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)`.

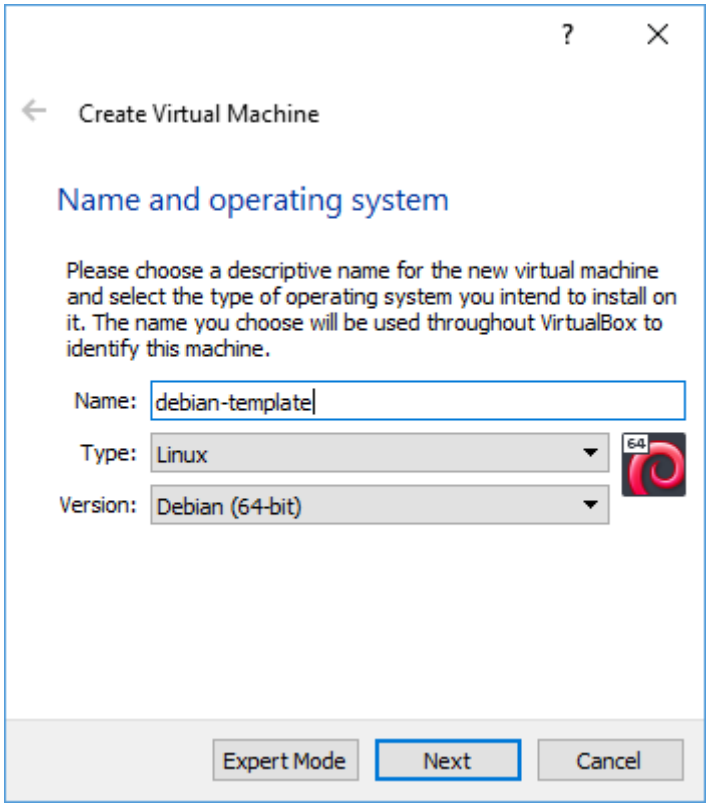

Figura 1. Criação de nova VM, parte 1

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*.

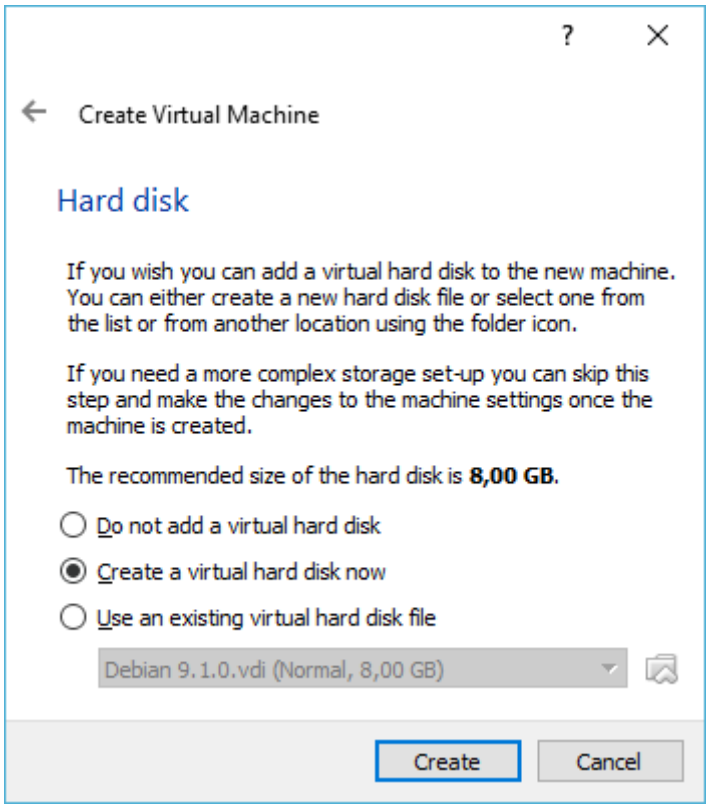

Figura 2. Criação de nova VM, parte 2

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*.

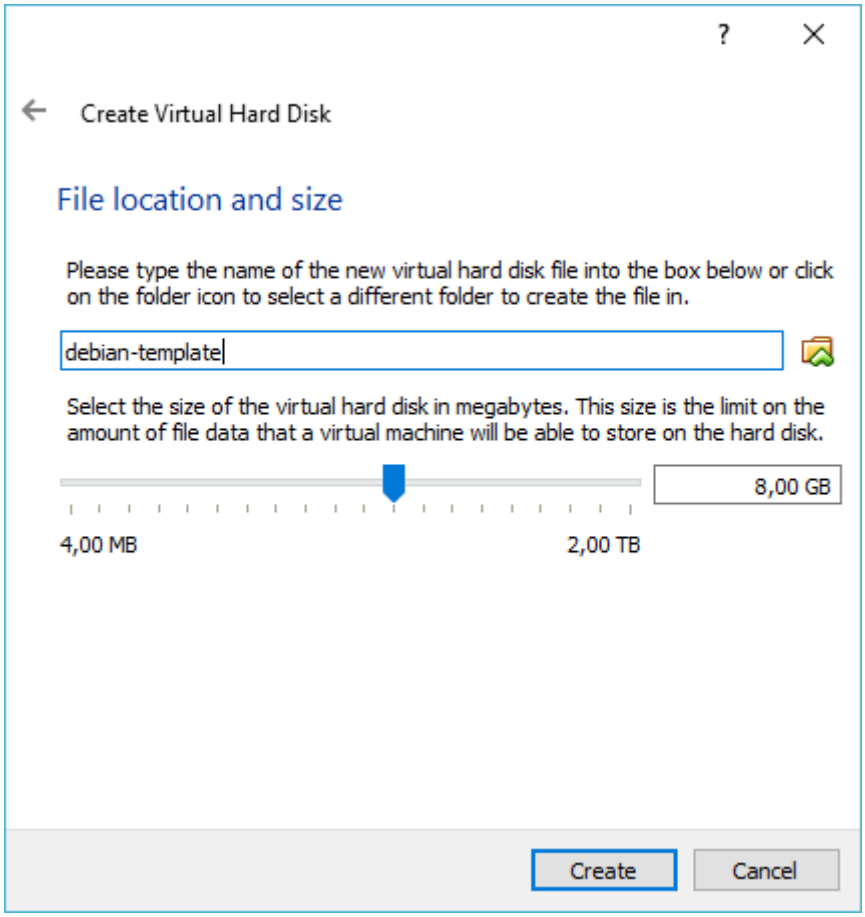

Figura 3. Criação de nova VM, parte 3

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.

