Austin Kingsland

Building a cloud capable Apache web server and MariaDB database on a virtual machine running Linux Server using Openstack.

VM user: Guest
VM pass: Password24

Why use a Virtual Machine for your Cloud Server?

Isolation: Running DevStack in a VM keeps your host system clean and unaffected. DevStack makes significant changes to the system's network configuration and installs many services. Isolating these changes within a VM prevents potential conflicts with your host system's setup.

Snapshot and Restore: VMs allow you to take snapshots of the entire state, which is extremely useful during development and testing. If something goes wrong, you can quickly revert to a previous snapshot, which is much easier than reinstalling or debugging the system.

Replicability and Portability: A VM can be easily cloned or moved to another machine, allowing you to replicate your OpenStack setup elsewhere without starting from scratch.

Testing and Learning: For learning purposes, using a VM allows you to experiment with OpenStack in a controlled and safe environment, without the risk of damaging your main operating system.

Linux Server - Operating System

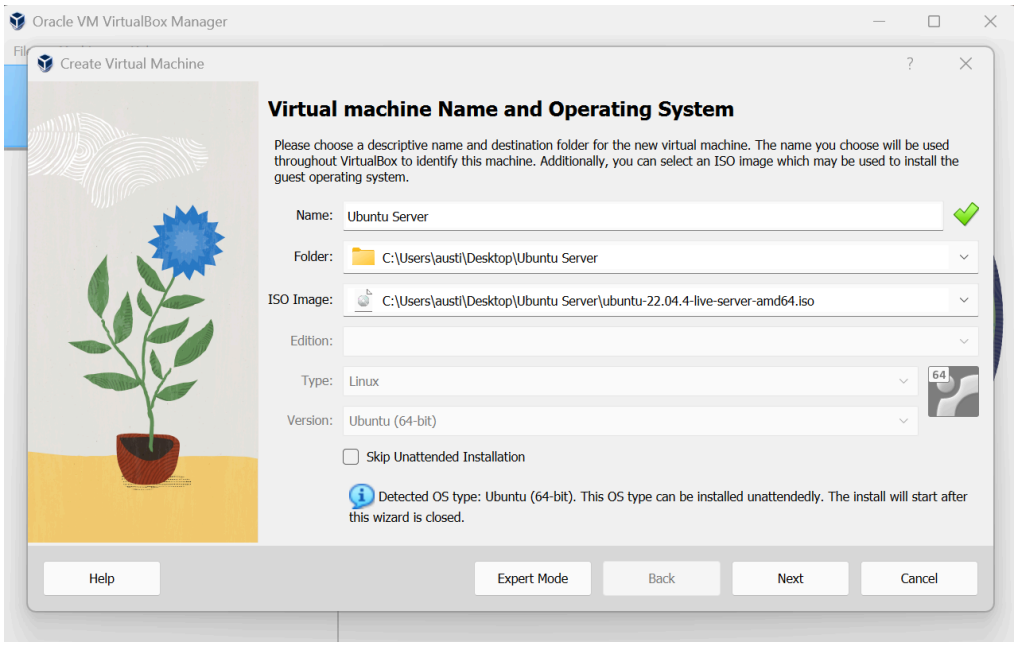
Download the .ISO file from the linux distribution.
https://ubuntu.com/download/server

Virtual Box - VM software

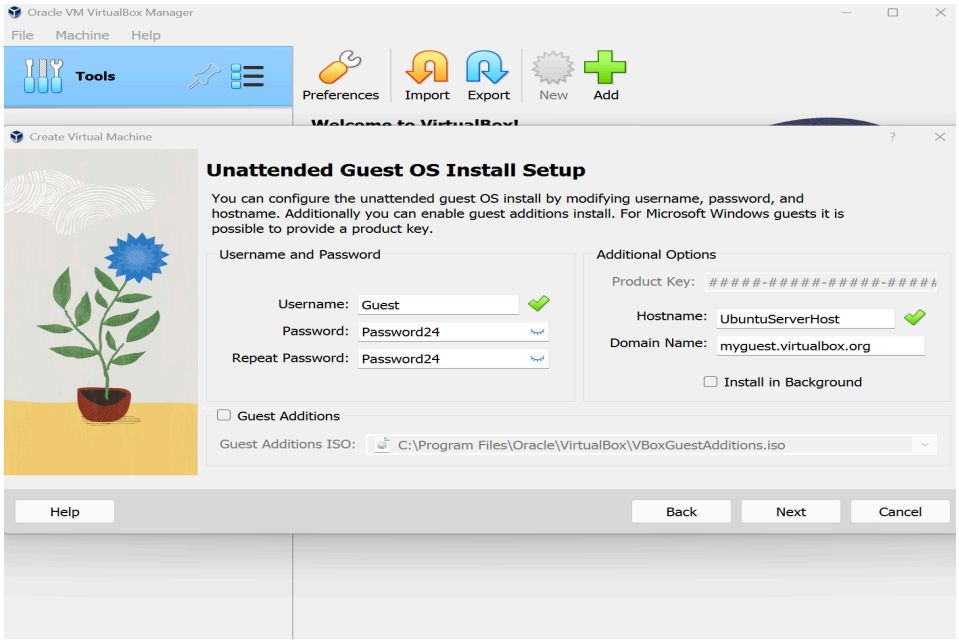
Download VirtualBox to your system.
https://www.virtualbox.org/wiki/Downloads

Installing and starting the Virtual Machine (VM) using VirtualBox.

