How to Install Ubuntu Server on VirtualBox

11 Dec 2019

In this post I'll show you how to install Ubuntu 18.04.3 LTS (Bionic Beaver) on Oracle's VirtualBox. I'll also demonstrate how to connect to the Ubuntu instance via SSH. This will form the basis for a second tutorial that will walk through installing and configuring Ruby on Rails on an Ubuntu server.

Let's get started!

What Is VirtualBox?

VirtualBox is a software virtualization package that you can install on your operating system (just as you would a normal program). It supports the creation and management of virtual machines into which you can install a second operating system.

In VirtualBox terminology, the operating system on which you install VirtualBox (i.e. your regular OS) is called the *host*. The operating system you install within VirtualBox (i.e. inside the virtual machine) is called the *guest*.

For this tutorial, I'll be using Linux Mint 19.2 as the host OS, but there's no reason you couldn't use a different Linux distro, or macOS, or Windows (if you're so inclined).

Install VirtualBox

The first thing to do is to get VirtualBox installed. I'll not go into much detail here, as there are comprehensive instructions for all of the main operating systems on the project's homepage.

Personally, I downloaded and installed the deb package for Ubuntu 18.04 / 18.10 / 19.04. This is because the VirtualBox version in the Mint repos is slightly outdated and I wanted to be running the latest version.

Download Ubuntu Server

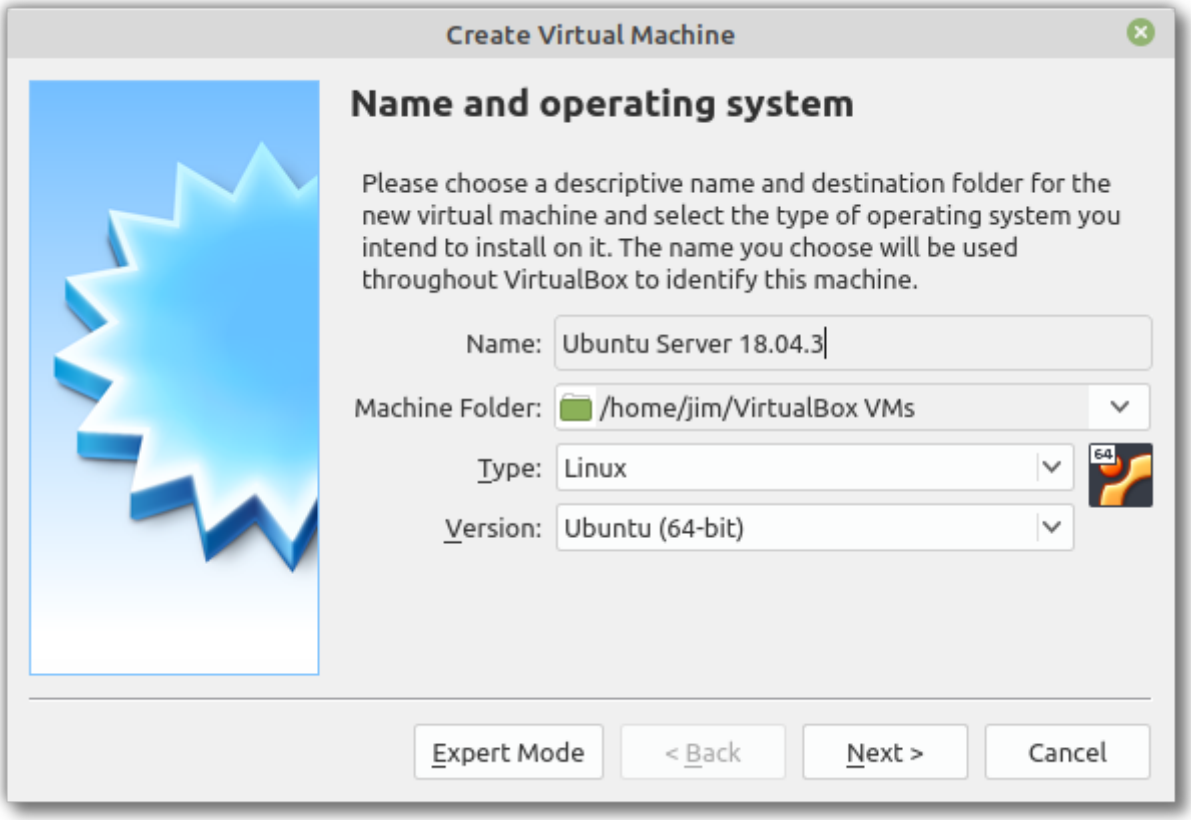
The next thing to do is to grab a copy of Ubuntu Server. You can do this from their download page. This will download a 889MB iso file to your PC.

At the time of writing the current LTS version is Ubuntu Server 18.04.3 and this is what I'll be using. It's supported until April 2023 and is available as 64-bit only.

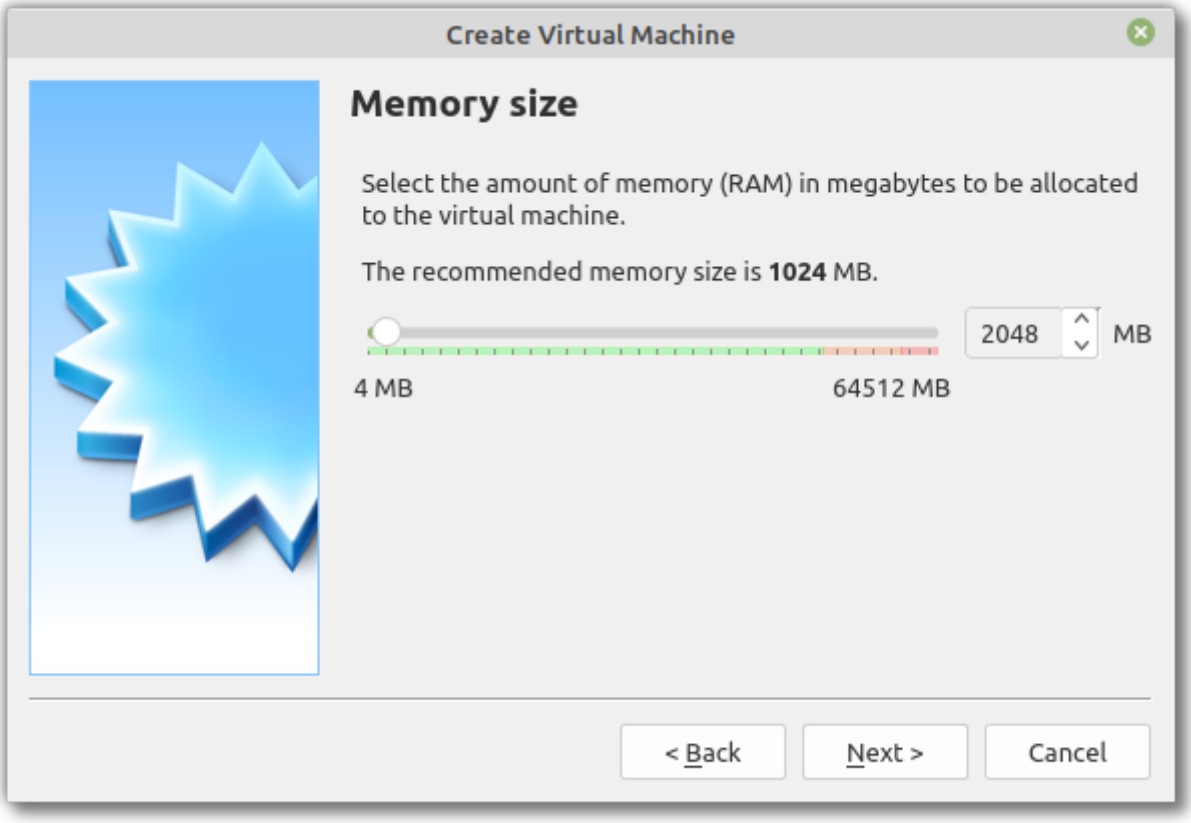
Create a New Virtual Machine

Start up VirtualBox. This should open the VirtualBox Manager, the interface from which you will administer all of your virtual machines.