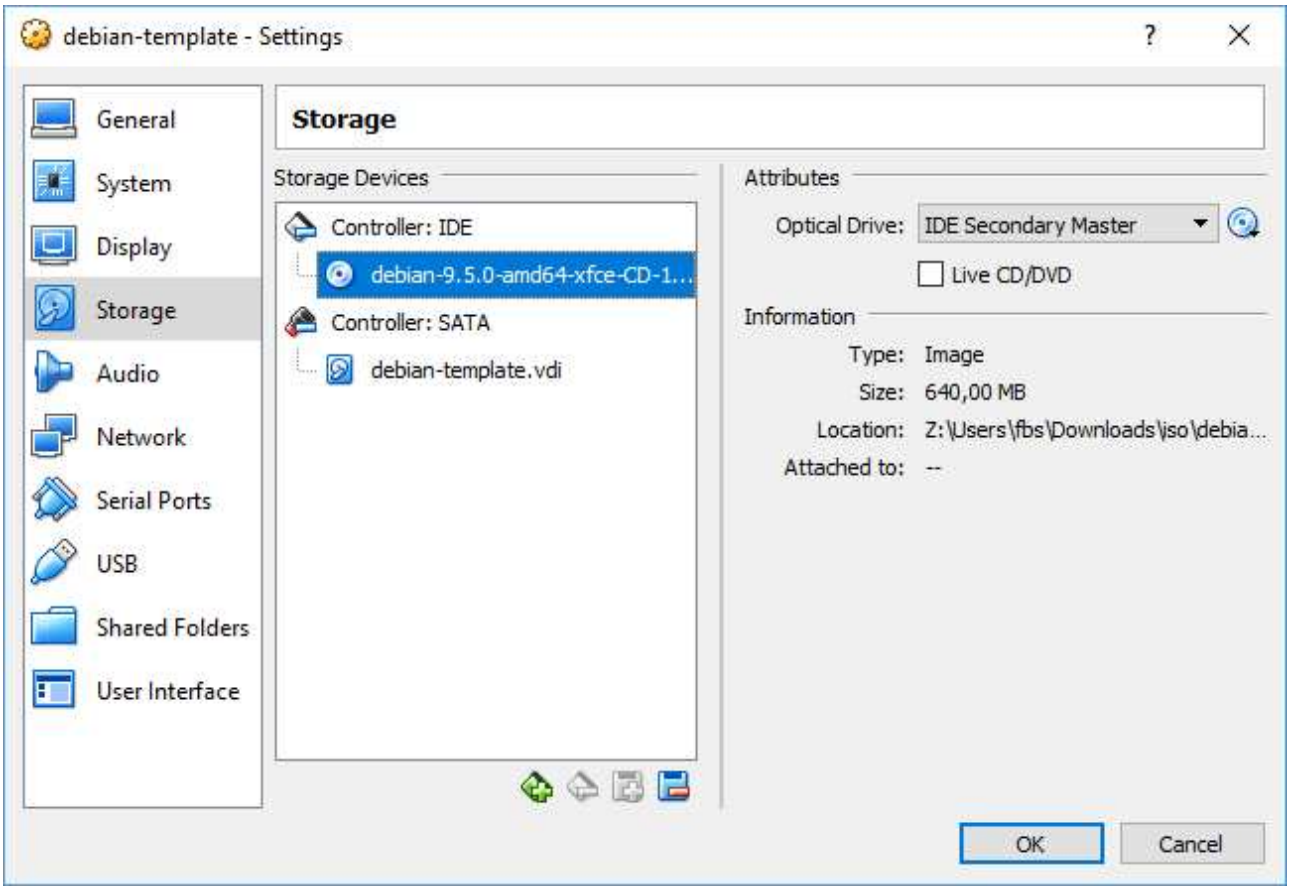

Figura 4. Configuração de nova VM

Em *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.

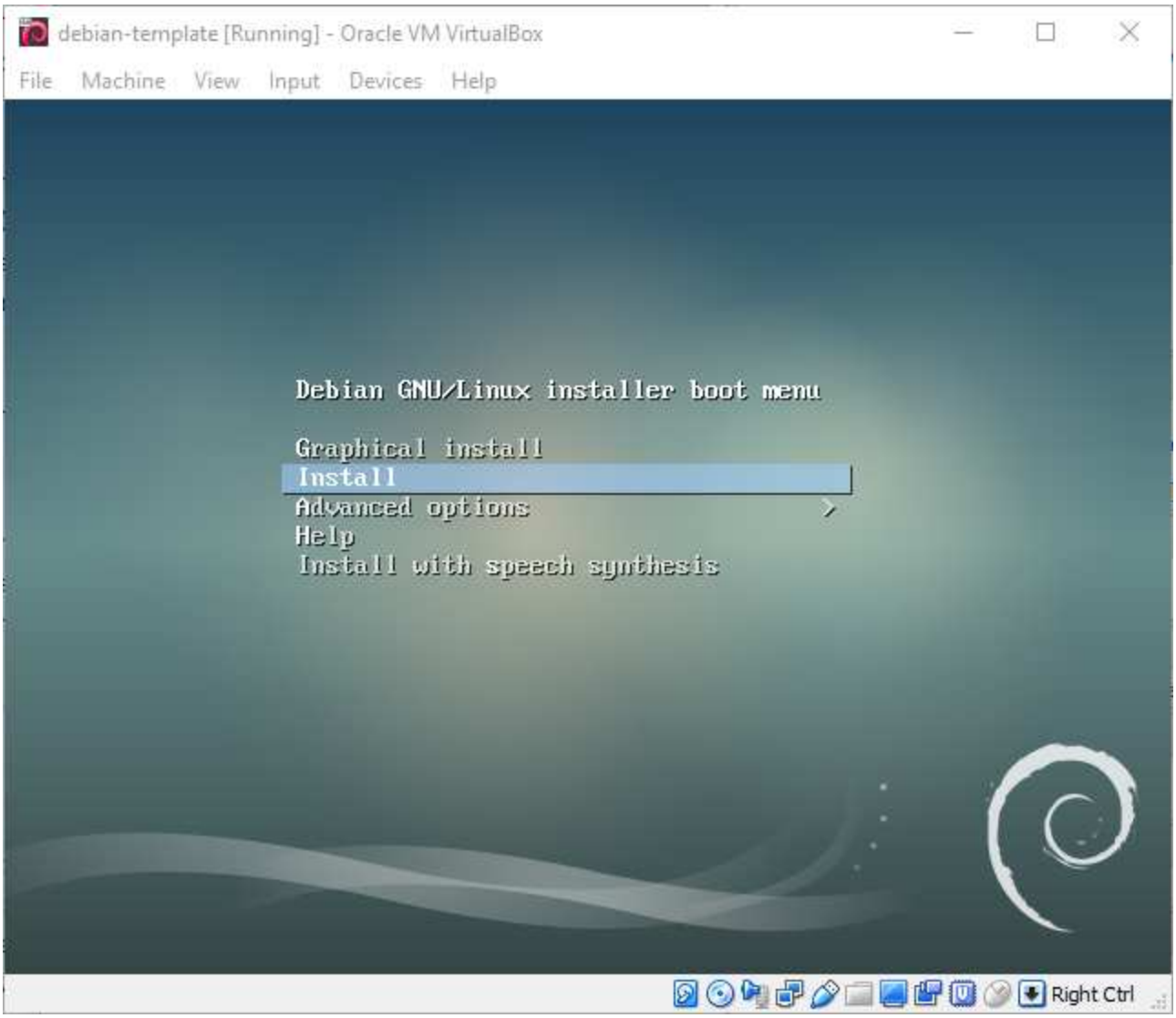

Figura 5. Instalação do Debian Linux, parte 1

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`.

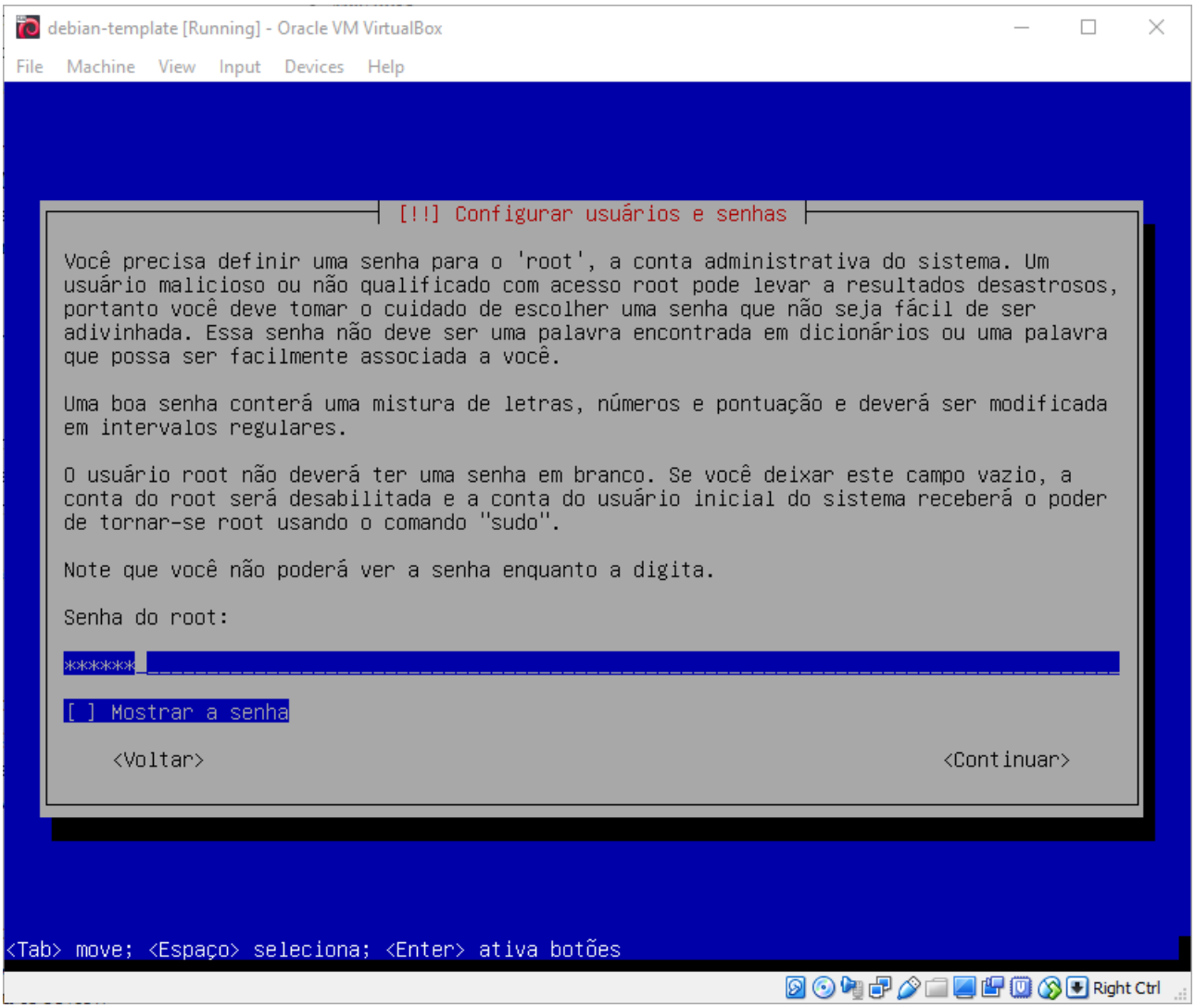

Figura 6. Instalação do Debian Linux, parte 2

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:

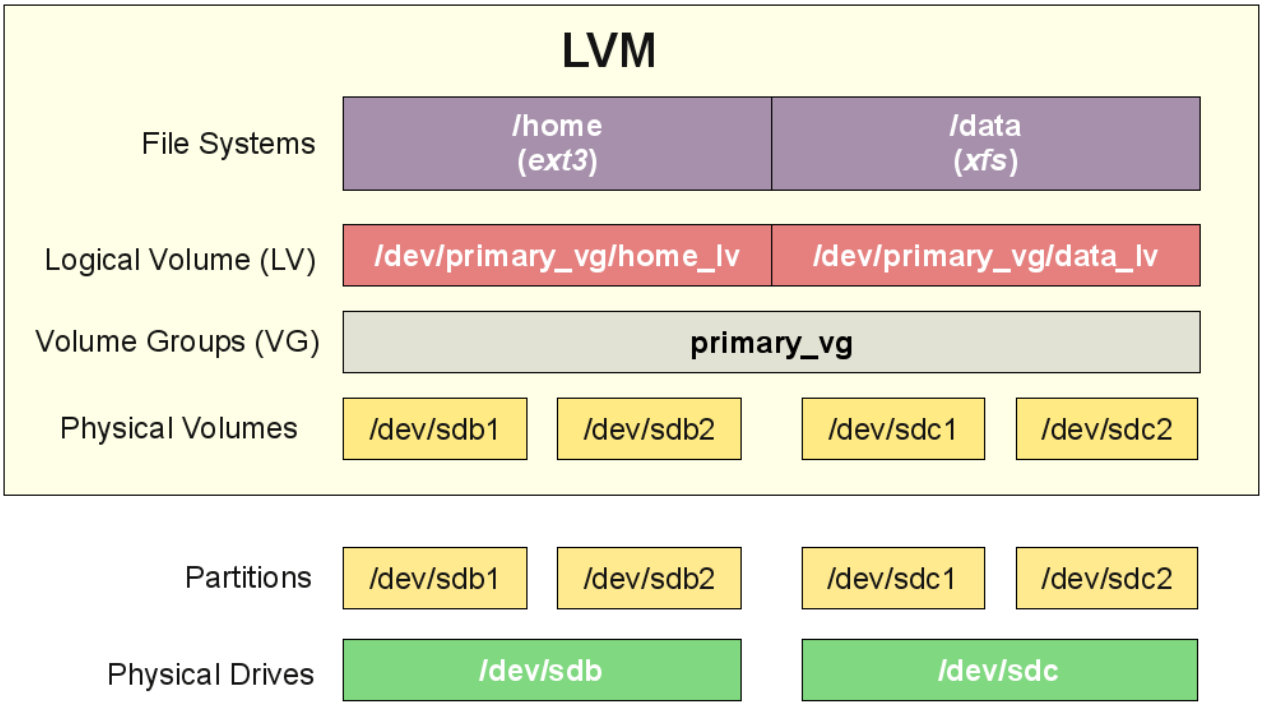

Figura 7. Organização do LVM

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*.

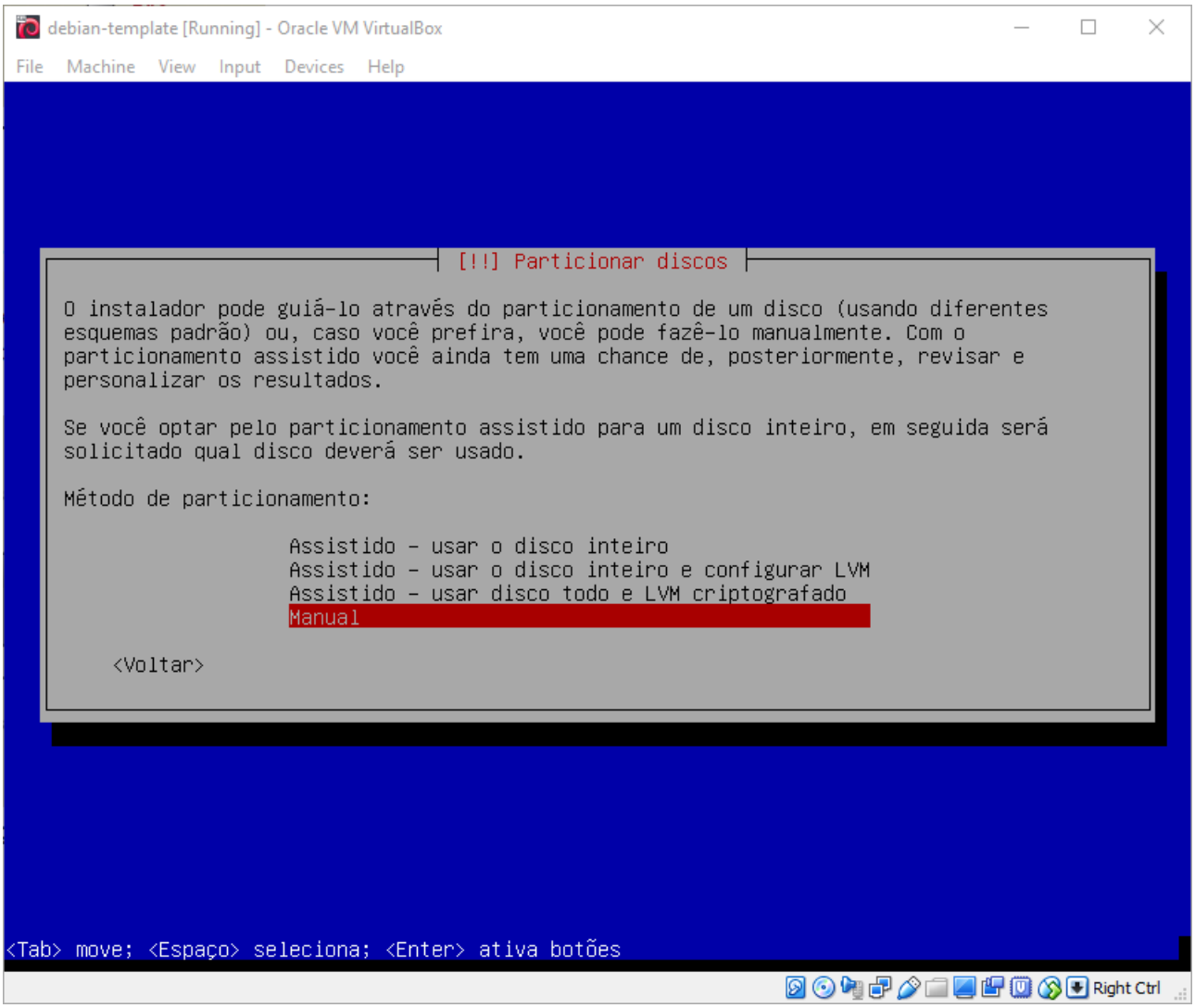

Figura 8. Instalação do Debian Linux, parte 3

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.