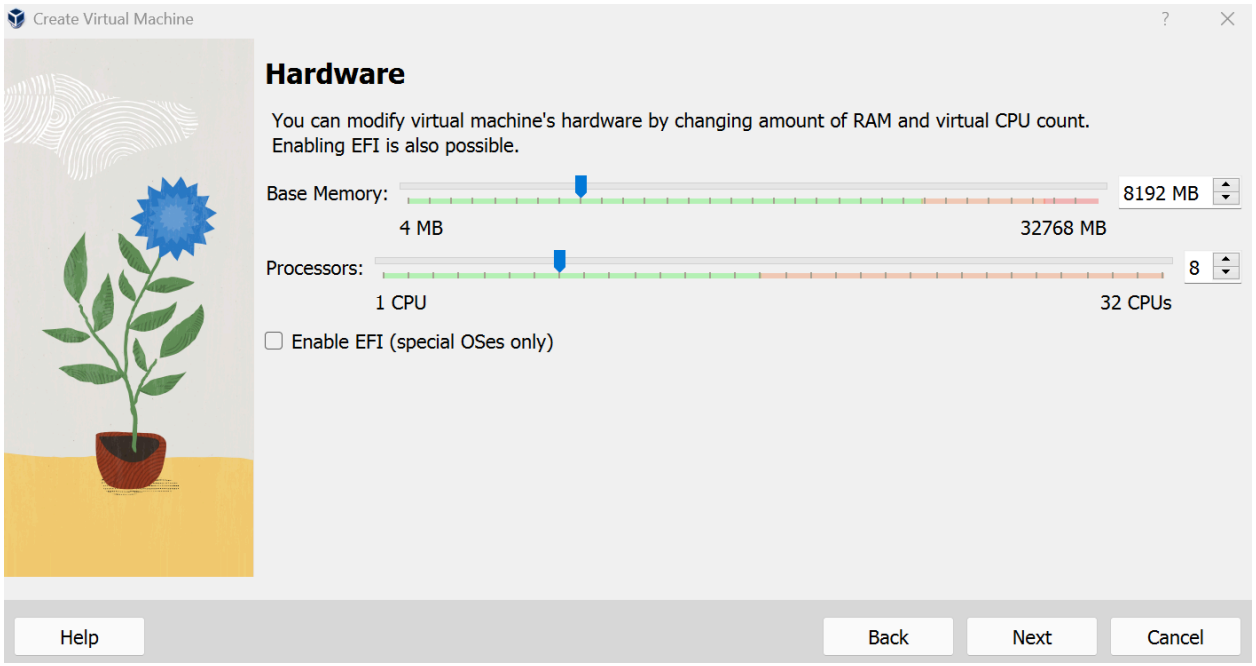
Attach the ISO file you chose for your OS.



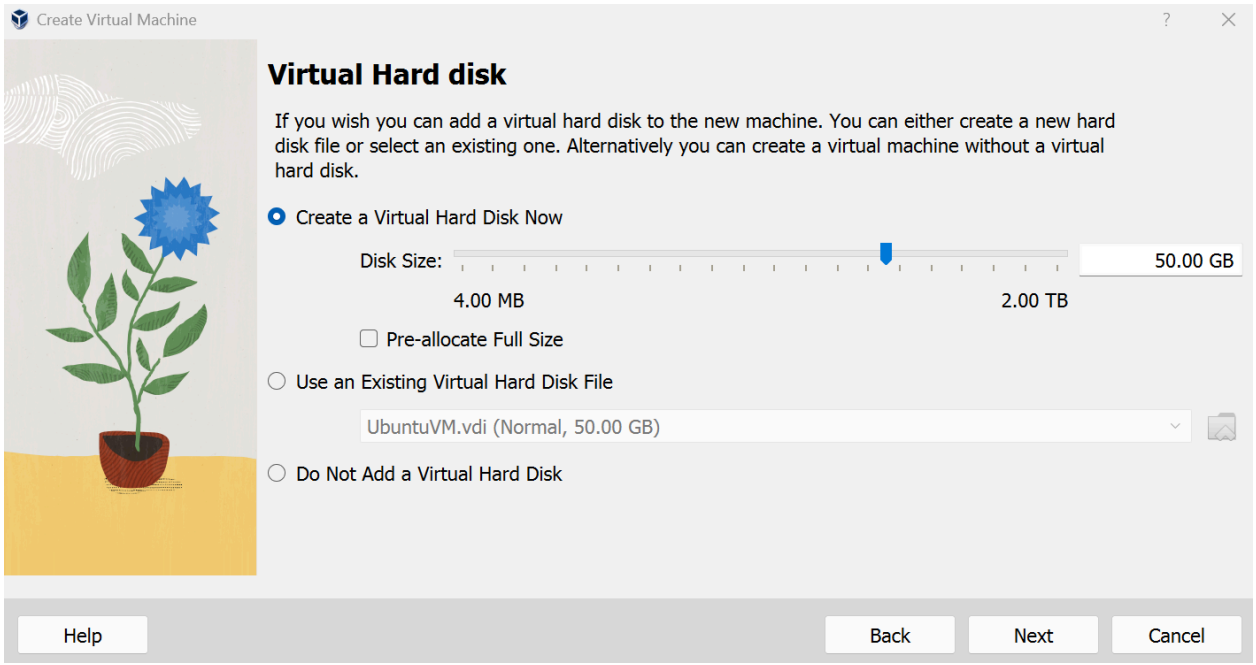
Set up the operating system users.