Next Click on *New* (in the top right of the VirtualBox Manager), give your virtual machine a name and the two drop down menus should automatically update.

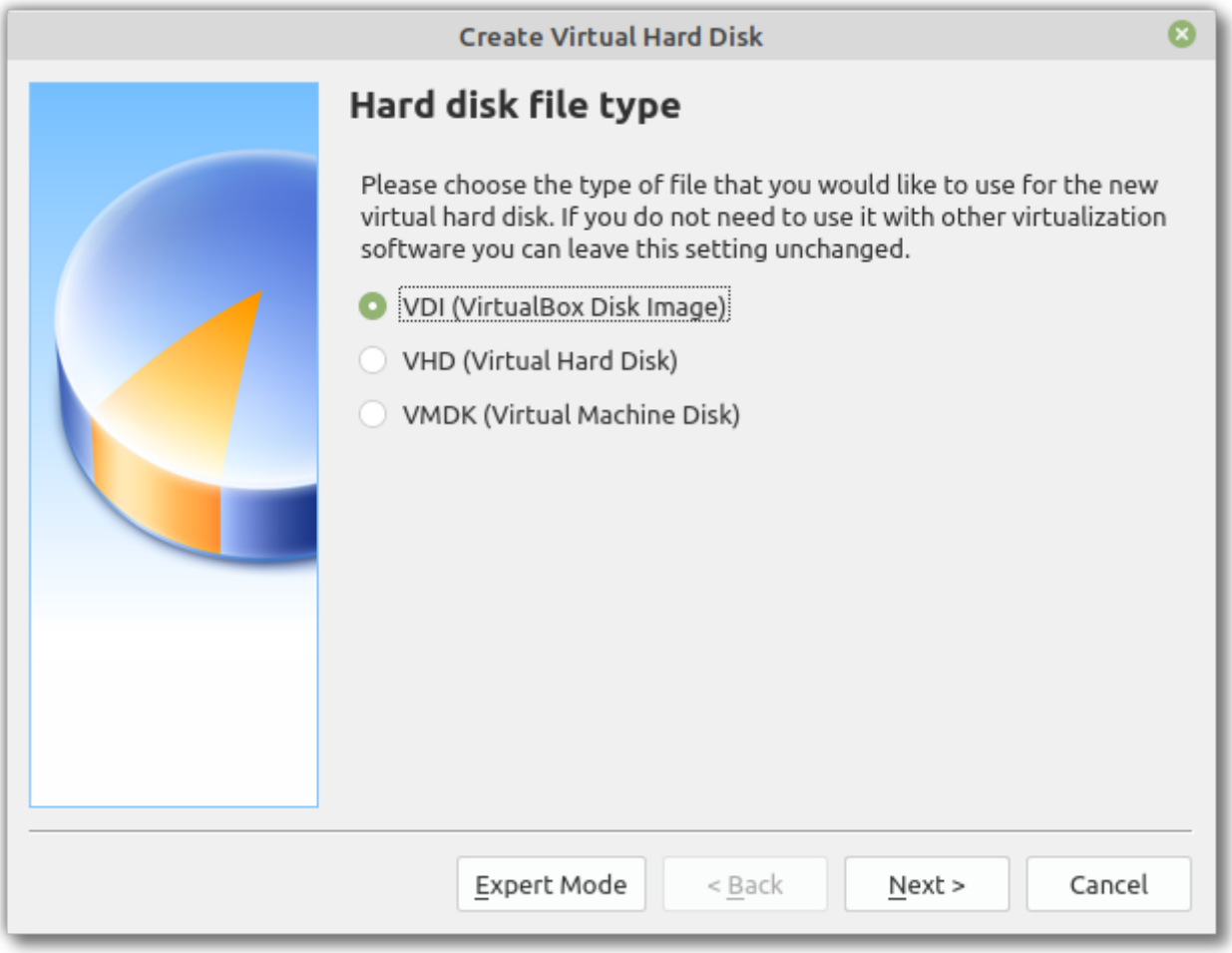


Click *Next*. The wizard will now ask you to select the amount of memory (RAM) in megabytes to be allocated to the virtual machine. I chose 2GB (2048 megabytes).

