `meta`: contém metadados sobre a *role*, como a definição de dependências entre *roles* em um mesmo diretório.
- `tasks`: contém a lista principais de tarefas a serem executados pela *role*.
- `templates`: contém *templates* que podem ser copiados através desta *roles* — *templates* podem basicamente ser entendidos como arquivos que se utilizam de variáveis para customizar seu estado final.
- `tests`: permite a criação de testes para verificar a correta execução de aspectos da *role* corrente. Frequentemente não é necessário desenvolver (muitos) testes no Ansible, já que o sistema de configuração modela um estado-alvo desejado, de forma declarativa.
- `vars`: define outras variáveis para a *role*, que têm precedência sobre as variáveis em `defaults`.

3. Retomemos o problema central, a configuração do arquivo `/etc/sudoers` de forma centralizada:

   - A colaboradora `leia` acaba de se juntar à equipe de `han`, o grupo `fwadm` em nosso sistema LDAP. Imagine que ela ficará responsável por editar regras no firewall de borda, a máquina `ns1`. Mas, por estar começando agora na empresa, `han` quer restringir o conjunto de comandos que `leia` pode executar na máquina, liberando apenas a edição do firewall via `iptables`. Sua senha será `seg10leia`. Nas demais máquinas (`ns2` e `nfs`) `leia` não deve ter qualquer acesso especial, apenas como um usuário regular.
   - O colaborador `chewie` foi contratado para auxiliar na manutenção da base LDAP da empresa. Para desempenhar suas tarefas, iremos colocá-lo em um novo grupo `ldapadm`. Os membros desse grupo devem ter acesso aos principais comandos de edição do LDAP (criação, modificação e deleção de usuários e grupos) na máquina `ns2`. Sua senha será `seg10chewie`. Nas demais máquinas (`ns1` e `nfs`) `chewie` não deve ter qualquer acesso especial, apenas como um usuário regular.
   - Os usuários atuais, `luke` e `han`, terão permissão para executar qualquer comando como o usuário `root`, em qualquer máquina.

   A criação de usuários e grupos já foi tratada no passo (1) da atividade (9) da sessão anterior, como visto. Foquemos, então, apenas arquivo `sudoers`. Como o `sudo` é um programa estático (isto é, não depende de nenhum *daemon* executando para funcionar), o único requerimento para uma *role* que configure o arquivo `/etc/sudoers` é copiar o arquivo corretamente para as máquinas, e ajustar suas permissões.

   Crie o arquivo novo `/home/ansible/roles/sudoers/files/sudoers`, com o seguinte conteúdo:

```
 1 Defaults        env_reset
 2 Defaults        mail_badpass
 3 Defaults        secure_path
="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
 4
 5 User_Alias ADMINS    = aluno, \
 6                        luke,  \
 7                        han
 8
 9 User_Alias FWUSERS   = leia
10
11 User_Alias LDAPUSERS = %ldapadm
12
13 Host_Alias FWHOSTS   = ns1
14
15 Host_Alias LDAPHOSTS = ns2
16
17 Cmnd_Alias FWCMDS    = /sbin/iptables
18
19 Cmnd_Alias LDAPCMDS  = /usr/sbin/ldapaddgroup,             \
20                        /usr/sbin/ldapadduser,              \
21                        /usr/sbin/ldapaddusertogroup,       \
22                        /usr/sbin/ldapdeletegroup,          \
23                        /usr/sbin/ldapdeleteuser,           \
24                        /usr/sbin/ldapdeleteuserfromgroup,  \
25                        /usr/sbin/ldapmodifygroup,          \
26                        /usr/sbin/ldapmodifymachine,        \
27                        /usr/sbin/ldapmodifyuser,           \
28                        /usr/sbin/ldaprenamegroup,          \
29                        /usr/sbin/ldaprenameuser,           \
30                        /usr/sbin/ldapsetpasswd,            \
31                        /usr/sbin/ldapsetprimarygroup
32
33 root      ALL=(ALL:ALL)    ALL
34
35 ansible   ALL=(ALL:ALL)    NOPASSWD: ALL
36
37 ADMINS    ALL=(ALL:ALL)    ALL
38
39 FWUSERS   FWHOSTS=(root)   FWCMDS
40
41 LDAPUSERS LDAPHOSTS=(root) LDAPCMDS
42
43 #includedir /etc/sudoers.d
```

O que esse arquivo faz? Vamos ver:

- Nas linhas [5-11] definimos *aliases* (apelidos) de usuários para agrupar os elementos que serão configurados para usar o `sudo`. Criamos um *alias* `ADMINS` para agrupar os usuários `aluno`, `luke` e `han`, `FWUSERS` para `leia` e `LDAPUSERS` para o **grupo** `ldapadm`. É especialmente

importante manter um *alias* apontando para um usuário local, como o usuário `aluno`, caso haja problemas com o sistema de autenticação LDAP.

- Nas linhas [13-15] definimos *aliases* para máquinas, `ns1` e `ns2`. Também poderíamos usar endereços IP, se desejado.
- Nas linhas [17-31] definimos *aliases* de comandos: para a máquina `ns1`, apenas o comando `/sbin/iptables` é suficiente; já para a máquina `ns2` configuramos uma lista detalhada dos comandos que o *alias* `LDAPUSERS` poderá usar.
- Nas linhas [33-41] fazemos a "amarração" dos *aliases* previamente definidos, atribuindo aos usuários/grupos em quais máquinas eles podem executar os comandos, como quais usuários, e quais são esses comandos. Note que ao usuário `ansible` é permitido executar qualquer comando como `root` em todas as máquinas sem a necessidade de digitação de senha, de forma bastante conveniente para a automação de tarefas via Ansible como faremos em sessões futuras.

Compare as regras sendo definidas pelos *aliases* no arquivo `sudoers` acima e os requisitos de acesso definidos no começo deste passo — todos estão sendo atendidos a contento. É claro, seria possível organizar os usuários/grupos de forma diferente — em particular, poderia ser interessante agrupar os usuário `han` e `luke` em um grupo dedicado de "super-admins" — mas funciona como uma boa prova de conceito.

Para o aluno atento, é claro observar que o conjunto de programas que estão sendo autorizados para o usuário `chewie` (de fato, todo o grupo `ldapadm`) garante a ele uma permissão efetiva **muito** maior que a prevista no escopo inicial do problema. Afinal, bastaria ao usuário `chewie` adicionar-se a si mesmo no grupo `fwadm` (via comando `ldapaddusertogroup`) para acessar a máquina `ns1`, ou alterar a senha dos usuários `han` ou `luke` (via comando `ldapsetpasswd`) e entrar como esses usuários em máquinas que permitam login via senha. Iremos "ignorar" esse problema nesta atividade, em nome da simplicidade.

De fato, fica aqui também um desafio: qual seria o conjunto adequado de programas e permissões garantidas via `sudoers` que permitiria ao usuário `chewie` executar seu trabalho de manutenção de contas no LDAP da empresa, e ao mesmo tempo controlar seu acesso de forma efetiva? Lembre-se que o `sudoers` permite ao administrador definir não apenas quais comandos serão autorizados, mas também parâmetros específicos de linha de comando, se desejado.

4. Criado o arquivo `sudoers` que atende aos requisitos iniciais, temos que criar a tarefa que irá distribuí-lo. Antes disso, observe as permissões do arquivo `sudoers` original:

```
$ ls -ld /etc/sudoers
-r--r----- 1 root root 669 jun  5  2017 /etc/sudoers
```

Com essas permissões em mente, edite o arquivo `/home/ansible/roles/sudoers/tasks/main.yml` com o seguinte conteúdo:

```
---
- name: Propagate sudoers configuration
  become: yes
  become_user: root
  become_method: su
  copy:
    src: sudoers
    dest: /etc
    owner: root
    group: root
    mode: 0440
```

Definimos uma tarefa de propagação do arquivo `sudoers`, com as seguintes características: iremos rodar a tarefa como o usuário `root`, usando o `su` para escalada de privilégio; será copiado o arquivo `sudoers` (do diretório `files/`, implícito aqui) para a pasta `/etc` na máquina remota, com `root` como usuário e grupo donos, e permissões `r—r-----`.

5. Tudo pronto? Vamos executar! Crie o arquivo `/home/ansible/srv.yml` para fazer a amarração entre o grupo de *hosts* e a nova *role*:

```
$ nano ~/srv.yml
(...)
```

```
$ cat ~/srv.yml
---
- hosts: srv
  roles:
    - sudoers
```

Como ainda **não** configuramos o `sudo`, e estamos usando o comando `su` para escalada de privilégio, iremos passar novamente a opção `--ask-become-pass` para que o Ansible solicite a senha do usuário `root` para escalada de privilégio nas máquinas remotas. Execute a role:

```
$ ansible-playbook -i ~/hosts ~/srv.yml --ask-become-pass
SUDO password:

PLAY [srv]
*************************************************************************************
***********************


TASK [Gathering Facts]
*************************************************************************************
************

ok: [ns2]
ok: [ns1]
ok: [nfs]

TASK [sudoers : Propagate sudoers configuration]
*******************************************************************

changed: [ns2]
changed: [ns1]
changed: [nfs]

PLAY RECAP
*************************************************************************************
***********************

nfs                        : ok=2    changed=1    unreachable=0    failed=0
ns1                        : ok=2    changed=1    unreachable=0    failed=0
ns2                        : ok=2    changed=1    unreachable=0    failed=0
```

6. Terá funcionado? Vamos ver: tente executar um comando distribuído com o Ansible usando o `sudo` como método de escalada de privilégio, e sem digitar senha.

```
$ ansible -i ~/hosts srv -b --become-user=root --become-method=sudo -m shell -a
'hostname --fqdn ; grep ansible /etc/sudoers'
nfs | CHANGED | rc=0 >>
nfs.intnet
ansible   ALL=(ALL:ALL)    NOPASSWD: ALL

ns2 | CHANGED | rc=0 >>
ns2.intnet
ansible   ALL=(ALL:ALL)    NOPASSWD: ALL

ns1 | CHANGED | rc=0 >>
ns1.intnet
ansible   ALL=(ALL:ALL)    NOPASSWD: ALL
```

Perfeito! Veja que o Ansible conseguiu acessar todas as máquinas, elevar privilégios usando o `sudo` sem necessidade de senha, e imprimir o conteúdo do arquivo `/etc/sudoers` — o qual pode ser lido apenas pelo `root` e, convenientemente, filtramos a linha em que a autorização ao usuário `ansible` está sendo configurada.

5) Testando os controles do sudo

Conseguimos distribuir o arquivo `sudoers` como objetivado, mas será que os controles estão funcionando?

1. Vamos testar o acesso de `leia` na máquina `ns1` — lembre-se, ela deve conseguir executar apenas o comando `iptables` como o usuário `root`, e nenhum outro. Acesse a máquina `ns1` como `leia`:

```
$ hostname ; whoami
ns1
leia
```

Agora, tente executar o comando `iptables` usando o `sudo`:

```
$ sudo iptables -L POSTROUTING -vn -t nat

Presumimos que você recebeu as instruções de sempre do administrador
de sistema local. Basicamente, resume-se a estas três coisas:

    #1) Respeite a privacidade dos outros.
    #2) Pense antes de digitar.
    #3) Com grandes poderes vêm grandes responsabilidades.

[sudo] senha para leia:
Chain POSTROUTING (policy ACCEPT 322 packets, 25437 bytes)
 pkts bytes target     prot opt in     out     source               destination
    1    60 MASQUERADE  tcp  --  *      enp0s3  10.0.42.0/24         0.0.0.0/0
multiport dports 80,443
    9   540 MASQUERADE  tcp  --  *      enp0s3  192.168.42.0/24      0.0.0.0/0
multiport dports 80,443
```

Excelente! E se tentarmos executar um comando não autorizado?

```
$ sudo rm /etc/shadow
Sinto muito, usuário leia não tem permissão para executar "/bin/rm /etc/shadow"
como root em ns1.intnet.
```

De fato, é possível listar exatamente quais comandos um usuário está apto a executar com o comando `sudo -l`:

```
$ sudo -l
Entradas de Defaults correspondentes a leia em ns1:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

Usuário leia pode executar os seguintes comandos em ns1:
    (root) /sbin/iptables
```

E quanto a `han`? Ele consegue executar qualquer comando como `root`?

```
$ hostname ; whoami
ns1
han
```

```
$ sudo -l

Presumimos que você recebeu as instruções de sempre do administrador
de sistema local. Basicamente, resume-se a estas três coisas:

    #1) Respeite a privacidade dos outros.
    #2) Pense antes de digitar.
    #3) Com grandes poderes vêm grandes responsabilidades.

[sudo] senha para han:
Entradas de Defaults correspondentes a han em ns1:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

Usuário han pode executar os seguintes comandos em ns1:
    (ALL : ALL) ALL
```

Excelente! Os acessos para a máquina `ns1` estão funcionando como previsto.

2. Vamos para o caso do usuário `chewie`. Acesse a máquina `ns2` como `chewie` e verifique quais comandos você está autorizado a executar usando o `sudo`:

```
$ hostname ; whoami
ns2
chewie
```

```
$ sudo -l

Presumimos que você recebeu as instruções de sempre do administrador
de sistema local. Basicamente, resume-se a estas três coisas:

    #1) Respeite a privacidade dos outros.
    #2) Pense antes de digitar.
    #3) Com grandes poderes vêm grandes responsabilidades.

[sudo] senha para chewie:
Entradas de Defaults correspondentes a chewie em ns2:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

Usuário chewie pode executar os seguintes comandos em ns2:
    (root) /usr/sbin/ldapaddgroup, /usr/sbin/ldapadduser,
/usr/sbin/ldapaddusertogroup, /usr/sbin/ldapdeletegroup,
        /usr/sbin/ldapdeleteuser, /usr/sbin/ldapdeleteuserfromgroup,
/usr/sbin/ldapmodifygroup,
        /usr/sbin/ldapmodifymachine, /usr/sbin/ldapmodifyuser,
/usr/sbin/ldaprenamegroup, /usr/sbin/ldaprenameuser,
        /usr/sbin/ldapsetpasswd, /usr/sbin/ldapsetprimarygroup
```

Tente criar um novo grupo no LDAP, `sudotest`, e em seguida delete-o.

```
$ sudo ldapaddgroup sudotest
Successfully added group sudotest to LDAP
```

```
$ sudo ldapdeletegroup sudotest
Successfully deleted group cn=sudotest,ou=Groups,dc=intnet from LDAP
```

Tente executar um comando não-autorizado:

```
$ sudo reboot
Sinto muito, usuário chewie não tem permissão para executar "/sbin/reboot" como
root em ns2.intnet.
```

Tudo de acordo com o esperado, muito bom.

6) Controle da senha do usuário root

1. Pergunte-se agora o seguinte: e se `leia` ou `chewie`, por qualquer motivo, conseguirem obter a senha do usuário `root`? O que não é exatamente difícil, já que estamos usando `rnpesr` como senha — o que ocorre então? Veja o que acontece ao usarmos o comando `su` como `chewie` na máquina `ns2`:

```
$ hostname ; whoami
ns2
chewie
```

```
$ su -
Senha:
```

```
# whoami
root
```

A solução ideal, nesse caso, é desabilitar a senha do `root`. Já que os acessos de superusuário estão corretamente configurados via arquivo `sudoers` distribuído centralmente via Ansible, não teremos prejuízo administrativo — nesse cenário, mesmo que usuários não-autorizados descubram a senha, está já estará desabilitada e não poderá ser usada para efetuar escalada de privilégio.

2. Acesse a máquina `client` como o usuário `ansible`. Para desabilitar a conta do `root`, basta usar o comando `passwd -l` — vamos executar esse comando em todos os servidores do *datacenter*:

```
$ ansible -i ~/hosts srv -b --become-user=root --become-method=sudo -m shell -a
'passwd -l root'
ns1 | CHANGED | rc=0 >>
passwd: informação de expiração de senha alterada.

nfs | CHANGED | rc=0 >>
passwd: informação de expiração de senha alterada.

ns2 | CHANGED | rc=0 >>
passwd: informação de expiração de senha alterada.
```

3. Com a senha desabilitada, apenas aqueles usuários que tenham permissão de `sudo` para executar comandos de escalada de privilégio poderão tornar-se o usuário `root` — todos os demais, restritos a um subconjunto de comandos controlados pelo arquivo `/etc/sudoers`, não conseguirão fazê-lo.

   Por exemplo, acesse a máquina `ns1` como o usuário `han`.

```
$ hostname ; whoami
ns1
han
```

   Note que mesmo o usuário `han`, que possui acesso irrestrito, não consegue executar `su` diretamente, mesmo digitando a senha correta:

```
$ su -
Senha:
su: Falha de autenticação
```

Apenas com comandos como `sudo su -`, `sudo -i` ou `sudo --login` (que equivale a invocar um *shell* de login, como executar `sudo bash`) é possível tornar-se `root`, como podemos ver:

```
$ sudo -i
```

```
# whoami
root
```

4. Mas... observe, nossa configuração não está ortogonal. Imagine que uma nova máquina seja criada em nosso *datacenter*, e queremos configurá-la com a distribuição do arquivo `sudoers` de forma centralizada, assim como fizemos com nossos servidores atuais. Sem problema, certo? Basta adicionar o novo *hostname* no arquivo `~/hosts` e disparar com o `ansible-playbook`!

   A configuração de *lockout* da senha do `root` que fizemos no passo (2), no entanto, não está integrada à *role* `sudoers`. De fato, executamos um comando direto usando o módulo *shell* — e se esquecermos de fazer esse passo com a próxima máquina? Nesse caso, a senha do `root` dessa máquina estará habilitada, e teremos uma configuração divergente em nossos servidores.

   Fica claro, portanto, que o método correto de integração de novas tarefas no Ansible é via *roles*, e não via comandos isolados, sob pena de criar uma grande confusão no parque de servidores num futuro próximo. Vamos reverter o comando que fizemos antes:

```
$ ansible -i ~/hosts srv -b --become-user=root --become-method=sudo -m shell -a
'passwd -u root'
ns1 | CHANGED | rc=0 >>
passwd: informação de expiração de senha alterada.

nfs | CHANGED | rc=0 >>
passwd: informação de expiração de senha alterada.

ns2 | CHANGED | rc=0 >>
passwd: informação de expiração de senha alterada.
```

5. Como desabilitar a conta do `root` do jeito certo? Vamos editar a lista de *tasks* da *role* `sudoers`. Adicione a tarefa a seguir no arquivo `/home/ansible/roles/sudoers/tasks/main.yml` (copie todo o comando, desde `cat` até a palavra `EOF`):

```
$ cat << EOF >> ~/roles/sudoers/tasks/main.yml

- name: Sets root account as expired
  user:
    name: root
    expires: 0
EOF
```

Uma curiosidade: a sintaxe acima é conhecida como *heredoc* (ou *Here Document*), um bloco de código que pode ser usado para diversas tarefas no *shell* Bash. No exemplo acima, estamos usando-o para inserir todas as linhas ao final do arquivo `/home/ansible/roles/sudoers/tasks/main.yml`, até que seja encontrada a *string* `EOF`. Consulte este link para mais informações sobre uso de *heredocs*: https://www.tldp.org/LDP/abs/html/here-docs.html .

De volta à tarefa em mãos, vamos ver como ficou o arquivo após a alteração:

```
$ cat /home/ansible/roles/sudoers/tasks/main.yml
---
- name: Propagate sudoers configuration
  become: yes
  become_user: root
  become_method: su
  copy:
    src: sudoers
    dest: /etc
    owner: root
    group: root
    mode: 0440

- name: Sets root account as expired
  user:
    name: root
    expires: 0
```

A tarefa que acabamos de adicionar irá se valer do módulo *user* do Ansible, e ajustar a data de expiração do usuário `root` para 0 — esse valor é contado em segundos a partir do dia 1 de janeiro de 1970, uma referência conhecida como *Unix Epoch*, ou *POSIX Time*. Com efeito, estamos dizendo que a conta deste usuário está expirada desde 1/1/1970.

6. Falta alguma coisa? Ah sim! Note que o `become_method` do arquivo `/home/ansible/roles/sudoers/tasks/main.yml` ainda está configurado como `su` — como já distribuímos o arquivo `sudoers` anteriormente com sucesso usando essa *role*, podemos alterá-lo para `sudo`.

   Mais além, observe que as diretivas `become*` são específicas da tarefa de propagação do arquivo `sudoers`, e não se aplicam à tarefa de ajuste de expiração de conta do usuário `root`. Ao invés de repetir o bloco `become + become_user + become_method` duas vezes (ou mais, dependendo do

número de tarefas que queremos realizar), o melhor caminho de ação é remover esse bloco do arquivo de *tasks* e inseri-lo no arquivo de configuração da *role*.

Vamos por partes. Primeiro, vamos remover as linhas `become*` do arquivo `/home/ansible/roles/sudoers/tasks/main.yml`:

```
$ t="$( mktemp )" ; \
    cat ~/roles/sudoers/tasks/main.yml | \
    awk 'gsub(/^ *become.*$/,""){printf $0;next;}1' > $t ; \
    mv $t ~/roles/sudoers/tasks/main.yml ; \
    unset t
```

```
$ cat ~/roles/sudoers/tasks/main.yml
---
- name: Propagate sudoers configuration
  copy:
    src: sudoers
    dest: /etc
    owner: root
    group: root
    mode: 0440

- name: Sets root account as expired
  user:
    name: root
    expires: 0
```

Agora, vamos inseri-las no arquivo de amarração da *role*:

```
$ sed -i '/\- hosts\: srv/a\  become: yes\n  become_user: root\n  become_method:
sudo' ~/srv.yml
```

```
$ cat ~/srv.yml
---
- hosts: srv
  become: yes
  become_user: root
  become_method: sudo
  roles:
    - sudoers
```

Observe atentamente a indentação do arquivo acima: caso a cópia do comando tenha inserido ou removido espaços acidentalmente, edite o arquivo diretamente e deixe-o **exatamente** como mostrado acima.

7. Pronto! Vamos disparar a *role* usando o `ansible-playbook` e verificar seu funcionamento:

```
$ ansible-playbook -i ~/hosts ~/srv.yml

PLAY [srv]
*********************************************************************************
***********************


TASK [Gathering Facts]
*********************************************************************************
***********

ok: [ns2]
ok: [ns1]
ok: [nfs]

TASK [sudoers : Propagate sudoers configuration]
*****************************************************************

ok: [ns2]
ok: [ns1]
ok: [nfs]

TASK [sudoers : Sets root account as expired]
*********************************************************************

changed: [ns2]
changed: [nfs]
changed: [ns1]

PLAY RECAP
*********************************************************************************
***********************

nfs                        : ok=3    changed=1    unreachable=0    failed=0
ns1                        : ok=3    changed=1    unreachable=0    failed=0
ns2                        : ok=3    changed=1    unreachable=0    failed=0
```

Terá funcionado? Vamos ver, usando o módulo *shell* do Ansible:

```
$ ansible -i ~/hosts srv -b --become-user=root --become-method=sudo -m shell -a
'chage -l root | grep "Conta expira"'
nfs | CHANGED | rc=0 >>
Conta expira                                          : jan 01, 1970

ns1 | CHANGED | rc=0 >>
Conta expira                                          : jan 01, 1970

ns2 | CHANGED | rc=0 >>
Conta expira                                          : jan 01, 1970
```

Excelente! Vamos fazer um teste *in loco* com o usuário `han` na máquina `ns1`:

```
$ hostname ; whoami
ns1
han
```

```
$ su -
Senha:
Sua conta expirou; entre em contato com o administrador do sistema
su: Falha de autenticação
```

6) Versionamento de configuração com git

À medida que fazemos novas adições de *tasks* e *roles* ao Ansible, pode surgir a necessidade de documentar as modificações sendo feitas, compartilhá-las com outros colegas de trabalho ou, em caso de mudanças incorretas, reverter o estado dos artefatos de configuração para uma versão anterior, conhecida e testada. De certa forma, as configuração que estamos fazendo em servidores do *datacenter* com o Ansible se parecem muito com o código-fonte de uma linguagem de programação — só que, ao invés de produzir programas, nosso código produz configurações e estados-alvo em servidores.

A ferramenta de controle de versão Git é certamente a mais utilizada mundialmente para cumprir as tarefas que delineamos acima. Criado em 2005 por Linus Torvalds (sim, o mesmo criador do kernel Linux, você não leu errado), o Git é um sistema de versionamento distribuído desenhado com o objetivo expresso de aprimorar as limitações dos sistemas de controle de versão mais usados à epoca: CVS e Subversion.

Iremos instalar um servidor Git na máquina `nfs`, e utilizá-lo como repositório para as configurações gerenciadas pelo Ansible. Assim, a cada nova modificação poderemos manter o registro de versão no repositório, comentar quais mudanças foram realizadas, bem como compartilhar o acesso ao código com outros colaboradores e aceitar suas submissões e *patches*, se desejado.

1. O primeiro passo é instalar o Git nas máquinas `nfs` e `client` — queremos usar o Ansible para fazer isso, mas o `sudo` não está configurado na máquina `client`... ainda.

Logue como o usuário `ansible` na máquina `client`.

```
$ hostname ; whoami
client
ansible
```

Como a máquina `client` não está no DNS, iremos usar seu endereço IP no arquivo de *hosts*. Adicione-o:

```
$ echo '192.168.42.2' >> ~/hosts
```

```
$ cat hosts
[srv]
ns1
ns2
nfs
192.168.42.2
```

Lembre-se que o `sudo` ainda não está configurado na máquina local, de modo que teremos que usar o `su` para escalar privilégios. Como o método padrão de escalada de privilégios especificado no arquivo `/home/ansible/srv.yml` é o `sudo`, teremos que configurar um *override* na linha de comando. Outro ponto: queremos executar a *role* apenas na máquina local, de forma que usaremos a opção `--limit`.

Esses fatores considerados, execute a *role*:

```
$ ansible-playbook -i hosts --limit='192.168.42.2' --ask-become-pass -e
ansible_become_method=su srv.yml
SUDO password:

PLAY [srv]
*************************************************************************************
**********************


TASK [Gathering Facts]
*************************************************************************************
***********

ok: [192.168.42.2]

TASK [sudoers : Propagate sudoers configuration]
*********************************************************************

changed: [192.168.42.2]

TASK [sudoers : Sets root account as expired]
*********************************************************************

changed: [192.168.42.2]

PLAY RECAP
*************************************************************************************
**********************

192.168.42.2               : ok=3    changed=2    unreachable=0    failed=0
```

Perfeito. Verifique que o `sudoers` foi configurado com sucesso, e que a conta do `root` foi desabilitada na máquina local:

```
$ sudo grep ansible /etc/sudoers
ansible   ALL=(ALL:ALL)    NOPASSWD: ALL
```

```
$ su -
Senha:
Sua conta expirou; entre em contato com o administrador do sistema
su: Falha de autenticação
```

2. Agora sim, vamos instalar o Git nas máquinas `nfs` e `client` usando o Ansible. Note o uso da opção `--limit` para especificar o escopo do comando:

```
$ ansible -i ~/hosts srv -b --become-user=root --become-method=sudo
--limit='nfs,192.168.42.2' -m shell -a 'apt-get install -y git'
 [WARNING]: Consider using the apt module rather than running apt-get.  If you need
to use command because apt is
insufficient you can add warn=False to this command task or set
command_warnings=False in ansible.cfg to get rid of this message.

nfs | CHANGED | rc=0 >>
(...)
Configurando git (1:2.11.0-3+deb9u4) ...

192.168.42.2 | CHANGED | rc=0 >>
(...)
Configurando git (1:2.11.0-3+deb9u4) ...
```

Terá funcionado? Vamos ver:

```
$ ansible -i ~/hosts srv -m shell -a 'which git'
ns2 | FAILED | rc=1 >>
non-zero return code

nfs | CHANGED | rc=0 >>
/usr/bin/git

ns1 | FAILED | rc=1 >>
non-zero return code

192.168.42.2 | CHANGED | rc=0 >>
/usr/bin/git
```

Perfeito, o Git foi instalado com sucesso, e apenas nas máquinas objetivadas.

3. Iremos usar o Git em sua capacidade mais básica — via SSH, usando os diretórios *home* de cada usuário. É bastante possível fazer uma configuração muito mais sofisticada, configurar uma interface como o GitLab (https://about.gitlab.com/install/), mas para manter a simplicidade das atividades aqui realizadas não iremos fazer isso.

   Primeiro, crie um repositório vazio de nome `ansible` em seu diretório *home* local:

   ```
   $ git init ansible
   Initialized empty Git repository in /home/ansible/ansible/.git/
   ```

   Agora, clone-o para o formato *bare* (mantendo apenas os arquivos administrativos do Git) com o comando:

```
$ git clone --bare ansible ansible.git
Cloning into bare repository 'ansible.git'...
warning: You appear to have cloned an empty repository.
done.
```

```
$ ls -1d ansible*
ansible
ansible.git
```

Devemos copiar o diretório `/home/ansible/ansible.git` para o servidor Git, que será a máquina `nfs`. Use o comando `scp`:

```
$ scp -r ~/ansible.git/ ansible@nfs:~
description
100%   73   179.2KB/s   00:00
HEAD
100%   23    76.8KB/s   00:00
pre-receive.sample
100%  544     1.7MB/s   00:00
commit-msg.sample
100%  896     2.7MB/s   00:00
pre-applypatch.sample
100%  424     1.3MB/s   00:00
pre-push.sample
100% 1348     3.4MB/s   00:00
prepare-commit-msg.sample
100% 1239     3.1MB/s   00:00
update.sample
100% 3610     9.4MB/s   00:00
post-update.sample
100%  189   478.5KB/s   00:00
pre-rebase.sample
100% 4898    13.2MB/s   00:00
applypatch-msg.sample
100%  478     1.3MB/s   00:00
pre-commit.sample
100% 1642     4.4MB/s   00:00
config
100%  113   367.7KB/s   00:00
exclude
100%  240   557.6KB/s   00:00
```

Pronto, o repositório Git remoto está configurado! Se quiser, remova os arquivos locais — iremos usar o repositório remoto a partir de agora:

```
$ rm -rf ~/ansible*
```

4. Para usar o repositório Git remoto, o primeiro passo é cloná-lo — por conveniência (e por já termos configurado o sistema de autenticação centralizado), iremos usar o protocolo SSH no acesso:

```
$ git clone ansible@nfs:/home/ansible/ansible.git
Cloning into 'ansible'...
warning: You appear to have cloned an empty repository.
```

O Git reporta que clonamos um repositório vazio... não por muito tempo! Copie os arquivos de trabalho do Ansible que usamos até aqui para dentro do repositório local:

```
$ mv ~/{hosts,roles,srv.yml} ~/ansible
```

Entre no repositório local e visualize seu estado com `git status`:

```
$ cd ~/ansible/ ; git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        hosts
        roles/
        srv.yml

nothing added to commit but untracked files present (use "git add" to track)
```

O Git reporta que os arquivos adicionados não estão sendo versionados. Para adicioná-los ao sistema de controle de versão, use o comando `git add`:

```
$ git add .
```

O que isso fez?

```
$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   hosts
        new file:   roles/sudoers/README.md
        new file:   roles/sudoers/defaults/main.yml
        new file:   roles/sudoers/files/sudoers
        new file:   roles/sudoers/handlers/main.yml
        new file:   roles/sudoers/meta/main.yml
        new file:   roles/sudoers/tasks/main.yml
        new file:   roles/sudoers/tests/inventory
        new file:   roles/sudoers/tests/test.yml
        new file:   roles/sudoers/vars/main.yml
        new file:   srv.yml
```

Agora sim, todos os arquivos foram adicionados ao sistema de versionamento. Para confirmar essas mudanças temos que realizar um *commit*, ou confirmação, das alterações — antes disso, no entanto, é recomendável que configuremos nosso nome de usuário e e-mail no Git, para evitar *warnings* futuros:

```
$ git config user.name Ansible
```

```
$ git config user.email ansible@intnet
```

Pronto, vamos ao *commit*. É sempre recomendável incluir uma mensagem explicativa junto com o *commit* (com a opção `-m`) indicando que alterações foram feitas naquela versão, criando um histórico do repositório.

```
$ git commit -m 'Importacao inicial, role sudoers adicionada'
[master (root-commit) 0c673e4] Importacao inicial, role sudoers adicionada
 11 files changed, 180 insertions(+)
 create mode 100644 hosts
 create mode 100644 roles/sudoers/README.md
 create mode 100644 roles/sudoers/defaults/main.yml
 create mode 100644 roles/sudoers/files/sudoers
 create mode 100644 roles/sudoers/handlers/main.yml
 create mode 100644 roles/sudoers/meta/main.yml
 create mode 100644 roles/sudoers/tasks/main.yml
 create mode 100644 roles/sudoers/tests/inventory
 create mode 100644 roles/sudoers/tests/test.yml
 create mode 100644 roles/sudoers/vars/main.yml
 create mode 100644 srv.yml
```

O passo final consiste em enviar as alterações locais para o repositório remoto através do comando `git push`:

```
$ git push
Counting objects: 22, done.
Compressing objects: 100% (10/10), done.
Writing objects: 100% (22/22), 3.27 KiB | 0 bytes/s, done.
Total 22 (delta 0), reused 0 (delta 0)
To nfs:/home/ansible/ansible.git
 * [new branch]      master -> master
```

Pronto, todos os dados foram enviados para o servidor `nfs` e estão — espera-se — seguros. Será mesmo? Vamos testar: retorne ao seu diretório *home* e remova todos os arquivos do repositório local:

```
$ cd ~ ; rm -rf ~/ansible
```

```
$ ls
scripts
```

Tragédia! Perdemos todo o trabalho realizado com o Ansible até aqui! Ou... perdemos mesmo? Tente clonar o repositório remoto a partir da máquina `nfs`:

```
$ git clone ansible@nfs:/home/ansible/ansible.git
Cloning into 'ansible'...
remote: Counting objects: 22, done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 22 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (22/22), done.
```

```
$ ls ~/ansible/
hosts  roles  srv.yml
```

Ufa, está tudo ali ainda. Se quiser verificar o registro dos *commits* anteriores, confirmando que nossas modificações anteriores foram de fato salvas no repositório remoto, basta usar o comando `git log` dentro do repositório local:

```
$ cd ~/ansible/ ; git log
commit 0c673e48dcdc7aaf2bd6738c8033c33815f10cc6
Author: Ansible <ansible@intnet>
Date:   Thu Nov 15 02:04:14 2018 -0200

    Importacao inicial, role sudoers adicionada
```

Sessão 6: Registro e correlacionamento de eventos

A gestão de registros de eventos em sistema computacionais é com certeza um dos temas mais difíceis — e importantes — de se tratar corretamente em um *datacenter* de grande porte. Imagine a seguinte situação: você quer monitorar todos os eventos acontecendo em um servidor web de alto tráfego em sua organização, buscando por anomalias; para isso, você monitora o log de acesso do servidor web com um comando como `tail -f`, e observa as mensagens voando em alta velocidade pela tela — após alguns poucos segundos fica claro que o enorme volume de mensagens é impossível de ser processado por um ser humano... são eventos demais! Agora, multiplique esse desafio por múltiplos servidores e máquinas virtuais no *datacenter*, cada qual com uma infinidade de serviços instalados: como dar conta desse trabalho?

Ferramentas SIEM (do inglês, *Security Information and Event Management*) são soluções de software que permitem que os eventos gerados por diversas aplicações de segurança (tais como firewalls, *proxies*, sistemas de prevenção a intrusão e antivírus) sejam coletados, normalizados, armazenados e correlacionados; essa gestão possibilita uma rápida identificação e resposta aos incidentes. Os SIEMs combinam ferramentas do tipo SIM (*Security Information Management*) e SEM (*Security Event Manager*) — enquanto ferramentas SEM oferecem monitoramento em tempo real dos eventos de segurança, coletando e agregando os dados (com resposta automática em alguns casos), ferramentas SIM oferecem análise histórica dos eventos de segurança, também coletando e correlacionando os eventos, porém não em tempo real; isso permite consultas mais complexas ao repositório.

Dentre as diversas funcionalidades desejáveis em ferramentas SIEM para auxiliar a gestão de logs corporativos, destacamos algumas:

- Acesso em tempo real, centralizado e consistente a todos os logs e eventos de segurança, independente do tipo de tecnologia e fabricante.
- Correlação de logs de tecnologias heterôgeneas, conectando atributos comuns e/ou significativos entre as fontes, de modo a transformar os dados em informação útil.
- Identificação de comportamentos, incidentes, fraudes, anomalias e quebras de *baseline* de segurança.
- Alertas e notificações que podem ser disparadas automaticamente no caso de não conformidade com as políticas de segurança e/ou normas regulatórias, ou ainda, de acordo com as regras de negócio pré-estabelecidas.
- Emissão de relatórios sofisticados sobre as condições de segurança do ambiente para equipes de SOC (*Security Operations Center*) auditoria ou resposta a incidentes.
- Retenção e indexação a longo prazo dos dados possibilitando posterior análise forense.

Nesta sessão iremos instalar e configurar o Graylog, uma das mais populares ferramentas SIEM *open-source* para armazenamento e correlação de eventos. Iremos integrá-lo com o RSyslog já instalado em nossos servidores, bem como com o sistema de autenticação centralizado via OpenLDAP. Juntamente com o Graylog, será necessário configurar um serviço NTP (*Network Time Protocol*) para manter a hora dos servidores sincronizada — a ferramenta OpenNTPD, do OpenBSD,

é uma alternativa simples e segura para solucionar esse problema. Finalmente, faremos também uso da ferramenta Snoopy para registrar comandos digitados pelos usuários no terminal, permitindo análise posterior e garantia de não-repúdio nos acessos remotos via SSH.

1) Topologia desta sessão

A figura abaixo mostra a topologia de rede que será utilizada nesta sessão, com as máquinas relevantes em destaque.

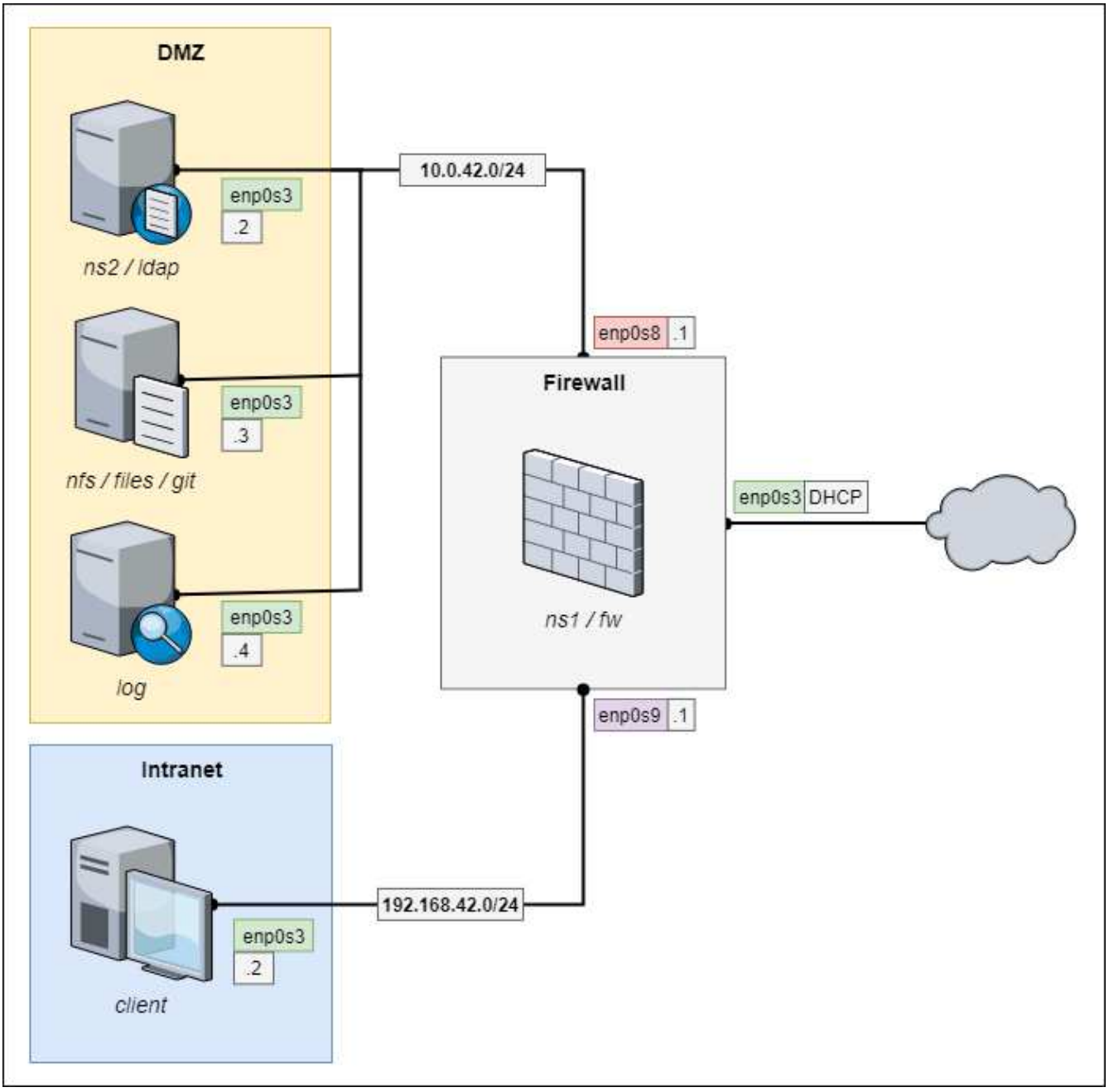

Figura 30. Topologia de rede desta sessão

Temos uma nova máquina, a saber:

- `log`, atuando como concentrador de logs dos servidores do *datacenter* e servidor NTP. Endereço IP 10.0.42.4/24.

    1. Os únicos registros DNS novos a serem criado são mapeamentos direto e reverso para a

máquina `log` — para fazer os ajustes, acesse a máquina `ns1` como o usuário `root`:

```
# hostname ; whoami
ns1
root
```

Edite o arquivo de zonas `/etc/nsd/zones/intnet.zone`, inserindo uma entrada A para a máquina `log`, como se segue. **Não se esqueça** de incrementar o valor do serial no topo do arquivo!

```
# nano /etc/nsd/zones/intnet.zone
(...)
```

```
# grep log /etc/nsd/zones/intnet.zone
log     IN    A                 10.0.42.4
```

Faça o mesmo para o arquivo de zona reversa:

```
# nano /etc/nsd/zones/10.0.42.zone
```

```
# grep log /etc/nsd/zones/10.0.42.zone
4       IN   PTR               log.intnet.
```

Assine os arquivos de zona usando o *script* criado anteriormente:

```
# bash /root/scripts/signzone-intnet.sh
reconfig start, read /etc/nsd/nsd.conf
ok
ok
ok
ok removed 6 rrsets, 4 messages and 0 key entries
```

Verifique a criação das entradas usando o comando `dig`:

```
# dig log.intnet +short
10.0.42.4
```

```
# dig -x 10.0.42.4 +short
log.intnet.
```

2) Criação da VM de gestão de logs

1. Como visualizado na topologia que abre esta sessão, teremos uma máquina dedicada para gestão de logs e serviços auxiliares, como o NTP. Devemos começar clonando a máquina `debian-template` e criar uma nova, que chamaremos de `log`. Essa máquina estará conectada a uma única rede *host-only*, com o mesmo nome que foi alocado para a interface de rede da máquina virtual `ns1`, configurada durante a sessão 2, que está conectada à DMZ. O IP da máquina será 10.0.42.4/24.

   Concluída a clonagem, ligue a máquina e faça login como o usuário `root`. Em seguida, use o script `/root/scripts/changehost.sh` para fazer a configuração automática:

   ```
   # hostname ; whoami
   debian-template
   root
   ```

   ```
   # bash ~/scripts/changehost.sh -h log -i 10.0.42.4 -g 10.0.42.1
   Signing ssh_host_ecdsa_key.pub key...
   Signing ssh_host_ed25519_key.pub key...
   Signing ssh_host_rsa_key.pub key...
   Configuring host key trust...
   Configuring user key trust...
   All done!
   ```

   ```
   $ ip addr show label 'enp0s*' | grep 'inet ' | awk '{print $2,$NF}' ; hostname ;
   whoami
   10.0.42.4/24 enp0s3
   log
   root
   ```

2. Agora, vamos configurar funcionamento do `sudo` na máquina `log`. Acesse a máquina `client` como o usuário `ansible`:

   ```
   $ hostname ; whoami
   client
   ansible
   ```

   Insira a máquina `log` no inventário gerenciado pelo Ansible:

   ```
   $ echo log >> ~/ansible/hosts
   ```

   Execute a *role* `sudoers`, lembrando-se de limitar o escopo à máquina `log` e alterando o método de escalação de privilégio para o `su` (já que o `sudo` não foi configurado ainda):

```
$ ansible-playbook -i ~/ansible/hosts -l log -Ke ansible_become_method=su
~/ansible/srv.yml
SUDO password:

PLAY [srv]
**************************************************************************************
**********************


TASK [Gathering Facts]
**************************************************************************************
***********

ok: [log]

TASK [sudoers : Propagate sudoers configuration]
*********************************************************************

changed: [log]

TASK [sudoers : Sets root account as expired]
**********************************************************************

changed: [log]

PLAY RECAP
**************************************************************************************
**********************

log                        : ok=3    changed=2    unreachable=0    failed=0
```

Fácil, não?

3. Note que fizemos uma alteração, ainda que ligeira, ao repositório de configuração do Ansible: adicionamos a máquina `log` ao grupo `srv` no inventário:

```
$ cd ~/ansible/ ; git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   hosts

no changes added to commit (use "git add" and/or "git commit -a")
```

```
$ git diff hosts
diff --git a/hosts b/hosts
index f81ed68..be6cf78 100644
--- a/hosts
+++ b/hosts
@@ -3,3 +3,4 @@ ns1
 ns2
 nfs
 192.168.42.2
+log
```

Vamos adicionar essa mudança ao repositório, efetuar o *commit* e publicar as mudanças para o repositório remoto:

```
$ git add hosts
```

```
$ git commit -m 'Maquina log adicionada ao inventario do Ansible'
[master 24bcab5] Maquina log adicionada ao inventario do Ansible
 1 file changed, 1 insertion(+)
```

```
$ git push
Counting objects: 3, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 350 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To nfs:/home/ansible/ansible.git
   0c673e4..24bcab5  master -> master
```

3) Ajuste das regras de firewall para o NTP

Iremos configurar a seguir um servidor NTP na máquina `log` para atuar como servidor de hora para os demais servidores do *datacenter*. O protocolo NTP opera por padrão na porta 123/UDP, para a qual não fizemos quaisquer liberações até aqui — caso não façamos a inserção de novas regras no firewall da rede, o serviço não funcionará.