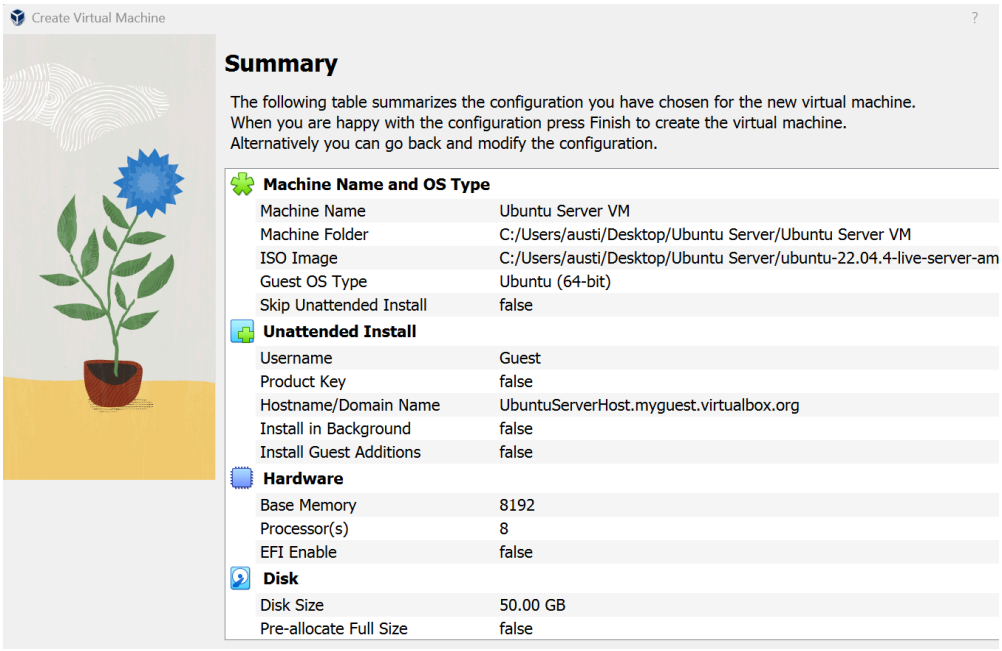
Set the initial RAM and CPU power used by the VM. Once finished, reserve storage space on the HD. We will cover changing these values later.



You will need at least 50GB of storage.

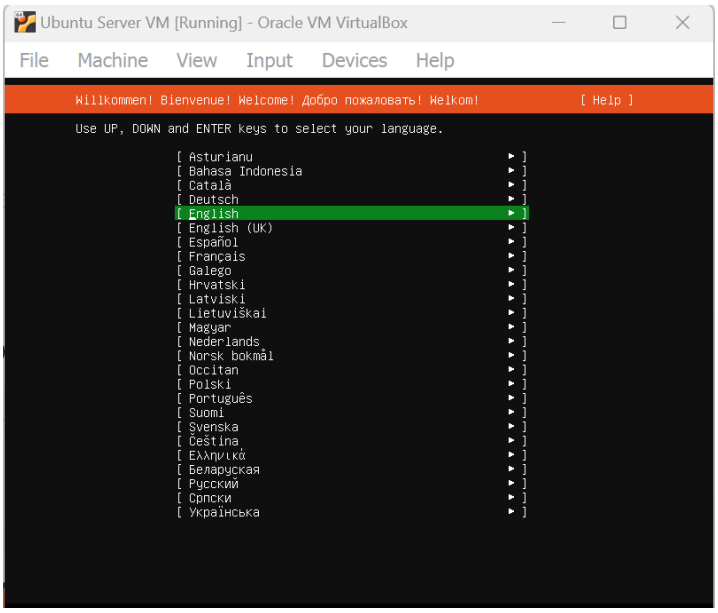


The confirmation screen will look like this when you are ready to create the VM.

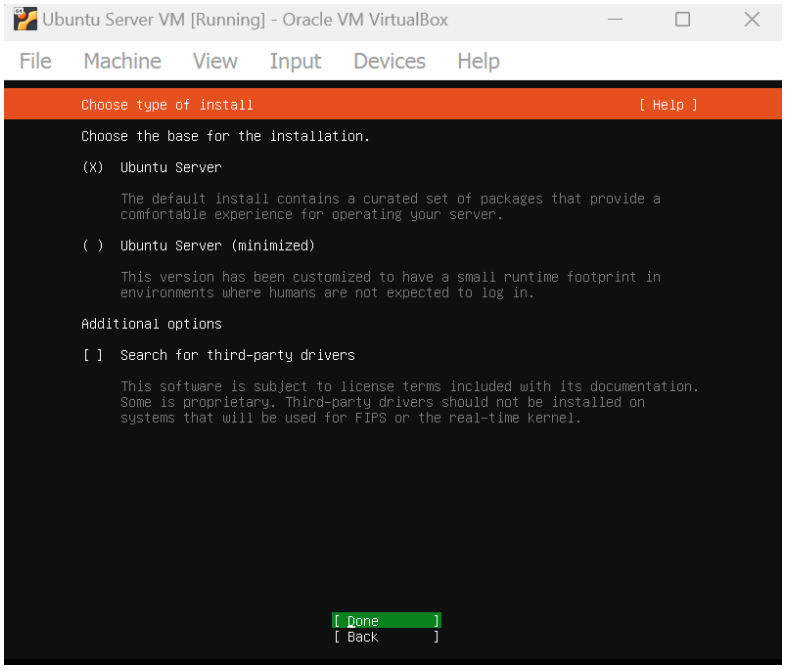


Configuring our Linux Server OS

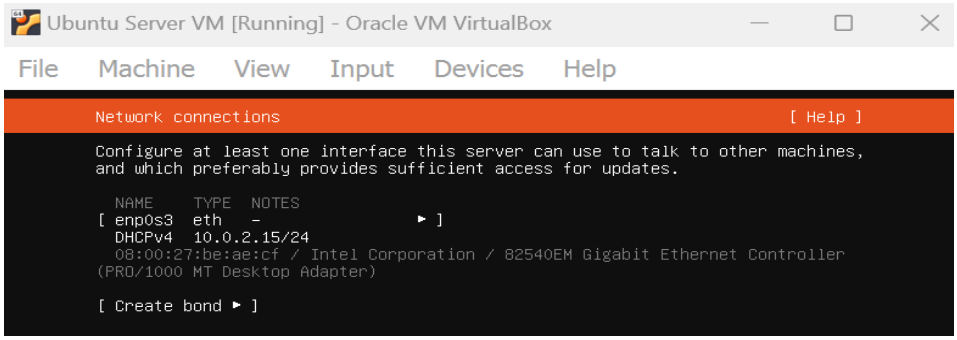
Once you confirm, the VM operating system will boot up.