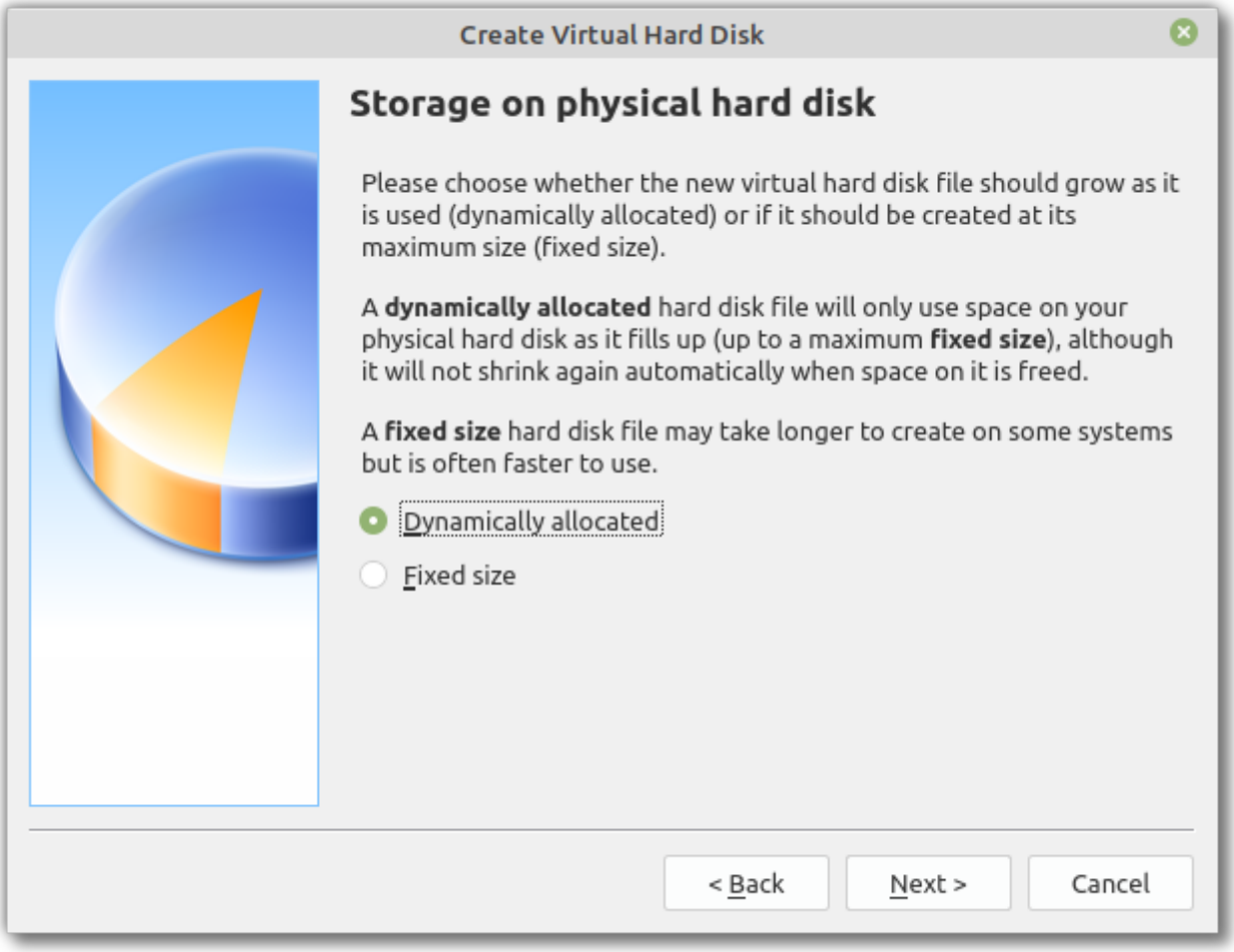


Click *Next* and you will be prompted to add a virtual hard disk to the new machine. Make sure that *Create a virtual hard disk now* is selected, then press *Create*.

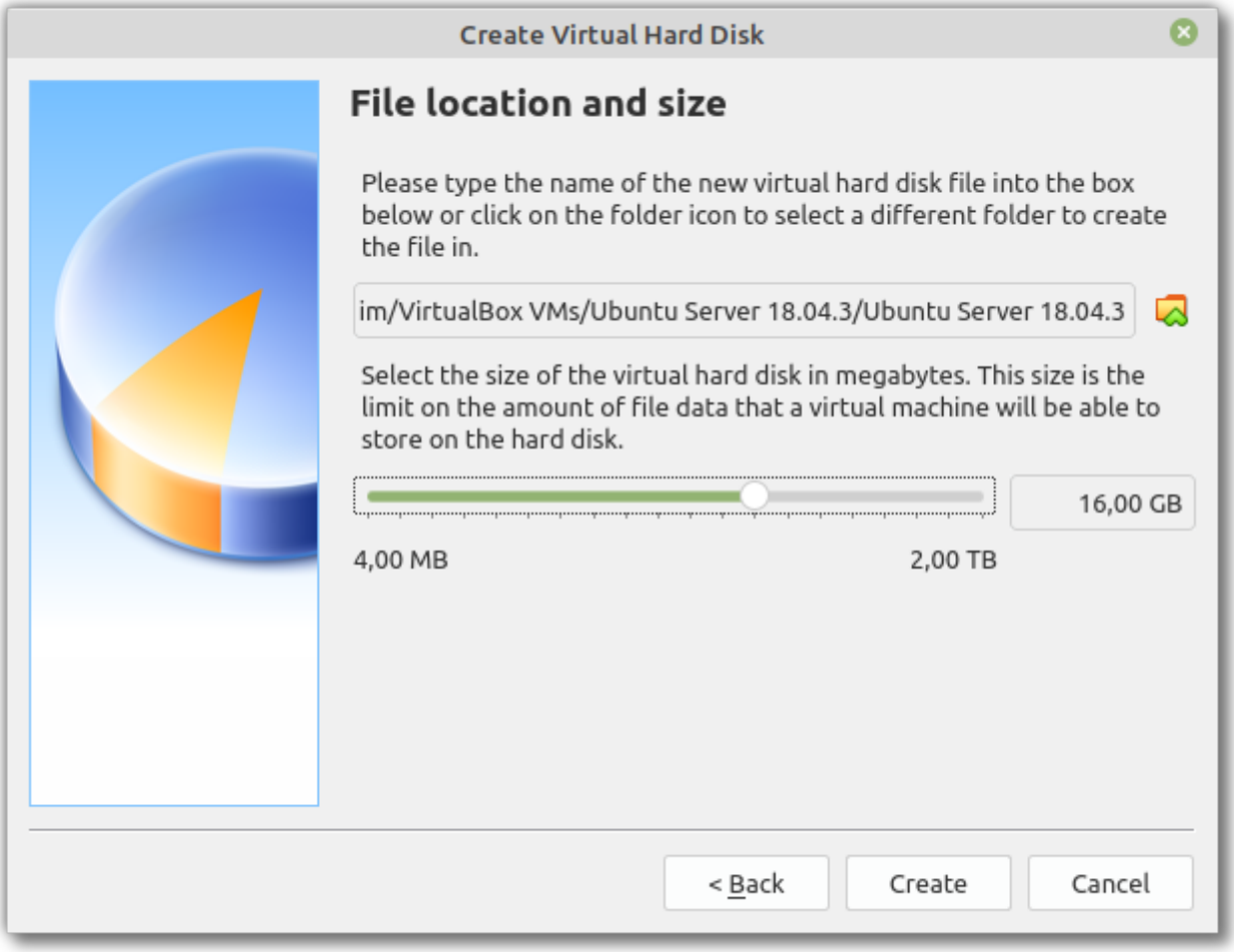
Now we need to choose the file type for the new virtual hard disk. Make sure that *VDI (VirtualBox Disk Image)* is checked and press *Next*.



On the next screen you will be asked whether the new virtual hard disk should grow as it is used (dynamically allocated) or if it should be created at its maximum size. Make sure that *dynamically allocated* is selected, then click *Next*.



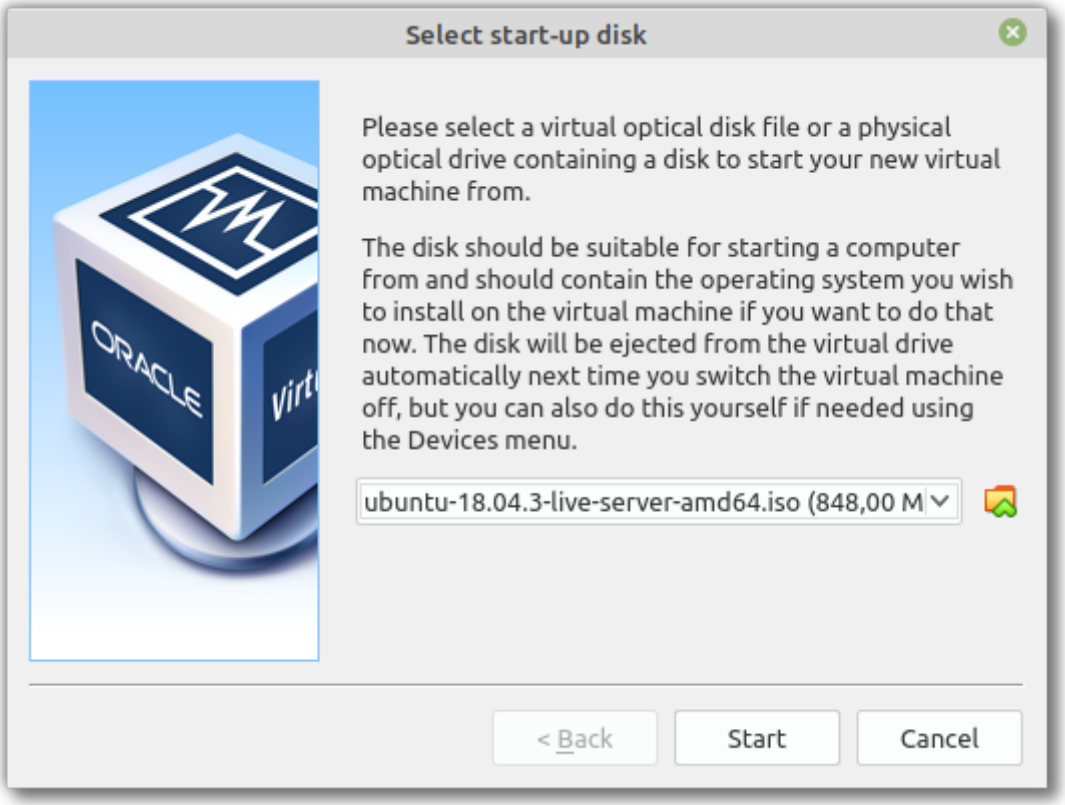
Finally, select the size of the virtual hard disk in megabytes. The default size of 10GB should be plenty, but feel free to increase this as you see fit. Then click *Create*.