Quais são os acessos a configurar? Vejamos:

a. A máquina `log` deve conseguir consultar servidores NTP na Internet, exigindo regras de acesso na *chain* FORWARD da tabela `filter` e na *chain* POSTROUTING da tabela `nat`. Poderíamos limitar o conjunto de IPs de destino, mas por simplicidade iremos liberar a conexão com qualquer *host* remoto.

b. Máquinas na DMZ devem conseguir acessar a máquina `log` na porta 123/UDP. Como essas máquinas estão todas na mesma sub-rede, os acessos não passam pelo firewall e nenhuma configuração adicional se faz necessária.

c. Máquinas na Intranet devem conseguir acessar a máquina `log` na porta 123/UDP. Como esse tráfego passa **através do** firewall, devemos inserir a regra na *chain* FORWARD.

1. Vamos lá: acesse a máquina `ns1` como o usuário `root`:

```
# hostname ; whoami
ns1
root
```

Para atender o requisito (a), insira as regras:

```
# iptables -A FORWARD -s 10.0.42.4/32 -o enp0s3 -p udp -m udp --dport 123 -j ACCEPT
```

```
# iptables -t nat -A POSTROUTING -s 10.0.42.4/32 -o enp0s3 -p udp -m udp --dport
123 -j MASQUERADE
```

Para o requisito (b), nenhuma ação é necessária. Finalmente, para o requisito (c) insira a regra a seguir:

```
# iptables -A FORWARD -s 192.168.42.0/24 -d 10.0.42.4/32 -p udp -m udp --dport 123
-j ACCEPT
```

Não se esqueça de gravar as novas regras na configuração permanente do firewall:

```
# /etc/init.d/netfilter-persistent save
[....] Saving netfilter rules...run-parts: executing /usr/share/netfilter-
persistent/plugins.d/15-ip4tables save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
done.
```

4) Configuração do NTP

Vamos ao NTP. Como estabelecido, a máquina `log` atuará como o servidor NTP da rede, mas quais serão seus clientes? Além dos servidores do *datacenter*, também iremos configurar a máquina `client` para consultar o servidor de hora na máquina `log`. Ao invés de instalar e configurar o *daemon* do OpenNTPD em cinco (!!) máquinas diferentes, temos aqui uma excelente oportunidade para colocar nossos recém-adquiridos conhecimentos com o Ansible à prova.

1. Acesse a máquina `client` como o usuário `ansible`:

```
$ hostname ; whoami
client
ansible
```

2. Para não termos que criar duas *roles* específicas — uma para gestão de servidores NTP, e outra para clientes — iremos usar a funcionalidade de grupos no inventário do Ansible para categorizar as máquinas-alvo. Edite o arquivo `/home/ansible/ansible/hosts`, deixando-o com o conteúdo a seguir:

```
$ nano ~/ansible/hosts
(...)
```

```
$ cat ~/ansible/hosts
[srv]
ns1
ns2
nfs
192.168.42.2
log

[ntp_server]
log
```

Note que incluímos a máquina `log` no novo grupo `ntp_server`, que conterá os servidores NTP da rede, deixando implícito que as demais máquinas atuarão como clientes NTP, como veremos a seguir.

Vamos agora criar uma nova *role* para gerenciar a instalação e configuração do OpenNTPD, com o nome `ntp`. Crie-a:

```
$ ansible-galaxy init --init-path=/home/ansible/ansible/roles ntp
```

3. O próximo passo consiste em editar os arquivos de configuração da *role*. Vamos começar com os *templates* de arquivos de configuração: crie o arquivo novo `/home/ansible/ansible/roles/ntp/templates/ntp_server.conf.j2` com o seguinte conteúdo:

```
1 listen on 127.0.0.1
2 listen on {{ ansible_default_ipv4.address }}
3 servers pool.ntp.br
```

O arquivo acima será o arquivo de configuração do OpenNTPD nos servidores NTP (no caso, a máquina `log`). As diretivas `listen` definem em quais interfaces de rede o *daemon* irá escutar: além da interface *loopback*, note que estamos utilizando a variável `ansible_default_ipv4.address` — ela contém o endereço IPv4 da interface de saída padrão da

máquina sendo configurada (10.0.42.4, no caso da VM `log`). A diretiva `servers` simplesmente define quais servidores remotos na Internet serão consultados para descobrir a hora correta.

Vamos agora para o *template* do arquivo de configuração dos clientes: crie o arquivo novo `/home/ansible/ansible/roles/ntp/templates/ntp_client.conf.j2` com o seguinte conteúdo:

```
1 server {{ ntpsrv }}
```

Temos apenas uma diretiva: `server`, que define o servidor remoto a ser consultado. Note que estamos usando uma variável para configurar esse campo, `ntpsrv`, que será definida a seguir.

4. Que variáveis são essas? Edite o arquivo `/home/ansible/ansible/roles/ntp/vars/main.yml` com o seguinte conteúdo:

```
1 ---
2 ntpsrv: "{{ groups['ntp_server'][0] }}.intnet"
3 fname: "{{ 'ntp_server' if 'ntp_server' in group_names else 'ntp_client' }}"
```

Primeiro, `ntpsrv`: essa variável contém o servidor NTP sendo configurado no momento — para obter seu valor, consultamos no arquivo de inventário qual grupo possui o nome `ntp_server`, e atribuímos a ela o nome do primeiro *host* desse grupo (a máquina `log`).

Depois `fname`: esse variável define qual nome de arquivo será usado para configurar um *host* — se a máquina pertencer ao grupo `ntp_server`, então o valor da variável será também `ntp_server`. Do contrário, ela valerá `ntp_client`.

5. Vamos às tarefas da *role*. Edite o arquivo `/home/ansible/ansible/roles/ntp/tasks/main.yml` com o seguinte conteúdo:

```
 1 ---
 2 - name: Install OpenNTPD
 3   apt:
 4     name: openntpd
 5     state: present
 6     update_cache: true
 7
 8 - name: Copy OpenNTPD configuration
 9   template:
10     src: '{{ fname }}.conf.j2'
11     dest: /etc/openntpd/ntpd.conf
12     owner: root
13     group: root
14     mode: 0644
15   notify:
16     - Restart OpenNTPD
```

Temos duas tarefas:

- *Install OpenNTPD* se encarrega de instalar o pacote do OpenNTPD, usando o módulo `apt` do Ansible. Antes da instalação, atualizamos a *cache* dos repositórios de forma análoga ao `apt-get update`.
- *Copy OpenNTPD configuration* copia o arquivo de configuração apropriado para `/etc/openntpd/ntpd.conf`. Note que o arquivo-origem é definido a partir da variável `fname`, que discutimos no passo anterior. O arquivo é ajustado para usuário-dono `root` e grupo-dono `root`, e permissões `rw-r—r--`. Ao final da cópia, invocamos um *handler* para reiniciar o OpenNTPD.

6. Que *handler* é esse? Edite o arquivo `/home/ansible/ansible/roles/ntp/handlers/main.yml` com o seguinte conteúdo:

```
1 ---
2 - name: Restart OpenNTPD
3   service:
4     name: openntpd
5     state: restarted
```

   O *handler* acima simplesmente utiliza o módulo `service` do Ansible para garantir que o *daemon* `openntpd` esteja com o estado "reiniciado" após sua execução.

7. A *role* está pronta! Para executá-la, vamos editar o arquivo `/home/ansible/ansible/srv.yml`:

```
$ echo '    - ntp' >> ~/ansible/srv.yml
```

```
$ cat ~/ansible/srv.yml
---
- hosts: srv
  become: yes
  become_user: root
  become_method: sudo
  roles:
    - sudoers
    - ntp
```

   Note que agora ao invocarmos o arquivo `/home/ansible/ansible/srv.yml` iremos não apenas configurar o `sudo` de forma automática, mas também instalar e configurar o OpenNTPD nas máquinas-alvo. Basta, é claro, que editemos o arquivo de inventário de antemão.

8. Ufa! Agora sim, vamos testar:

```
$ ansible-playbook -i ~/ansible/hosts ~/ansible/srv.yml

PLAY [srv]
*********************************************************************************
**********************

(...)

TASK [ntp : Install OpenNTPD]
*********************************************************************************
****

changed: [192.168.42.2]
changed: [log]
changed: [nfs]
changed: [ns1]
changed: [ns2]

TASK [ntp : Copy OpenNTPD configuration]
*********************************************************************************

changed: [ns2]
changed: [ns1]
changed: [log]
changed: [nfs]
changed: [192.168.42.2]

RUNNING HANDLER [ntp : Restart OpenNTPD]
*********************************************************************************

changed: [log]
changed: [nfs]
changed: [ns1]
changed: [ns2]
changed: [192.168.42.2]

PLAY RECAP
*********************************************************************************
**********************

192.168.42.2               : ok=6    changed=4    unreachable=0    failed=0
log                        : ok=6    changed=4    unreachable=0    failed=0
nfs                        : ok=6    changed=4    unreachable=0    failed=0
ns1                        : ok=6    changed=4    unreachable=0    failed=0
ns2                        : ok=6    changed=4    unreachable=0    failed=0
```

Terá nossa "magia" funcionado? A princípio, o OpenNTPD foi instalado e configurado corretamente nas máquinas-alvo. Vamos checar:

```
$ ansible -i ~/ansible/hosts srv -b -m shell -a 'which openntpd ; ps auxmw | grep
'[/]usr/sbin/ntpd' ; cat /etc/openntpd/ntpd.conf'
log | CHANGED | rc=0 >>
/usr/sbin/openntpd
root      9286  0.0  0.0   7564   124 ?        -    17:32   0:00 /usr/sbin/ntpd -f
/etc/openntpd/ntpd.conf
listen on 127.0.0.1
listen on 10.0.42.4
servers pool.ntp.br

ns1 | CHANGED | rc=0 >>
/usr/sbin/openntpd
root      7163  0.0  0.0   7564   124 ?        -    17:32   0:00 /usr/sbin/ntpd -f
/etc/openntpd/ntpd.conf
server log.intnet

ns2 | CHANGED | rc=0 >>
/usr/sbin/openntpd
root      7117  0.0  0.0   7564   124 ?        -    17:32   0:00 /usr/sbin/ntpd -f
/etc/openntpd/ntpd.conf
server log.intnet

nfs | CHANGED | rc=0 >>
/usr/sbin/openntpd
root      6967  0.0  0.0   7564   120 ?        -    17:32   0:00 /usr/sbin/ntpd -f
/etc/openntpd/ntpd.conf
server log.intnet

192.168.42.2 | CHANGED | rc=0 >>
/usr/sbin/openntpd
root     14410  0.0  0.0   7564   120 ?        -    17:32   0:00 /usr/sbin/ntpd -f
/etc/openntpd/ntpd.conf
server log.intnet
```

Uhm... o OpenNTPD está instalado em todas as máquinas, como verificado pelo `which`. O processo do `ntpd` também está ativo, como verificado via `ps`. E, finalmente, os arquivos de configuração de todas as máquinas cliente possui apenas a linha `server log.intnet`, como esperado, e o arquivo de configuração da máquina `log` corresponde ao que objetivávamos.

Muito legal, não é mesmo?

9. Muitas alterações em nosso repositório local — vamos fazer um *commit* e enviá-las:

```
$ cd ~/ansible/
```

```
$ git add .
```

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   hosts
        new file:   roles/ntp/README.md
        new file:   roles/ntp/defaults/main.yml
        new file:   roles/ntp/handlers/main.yml
        new file:   roles/ntp/meta/main.yml
        new file:   roles/ntp/tasks/main.yml
        new file:   roles/ntp/templates/ntp_client.conf.j2
        new file:   roles/ntp/templates/ntp_server.conf.j2
        new file:   roles/ntp/tests/inventory
        new file:   roles/ntp/tests/test.yml
        new file:   roles/ntp/vars/main.yml
        modified:   srv.yml
```

```
$ git commit -m 'Adicionada role para instalacao e configuracao do servico NTP'
[master 17486a0] Adicionada role para instalacao e configuracao do servico NTP
 12 files changed, 146 insertions(+)
 create mode 100644 roles/ntp/README.md
 create mode 100644 roles/ntp/defaults/main.yml
 create mode 100644 roles/ntp/handlers/main.yml
 create mode 100644 roles/ntp/meta/main.yml
 create mode 100644 roles/ntp/tasks/main.yml
 create mode 100644 roles/ntp/templates/ntp_client.conf.j2
 create mode 100644 roles/ntp/templates/ntp_server.conf.j2
 create mode 100644 roles/ntp/tests/inventory
 create mode 100644 roles/ntp/tests/test.yml
 create mode 100644 roles/ntp/vars/main.yml
```

```
$ git push
Counting objects: 19, done.
Compressing objects: 100% (13/13), done.
Writing objects: 100% (19/19), 1.66 KiB | 0 bytes/s, done.
Total 19 (delta 1), reused 0 (delta 0)
To nfs:/home/ansible/ansible.git
   24bcab5..17486a0  master -> master
```

5) Registro de comandos digitados com SnoopyLog

1. Vamos agora instalar o Snoopy (https://github.com/a2o/snoopy), um programa bastante simples que serve para registrar todos os comandos digitados no terminal, por qualquer usuário, nos logs do sistema. Assim, é possível gerar uma trilha de auditoria de comandos para realização de

análise forense e garantia de não-repúdio em caso de incidentes.

Assim como no caso no NTP, vamos automatizar a instalação e configuração desse pacote como uma *baseline* de segurança em todos os servidores (atuais e futuros) do *datacenter* usando o Ansible. Crie a *role*:

```
$ ansible-galaxy init --init-path=/home/ansible/ansible/roles snoopy
```

2. Felizmente, a configuração do Snoopy é bem mais simples que a do OpenNTPD, já que não temos servidores/clientes distintos no inventário. Edite o arquivo `/home/ansible/ansible/roles/snoopy/tasks/main.yml` com o seguinte conteúdo:

```
1  ---
2  - name: Install Snoopy
3    apt:
4      name: snoopy
5      state: present
6      update_cache: true
7
8  - name: Configure ld.so.preload
9    lineinfile:
10     path: /etc/ld.so.preload
11     line: '/lib/snoopy.so'
12     insertafter: EOF
13     create: yes
```

   Novamente, usamos o módulo `apt` para instalar o pacote `snoopy`. Em seguida, adicionamos ao final do arquivo `/etc/ld.so.preload` (criando-o se ele não existir) a linha `/lib/snoopy.so`, garantindo que a biblioteca compartilhada do Snoopy será carregada antes da execução de qualquer binário no sistema — garantindo assim que os comandos serão de fato logados.

3. Adicione a *role* do Snoopy ao arquivo `/home/ansible/ansible/srv.yml`:

```
echo '    - snoopy' >> ~/ansible/srv.yml
```

4. Finalmente, execute a *role*:

```
$ ansible-playbook -i ~/ansible/hosts ~/ansible/srv.yml

PLAY [srv]
**********************************************************************************
**********************

(...)

TASK [snoopy : Install Snoopy]
**********************************************************************************
***

changed: [ns2]
changed: [ns1]
changed: [nfs]
changed: [log]
changed: [192.168.42.2]

TASK [snoopy : Configure ld.so.preload]
**************************************************************************

changed: [ns1]
changed: [ns2]
changed: [nfs]
changed: [log]
changed: [192.168.42.2]

PLAY RECAP
**********************************************************************************
**********************

192.168.42.2               : ok=7    changed=3    unreachable=0    failed=0
log                        : ok=7    changed=3    unreachable=0    failed=0
nfs                        : ok=7    changed=3    unreachable=0    failed=0
ns1                        : ok=7    changed=3    unreachable=0    failed=0
ns2                        : ok=7    changed=3    unreachable=0    failed=0
```

5. Terá funcionado? Logue via SSH na máquina `ns1`, ainda como o usuário `ansible`:

```
$ hostname ; whoami
client
ansible
```

```
$ ssh ns1
Linux ns1 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

Last login: Fri Nov 16 00:16:07 2018 from 192.168.42.2
ansible@ns1:~$
```

Use o comando `sudo -i` para escalar privilégio:

```
$ sudo -i
```

Agora, verifique o conteúdo do arquivo `/var/log/auth.log` — de fato, procure por linhas que possuam a *string* `snoopy`:

```
# grep 'ns1 snoopy' /var/log/auth.log | grep -v '(none)'
Nov 16 00:17:07 ns1 snoopy[10150]: [uid:10005 sid:10150 tty:/dev/pts/0
cwd:/home/ansible filename:/bin/bash]: -bash Nov 16 00:17:07 ns1 snoopy[10152]:
[uid:10005 sid:10150 tty:/dev/pts/0 cwd:/home/ansible filename:/usr/bin/id]: id -u
Nov 16 00:17:07 ns1 snoopy[10154]: [uid:10005 sid:10150 tty:/dev/pts/0
cwd:/home/ansible filename:/bin/ls]: ls /etc/bash_completion.d
Nov 16 00:17:07 ns1 snoopy[10156]: [uid:10005 sid:10150 tty:/dev/pts/0
cwd:/home/ansible filename:/usr/bin/dircolors]: dircolors -b
Nov 16 00:17:07 ns1 snoopy[10158]: [uid:10005 sid:10150 tty:/dev/pts/0
cwd:/home/ansible filename:/bin/ls]: ls /etc/bash_completion.d
Nov 16 00:17:11 ns1 snoopy[10161]: [uid:10005 sid:10150 tty:/dev/pts/0
cwd:/home/ansible filename:/usr/bin/sudo]: sudo -i
Nov 16 00:17:11 ns1 snoopy[10162]: [uid:0 sid:10150 tty:/dev/pts/0 cwd:/root
filename:/bin/bash]: -bash
Nov 16 00:17:11 ns1 snoopy[10164]: [uid:0 sid:10150 tty:/dev/pts/0 cwd:/root
filename:/usr/bin/id]: id -u
Nov 16 00:17:11 ns1 snoopy[10167]: [uid:0 sid:10150 tty:/dev/pts/0 cwd:/root
filename:/bin/ls]: ls /etc/bash_completion.d
Nov 16 00:17:11 ns1 snoopy[10168]: [uid:0 sid:10150 tty:/dev/pts/0 cwd:/root
filename:/usr/bin/mesg]: mesg n
Nov 16 00:20:27 ns1 snoopy[10185]: [uid:0 sid:10150 tty:/dev/pts/0 cwd:/root
filename:/bin/grep]: grep ns1 snoopy /var/log/auth.log
```

Note como o Snoopy registrou todos os nossos comandos desde o login no sistema — desde o carregamente de opções do perfil do *shell* Bash, passando pela escalação de privilégio usando o `sudo`, e chegando até o próprio `grep` que acabamos de executar!

Agora, se seus usuários fizerem qualquer ação indevida nos servidores, ficará bem mais difícil negar o ocorrido! Especialmente com o uso do concentrador de logs e SIEM que instalaremos a seguir.

6. Como fizemos mais alterações nos repositórios locais, vamos enviá-las:

```
$ cd ~/ansible/ ; git add . ; git commit -m 'Adicionada role para instalacao e
configuracao do Snoopy logger' ; git push
[master ada4705] Adicionada role para instalacao e configuracao do Snoopy logger
 10 files changed, 130 insertions(+)
 create mode 100644 roles/snoopy/README.md
 create mode 100644 roles/snoopy/defaults/main.yml
 create mode 100644 roles/snoopy/handlers/main.yml
 create mode 100644 roles/snoopy/meta/main.yml
 create mode 100644 roles/snoopy/tasks/main.yml
 create mode 100644 roles/snoopy/tests/inventory
 create mode 100644 roles/snoopy/tests/test.yml
 create mode 100644 roles/snoopy/vars/main.yml
 create mode 100644 srv.retry
Counting objects: 16, done.
Compressing objects: 100% (8/8), done.
Writing objects: 100% (16/16), 1.34 KiB | 0 bytes/s, done.
Total 16 (delta 1), reused 0 (delta 0)
To nfs:/home/ansible/ansible.git
   17486a0..ada4705  master -> master
```

Já estamos construindo um histório interessante no repositório, não é mesmo? Confira com o `git log`:

```
$ git log
commit ada4705b62e53e0802c06fc75e67a89d83424467
Author: Ansible <ansible@intnet>
Date:   Fri Nov 16 00:29:32 2018 -0200

    Adicionada role para instalacao e configuracao do Snoopy logger

commit 17486a03a71405cb22737c026f068ac0a6a17384
Author: Ansible <ansible@intnet>
Date:   Fri Nov 16 00:03:36 2018 -0200

    Adicionada role para instalacao e configuracao do servico NTP

commit 24bcab569ee2f9504c54a65081f75f5cd5c400e5
Author: Ansible <ansible@intnet>
Date:   Thu Nov 15 16:12:51 2018 -0200

    Maquina log adicionada ao inventario do Ansible

commit 0c673e48dcdc7aaf2bd6738c8033c33815f10cc6
Author: Ansible <ansible@intnet>
Date:   Thu Nov 15 02:04:14 2018 -0200

    Importacao inicial, role sudoers adicionada
```

6) Instalação e configuração inicial do Graylog

Vamos proceder à instalação do SIEM *open-source* Graylog. Como a instalação será feita em uma única máquina (`log`), e realizaremos um grande número de passos, faremos o processo "à moda antiga" — via comandos diretos no terminal. Seguiremos os passos de instalação indicados na documentação oficial do Graylog, em http://docs.graylog.org/en/latest/pages/installation/os/debian.html .

1. Antes de mais nada desligue a VM `log`. O Graylog exige uma quantidade bastante significativa de recursos — na console do Virtualbox, acesse *Settings* > *System* > *Base Memory* e aumente-a para 4096 MB (4 GB). Em seguida, religue a VM `log` e acesse-a como o usuário `root`:

```
# hostname ; whoami
log
root
```

2. Agora, instale as dependências do Graylog com o comando a seguir:

```
# apt-get update ; apt-get install -y apt-transport-https openjdk-8-jre-headless
uuid-runtime pwgen
```

3. O próximo passo é a instalação do MongoDB, uma base de dados NoSQL *open-source* orientada ao armazenamento e gestão de documentos:

```
# apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
2930ADAE8CAF5059EE73BB4B58712A2291FA4AD5
Executing: /tmp/apt-key-gpghome.yMHAQrCigl/gpg.1.sh --keyserver
hkp://keyserver.ubuntu.com:80 --recv 2930ADAE8CAF5059EE73BB4B58712A2291FA4AD5
gpg: key 58712A2291FA4AD5: public key "MongoDB 3.6 Release Signing Key
<packaging@mongodb.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1
```

```
# echo "deb http://repo.mongodb.org/apt/debian stretch/mongodb-org/3.6 main" >
/etc/apt/sources.list.d/mongodb-org-3.6.list
```

```
# apt-get update ; apt-get install -y mongodb-org
```

Em seguida, inicie o serviço do MongoDB:

```
# systemctl daemon-reload
```

```
# systemctl enable mongod.service
Created symlink /etc/systemd/system/multi-user.target.wants/mongod.service →
/lib/systemd/system/mongod.service.
```

```
# systemctl restart mongod.service
```

4. Agora, vamos instalar o Elasticsearch, um *engine* de busca distribuída e *open-source* baseada na biblioteca Apache Lucene:

```
# wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | apt-key add -
OK
```

```
# echo "deb https://artifacts.elastic.co/packages/5.x/apt stable main" >
/etc/apt/sources.list.d/elastic-5.x.list
```

```
# apt-get update ; apt-get install -y elasticsearch
```

Temos que configurar o Elasticsearch para operar com o Graylog — para isso, basta informarmos um nome de *cluster* no arquivo de configuração `/etc/elasticsearch/elasticsearch.yml`:

```
# sed -i 's/^#\(cluster\.name:\).*/\1 graylog/'
/etc/elasticsearch/elasticsearch.yml
```

Feito isso, basta iniciar o serviço do Elasticsearch:

```
# systemctl daemon-reload
```

```
# systemctl enable elasticsearch.service
Synchronizing state of elasticsearch.service with SysV service script with
/lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable elasticsearch
Created symlink /etc/systemd/system/multi-user.target.wants/elasticsearch.service
→ /usr/lib/systemd/system/elasticsearch.service.
```

```
# systemctl restart elasticsearch.service
```

5. Finalmente, vamos instalar o Graylog:

```
# wget -q https://packages.graylog2.org/repo/packages/graylog-2.4-
repository_latest.deb
```

```
# dpkg -i graylog-2.4-repository_latest.deb ; rm graylog-2.4-repository_latest.deb
```

```
# apt-get update ; usermod -e -1 root ; apt-get install -y graylog-server ; usermod
-e 0 root
```

Agora, temos que configurar uma senha randômica para garantir a segurança do armazenamento de senhas dos usuários do Graylog, bem como uma senha de acesso admnistrativo para o Graylog (como de costume, iremos usar `rnpesr`). Os comandos a seguir irão realizar essas ações:

```
# SECRET=$(pwgen -s 96 1) ; sed -i -e 's/password_secret =.*/password_secret =
'$SECRET'/' /etc/graylog/server/server.conf ; unset SECRET
```

```
# PASSWORD=$(echo -n 'rnpesr' | shasum -a 256 | awk '{print $1}') ; sed -i -e
's/root_password_sha2 =.*/root_password_sha2 = '$PASSWORD'/'
/etc/graylog/server/server.conf ; unset PASSWORD
```

O Graylog utiliza a *timezone* UTC (*Universal Time Coordinated*) como padrão, que não é a mesma que utilizamos no Brasil. Assumindo que o curso está sendo realizado no fuso de Brasília, o comando a seguir irá ajustar a *timezone* corretamente:

```
# sed -i 's/^#\(root_timezone =\).*/\1 America\/Sao\_Paulo/'
/etc/graylog/server/server.conf
```

Finalmente, vamos iniciar o Graylog:

```
# systemctl daemon-reload
```

```
# systemctl enable graylog-server.service
Synchronizing state of graylog-server.service with SysV service script with
/lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable graylog-server
Created symlink /etc/systemd/system/multi-user.target.wants/graylog-server.service
→ /usr/lib/systemd/system/graylog-server.service.
```

```
# systemctl start graylog-server.service
```

6. Para não termos que configurar o Graylog escutando diretamente por conexões do mundo externo, iremos instalar o servidor HTTP Nginx para atuar como um *proxy* reverso, filtrando e repassando as conexões para o Graylog. Primeiro, instale o Nginx:

```
# apt-get install -y nginx
```

Remova o arquivo de configuração do website padrão do Nginx — iremos substituí-lo a seguir:

```
# rm /etc/nginx/sites-enabled/default
```

Crie o arquivo novo `/etc/nginx/sites-available/graylog` com o conteúdo a seguir. **IMPORTANTE:** ao editar o arquivo abaixo, substitua as duas ocorrências da *string* `NS1_ENP0S3_IPADDR` pelo endereço da interface `enp0s3` da sua máquina `ns1`.

```
 1 server
 2 {
 3   listen      80 default_server;
 4   listen      [::]:80 default_server ipv6only=on;
 5   server_name NS1_ENP0S3_IPADDR;
 6
 7   location /
 8     {
 9         proxy_set_header    Host $http_host;
10         proxy_set_header    X-Forwarded-Host $host;
11         proxy_set_header    X-Forwarded-Server $host;
12         proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
13         proxy_set_header    X-Graylog-Server-URL
http://NS1_ENP0S3_IPADDR:9080/api/;
14         proxy_pass          http://127.0.0.1:9000;
15     }
16 }
```

Suponha, por exemplo, que o endereço IP da interface `enp0s3` da sua máquina `ns1` seja 200.130.46.254; você poderia usar o comando `sed` para fazer as substituições de forma automática da seguinte forma:

```
# sed -i 's/NS1_ENP0S3_IPADDR/200.130.46.254/' /etc/nginx/sites-available/graylog
```

Agora, crie um link simbólico do arquivo recém-criado para o arquivo `/etc/nginx/sites-enabled/graylog`, habilitando-o no Nginx, como se segue:

```
# p=$PWD ; cd /etc/nginx/sites-enabled/ ; ln -s ../sites-available/graylog . ; cd
$p ; unset p
```

Finalmente, reinicie o Nginx:

```
# systemctl restart nginx
```

7) Ajuste das regras de firewall para o Graylog

O uso do Graylog exigirá alguns ajustes no firewall, a saber:

a. Iremos acessar a interface web do Graylog a partir da máquina física, usando o endereço IP público da máquina `ns1`. Para que consigamos atingir o Graylog, será necessário criar uma regra de DNAT na tabela `nat`, *chain* PREROUTING, fazendo o redirecionamento de endereço/porta, além de uma regra na tabela `filter`, *chain* FORWARD, correspondente. Especificamente, faremos o mapeamento da porta externa 9080/TCP para a porta interna 80/TCP.

b. Máquinas na DMZ devem conseguir acessar a máquina `log` na porta 5140/UDP, para envio dos logs locais usando o Rsyslog. Como essas máquinas estão todas na mesma sub-rede, os acessos não passam pelo firewall e nenhuma configuração adicional se faz necessária.

c. Máquinas na Intranet devem conseguir acessar a máquina `log` na porta 5140/UDP, para envio dos logs locais usando o Rsyslog. Como esse tráfego passa **através do** firewall, devemos inserir a regra na *chain* FORWARD.

1. Vamos lá: acesse a máquina `ns1` como o usuário `root`:

```
# hostname ; whoami
ns1
root
```

Para atender o requisito (a), insira as regras:

```
# iptables -t nat -A PREROUTING -i enp0s3 -p tcp -m tcp --dport 9080 -j DNAT --to
-destination 10.0.42.4:80
```

```
# iptables -A FORWARD -i enp0s3 -d 10.0.42.4/32 -p tcp -m tcp --dport 80 -j ACCEPT
```

Para o requisito (b), nenhuma ação é necessária. Finalmente, para o requisito (c) insira a regra a seguir:

```
# iptables -A FORWARD -s 192.168.42.0/24 -d 10.0.42.4/32 -p udp -m udp --dport 5140
-j ACCEPT
```

Grave as novas regras na configuração permanente do firewall:

```
# /etc/init.d/netfilter-persistent save
[....] Saving netfilter rules...run-parts: executing /usr/share/netfilter-
persistent/plugins.d/15-ip4tables save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
done.
```

8) Visualizando logs de máquinas no Graylog

1. Tudo pronto? Vamos acessar a interface do Graylog: em sua máquina física, abra o navegador e aponte-o para http://NS1_ENP0S3_IPADDR:9080 (substitua a *string* `NS1_ENP0S3_IPADDR` pelo endereço IP da interface `enp0s3` da sua máquina `ns1`). Você deverá ver a tela a seguir:

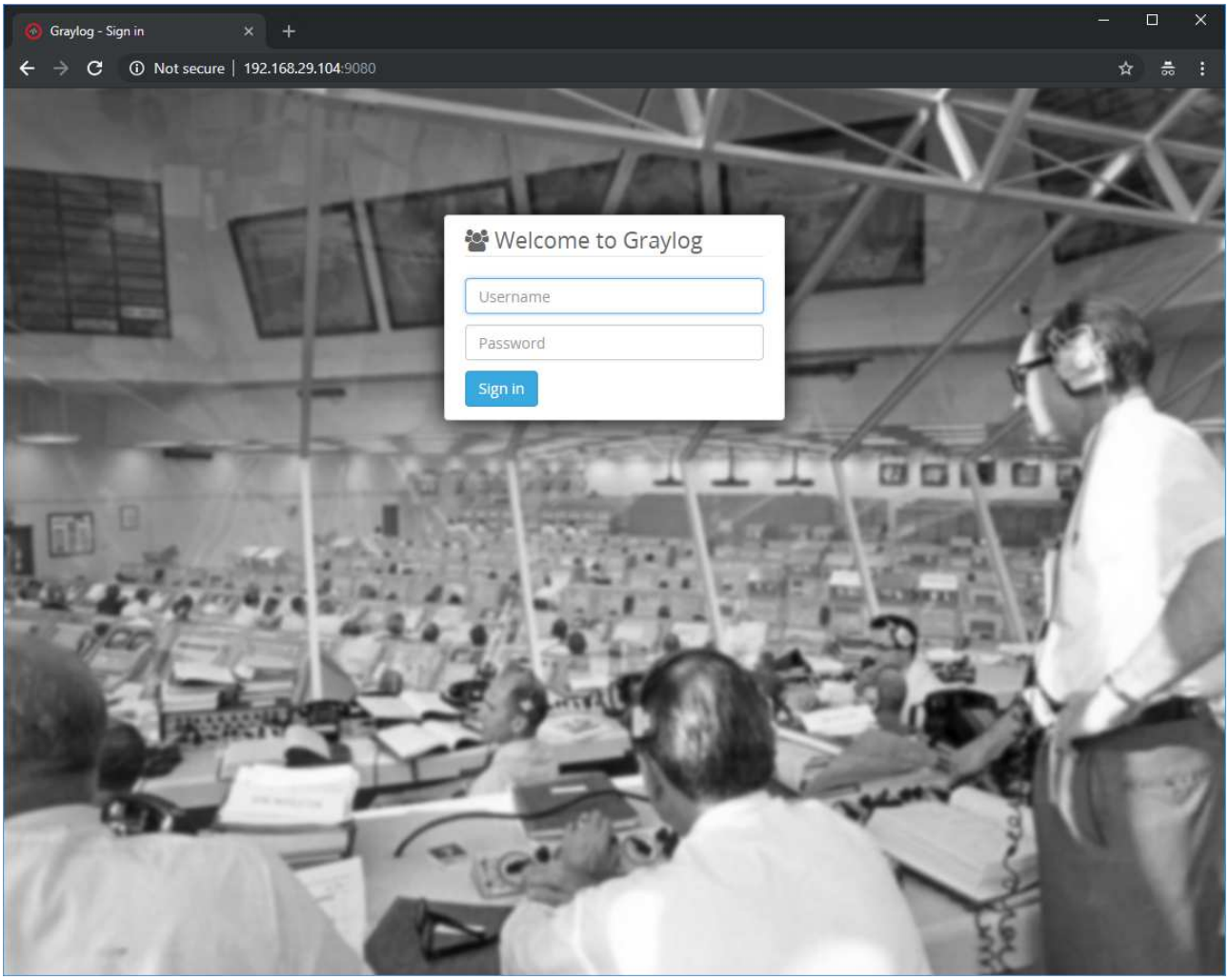

Figura 31. Tela de login do Graylog

   Faça login com o usuário `admin` e senha `rnpesr`, como definimos anteriormente.

2. Vamos fazer alguns ajustes iniciais. Acesse o menu *System* > *Indices* e clique em *Edit*. Na nova tela, configure o valor do campo *Index Shards* como 1, e em seguida clique em *Save* na base da página. Como estamos em um ambiente simulado, essa configuração irá auxiliar na economia de recursos da máquina.

3. Agora, acesse o menu *System* > *Inputs*. Na caixa *Select input* desça a barra de rolagem, selecione a opção *Syslog UDP* e clique em *Launch new input*.

Na nova janela, clique no campo *Select Node* e selecione a única opção disponível; no campo *Title* , escreva `syslog`; em *Bind address*, digite o endereço local da máquina, `10.0.42.4`; finalmente, no campo *Port* informe o valor `5140`. Sua tela deve ficar assim:

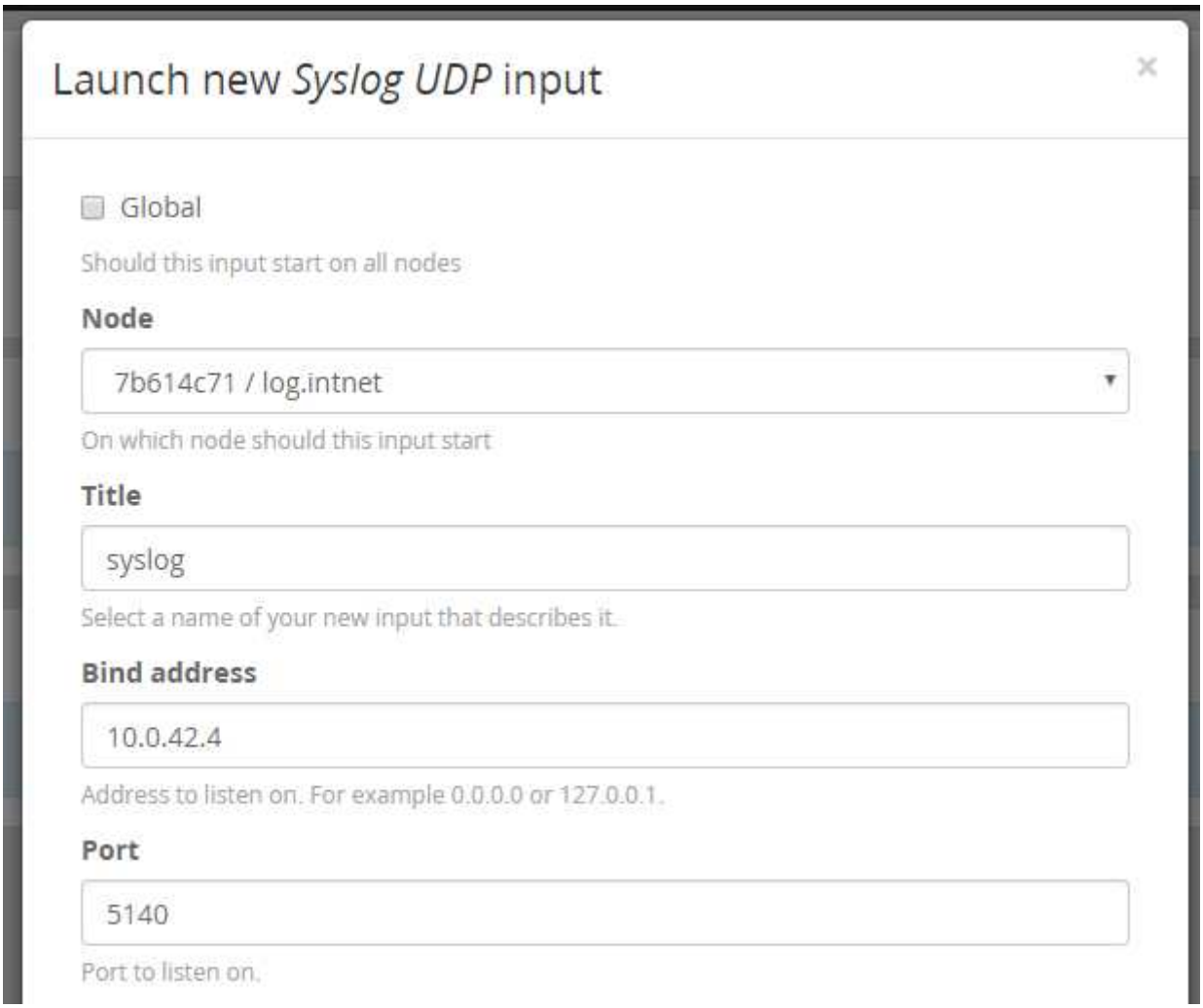

Figura 32. Criação de um input Syslog/UDP no Graylog

Na base da janela, clique em *Save*.

4. O *input* via Syslog está configurado; temos, agora, que configurar o envio de logs para esse *input*. Novamente, o Ansible é a ferramenta para a tarefa.

   Acesse a máquina `client` como o usuário `ansible`:

   ```
   $ hostname ; whoami
   client
   ansible
   ```

   Vamos criar uma *role* que envie todos os logs das máquinas do *datacenter* para o *input* que configuramos no Graylog:

   ```
   $ ansible-galaxy init --init-path /home/ansible/ansible/roles/ syslog
   - syslog was created successfully
   ```

Vamos usar uma técnica similar à que fizemos no caso do NTP: insira um novo grupo, `log_server`, no inventário do Ansible:

```
$ echo -e '\n[log_server]\nlog' >> ~/ansible/hosts
```

```
$ cat ~/ansible/hosts
[srv]
ns1
ns2
nfs
192.168.42.2
log

[ntp_server]
log

[log_server]
log
```

Agora, edite o arquivo `/home/ansible/ansible/roles/syslog/vars/main.yml` com o seguinte conteúdo:

```
1 ---
2 logsrv: "{{ groups['log_server'][0] }}.intnet"
```

Onde vamos chegar com isso? Edite o arquivo `/home/ansible/ansible/roles/syslog/tasks/main.yml` com o seguinte conteúdo e confira:

```
1 ---
2 - name: Configure centralized syslog
3   lineinfile:
4     path: /etc/rsyslog.d/99-graylog.conf
5     line: '*.* @{{ logsrv }}:5140;RSYSLOG_SyslogProtocol23Format'
6     insertafter: EOF
7     create: yes
```

A tarefa acima irá criar o arquivo `/etc/rsyslog.d/99-graylog.conf`, se inexistente, em todos os *hosts* aplicáveis e instruir o Rsyslog a enviar todos os logs para a máquina `logsrv` (que definimos no arquivo `vars`, acima) na porta 5140/UDP. Feito isso, chama-se um *handler* que reinicia o Rsyslog.

É claro, temos que criar o *handler*. Edite o arquivo `/home/ansible/ansible/roles/syslog/handlers/main.yml` com o seguinte conteúdo:

```
---
- name: Restart Rsyslog
  service:
    name: rsyslog
    state: restarted
```

O de sempre: usamos o módulo `service` para reiniciar o Rsyslog local. Adicione a nova *role* ao arquivo de amarração de inventário `/home/ansible/ansible/srv.yml`:

```
$ echo '    - syslog' >> ~/ansible/srv.yml
```

Vamos testar? Dispare a *role*:

```
$ ansible-playbook -i ~/ansible/hosts ~/ansible/srv.yml

PLAY [srv]
**********************************************************************************
**********************

(...)