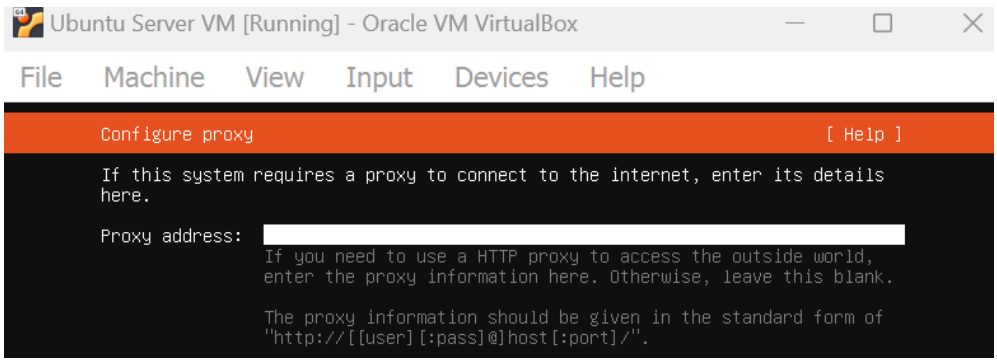


Select “Ubuntu Server” for the installation.

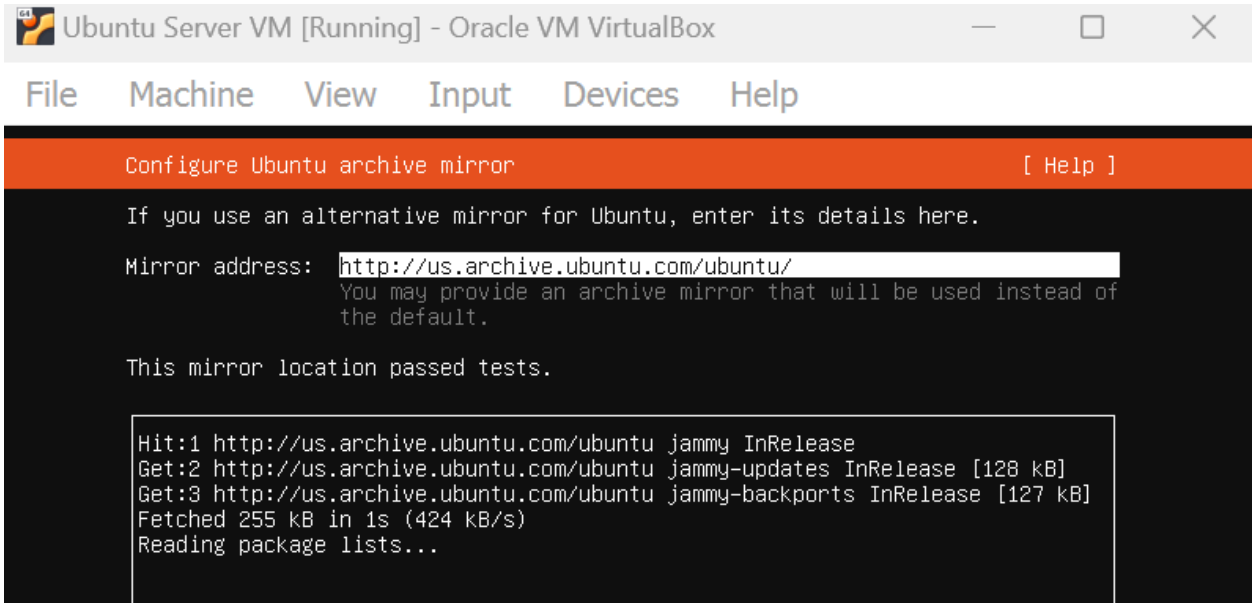


Enp0s3 works for our Network connection, and we do not need a proxy.