The hard disk should now be created and after a short while you should find yourself back in the VirtualBox Manager. You should be able to see your newly created virtual machine listed on the left.

Install Ubuntu Server in the Virtual Machine

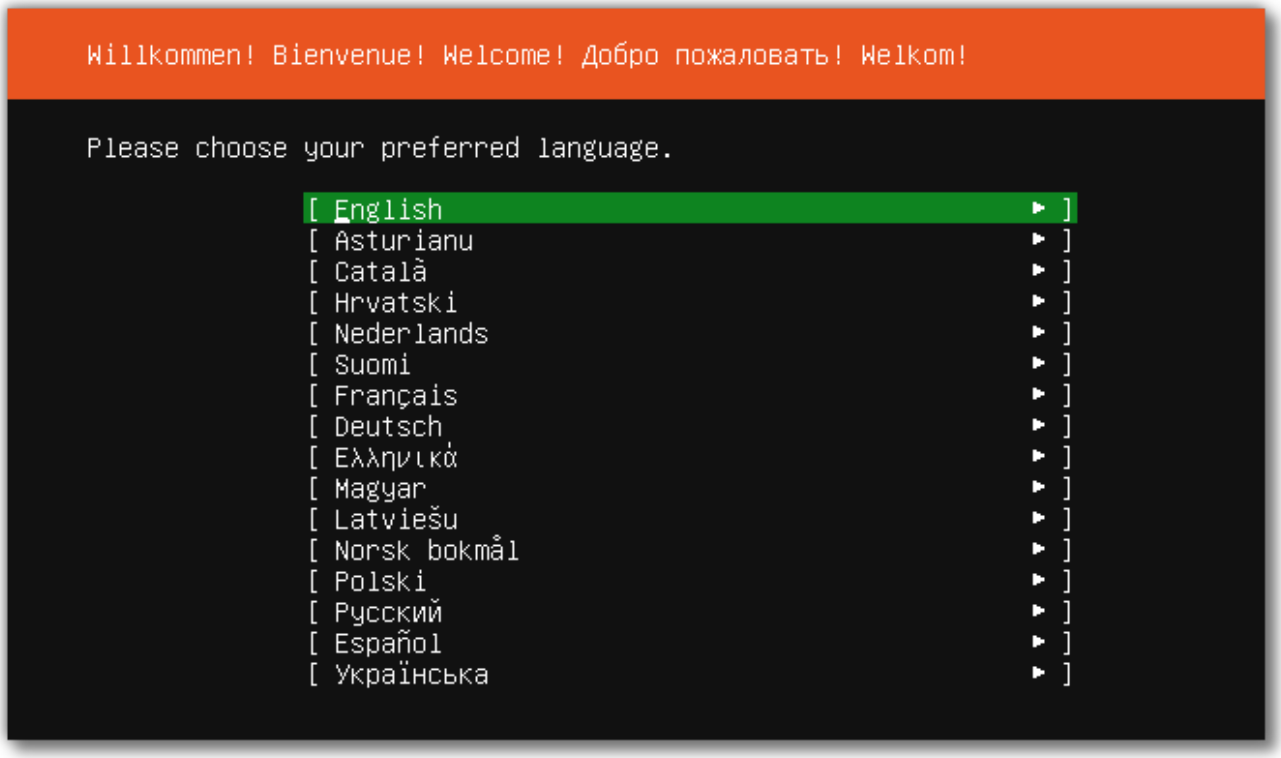
Make sure your virtual machine is selected and press *Start*. VirtualBox Manager will ask you to select a virtual optical disk file or a physical optical drive to start the virtual machine from. Select the iso file you downloaded previously and press *Start*.



The Ubuntu installation process will now begin. It consists of thirteen steps and is quite painless.

The Welcome Screen

Here you should select your preferred language. I'm using English.

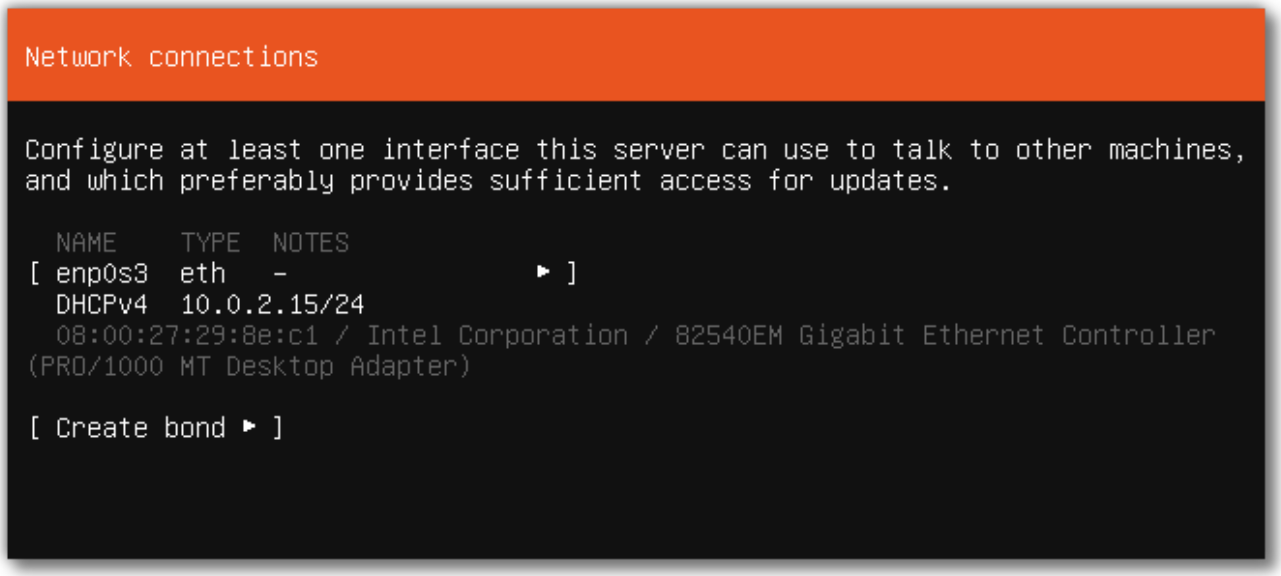


The Keyboard Configuration Screen

Here you should select a keyboard layout. As I'm using a German keyboard, I asked Ubuntu to detect my layout, which it did with a couple of simple questions.

The Network Connections Screen

Here Ubuntu will attempt to configure the standard network interface. Normally you can just accept the default and select *Done*.