TASK [syslog : Configure centralized syslog]
**********************************************************************

changed: [ns1]
changed: [ns2]
changed: [nfs]
changed: [192.168.42.2]
changed: [log]

RUNNING HANDLER [syslog : Restart Rsyslog]
**********************************************************************

changed: [nfs]
changed: [ns1]
changed: [ns2]
changed: [192.168.42.2]
changed: [log]

PLAY RECAP
**********************************************************************************
**********************

192.168.42.2               : ok=9    changed=3    unreachable=0    failed=0
log                        : ok=9    changed=3    unreachable=0    failed=0
nfs                        : ok=9    changed=3    unreachable=0    failed=0
ns1                        : ok=9    changed=3    unreachable=0    failed=0
ns2                        : ok=9    changed=3    unreachable=0    failed=0
```

5. Como de costume, não se esqueça de enviar as mudanças para o repositório remoto do Git:

```
$ cd ~/ansible/ ; git add . ; git commit -m 'Adicionada role para envio de logs
para o servidor Graylog' ; git push
[master b1e7a24] Adicionada role para envio de logs para o servidor Graylog
 10 files changed, 127 insertions(+)
 create mode 100644 roles/syslog/README.md
 create mode 100644 roles/syslog/defaults/main.yml
 create mode 100644 roles/syslog/handlers/main.yml
 create mode 100644 roles/syslog/meta/main.yml
 create mode 100644 roles/syslog/tasks/main.yml
 create mode 100644 roles/syslog/tests/inventory
 create mode 100644 roles/syslog/tests/test.yml
 create mode 100644 roles/syslog/vars/main.yml
Counting objects: 16, done.
Compressing objects: 100% (11/11), done.
Writing objects: 100% (16/16), 1.39 KiB | 0 bytes/s, done.
Total 16 (delta 3), reused 0 (delta 0)
To nfs:/home/ansible/ansible.git
   ada4705..b1e7a24  master -> master
```

6. Volte para o navegador em sua máquina física, e acesse a aba *Search*. Você deverá ver algo parecido com a imagem abaixo:

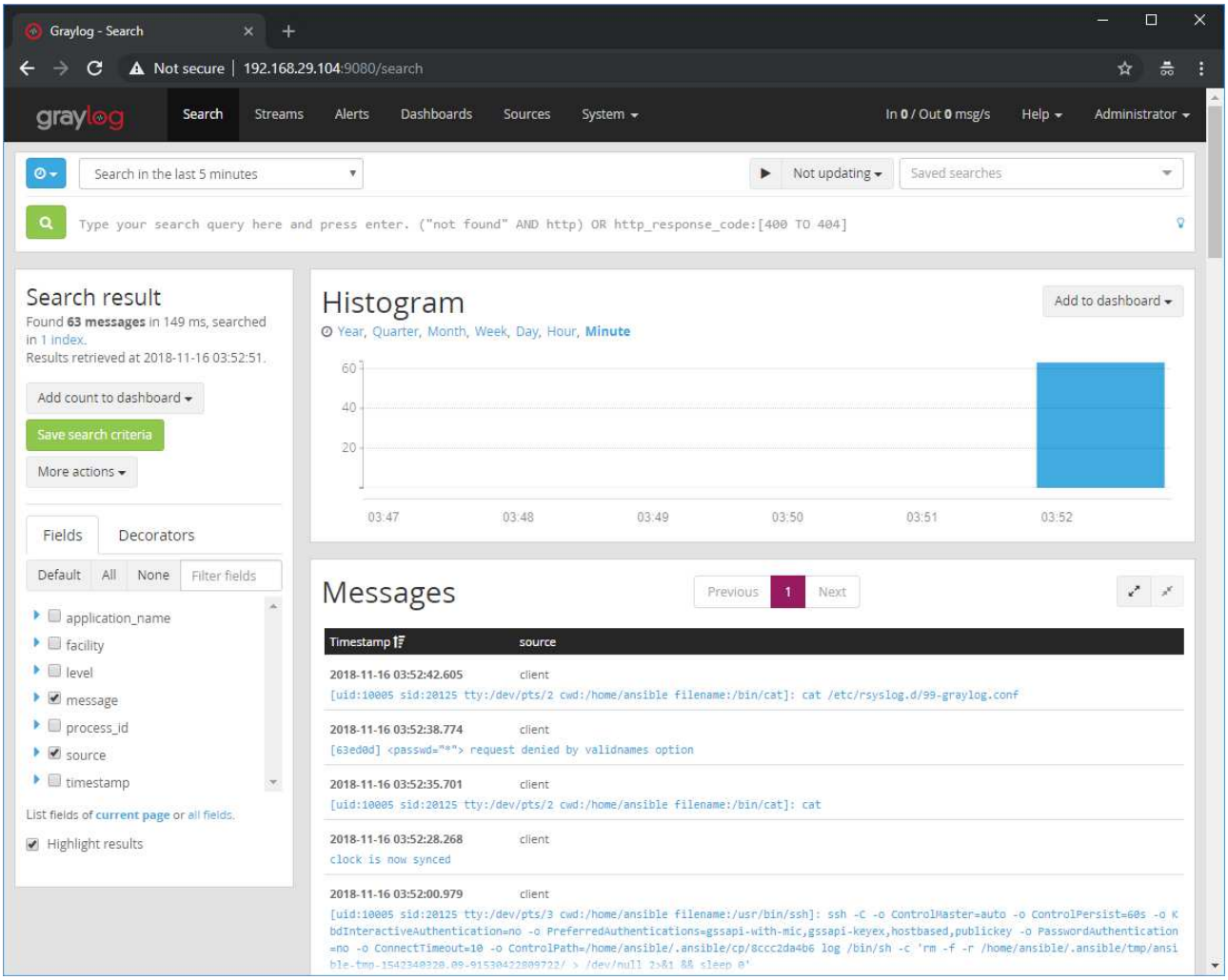

Figura 33. Visualização de logs remotos no Graylog

Legal, não é mesmo? O Graylog está recebendo todos os logs enviados pelos servidores do *datacenter* (e também da máquina `client`), tornando-os acessíveis de forma fácil e pesquisável através de uma interface bastante poderosa.

Vamos testar, por exemplo, as funções de busca do Graylog. Faça um login via SSH na máquina `ns1`, e `sudo` para `root`:

```
$ hostname ; whoami
client
ansible
```

```
$ ssh ns1
Linux ns1 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

Last login: Fri Nov 16 02:07:12 2018 from 192.168.42.2
ansible@ns1:~$
```

```
$ sudo -i
```

```
# hostname ; whoami
ns1
root
```

Agora, volte à interface web do Graylog e busque (no canto superior esquerdo da tela, num campo identificado por uma lupa de cor verde) pelo termo `source:ns1 AND application_name:snoopy`. Esse termo de busca irá mostrar todos os logs oriundos da máquina `ns1` e que tenham sido gerados pela aplicação Snoopy, como vemos abaixo:

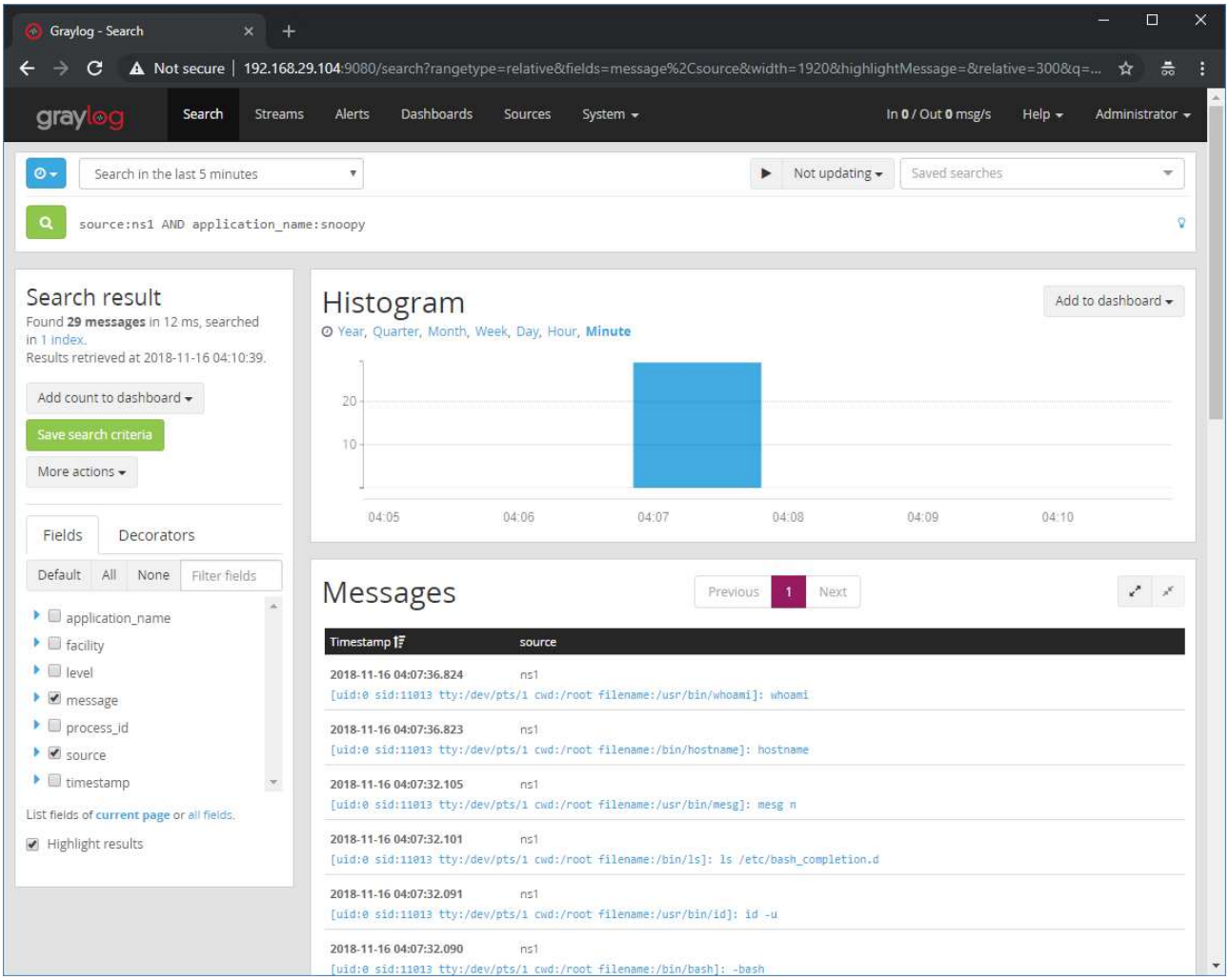

Figura 34. Filtros de busca no Graylog

Vamos um passo além: imagine que você quer encontrar todos os eventos de `sudo` executados na máquina `ns1`. Expanda o termo de busca para `source:ns1 AND application_name:snoopy AND message:sudo`. Você deverá encontrar o evento de escalação de privilégio que fizemos logo acima — clique sobre o evento para expandir os campos do log:

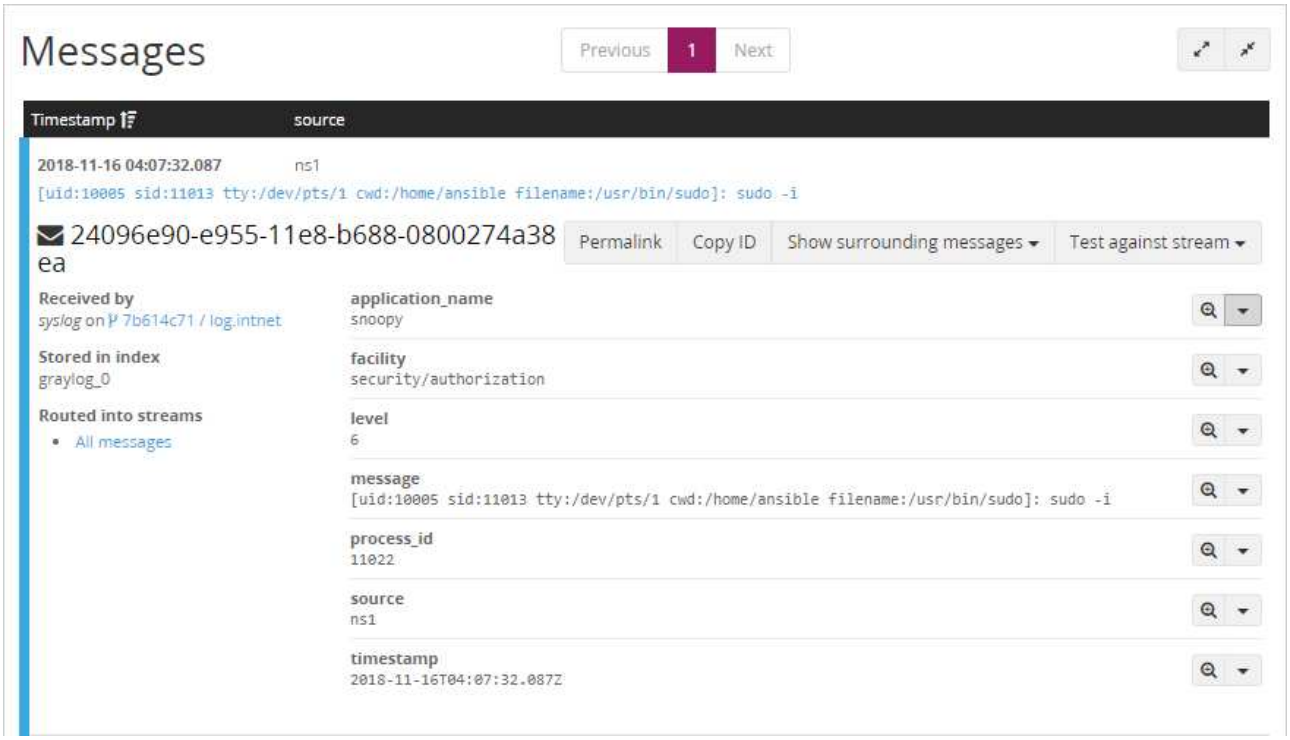

Figura 35. Visualizando eventos específicos

Note que a mensagem mostra, inclusive, o UID do usuário que executou o comando: 10005. Conseguimos descobrir nosso "culpado" facilmente, consultando a base de usuários do LDAP:

```
$ getent passwd | grep 10005
ansible:*:10005:10001:ansible:/home/ansible:/bin/bash
```

9) Autenticação centralizada via LDAP no Graylog

Estamos usando o usuário `admin` para realizar todas as ações no Graylog, o que obviamente não é sustentável — e se quisermos que vários colaboradores tenham acesso à ferramenta e possam pesquisar eventos e investigar logs dos sistemas? Felizmente, a integração do Graylog com sistemas externos de autenticação, como o LDAP, é bastante fácil.

1. Acesse o menu *System* > *Authentication*. Na aba à esquerda, selecione *LDAP/Active Directory* e clique na caixa *Enable LDAP*.
2. Em *Server configuration*, mantenha *Server Type* como LDAP; em *Server Address*, digite `ldap://ldap.intnet:389`; informe *System Username* como `cn=admin,dc=intnet`; finalmente, em `System Password` digite `rnpesr`.

   Clique em *Test Server Connection* para verificar a correta conexão com o servidor LDAP.
3. Continuando, em *User Mapping* defina *Search Base DN* como `ou=People,dc=intnet`; em *User Search Pattern*, digite `(&(objectClass=posixAccount)(uid={0}))`; depois, em *Display Name Attribute* informe `cn`.

   Até o momento, sua tela de configuração deverá estar da seguinte forma:

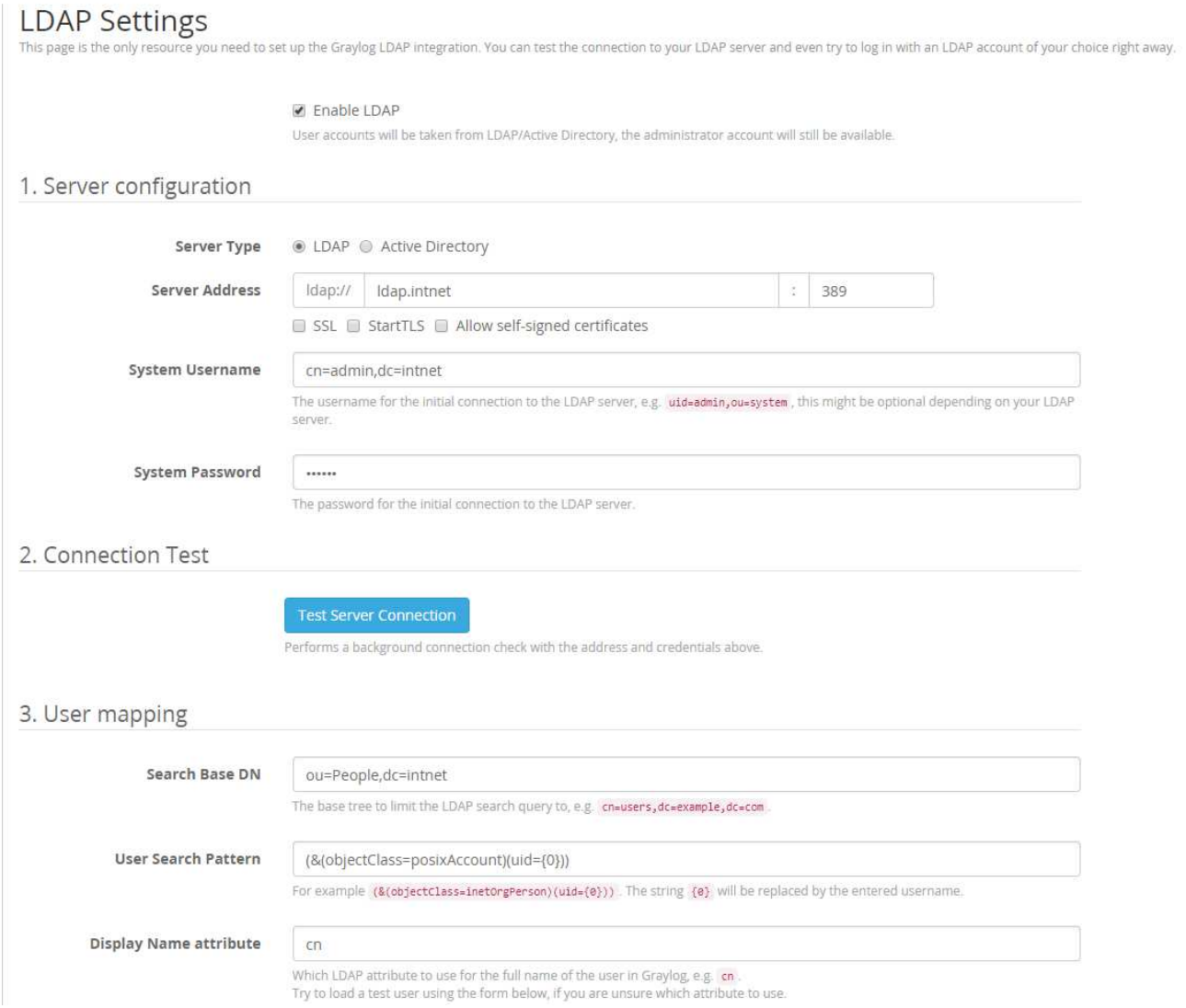

Figura 36. Configuração do Graylog com o LDAP, parte 1

4. Em *Group Mapping*, defina *Group Search Base DN* como `ou=Groups,dc=intnet`; em *Group Search Pattern*, digite `(objectClass=posixGroup)`; para *Group Name Attribute*, informe `cn`; finalmente, mantenha *Default User Role* como *Reader - basic access*.

   A segunda parte da tela de configuração ficará assim:

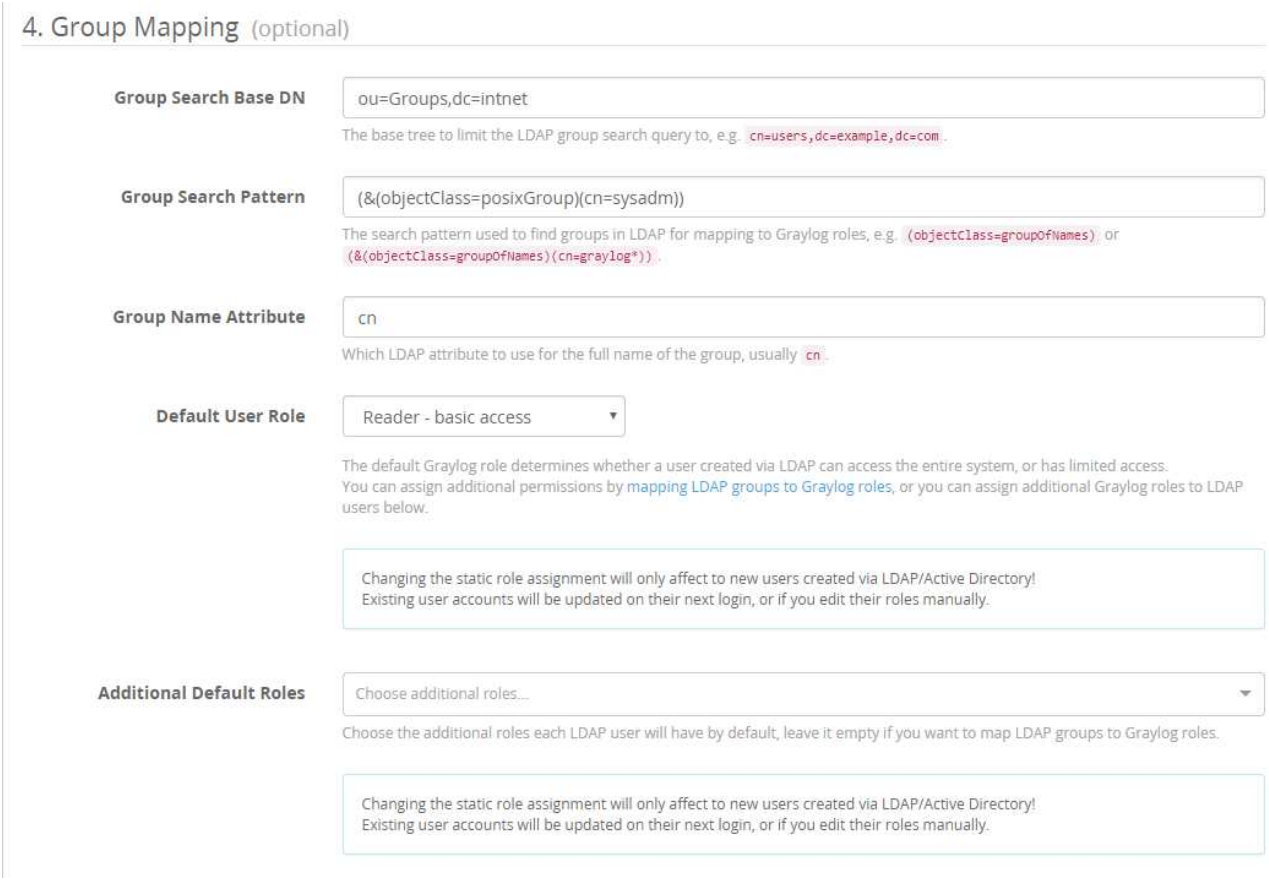

Figura 37. Configuração do Graylog com o LDAP, parte 2

Na base da tela, clique em *Save LDAP settings*.

5. Após salvar as configurações, note que em *Group Mapping* > *Default User Role* há um link destacado em azul que diz *mapping LDAP groups to Graylog roles* — clique neste link. Você verá a tela abaixo:

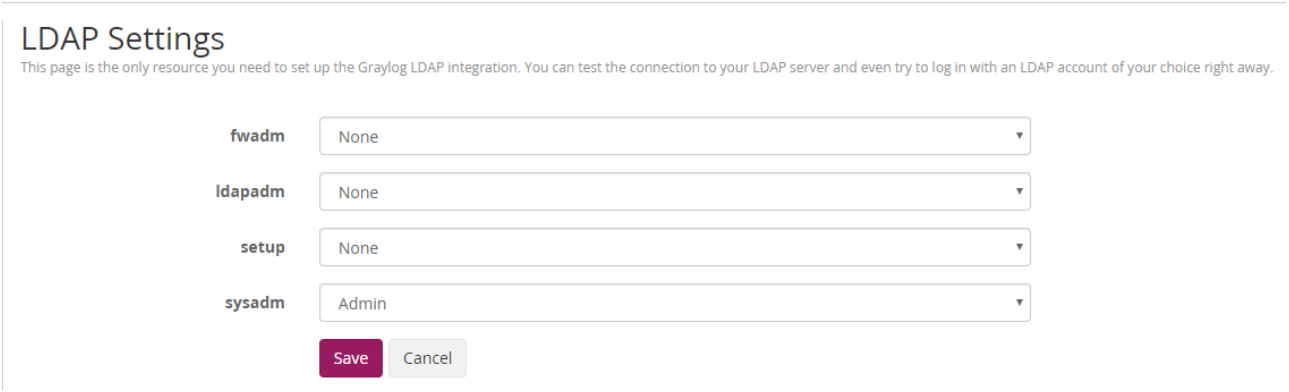

Figura 38. Mapeamento de grupos do LDAP e roles no Graylog

Mapeie o grupo `sysadm` com a *role* `Admin`, e clique em *Save*. Depois, no canto superior da tela, clique em *Administrator* > *Logout*.

6. Vamos testar? Logue com o usuário `luke`, membro do grupo `sysadm`. Use a mesma senha do usuário no LDAP. Observe o nível de acesso do usuário — ele consegue ver todas as abas e configurações às quais o usuário `admin` possui acesso.

   Agora, faça logout e acesse como o usuário `han`. Note: apenas um pequeno número de abas e opções estão disponíveis, e todas como somente leitura. O usuário possui acesso apenas a

streams de logs e *dashboards* pré-configurados pelo administrador, e não consegue alterar quaisquer configurações do sistema.

O sistema de *roles* do Graylog é bastante poderoso, permitindo a customização do nível de acesso de usuários com boa granularidade.

10) Configurando inputs customizados no Graylog

Acesse o Graylog novamente com o usuário `admin` (ou `luke`, se preferir). Busque por mensagens com o padrão `source:ns2 AND application_name:slapd AND message:dc\=intnet`, e expanda uma qualquer, como mostrado abaixo:

Figura 39. Mensagens não interpretadas pelo Graylog

Observe: a mensagem acima é proveniente do log do OpenLDAP, e possui várias informações relevantes no campo `message` — temos a base de busca, escopo e filtro utilizado. Porém, como o Graylog não está configurado para processar e interpretar a mensagem acima, todos os campos ficam agrupados em uma "massa" única, dificultando operações de busca e criação de filtros e alertas. Não podemos, por exemplo, usar um termo como `filter:uid\=ansible` no campo de busca do Graylog.

Vamos instalar um *content pack* para o Graylog que irá instruí-lo sobre como processar logs de aplicações específicas; usaremos, para o nosso exemplo, o servidor web Nginx que está instalado na máquina `log`.

1. No navegador em sua máquina física, faça o download do arquivo https://raw.githubusercontent.com/graylog-labs/graylog-contentpack-nginx/master/content_pack.json para sua Área de Trabalho.
2. Na interface web do Graylog, acesse *System* > *Content Packs*. Clique na caixa *Import Content Pack* e em seguida em *Choose File*, apontando o arquivo que baixamos no passo anterior. Depois, clique em *Upload*.

   Uma nova caixa, *Web Servers*, irá surgir. Clique nessa caixa, marque o botão *nginx* e, na aba à direita, clique em *Apply Content*.

3. Vá para *System* > *Inputs*, e note que temos dois novos *inputs*, `nginx access_log` e `nginx error_log`. Em ambos, clique no botão *Start Input* — frequentemente, essa operação irá reportar erro. Se for esse o caso, acesse a máquina `log` como o usuário `root` e reinicie o *daemon* do Graylog:

```
# hostname ; whoami
log
root
```

```
# systemctl restart graylog-server
```

Após o reinício, volte a *System* > *Inputs* e verifique que ambos os novos *inputs* estão em estado RUNNING, como mostrado abaixo:

Figura 40. Inputs do nginx ativos

4. Agora, temos que instruir o Nginx instalado na máquina `log` a enviar seus logs para os *inputs* que acabamos de configurar. Vamos ao Ansible!

   Acesse a máquina `client` como o usuário `ansible`:

```
$ hostname ; whoami
client
ansible
```

Vamos criar uma *role* que envie os logs de acesso e erro do Nginx de servidores web do *datacenter* para o *input* que configuramos no Graylog:

```
$ ansible-galaxy init --init-path /home/ansible/ansible/roles/ nginxlog
- nginxlog was created successfully
```

Desta vez, apenas um subconjunto de máquinas do *datacenter* deve ser o alvo da nossa *role*. Crie o novo grupo `web_server` no inventário do Ansible:

```
$ echo -e '\n[web_server]\nlog' >> ~/ansible/hosts
```

```
$ cat ~/ansible/hosts
[srv]
ns1
ns2
nfs
192.168.42.2
log

[ntp_server]
log

[log_server]
log

[web_server]
log
```

Agora, edite o arquivo `/home/ansible/ansible/roles/nginxlog/vars/main.yml` com o seguinte conteúdo:

```
1 ---
2 logsrv: "{{ groups['log_server'][0] }}.intnet"
```

Sem surpresas até aí — queremos configurar o envio de logs para o concentrador de eventos da rede, que está no grupo `log_server`. Vamos editar o *template* de configuração do Nginx que irá usar a variável acima: crie o arquivo novo `/home/ansible/ansible/roles/nginxlog/templates/nginx_graylog.conf.j2` com o seguinte conteúdo:

```
log_format  graylog2_format  '$remote_addr - $remote_user [$time_local]
"$request" $status $body_bytes_sent "$http_referer" "$http_user_agent"
"$http_x_forwarded_for"
<msec=$msec|connection=$connection|connection_requests=$connection_requests|cache_s
tatus=$upstream_cache_status|cache_control=$upstream_http_cache_control|expires=$up
stream_http_expires|millis=$request_time>';

access_log syslog:server={{ logsrv }}:12301 graylog2_format;
error_log syslog:server={{ logsrv }}:12302;
```

O arquivo acima será incluído na configuração padrão do Nginx instalado na máquina `log` (e, de fato, de qualquer outro servidor web futuro que adicionemos ao *datacenter*), informando que os logs de acesso e erros devem ser enviados para a máquina remota `logsrv` (variável que definimos no arquivo `vars`, acima) num formato compatível com o esperado pelo processador do Graylog.

Vamos às tarefas: edite o arquivo `/home/ansible/ansible/roles/nginxlog/tasks/main.yml` com o seguinte conteúdo e confira:

```
---
- name: Setup Nginx <> Graylog logging
  template:
    src: nginx_graylog.conf.j2
    dest: /etc/nginx/conf.d/99-graylog.conf
    owner: root
    group: root
    mode: 0644
  notify:
    - Restart Nginx
```

Usando o módulo `template`, a *task* acima copia o *template* que configuramos anteriormente para o diretório `/etc/nginx/conf.d` (o qual é incluído pelo arquivo-padrão `/etc/nginx/nginx.conf`), ajusta suas permissões e reinicia o *daemon* do Nginx.

É claro, temos que criar o *handler*. Edite o arquivo `/home/ansible/ansible/roles/nginxlog/handlers/main.yml` com o seguinte conteúdo:

```
---
- name: Restart Nginx
  service:
    name: nginx
    state: restarted
```

Nada de novo: usamos o módulo `service` para reiniciar o Nginx após a execução da *task*.

Como essa nova *role* se aplica apenas a um subconjunto de *hosts* do inventário (os membros do grupo `web_server`), vamos adicionar uma nova seção ao *playbook* `/home/ansible/ansible/srv.yml`:

```
$ cat << EOF >> ~/ansible/srv.yml

- hosts: web_server
  become: yes
  become_user: root
  become_method: sudo
  roles:
    - nginxlog
EOF
```

```
$ cat ~/ansible/srv.yml
---
- hosts: srv
  become: yes
  become_user: root
  become_method: sudo
  roles:
    - sudoers
    - ntp
    - snoopy
    - syslog

- hosts: web_server
  become: yes
  become_user: root
  become_method: sudo
  roles:
    - nginxlog
```

Agora, vamos executar o *playbook*. Como apenas a máquina `log` será alvo das modificações que fizemos, o uso da opção `--limit` (ou, de forma mais simples, `-l`) é interessante para acelerar a execução do Ansible:

```
$ ansible-playbook -i ~/ansible/hosts ~/ansible/srv.yml -l web_server

PLAY [srv]
*********************************************************************************
**********************

(...)

PLAY [web_server]
*********************************************************************************
****************


TASK [Gathering Facts]
*********************************************************************************
***********

ok: [log]

TASK [nginxlog : Setup Nginx <> Graylog logging]
*******************************************************************

changed: [log]

RUNNING HANDLER [nginxlog : Restart Nginx]
*************************************************************************

changed: [log]

PLAY RECAP
*********************************************************************************
**********************

log                        : ok=11   changed=3    unreachable=0    failed=0
```

5. E agora? Se temos alterações no repositório local, é hora de enviá-las para o Git remoto:

```
$ cd ~/ansible/ ; git add . ; git commit -m 'Adicionada role para envio de logs
formatados do Nginx de servidores web para o servidor Graylog' ; git push
[master ad7b12d] Adicionada role para envio de logs formatados do Nginx de
servidores web para o servidor Graylog
 11 files changed, 138 insertions(+)
 create mode 100644 roles/nginxlog/README.md
 create mode 100644 roles/nginxlog/defaults/main.yml
 create mode 100644 roles/nginxlog/handlers/main.yml
 create mode 100644 roles/nginxlog/meta/main.yml
 create mode 100644 roles/nginxlog/tasks/main.yml
 create mode 100644 roles/nginxlog/templates/nginx_graylog.conf.j2
 create mode 100644 roles/nginxlog/tests/inventory
 create mode 100644 roles/nginxlog/tests/test.yml
 create mode 100644 roles/nginxlog/vars/main.yml
Counting objects: 16, done.
Compressing objects: 100% (11/11), done.
Writing objects: 100% (16/16), 1.69 KiB | 0 bytes/s, done.
Total 16 (delta 2), reused 0 (delta 0)
To nfs:/home/ansible/ansible.git
   b1e7a24..ad7b12d  master -> master
```

6. Vamos voltar para o Graylog. Em seu navegador, vá para *System* > *Inputs* e em `nginx access_log`, clique no botão *Show received messages*.

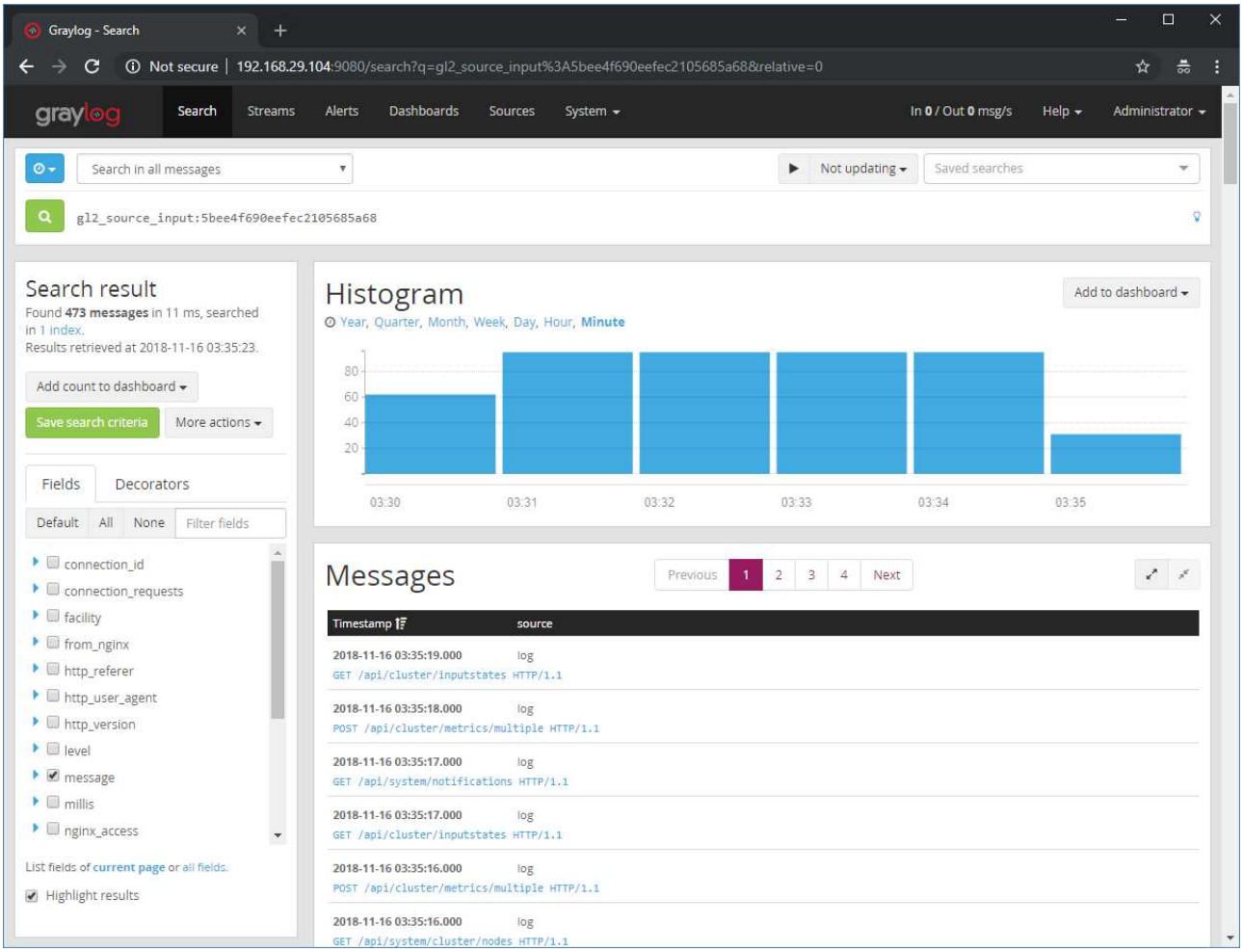

Figura 41. Recebimento de logs formatados do Nginx

Primeiro bom sinal: os logs do Nginx estão sendo recebidos e processados pelo *input* do Graylog, o que significa que nossa *role* no Ansible funcionou a contento. Mas, tirando esse fato, o que mudou? As mensagens mostradas pelo Graylog parecem, em grande parte, as mesmas de antes.

7. Expanda um dos eventos de log recebidos pelo Graylog nesse *input* `nginx access_log`, como mostrado abaixo:

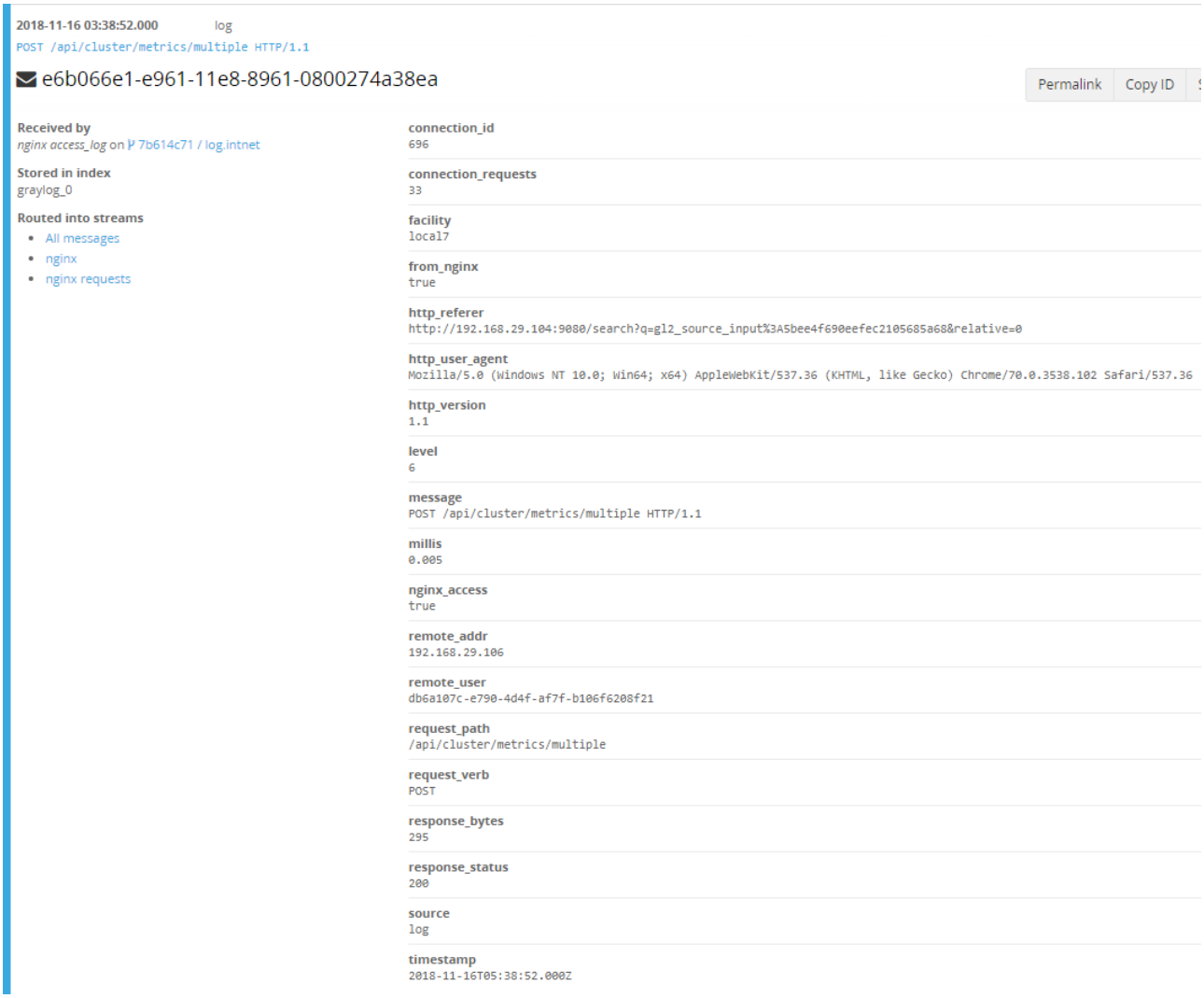

Figura 42. Processamento de campos individuais em mensagens

Observe o evento acima, e compare-o com o do OpenLDAP que analisamos no começo desta atividade — em lugar de um campo `message` pouco específico, temos agora um grande conjunto de campos pesquisáveis, como:

- `http_referer`
- `http_user_agent`
- `remote_addr`
- `request_path`
- `request_verb`
- `response_bytes`
- `response_status`

Todos esses campos são extremamente relevantes em um pacote HTTP, e seu processamento e pesquisa facilita enormemente o trabalho do analista de segurança. E se quisermos descobrir quais pacotes com método POST foram enviados para uma URL específica do servidor web, filtrando pelo IP de origem? Com os campos acima, pesquisas complexas como essa tornam-se totalmente viáveis.

8. Encerradas as nossas atividades com o Graylog, recomenda-se que o aluno mantenha a máquina `log` desligada a partir desta sessão. Apesar de ser interessante que recuperemos os logs de todas as VMs do *datacenter* simulado para análise, o fato de essa máquina exigir 4 GB de

memória RAM para operar a contento torna-a um peso muito grande na execução das atividades das próximas sessões.

O *Content Pack* que estamos usando para processar essas mensagens do Graylog é um entre muitos que podem ser encontrados no *Graylog Marketplace*: https://marketplace.graylog.org/ . Esse website contém centenas de *add-ons* e *plugins* para as versões gratuita e empresarial do Graylog, que permitem estender suas funcionalidades de forma conveniente.

A construção de expressões regulares e regras de processamento de logs para o Graylog é um trabalho árduo, porém necessário para tornar a ferramenta SIEM verdadeiramente efetiva e produtiva para os analistas de segurança. Para facilitar seu trabalho, consulte o nome das ferramentas mais utilizadas na sua organização no *Graylog Marketplace* — quem sabe algum outro usuário já fez um *plugin* que irá facilitar bastante seu trabalho de integração?

Sessão 7: Hardening de sistemas web

Nesta sessão, iremos configurar um website usando o CMS Wordpress seguindo as melhores práticas de segurança, bem como tornando-o altamente disponível através do balanceamento de carga em múltiplos nodos de *frontend* web.

1) Topologia desta sessão

Criaremos quatro novas máquinas nesta sessão, a saber:

- `www1`, primeiro nodo de atendimento web, instalado manualmente. Endereço IP 10.0.42.5/24.
- `www2`, segundo nodo de atendimento web, instalado de forma automática via Ansible. Endereço IP 10.0.42.6/24.
- `db`, servidor de banco de dados para as máquinas `www1` e `www2`. Endereço IP 10.0.42.7/24.
- `lb`, balanceador de carga HTTP/HTTPS que redirecionará requisições para as VMs `www1` e `www2`. Endereço IP 10.0.42.8/24.

1. Temos que criar quatro novos registros DNS diretos e reversos. Acesse a máquina `ns1` como o usuário `root`:

```
# hostname ; whoami
ns1
root
```

Edite o arquivo de zonas `/etc/nsd/zones/intnet.zone`, inserindo entradas A para a máquinas indicadas no começo desta atividade. **Não se esqueça** de incrementar o valor do serial no topo do arquivo!

```
# nano /etc/nsd/zones/intnet.zone
(...)
```

```
# grep 'www1\|www2\|db\|lb' /etc/nsd/zones/intnet.zone
www1    IN    A                10.0.42.5
www2    IN    A                10.0.42.6
db      IN    A                10.0.42.7
lb      IN    A                10.0.42.8
```

Faça o mesmo para o arquivo de zona reversa:

```
# nano /etc/nsd/zones/10.0.42.zone
```

```
# grep 'www1\|www2\|db\|lb' /etc/nsd/zones/10.0.42.zone
5       IN   PTR               www1.intnet.
6       IN   PTR               www2.intnet.
7       IN   PTR               db.intnet.
8       IN   PTR               lb.intnet.
```

Assine o arquivo de zonas usando o *script* criado anteriormente:

```
# bash /root/scripts/signzone-intnet.sh
reconfig start, read /etc/nsd/nsd.conf
ok
ok
ok
ok removed 0 rrsets, 0 messages and 0 key entries
```

Verifique a criação das entradas usando o comando `dig`:

```
# for host in www1 www2 db lb; do echo -n "$host: "; dig ${host}.intnet +short;
done
www1: 10.0.42.5
www2: 10.0.42.6
db: 10.0.42.7
lb: 10.0.42.8
```

```
# for ip in 5 6 7 8; do echo -n "10.0.42.${ip}: "; dig -x 10.0.42.${ip} +short;
done
10.0.42.5: www1.intnet.
10.0.42.6: www2.intnet.
10.0.42.7: db.intnet.
10.0.42.8: lb.intnet.
```

2) Configuração do servidor de banco de dados

1. Vamos começar criando a máquina que atuará como servidor de banco de dados em nossa topologia. Clone a máquina `debian-template` para uma nova, de nome `db`, com uma única interface de rede conectada à DMZ. O IP da máquina será 10.0.42.7/24.

   Concluída a clonagem, ligue a máquina e logue como `root`. Depois, use o script `/root/scripts/changehost.sh` para fazer a configuração automática:

   ```
   # hostname ; whoami
   debian-template
   root
   ```

```
# bash ~/scripts/changehost.sh -h db -i 10.0.42.7 -g 10.0.42.1
Signing ssh_host_ecdsa_key.pub key...
Signing ssh_host_ed25519_key.pub key...
Signing ssh_host_rsa_key.pub key...
Configuring host key trust...
Configuring user key trust...
All done!
```

```
$ ip addr show label 'enp0s*' | grep 'inet ' | awk '{print $2,$NF}' ; hostname ;
whoami
10.0.42.7/24 enp0s3
db
root
```

2. Vamos aplicar à máquina `db` o *baseline* de segurança que configuramos no Ansible: `sudo`, OpenNTPD, Snoopy e Rsyslog centralizado. Acesse a máquina `client` como o usuário `ansible`:

```
$ hostname ; whoami
client
ansible
```

Insira a máquina `db` no inventário gerenciado pelo Ansible:

```
$ sed -i '/\[srv\]/a db' ~/ansible/hosts
```

```
$ cat ~/ansible/hosts
[srv]
db
ns1
ns2
nfs
192.168.42.2
log

[ntp_server]
log

[log_server]
log

[web_server]
log
```

Execute o *playbook* `/home/ansible/ansible/srv.yml`, limitando o escopo à máquina `db` e alterando

a escala de privilégio para o `su`:

```
$ ansible-playbook -i ~/ansible/hosts -l db -Ke ansible_become_method=su
~/ansible/srv.yml
SUDO password:

PLAY [srv]
**********************************************************************************
**********************


TASK [Gathering Facts]
**********************************************************************************
***********

ok: [db]

TASK [sudoers : Propagate sudoers configuration]
*********************************************************************

changed: [db]

TASK [sudoers : Sets root account as expired]
**********************************************************************

changed: [db]

TASK [ntp : Install OpenNTPD]
**********************************************************************************
****

fatal: [db]: FAILED! => {"changed": false, "module_stderr": "Shared connection to
db closed.\r\n", "module_stdout": "\r\nSua conta expirou; entre em contato com o
administrador do sistema\r\nsu: Falha de autenticação\r\n", "msg": "MODULE
FAILURE\nSee stdout/stderr for the exact error", "rc": 1}
        to retry, use: --limit @/home/ansible/ansible/srv.retry

PLAY RECAP
**********************************************************************************
**********************

db                         : ok=3    changed=2    unreachable=0    failed=1
```

A *role* irá falhar no passo de instalação do OpenNTPD, pois neste momento o `sudo` acaba de ser configurado e a conta do usuário `root`, expirada. Execute o *playbook* novamente, desta vez sem customizar o método de escalação de privilégio:

```
$ ansible-playbook -i ~/ansible/hosts -l db ~/ansible/srv.yml
```

```
PLAY [srv]
*************************************************************************************
**********************


TASK [Gathering Facts]
*************************************************************************************
***********

ok: [db]

TASK [sudoers : Propagate sudoers configuration]
*******************************************************************

ok: [db]

TASK [sudoers : Sets root account as expired]
*******************************************************************

changed: [db]

TASK [ntp : Install OpenNTPD]
*************************************************************************************
****

changed: [db]

TASK [ntp : Copy OpenNTPD configuration]
*******************************************************************************

changed: [db]

TASK [snoopy : Install Snoopy]
*************************************************************************************
***

changed: [db]

TASK [snoopy : Configure ld.so.preload]
*********************************************************************************

changed: [db]

TASK [syslog : Configure centralized syslog]
*******************************************************************

changed: [db]

RUNNING HANDLER [ntp : Restart OpenNTPD]
*******************************************************************************
```

```
changed: [db]

RUNNING HANDLER [syslog : Restart Rsyslog]
********************************************************************

changed: [db]

PLAY [web_server]
*****************************************************************************
****************

skipping: no hosts matched

PLAY RECAP
*****************************************************************************
**********************

db                         : ok=10   changed=8    unreachable=0    failed=0
```

Perfeito, a máquina está configurada para operar em nosso *datacenter*.

3. Agora, acesse a máquina `db` como o usuário `root`. Vamos instalar o SGBD MariaDB:

```
# usermod -e -1 root ; apt-get install -y --no-install-recommends mariadb-server ;
usermod -e 0 root
```

4. Vamos criar uma base de dados para nosso website, com o nome `seg10web`. Para acessar a base, criaremos um usuário `seg10user`, com senha `seg10pass`. Execute os comandos a seguir:

```
# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 10.1.26-MariaDB-0+deb9u1 Debian 9.1

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]>
```

```
MariaDB [(none)]> create database seg10web;
Query OK, 1 row affected (0.00 sec)
```

```
MariaDB [(none)]> grant all on seg10web.* to seg10user@localhost identified by
'seg10pass';
Query OK, 0 rows affected (0.00 sec)
```

```
MariaDB [(none)]> grant all on seg10web.* to seg10user@'10.0.42.%' identified by
'seg10pass';
Query OK, 0 rows affected (0.00 sec)
```

```
MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

```
MariaDB [(none)]> quit
Bye
```

Note que demos permissão para login a partir de *localhost*, e também da rede 10.0.42.0/24 — logins oriundos de qualquer outro IP serão negados.