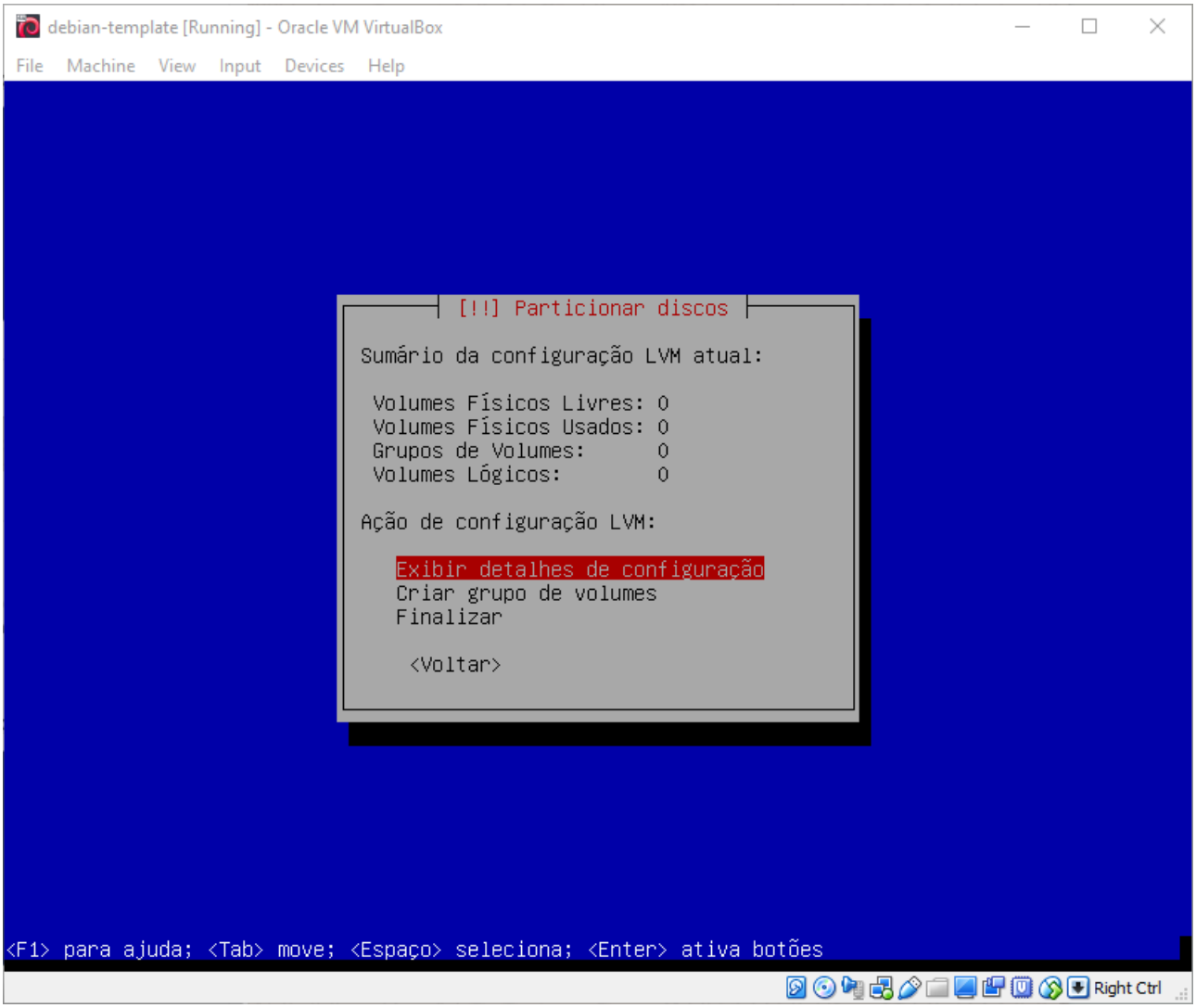

Figura 9. Instalação do Debian Linux, parte 4

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:

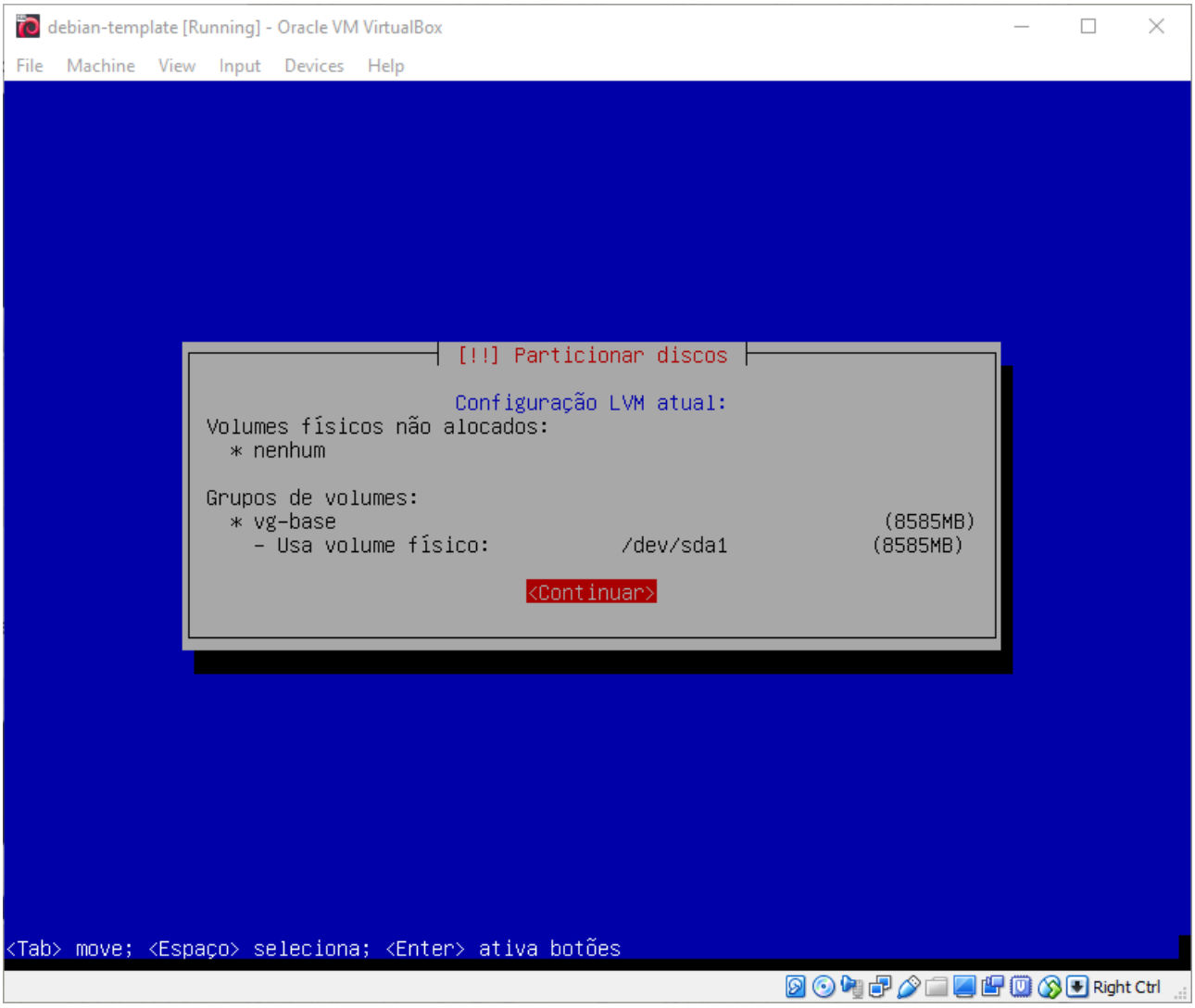

Figura 10. Instalação do Debian Linux, parte 5

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 512 MB para essa área.
    - `lv-root`: LV quer irá armazenar a raiz do sistema, `/`, com tamanho de 1280 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 + 512 + 1280] = 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:

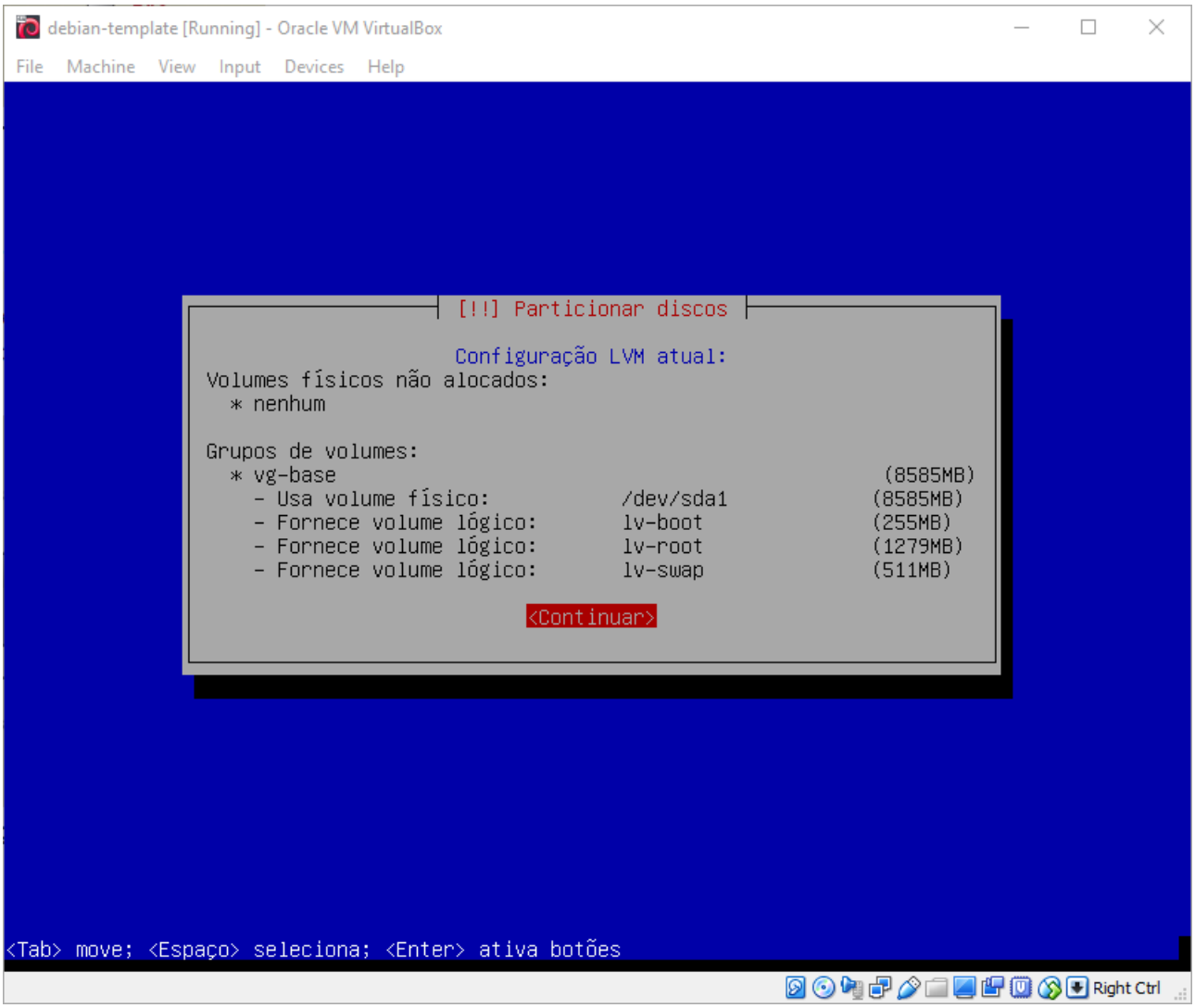

Figura 11. Instalação do Debian Linux, parte 6

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:

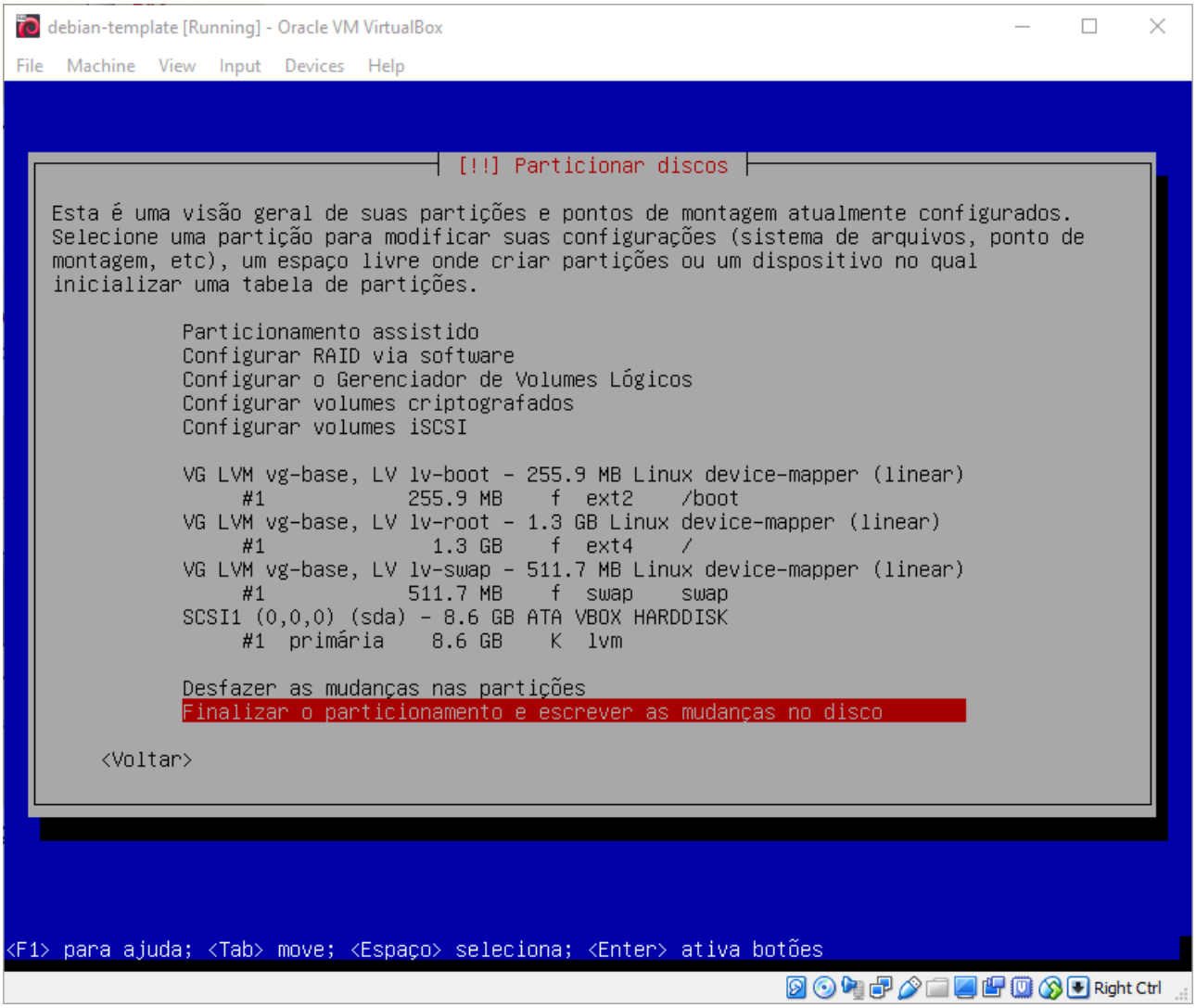

Figura 12. Instalação do Debian Linux, parte 7

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*.