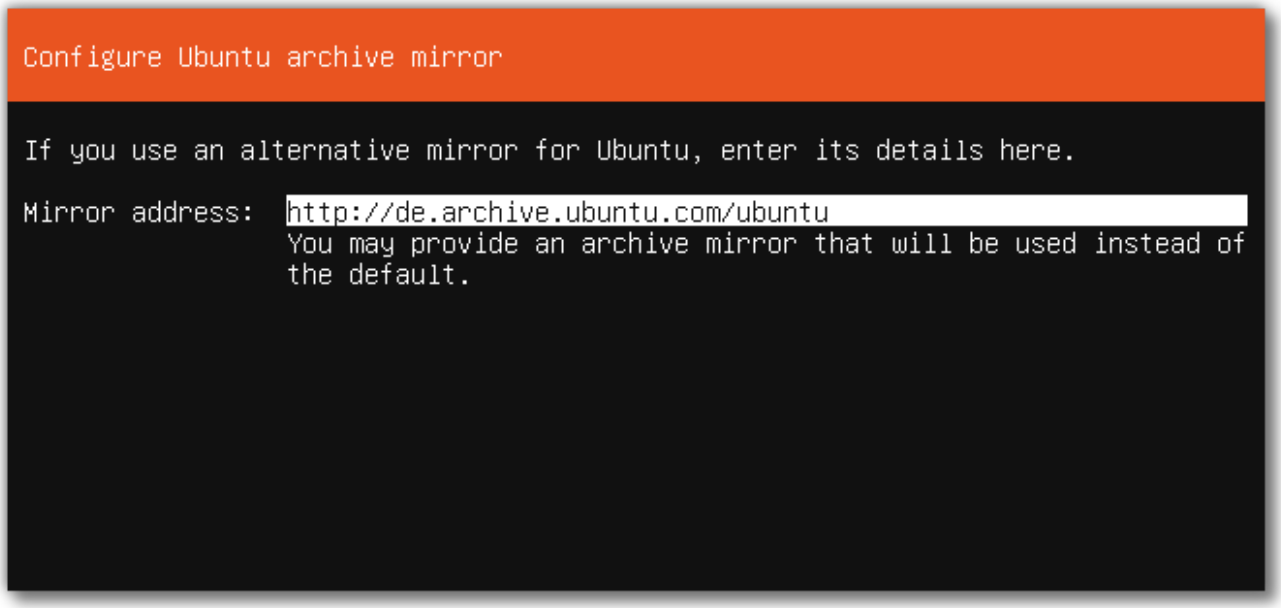


The Configure Proxy Screen

If your system requires a proxy to connect to the internet (mine doesn't), enter its details in the next dialogue. Then select *Done*.

The Ubuntu Archive Mirror Screen

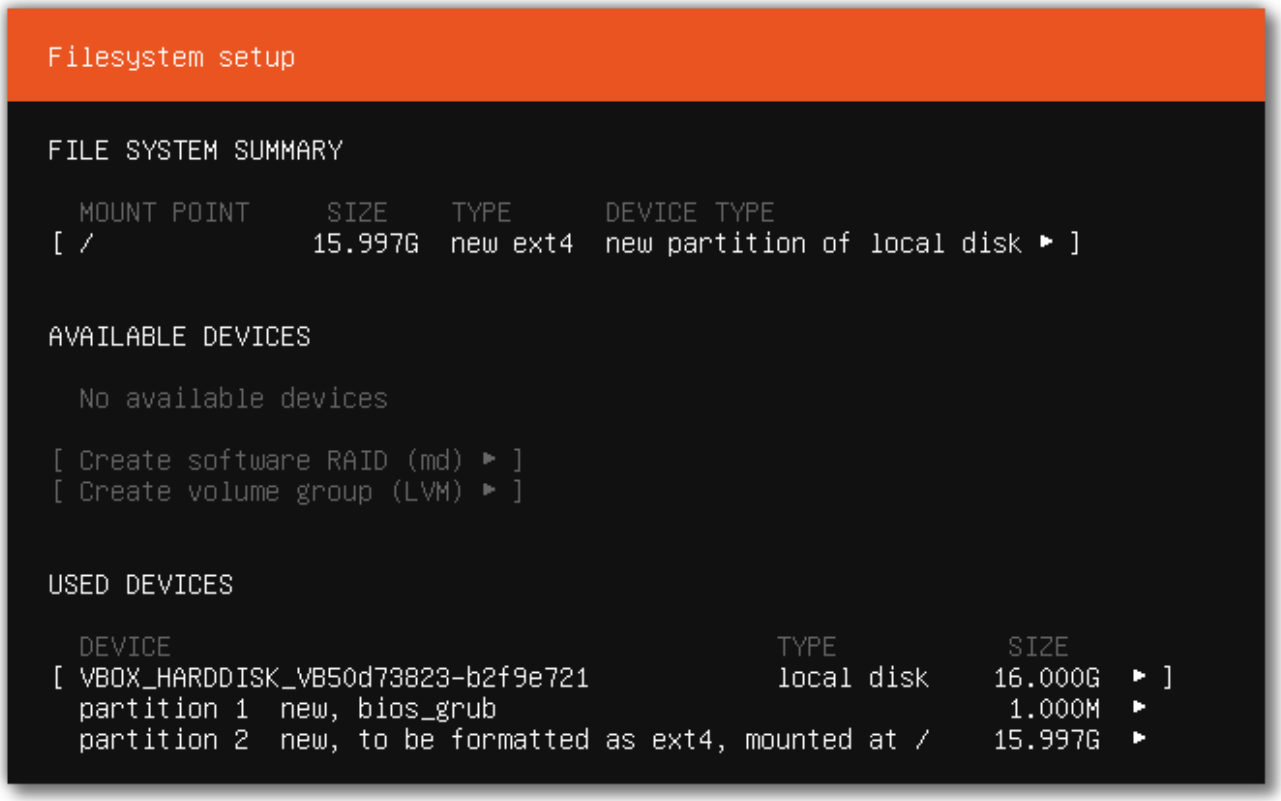
If you wish to use an alternative mirror for Ubuntu, you can enter the details here. Otherwise accept the default mirror by selecting *Done*.



The Filesystem Setup Screen

The installer can guide you through partitioning an entire disk or, if you prefer, you can do it manually. If you choose to partition an entire disk you will still have a chance to review and modify the results before Ubuntu is installed. I selected *Use An Entire Disk*.

I was then prompted to select my virtual machine's hard disk as the disk to install to, before being shown a summary of what the installer would do. As this is a "destructive action". I was asked to confirm my choice with *Continue*.



The Profile Setup Screen

Here you are required to enter:

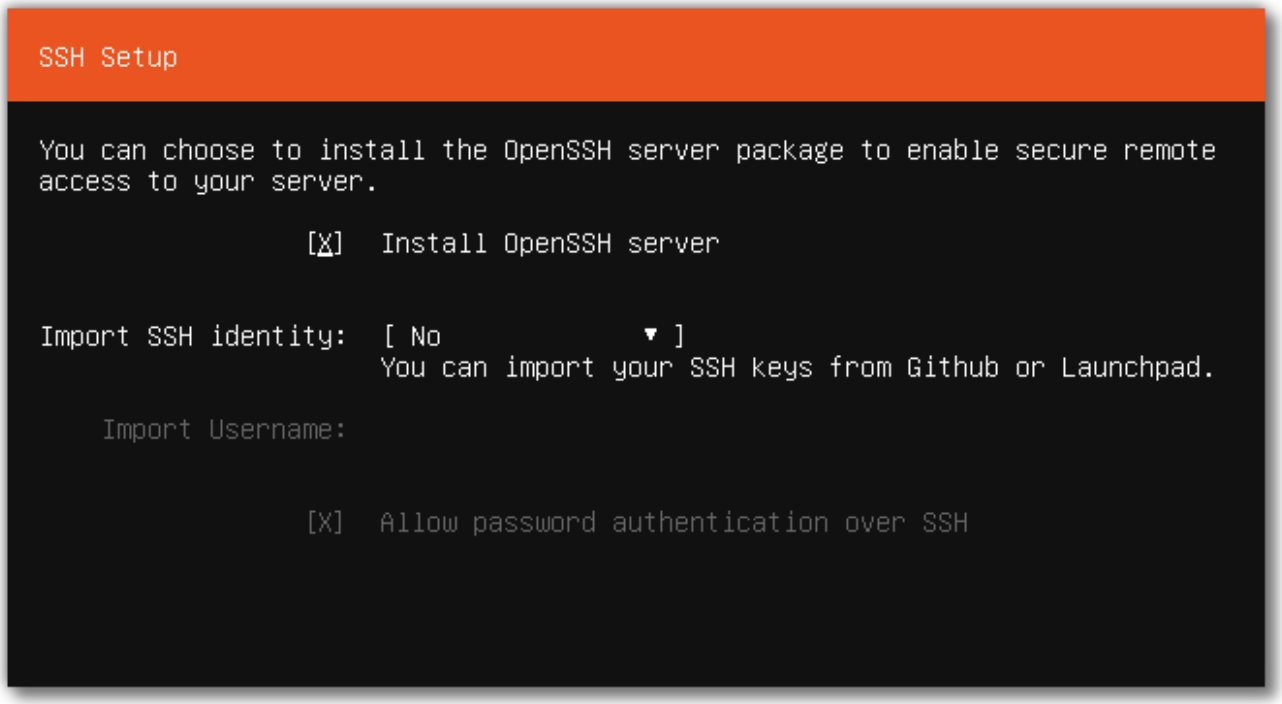
- Your (real) name
- Your server's name
- Your username
- Password

Fill these details out as you see fit.

The SSH Setup Screen

Here we have a chance to install the OpenSSH server package. We'll need this to connect to the virtual machine via SSH later on, so ensure that you select it.

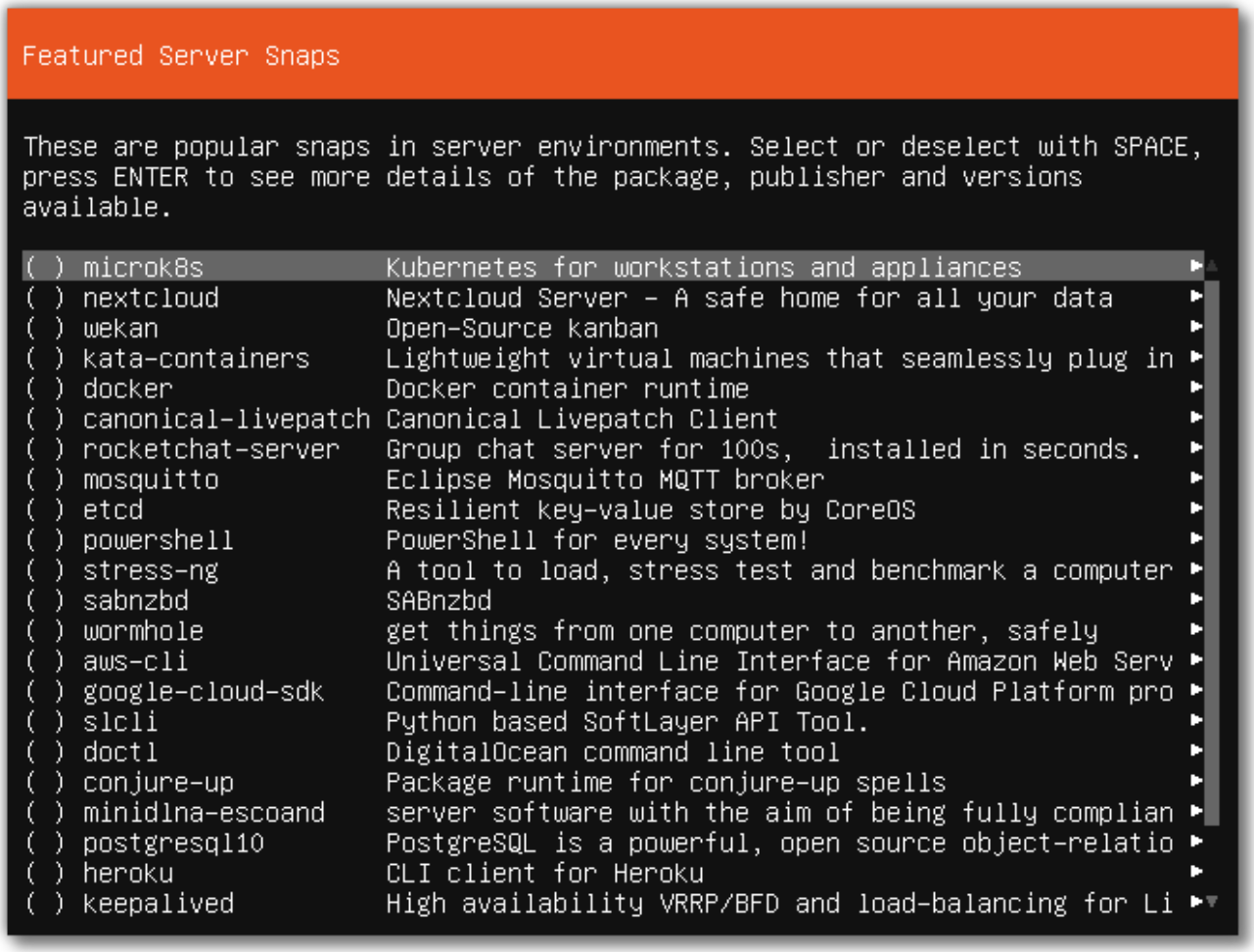
You also have the opportunity to import your SSH keys from GitHub or Launchpad. I selected *No* for this option.



The Featured Server Snaps screen

Here you can select from a list of popular snaps to install on your system. Snaps are self-contained software packages that work across a range of Linux distributions. I didn't

select any.



And that's it, the installation is complete. Ubuntu will ask you to remove the installation medium (which you can do via *Devices* > *Optical Drives* > *Remove disk from virtual drive* > *Force unmount*) and then reboot.

If this option is grayed out, you're good to go. Just reboot.

Up and Running with SSH

Once your virtual machine has rebooted and you have logged in, you'll probably notice that some packages can be updated.

Let's fix that:

```
sudo apt-get update
sudo apt-get upgrade
```

Now let's double check that SSH is installed (it should be if you selected the option *Install OpenSSH server* during instalation).

```
jim@odin:~$ ssh

usage: ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-b bind_address] [-c cipher_spec]
```