5. O MariaDB escuta apenas a conexões vindas de *localhost*, por padrão. Vamos corrigir isso:

```
# sed -i 's/^\(bind-address[\t ]*\=\).*/\1 0.0.0.0/' /etc/mysql/mariadb.conf.d/50-
server.cnf
```

```
# systemctl restart mariadb
```

Note que o SGBD está escutando em todas as interfaces:

```
# ss -tnlp | grep 3306
LISTEN     0      80           *:3306                     *:*
users:(("mysqld",pid=3579,fd=17))
```

Essa é uma limitação do MariaDB/MySQL — apenas uma interface pode ser especificada na diretiva `bind-address` do arquivo de configuração. Se for necessário escutar em mais de uma interface de rede, e necessário especificar o *catch-all* 0.0.0.0 (referência: https://mariadb.com/kb/en/library/configuring-mariadb-for-remote-client-access/).

6. Para consertar essa limitação, vamos usar o firewall local. Insira uma regra que permite que as máquinas *localhost*, `www1` e `www2` conectem-se ao SGBD, e nenhuma outra:

```
# iptables -A INPUT -i lo -p tcp -m tcp --dport 3306 -j ACCEPT
```

```
# iptables -A INPUT -s 10.0.42.5,10.0.42.6 -p tcp -m tcp --dport 3306 -j ACCEPT
```

```
# iptables -A INPUT -p tcp --dport 3306 -j DROP
```

7. Para manter as regras entre *reboots*, instale o pacote `iptables-persistent`:

```
# apt-get install -y iptables-persistent
```

Na instalação do pacote, quando perguntado, responda:

Tabela 9. Configurações do iptables-persistent

Pergunta	Resposta
Salvar as regras IPv4 atuais?	Sim
Salvar as regras IPv6 atuais?	Sim

4) Configuração do servidor web www1

1. Agora, vamos para o primeiro servidor web. Clone a máquina `debian-template` para uma de nome `www1`, com uma única interface de rede conectada à DMZ. O IP da máquina será 10.0.42.5/24.

   Concluída a clonagem, ligue a máquina e logue como `root`. Depois, use o script `/root/scripts/changehost.sh` para fazer a configuração automática, como de costume.

```
# hostname ; whoami
debian-template
root
```

```
# bash ~/scripts/changehost.sh -h www1 -i 10.0.42.5 -g 10.0.42.1
Signing ssh_host_ecdsa_key.pub key...
Signing ssh_host_ed25519_key.pub key...
Signing ssh_host_rsa_key.pub key...
Configuring host key trust...
Configuring user key trust...
All done!
```

2. Para aplicar o *baseline* de segurança via Ansible à máquina `www1`, repita o que fizemos no passo (2) da atividade (2) desta sessão:

```
$ hostname ; whoami
client
ansible
```

```
$ sed -i '/\[srv\]/a www1' ~/ansible/hosts
```

```
$ ansible-playbook -i ~/ansible/hosts -l www1 -Ke ansible_become_method=su
~/ansible/srv.yml ; ansible-playbook -i ~/ansible/hosts -l www1 ~/ansible/srv.yml
SUDO password:

(...)

PLAY RECAP
*********************************************************************************
**********************

www1                       : ok=10   changed=8    unreachable=0    failed=0
```

3. Agora, acesse a máquina `www1` como o usuário `root`:

```
# hostname ; whoami
www1
root
```

4. Vamos instalar as dependências para o correto funcionamento do CMS Wordpress em nosso servidor:

```
# apt-get install -y nginx php-fpm php-mysql
```

5. Primeiro, vamos configurar o servidor web Nginx. Apague a configuração-padrão de websites publicados do Nginx:

```
# rm /etc/nginx/sites-enabled/default
```

   Crie o arquivo novo `/etc/nginx/sites-available/seg10` com o seguinte conteúdo:

```
upstream php {
        server unix:/run/php/php7.0-fpm.sock;
}

server {
        server_name seg10web.intnet;
        root /var/www/seg10;
        index index.php;

        location = /favicon.ico {
                log_not_found off;
                access_log off;
        }

        location = /robots.txt {
                allow all;
                log_not_found off;
                access_log off;
        }

        location / {
                try_files $uri $uri/ /index.php?$args;
        }

        location ~ \.php$ {
                include fastcgi.conf;
                fastcgi_intercept_errors on;
                fastcgi_pass php;
                fastcgi_buffers 16 16k;
                fastcgi_buffer_size 32k;
        }

        location ~* \.(js|css|png|jpg|jpeg|gif|ico)$ {
                expires max;
                log_not_found off;
        }
}
```

Habilite-o na configuração do Nginx:

```
# p=$PWD ; cd /etc/nginx/sites-enabled/ ; ln -s ../sites-available/seg10 . ; cd $p
; unset p
```

Reinicie o Nginx:

```
# systemctl restart nginx.service
```

6. O Wordpress recomenda uma ligeira alteração na configuração do PHP-FPM, como se segue:

```
# sed -i 's/^;\(cgi\.fix\_pathinfo=\).*/\10/' /etc/php/7.0/fpm/php.ini
```

Reinicie o *daemon*:

```
# systemctl restart php7.0-fpm.service
```

7. Vamos fazer o download do Wordpress, desempacotá-lo e renomear o diretório de acordo com a configuração do Nginx:

```
# cd /var/www/ ; \
  wget -q https://wordpress.org/latest.tar.gz ; \
  tar zxf latest.tar.gz ; \
  rm latest.tar.gz ; \
  mv wordpress seg10 ; \
  chown -R www-data. /var/www
```

```
# ls -1 /var/www/
html
seg10
```

8. Para conseguir acessar a máquina `www1` através do IP público do firewall, teremos que fazer alguns ajustes nas regras atuais. Acesse a máquina `ns1` como `root`:

```
# hostname ; whoami
ns1
root
```

Para que consigamos atingir a máquina `www1` será necessário criar uma regra de DNAT na tabela `nat`, *chain* PREROUTING, além de uma regra na tabela `filter`, *chain* FORWARD, correspondente. Mapearemos a porta externa 80/TCP para a porta interna 80/TCP, sem alterações.

```
# iptables -t nat -A PREROUTING -i enp0s3 -p tcp -m tcp --dport 80 -j DNAT --to
-destination 10.0.42.5
```

```
# iptables -A FORWARD -i enp0s3 -d 10.0.42.5/32 -p tcp -m tcp --dport 80 -j ACCEPT
```

9. Perfeito! Em sua máquina física, abra o navegador e aponte-o para o IP da interface `enp0s3` da máquina `ns1`, o IP público do nosso *datacenter* simulado. Você deverá ver a tela de instalação inicial do Wordpress, como se segue:

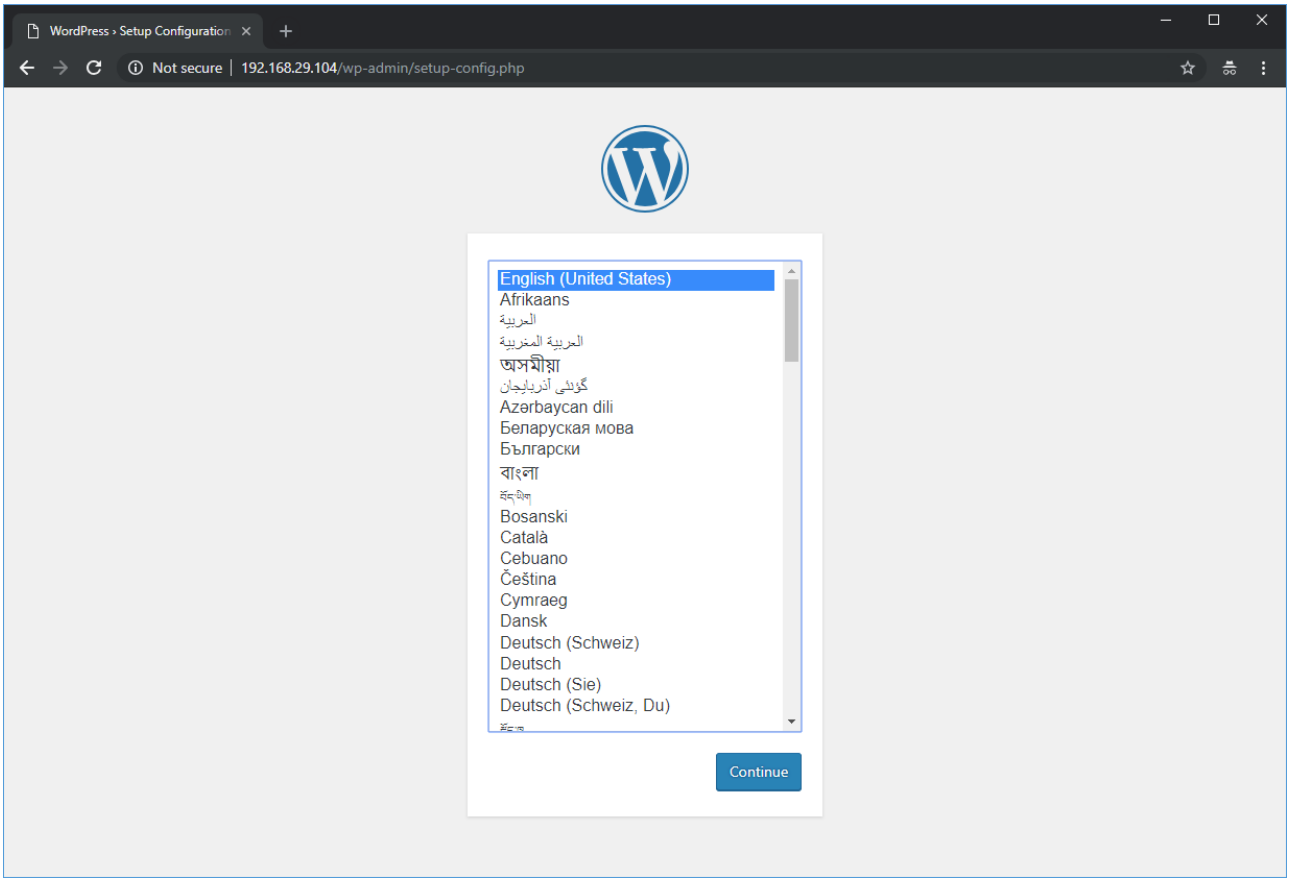

Figura 43. Tela inicial de instalação do Wordpress

Escolha o idioma *Português do Brasil*, e clique em *Continuar*. Na tela seguinte, clique em *Vamos lá!*.

Na tela de configuração da conexão com o banco de dados, digite `seg10web` em *Nome do banco de dados*; para *Nome de usuário*, informe `seg10user`; em *Senha*, `seg10pass`; para *Servidor do banco de dados*, informe `db.intnet`; para o *Prefixo da tabela*, mantenha o padrão `wp_`. Sua tela deverá ficar assim:

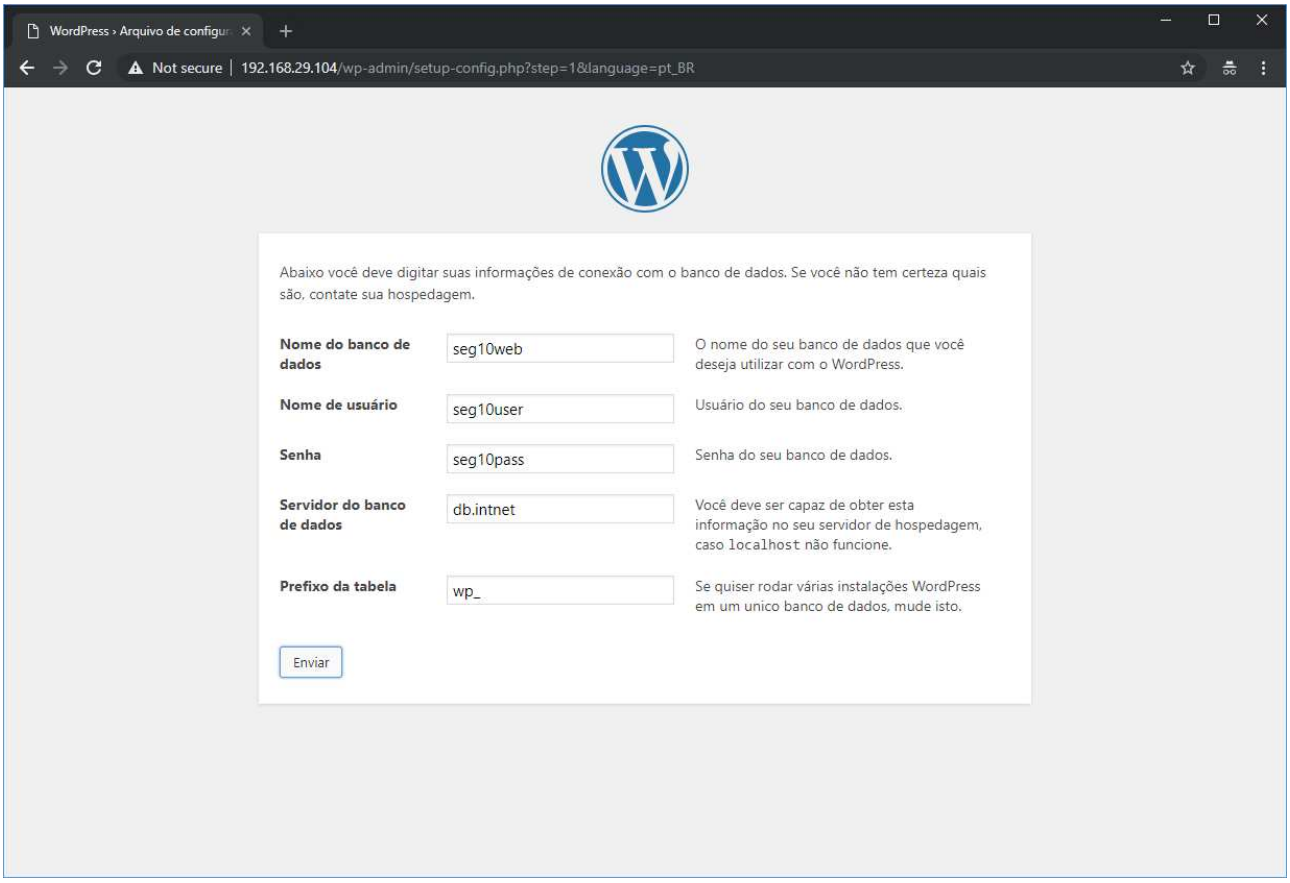

Figura 44. Configuração da conexão com o banco de dados

Clique em *Enviar*, e depois em *Instalar*.

Na tela de informações do site, digite `Seg10 Web` para o *Título do site*; em *Nome de usuário*, defina `admin`; para *Senha*, escolha `rnpesr123`; em *O seu e-mail*, informe `seg10web@int.net`; finalmente, mantenha a caixa *Visibilidade nos mecanismos de busca* desmarcada. Ao final do processo, sua tela deverá estar da seguinte forma:

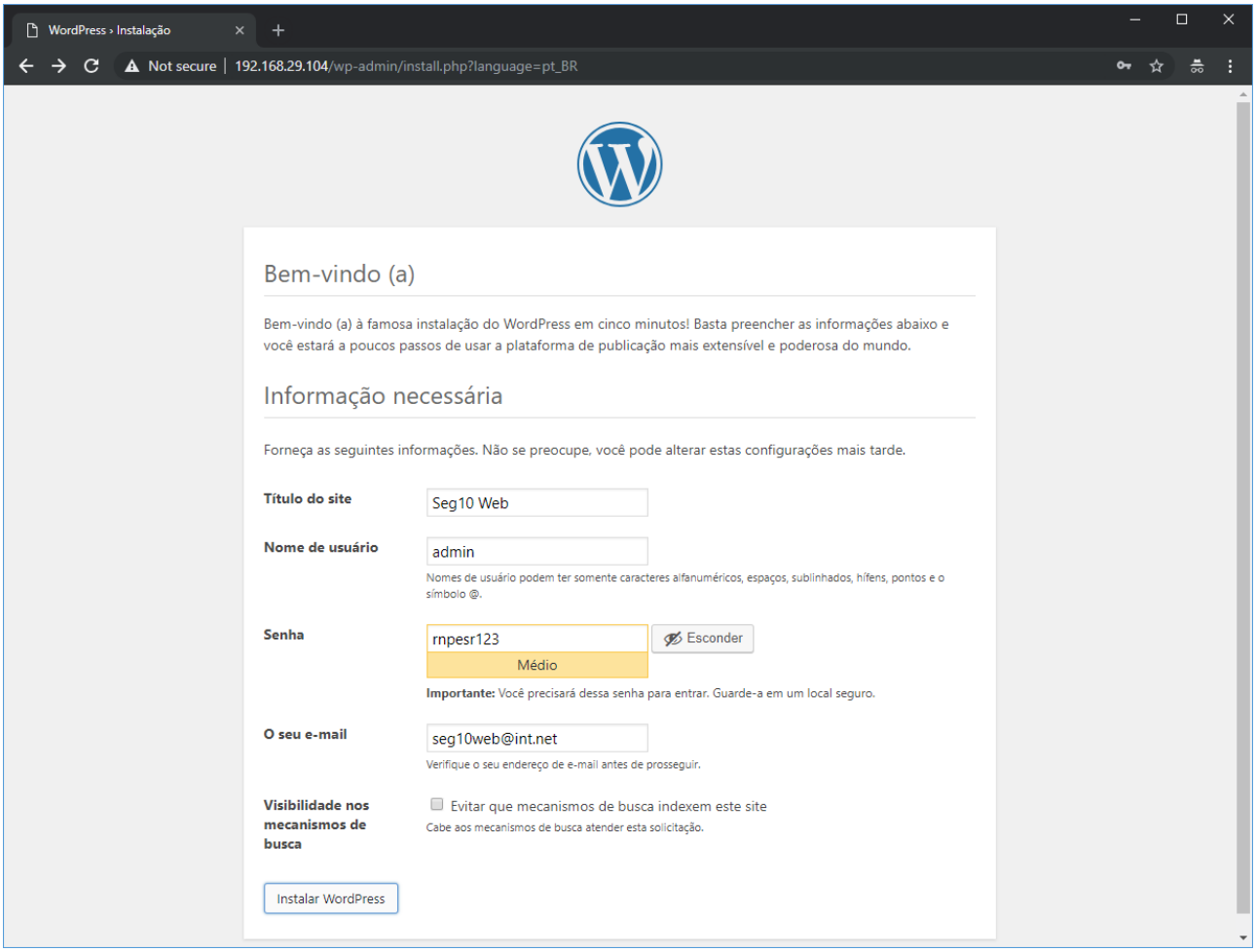

Figura 45. Configuração das informações do site

Clique em *Instalar WordPress*. Ao final do processo de instalação, clique em *Acessar* para entrar na interface administrativa do Wordpress. Alternativamente, digite o endereço IP da interface `enp0s3` da máquina `ns1` na URL do navegador para acessar a página inicial do nosso portal, como mostra a figura abaixo:

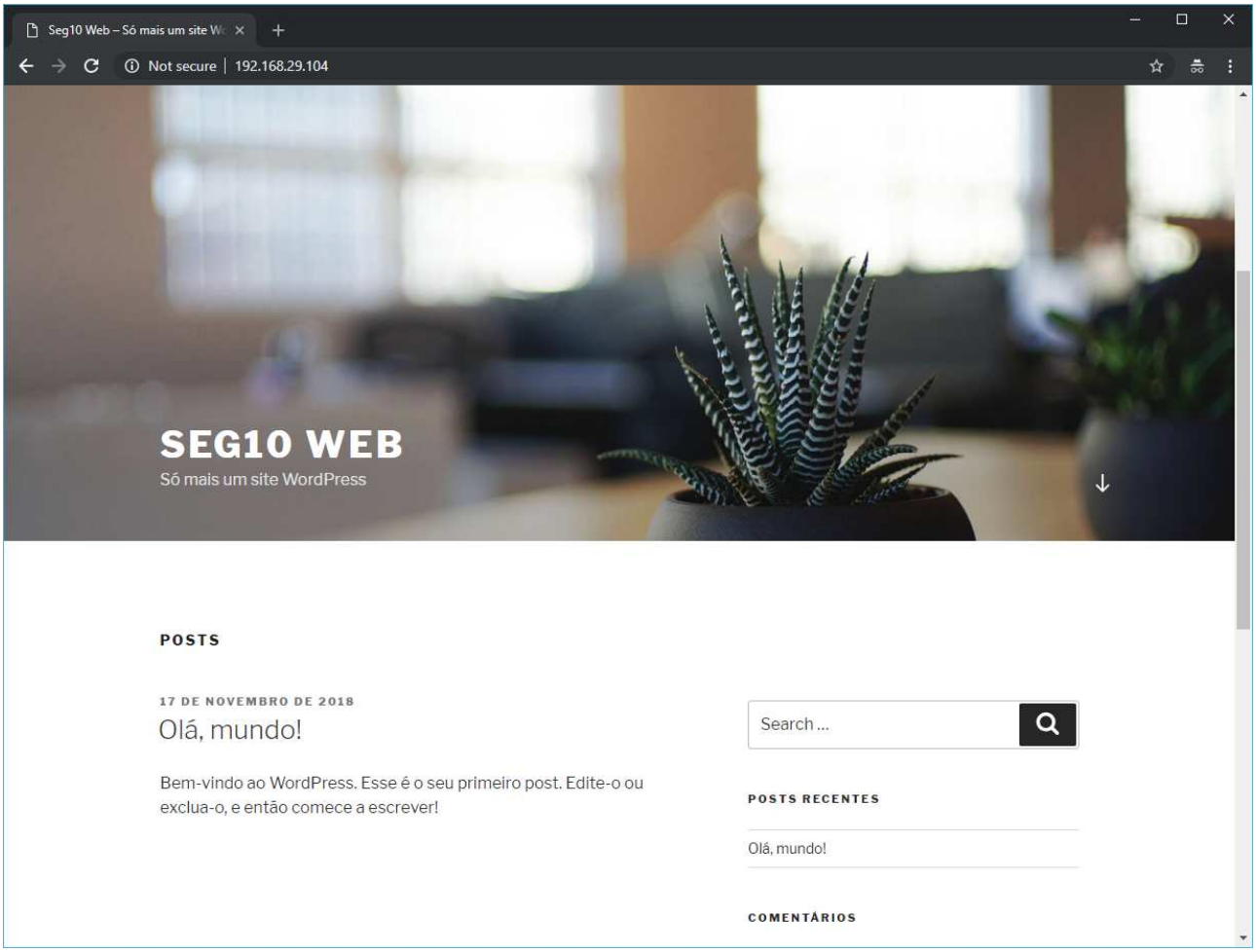

Figura 46. Página inicial do portal Seg10 Web

5) Configuração automática do servidor web www2

1. Para o segundo servidor web, a ideia é automatizar sua configuração completamente usando o Ansible. Antes de mais nada, clone a máquina `debian-template` para uma de nome `www2`, com uma única interface de rede conectada à DMZ. O IP da máquina será 10.0.42.6/24.

   Concluída a clonagem, ligue a máquina e logue como `root`. Depois, use o script `/root/scripts/changehost.sh` para fazer a configuração automática, como de costume.

```
# hostname ; whoami
debian-template
root
```

```
# bash ~/scripts/changehost.sh -h www2 -i 10.0.42.6 -g 10.0.42.1
Signing ssh_host_ecdsa_key.pub key...
Signing ssh_host_ed25519_key.pub key...
Signing ssh_host_rsa_key.pub key...
Configuring host key trust...
Configuring user key trust...
All done!
```

2. Para aplicar o *baseline* de segurança via Ansible à máquina `www2`, repita o que fizemos no passo (2) da atividade (2) desta sessão:

```
$ hostname ; whoami
client
ansible
```

```
$ sed -i '/\[srv\]/a www2' ~/ansible/hosts
```

```
$ ansible-playbook -i ~/ansible/hosts -l www2 -Ke ansible_become_method=su
~/ansible/srv.yml ; ansible-playbook -i ~/ansible/hosts -l www2 ~/ansible/srv.yml
SUDO password:

(...)

PLAY RECAP
*********************************************************************************
**********************

www2                       : ok=10   changed=8    unreachable=0    failed=0
```

3. Agora, ainda como o usuário `ansible` na máquina `client`, vamos criar uma *role* para automatizar totalmente a instalação e configuração do portal *Seg10 Web* na máquina `www2` (e em quaisquer outros nodos web que venhamos a adicionar no futuro):

```
$ ansible-galaxy init --init-path=/home/ansible/ansible/roles seg10-web
- seg10-web was created successfully
```

4. Vamos começar adicionando o arquivo de configuração do servidor Nginx — crie o arquivo novo `/home/ansible/ansible/roles/seg10-web/files/nginx_seg10` com o seguinte conteúdo:

```
upstream php {
        server unix:/run/php/php7.0-fpm.sock;
}

server {
        server_name seg10web.intnet;
        root /var/www/seg10;
        index index.php;

        location = /favicon.ico {
                log_not_found off;
                access_log off;
        }

        location = /robots.txt {
                allow all;
                log_not_found off;
                access_log off;
        }

        location / {
                try_files $uri $uri/ /index.php?$args;
        }

        location ~ \.php$ {
                include fastcgi.conf;
                fastcgi_intercept_errors on;
                fastcgi_pass php;
                fastcgi_buffers 16 16k;
                fastcgi_buffer_size 32k;
        }

        location ~* \.(js|css|png|jpg|jpeg|gif|ico)$ {
                expires max;
                log_not_found off;
        }
}
```

5. O próximo passo é o arquivo de *tasks*. Edite o arquivo `/home/ansible/ansible/roles/seg10-web/tasks/main.yml` com o seguinte conteúdo:

```
---
- name: Install nginx and deps
  apt:
    name: '{{ packages }}'
    state: present
    update_cache: true
    install_recommends: no
  vars:
```

```
    packages:
    - nginx
    - php-fpm
    - php-mysql
  notify:
    - Start nginx
    - Start php-fpm

- name: Add seg10 config
  copy:
    src: nginx_seg10
    dest: /etc/nginx/sites-available/seg10
    owner: root
    group: root

- name: Disable default site configuration
  file:
    dest: /etc/nginx/sites-enabled/default
    state: absent

- name: Enable seg10 site config
  file:
    src: /etc/nginx/sites-available/seg10
    dest: /etc/nginx/sites-enabled/seg10
    state: link

- name: Copy web root
  synchronize:
    src: seg10
    dest: /var/www
    recursive: yes

- name: Web Root Permissions
  file:
   dest: /var/www/seg10
   mode: u=rwX,g=rX,o=rX
   state: directory
   owner: www-data
   group: www-data
   recurse: yes
  notify:
    - Restart nginx

- name: Adjust php-fpm fix_pathinfo
  replace:
    path: /etc/php/7.0/fpm/php.ini
    regexp: '^;(cgi\.fix_pathinfo).*$'
    replace: '\1=0'
  notify:
    - Restart php-fpm
```

A lista de tarefas acima irá instalar os pacotes na máquina-alvo, copiar a configuração do Nginx, desativar a configuração-padrão e criar o symlink apropriado, sincronizar usando o `rsync` os arquivos do website localizados na pasta `files` para o diretório `/var/www` no destino (copiaremos esses arquivos a seguir), configurará as permissões do *webroot* e ajustará a variável `cgi.fix_pathinfo` do PHP-FPM.

6. Temos *handlers* no arquivo acima, como visto. Edite `/home/ansible/ansible/roles/seg10-web/handlers/main.yml` com o seguinte conteúdo:

```
1 ---
2 - name: Start nginx
3   service:
4     name: nginx
5     state: started
6
7 - name: Start php-fpm
8   service:
9     name: php7.0-fpm
10    state: started
11
12 - name: Restart nginx
13   service:
14    name: nginx
15    state: restarted
16
17 - name: Restart php-fpm
18   service:
19    name: php7.0-fpm
20    state: restarted
```

7. Para que a sincronização dos arquivos do website funcione, temos que obtê-los — use o `scp` para fazer isso:

```
$ scp -rpq www1:/var/www/seg10 /home/ansible/ansible/roles/seg10-web/files/
```

8. Temos que ajustar o escopo das máquinas-alvo da nossa nova *role*. Adicione o novo grupo `seg10_server` ao inventário do Ansible:

```
$ cat << EOF >> ~/ansible/hosts

[seg10_server]
www1
www2
EOF
```

Ao final, o inventário deverá ficar assim:

```
$ cat ~/ansible/hosts
[srv]
www2
www1
db
ns1
ns2
nfs
192.168.42.2
log

[ntp_server]
log

[log_server]
log

[web_server]
log

[seg10_server]
www1
www2
```

9. Vamos criar um novo *playbook* para utilizar a *role* que acabamos de criar:

```
$ cat << EOF >> ~/ansible/seg10web.yml
- hosts: seg10_server
  become: yes
  become_user: root
  become_method: sudo
  roles:
    - seg10-web
EOF
```

10. Hora da verdade! Execute o *playbook*:

```
$ ansible-playbook -i ~/ansible/hosts ~/ansible/seg10web.yml

PLAY [seg10_server]
*********************************************************************************
**************


TASK [Gathering Facts]
*********************************************************************************
***********
```

```
ok: [www1]
ok: [www2]

TASK [seg10-web : Install nginx and deps]
*********************************************************************

ok: [www1]
changed: [www2]

TASK [seg10-web : Add seg10 config]
*************************************************************************************

changed: [www2]
ok: [www1]

TASK [seg10-web : Disable default site configuration]
*************************************************************

ok: [www1]
changed: [www2]

TASK [seg10-web : Enable seg10 site config]
*********************************************************************

ok: [www1]
changed: [www2]

TASK [seg10-web : Copy web root]
*************************************************************************************
*

changed: [www1]
changed: [www2]

TASK [seg10-web : Web Root Permissions]
*********************************************************************

changed: [www1]
changed: [www2]

TASK [seg10-web : Adjust php-fpm fix_pathinfo]
*********************************************************************

changed: [www2]
ok: [www1]

RUNNING HANDLER [seg10-web : Start nginx]
*********************************************************************

ok: [www2]
```

```
RUNNING HANDLER [seg10-web : Start php-fpm]
*********************************************************************

ok: [www2]

RUNNING HANDLER [seg10-web : Restart nginx]
*********************************************************************

changed: [www1]
changed: [www2]

RUNNING HANDLER [seg10-web : Restart php-fpm]
*********************************************************************

changed: [www2]

PLAY RECAP
*********************************************************************************
***********************

www1                       : ok=9    changed=3    unreachable=0    failed=0
www2                       : ok=12   changed=9    unreachable=0    failed=0
```

11. Terá funcionado? Vamos alterar o firewall para redirecionar os pacotes chegando na porta 80/TCP para a máquina `www2`, ao invés da `www1`. Acesse a máquina `ns1` como `root`:

```
# hostname ; whoami
ns1
root
```

Remova as regras que criamos originalmente para a máquina `www1`:

```
# iptables -t nat -D PREROUTING -i enp0s3 -p tcp -m tcp --dport 80 -j DNAT --to
-destination 10.0.42.5
```

```
# iptables -D FORWARD -i enp0s3 -d 10.0.42.5/32 -p tcp -m tcp --dport 80 -j ACCEPT
```

Insira as mesmas regras no firewall, mas agora para a máquina `www2`:

```
# iptables -t nat -A PREROUTING -i enp0s3 -p tcp -m tcp --dport 80 -j DNAT --to
-destination 10.0.42.6
```

```
# iptables -A FORWARD -i enp0s3 -d 10.0.42.6/32 -p tcp -m tcp --dport 80 -j ACCEPT
```

12. Vamos testar. Na máquina `www2`, como o usuário `root`, monitore o log de acesso do Nginx:

```
# hostname ; whoami
www2
root
```

```
# tail -f -n0 /var/log/nginx/access.log
```

No navegador da sua máquina física, acesse novamente o endereço IP da interface `enp0s3` da máquina `ns1`... o **mesmo** website que havíamos configurado antes aparece! Volte a visualizar o log de acesso do Nginx instalado na máquina `www2`:

```
192.168.29.106 - - [17/Nov/2018:03:12:38 -0200] "GET / HTTP/1.1" 200 20854 "-"
"Mozilla/5.0 (Windows NT 10.0; Win64;x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/70.0.3538.102 Safari/537.36"
192.168.29.106 - - [17/Nov/2018:03:12:38 -0200] "GET /wp-
content/themes/twentyseventeen/style.css?ver=4.9.8 HTTP/1.1" 200 83401
"http://192.168.29.104/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.102 Safari/537.36"
192.168.29.106 - - [17/Nov/2018:03:12:38 -0200] "GET /wp-
includes/js/jquery/jquery.js?ver=1.12.4 HTTP/1.1" 200 97184"http://192.168.29.104/"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/70.0.3538.102 Safari/537.36"
```

De fato, os acessos estão chegando à nova máquina, e o website está funcionando como esperado. Note que não fizemos **qualquer** modificação manual na máquina `www2` — todos as dependências, arquivos de configuração e arquivos do site foram instalados de forma automática pelo Ansible. Se quiséssemos lançar agora outros dois, três ou dez nodos de atendimento web, bastaria adicionar as máquinas ao inventário do Ansible e disparar a *role*; toda a configuração é feita de forma automática.

13. Como fizemos uma nova *role* para os sistemas web, é uma boa ideia enviar nossas modificações para o repositório Git central. Acesse a máquina `client` como o usuário `ansible`:

```
$ hostname ; whoami
client
ansible
```

Envie as alterações, adicionando uma mensagem significativa:

```
$ cd ~/ansible/ ; git add . ; git commit -m 'Adicionada role para configuracao
automatica de servidores web do portal seg10' ; git push

(...)

Counting objects: 1665, done.
Compressing objects: 100% (1632/1632), done.
Writing objects: 100% (1665/1665), 8.95 MiB | 8.87 MiB/s, done.
Total 1665 (delta 170), reused 0 (delta 0)
remote: Resolving deltas: 100% (170/170), completed with 1 local object.
To nfs:/home/ansible/ansible.git
   467f80d..bd3fbc2  master -> master
```

6) Configuração do balanceador de carga

1. Vamos para a instalação e configuração do balanceador de carga. Clone a máquina `debian-template` para uma de nome `lb`, com uma única interface de rede conectada à DMZ. O IP da máquina será 10.0.42.8/24.

   Concluída a clonagem, ligue a máquina e logue como `root`. Depois, use o script `/root/scripts/changehost.sh` para fazer a configuração automática, como de costume.

   ```
   # hostname ; whoami
   debian-template
   root
   ```

   ```
   # bash ~/scripts/changehost.sh -h lb -i 10.0.42.8 -g 10.0.42.1
   Signing ssh_host_ecdsa_key.pub key...
   Signing ssh_host_ed25519_key.pub key...
   Signing ssh_host_rsa_key.pub key...
   Configuring host key trust...
   Configuring user key trust...
   All done!
   ```

2. Aplique o *baseline* de segurança à máquina `lb`, repetindo o que fizemos no passo (2) da atividade (2) desta sessão:

   ```
   $ hostname ; whoami
   client
   ansible
   ```

   ```
   $ sed -i '/\[srv\]/a lb' ~/ansible/hosts
   ```

```
$ ansible-playbook -i ~/ansible/hosts -l lb -Ke ansible_become_method=su
~/ansible/srv.yml ; ansible-playbook -i ~/ansible/hosts -l lb ~/ansible/srv.yml
SUDO password:

(...)

PLAY RECAP
*********************************************************************************
**********************

lb                         : ok=10   changed=8    unreachable=0    failed=0
```

3. Acesse a máquina `lb` como o usuário `root`:

```
# hostname ; whoami
lb
root
```

4. Vamos instalar o balanceador de carga: usaremos o software HAProxy, uma solução *open-source* para balanceamento de carga e alta disponibilidade para aplicações baseadas em TCP e HTTP, reconhecido por sua excepcional performance e eficiência.

```
# apt-get install -y haproxy
```

5. Edite o arquivo de configuração do HAProxy, `/etc/haproxy/haproxy.cfg`, deixando-o exatamente com o conteúdo a seguir:

```
 1 global
 2   log /dev/log    local0
 3   log /dev/log    local1 notice
 4   chroot /var/lib/haproxy
 5   stats socket /run/haproxy/admin.sock mode 660 level admin
 6   stats timeout 30s
 7   user haproxy
 8   group haproxy
 9   daemon
10   maxconn 2048
11
12   # Default SSL material locations
13   ca-base /etc/ssl/certs
14   crt-base /etc/ssl/private
15
16   # Default ciphers to use on SSL-enabled listening sockets.
17   # For more information, see ciphers(1SSL). This list is from:
18   #  https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/
19   # An alternative list with additional directives can be obtained from
```

```
20   #  https://mozilla.github.io/server-side-tls/ssl-config-
generator/?server=haproxy
21   ssl-default-bind-ciphers ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH
+AES128:DH+AES:RSA+AESGCM:RSA+AES:!aNULL:!MD5:!DSS
22   ssl-default-bind-options no-sslv3
23   tune.ssl.default-dh-param 2048
24
25 defaults
26   log     global
27   mode    http
28   option  httplog
29   option  dontlognull
30   option  forwardfor
31   option  http-server-close
32   timeout connect 5000
33   timeout client  50000
34   timeout server  50000
35   errorfile 400 /etc/haproxy/errors/400.http
36   errorfile 403 /etc/haproxy/errors/403.http
37   errorfile 408 /etc/haproxy/errors/408.http
38   errorfile 500 /etc/haproxy/errors/500.http
39   errorfile 502 /etc/haproxy/errors/502.http
40   errorfile 503 /etc/haproxy/errors/503.http
41   errorfile 504 /etc/haproxy/errors/504.http
42
43   stats enable
44   stats uri /stats
45   stats realm Haproxy\ Statistics
46   stats auth admin:rnpesr123
47
48 frontend www-http
49   bind 10.0.42.8:80
50   reqadd X-Forwarded-Proto:\ http
51   default_backend www-backend
52
53 frontend www-https
54   bind 10.0.42.8:443 ssl crt /etc/ssl/private/seg10web.pem
55   reqadd X-Forwarded-Proto:\ https
56   default_backend www-backend
57
58 backend www-backend
59   redirect scheme https if !{ ssl_fc }
60   server www1 www1.intnet:80 check
61   server www2 www2.intnet:80 check
```

Em linhas gerais, definimos dois *frontends* de atendimento, um para conexões HTTP e outro para HTTPS. Ambos enviam suas requisições para o mesmo *backend*, o qual é atendido pelos servidores `www1` e `www2`, escutando na porta 80/TCP.

6. Note que para o atendimento das requisições HTTPS (o chamado *SSL offloading*, ou *SSL*

termination), devemos configurar um par de chaves pública/privada auto-assinadas em formato PEM. Vamos fazer isso:

```
# openssl req -x509 -nodes -days 730 -newkey rsa:4096 -keyout
/etc/ssl/private/seg10web.key -out /etc/ssl/certs/seg10web.crt
Generating a 4096 bit RSA private key
..............................................++
.....................................................................................
.....................................................................................
.....................................................................................
.....................................++
writing new private key to '/etc/ssl/private/seg10web.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:BR
State or Province Name (full name) [Some-State]:DF
Locality Name (eg, city) []:Brasilia
Organization Name (eg, company) [Internet Widgits Pty Ltd]:RNP
Organizational Unit Name (eg, section) []:ESR
Common Name (e.g. server FQDN or YOUR name) []:seg10web.intnet
Email Address []:seg10web@int.net
```

Concatene as duas chaves em um arquivo em formato PEM:

```
# cat /etc/ssl/certs/seg10web.crt /etc/ssl/private/seg10web.key >
/etc/ssl/private/seg10web.pem
```

```
# chmod 600 /etc/ssl/private/seg10web.pem
```

7. Agora que tudo está configurado, reinicie o HAProxy:

```
# systemctl restart haproxy.service
```

8. Temos que alterar o firewall uma última vez, desta vez para redirecionar os pacotes chegando na porta 80/TCP para a máquina `lb`. Acesse a máquina `ns1` como `root`:

```
# hostname ; whoami
ns1
root
```

Remova as regras que criamos para a máquina `www2`:

```
# iptables -t nat -D PREROUTING -i enp0s3 -p tcp -m tcp --dport 80 -j DNAT --to
-destination 10.0.42.6
```

```
# iptables -D FORWARD -i enp0s3 -d 10.0.42.6/32 -p tcp -m tcp --dport 80 -j ACCEPT
```

Adicione-as para o balanceador de carga — observe que como o HAProxy escuta tanto na porta 80/TCP como na 443/TCP, iremos redirecioná-las ambas:

```
# iptables -t nat -A PREROUTING -i enp0s3 -p tcp -m multiport --dports 80,443 -j
DNAT --to-destination 10.0.42.8
```

```
# iptables -A FORWARD -i enp0s3 -d 10.0.42.8/32 -p tcp -m multiport --dports 80,443
-j ACCEPT
```

Salve a configuração do firewall desta vez, já que ela será definitiva:

```
# /etc/init.d/netfilter-persistent save
[....] Saving netfilter rules...run-parts: executing /usr/share/netfilter-
persistent/plugins.d/15-ip4tables save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
done.
```

9. Vamos ao teste! No navegador da sua máquina física, acesse novamente o endereço IP da interface `enp0s3` da máquina `ns1`. De imediato, notamos uma diferença: o acesso está sendo feito em HTTPS, já que a diretiva `redirect scheme https if !{ ssl_fc }` do arquivo de configuração do HAProxy força esse protocolo aos clientes que estão se conectando.