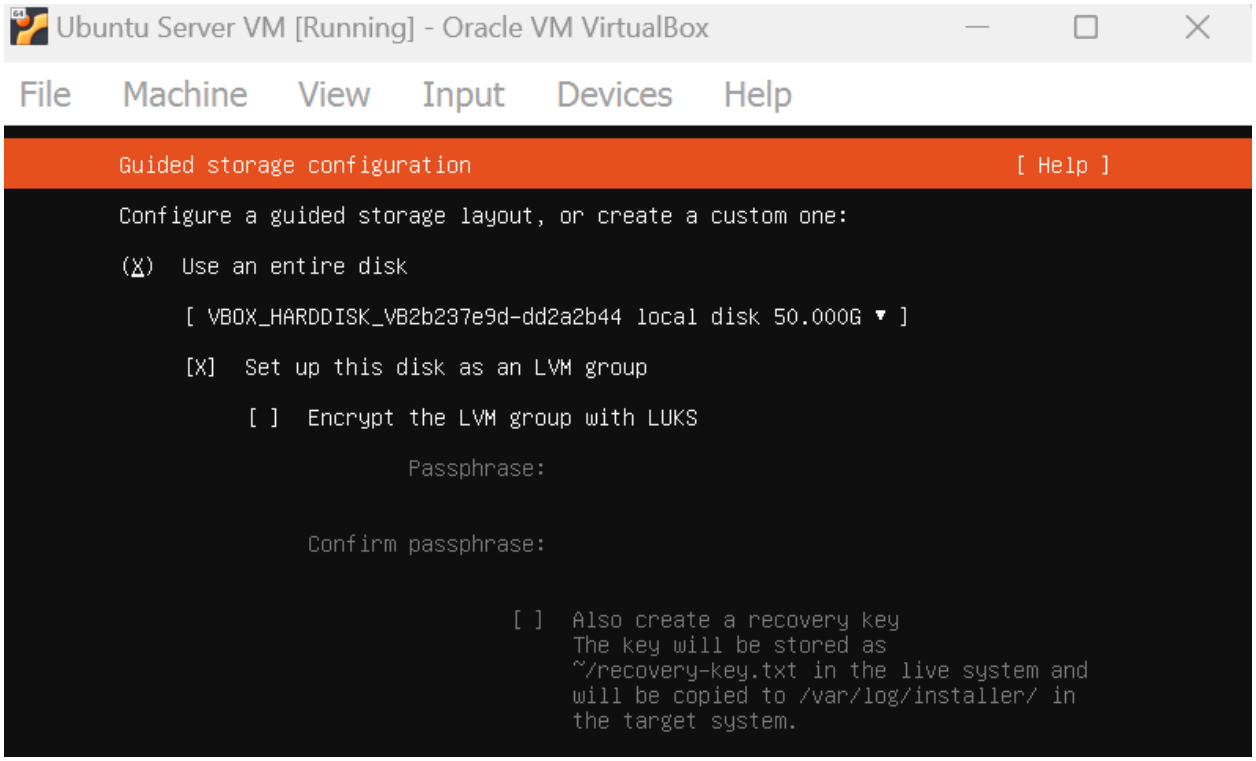




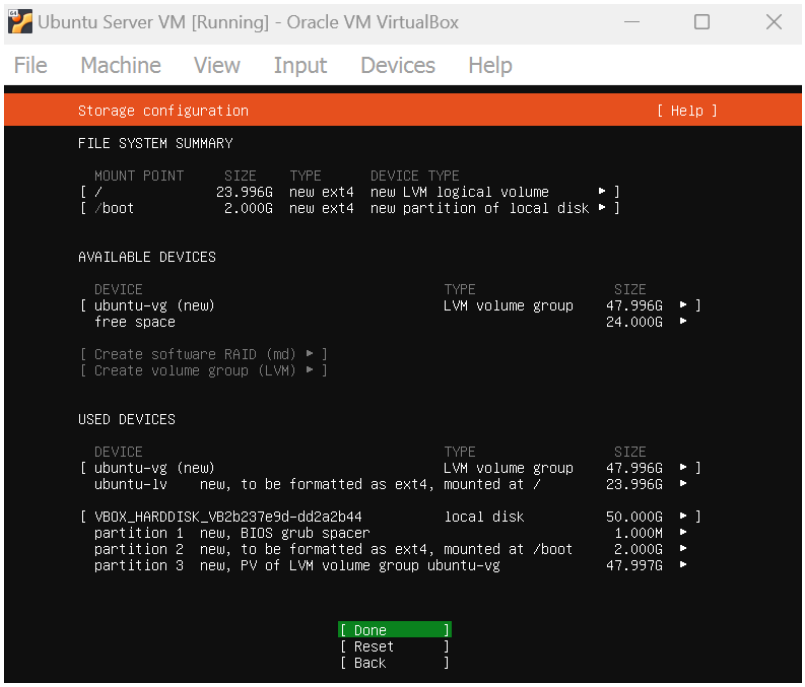
Ubuntu will run a mirror test for the archive files.



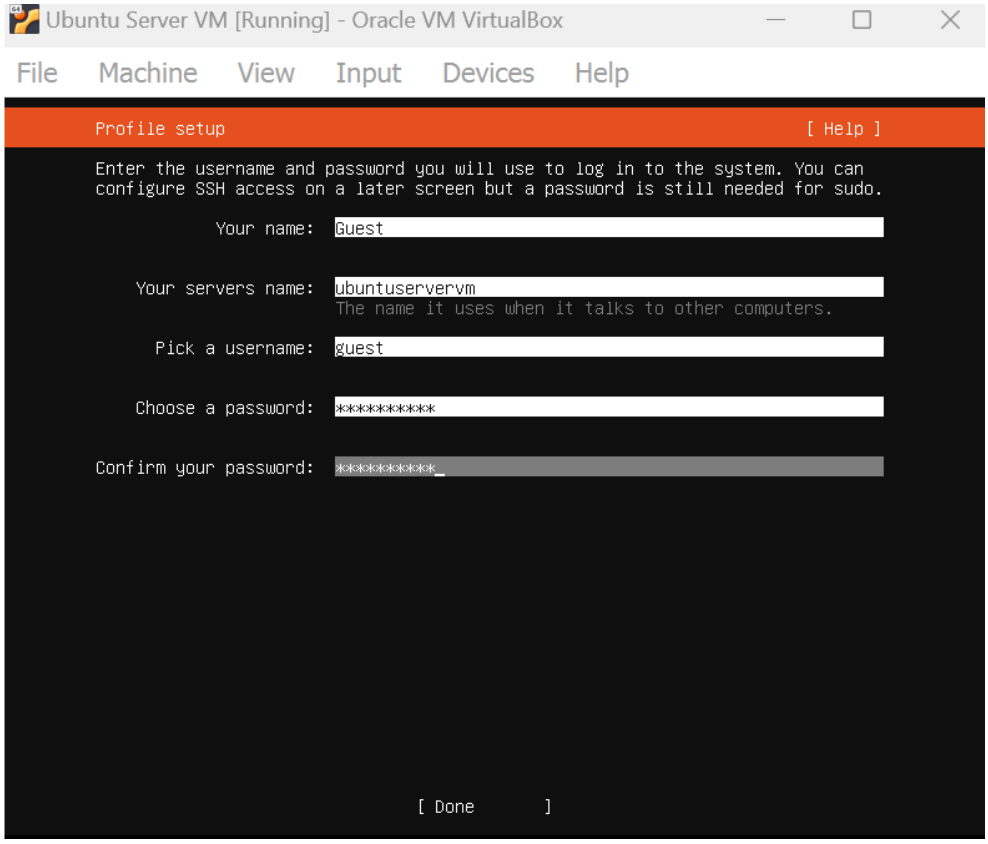
Choose to utilize the full 50 GB allocated for our VM. This space is needed to install Openstack.