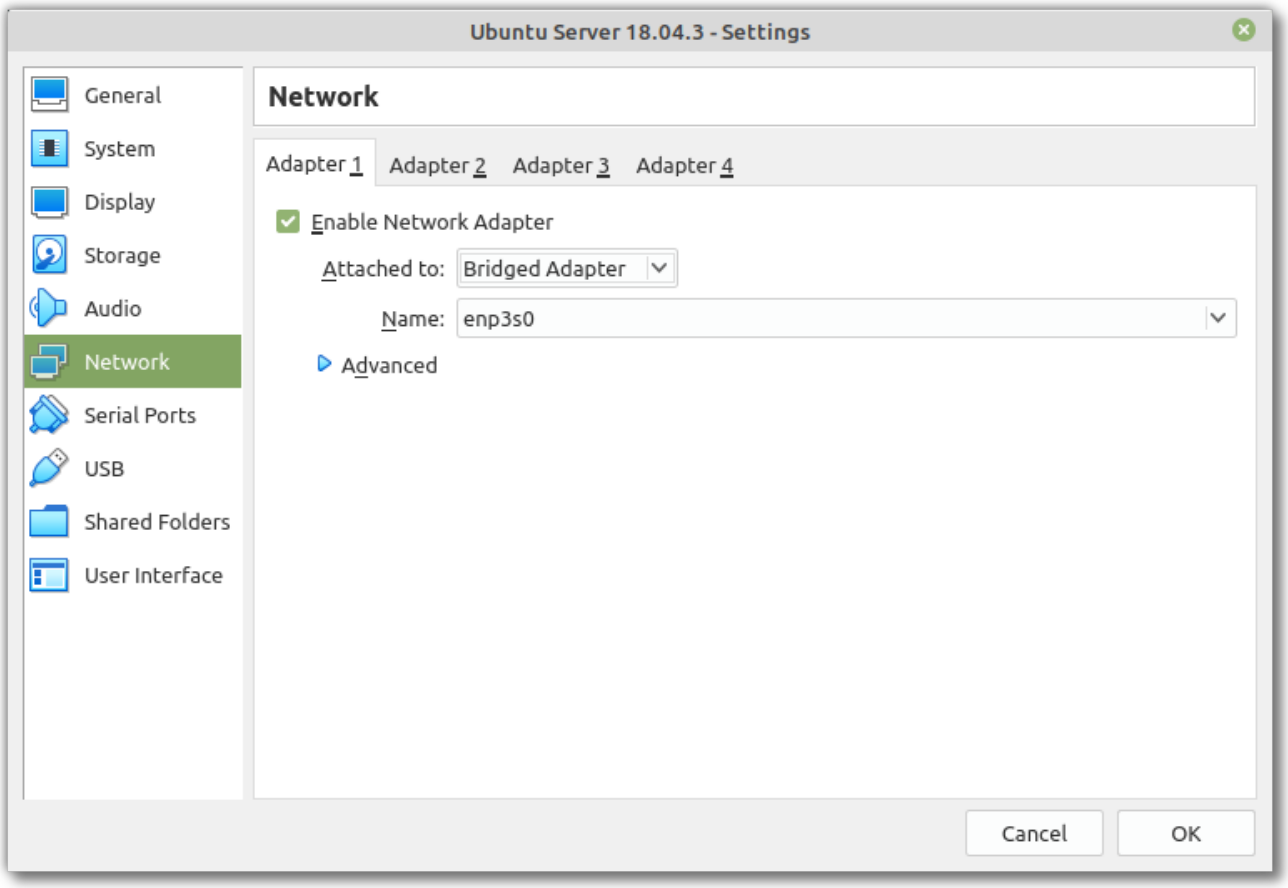
```
            [-D [bind_address:]port] [-E log_file] [-e escape_char]
            [-F configfile] [-I pkcs11] [-i identity_file]
            [-J [user@]host[:port]] [-L address] [-l login_name] [-m mac_spec]
            [-O ctl_cmd] [-o option] [-p port] [-Q query_option] [-R address]
            [-S ctl_path] [-W host:port] [-w local_tun[:remote_tun]]
            [user@]hostname [command]
```

If you get a “command not found” error, you can install it with:

```
sudo apt-get install openssh-server
```

The next step is to give our Ubuntu server an IP address on our local network. To do this, power off the virtual machine using `sudo poweroff` or *Machine > ACPI Shutdown*.

Then, in VirtualBox Manager, click on the *Network* panel on the right. Change the setting *Adapter 1 > Attached to* to “Bridged Adapter” and click *OK*.



Start up your virtual machine, then enter `ifconfig` (in the guest) and note the IP address assigned to your main network adapter. In my case this was `192.168.178.66`.

Note: it is also possible to stick with the original NAT interface and SSH into the guest using port forwarding. You can read more about that here. You can find information on all of the VBox network settings in this comprehensive guide.

Starting and Stopping VirtualBox in Headless Mode

You might have noticed, working with the VirtualBox Manager and the guest OS is a bit of a pain. If you're going to continue doing this, you should at least install the guest additions, as well as enable clipboard support.

There is a slightly nicer way however — you can start and stop the virtual machine using the VBoxManage command from your terminal.

To power on:

```
VBoxManage startvm "Ubuntu Server 18.04.3" --type headless
```

And to power off:

```
VBoxManage controlvm "Ubuntu Server 18.04.3" poweroff
```

Where "Ubuntu Server 18.04.3" is whatever you called your virtual machine (the name it has in the VirtualBox Manager GUI).

Connecting to the Ubuntu Server

Let's go ahead and start the Ubuntu server in headless mode, before connecting to it via SSH.

> Note: The following commands should be run on your host.

On most *nix systems, the SSH client software should be part of the default installation. If you don't have it available, you should be able to grab it from the repos, like so:

```
sudo apt install openssh-client
```

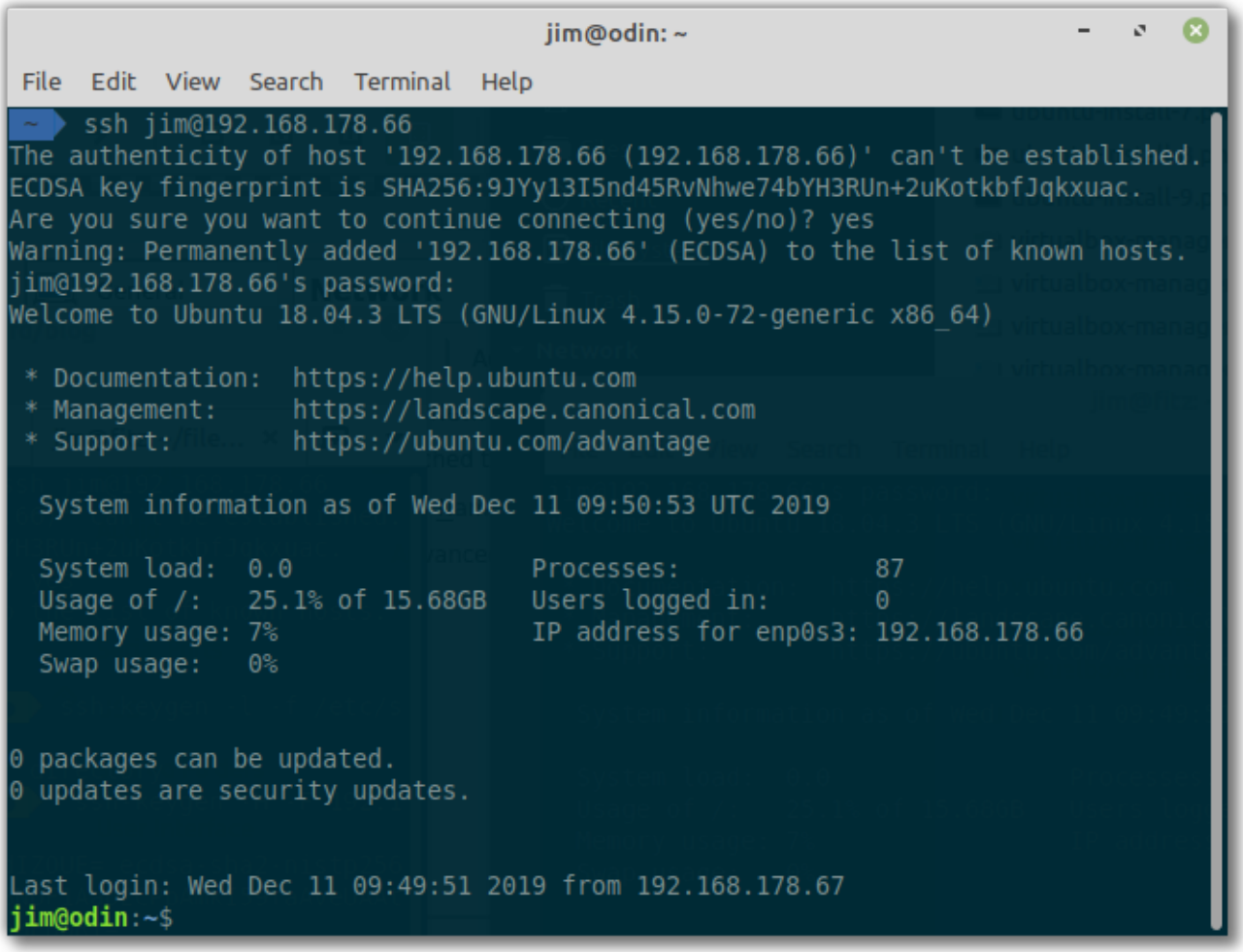
or just hit DuckDuckGo.