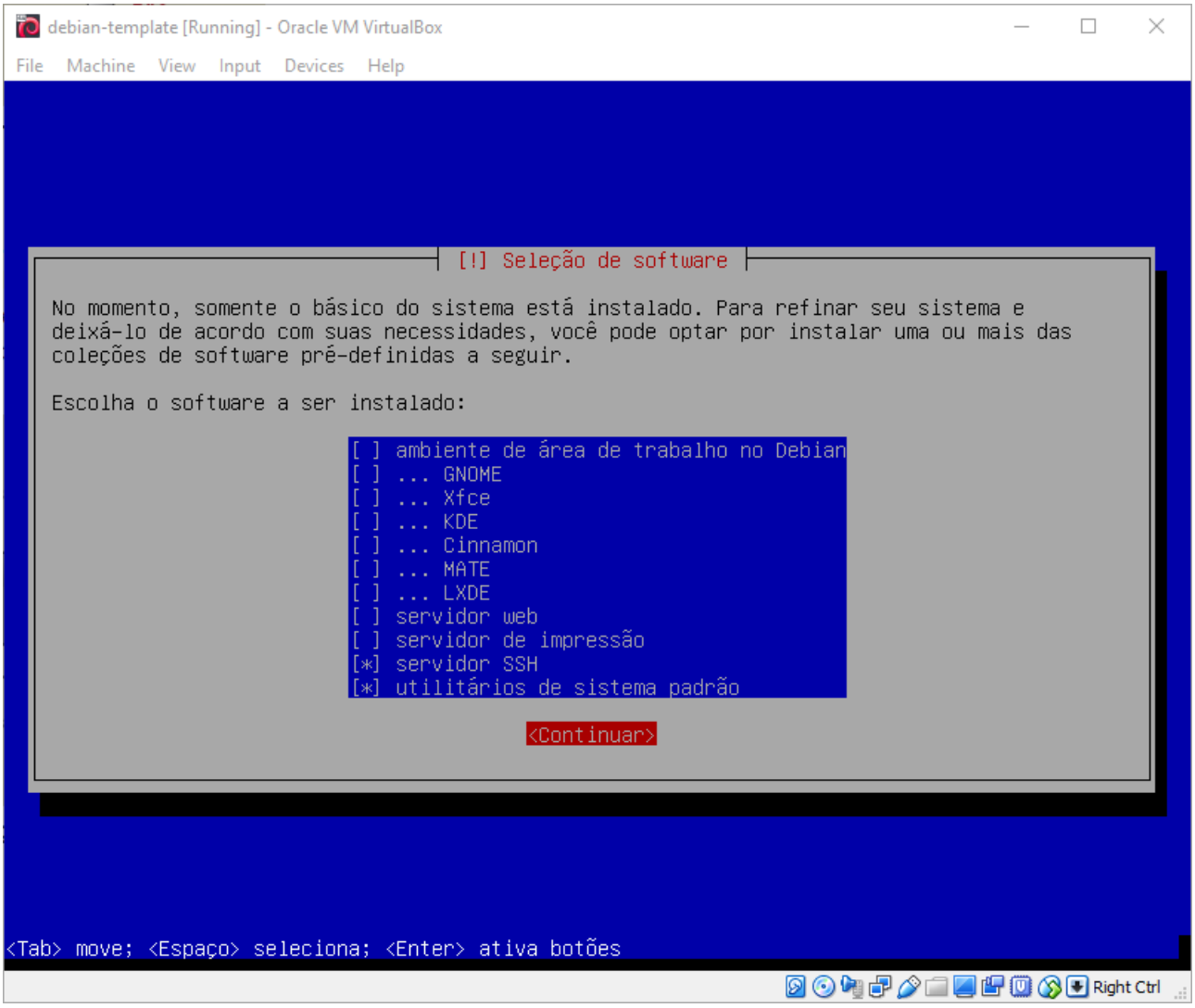

Figura 13. Instalação do Debian Linux, parte 8

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:

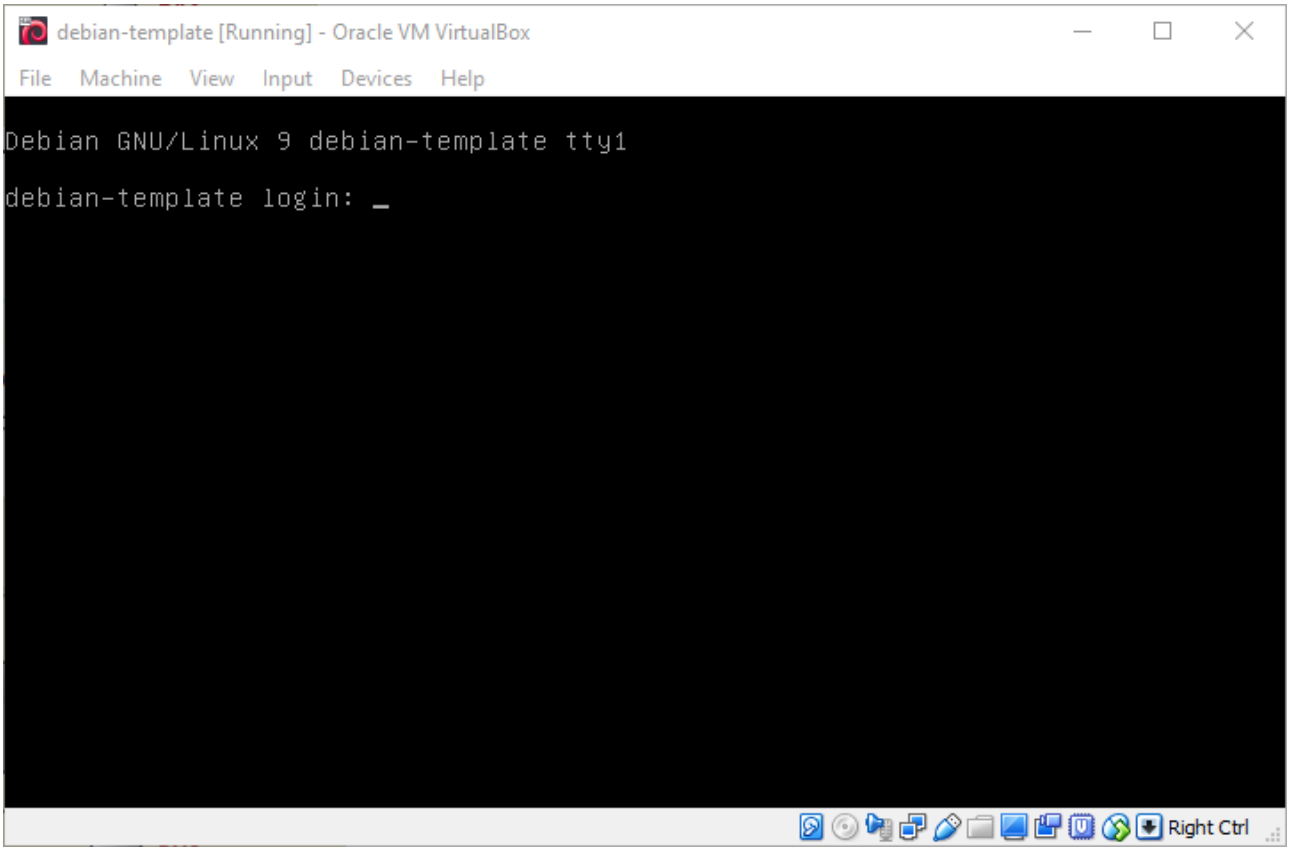

Figura 14. Instalação do Debian Linux, concluída

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 192.168.29.100/24 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. 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://security.debian.org/debian-security stretch/updates main contrib non-
free
deb http://ftp.br.debian.org/debian/ stretch-updates main contrib non-free
```

5. 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.

   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
   ```

6. 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 o `rsync`, que será usado a seguir. Execute:

   ```
   # apt-get install rsync
   ```

7. 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.
```

8. 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/changehost.sh`, com o seguinte conteúdo:

```
#!/bin/bash

[ -z $1 ] && { echo "Usage: $0 NEWHOSTNAME"; exit 1; }

sed -i "s/debian-template/$1/g" /etc/hosts
sed -i "s/debian-template/$1/g" /etc/hostname

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

rm -f /etc/ssh/ssh_host_* 2> /dev/null
dpkg-reconfigure openssh-server &> /dev/null
```

   O *script* acima 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                488,00 MiB
  --- Logical volume ---
  LV Path                /dev/vg-base/lv-root
  LV Name                lv-root
  LV Size                1,19 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               t42fDH-Dnqm-m1b5-6Edx-SSqV-stH8-uDi3IZ
```

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               jRYDou-dvIq-HRZw-llOL-JGun-JZvP-4fAs7Q
```

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                488,00 MiB
  --- Logical volume ---
  LV Path                /dev/vg-base/lv-root
  LV Name                lv-root
  LV Size                1,19 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.

Por 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/syncdirs.sh`, com o seguinte conteúdo:

```
#!/bin/bash

for d in tmp var usr; do
  [ -d /mnt/${d} ] || mkdir /mnt/${d}
  mount /dev/mapper/vg--base-lv--${d} /mnt/${d}
  rsync -av /${d}/ /mnt/${d}
  umount /mnt/${d}
  rmdir /mnt/${d}
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 ~/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
173     /var
434     /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--usr on /usr type ext4 (rw,nodev,relatime,data=ordered)
/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)
```

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   178       1210  13% /var
/dev/mapper/vg--base-lv--usr         4059   457       3377  12% /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        1169   863        230  79% /
```

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
```

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