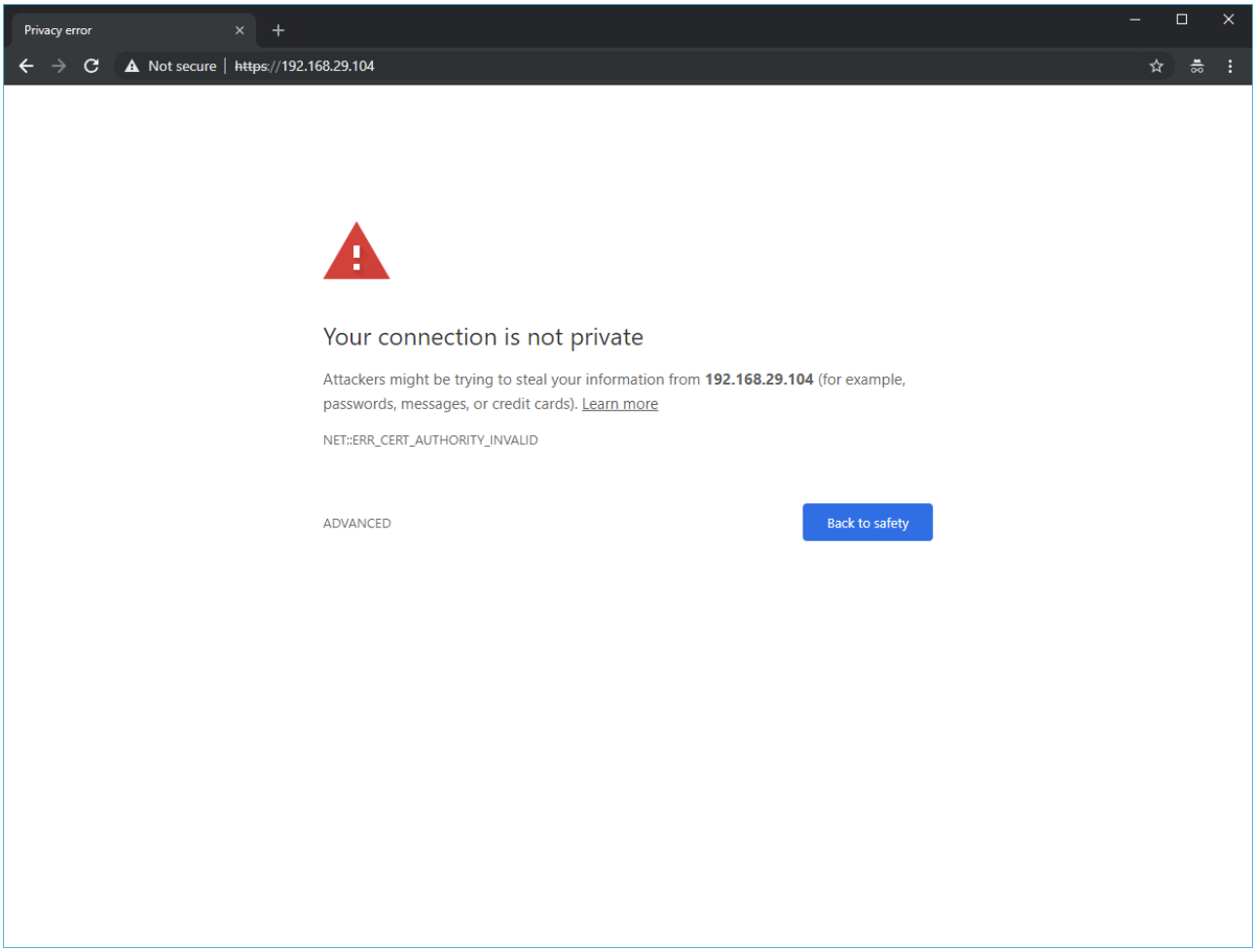

Figura 47. Acesso desviado para HTTPS pelo HAProxy

Aceite o certificado auto-assinado, e prossiga para a página do portal *Seg10 Web*. Temos um problema! O CSS da página não foi carregado corretamente:

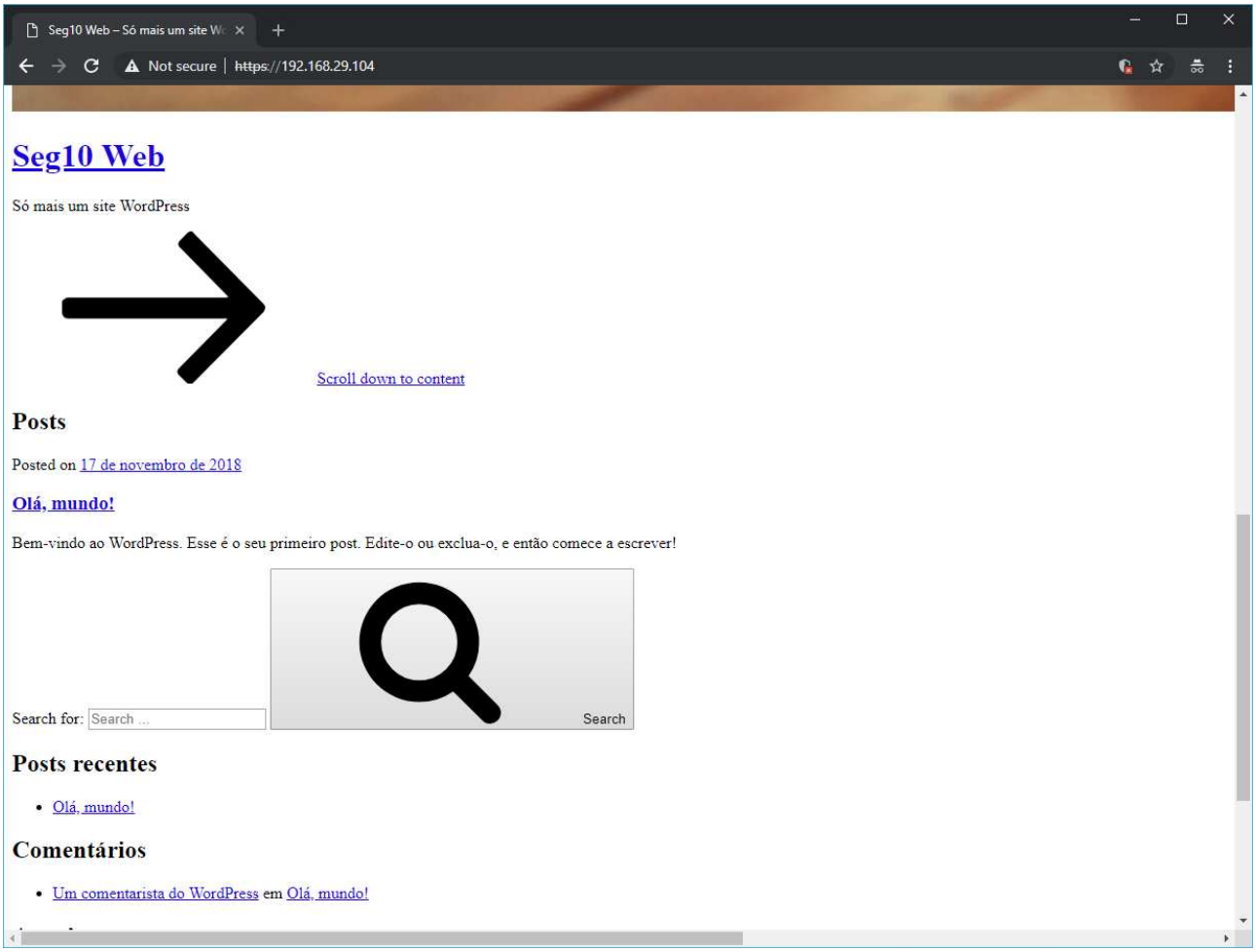

Figura 48. CSS não foi carregado na conexão HTTPS

O que fazemos? Felizmente, esse problema é facilmente solucionável. Acesse a máquina `client` como o usuário `ansible`.

```
$ hostname ; whoami
client
ansible
```

Basta inserir a linha `if ($_SERVER['HTTP_X_FORWARDED_PROTO'] == 'https') $_SERVER['HTTPS']='on';` antes da última linha do arquivo `/home/ansible/ansible/roles/seg10-web/files/seg10/wp-config.php`. O comando `sed` a seguir faz a alteração de forma direta:

```
$ sed -i "/wp-settings/i if (\$_SERVER['HTTP_X_FORWARDED_PROTO'] == 'https')
\$_SERVER['HTTPS']='on';" /home/ansible/ansible/roles/seg10-web/files/seg10/wp-
config.php
```

Uhm... como distribuir essas mudanças? Simples: basta re-executar nossa *role* `seg10-web` no Ansible:

```
$ ansible-playbook -i ~/ansible/hosts ~/ansible/seg10web.yml

(...)

TASK [seg10-web : Copy web root]
*********************************************************************************
*

changed: [www1]
changed: [www2]

TASK [seg10-web : Web Root Permissions]
*********************************************************************

changed: [www1]
changed: [www2]

(...)

PLAY RECAP
*********************************************************************************
************************

www1                       : ok=9    changed=3    unreachable=0    failed=0
www2                       : ok=9    changed=3    unreachable=0    failed=0
```

O Ansible detecta que a cópia local está diferente da observada nas máquinas remotas, e sobrescreve o arquivo `/var/www/seg10/wp-config.php` em ambos os servidores web com as modificações locais.

Vamos verificar se isso solucionou o problema no portal. Recarregue a página web no navegador em sua máquina física:

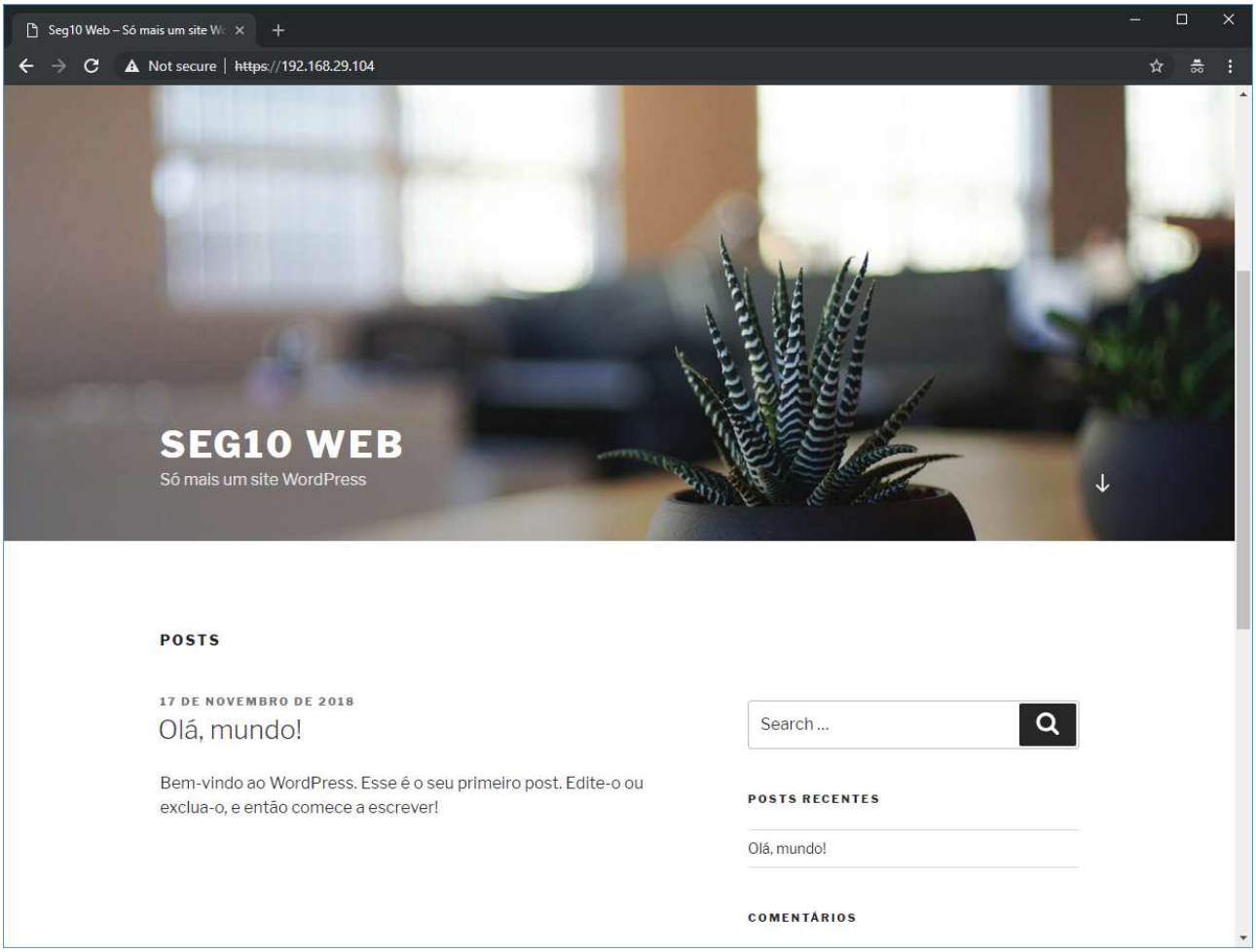

Figura 49. Correção na visualização da página via HTTPS

Excelente, tudo funcionando a contento.

10. O HAProxy possui uma página de estatísticas bastante interessante para acompanhar o funcionamento dos diferentes *frontends* e *backends* configurados. Adicione o sufixo `/stats` à URL do seu navegador, e faça login com o usuário `admin` e senha `rnpesr123` quando solicitado. Você verá a página a seguir:

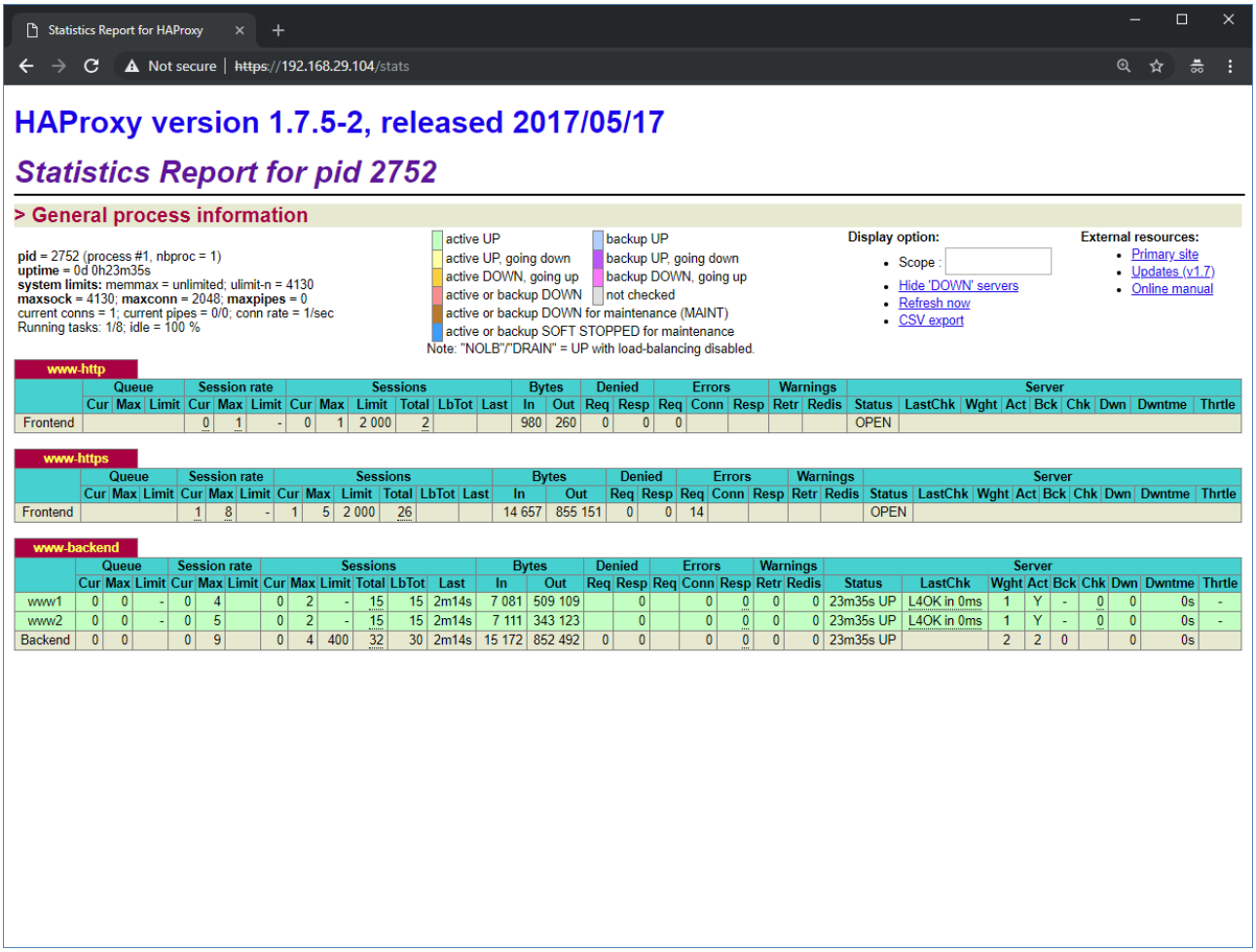

Figura 50. Página de estatísticas do HAProxy

A partir desta página, é possível verificar quais *frontends* e *backends* estão ativos, quais estão inativos ou com problemas de acesso, e quantos bytes e sessões foram trafegados para cada um dos *hosts* envolvidos.

Tente recarregar a portal *Seg10 Web* algumas vezes, e confira o número de sessões registradas para cada um dos *backends*: o HAProxy envia requisições alternadas para cada um deles, segundo um algoritmo *round robin*. Com o HAProxy, é fácil desativar seletivamente alguns *backends* para atualizações ou manutenção, e levantá-los posteriormente, mantendo a disponibilidade do serviço e minimizando o impacto para os clientes.

11. Encerradas as nossas atividades com os servidores, recomenda-se que o aluno mantenha as máquinas `www1`, `www2`, `db` e `lb` desligadas a partir desta sessão. Apesar de as máquinas individualmente não ocuparem tantos recursos de memória e processamento, o fato de não precisarmos mais usá-las e a exigência de alteração de contexto da CPU para atendê-las faz com que a liberação de recursos, nesse caso, seja recomendável.

Sessão 8: Isolamento de processos e conteinerização

Nesta sessão, iremos configurar o sistema de conteinerização Docker, testando suas funcionalidades de criação rápida de containers, escalabilidade e orquestração via Docker Swarm. Compararemos a facilidade e rapidez de configuração com um sistema web mais "tradicional", como o que fizemos na sessão anterior.

1) Topologia desta sessão

Criaremos duas novas máquinas nesta sessão, a saber:

- `docker1`, nodo-mestre de configuração do Docker Compose/Swarm. Endereço IP 10.0.42.9/24.
- `docker2`, nodo-escravo (ou *worker*) do Docker Compose/Swarm. Endereço IP 10.0.42.10/24.

1. Como de costume, vamos à criação dos registros DNS. Acesse a máquina `ns1` como o usuário `root`:

   ```
   # hostname ; whoami
   ns1
   root
   ```

   Edite o arquivo de zonas `/etc/nsd/zones/intnet.zone`, inserindo entradas A para a máquinas indicadas no começo desta atividade. **Não se esqueça** de incrementar o valor do serial no topo do arquivo!

   ```
   # nano /etc/nsd/zones/intnet.zone
   (...)
   ```

   ```
   # grep docker /etc/nsd/zones/intnet.zone
   docker1 IN    A                10.0.42.9
   docker2 IN    A                10.0.42.10
   ```

   Faça o mesmo para o arquivo de zona reversa:

   ```
   # nano /etc/nsd/zones/10.0.42.zone
   ```

   ```
   # grep docker /etc/nsd/zones/10.0.42.zone
   9       IN   PTR               docker1.intnet.
   10      IN   PTR               docker2.intnet.
   ```

   Assine o arquivo de zonas usando o *script* criado anteriormente:

```
# bash /root/scripts/signzone-intnet.sh
reconfig start, read /etc/nsd/nsd.conf
ok
ok
ok
ok removed 6 rrsets, 2 messages and 0 key entries
```

Verifique a criação das entradas usando o comando `dig`:

```
# for host in docker1 docker2; do echo -n "$host: "; dig ${host}.intnet +short;
done
docker1: 10.0.42.9
docker2: 10.0.42.10
```

```
# for ip in 9 10; do echo -n "10.0.42.${ip}: "; dig -x 10.0.42.${ip} +short; done
10.0.42.9: docker1.intnet.
10.0.42.10: docker2.intnet.
```

2) Criação da VM docker1 e instalação

1. Primeiro, vamos criar a VM `docker1` e instalar o software Docker nela. Clone a máquina `debian-template` para uma de nome `docker1`, com uma única interface de rede conectada à DMZ. O IP da máquina será 10.0.42.9/24.

   Concluída a clonagem, ligue a máquina e logue como `root`. Depois, use o script `/root/scripts/changehost.sh` para fazer a configuração automática, como de costume.

   ```
   # hostname ; whoami
   debian-template
   root
   ```

   ```
   # bash ~/scripts/changehost.sh -h docker1 -i 10.0.42.9 -g 10.0.42.1
   Signing ssh_host_ecdsa_key.pub key...
   Signing ssh_host_ed25519_key.pub key...
   Signing ssh_host_rsa_key.pub key...
   Configuring host key trust...
   Configuring user key trust...
   All done!
   ```

2. Aplique o *baseline* de segurança à máquina `docker1`, repetindo o que fizemos no passo (2), atividade (2) da sessão 7:

```
$ hostname ; whoami
client
ansible
```

```
$ sed -i '/\[srv\]/a docker1' ~/ansible/hosts
```

```
$ ansible-playbook -i ~/ansible/hosts -l docker1 -Ke ansible_become_method=su
~/ansible/srv.yml ; ansible-playbook -i ~/ansible/hosts -l docker1
~/ansible/srv.yml
SUDO password:

(...)

PLAY RECAP
*********************************************************************************
***********************

docker1                    : ok=10   changed=8    unreachable=0    failed=0
```

3. Agora, acesse a máquina `docker1` como o usuário `root`:

```
# hostname ; whoami
docker1
root
```

4. Vamos proceder com os passos de instalação seguindo o manual oficial do Docker, disponível https://docs.docker.com/install/linux/docker-ce/debian/#install-docker-ce . Primeiramente, vamos habilitar a instalação de pacotes APT via HTTPS, instalando os pacotes a seguir:

```
# apt-get install -y \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg2 \
    software-properties-common
```

5. Agora, adicione a chave GPG do repositório do Docker com o comando:

```
# curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -
OK
```

Verifique que a chave foi recebida corretamente:

```
# apt-key fingerprint 0EBFCD88
pub   rsa4096 2017-02-22 [SCEA]
      9DC8 5822 9FC7 DD38 854A  E2D8 8D81 803C 0EBF CD88
uid           [ unknown] Docker Release (CE deb) <docker@docker.com>
sub   rsa4096 2017-02-22 [S]
```

6. Adicione o repositório do Docker à lista de repositórios disponíveis para instalação de pacotes:

```
# echo "deb [arch=amd64] https://download.docker.com/linux/debian \
    $(lsb_release -cs) \
    stable" > \
    /etc/apt/sources.list.d/docker.list
```

   Atualize a lista de pacotes disponíveis, e instale o `docker-ce`:

```
# apt-get update ; apt-get install -y docker-ce
```

7. Cheque qual versão do Docker foi instalada:

```
# docker --version
Docker version 18.09.0, build 4d60db4
```

   Verifique a correta instalação do Docker rodando a imagem `hello-world`, uma imagem de teste:

```
# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
d1725b59e92d: Pull complete
Digest: sha256:0add3ace90ecb4adbf7777e9aacf18357296e799f81cabc9fde470971e499788
Status: Downloaded newer image for hello-world:latest

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

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

O Docker detecta que a imagem `hello-world` não está disponível localmente, faz o download da mesma a partir do *registry* global do Docker Hub, e a executa. Falaremos mais sobre o *registry* global em atividades posteriores.

8. Agora, desligue a VM `docker1`. Para não ter que repetir os passos de instalação na VM `docker2`, vamos cloná-la a partir da `docker1` e aproveitar o trabalho que já realizamos até aqui.

```
# halt -p
```

3) Criação da VM docker2

1. Com a VM `docker1` desligada, clone-a para uma máquina de nome `docker2`. O IP dessa máquina será 10.0.42.10/24.

   Concluída a clonagem, ligue **apenas** a máquina `docker2` e logue como `root`. Não ligue a máquina `docker1` ainda, ou haverá um conflito de IP na rede. Observe: como a máquina `docker1` já estava configurada para operar com o `sudo` distribuído via Ansible, será necessário escalar privilégio a partir de um usuário autorizado, como `aluno`.

```
$ hostname ; whoami
docker1
aluno
```

```
$ sudo -i
[sudo] senha para aluno:
root@docker1:~#
```

Depois, use o script `/root/scripts/changehost.sh` para fazer a configuração automática, como de costume.

```
# bash ~/scripts/changehost.sh -h docker2 -i 10.0.42.10 -g 10.0.42.1
Signing ssh_host_ecdsa_key.pub key...
Signing ssh_host_ed25519_key.pub key...
Signing ssh_host_rsa_key.pub key...
Configuring host key trust...
Configuring user key trust...
All done!
```

2. Para manter a organização em nosso ambiente, adicione a máquina `docker2` ao inventário do Ansible. Como o usuário `ansible` na máquina `cliente`, execute:

```
$ hostname ; whoami
client
ansible
```

```
$ sed -i '/\[srv\]/a docker2' ~/ansible/hosts
```

Note que não é necessário re-executar o *playbook* para a máquina `docker2`, já que todos os controles de segurança foram aplicados anteriormente à máquina `docker1`, a partir da qual fizemos a clonagem.

3. Feito isso, ligue também a máquina `docker1`, e prossiga com as atividades desta sessão.

4) Trabalhando com containers

1. Acesse a máquina `docker1` como o usuário `root`:

```
# hostname ; whoami
docker1
root
```

Agora, crie um diretório vazio `/root/docker`, e entre nele.

```
# mkdir ~/docker ; cd ~/docker
```

2. Vamos criar um *Dockerfile* — um arquivo que define o que será instalado e configurado dentro do seu container. Nesse arquivo são mapeados acessos a recursos como interfaces de rede e volumes de disco virtualizados, em um ambiente isolado do restante do sistema operacional.

   Crie o arquivo novo `/root/docker/Dockerfile` com o seguinte conteúdo:

```
 1 # Usar uma imagem oficial do runtime Python como imagem-pai
 2 FROM python:2.7-slim
 3
 4 # Configurar o diretorio de trabalho como /app
 5 WORKDIR /app
 6
 7 # Copiar o conteudo do diretorio corrent para dentro do container em /app
 8 COPY . /app
 9
10 # Instalar quaisquer dependencias do Python especificadas no arquivo
requirements.txt
11 RUN pip install --trusted-host pypi.python.org -r requirements.txt
12
13 # Expor a porta 80 para o mundo externo, fora do container
14 EXPOSE 80
15
16 # Definir uma variavel de ambiente $World
17 ENV NAME World
18
19 # Rodar a aplicacao app.py ao lancar o container
20 CMD ["python", "app.py"]
```

   Esse *Dockerfile* faz referência a dois arquivos que ainda não criamos — `app.py` e `requirements.txt`. Vamos criá-los.

3. Primeiro, crie o arquivo novo `/root/docker/requirements.txt` com o seguinte conteúdo:

```
1 Flask
2 Redis
```

   O comando `pip install -r requirements.txt`, invocado no *Dockerfile*, irá portanto instalar as bibliotecas Flask e Redis para o ambiente Python do container.

4. Agora, vamos à aplicação em si. Crie o arquivo novo `/root/docker/app.py` com o seguinte conteúdo:

```
from flask import Flask
from redis import Redis, RedisError
import os
import socket

# Connect to Redis
redis = Redis(host="redis", db=0, socket_connect_timeout=2, socket_timeout=2)

app = Flask(__name__)

@app.route("/")
def hello():
    try:
        visits = redis.incr("counter")
    except RedisError:
        visits = "<i>cannot connect to Redis, counter disabled</i>"

    html = "<h3>Hello {name}!</h3>" \
           "<b>Hostname:</b> {hostname}<br/>" \
           "<b>Visits:</b> {visits}"
    return html.format(name=os.getenv("NAME", "world"), hostname=socket
.gethostname(), visits=visits)

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80)
```

A aplicação acima, bastante simples, irá exibir a *string* `Hello World!`, uma página web que mostra o *hostname* da máquina local (no caso, o identificador do container) e um contador do número de visitas realizadas ao site. Esse contador é mantido em um volume uniforme, acessível por todos os containers da aplicação, com a biblioteca Redis.

5. Liste o conteúdo do diretório `/root/docker`. Você deve ter os arquivos abaixo:

```
# ls -1 ~/docker/
app.py
Dockerfile
requirements.txt
```

Para fazer o *build* do container, basta rodar o comando `docker build`:

```
# cd ~/docker ; docker build -t pyhello .
Sending build context to Docker daemon   5.12kB
Step 1/7 : FROM python:2.7-slim
2.7-slim: Pulling from library/python
a5a6f2f73cd8: Pull complete
8da2a74f37b1: Pull complete
09b6f498cfd0: Pull complete
f0afb4f0a079: Pull complete
Digest: sha256:f82db224fbc9ff3309b7b62496e19d673738a568891604a12312e237e01ef147
Status: Downloaded newer image for python:2.7-slim
 ---> 0dc3d8d47241
Step 2/7 : WORKDIR /app
(...)
Step 3/7 : COPY . /app
(...)
Step 4/7 : RUN pip install --trusted-host pypi.python.org -r requirements.txt
(...)
Step 5/7 : EXPOSE 80
(...)
Step 6/7 : ENV NAME World
(...)
Step 7/7 : CMD ["python", "app.py"]
(...)
Successfully built d2923f9142e3
Successfully tagged pyhello:latest
```

O Docker irá executar os comandos do *Dockerfile*, em ordem:

1. Ao detectar que a imagem `python:2.7-slim` não existe na máquina local, ela será baixada do *registry* global do Docker Hub, como feito anteriormente com a imagem `hello-world.`
2. Deriva-se uma nova imagem a partir de `python:2.7-slim`, e o diretório `/app` é criado na raiz do container.
3. Os arquivos da pasta local são copiadas para `/app`.
4. O comando `pip install -r requirements.txt` instala as bibliotecas necessárias ao funcionamento da aplicação, Flask e Redis, bem como suas dependências.
5. A porta 80/TCP do container é exposta para o mundo externo.
6. Cria-se uma nova variável de ambiente, `$World`.
7. Roda-se o comando `python app.py`, executando a aplicação. Como este comando objetiva apenas a criação da imagem do container, a aplicação é encerrada logo em seguida, e a imagem do container é finalizada sob a *tag* `pyhello`.

Para listar a imagem recém-criada, use o comando `docker image ls`:

```
# docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
pyhello             latest              d2923f9142e3        6 minutes ago
131MB
python              2.7-slim            0dc3d8d47241        36 hours ago
120MB
hello-world         latest              4ab4c602aa5e        2 months ago
1.84kB
```

6. Para rodar o container, basta executar `docker run`:

```
# docker run -p 7080:80 pyhello
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:80/ (Press CTRL+C to quit)
```

   O comando acima irá iniciar o container escutando na porta 80/TCP, e mapeando-a para a porta 7080/TCP da máquina virtual `docker1`.

7. Para conseguir acessar o container a partir do IP público do firewall (interface `enps0s3` da máquina `ns1`), precisamos adicionar algumas regras novas. Acesse a máquina `ns1` como `root`:

```
# hostname ; whoami
ns1
root
```

   Para que consigamos atingir a máquina `docker1` será necessário criar uma regra de DNAT na tabela `nat`, *chain* PREROUTING, além de uma regra na tabela `filter`, *chain* FORWARD, correspondente. Mapearemos a porta externa 7080/TCP para a porta interna 7080/TCP, sem alterações.

```
# iptables -t nat -A PREROUTING -i enp0s3 -p tcp -m tcp --dport 7080 -j DNAT --to
-destination 10.0.42.9
```

```
# iptables -A FORWARD -i enp0s3 -d 10.0.42.9/32 -p tcp -m tcp --dport 7080 -j
ACCEPT
```

8. Em sua máquina física, abra o navegador e aponte-o para o IP público do firewall (interface `enps0s3` da máquina `ns1`), na porta 7080/TCP:

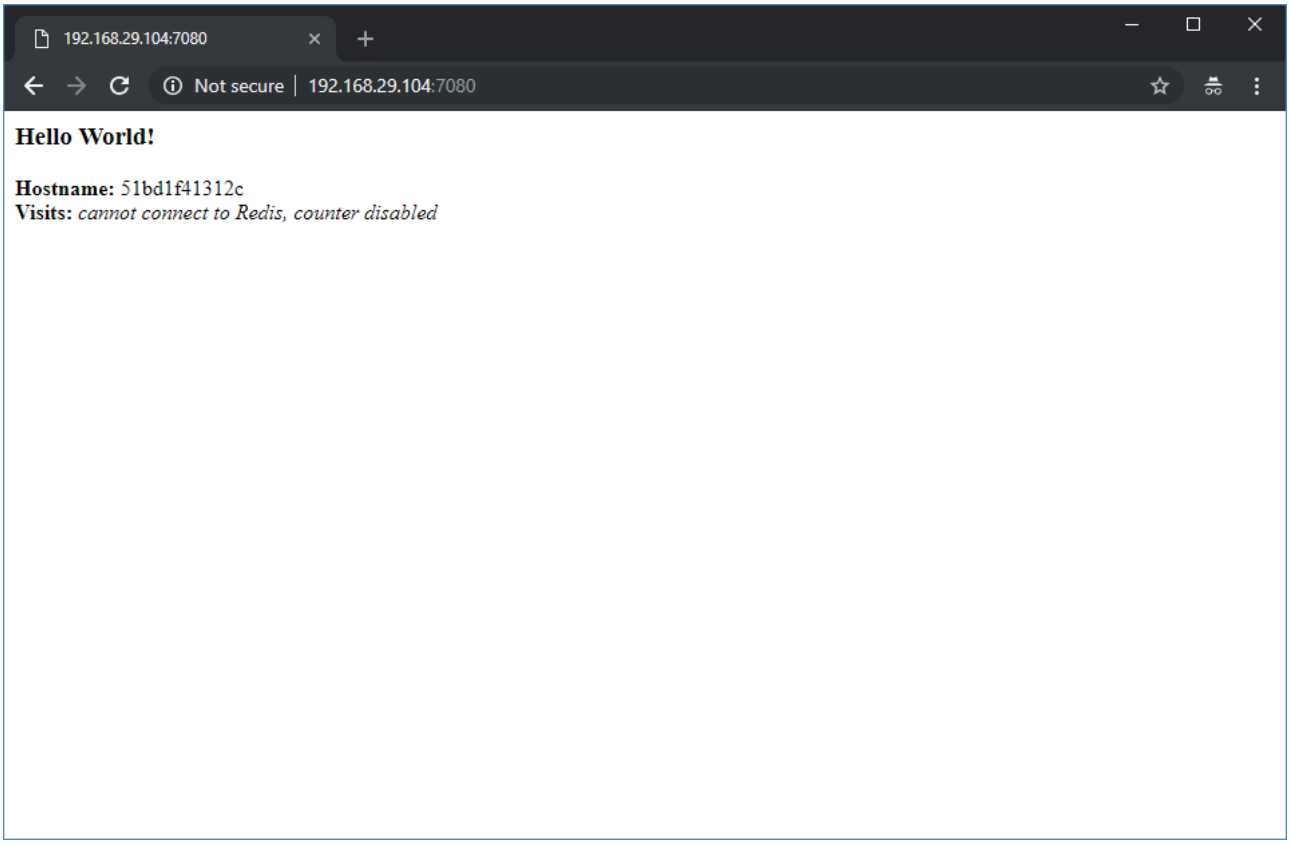

Figura 51. Container operacional no Docker

Tudo certo!

9. De volta à máquina `docker1` como `root`, note que o acesso que fizemos foi registrado na console, com as seguintes mensagens:

```
192.168.29.106 - - [17/Nov/2018 20:10:53] "GET / HTTP/1.1" 200 -
192.168.29.106 - - [17/Nov/2018 20:10:53] "GET /favicon.ico HTTP/1.1" 404 -
```

Para encerrar o container, digite `CTRL + C`. Vamos reexecutá-lo em *background* com a opção `-d` (*detached*):

```
# docker run -d -p 7080:80 pyhello
2d93cc85f066dc6afa9a5c9ea5d0fd08112f52cec99aad8d1d862608d67ddc81
```

O ID do container é mostrado, e retomamos controle do terminal. Para visualizar quais containers estão em operação neste momento, use o comando `docker container ls`:

```
# docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS
      NAMES
2d93cc85f066        pyhello             "python app.py"     42 seconds ago      Up
41 seconds       0.0.0.0:7080->80/tcp   modest_stallman
```

Tente acessar novamente o container no navegador em sua máquina física — ele está funcionando normalmente.

Para parar um container rodando em *background*, use `docker container stop` e passe como parâmetro o ID do container, assim:

```
# docker container stop 2d93cc85f066
2d93cc85f066
```

5) Distribuindo containers para um *registry* externo

1. Vamos distribuir o container que criamos no passo anterior para o *registry* global do Docker Hub. Para isso, o primeiro passo é criar uma conta em https://hub.docker.com/ . Acesse essa página através do navegador em sua máquina física e preencha os campos em *New to Docker?*; **importante**: use um endereço de e-mail real em seu cadastro.

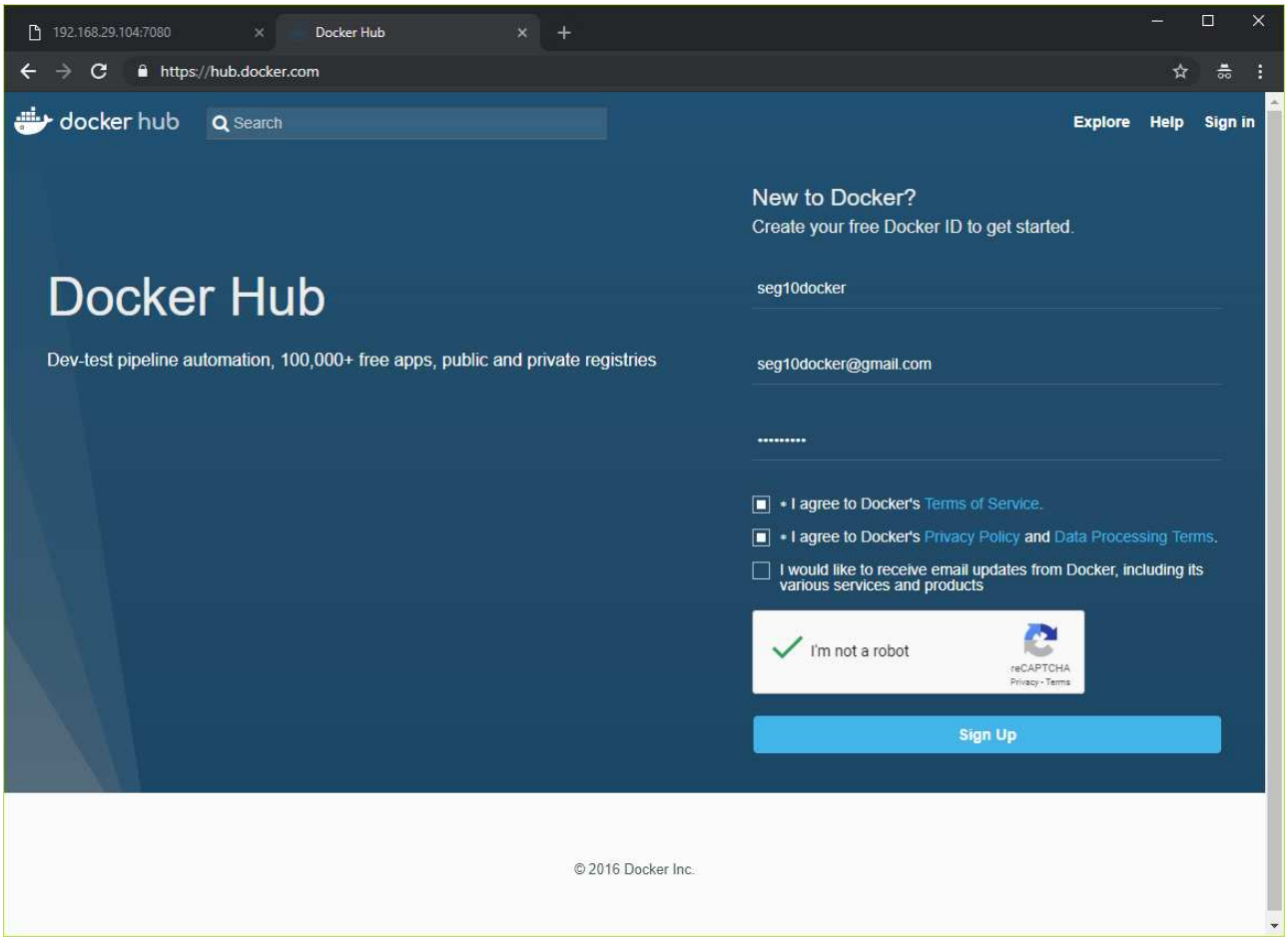

Figura 52. Cadastro no Docker Hub

Após seu cadastro, o Docker Hub irá enviar um e-mail de confirmação. Acesse a conta de e-mail informada e clique no botão para completar o cadastro. Finalmente, faça login no Docker Hub usando sua conta:

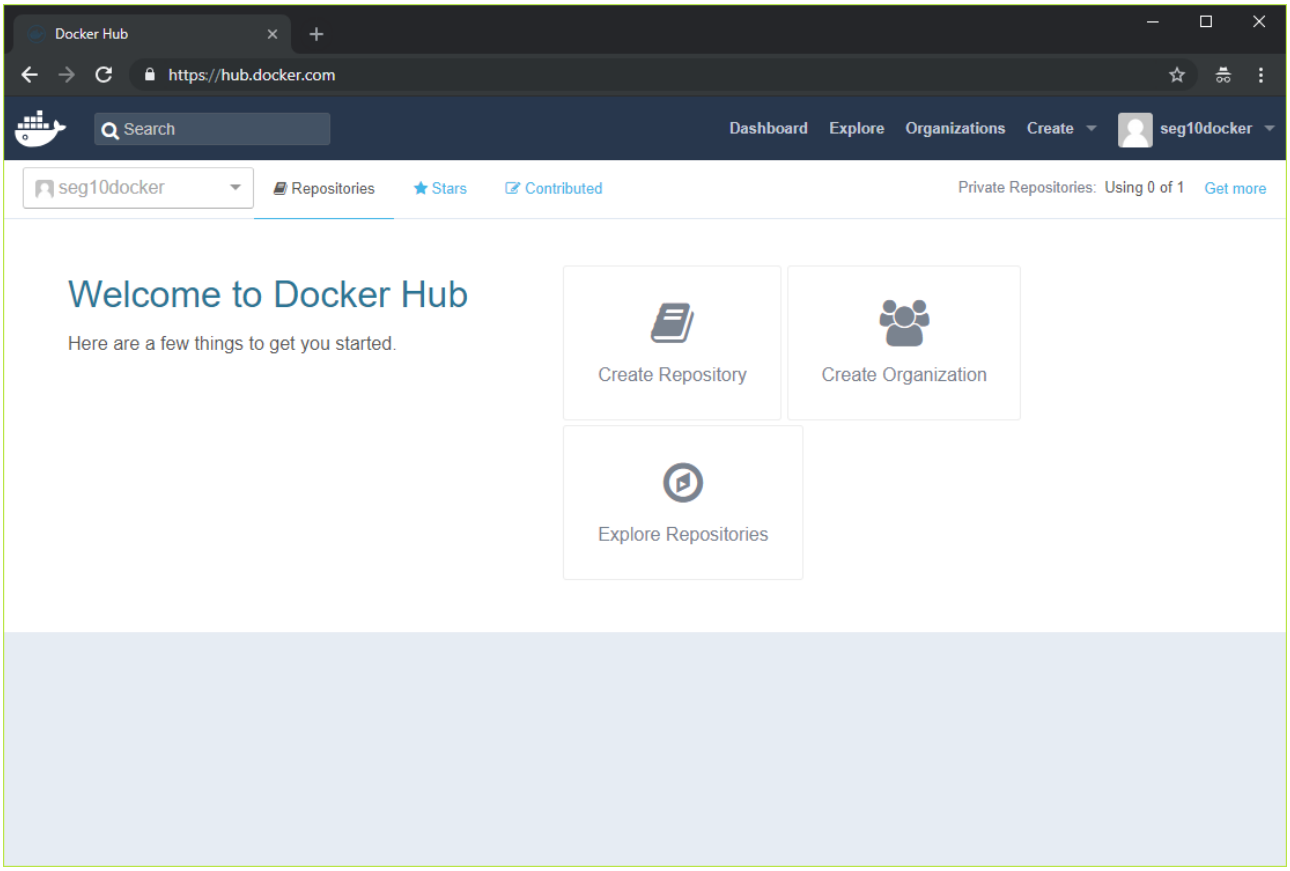

Figura 53. Interface do Docker Hub

Na tela acima, clique em *Create Repository*. Na nova tela, digite `seg10` como o nome do repositório; em *Description*, informe `Repositório-teste para o curso SEG10`; mantenha *Visibility* como `Public`.

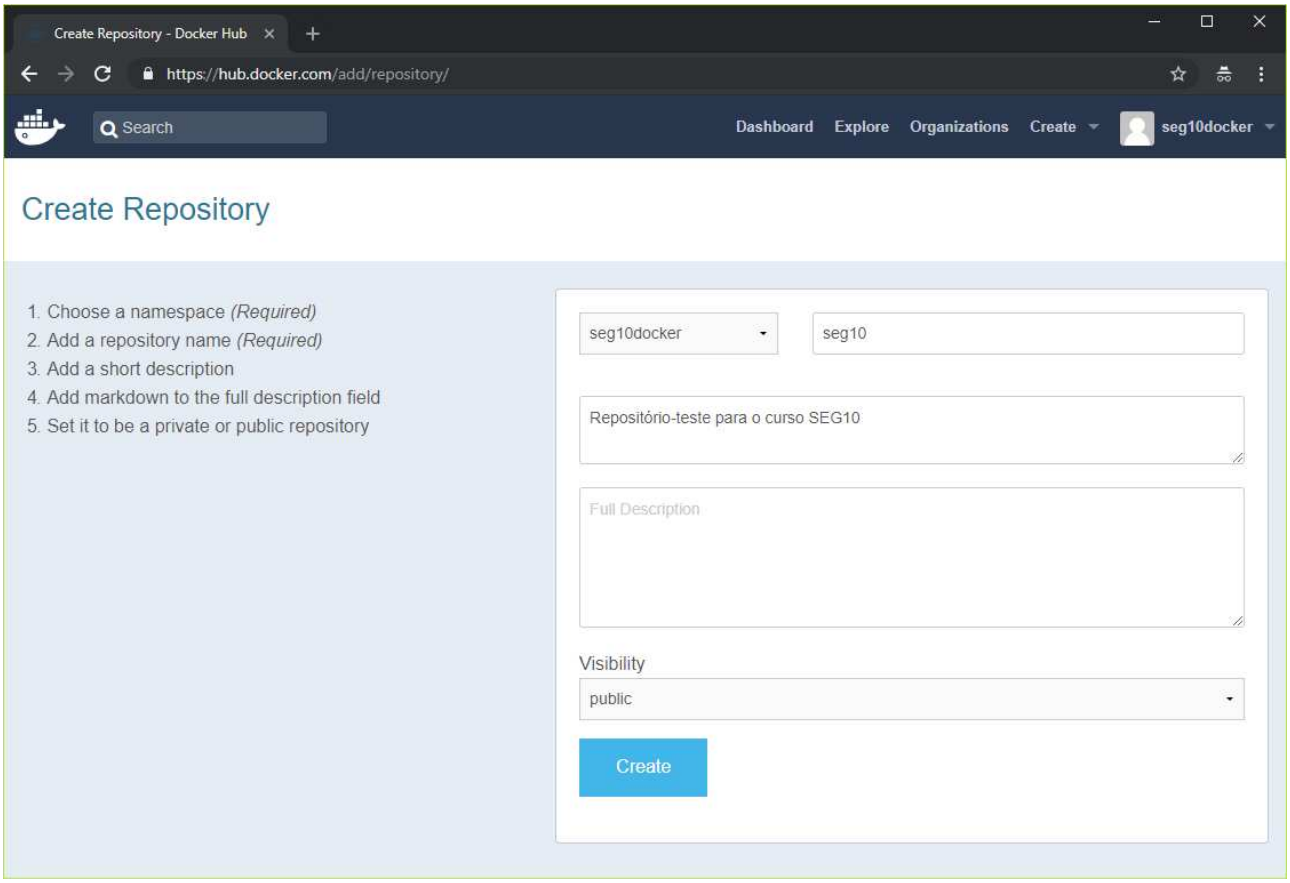

Figura 54. Criação de novo repositório no Docker Hub

Clique em *Create*. Concluído o processo, você verá seu novo repositório como na tela a seguir:

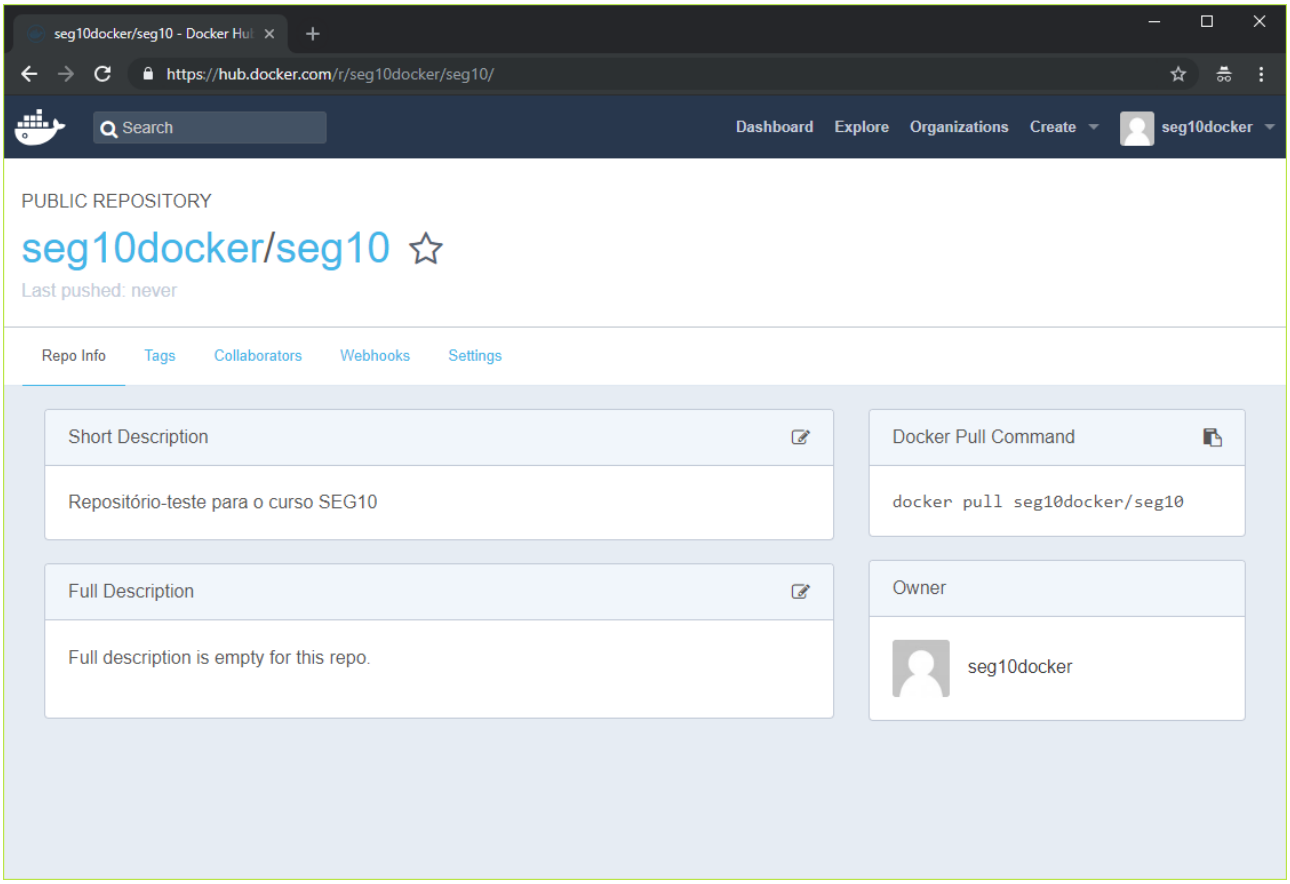

Figura 55. Repositório seg10 no Docker Hub

2. Agora, volte à máquina `docker1` como o usuário `root`.