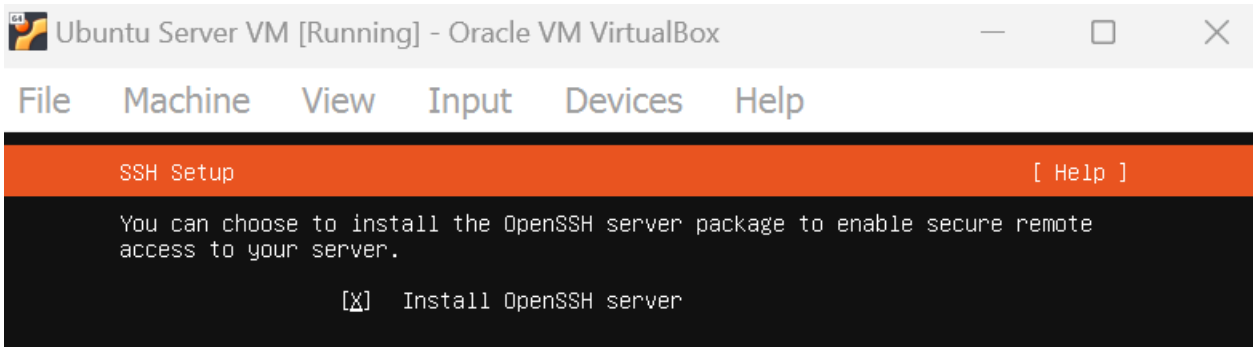
You will then arrive at the summary page. If everything appears as such, continue.



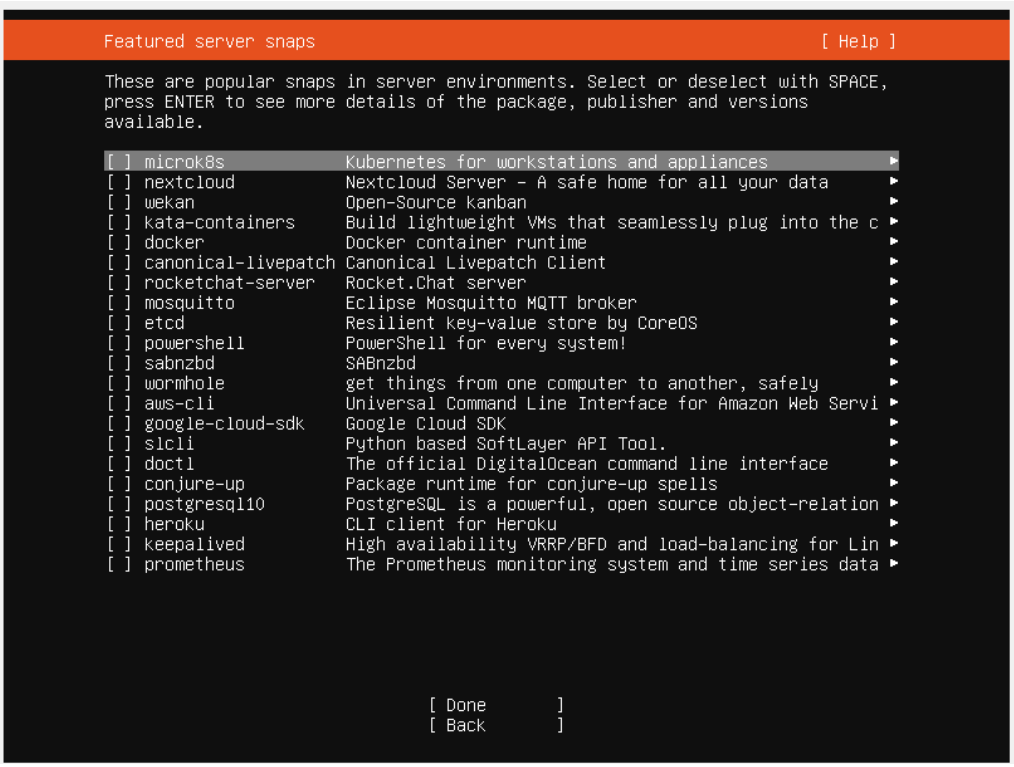
You will need to configure a password for the profile. Here we use “guest” for our user, and “Password24” for the password.