```
root@debian-template:~# 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         1169   256        837  24% /
```

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:

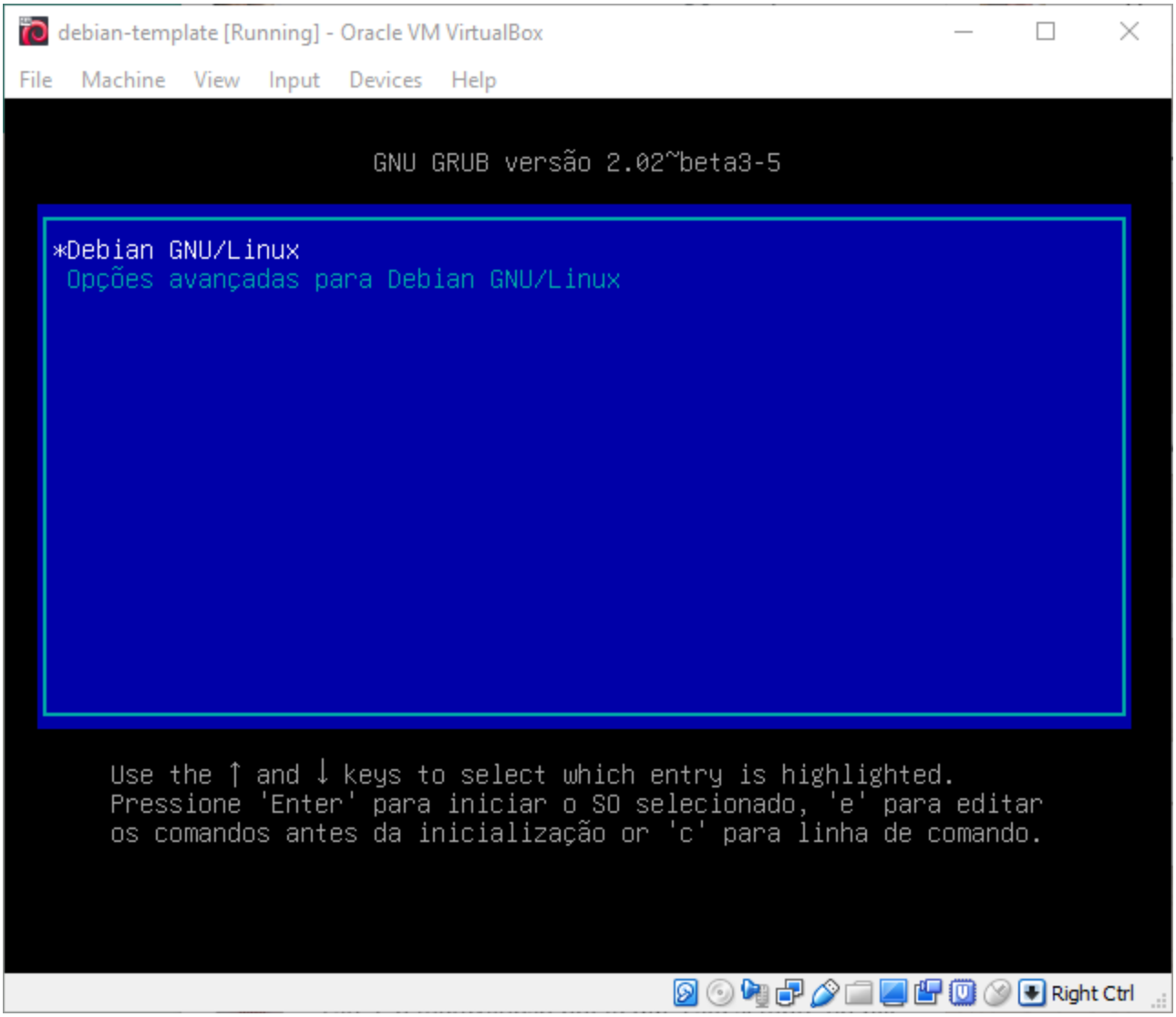

Figura 15. Edição de opções no GRUB

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

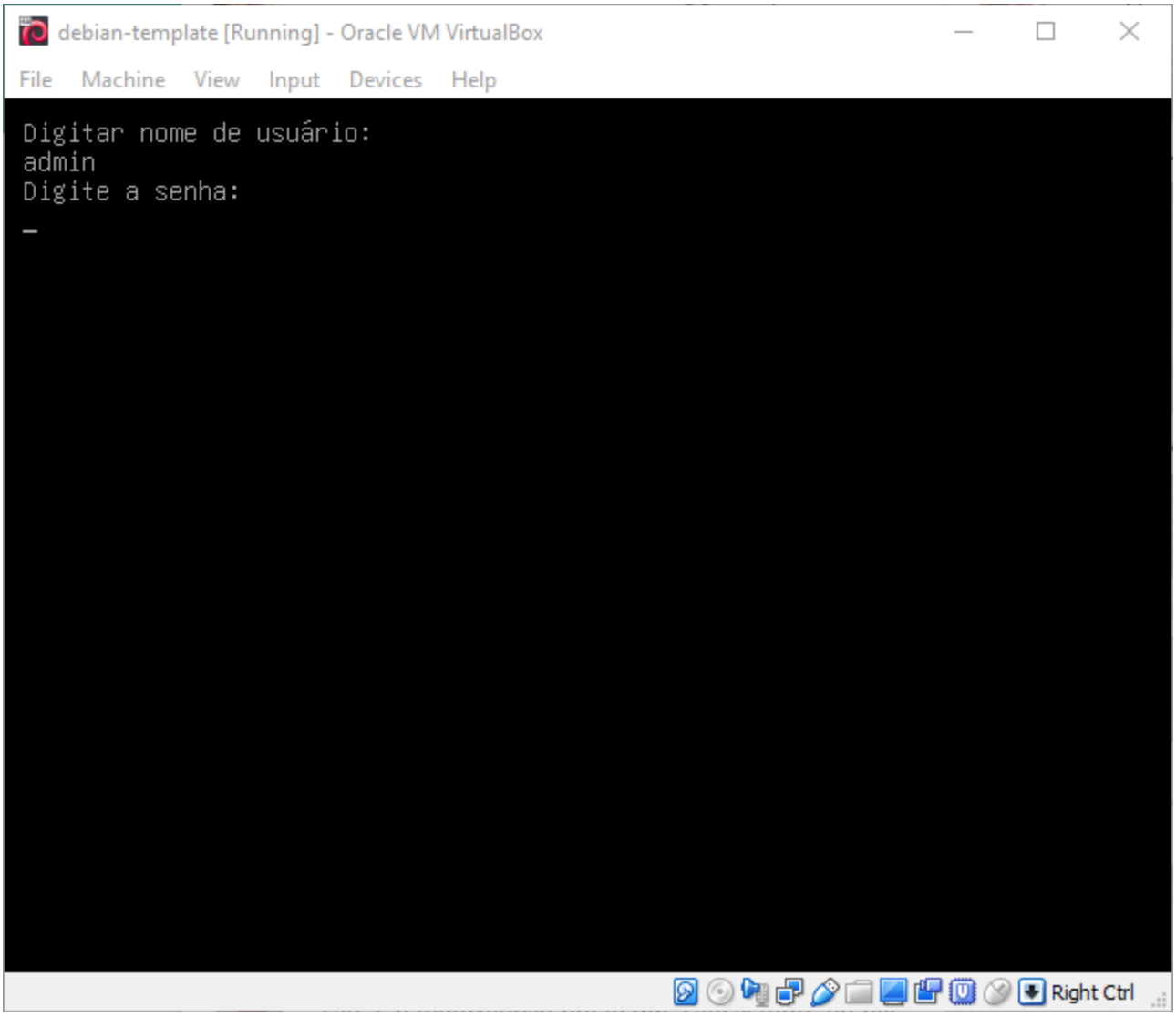

Figura 16. Inserção de usuário/senha no GRUB

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

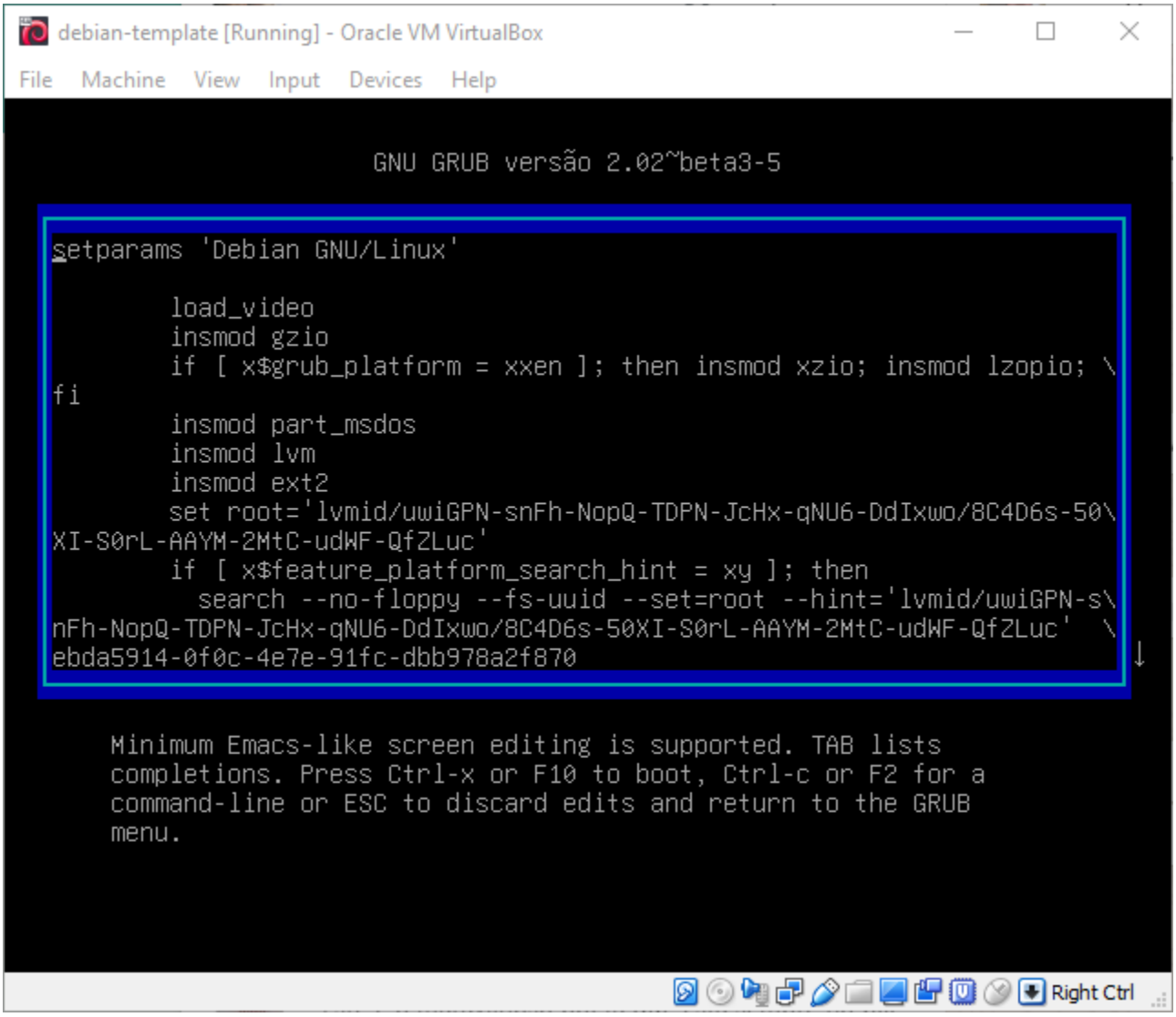

Figura 17. Edição de opções de boot no GRUB

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.