```
# hostname ; whoami
docker1
root
```

Para fazer login no Docker Hub via linha de comando, use `docker login`. Use a mesma combinação de usuário e senha que você criou no passo (1) desta atividade.

```
# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't
have a Docker ID, head over to https://hub.docker.com to create one.
Username: seg10docker
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

3. O próximo passo é criar uma *tag* para a imagem de container que criamos na atividade anterior. Uma *tag* é descrita por uma combinação `usuário/repositório:tag`, e serve para identificar e versionar imagens de containers no Docker.

Para criar a *tag*, use o comando `docker tag image` — substitua no comando abaixo o nome de usuário `seg10docker` pelo usuário que você criou no Docker Hub no passo (1) desta atividade:

```
# docker tag pyhello seg10docker/seg10:pyhello-v1
```

Para ver a nova imagem criada com a *tag*, use `docker image ls`:

```
# docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
seg10docker/seg10   pyhello-v1          d2923f9142e3        37 minutes ago
131MB
pyhello             latest              d2923f9142e3        37 minutes ago
131MB
python              2.7-slim            0dc3d8d47241        37 hours ago
120MB
hello-world         latest              4ab4c602aa5e        2 months ago
1.84kB
```

4. Para publicar a imagem à qual aplicamos a *tag* para o *registry* global do Docker Hub, use o comando `docker push`:

```
# docker push seg10docker/seg10:pyhello-v1
The push refers to repository [docker.io/seg10docker/seg10]
1efa6f0c4ef3: Pushed
2134161361ab: Pushed
1acdc2a51c84: Pushed
6cffeea81e5d: Mounted from library/python
614a79865f6d: Mounted from library/python
612d27bb923f: Mounted from library/python
ef68f6734aa4: Mounted from library/python
pyhello-v1: digest:
sha256:2879351d8aa37d80e5cebeb676f70af2a93de107d0b801bb51e47d937f99f3ff size: 1787
```

Concluído este processo, a imagem estará publicada e disponível no Docker Hub. De volta ao navegador em sua máquina física, acesse a aba *Tags* em seu repositório para visualizar a imagem que foi enviada:

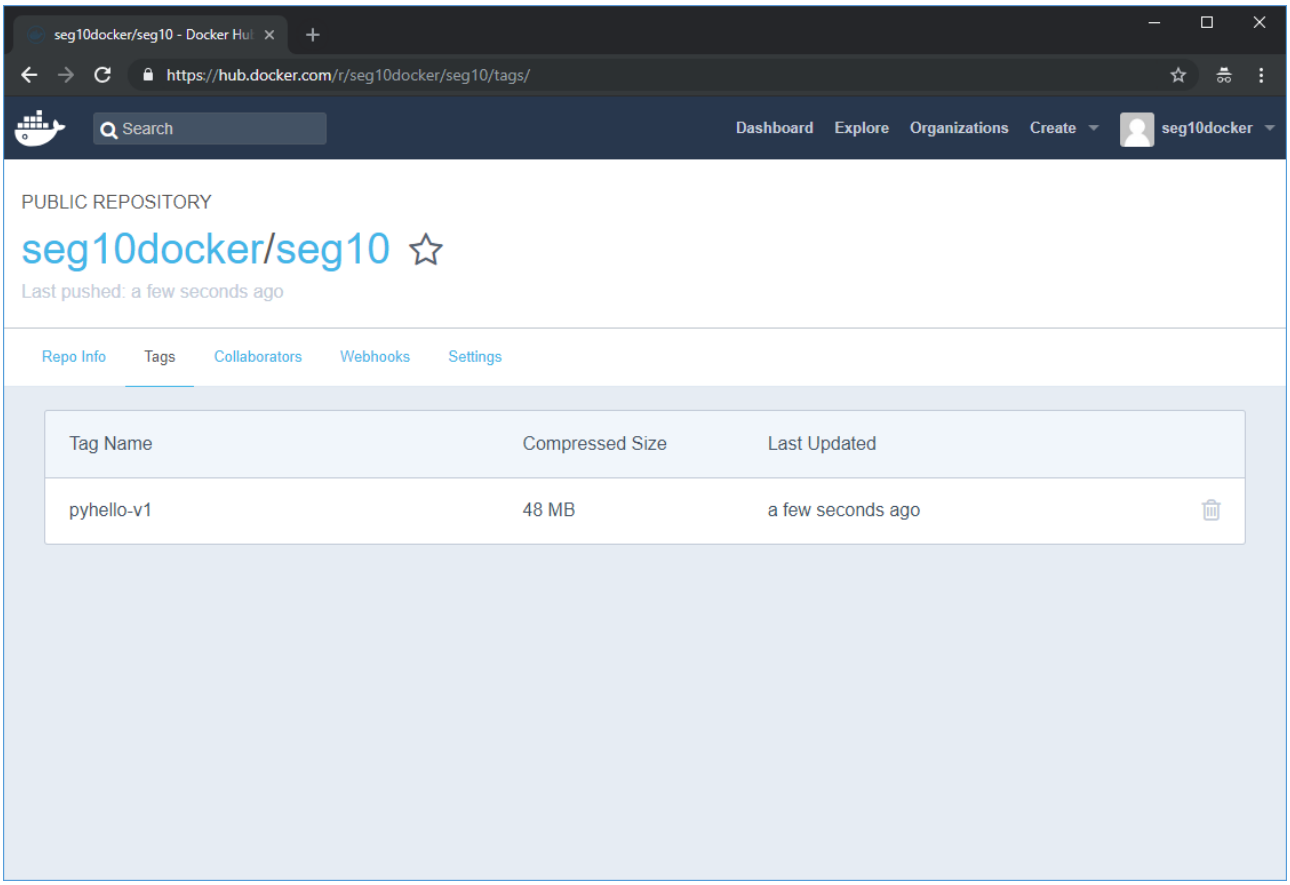

Figura 56. Imagem publicada no Docker Hub

5. A partir deste momento, é possível executar a imagem do container que publicamos para o Docker Hub a partir de qualquer máquina que possua o Docker instalado, diretamente. Por exemplo, acesse a máquina `docker2` como o usuário `root`:

```
# hostname ; whoami
docker2
root
```

Agora, execute a imagem com o comando:

```
# docker run -p 7080:80 seg10docker/seg10:pyhello-v1
Unable to find image 'seg10docker/seg10:pyhello-v1' locally
pyhello-v1: Pulling from seg10docker/seg10
a5a6f2f73cd8: Pull complete
8da2a74f37b1: Pull complete
09b6f498cfd0: Pull complete
f0afb4f0a079: Pull complete
b0ce05758094: Pull complete
dde7e744bb50: Pull complete
0662477f0e17: Pull complete
Digest: sha256:2879351d8aa37d80e5cebeb676f70af2a93de107d0b801bb51e47d937f99f3ff
Status: Downloaded newer image for seg10docker/seg10:pyhello-v1
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:80/ (Press CTRL+C to quit)
```

Observe que o Docker detecta que a imagem não está disponível localmente, e então faz o download da mesma, instala as dependências do Python via `requirements.txt` e executa a aplicação. Tudo é gerenciado de forma transparente, de forma que apenas tivemos que invocar o container que preparamos e enviamos para o *registry* anteriormente.

Antes de prosseguir, encerre o container `CTRL + C`.

6) Construindo serviços com o Docker

1. Agora, vamos escalar a execução do nosso container para um ambiente simulado de produção. Acesse a máquina `docker1` como o usuário `root`:

```
# hostname ; whoami
docker1
root
```

2. Crie o arquivo novo `/root/docker/docker-compose.yml` com o conteúdo abaixo:

```
 1 version: "3"
 2 services:
 3   web:
 4     # substitua username/repo:tag com suas informacoes de nome de usuario,
repositorio e imagem
 5     image: username/repo:tag
 6     deploy:
 7       replicas: 5
 8       resources:
 9         limits:
10           cpus: "0.1"
11           memory: 50M
12       restart_policy:
13         condition: on-failure
14     ports:
15       - "7080:80"
16     networks:
17       - webnet
18 networks:
19   webnet:
```

Como mencionado no comentário acima, não se esqueça de substituir a *string* `username/repo:tag`, que identifica a imagem a ser executada, pelas informações de usuário, repositório e imagem que foram criadas por você na atividade anterior. Por exemplo, no caso do usuário `seg10docker` que foi ilustrado até aqui, a linha ficaria assim:

```
# grep 'image:' ~/docker/docker-compose.yml
    image: seg10docker/seg10:pyhello-v1
```

O arquivo acima irá:

- Baixar a imagem que enviamos para o *registry* global do Docker Hub, se necessário.
- Rodar 5 instâncias de um serviço denominado `web`, com cada instância ocupando no máximo 10% de CPU e 50 MB de memória RAM.
- Reiniciar imediatamente quaisquer containers que venham a falhar.
- Mapear a porta 7080 do *host* Docker para a porta 80 do container.
- Instruir os containers do serviço `web` a compartilhar a porta 80 através de uma rede com balanceador de carga denominada `webnet`.
- Definir a rede `webnet` com opções padrão (no caso, uma rede com balanceador de carga em *overlay*).

3. Vamos testar? Primeiro, temos que iniciar o *swarm*, que consiste em um conjunto de máquinas rodando o Docker que operam em *cluster*. Como, neste momento, apenas a máquina `docker1` integrará esse *cluster*, ela atuará como o administrador do *swarm*.

```
# docker swarm init
Swarm initialized: current node (1iys4vxt3k1pkcasdufb4m5vc) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-
3jpdh6q1i5a9fay1y3nwavb18enoqhujwxk9bgin2k4as1p38z-6ft56u5yq0zxpc7wcsmvgzsux
10.0.42.9:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow
the instructions.
```

Agora sim, vamos iniciar a *stack* do serviço. Iremos nomeá-la `pyhello-stack`:

```
# cd ~/docker ; docker stack deploy -c docker-compose.yml pyhello-stack
Creating network pyhello-stack_webnet
Creating service pyhello-stack_web
```

A partir desse momento, a *stack* do serviço está rodando 5 instâncias de containers da imagem `pyhello-v1`.

4. Verifique se, de fato, todos esses containers estão rodando com `docker service ls`:

```
# docker service ls
ID                  NAME                MODE                REPLICAS
IMAGE                          PORTS
nrglkuxxdyer        pyhello-stack_web   replicated          5/5
seg10docker/seg10:pyhello-v1   *:7080->80/tcp
```

O serviço reporta que há 5 réplicas operando. Para visualizá-las individualmente, use `docker service ps`:

```
# docker service ps pyhello-stack_web
ID                  NAME                  IMAGE                          NODE
DESIRED STATE       CURRENT STATE           ERROR                PORTS
x69lu01rscpt        pyhello-stack_web.1   seg10docker/seg10:pyhello-v1   docker1
Running             Running 2 minutes ago
i0id1ygbu7ba        pyhello-stack_web.2   seg10docker/seg10:pyhello-v1   docker1
Running             Running 2 minutes ago
uiu2pvojpzvw        pyhello-stack_web.3   seg10docker/seg10:pyhello-v1   docker1
Running             Running 2 minutes ago
9tpopx8y3prr        pyhello-stack_web.4   seg10docker/seg10:pyhello-v1   docker1
Running             Running 2 minutes ago
wp539pcus74u        pyhello-stack_web.5   seg10docker/seg10:pyhello-v1   docker1
Running             Running 2 minutes ago
```

Também é possível listar todos os containers operando via `docker container ls`:

```
# docker container ls -q
f12b9b2db505
81c893056bfa
4b0c462de8f5
70b9a762aed2
0e4742201217
```

No navegador em sua máquina física, acessando o IP da interface `enp0s3` da máquina `ns1`, porta 7080, solicite o recarregamento da página diversas vezes com o atalho `F5`. Note como, a cada *refresh*, o ID do container muda no campo *Hostname*.

5. Suponhamos que temos um pico de acessos, e é necessário aumentar de 5 para 8 o número de réplicas de container ativas. O Docker permite que façamos isso sem ter que reiniciar o serviço, veja: primeiro, edite o arquivo `/root/docker/docker-compose.yml`.

```
# sed -i 's/^\([[:space:]]*replicas:\).*/\1 8/' ~/docker/docker-compose.yml
```

```
# grep replicas ~/docker/docker-compose.yml
      replicas: 8
```

Agora, faça o *redeploy* do serviço:

```
# cd ~/docker ; docker stack deploy -c docker-compose.yml pyhello-stack
Updating service pyhello-stack_web (id: nrglkuxxdyer9andac5feb51t)
```

Note que o número de réplicas de containers aumenta imediatamente:

```
# docker service ls
ID                  NAME                MODE                REPLICAS
IMAGE                          PORTS
nrglkuxxdyer        pyhello-stack_web   replicated          8/8
seg10docker/seg10:pyhello-v1   *:7080->80/tcp
```

```
# docker container ls -q
fc5d90deda22
afeb8dc18eea
bb599b864df5
f12b9b2db505
81c893056bfa
4b0c462de8f5
70b9a762aed2
0e4742201217
```

6. Para interromper o serviço, remove a *stack* com o comando `docker stack rm`:

```
# docker stack rm pyhello-stack
Removing service pyhello-stack_web
Removing network pyhello-stack_webnet
```

Depois, abandone o *swarm* usando `docker swarm leave`. Como a máquina `docker1` é um administrador do *swarm*, temos que usar o parâmetro `--force`:

```
# docker swarm leave --force
Node left the swarm.
```

Observe que todos os containers foram encerrados, como esperado.

```
# docker container ls -q | wc -l
0
```

7) Operando com múltiplos membros no cluster

1. Operar com múltiplas máquinas no *cluster* (ao invés de apenas uma, como fizemos com a máquina `docker1` na atividade anterior) é bastante fácil. Primeiro, acesse a máquina `docker1` como `root`:

```
# hostname ; whoami
docker1
root
```

2. Inicie o *swarm* como fizemos anteriormente, com o comando `docker swarm init`:

```
# docker swarm init
Swarm initialized: current node (k0nruds0qup7nscmf0j43jb30) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-
3mw379ioa403z9kpu2rfbnjdzct1o4p33xh83zngsyey0rw9lp-afg4svcx1rjprgvug53vm21v4
10.0.42.9:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow
the instructions.
```

Observe o que fala a segunda linha do *output* acima: para adicionar novas máquinas ao *swarm*, execute o comando abaixo na máquina-alvo. Vamos fazer exatamente isso.

3. Acesse a máquina `docker2` como `root`.

```
# hostname ; whoami
docker2
root
```

Copie o comando `docker swarm join` mostrado no *output* do passo (2), acima, e execute-o na máquina `docker2`:

```
# docker swarm join --token SWMTKN-1-
3mw379ioa403z9kpu2rfbnjdzct1o4p33xh83zngsyey0rw9lp-afg4svcx1rjprgvug53vm21v4
10.0.42.9:2377
This node joined a swarm as a worker.
```

Pronto! As máquinas `docker1` e `docker2` estão agora juntas no *cluster*, sendo a máquina `docker1` o *manager* e a `docker2` o *worker* nesse cenário.

4. Volte à máquina `docker1`, como `root`, e faça o *deploy* do serviço `pyhello-stack`:

```
# hostname ; whoami
docker1
root
```

```
# cd ~/docker ; docker stack deploy -c docker-compose.yml pyhello-stack
Creating network pyhello-stack_webnet
Creating service pyhello-stack_web
```

Verifique quantos containers estão executando na máquina `docker1`:

```
# docker container ls -q | wc -l
4
```

Ué, apenas 4 containers? Onde estão os outros 4? O comando `docker service ps` nos dá uma visão mais ampla da situação:

```
# docker service ps pyhello-stack_web
ID                  NAME                  IMAGE                              NODE
DESIRED STATE       CURRENT STATE            ERROR                PORTS
pmshw6028oin        pyhello-stack_web.1   seg10docker/seg10:pyhello-v1   docker2
Running             Running 42 seconds ago
ilcy0qubdj7v        pyhello-stack_web.2   seg10docker/seg10:pyhello-v1   docker1
Running             Running 44 seconds ago
qm8frg324qjj        pyhello-stack_web.3   seg10docker/seg10:pyhello-v1   docker2
Running             Running 42 seconds ago
vdijie0369g8        pyhello-stack_web.4   seg10docker/seg10:pyhello-v1   docker1
Running             Running 43 seconds ago
nqt5l1sm678e        pyhello-stack_web.5   seg10docker/seg10:pyhello-v1   docker2
Running             Running 43 seconds ago
8czx47nhep2n        pyhello-stack_web.6   seg10docker/seg10:pyhello-v1   docker1
Running             Running 44 seconds ago
ryrf4ovcq7d6        pyhello-stack_web.7   seg10docker/seg10:pyhello-v1   docker2
Running             Running 42 seconds ago
4fr3zlwdgqjg        pyhello-stack_web.8   seg10docker/seg10:pyhello-v1   docker1
Running             Running 44 seconds ago
```

Veja que temos 4 containers rodando na máquina `docker1`, e outros 4 rodando na máquina `docker2`.

5. Agora, pode surgir uma questão em sua mente: "Ora, se as configurações que fizemos no firewall instruem o repasse de pacotes na porta 7080/TCP diretamente para a máquina `docker1`, então é evidente que os 4 containers rodando na máquina `docker2` estão inacessíveis, pelo menos até que corrijamos as regras de firewall, certo?"

   Será mesmo? Na máquina `docker2`, como `root`, liste os containers em operação:

```
# hostname ; whoami
docker2
root
```

```
# docker container ls
CONTAINER ID        IMAGE                          COMMAND             CREATED
STATUS              PORTS
              NAMES
37bbca732ec7        seg10docker/seg10:pyhello-v1   "python app.py"     6 minutes
ago       Up 6 minutes        80/tcp
              pyhello-stack_web.7.ryrf4ovcq7d6cvrxyoyh372kk
ab9171ebcf69        seg10docker/seg10:pyhello-v1   "python app.py"     6 minutes
ago       Up 6 minutes        80/tcp
              pyhello-stack_web.5.nqt5l1sm678e96ctsv2kwuk9f
f0d973c6c635        seg10docker/seg10:pyhello-v1   "python app.py"     6 minutes
ago       Up 6 minutes        80/tcp
              pyhello-stack_web.1.pmshw6028oinz61bh509suza3
c420565a43f1        seg10docker/seg10:pyhello-v1   "python app.py"     6 minutes
ago       Up 6 minutes        80/tcp
              pyhello-stack_web.3.qm8frg324qjj5roggzfwwznbn
```

Agora, em seu navegador na máquina física, recarregue a página algumas vezes e observe os IDs de container que são mostrados:

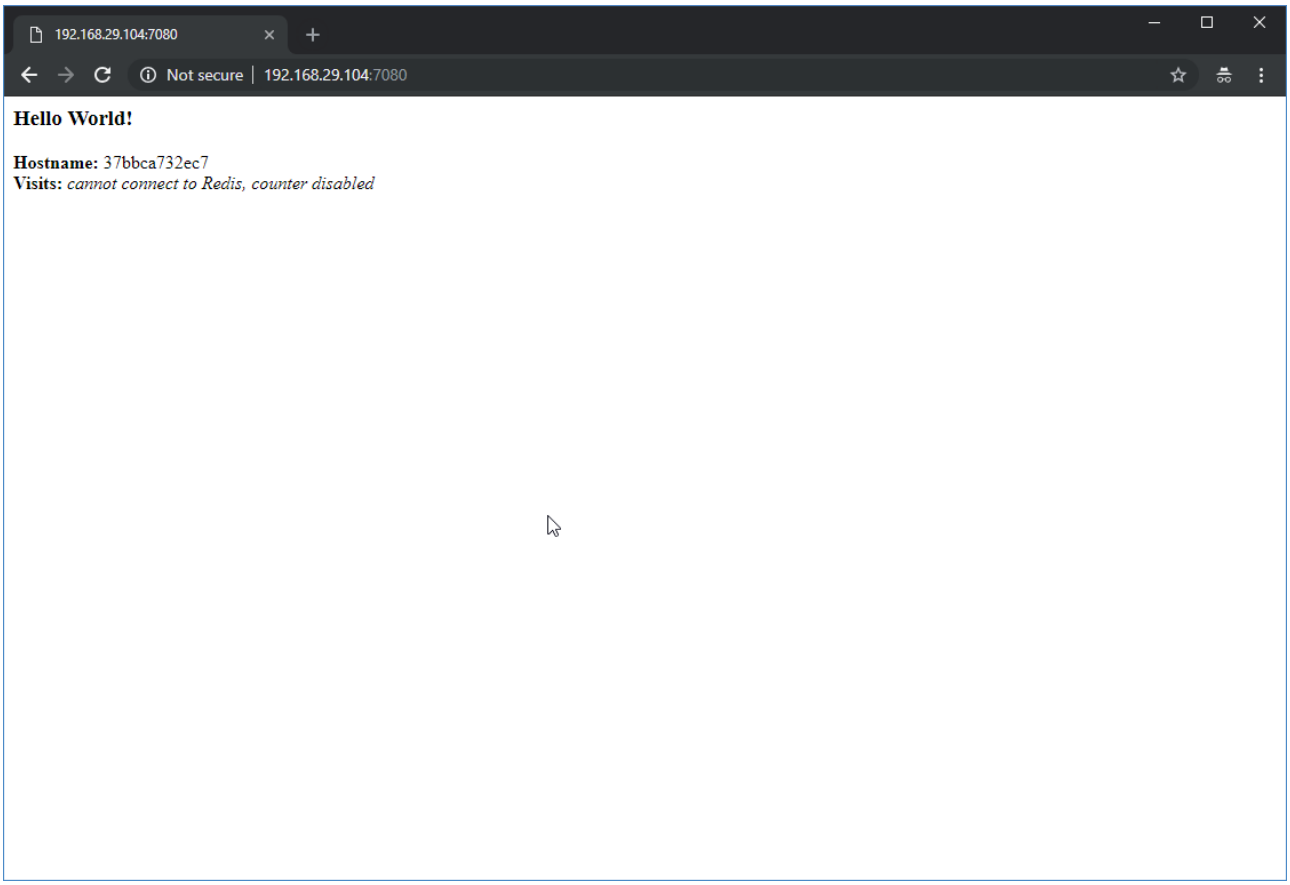

Figura 57. Container da máquina docker2 sendo acessado

Note que o ID de container mostrado acima está executando na máquina `docker2`, mas claramente é impossível que tenhamos acessado essa máquina, já que as regras de firewall criadas anteriormente redirecionam pacotes **apenas** para a máquina `docker1`. Como isso está acontecendo?

A razão pela qual as duas máquinas estão acessíveis é porque um nodo do *swarm* Docker participa de um roteamento ingresso do tipo *mesh*, como ilustrado pela figura a seguir. Esse roteamento garante que um serviço alocado a uma porta em seu *swarm* sempre terá essa porta reservada para si, independente de qual nodo esteja rodando o container. Note, no exemplo, que é perfeitamente possível que uma requisição chegue à máquina `docker1` mas seja atendida por um container em `docker2`, ou vice-versa.

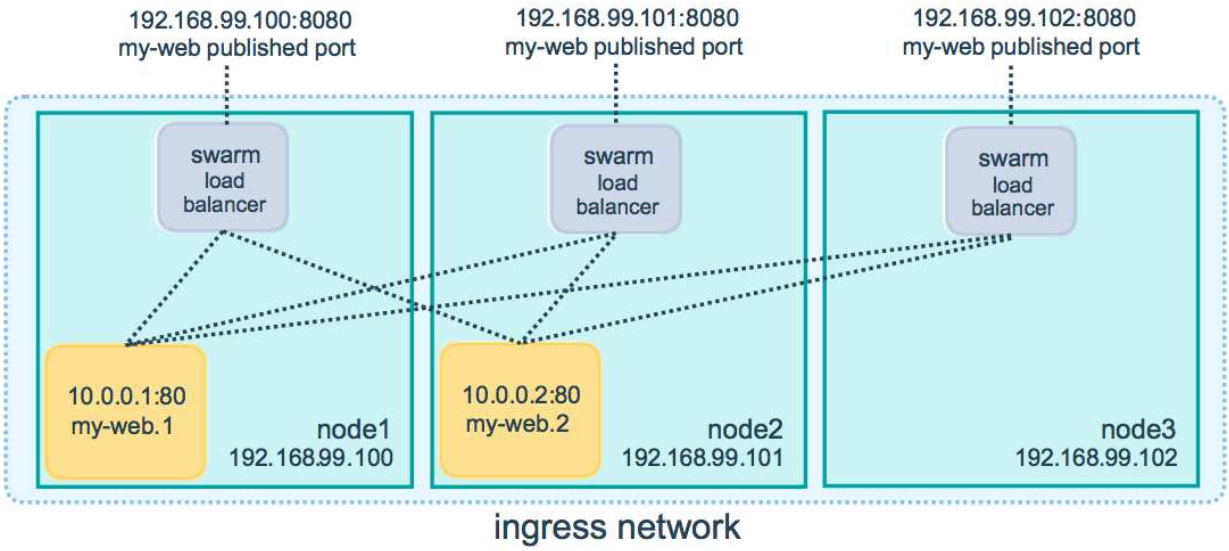

Figura 58. Roteamento do tipo mesh no Docker

6. Mesmo com a funcionalidade acima, é interessante que o firewall envie requisições para ambas as máquinas `docker1` e `docker2`, indistintamente. Volte à máquina `ns1`, como `root`.

```
# hostname ; whoami
ns1
root
```

Apague as regras que havíamos criado anteriormente.

```
# iptables -t nat -D PREROUTING -i enp0s3 -p tcp -m tcp --dport 7080 -j DNAT --to
-destination 10.0.42.9
```

```
# iptables -D FORWARD -i enp0s3 -d 10.0.42.9/32 -p tcp -m tcp --dport 7080 -j
ACCEPT
```

Em seu lugar, insira regras que redirecionem o tráfego para ambas as máquinas, em modalidade *round robin*:

```
# iptables -t nat -A PREROUTING -i enp0s3 -p tcp -m tcp --dport 7080 -j DNAT --to
-destination 10.0.42.9-10.0.42.10
```

```
# iptables -A FORWARD -i enp0s3 -d 10.0.42.9/32,10.0.42.10/32 -p tcp -m tcp --dport
7080 -j ACCEPT
```

Em seu navegador na máquina física, verifique que o serviço continua ativo, recarregando a página algumas vezes.

8) Adicionando novos serviços ao cluster

1. É bastante fácil adicionar novos serviços a um *cluster* Docker, mesmo quando em operação. Acesse a máquina `docker1` como `root`:

```
# hostname ; whoami
docker1
root
```

2. Vamos adicionar um serviço *visualizer*, um container pré-pronto do Docker que permite que observemos como está a alocação de containers em nosso *cluster*. Edite o arquivo de configuração do *swarm*, `/root/docker/docker-compose.yml`, com o comando `sed` a seguir:

```
# sed -i '/^networks:$/i\
  visualizer:\
    image: dockersamples/visualizer:stable\
    ports:\
      - "8080:8080"\
    volumes:\
      - "/var/run/docker.sock:/var/run/docker.sock"\
    deploy:\
      placement:\
        constraints: [node.role == manager]\
    networks:\
      - webnet' ~/docker/docker-compose.yml
```

O comando acima irá:

- Adicionar um novo serviço, `visualizer`, usando a imagem de container `dockersamples/visualizer:stable` baixada do *registry* global do Docker Hub.
- Mapear a porta externa 8080 para a porta interna 8080, dentro do contexto do container.
- Criar um mapeamento de volume do *socket* `/var/run/docker.sock` na máquina *host* para o mesmo caminho dentro do container. Vale observar que este *socket* está disponível exclusivamente no nodo *manager* do *cluster*.
- Por esse motivo, cria-se uma restrição de alocação desse container, que deve rodar exclusivamente em nodos que possuam a *role* de *manager* no *cluster*.
- Finalmente, conecta-se o serviço à mesma rede `webnet` que havíamos criado antes.

3. Para lançar o novo serviço, basta executar um *redeploy* da *stack*:

```
# cd ~/docker ; docker stack deploy -c docker-compose.yml pyhello-stack
Creating service pyhello-stack_visualizer
Updating service pyhello-stack_web (id: fld46go1hn34kirew3xn3019v)
```

   O Docker detecta que a imagem `dockersamples/visualizer:stable` está indisponível localmente, e faz o download da mesma do *registry* global, lançando-a em seguida.

4. Presumindo que o serviço está ativo, note que estamos fazendo um novo mapeamento de portas: da 8080 (externa) para a 8080 (interna). Temos, naturalmente, que fazer ajustes em nosso firewall de rede. Acesse a máquina `ns1` como `root`:

```
# hostname ; whoami
ns1
root
```

   Apague as regras recém-criadas:

```
# iptables -t nat -D PREROUTING -i enp0s3 -p tcp -m tcp --dport 7080 -j DNAT --to
-destination 10.0.42.9-10.0.42.10
```

```
# iptables -D FORWARD -i enp0s3 -d 10.0.42.9/32,10.0.42.10/32 -p tcp -m tcp --dport
7080 -j ACCEPT
```

   Torne-as mais abrangentes, incluindo também a porta 8080:

```
# iptables -t nat -A PREROUTING -i enp0s3 -p tcp -m multiport --dports 7080,8080 -j
DNAT --to-destination 10.0.42.9-10.0.42.10
```

```
# iptables -A FORWARD -i enp0s3 -d 10.0.42.9/32,10.0.42.10/32 -p tcp -m multiport
--dports 7080,8080 -j ACCEPT
```

   Agora sim, sendo estas regras definitivas, grave-as na configuração do firewall local:

```
# /etc/init.d/netfilter-persistent save
[....] Saving netfilter rules...run-parts: executing /usr/share/netfilter-
persistent/plugins.d/15-ip4tables save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
done.
```

5. Em sua máquina física, aponte agora o navegador para o IP público do *datacenter*, o endereço

do interface `enp0s3` da máquina `ns1` na porta 8080:

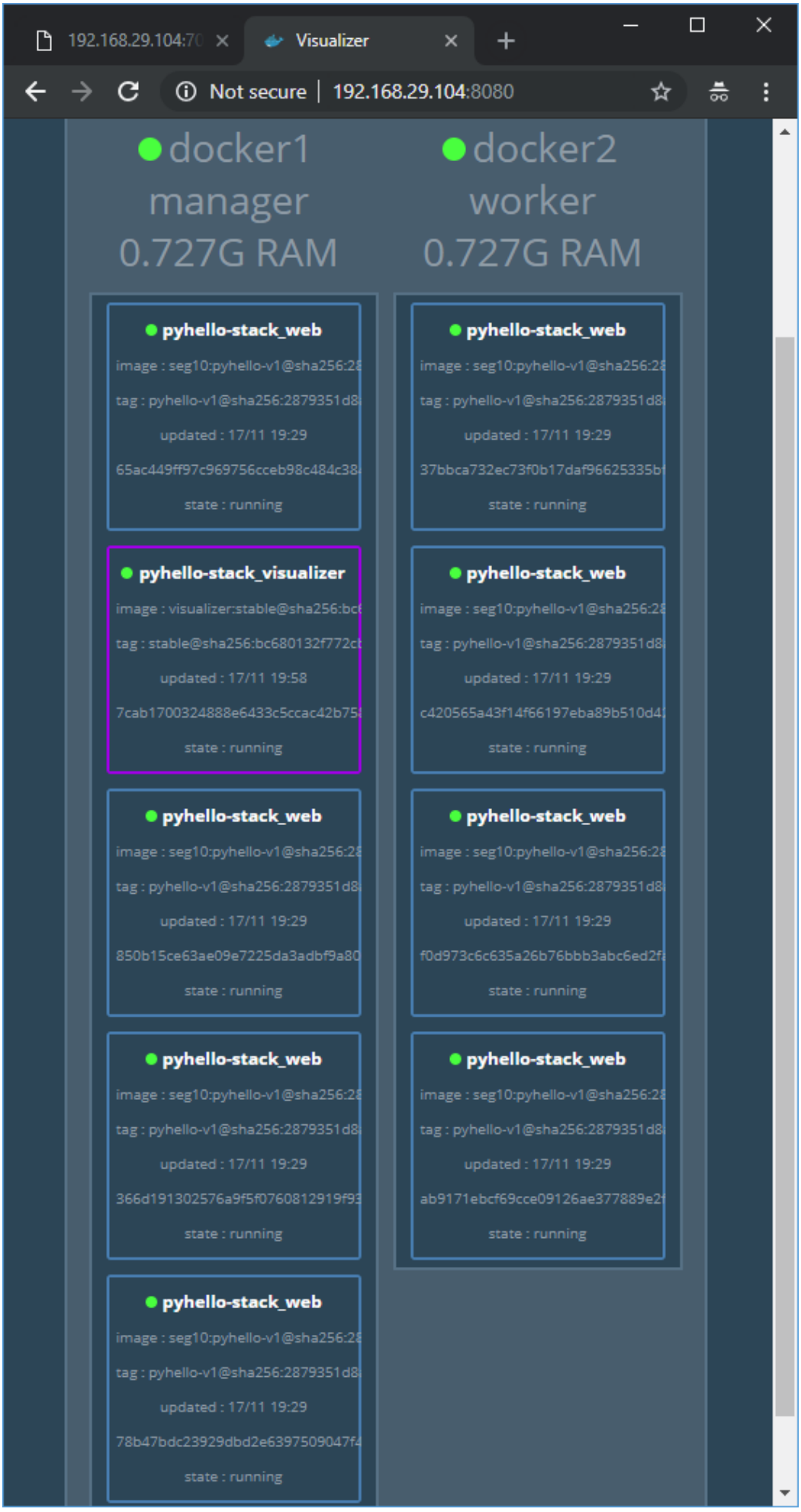

Figura 59. Visualizando a operação do cluster Docker

O container `visualizer` nos mostra, de forma fácil, como está o estado do *cluster* Docker. Note como o container `visualizer` está operando apenas na máquina `docker1`, já que ela é o *manager* do *swarm*. Os containers `web`, por outro lado, podem operar em qualquer nodo, estando distribuídos 4 a cada lado.

9) Configurando a persistência dos dados

1. Você deve ter notado que o contador de visitantes do website, por algum motivo, não está funcionando. Isso se deve ao fato que o serviço do Redis, que fornece uma espécie de banco de dados em arquivo, não está operacional. Vamos corrigir isso: acesse a máquina `docker1` como `root`.

```
# hostname ; whoami
docker1
root
```

2. Para adicionar um serviço para o Redis, edite o arquivo de configuração `/root/docker/docker-compose.yml` com o comando `sed` a seguir:

```
# sed -i '/^networks:$/i\
  redis:\
    image: redis\
    ports:\
      - "6379:6379"\
    volumes:\
      - "/root/data:/data"\
    deploy:\
      placement:\
        constraints: [node.role == manager]\
    command: redis-server --appendonly yes\
    networks:\
      - webnet' ~/docker/docker-compose.yml
```

O comando acima irá:

- Adicionar um novo serviço, `redis`, usando a imagem de container `redis` baixada do *registry* global do Docker Hub.
- Mapear a porta externa 6379 para a porta interna 6379, dentro do contexto do container. Como esse serviço não será acessado externamente, não será necessário criar novas regras no firewall `ns1`.
- Criar um mapeamento de volume do diretório `/root/data` na máquina *host* para o caminho `/data` dentro do container. A persistência de dados de vistantes do website será armazenada nesse diretório.
- Cria-se uma restrição de alocação desse container, que deve rodar exclusivamente em nodos que possuam a *role* de *manager* no *cluster*. Assim, todos os containers do *cluster* terão a

mesma visão dos dados em persistência.

- Finalmente, conecta-se o serviço à mesma rede `webnet` que havíamos criado antes.

Naturalmente, temos que criar o diretório `/root/data`, que ainda não existe:

```
# mkdir ~/data
```

3. Faça o *redeploy* da *stack*:

```
# cd ~/docker ; docker stack deploy -c docker-compose.yml pyhello-stack
Updating service pyhello-stack_visualizer (id: kzzk52hi201lz8jlsoif86p45)
Creating service pyhello-stack_redis
Updating service pyhello-stack_web (id: fld46go1hn34kirew3xn3019v)
```

4. Em sua máquina física, aponte o navegador para o IP público do *datacenter*, o endereço do interface `enp0s3` da máquina `ns1` na porta 7080. Note que o contador de visitas está, agora sim, sendo contabilizado corretamente:

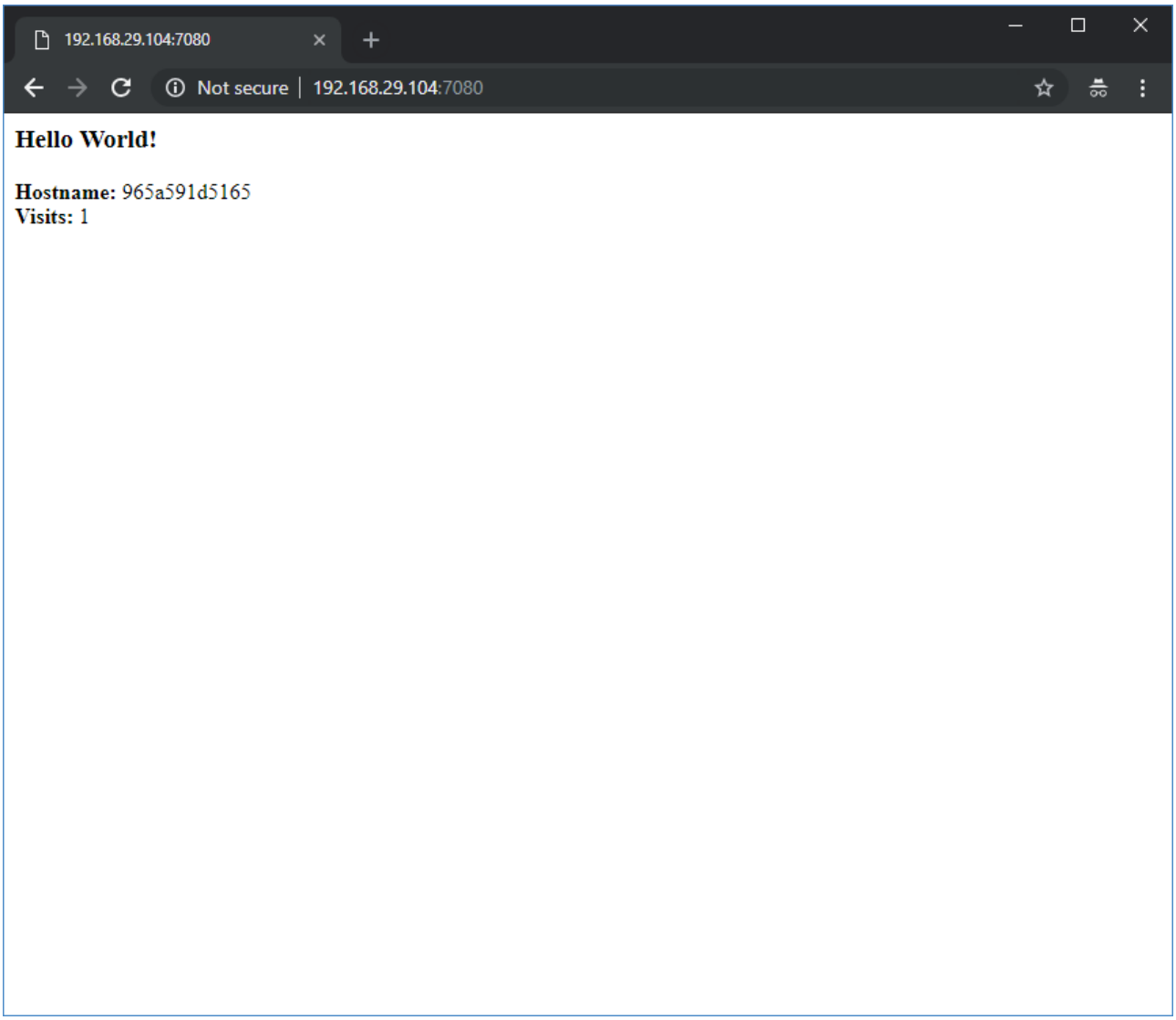

Figura 60. Contador de visitas operacional

Recarregue a página algumas vezes, e observe que o contador continua aumentando independentemente do fato de estarmos sendo atendidos por um container localizado na máquina `docker1` ou na máquina `docker2`. Como todos os containers tem a mesma visão da "verdade", isto é, a base de dados compartilhada no diretório `/root/data` da máquina `docker1`, conseguimos obter o valor correto e incrementá-lo sem importar de onde está partindo a requisição.

Observando o `visualizer`, que opera na porta 8080, note que o container do Redis executa exclusivamente na máquina `docker1`, como configurado:

Figura 61. Máquina docker1 executando visualizador e Redis

5. Encerradas as nossas atividades com containers, encerre o *stack* e *swarm* em ambas as máquinas `docker1` e `docker2`, e desligue-as. Para manter reduzido o uso de recursos durante o decorrer das próximas sessões, é interessante que tenhamos o mínimo de VMs operacionais.

Sessão 9: Criação de sistemas Linux customizados

Nesta sessão iremos aprender como criar uma distribuição Linux sob medida usando o *live-build*, um sistema de construção de distribuições Debian sob medida.