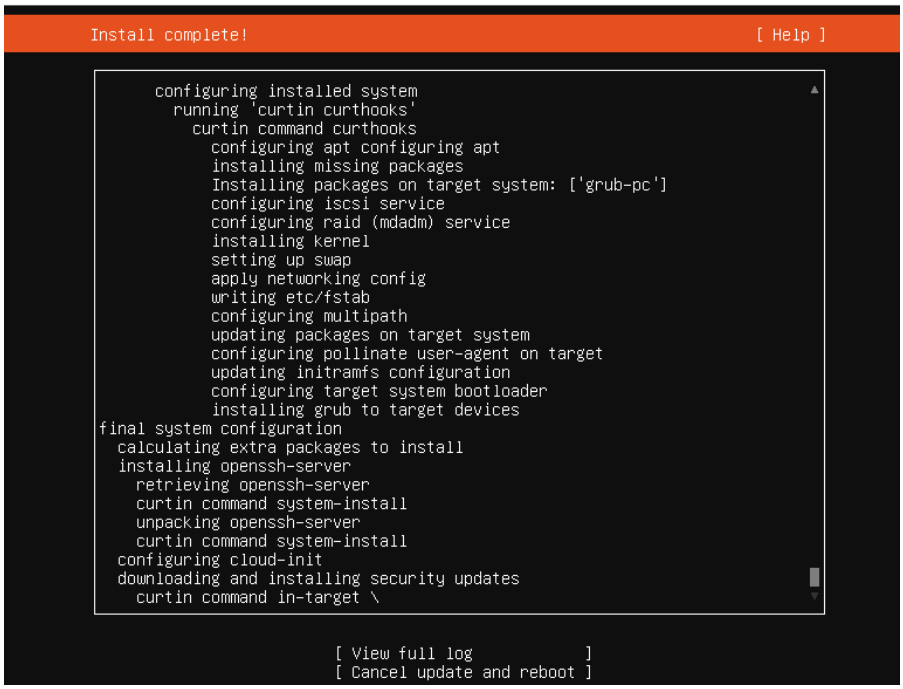
We will want to install OpenSSH to allow remote access to the server. Although this is not necessary for the project, it is common to ssh into servers to work on them remotely.



You can then optionally install some well used snap products.



With that, our Virtual Machine is ready to go!



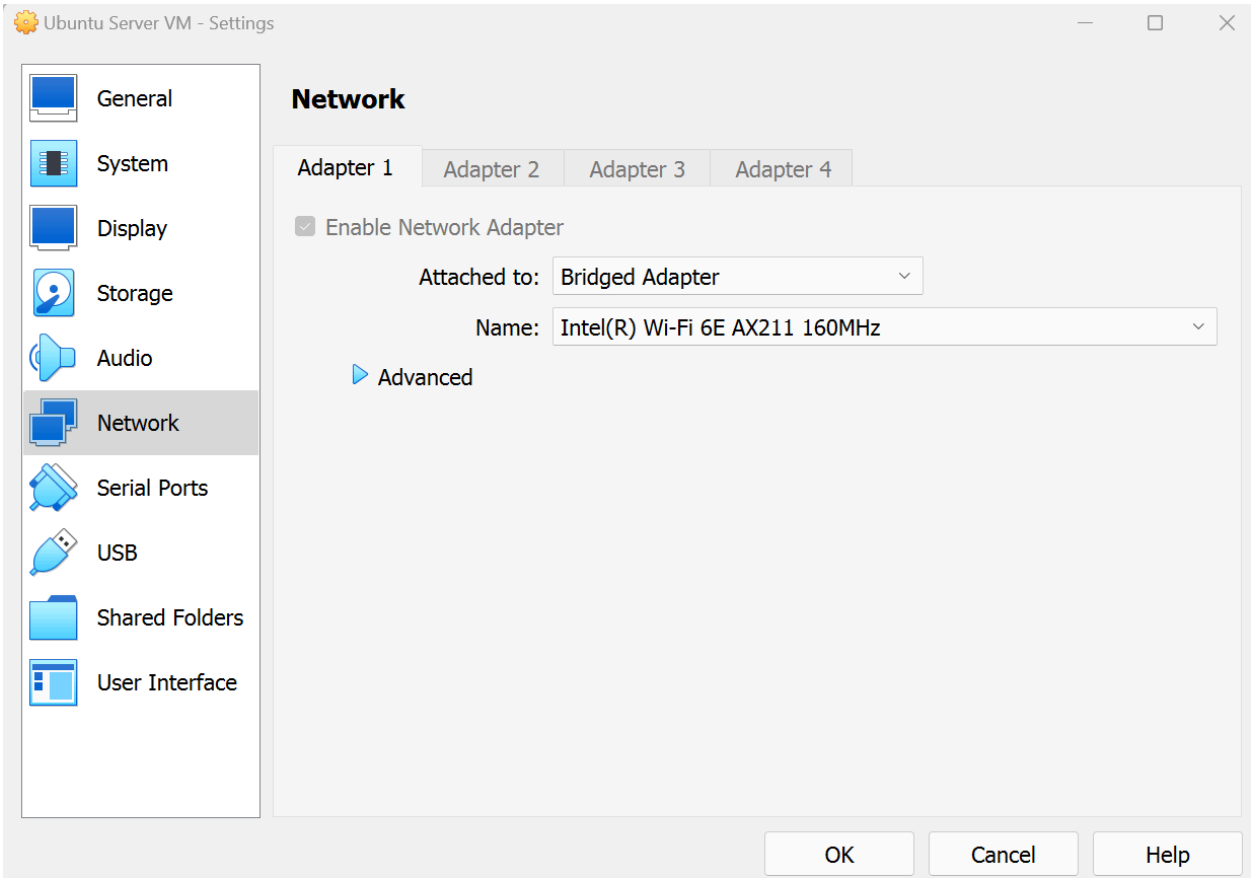
Switching to a Bridged Adapter

Now that we have successfully created a VM running linux server we have a couple options for our network. We can either set up complex port forwarding with our NAT connection and configure Openstack to use the IP address assigned to it from a pool, or we can use a bridged adapter.

With a bridged adapter, the VM becomes part of your host network and therefore port forwarding is unnecessary.

To switch to the bridged adapter, first power down your VM.

Next, right click on your VM, and navigate to settings > network. Switch to a bridged adapter in the top section.



Now simply restart your Virtual Machine and run the command to check your IP. Make sure any changes to files where we add the IP address are consistent with the new IP shown in the VM.

Installing Openstack and it's Services

Openstack is a free cloud computing software. Since we want to use cloud computation for both our Apache Web Server and our MariaDB Server we will set up Openstack before configuring them.

First, since our VM is still in a fresh state, we will need to run basic updates to our software.

Run in the VM: *sudo apt update* and *sudo apt upgrade*. The updates should be relatively light.