Then (ensuring that you replace "jim" and the IP address with your corresponding values) you can connect like so:

```
ssh jim@192.168.178.66
```

This will give you a warning that the host's authenticity cannot be established and ask you if you want to continue connecting. Answer "yes".

Next, it will prompt you for your password. Enter it and you will be connected to your Ubuntu server from your host OS.

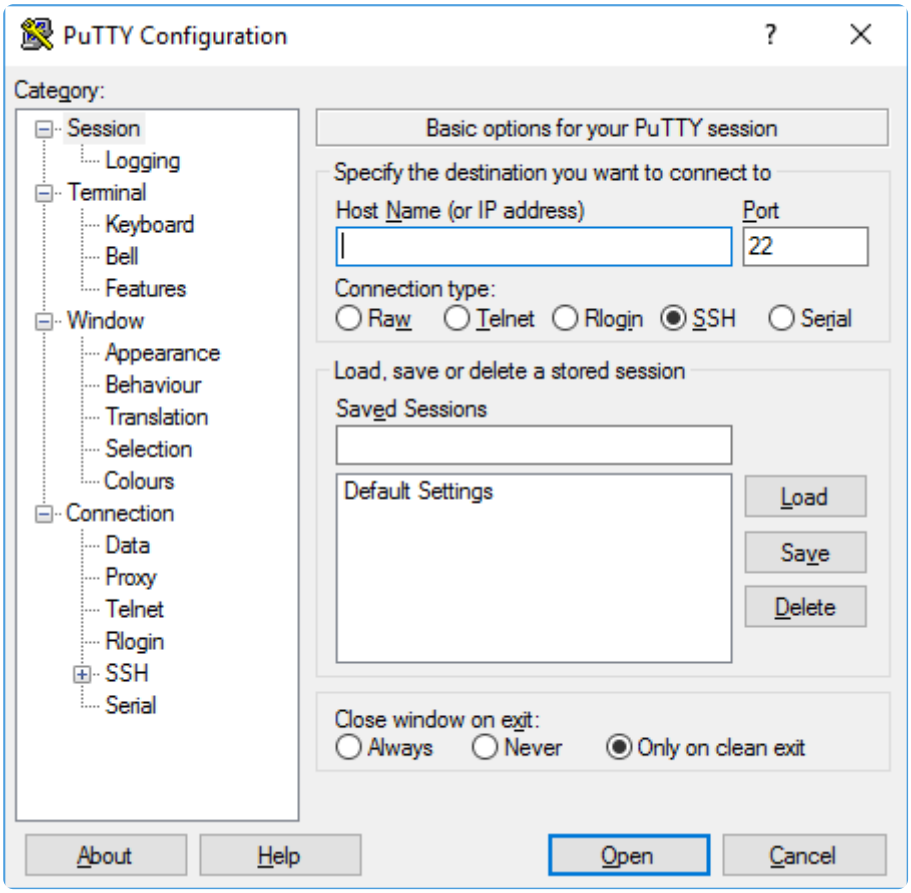


For Windows Users

If you're running Windows you'll need to install a SSH client such as PuTTY.

When PuTTY starts, a window titled *PuTTY Configuration* should open. This window has a configuration pane on the left, a *Host Name* field and other options in the middle, and a pane for saving session profiles in the lower right area.

For simple use, all you need to do is to enter the IP address of the host you want to connect to in the *Host Name* field and click *Open*.



Generate and Install a SSH Key Pair

SSH keys offer a secure manner of logging into a server without the need of a password.

In a nutshell, this depends upon you generating a public and a private SSH key pair. The private key is kept on your PC (and should be guarded carefully). The public key is copied over to the server you wish to connect to.

SSH keys are a complex subject and as such, out of the scope of this tutorial. If you'd like to find out more, I recommend looking for a dedicated tutorial (such as this one).

Generate the Keys

On *nix systems (Windows users see the next section), you can generate your key pair with the following command:

```
ssh-keygen -o -b 4096 -t rsa
```

The `-o` option instructs ssh-keygen to store the private key in the new OpenSSH format instead of the old (and more compatible PEM format). This is advisable, as the new OpenSSH format has an increased resistance to brute-force password cracking.

The `-b` option is used to set the key length to 4096 bits instead of the default 1024 bits for security reasons.

In the following dialogue you will be required to answer a couple of questions:

- Where to save the newly generated key pair
- Which passphrase to use

Here you can accept the default location and leave the passphrase blank by pressing `Return`.

ssh-keygen will then output a summary of what it has done:

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jim/.ssh/id_rsa):
Created directory '/home/jim/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/jim/.ssh/id_rsa.
Your public key has been saved in /home/jim/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:sx5uJeVdH/cT/1+GxsSWYzmjf5hUaE33f/e57EbqBfY jim@fitz
The key's randomart image is:
+---[RSA 4096]----+
|                 |
|                o|
|               +o|
|          .  .+==|
|        So . =@oB|
|        .oo o*+BB|
|        oo  ..*EX|
|       o..   +=+=|
|       .o   ..+=+|
+----[SHA256]-----+
```

Copy the Public Key to the Ubuntu Server

To copy the public key to the Ubuntu server use:

```
ssh-copy-id -i ~/.ssh/id_rsa.pub jim@192.168.178.66
```

Where `~/.ssh/id_rsa.pub` is the path to your public key, taken from the output above. And where `jim@192.168.178.66` should be altered to reflect your details.

The command will run and you should be asked for your server password. Enter it, then attempt to log into the server like so:

```
ssh jim@192.168.178.66
```

This time you should be in without a password.

For Windows Users

You should be able to use a tool like PuTTYgen to achieve the same thing. Here is a tutorial on using PuTTYgen to create a new key pair for authentication.

You will have a little more leg work when it comes to copying the key to the server, where you will need to add the public key to a `~/.ssh/authorized_keys` file.

You can do that like so:

```
cd
mkdir .ssh
cd .ssh
nano authorized_keys
```

This will create the appropriate file, then open the nano editor into which you can copy your newly generated public key.

When you're done, press `Ctrl` + `X` to save your changes and exit nano.

Conclusion

This has been quite a long post, but by the end of it you should have a working installation of Ubuntu Server running on VirtualBox that you can connect to from your host operating system via SSH.

As mentioned, this will form the basis for a future tutorial on deploying a Ruby on Rails app to an Ubuntu server.