1) Topologia desta sessão

Criaremos apenas uma nova máquina nesta sessão, a saber:

- `live`, sistema para construção de imagens Linux customizadas usando o sistema *live-build*. Endereço IP 10.0.42.11/24.

1. Como de costume, vamos à criação dos registros DNS. Acesse a máquina `ns1` como o usuário `root`:

```
# hostname ; whoami
ns1
root
```

Edite o arquivo de zonas `/etc/nsd/zones/intnet.zone`, inserindo entradas A para a máquinas indicadas no começo desta atividade. **Não se esqueça** de incrementar o valor do serial no topo do arquivo!

```
# nano /etc/nsd/zones/intnet.zone
(...)
```

```
# grep live /etc/nsd/zones/intnet.zone
live    IN    A                10.0.42.11
```

Faça o mesmo para o arquivo de zona reversa:

```
# nano /etc/nsd/zones/10.0.42.zone
```

```
# grep live /etc/nsd/zones/10.0.42.zone
11      IN   PTR               live.intnet.
```

Assine o arquivo de zonas usando o *script* criado anteriormente:

```
# bash /root/scripts/signzone-intnet.sh
reconfig start, read /etc/nsd/nsd.conf
ok
ok
ok
ok removed 6 rrsets, 2 messages and 0 key entries
```

Verifique a criação das entradas usando o comando `dig`:

```
# dig live.intnet +short
10.0.42.11
```

```
# dig -x 10.0.42.11 +short
live.intnet.
```

2) Criação da VM de build

1. Vamos criar a VM `live` e utilizá-la para criar imagens Linux customizadas. Clone a máquina `debian-template` para uma de nome `live`, com uma única interface de rede conectada à DMZ. O IP da máquina será 10.0.42.11/24.

   Concluída a clonagem, na console principal do Virtualbox, acesse o menu *Settings*. Em *System* > *Motherboard* > *Base Memory*, aloque ao menos 4 GB de RAM para essa VM. Adicionalmente, vá para *System* > *Processor* e aumente o número de processadores disponíveis para a máquina, tanto quanto possível: se você tiver à disposição uma máquina *quad-core*, por exemplo, aloque dois processadores; caso sejam oito *cores*, aloque quatro CPUs, e assim sucessivamente.

   Essas alterações são necessárias pois o processo de *build* de uma nova distribuição é bastante intensivo computacionalmente, e quanto mais processamento tivermos à disposição, mais cedo concluiremos os passos.

   Em *Settings* > *Storage* > *Controller: SATA*, adicione um novo disco à VM. Escolha o formato VDI, alocação dinâmica de espaço, nome da unidade `live-build` e 20 GB de tamanho. Iremos usar este espaço para fazer o download e instalação dos pacotes das distribuições customizadas em um *chroot* dedicado.

   Finalmente, clique em *OK* e ligue a VM. Logue como `root` e use o script `/root/scripts/changehost.sh` para fazer a configuração automática, como de costume.

```
# hostname ; whoami
debian-template
root
```

```
# bash ~/scripts/changehost.sh -h live -i 10.0.42.11 -g 10.0.42.1
Signing ssh_host_ecdsa_key.pub key...
Signing ssh_host_ed25519_key.pub key...
Signing ssh_host_rsa_key.pub key...
Configuring host key trust...
Configuring user key trust...
All done!
```

2. Aplique o *baseline* de segurança à máquina `live`, repetindo o que fizemos no passo (2), atividade (2) da sessão 7:

```
$ hostname ; whoami
client
ansible
```

```
$ sed -i '/\[srv\]/a live' ~/ansible/hosts
```

```
$ ansible-playbook -i ~/ansible/hosts -l live -Ke ansible_become_method=su
~/ansible/srv.yml ; ansible-playbook -i ~/ansible/hosts -l live ~/ansible/srv.yml
SUDO password:

(...)

PLAY RECAP
*********************************************************************************
***********************

live                       : ok=10   changed=8    unreachable=0    failed=0
```

3) Construindo uma distribuição mínima

1. Acesse a máquina `live` como o usuário `root`:

```
# hostname ; whoami
docker1
root
```

2. Vamos preparar o disco para uso. Descubra sob qual nome ele foi detectado:

```
# dmesg | grep 'GiB'
[    1.507032] sd 2:0:0:0: [sda] 16777216 512-byte logical blocks: (8.59 GB/8.00
GiB)
[    1.507324] sd 3:0:0:0: [sdb] 41943040 512-byte logical blocks: (21.5 GB/20.0
GiB)
```

Perfeito, o nome do disco é `/dev/sdb`. Use o `fdisk` para formatá-lo — crie uma única partição do tipo LVM ocupando a totalidade do espaço.

```
# fdisk /dev/sdb

Bem-vindo ao fdisk (util-linux 2.29.2).
As alterações permanecerão apenas na memória, até que você decida gravá-las.
Tenha cuidado antes de usar o comando de gravação.

A unidade não contém uma tabela de partição conhecida.
Criado um novo rótulo de disco DOS com o identificador de disco 0x19d210eb.
```

```
Comando (m para ajuda): o
Criado um novo rótulo de disco DOS com o identificador de disco 0xdaead0ba.
```

```
Comando (m para ajuda): n
Tipo da partição
   p   primária (0 primárias, 0 estendidas, 4 livre)
   e   estendida (recipiente para partições lógicas)
Selecione (padrão p): p
Número da partição (1-4, padrão 1): 1
Primeiro setor (2048-41943039, padrão 2048):
Último setor, +setores ou +tamanho{K,M,G,T,P} (2048-41943039, padrão 41943039):

Criada uma nova partição 1 do tipo "Linux" e de tamanho 20 GiB.
```

```
Comando (m para ajuda): t
Selecionou a partição 1
Tipo de partição (digite L para listar todos os tipos): 8e
O tipo da partição "Linux" foi alterado para "Linux LVM".
```

```
Comando (m para ajuda): w
A tabela de partição foi alterada.
Chamando ioctl() para reler tabela de partição.
Sincronizando discos.
```

Inicialize o disco para o sistema LVM:

```
# pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created.
```

Crie novos VG/LV para o disco:

```
# vgcreate vg-live /dev/sdb1
  Volume group "vg-live" successfully created
```

```
# lvcreate -l +100%FREE -n lv-live vg-live
  Logical volume "lv-live" created.
```

Formate-o sob o sistema de arquivos `ext4`:

```
# mkfs.ext4 /dev/mapper/vg--live-lv--live
mke2fs 1.43.4 (31-Jan-2017)
Creating filesystem with 5241856 4k blocks and 1310720 inodes
Filesystem UUID: 0716d2c6-cd6a-43a0-ab04-83075886e790
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

Configure a montagem automática via `/etc/fstab` no diretório `/live`, e monte o disco:

```
# mkdir /live
```

```
# echo '/dev/mapper/vg--live-lv--live /live ext4 defaults 0 2' >> /etc/fstab
```

```
# mount -a
```

```
# df -h | grep live
/dev/mapper/vg--live-lv--live   20G   45M   19G   1% /live
```

3. Instale a ferramenta que iremos utilizar para construir imagens customizadas, o *live-build*:

```
# apt-get install -y live-build
```

4. Vamos fazer uma imagem mínima para experimentar com o sistema do *live-build*. Crie o diretório novo `/live/basic`, e entre nele:

```
# mkdir /live/basic ; cd /live/basic
```

O comando `lb config` irá criar uma estrutura de diretórios padrão para o *live-build* operar:

```
# lb config
```

Antes de disparar o *build*, vamos apenas customizar o conjunto de repositórios usados para buscar os pacotes de instalação para uma opção mais veloz:

```
# sed -i 's|ftp.debian.org|ftp.br.debian.org|g' /live/basic/config/bootstrap
```

```
# sed -i 's|ftp.debian.org|ftp.br.debian.org|g' /live/basic/config/build
```

Desabilite também a instalação de *firmwares* diversos para o kernel Linux — como iremos rodar a distribuição exclusivamente em um ambiente virtualizado, não há porque atrasar o *build* e aumentar o tamanho da imagens com *drivers* que não utilizaremos no sistema finalizado.

```
# sed -i 's|^\(LB_FIRMWARE_[A-Z]*=\).*|\1\"false\"|' /live/basic/config/binary
```

Finalmente, inicie o *build* do sistema customizado, e aguarde sua construção. Esse passo pode ser relativamente demorado, então tenha paciência.

```
# lb build
[2018-11-17 23:51:27] lb build
P: live-build 1:20170213
P: Building config tree for a debian/stretch/amd64 system

(...)

Reading package lists... Done
Building dependency tree
Reading state information... Done
[2018-11-17 23:57:01] lb source
```

Para ilustrar o tempo médio a esperar, note que o *build* acima levou cerca de seis minutos para concluir. Esse tempo pode variar para mais, ou para menos, dependendo da velocidade do processador, memória e disco da máquina física, bem como o volume de recursos alocados à

VM `live`.

5. O que esse comando produziu? Vejamos:

```
# ls *.iso
live-image-amd64.hybrid.iso
```

```
# du -sm live-image-amd64.hybrid.iso
216     live-image-amd64.hybrid.iso
```

Foi criada uma imagem ISO de 216 MB, com um sistema *live* perfeitamente bootável, que testaremos a seguir.

6. Copie a imagem ISO produzida no passo (4) acima para sua máquina física. Há vários métodos para se atingir esse objetivo — um dos meus favoritos é usar o comando `scp` no Cygwin, que permite cópia direta da VM para a máquina física, sem a necessidade de instalação de qualquer software adicional. Caso não tenha o Cygwin instalado, considere usar o programa WinSCP, uma ferramenta gráfica que provê funcionalidade semelhante.

   No exemplo abaixo, copiaremos via `scp` + Cygwin a ISO para a Área de Trabalho do usuário `fbs`, na máquina física Windows:

```
$ scp aluno@10.0.42.11:/live/basic/live-image-amd64.hybrid.iso
/cygdrive/c/Users/fbs/Desktop/
aluno@10.0.42.11's password:
live-image-amd64.hybrid.iso
100%  216MB  59.5MB/s   00:03
```

7. Na console principal do Virtualbox, crie uma nova máquina virtual. Como nome, define `iso-test`, *Type* `Linux` e *Version* `Debian (64-bit)`. Mantenha o valor padrão de memória RAM, 1024 GB. Quando perguntado sobre o disco rígida da VM, escolha a opção *Do not add a virtual hard disk* e clique em *Create*, confirmando sua escolha.

   Selecione a nova VM e acesse o menu *Settings* > *Storage* > *Controller: IDE*. Selecione o *drive* de CD vazio e, em *Attributes*, escolha o arquivo de disco ótico virtual que copiamos no passo anterior, a imagem `live-image-amd64.hybrid.iso`.

   Clique em *OK* e ligue a máquina virtual. Após a tela de BIOS do Virtualbox, você verá o *bootloader* do sistema customizado que construímos, como mostrado abaixo:

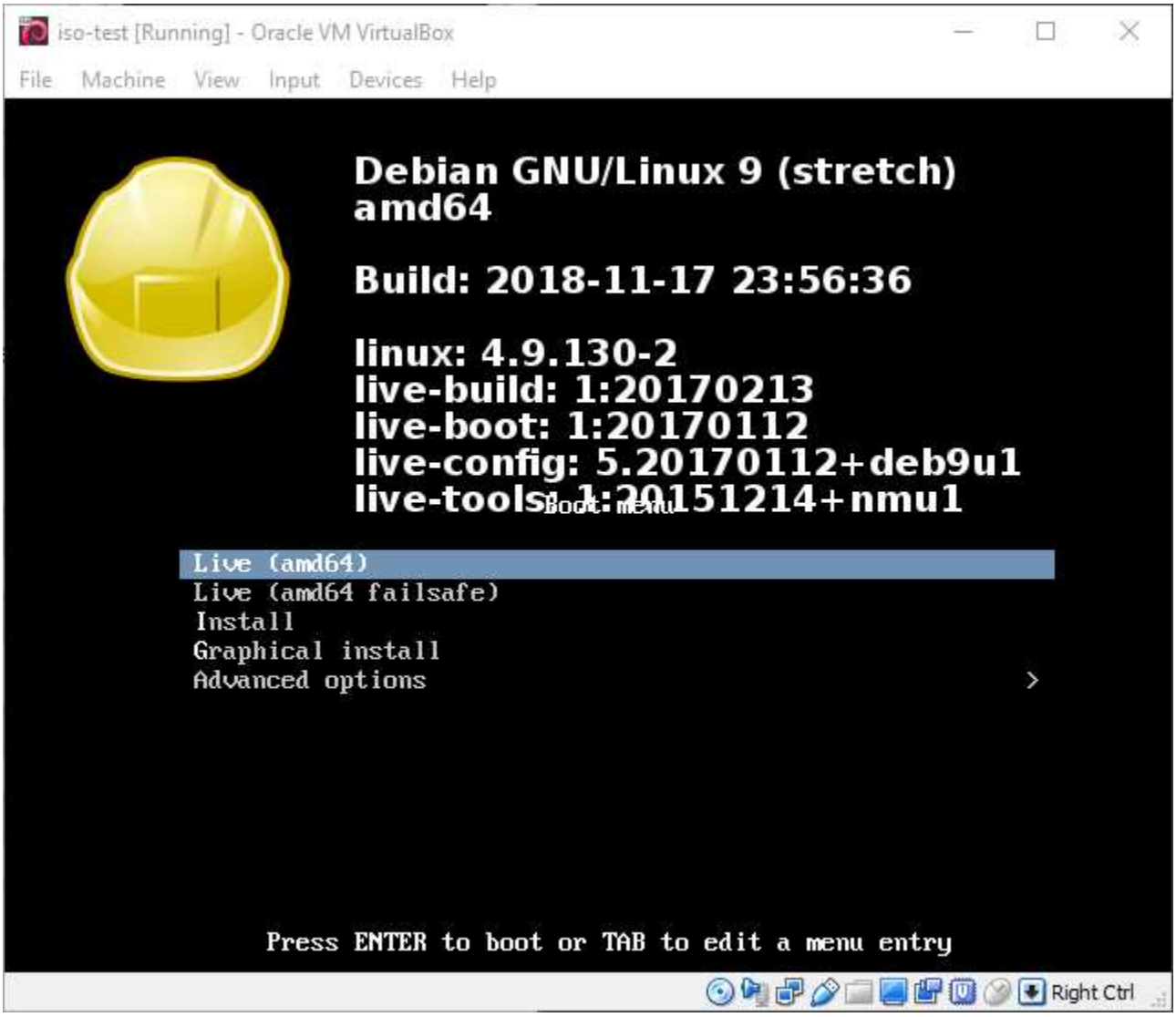

Figura 62. Bootloader do sistema live

Selecione a primeira opção, e prossiga com o *boot*. Brevemente surgirá um *shell* para interação com o sistema — estamos logados com o usuário `user`:

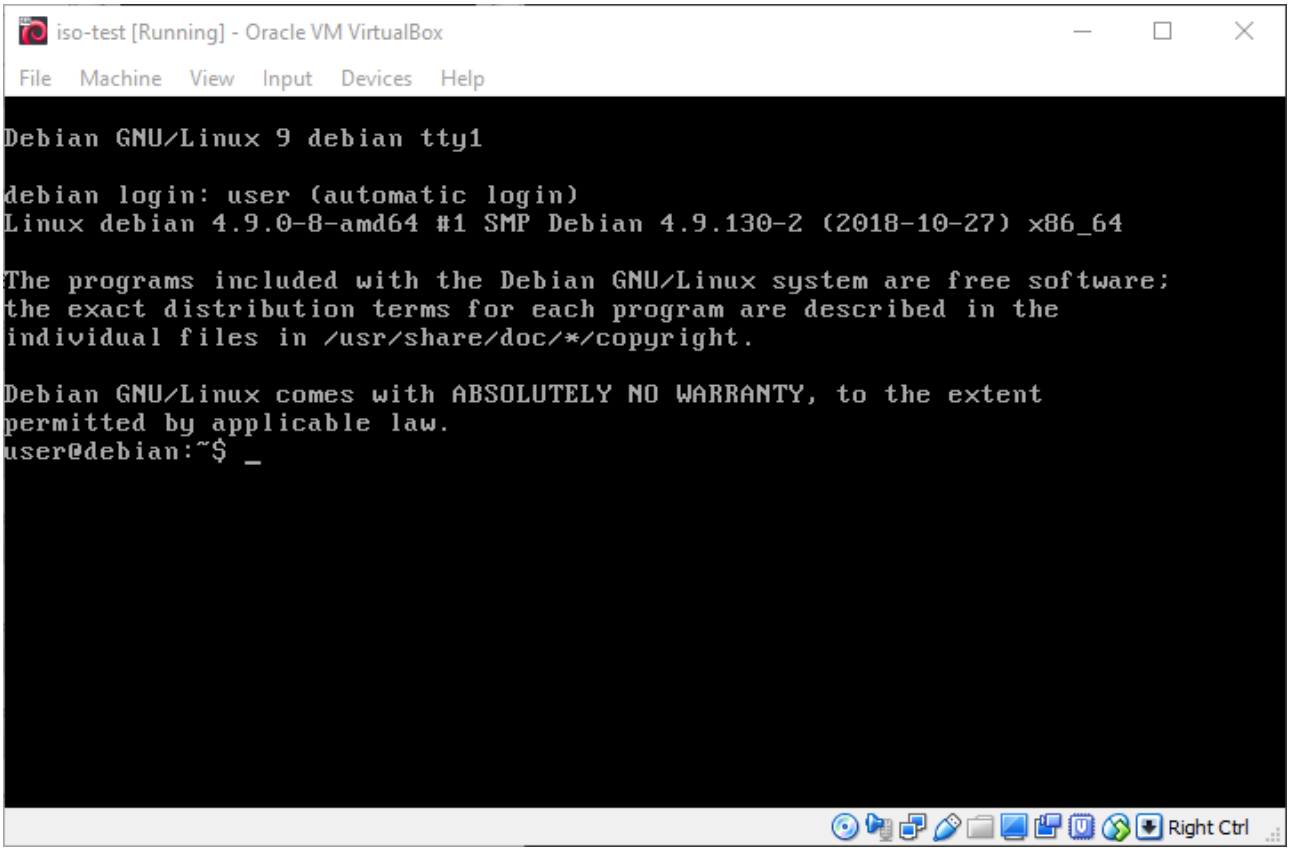

Figura 63. Shell do sistema live

Temos em mãos um sistema plenamente funcional, operando em memória. Para escalar privilégio para `root`, execute `sudo -i`. Para colocar em perspectiva o quão enxuto é o sistema que estamos trabalhando, observe a memória utilizada:

```
# free -m
              total        used        free      shared  buff/cache   available
Mem:            996          26         872           7          97         847
Swap:             0           0           0
```

Apenas 26 MB de RAM! Conte o número de pacotes instalados:

```
# dpkg -l | grep '^ii' | wc -l
198
```

Compare com a máquina `live`, por exemplo, que é derivada de um sistema bastante minimalista que construímos durante a primeira sessão deste curso: ela possui 360 pacotes instalados.

```
# hostname ; dpkg -l | grep '^ii' | wc -l
live
360
```

8. Como o sistema customizado que fizemos está perfeitamente funcional, vamos instalar o cliente OpenSSH:

```
# apt-get install -y openssh-client
```

O APT consegue instalar pacotes mesmo operando em memória, como podemos ver. Produza uma lista dos pacotes instalados e copie-a para a máquina `live`:

```
# dpkg -l | grep '^ii' > /tmp/basic-packages.txt
```

```
# scp /tmp/basic-packages.txt aluno@10.0.42.11:~
aluno@10.0.42.11's password:
basic-packages.txt                                100%   24KB  20.4MB/s   00:00
```

9. Apesar de interessante, nosso sistema customizado ainda não faz muita coisa. Desligue-o, e vamos tentar incrementá-lo.

```
# halt -p
```

4) Utilizando um repositório local de pacotes

1. Volte à máquina `live`, como o usuário `root`. Em seguida, verifique o tamanho do diretório `/live/basic`.

```
# hostname ; whoami
live
root
```

```
# du -sh /live/basic/
1,6G    /live/basic/
```

Note que mesmo para produzir um sistema tão básico quando o que fizemos na atividade (3), o sistema de *build* ocupou 1,6 **gigabytes** de espaço em disco! Além da necessidade de criar um sistema de *bootstrap* e um *chroot* que contém o sistema customizado, é também necessário armazenar todos os pacotes `.deb` que foram baixados durante o *build*.

2. Para limpar os arquivos de trabalho do *live-build*, utilize o comando `lb clean`. Para remover a totalidade dos arquivos gerados pelo *live-build*, como diretórios de *cache*, *chroot*, binários e fontes, use a opção `--purge`.

```
# cd /live/basic ; lb clean --purge
```

Volte a verificar o tamanho da pasta `/live/basic`:

```
# du -sh /live/basic/
208K    /live/basic/
```

3. Se quisermos produzir um novo sistema, como fizemos antes, basta rodar o comando `lb build` novamente. Note, porém, que todos os pacotes `.deb` de instalação do *bootstrap* e sistema-base serão baixados novamente — como a construção de sistemas customizados normalmente envolve uma boa quantidade de tentativa e erro, a quantidade de dados a serem baixados a cada *build* irá rapidamente se tornar a principal fonte de atraso no processo.

   Para suplantar esse problema, podemos construir um repositório local de pacotes, usando por exemplo a ferramenta Aptly. Ao contrário de um espelho total do repositório de pacotes do Debian, que é gigantesco (estimado em 326 GB quando da escrita desta atividade, ref. https://www.debian.org/mirror/size), o Aptly permite que façamos um repositório local bastante enxuto, contendo apenas os pacotes necessários à instalação do sistema-base e suas dependências.

   Copie o diretório `/live/basic` para `/live/aptly-basic`, e entre dentro do novo diretório.

```
# cp -a /live/basic /live/aptly-basic ; cd /live/aptly-basic
```

   Agora, crie o arquivo novo `/live/aptly-basic/makebuild.sh`, com o seguinte conteúdo:

```
 1 #!/bin/bash
 2
 3 ABS_PATH=`readlink -f $0 | sed 's/\/[^\/]*$//'`
 4
 5 if uname -r | egrep '686-pae$' &> /dev/null; then
 6   LB_DISK="${ABS_PATH}/live-image-i386.img"
 7   RAW_DISK="${ABS_PATH}/live-image-i386.raw"
 8   LINUX_HEADERS="linux-headers-686-pae"
 9   REPO_ARCH="i386"
10 else
11   LB_DISK="${ABS_PATH}-amd64.img"
12   RAW_DISK="${ABS_PATH}/live-image-amd64.raw"
13   LINUX_HEADERS="linux-headers-amd64"
14   REPO_ARCH="amd64"
15 fi
16
17 DEPSOK="${ABS_PATH}/.depsok"
18
19 MIRROR_DIR="${ABS_PATH}/aptly"
20 PKGL_LB="${MIRROR_DIR}/lb_packages.list"
21 PKGL_DIR="${ABS_PATH}/config/package-lists"
22 PKGL_STR="$( cat ${PKGL_LB} ${PKGL_DIR}/* | grep -v '^#' | sed '/^$/d' | sort |
paste -s -d'|' | sed 's/|/ | /g' )" FILTER="Priority (required) | Priority
(important) | Priority (standard) | ${PKGL_STR}"
23
```

```
OPTS="(update | img | clean | purge)"

# - - - - -


err() {
  echo
  echo "  [*] Error: $1"
}


usage() {
  echo
  echo
"------------------------------------------------------------------------------
------"
  echo
  echo "  Usage: $0 -o $OPTS"
  echo
  echo "  Run 'update' first if executing for the first time. Must be connected
to the Internet."
  echo "  Run 'img' to build ISO/HDD file. Inbetween builds, run 'clean' or
'purge'."
  echo
  echo
"------------------------------------------------------------------------------
------"
  echo
  exit 1
}


setmirror() {
  cat ${MIRROR_DIR}/aptly.conf | sed "s|\(^ *\"rootDir\": \).*|\1\"${
MIRROR_DIR}\",|" > /root/.aptly.conf
}


imgbuild() {
  setmirror

  # check if repo key is in place
  if ! [ -f ${ABS_PATH}/config/archives/aptly.key ]; then
    gpg --export --armor > ${ABS_PATH}/config/archives/aptly.key
  fi

  # create and publish mirror snapshots
  aptly snapshot create stretch-main-spei from mirror stretch-main
  aptly snapshot create stretch-updates-spei from mirror stretch-updates
  aptly snapshot create stretch-security-spei from mirror stretch-security

```

```
  aptly snapshot merge -latest stretch-final-spei stretch-main-spei stretch-updates-spei stretch-security-spei
  aptly publish snapshot -distribution=stretch stretch-final-spei

  # ensure mirror is not yet running, then run it
  kill $( pgrep -f "aptly serve" ) 2> /dev/null
  aptly serve &

  # run build
  lb build

  # stop mirror and wipe snapshots
  kill $( pgrep -f "aptly serve" ) 2> /dev/null

  aptly publish drop stretch

  aptly snapshot drop stretch-final-spei
  aptly snapshot drop stretch-security-spei
  aptly snapshot drop stretch-updates-spei
  aptly snapshot drop stretch-main-spei
}


update() {
  if ! [ -d /root/.gnupg ]; then
    rngd -r /dev/urandom
    gpg --gen-key --batch ${MIRROR_DIR}/genkey.unattended
    gpg --no-default-keyring --keyring /usr/share/keyrings/debian-archive-keyring.gpg --export | gpg --no-default-keyring --keyring trustedkeys.gpg --import
    killall rngd
  fi

  setmirror

  mirrors="$( aptly mirror list | grep '^ * ' | sed 's/.*\[\([A-Za-z-]*\).*/\1/' )"

  echo "$mirrors" | grep stretch-main      &> /dev/null && aptly mirror drop stretch-main
  echo "$mirrors" | grep stretch-updates   &> /dev/null && aptly mirror drop stretch-updates
  echo "$mirrors" | grep stretch-security  &> /dev/null && aptly mirror drop stretch-security

  aptly mirror create -architectures=${REPO_ARCH} -filter="$FILTER" -filter -with-deps stretch-main http://ftp.br.debian.org/debian/ stretch main contrib non-free
  aptly mirror create -architectures=${REPO_ARCH} -filter="$FILTER" -filter -with-deps stretch-updates http://ftp.br.debian.org/debian/ stretch-updates main contrib non-free
  aptly mirror create -architectures=${REPO_ARCH} -filter="$FILTER" -filter
```

```
-with-deps stretch-security http://security.debian.org/debian-security/
stretch/updates main contrib non-free
109
110   aptly mirror update stretch-main
111   aptly mirror update stretch-updates
112   aptly mirror update stretch-security
113 }
114
115
116 clean () {
117   rm -rf ${LB_DISK}
118   [ -n "$1" ] && lb clean --purge || lb clean
119 }
120
121
122 deps() {
123   # add necessary repository sections & update
124   sed -i 's/\(main\) *$/\1 contrib non-free/' /etc/apt/sources.list
125   apt-get update
126
127   apt-get -y install --no-install-recommends ${LINUX_HEADERS} live-build mbr
netcat-traditional syslinux aptly rng-tools dirmngr
128 }
129
130
131 # - - - - -
132
133
134 if [ $( id -u ) -ne 0 ]; then
135   err "$0 must be run as root. Aborting..."
136   exit 1
137 fi
138
139 if ! uname -r | egrep '686-pae$|amd64$' &> /dev/null; then
140   err "Must run on '686-pae' or 'amd64' kernel archs. Aborting..."
141   exit 1
142 fi
143
144 while getopts ":o:" opt; do
145   case "$opt" in
146     o)
147       option=${OPTARG}
148       ;;
149     *)
150       usage
151       ;;
152   esac
153 done
154
155 [ -z $option ] && { err "No option specified, aborting."; usage; }
156
```

```
157 # check deps
158 if [ ! -f ${DEPSOK} ]; then
159   if [ $( which lb ) ] && [ $( which install-mbr ) ] && [ $( which nc ) ] && [
$( which syslinux ) ] && [ $( which aptly ) ] && [ $( which rngd ) ] && [ $( which
dirmngr ) ]; then
160     echo "All dependencies met. Continuing..."
161   else
162     echo "Missing dependencies. Installing..."
163     deps
164   fi
165
166   touch ${DEPSOK}
167 fi
168
169 case ${option} in
170   "img")
171     mirrors="$( aptly mirror list | grep '^ * ' | sed 's/.*\[\([A-Za-z-
]*\).*/\1/' )"
172     ! echo "$mirrors" | grep stretch-main &> /dev/null && { err "No 'stretch-
main' mirror detected, run '$0 -o update' first."; usage; }
173     ! echo "$mirrors" | grep stretch-updates &> /dev/null && { err "No
'stretch-updates' mirror detected, run '$0 -oupdate' first."; usage; }
174     ! echo "$mirrors" | grep stretch-security &> /dev/null && { err "No
'stretch-security' mirror detected, run '$0 -o update' first."; usage; }
175
176     imgbuild
177     ;;
178   "update")
179     update
180     ;;
181   "clean")
182     clean
183     ;;
184   "purge")
185     clean all
186     ;;
187   *)
188     usage
189     ;;
190 esac
```

O *script* acima é relativamente complexo, então convidamos o aluno a estudá-lo atentamente. Em linhas gerais, o objetivo é automatizar o uso e criação de repositórios locais usando o Aptly antes de iniciar o *build* de um sistema customizado. Também há a checagem de dependências dos pacotes necessários ao funcionamento do *live-build* e do Aptly, conjuntamente.

4. Vamos agora gerar a lista de pacotes que o Aptly deve manter localmente para acelerar a construção do sistema customizado. Crie um diretório de nome `aptly` dentro da pasta atual:

```
# mkdir /live/aptly-basic/aptly
```

Dentro dele, crie o arquivo novo `/live/aptly-basic/aptly/aptly.conf` com o seguinte conteúdo:

```
{
  "rootDir": "/live/aptly-basic/aptly",
  "downloadConcurrency": 4,
  "downloadSpeedLimit": 0,
  "architectures": [],
  "dependencyFollowSuggests": false,
  "dependencyFollowRecommends": false,
  "dependencyFollowAllVariants": false,
  "dependencyFollowSource": false,
  "dependencyVerboseResolve": false,
  "gpgDisableSign": false,
  "gpgDisableVerify": false,
  "gpgProvider": "gpg",
  "downloadSourcePackages": false,
  "skipLegacyPool": true,
  "ppaDistributorID": "ubuntu",
  "ppaCodename": "",
  "skipContentsPublishing": false,
  "FileSystemPublishEndpoints": {},
  "S3PublishEndpoints": {},
  "SwiftPublishEndpoints": {}
}
```

Basicamente, no arquivo acima definimos a raiz do repositório local que será criado, bem como quais pacotes serão baixados pelo Aptly (se apenas dependências básicas, ou também pacotes recomendados/sugeridos).

5. Agora, crie o arquivo novo `/live/aptly-basic/aptly/genkey.unattended` com o seguinte conteúdo:

```
    Key-Type: default
    Subkey-Type: default
    Name-Real: ESR
    Name-Email: suporte@esr.rnp.br
    Expire-Date: 0
    %no-protection
    %commit
    %echo done
```

Todos os pacotes mantidos no repositório local do Aptly serão assinados com um par de chaves criado sob demanda — as informações de geração das chaves são definidas no arquivo acima.

6. Crie o arquivo novo `/live/aptly-basic/aptly/lb_packages.list` com o seguinte conteúdo:

```
adduser
apt
apt-utils
base-files
base-passwd
bash
bsdmainutils
bsdutils
busybox
coreutils
cpio
cron
dash
dbus
dctrl-tools
debconf
debconf-i18n
debian-archive-keyring
debianutils
diffutils
dmidecode
dmsetup
dosfstools
dpkg
e2fslibs
e2fsprogs
extlinux
findutils
firmware-linux-free
gcc-6-base
gnupg
gnupg-agent
gpgv
grep
grub-common
grub-efi-amd64-bin
grub-efi-ia32-bin
gzip
hdmi2usb-fx2-firmware
hostname
ifupdown
init
initramfs-tools
initramfs-tools-core
init-system-helpers
iproute2
iptables
iputils-ping
irqbalance
isc-dhcp-client
isc-dhcp-common
```

```
isolinux
ixo-usb-jtag
keyboard-configuration
klibc-utils
kmod
krb5-locales
libacl1
libapparmor1
libapt-inst2.0
libapt-pkg5.0
libassuan0
libattr1
libaudit1
libaudit-common
libblkid1
libbsd0
libbz2-1.0
libc6
libcap2
libcap-ng0
libc-bin
libc-l10n
libcomerr2
libcryptsetup4
libdb5.3
libdbus-1-3
libdebconfclient0
libdevmapper1.02.1
libdns-export162
libedit2
libelf1
libestr0
libexpat1
libfastjson4
libfdisk1
libffi6
libgcc1
libgcrypt20
libgdbm3
libglib2.0-0
libglib2.0-data
libgmp10
libgnutls30
libgpg-error0
libgssapi-krb5-2
libhogweed4
libicu57
libidn11
libidn2-0
libip4tc0
libip6tc0
```

```
libiptc0
libisc-export160
libk5crypto3
libkeyutils1
libklibc
libkmod2
libkrb5-3
libkrb5support0
libksba8
liblocale-gettext-perl
liblogging-stdlog0
liblognorm5
liblz4-1
liblzma5
libmnl0
libmount1
libncurses5
libncursesw5
libnetfilter-conntrack3
libnettle6
libnewt0.52
libnfnetlink0
libnpth0
libnuma1
libp11-kit0
libpam0g
libpam-modules
libpam-modules-bin
libpam-runtime
libpcre3
libpipeline1
libpopt0
libprocps6
libpsl5
libreadline7
librsvg2-bin
libseccomp2
libselinux1
libsemanage1
libsemanage-common
libsepol1
libslang2
libsmartcols1
libsqlite3-0
libss2
libssl1.0.2
libssl1.1
libstdc++6
libsystemd0
libtasn1-6
libtext-charwidth-perl
```

```
libtext-iconv-perl
libtext-wrapi18n-perl
libtinfo5
libudev1
libunistring0
libustr-1.0-1
libuuid1
libx11-6
libx11-data
libxapian30
libxau6
libxcb1
libxdmcp6
libxext6
libxml2
libxmuu1
libxtables12
linux-base
linux-image-4.9.0-8-amd64
linux-image-amd64
live-boot
live-boot-doc
live-boot-initramfs-tools
live-config
live-config-doc
live-config-systemd
live-tools
locales
login
logrotate
lsb-base
mawk
mount
multiarch-support
nano
ncurses-base
ncurses-bin
netbase
openssh-client
parted
passwd
perl-base
pinentry-curses
procps
readline-common
rsync
rsyslog
sed
sensible-utils
sgml-base
shared-mime-info
```

```
squashfs-tools
sudo
syslinux
syslinux-common
systemd
systemd-sysv
sysvinit-utils
tar
tasksel
tasksel-data
tzdata
udev
user-setup
util-linux
uuid-runtime
vim-common
vim-tiny
wget
whiptail
xauth
xdg-user-dirs
xml-core
xorriso
xxd
zlib1g
zsync
```

A lista acima foi construída a partir da lista de pacotes instalados automaticamente no sistema-base (que copiamos no passo 8 da atividade anterior), bem como através de tentativa-e-erro durante *builds* consecutivos usando o repositório local. Caso algum pacote essencial esteja faltando, o *build* irá falhar e reclamar que o pacote não está disponível no repositório local — nesse caso, adicionamos o pacote à lista acima e repetimos o *build*, até que não ocorram mais erros.

Se você estiver se perguntando: sim, produzir a lista acima levou UM BOM número de tentativas.

7. Temos que trocar os repositórios a serem usados durante o *build*: ao invés de usar os repositórios `ftp.br.debian.org` e `security.debian.org`, iremos usar o Aptly local, escutando em `127.0.0.1:8080`:

```
# sed -i 's|http://ftp\.br\.debian\.org/debian/|http://127\.0\.0\.1:8080/|'
/live/aptly-basic/config/bootstrap
```

```
# sed -i 's|http://security\.debian\.org/|http://127\.0\.0\.1:8080/|' /live/aptly-
basic/config/bootstrap
```

```
# sed -i 's|http://httpredir\.debian\.org/debian/|http://127\.0\.0\.1:8080/|'
/live/aptly-basic/config/bootstrap
```

```
# sed -i 's|http://ftp\.br\.debian\.org/debian/|http://127\.0\.0\.1:8080/|'
/live/aptly-basic/config/build
```

Uma vez que todos os repositórios serão concatenados em um único, gerenciado localmente pelo Aptly, não precisamos incluir as seções `security` ou `updates` no *chroot*:

```
# sed -i 's|^\(LB_SECURITY=\).*|\1\"false\"|' /live/aptly-basic/config/chroot
```

```
# sed -i 's|^\(LB_UPDATES=\).*|\1\"false\"|' /live/aptly-basic/config/chroot
```

Finalmente, vamos desabilitar a instalação de pacotes recomendados, bem como a checagem de confiança da chave de assinatura dos pacotes no repositório local (já que iremos usar uma chave auto-assinada):

```
# sed -i 's|^\(LB_APT_RECOMMENDS=\).*|\1\"false"|' /live/aptly-basic/config/common
```

```
# sed -i 's|^\(LB_APT_SECURE=\).*|\1\"false"|' /live/aptly-basic/config/common
```

```
# sed -i 's|^\(LB_APT_SOURCE_ARCHIVES=\).*|\1\"false"|' /live/aptly-
basic/config/common
```

```
# sed -i 's|^\(APT_OPTIONS.*\)\"|\1 -o Acquire::ForceIPv4=true\"|' /live/aptly-
basic/config/common
```

```
# sed -i 's|^\(DEBOOTSTRAP_OPTIONS.*\)\"|\1--no-check-gpg\"|' /live/aptly-
basic/config/common
```

8. Algumas das coisas que mais ocupam espaço em instalações minimalistas — além de bibliotecas e *drivers* essenciais — são artefatos como páginas de manual, documentação e *locales* (traduções de *strings* para diferentes linguagens). Não precisamos de nada disso em nosso sistema!

   Crie o arquivo novo `/live/aptly-basic/config/hooks/normal/0450-stripped.hook.chroot` com o seguinte conteúdo:

```
#!/bin/sh

set -e

# remover pacotes desnecessarios
for PACKAGE in apt-utils aptitude man-db manpages info wget dselect
do
    if ! apt-get remove --purge --yes "${PACKAGE}"
    then
        echo "WARNING: ${PACKAGE} isn't installed"
    fi
done

# limpar a cache apt por completo
apt-get autoremove --yes || true
apt-get clean
find /var/cache/apt/ -type f -exec rm -f {} \;
find /var/lib/apt/lists/ -type f -exec rm -f {} \;

# remover arquivos temporarios
find . -name *~ -print0 | xargs -0 rm -f

# remover locales ! en/pt-br
find /usr/share/locale -maxdepth 1 -type d -not -regex
'.*\(locale\|en\|pt_BR\)$' | xargs rm -rf

# remover paginas de manual e documentacao
rm -rf /usr/share/groff/*
rm -rf /usr/share/doc/*
rm -rf /usr/share/man/*
rm -rf /usr/share/info/*
rm -rf /usr/share/lintian/*
rm -rf /usr/share/linda/*
rm -rf /var/cache/man/*

# truncar logs
for FILE in $(find /var/log/ -type f)
do
    : > ${FILE}
done
```

O *script* acima será executado ao final do passo de *chroot*, e irá remover boa parte dos arquivos que não são integralmente necessários ao funcionamento do sistema, reduzindo o tamanho da imagem final consideravelmente.

9. Ufa! Chega de configurações — vamos atualizar o repositório local:

```
# cd /live/aptly-basic/ ; bash makebuild.sh -o update
```

O *script* irá detectar dependências faltantes e instalá-las, e posteriormente irá baixar todos os pacotes que mapeamos no passo (6) desta atividade para o repositório APT local.

10. Uma vez concluído o download, rode novamente o *build* do sistema customizado:

```
# cd /live/aptly-basic/ ; date > buildtime ; bash makebuild.sh -o img  ; date >>
buildtime
```

O Aptly irá publicar um repositório local com todos os pacotes que baixamos no passo anterior, e logo a seguir o comando `lb build` será invocado. Note como a velocidade de obtenção dos pacotes é significativamente superior, desta vez.

Note que criamos um arquivo `/live/aptly-basic/buildtime` para registrar o tempo de *build*, desta vez. Confira seu conteúdo:

```
# cat /live/aptly-basic/buildtime
dom nov 18 01:24:35 -02 2018
dom nov 18 01:26:56 -02 2018
```

No mesmo sistema que produziu o *build* da atividade (3), note que o tempo caiu de cerca de seis minutos para, agora, cerca de 2 minutos e 20 segundos. Significativo, não? E outra vantagem — se precisarmos refazer o *build*, todos os pacotes já estão na *cache* local, e não precisam ser baixados novamente!

11. Vamos ver se nossas otimizações de espaço surtiram efeito no tamanho da imagem:

```
# du -sh /live/aptly-basic/live-image-amd64.hybrid.iso
129M    /live/aptly-basic/live-image-amd64.hybrid.iso
```

De 216 MB na imagem anterior, temos agora uma imagem equivalente de tamanho igual a 129 MB, apenas com a remoção de documentação, *locales* e outros arquivos acessórios. Uma redução de 40%!

5) Construindo uma imagem mais... divertida?

É bem verdade que apesar de extremamente enxuto, nosso sistema customizado não faz nada... interessante, até aqui. Para incrementar suas funcionalidades, vamos produzir um sistema que possua um ambiente gráfico e um navegador.

1. Entre na pasta `/live/aptly-basic` e limpe os arquivos de trabalho do *build* anterior. Vamos usá-la como base para nossa próxima imagem.

```
# cd /live/aptly-basic/ ; bash makebuild.sh -o purge
[2018-11-18 01:35:53] lb clean --purge
P: Cleaning chroot
```

Note que o diretório ainda é significativamente grande, em razão da *cache* de pacotes do Aptly:

```
# du -sh /live/aptly-basic/
263M    /live/aptly-basic/
```

2. Copie o diretório `/live/aptly-basic` para um novo `/live/aptly-x`, e entre nesse diretório:

```
# cp -a /live/aptly-basic /live/aptly-x ; cd /live/aptly-x
```

3. Podemos customizar a lista de pacotes instalados em uma imagem do *live-build* criando arquivos de pacotes no diretório `config/package-lists`. Crie o arquivo novo `/live/aptly-x/config/package-lists/my.list.chroot` com o seguinte conteúdo:

```
 1 feh
 2 firefox-esr
 3 fluxbox
 4 initramfs-tools
 5 keyboard-configuration
 6 live-tools
 7 locales
 8 nodm
 9 openssl
10 rsync
11 sudo
12 task-desktop
13 user-setup
14 uuid-runtime
15 xterm
16 x11-xserver-utils
```

No arquivo acima informamos que, além dos pacotes básicos do sistema *live*, instalaremos também o sistema gráfico X.Org, o gerenciador de janelas Fluxbox e o navegador web Mozilla Firefox, dentre outros pacotes.

4. É também possível customizar quais arquivos estão presentes no *build* final, inserindo-os em uma raiz alternativa no diretório `config/includes.chroot`. Suponha que queiramos que o usuário `user`, ao fazer login no sistema *live*, tenha lançado para si o gerenciador de janelas Fluxbox.

   Crie o caminho de diretórios apropriado:

```
# mkdir -p /live/aptly-x/config/includes.chroot/home/user/
```

   Agora, crie o arquivo `.xinitrc` com a configuração adequada:

```
# echo "startfluxbox" > /live/aptly-x/config/includes.chroot/home/user/.xinitrc
```

Quando iniciarmos o sistema *live*, os diretórios e arquivos que criamos acima já estarão presentes na distribuição, e o gerenciado de login `nodm` se encarregará de iniciar o Fluxbox automaticamente. Esse é um excelente método para distribuir arquivos e configurações em sistemas especialistas, como um servidor web embarcado, por exemplo.

5. Tudo pronto? Vamos atualizar a lista de pacotes do repositório local, já que fizemos várias adições novas no passo (3):

```
# cd /live/aptly-x ; bash makebuild.sh -o update
```

Os pacotes faltantes e suas dependências serão baixados para o *mirror* Aptly local, como esperado.

6. Faça o *build* do sistema customizado:

```
# cd /live/aptly-x/ ; date > buildtime ; bash makebuild.sh -o img  ; date >>
buildtime
```

Vamos ver qual foi o tempo de *build* para esse sistema, um pouco mais complexo que o anterior:

```
# cat /live/aptly-x/buildtime
dom nov 18 01:51:13 -02 2018
dom nov 18 01:55:12 -02 2018
```

Cerca de quatro minutos, muito bom. E quanto ao tamanho?