```
# 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
```

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.

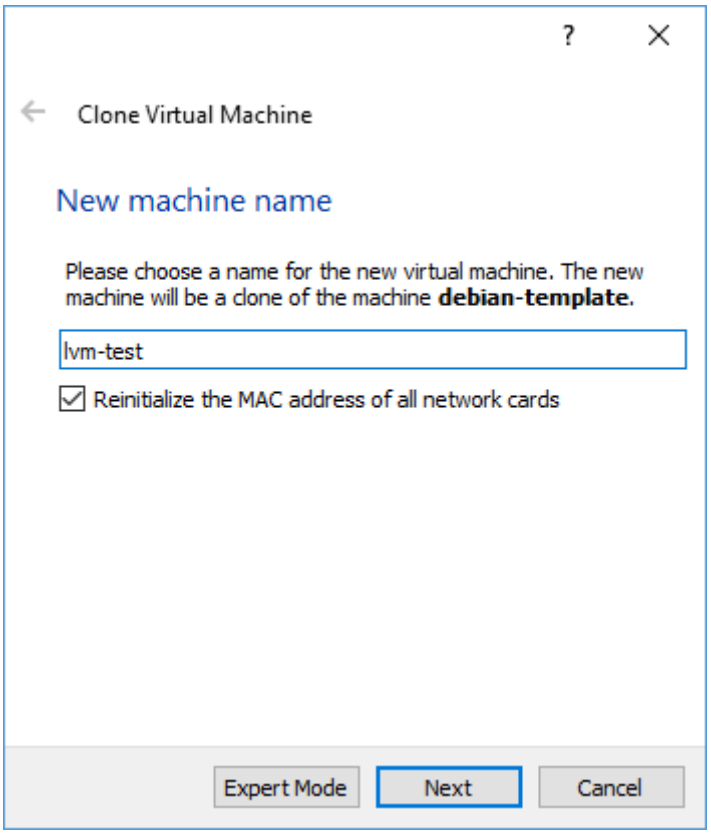

Figura 18. Clonagem de máquinas virtuais no Virtualbox

   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:

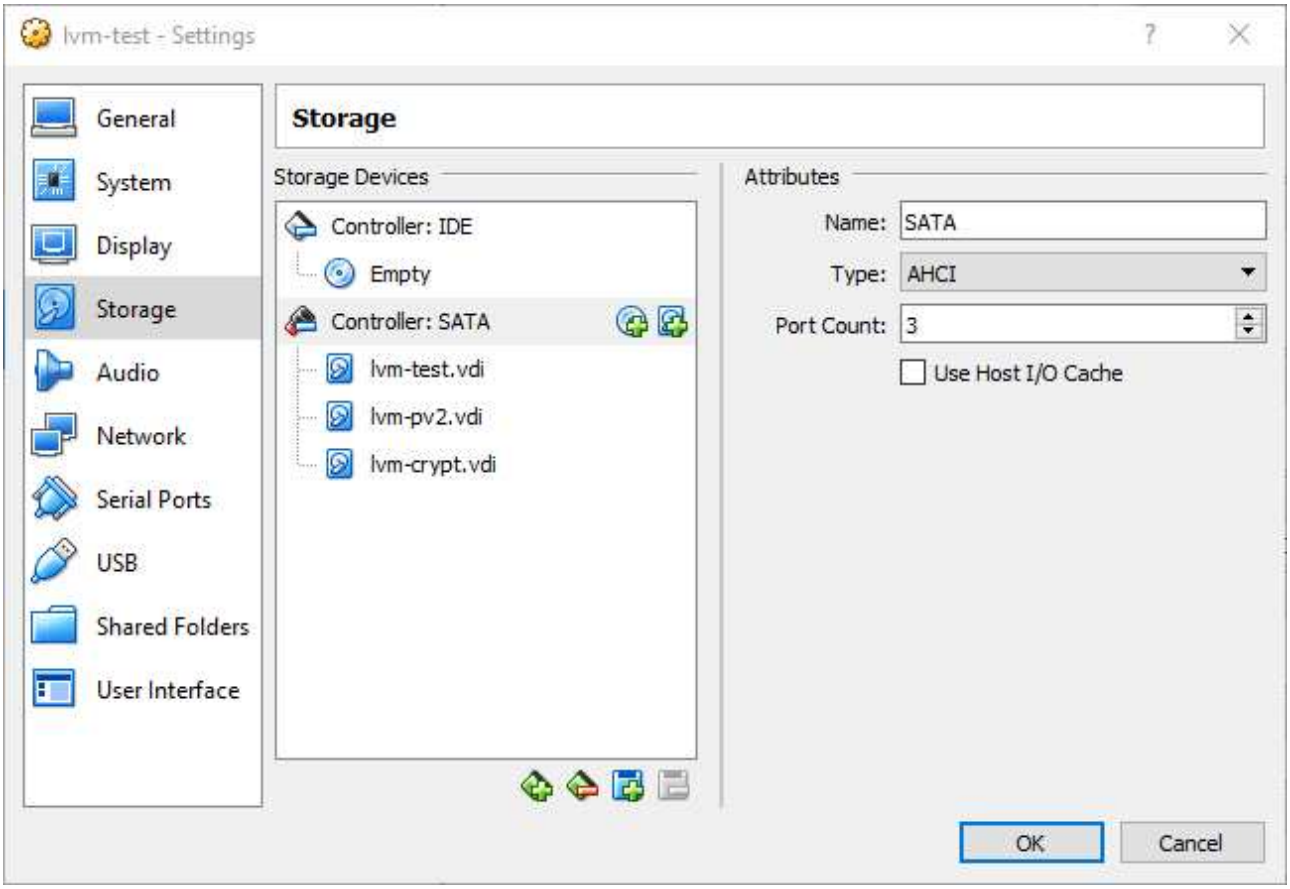

Figura 19. Discos adicionados à máquina lvm-test

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

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

```
# hostname
debian-template
```

```
# bash ~/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               n3dXDL-gCma-P258-pl31-Ow9A-Spcp-2mjsCG
```

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:

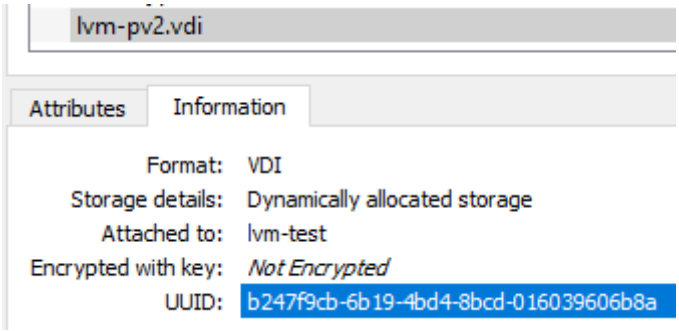

Figura 20. Identificando o UUID de um disco no Virtualbox

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.

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:
```

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
```

Sessão 2: A definir

Sessão 3: A definir

Sessão 4: A definir

Sessão 5: A definir

Sessão 6: A definir

Sessão 7: A definir

Sessão 8: A definir

Sessão 9: A definir

Sessão 10: A definir