```
# du -sh /live/aptly-x/live-image-amd64.hybrid.iso
262M    /live/aptly-x/live-image-amd64.hybrid.iso
```

Com 262 MB, o tamanho é superior ao que tínhamos obtido com o sistema básico anterior, mas ainda é certamente muito inferior ao que poderíamos esperar de uma distribuição Linux de propósito geral — especialmente ao considerar que essa imagem inclui sistema gráfico, ambiente de janelas e o navegador Mozilla Firefox.

7. Vamos aos testes. Copie a imagem para sua máquina física usando o comando `scp` ou o programa WinSCP:

```
$ scp aluno@10.0.42.11:/live/aptly-x/live-image-amd64.hybrid.iso
/cygdrive/c/Users/fbs/Desktop/
aluno@10.0.42.11's password:
live-image-amd64.hybrid.iso
100%  262MB  59.6MB/s   00:04
```

Inicie a VM `iso-test` na console principal do Virtualbox. Não é necessário reconfigurá-la, já que o caminho do CD de *boot* aponta para o mesmo arquivo que sobrescrevemos na cópia acima.

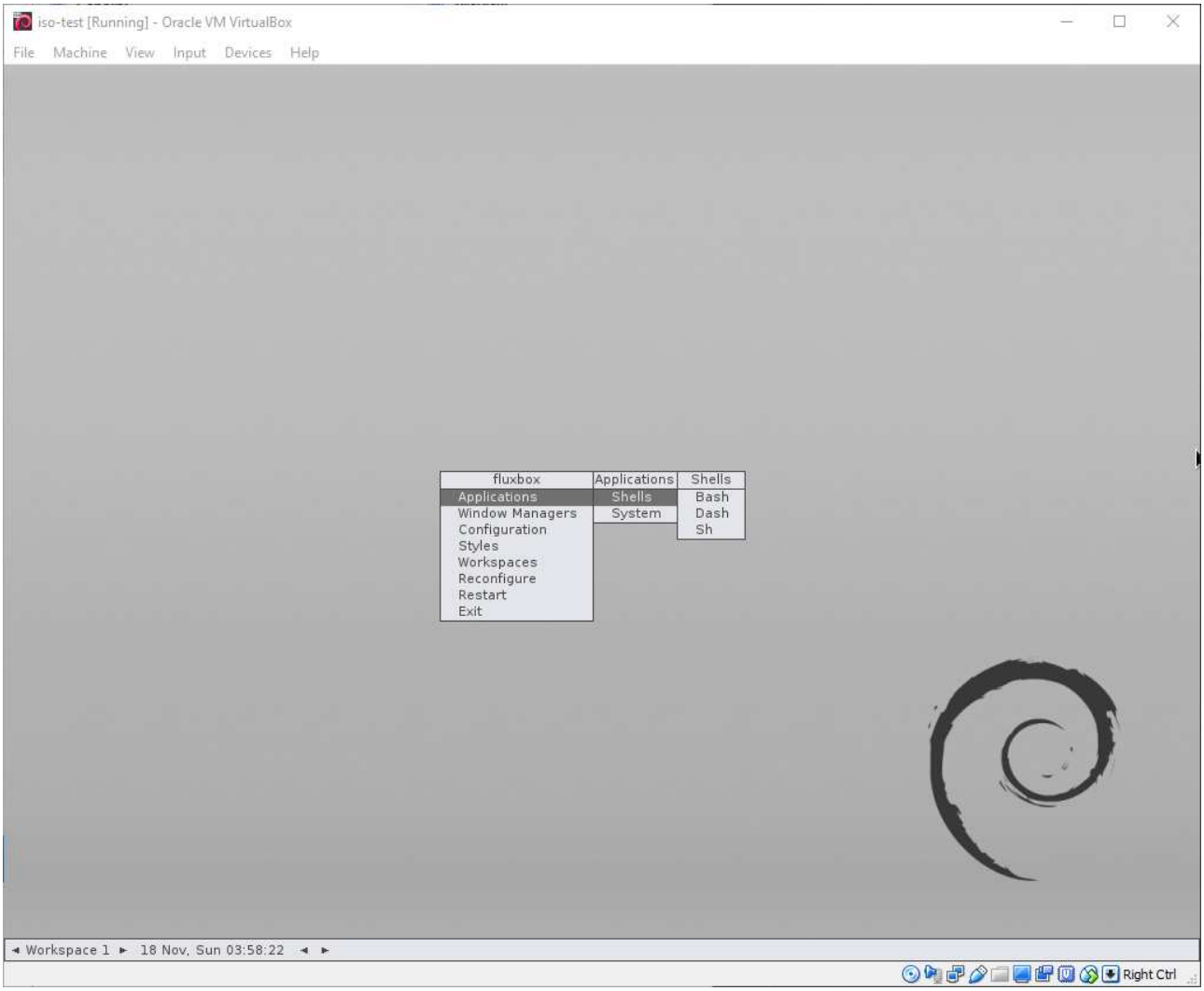

Figura 64. Fluxbox iniciado no sistema live

Legal, não é mesmo? Que tal navegar na Internet? Lance o `xterm` com o atalho `ALT + F1`, e invoque o Mozilla Firefox com o comando `firefox &`:

Figura 65. Navegação no Firefox com o sistema live

Temos aberto acima o site da Escola Superior de Redes, rodando dentro de um sistema customizado com tamanho inferior a 300MB, e que fizemos em alguns poucos passos.

8. Daqui pra frente, seu limite passa a ser sua imaginação:

   - Como poderíamos tornar a imagem acima ainda mais leve e eficiente? Há outros programas e navegadores que seriam mais apropriados para esse tipo de uso?
   - Para quais outras aplicações seria interessante produzir sistemas especialistas como o que fizemos aqui?
   - Que outras opções de segurança e *lockdown* poderíamos ativar em nosso sistema customizado para torná-lo ainda mais seguro?

9. Encerradas as nossas atividades com a máquina `live`, recomenda-se que o aluno a mantenha desligada a partir desta sessão. A grande quantidade de recursos demandada por esse sistema para operar com sucesso a torna um peso muito grande na execução das atividades das próximas sessões.

Sessão 10: Módulos de segurança do kernel

Nesta sessão iremos trabalhar com o módulo de segurança AppArmor, que implementa um modeloe de segurança MAC (*Mandatory Access Controls*) para o kernel Linux.

1) Topologia desta sessão

Criaremos apenas uma nova máquina nesta sessão, a saber:

- `lsm`, um ambiente para testes de módulos de segurança do kernel, os *Linux Security Modules*. Endereço IP 10.0.42.12/24.

1. Como de costume, vamos à criação dos registros DNS. Acesse a máquina `ns1` como o usuário `root`:

   ```
   # hostname ; whoami
   ns1
   root
   ```

   Edite o arquivo de zonas `/etc/nsd/zones/intnet.zone`, inserindo entradas A para a máquinas indicadas no começo desta atividade. **Não se esqueça** de incrementar o valor do serial no topo do arquivo!

   ```
   # nano /etc/nsd/zones/intnet.zone
   (...)
   ```

   ```
   # grep lsm /etc/nsd/zones/intnet.zone
   lsm     IN    A                10.0.42.12
   ```

   Faça o mesmo para o arquivo de zona reversa:

   ```
   # nano /etc/nsd/zones/10.0.42.zone
   ```

   ```
   # grep lsm /etc/nsd/zones/10.0.42.zone
   12      IN   PTR               lsm.intnet.
   ```

   Assine o arquivo de zonas usando o *script* criado anteriormente:

```
# bash /root/scripts/signzone-intnet.sh
reconfig start, read /etc/nsd/nsd.conf
ok
ok
ok
ok removed 11 rrsets, 10 messages and 0 key entries
```

Verifique a criação das entradas usando o comando `dig`:

```
# dig lsm.intnet +short
10.0.42.12
```

```
# dig -x 10.0.42.12 +short
lsm.intnet.
```

2) Criação do ambiente de segurança

1. Vamos criar a VM `lsm` e utilizá-la para testes de módulos de segurança do kernel. Clone a máquina `debian-template` para uma de nome `lsm`, com uma única interface de rede conectada à DMZ. O IP da máquina será 10.0.42.12/24.

   Concluída a clonagem, ligue a VM e logue como `root`. Use o script `/root/scripts/changehost.sh` para fazer a configuração automática, como de costume.

   ```
   # hostname ; whoami
   debian-template
   root
   ```

   ```
   # bash ~/scripts/changehost.sh -h lsm -i 10.0.42.12 -g 10.0.42.1
   Signing ssh_host_ecdsa_key.pub key...
   Signing ssh_host_ed25519_key.pub key...
   Signing ssh_host_rsa_key.pub key...
   Configuring host key trust...
   Configuring user key trust...
   All done!
   ```

2. Aplique o *baseline* de segurança à máquina `lsm`, repetindo o que fizemos no passo (2), atividade (2) da sessão 7:

   ```
   $ hostname ; whoami
   client
   ansible
   ```

```
$ sed -i '/\[srv\]/a lsm' ~/ansible/hosts
```

```
$ ansible-playbook -i ~/ansible/hosts -l lsm -Ke ansible_become_method=su
~/ansible/srv.yml ; ansible-playbook -i ~/ansible/hosts -l lsm ~/ansible/srv.yml
SUDO password:

(...)

PLAY RECAP
*********************************************************************************
***********************

lsm                        : ok=10   changed=8    unreachable=0    failed=0
```

3) Instalação do AppArmor

1. Vamos agora instalar o AppArmor, um sistema MAC que atua como módulo de segurança do kernel Linux com o objetivo de confinar programas a um conjunto limitado de recursos.

   Acesse a máquina `lsm` como o usuário `root`.

   ```
   # hostname ; whoami
   lsm
   root
   ```

2. Instale os pacotes:

   ```
   # apt-get install -y apparmor apparmor-utils
   ```

3. Habilite o AppArmor durante o *boot* do kernel — para isso, basta alterar a linha de *boot* padrão do GRUB usando os comandos a seguir:

   ```
   # mkdir -p /etc/default/grub.d
   ```

   ```
   # echo 'GRUB_CMDLINE_LINUX_DEFAULT="$GRUB_CMDLINE_LINUX_DEFAULT apparmor=1
   security=apparmor"' \
       > /etc/default/grub.d/apparmor.cfg
   ```

   Reconstrua a configuração do GRUB com o comando:

   ```
   # update-grub
   ```

Em seguida, reinicie a máquina para que as configurações realizadas sejam carregadas durante o próximo *boot*:

```
# reboot
```

4. Após o *reboot*, logue novamente como `root` e verifique o estado de execução do AppArmor:

```
# aa-status
apparmor module is loaded.
0 profiles are loaded.
0 profiles are in enforce mode.
0 profiles are in complain mode.
0 processes have profiles defined.
0 processes are in enforce mode.
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
```

4) Criação de um perfil AppArmor para o servidor web Nginx

Em que o AppArmor pode incrementar a segurança do sistema? Vamos fazer um caso de teste com o servidor web Nginx: iremos criar um perfil de segurança para essa aplicação, definindo com precisão o que ela está ou não autorizada a fazer no sistema.

1. Primeiro, instale o Nginx:

```
# apt-get install -y nginx
```

Em sua máquina física, aponte o navegador para o IP da máquina `lsm`, 10.0.42.12. Voce deverá ver a página a seguir, comprovando que o Nginx foi instalado com sucesso:

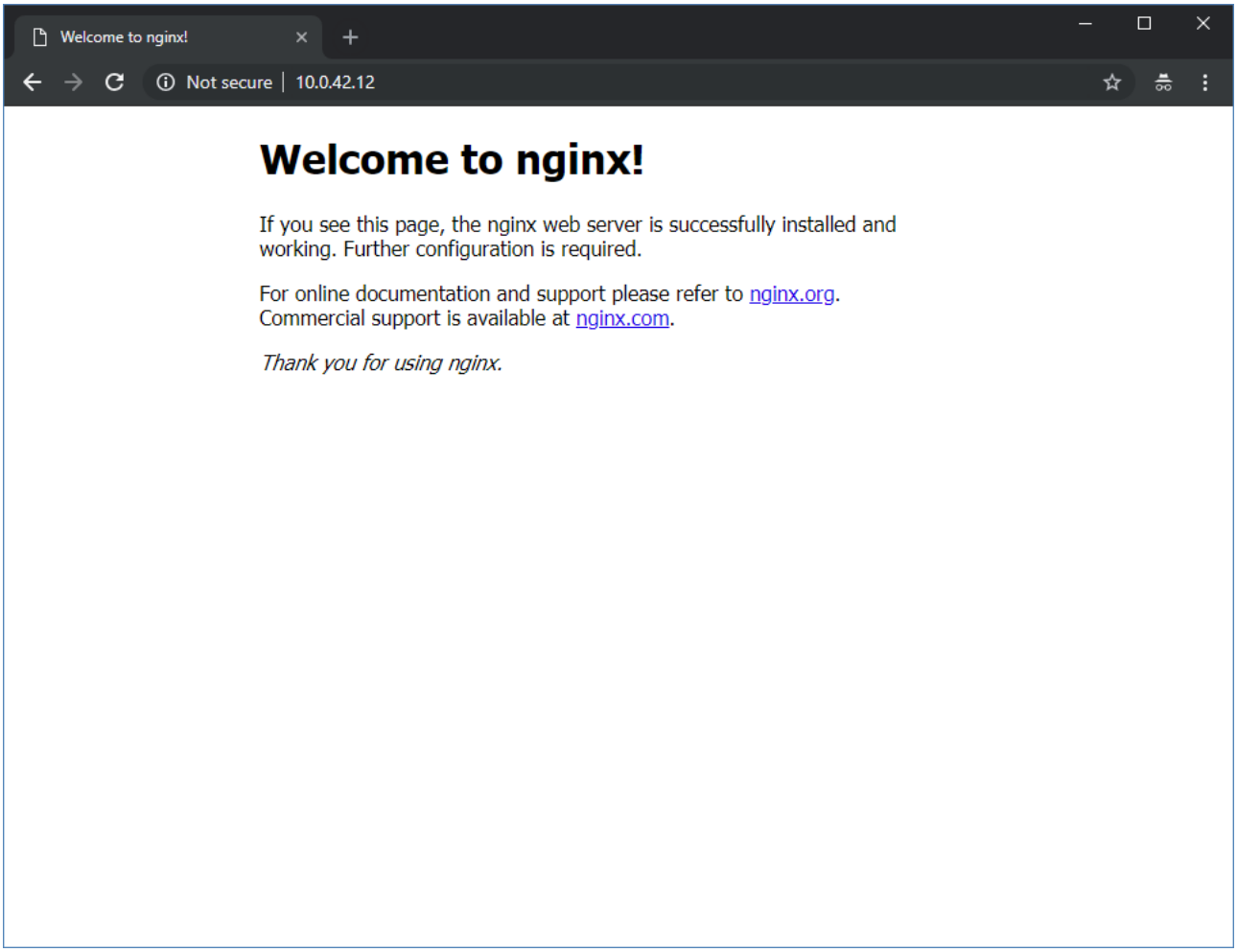

Figura 66. Nginx instalado com sucesso

2. Vamos criar duas pastas: `/data/www/safe`, à qual o Nginx deve ter acesso, e `/data/www/unsafe`, cujo acesso deve ser negado ao Nginx pelo AppArmor.

```
# mkdir -p /data/www/safe
```

```
# mkdir -p /data/www/unsafe
```

Em cada uma das pastas, crie um arquivo `index.html` que indique de forma clara no navegador que estamos, de fato, navegando no local pretendido.

```
# cat << EOF >> /data/www/safe/index.html
<html>
    <b>Oi! Acessar este arquivo e permitido.</b>
</html>
EOF
```

```
# cat << EOF >> /data/www/unsafe/index.html
<html>
    <b>Oi! Acessar este arquivo NAO e permitido.</b>
</html>
EOF
```

3. Altere a configuração do Nginx para servir esses arquivos na porta 8080/TCP, como se segue:

```
# cat << EOF >> /etc/nginx/conf.d/apparmor.conf
server {
    listen 8080;
    location / {
            root /data/www;
    }
}
EOF
```

Recarregue a configuração do Nginx:

```
# systemctl reload nginx
```

Teste o acesso à URL http://10.0.42.12:8080/safe/:

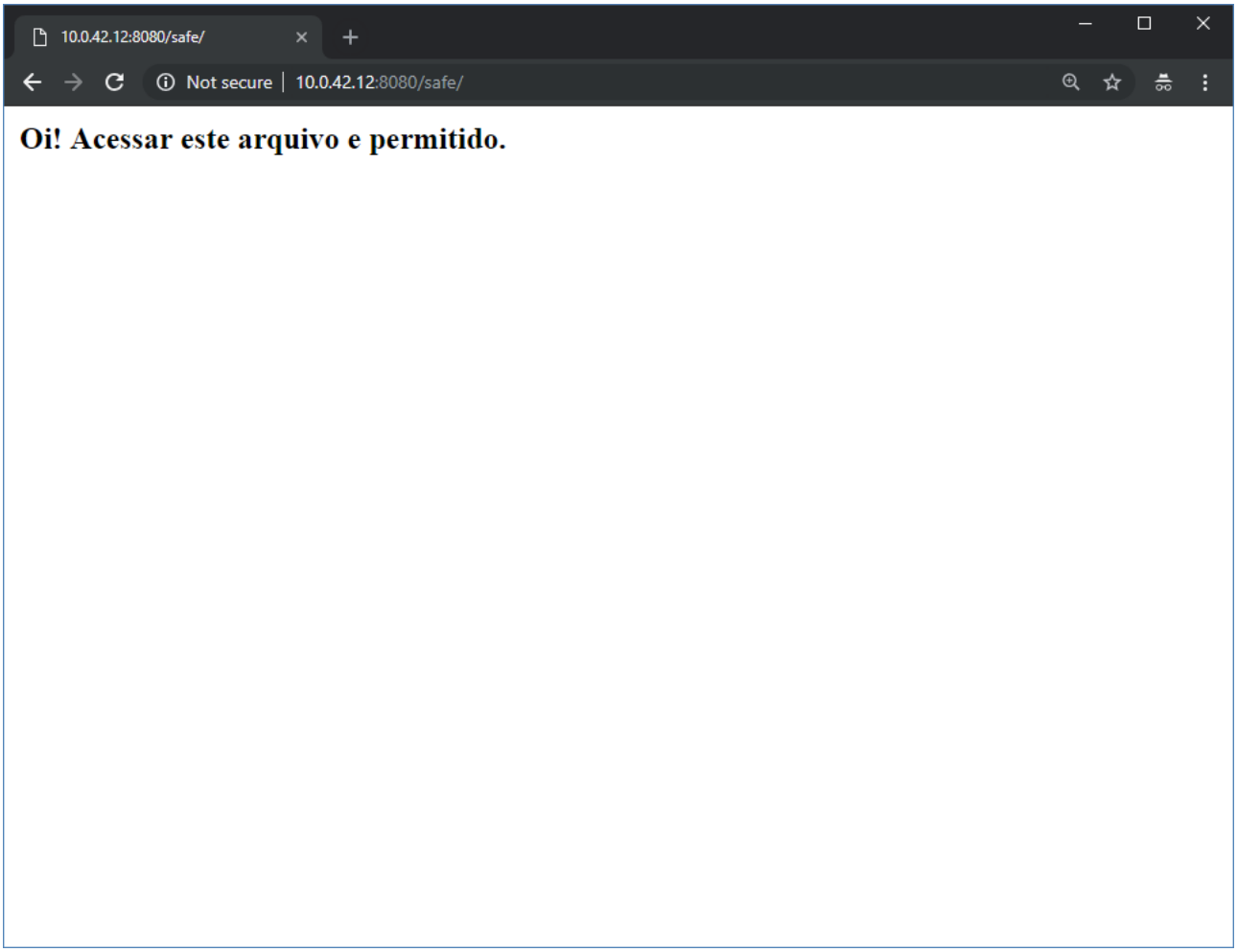

Figura 67. Acesso permitido à pasta SAFE

O acesso é permitido, como esperado. Evidentemente, o acesso a `http://10.0.42.12:8080/unsafe/` também é autorizado, já que ainda não configuramos o AppArmor.

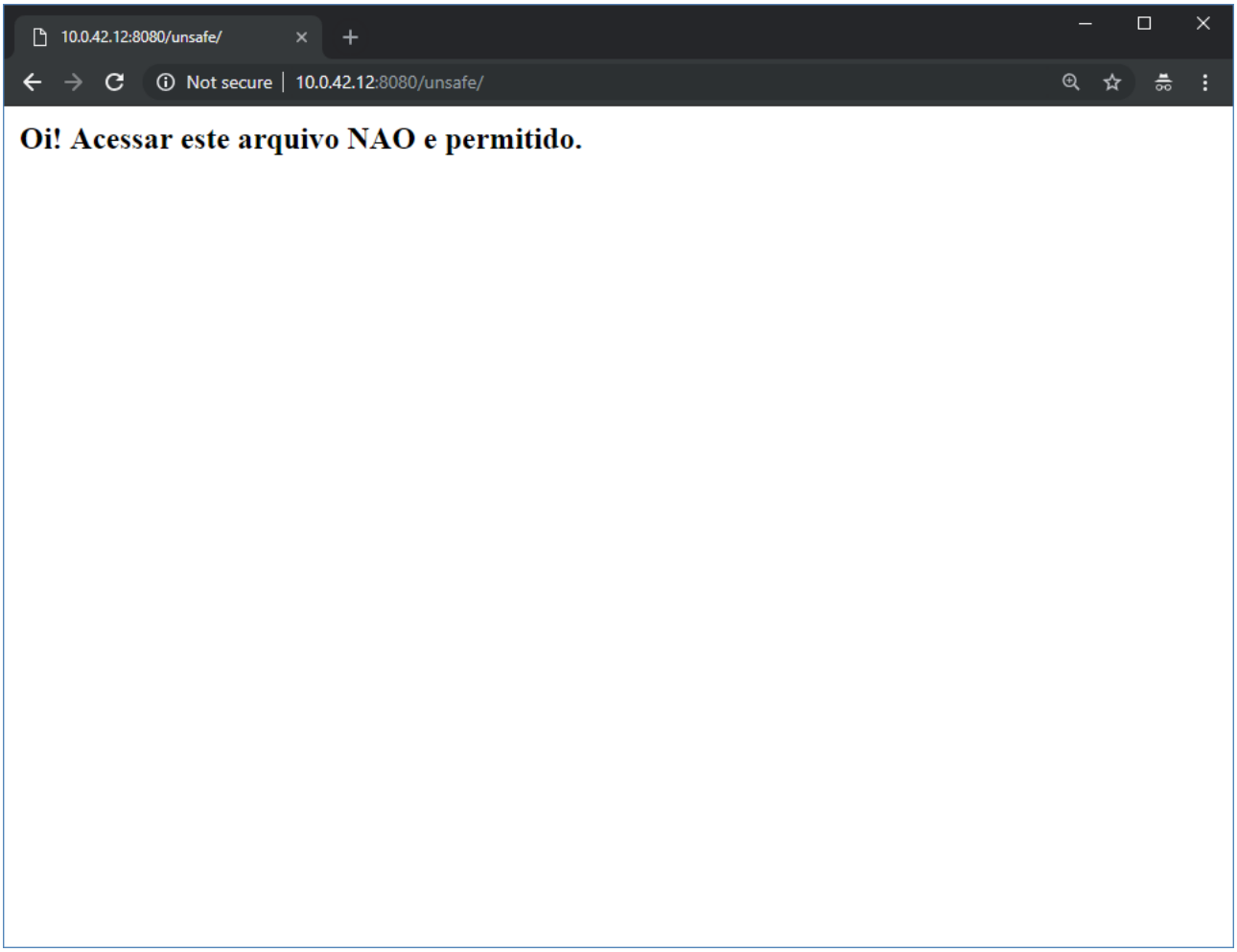

Figura 68. Acesso permitido à pasta UNSAFE

4. O Debian possui um conjunto de perfis pré-prontos para o AppArmor que irão nos auxiliar na tarefa de configuração — instale o pacote `apparmor-profiles`:

```
# apt-get install -y apparmor-profiles
```

Para verificar o estado dos perfis do AppArmor, execute o comando `apparmor_status`:

```
# apparmor_status
apparmor module is loaded.
33 profiles are loaded.
0 profiles are in enforce mode.
33 profiles are in complain mode.
   /usr/lib/dovecot/anvil
   /usr/lib/dovecot/auth
   /usr/lib/dovecot/config
   /usr/lib/dovecot/deliver
   /usr/lib/dovecot/dict
   /usr/lib/dovecot/dovecot-auth
   /usr/lib/dovecot/dovecot-lda
   /usr/lib/dovecot/dovecot-lda///usr/sbin/sendmail
   /usr/lib/dovecot/imap
   /usr/lib/dovecot/imap-login
   /usr/lib/dovecot/lmtp
   /usr/lib/dovecot/log
   /usr/lib/dovecot/managesieve
   /usr/lib/dovecot/managesieve-login
   /usr/lib/dovecot/pop3
   /usr/lib/dovecot/pop3-login
   /usr/lib/dovecot/ssl-params
   /usr/sbin/avahi-daemon
   /usr/sbin/dnsmasq
   /usr/sbin/dnsmasq//libvirt_leaseshelper
   /usr/sbin/dovecot
   /usr/sbin/identd
   /usr/sbin/mdnsd
   /usr/sbin/nmbd
   /usr/sbin/nscd
   /usr/sbin/smbd
   /usr/sbin/smbldap-useradd
   /usr/sbin/smbldap-useradd///etc/init.d/nscd
   /usr/{sbin/traceroute,bin/traceroute.db}
   klogd
   ping
   syslog-ng
   syslogd
1 processes have profiles defined.
0 processes are in enforce mode.
0 processes are in complain mode.
1 processes are unconfined but have a profile defined.
   /usr/sbin/nscd (400)
```

5. Vamos criar um perfil de acesso customizado para o Nginx. Entre na pasta `/etc/apparmor.d` e use o comando `aa-autodep nginx` para criar um perfil em branco:

```
# cd /etc/apparmor.d
```

```
# aa-autodep nginx
Writing updated profile for /usr/sbin/nginx.
```

Uma vez criado o perfil, use o comando `aa-complain nginx` para colocá-lo no modo *complain* — nesse perfil, violações serão autorizadas, e apenas um alerta será gerado:

```
# aa-complain nginx
Setting /usr/sbin/nginx to complain mode.
```

Finalmente, reinicie o Nginx:

```
# systemctl restart nginx
```

6. Em seu navegador, acesse a URL `http://10.0.42.12:8080/safe/` uma vez, e **apenas** essa URL (i.e. não acesse a área *unsafe*). Essa requisição gerará um evento no log de acessos do Nginx, o qual processaremos a seguir.
7. Feito isso, use o comando `aa-logprof` para processar os eventos observados no log do Nginx e gerar um perfil de acesso para a aplicação.

   O comando acima irá processar os logs do Nginx e atualizar o perfil de acesso do programa. Para cada tipo de acesso identificado, você deverá responder se deseja autorizar ou negar uma capacidade à aplicação. Assumindo que o sistema não está sob ataque, é razoável supor que todos os acessos são legítimos, portanto autorize todos com o atalho `A` (para *Allow*).

   Ao final do processo, o programa irá perguntar se você deseja salvar as informações no perfil do Nginx. Confirme com o atalho `S` (para *Save Changes*).

   Temos abaixo uma execução típica do `aa-logprof` para esse cenário:

```
# aa-logprof
Reading log entries from /var/log/syslog.
Updating AppArmor profiles in /etc/apparmor.d.
Complain-mode changes:

Profile:    /usr/sbin/nginx
Capability: dac_override
Severity:   9

 [1 - capability dac_override,]
(A)llow / [(D)eny] / (I)gnore / Audi(t) / Abo(r)t / (F)inish
Adding capability dac_override, to profile.

Profile:  /usr/sbin/nginx
Path:     /data/www/safe/index.html
New Mode: r
Severity: unknown
```

```
 [1 - /data/www/safe/index.html r,]
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Audi(t) /
Abo(r)t / (F)inish
Adding /data/www/safe/index.html r, to profile.

Profile:  /usr/sbin/nginx
Path:     /etc/ssl/openssl.cnf
New Mode: r
Severity: 2

 [1 - #include <abstractions/openssl>]
  2 - #include <abstractions/ssl_keys>
  3 - /etc/ssl/openssl.cnf r,
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Audi(t) /
Abo(r)t / (F)inish
Adding #include <abstractions/openssl> to profile.

Profile:  /usr/sbin/nginx
Path:     /usr/share/nginx/modules-available/mod-http-dav-ext.conf
New Mode: r
Severity: unknown

 [1 - /usr/share/nginx/modules-available/mod-http-dav-ext.conf r,]
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Audi(t) /
Abo(r)t / (F)inish
Adding /usr/share/nginx/modules-available/mod-http-dav-ext.conf r, to profile.

Profile:  /usr/sbin/nginx
Path:     /usr/lib/nginx/modules/ngx_http_auth_pam_module.so
Old Mode: r
New Mode: mr
Severity: unknown

 [1 - #include <abstractions/ubuntu-browsers.d/plugins-common>]
  2 - /{usr/,}lib{,32,64}/** mr,
  3 - /usr/lib/nginx/modules/ngx_http_auth_pam_module.so mr,
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Audi(t) /
Abo(r)t / (F)inish
Adding #include <abstractions/ubuntu-browsers.d/plugins-common> to profile.

Profile:  /usr/sbin/nginx
Path:     /etc/nginx/nginx.conf
New Mode: r
Severity: unknown

 [1 - /etc/nginx/nginx.conf r,]
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Audi(t) /
Abo(r)t / (F)inish
Adding /etc/nginx/nginx.conf r, to profile.
```

```
Profile:  /usr/sbin/nginx
Path:     /etc/nginx/modules-enabled/
New Mode: r
Severity: unknown

 [1 - /etc/nginx/modules-enabled/ r,]
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Audi(t) /
Abo(r)t / (F)inish
Adding /etc/nginx/modules-enabled/ r, to profile.

Profile:  /usr/sbin/nginx
Path:     /usr/share/nginx/modules-available/mod-http-auth-pam.conf
New Mode: r
Severity: unknown

 [1 - /usr/share/nginx/modules-available/mod-http-auth-pam.conf r,]
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Audi(t) /
Abo(r)t / (F)inish
Adding /usr/share/nginx/modules-available/mod-http-auth-pam.conf r, to profile.

Profile:  /usr/sbin/nginx
Path:     /var/log/nginx/access.log
New Mode: w
Severity: 8

 [1 - /var/log/nginx/access.log w,]
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Audi(t) /
Abo(r)t / (F)inish
Adding /var/log/nginx/access.log w, to profile.

Profile:  /usr/sbin/nginx
Path:     /var/log/nginx/error.log
New Mode: w
Severity: 8

 [1 - /var/log/nginx/error.log w,]
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Audi(t) /
Abo(r)t / (F)inish
Adding /var/log/nginx/error.log w, to profile.

Profile:  /usr/sbin/nginx
Path:     /usr/share/nginx/modules-available/mod-http-echo.conf
New Mode: r
Severity: unknown

 [1 - /usr/share/nginx/modules-available/mod-http-echo.conf r,]
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Audi(t) /
Abo(r)t / (F)inish
Adding /usr/share/nginx/modules-available/mod-http-echo.conf r, to profile.

= Changed Local Profiles =
```

```
The following local profiles were changed. Would you like to save them?

 [1 - /usr/sbin/nginx]
(S)ave Changes / Save Selec(t)ed Profile / [(V)iew Changes] / View Changes b/w
(C)lean profiles / Abo(r)t
Writing updated profile for /usr/sbin/nginx.
```

8. O isso fez? Confira o conteúdo do arquivo `/etc/apparmor.d/usr.sbin.nginx`:

```
# cat /etc/apparmor.d/usr.sbin.nginx
# Last Modified: Sun Nov 18 02:48:57 2018
#include <tunables/global>

/usr/sbin/nginx flags=(complain) {
  #include <abstractions/base>
  #include <abstractions/openssl>
  #include <abstractions/ubuntu-browsers.d/plugins-common>

  capability dac_override,

  /data/www/safe/index.html r,
  /etc/nginx/modules-enabled/ r,
  /etc/nginx/nginx.conf r,
  /lib/x86_64-linux-gnu/ld-*.so mr,
  /usr/sbin/nginx mr,
  /usr/share/nginx/modules-available/mod-http-auth-pam.conf r,
  /usr/share/nginx/modules-available/mod-http-dav-ext.conf r,
  /usr/share/nginx/modules-available/mod-http-echo.conf r,
  /var/log/nginx/access.log w,
  /var/log/nginx/error.log w,

}
```

   Cada um dos acessos autorizados foi adicionado em uma linha do arquivo acima, com a permissão correspondente ao final. Observe que **qualquer** tipo de acesso que não conste do arquivo acima será negado pelo AppArmor ao ativarmos o modo *enforcing* do sistema.

9. Tudo pronto... certo? Coloque o Nginx em modo *enforce*:

```
# aa-enforce nginx
Setting /usr/sbin/nginx to enforce mode.
```

   E reinicie ambos AppArmor e Nginx:

```
# systemctl reload apparmor
```

```
# systemctl restart nginx
Job for nginx.service failed because the control process exited with error code.
See "systemctl status nginx.service" and "journalctl -xe" for details.
```

OOPS! Temos um problema. Confira o que aconteceu verificando o log de erros do Nginx:

```
# tail /var/log/nginx/error.log -n1
2018/11/18 02:56:10 [emerg] 1850#1850: open() "/etc/nginx/modules-enabled/50-mod-
http-geoip.conf" failed (13: Permission denied) in /etc/nginx/nginx.conf:4
```

Uhm, aparentemente o Nginx precisa acessar o arquivo `/etc/nginx/modules-enabled/50-mod-http-geoip.conf`, mas essa permissão não consta do seu perfil. Note que esse arquivo é um *symlink* para o `/usr/share/nginx/modules-available/mod-http-geoip.conf`:

```
# ls -ld /etc/nginx/modules-enabled/50-mod-http-geoip.conf
lrwxrwxrwx 1 root root 54 nov 18 02:31 /etc/nginx/modules-enabled/50-mod-http-
geoip.conf -> /usr/share/nginx/modules-available/mod-http-geoip.conf
```

Sem problema, vamos adicionar essa permissão:

```
# sed -i '/\/usr\/share\/nginx\/modules-available\/mod-http-echo.conf/a\
\/usr\/share\/nginx\/modules-available\/mod-http-geoip.conf r,'
/etc/apparmor.d/usr.sbin.nginx
```

Veja como ficou o arquivo, agora:

```
# cat /etc/apparmor.d/usr.sbin.nginx
# Last Modified: Sun Nov 18 02:48:57 2018
#include <tunables/global>

/usr/sbin/nginx {
  #include <abstractions/base>
  #include <abstractions/openssl>
  #include <abstractions/ubuntu-browsers.d/plugins-common>

  capability dac_override,

  /data/www/safe/index.html r,
  /etc/nginx/modules-enabled/ r,
  /etc/nginx/nginx.conf r,
  /lib/x86_64-linux-gnu/ld-*.so mr,
  /usr/sbin/nginx mr,
  /usr/share/nginx/modules-available/mod-http-auth-pam.conf r,
  /usr/share/nginx/modules-available/mod-http-dav-ext.conf r,
  /usr/share/nginx/modules-available/mod-http-echo.conf r,
  /usr/share/nginx/modules-available/mod-http-geoip.conf r,
  /var/log/nginx/access.log w,
  /var/log/nginx/error.log w,

}
```

Agora sim! Recarregue o AppArmor e reinicie o Nginx:

```
# systemctl reload apparmor
```

```
# systemctl restart nginx
Job for nginx.service failed because the control process exited with error code.
See "systemctl status nginx.service" and "journalctl -xe" for details.
```

Oh não... o que foi dessa vez? Vamos ver:

```
# tail /var/log/nginx/error.log -n1
2018/11/18 03:03:04 [emerg] 1976#1976: open() "/etc/nginx/modules-enabled/50-mod-
http-image-filter.conf" failed (13:Permission denied) in /etc/nginx/nginx.conf:4
```

Podemos adicionar essa permissão, mas já posso lhe adiantar que teremos OUTRO erro logo a seguir. Que coisa, não?

10. Após algum tempo em tentativas e erros, chegamos a um arquivo de perfil que funciona para o nosso caso. Edite o arquivo `/etc/apparmor.d/usr.sbin.nginx` e deixe seu conteúdo **exatamente** como o que se segue:

```
# Last Modified: Sun Nov 18 03:32:21 2018
#include <tunables/global>

/usr/sbin/nginx {
  #include <abstractions/apache2-common>
  #include <abstractions/base>
  #include <abstractions/nameservice>
  #include <abstractions/openssl>
  #include <abstractions/ubuntu-browsers.d/plugins-common>

  capability dac_override,
  capability setgid,
  capability setuid,

  deny /data/www/unsafe/* r,

  /data/www/safe/* r,
  /etc/group r,
  /etc/nginx/conf.d/ r,
  /etc/nginx/conf.d/apparmor.conf r,
  /etc/nginx/mime.types r,
  /etc/nginx/modules-enabled/ r,
  /etc/nginx/nginx.conf r,
  /etc/nginx/sites-available/default r,
  /etc/nginx/sites-enabled/ r,
  /etc/nsswitch.conf r,
  /etc/passwd r,
  /etc/ssl/openssl.cnf r,
  /lib/x86_64-linux-gnu/ld-*.so mr,
  /run/nginx.pid rw,
  /usr/sbin/nginx mr,
  /usr/share/nginx/modules-available/* r,
  /var/log/nginx/access.log w,
  /var/log/nginx/error.log w,

}
```

Note que há um número significativo de diferenças entre o arquivo acima e o que tínhamos originalmente — você pode imaginar o tempo necessário para construir um perfil desse tipo. E observe que estamos trabalhando aqui no perfil de UMA única aplicação, o servidor web Nginx. Imagine se tivéssemos um número maior de programas, ou requerimentos de acesso mais complexos.

Observe ainda que estamos detalhando políticas expressas de acesso para as pastas *safe* e *unsafe*:

- Na linha `/data/www/safe/* r,`, o acesso de leitura a todos os arquivos dentro da pasta *safe* é autorizado ao Nginx.
- Já na linha `deny /data/www/unsafe/* r,`, o mesmo tipo de acesso na pasta *unsafe* é negado.

11. Enfim, reescrito o perfil acima, recarregue o AppArmor e reinicie o Nginx:

```
# systemctl reload apparmor
```

```
# systemctl restart nginx
```

Sucesso! Vamos ver se nossa proteção funcionou: retorne ao navegador e carregue a URL `http://10.0.42.12:8080/safe/`:

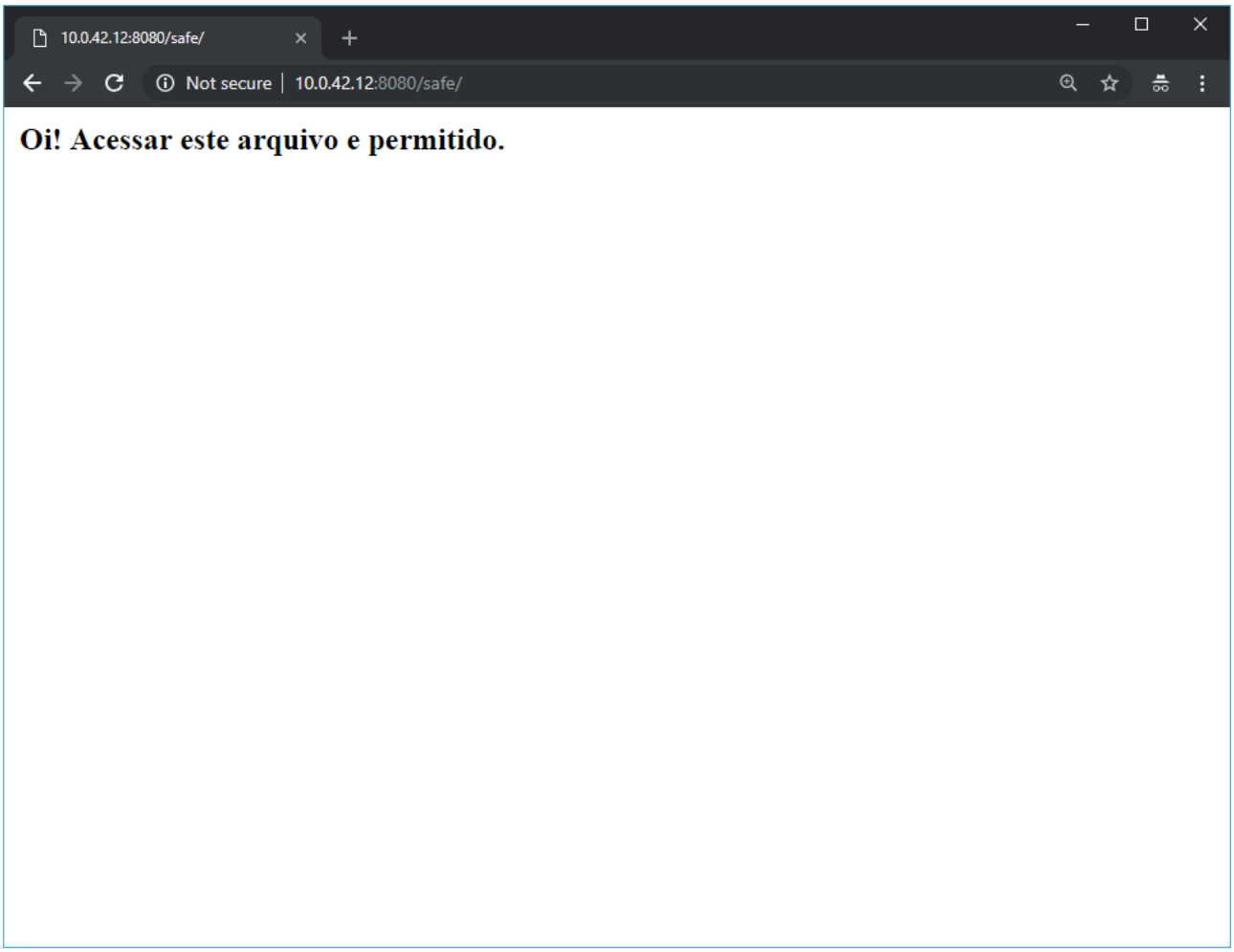

Figura 69. Acesso autorizado à área safe

Perfeito, o acesso foi autorizado. E quanto a `http://10.0.42.12:8080/unsafe/`?

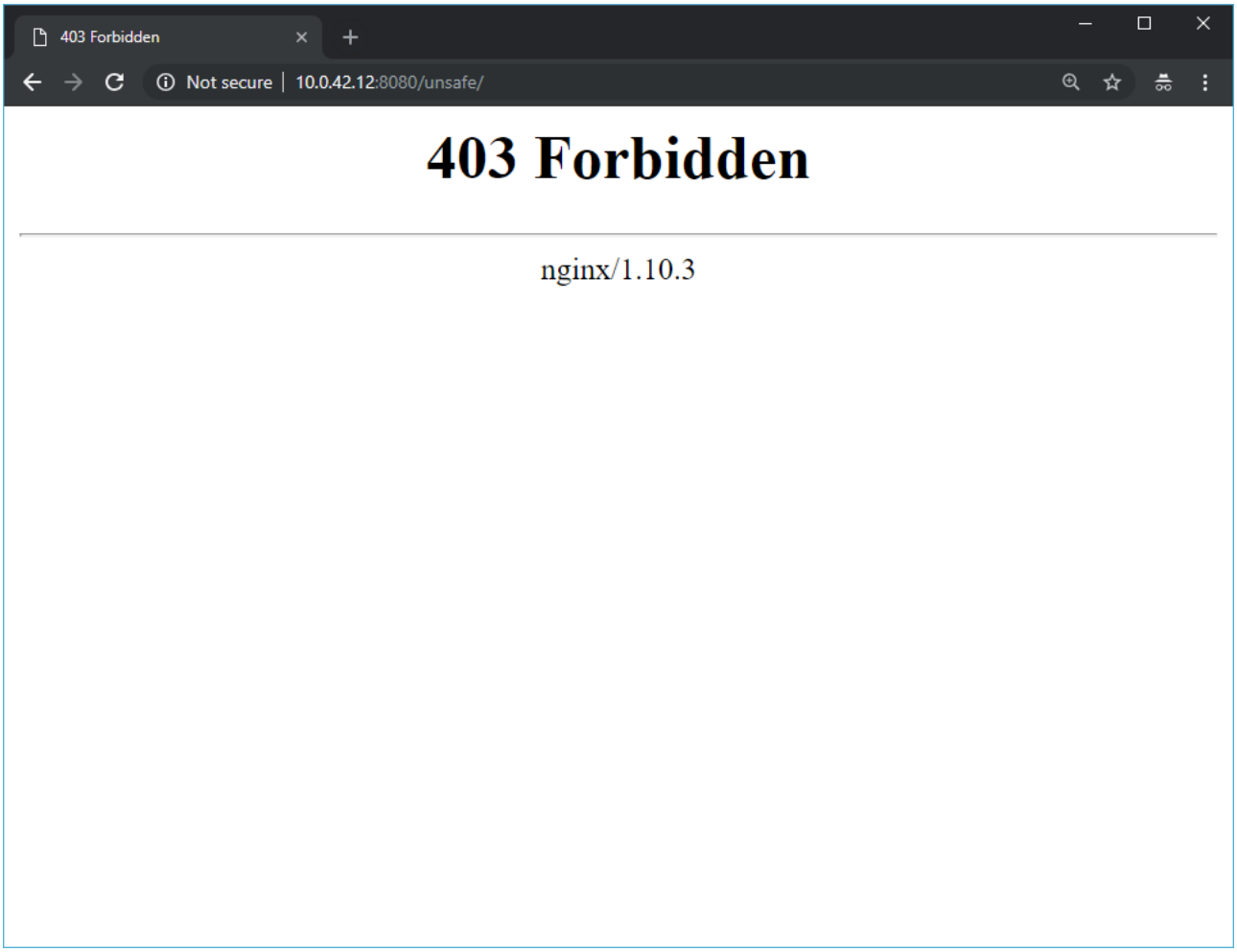

Figura 70. Acesso negado à área unsafe

Como esperado, o acesso foi negado pelo AppArmor. Observe a mensagem de erro no log do Nginx:

```
# tail -f -n0 /var/log/nginx/error.log
2018/11/18 03:36:10 [error] 4035#4035: *2 open() "/data/www/unsafe/index.html"
failed (13: Permission denied), client: 10.0.42.254, server: , request: "GET
/unsafe/ HTTP/1.1", host: "10.0.42.12:8080"
```

12. Como objetivado, conseguimos usar o AppArmor para construir um perfil de segurança específico para o servidor web Nginx, e testar esse perfil com sucesso. Fica claro, também, que a construção de perfis por meio de ferramentas MAC (como o AppArmor ou o SELinux), apesar de se apresentar como uma ferramenta muito poderosa, também traz consigo um custo administrativo bastante alto.

    Leve isso em consideração ao implantar esquemas de segurança em seus servidores. É possível que haja um ganho de segurança significativo por meio da construção de perfis de segurança, mas convém analisar a criticidade dos sistemas-alvo para determinar se o custo administrativo compensa, de fato, o risco que se incorre com a falta da proteção.