Next, we will install git. Run *sudo apt install git*.If it has already been installed, it will simply switch to manual install.

Now we will clone the DevStack repository from git. Devstack is an all in one openstack configuration for developers. Run *git clone https://opendev.org/openstack/devstack*.

Next, change into the devstack directory. Run *cd devstack*.

```
 systemctl restart packagekit.service udisks2.service
Service restarts being deferred:
 systemctl restart unattended-upgrades.service

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
guest@ubuntuservervm:~$ sudo apt install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.34.1-1ubuntu1.11).
git set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
guest@ubuntuservervm:~$ git clone https://opendev.org/openstack/devstack
Cloning into 'devstack'...
remote: Enumerating objects: 50905, done.
remote: Counting objects: 100% (30990/30990), done.
remote: Compressing objects: 100% (10388/10388), done.
remote: Total 50905 (delta 30239), reused 20602 (delta 20602), pack-reused 19915
Receiving objects: 100% (50905/50905), 9.49 MiB | 5.90 MiB/s, done.
Resolving deltas: 100% (36155/36155), done.
guest@ubuntuservervm:~$ cd devstack
guest@ubuntuservervm:~/devstack$ ./stack.sh
```

If we try to install openstack right now, we will get an error when determining the host ip address. We will need to configure an IP address in the local.conf file to represent our VM IP address.

```
  GNU nano 6.2                                          local.c
[[local|localrc]]
ADMIN_PASSWORD=Password24
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD
HOST_IP=192.168.1.224 # Updated to new bridged IP
```

Use *ip addr show* to see your VM's current IP address.

```
guest@ubuntuservervm:~/devstack$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1
0
    link/ether 08:00:27:8a:6c:c2 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 metric 100 brd 10.0.2.255 scope global dynamic enp0s3
       valid_lft 83960sec preferred_lft 83960sec
    inet6 fe80::a00:27ff:fe8a:6cc2/64 scope link
       valid_lft forever preferred_lft forever
3: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether e2:96:c6:bb:a4:66 brd ff:ff:ff:ff:ff:ff
4: br-int: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether aa:4c:a7:81:b0:4e brd ff:ff:ff:ff:ff:ff
5: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default ql
1000
    link/ether 52:54:00:86:9a:5a brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
       valid_lft forever preferred_lft forever
6: br-ex: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default ql
1000
    link/ether 52:6b:51:53:dd:43 brd ff:ff:ff:ff:ff:ff
    inet 172.24.4.1/24 scope global br-ex
       valid_lft forever preferred_lft forever
    inet6 2001:db8::2/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::506b:51ff:fe53:dd43/64 scope link
       valid_lft forever preferred_lft forever
guest@ubuntuservervm:~/devstack$
```

Edit the local.conf file to contain the Host_IP. If using a bridged adapter, the IP will reflect the local network instead. If you make changes to this after running ./stack.sh then you will have to unstack and stack again.

```
  GNU nano 6.2                                        local.conf
[[local|localrc]]
ADMIN_PASSWORD=Password24
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD
HOST_IP=10.0.2.15
```

Once this file is properly configured, run ./stack.sh while in the devstack directory. This command may take a long time to run as it is a large download.

Once the download finished without error, we must manually set the Admin Environmental Variables. Run the following commands from the devstack directory. Your IP address will be different.

```
export OS_AUTH_URL=http://10.0.2.15/identity
export OS_USERNAME=admin
export OS_PASSWORD=Password24
export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_IDENTITY_API_VERSION=3
```

```
guest@ubuntuservervm:~/devstack$ export OS_AUTH_URL=http://10.0.2.15/identity
guest@ubuntuservervm:~/devstack$ export OS_USERNAME=admin
guest@ubuntuservervm:~/devstack$ export OS_PASSWORD=Password24
guest@ubuntuservervm:~/devstack$ export OS_PROJECT_NAME=admin
guest@ubuntuservervm:~/devstack$ export OS_USER_DOMAIN_NAME=Default
guest@ubuntuservervm:~/devstack$ export OS_PROJECT_DOMAIN_NAME=Default
guest@ubuntuservervm:~/devstack$ export OS_IDENTITY_API_VERSION=3
guest@ubuntuservervm:~/devstack$ _
```

Then we can check the changes by running *env | grep OS_*.

```
guest@ubuntuservervm:~/devstack$ env | grep OS_
OS_REGION_NAME=RegionOne
OS_PROJECT_DOMAIN_ID=default
OS_CACERT=
OS_AUTH_URL=http://10.0.2.15/identity
OS_PROJECT_DOMAIN_NAME=Default
OS_USER_DOMAIN_ID=default
OS_USERNAME=admin
OS_VOLUME_API_VERSION=3
OS_AUTH_TYPE=password
OS_USER_DOMAIN_NAME=Default
OS_PROJECT_NAME=admin
OS_PASSWORD=Password24
OS_IDENTITY_API_VERSION=3
guest@ubuntuservervm:~/devstack$ _
```

Verify the status of openstack services using *openstack service list*.

```
guest@ubuntuservervm:~/devstack$ openstack service list
+----------------------------------+-------------+----------------+
| ID                               | Name        | Type           |
+----------------------------------+-------------+----------------+
| 47a966fa17f84b618825544d3284b5ab | glance      | image          |
| 50e4ae3eedd9426bbd9dd1a1cdad1dba | placement   | placement      |
| a00f6b53ea694e93abe863ff82ef50bd | neutron     | network        |
| b24066afeff844b184fbb340c52267c5 | cinder      | block-storage  |
| c5e6682dc7e04b3fa451d7f9ed5fd8bf | cinderv3    | volumev3       |
| d01dbd8ecbb841e4ac76374f560c14a9 | nova        | compute        |
| df85c0a12d504437b67f162cbac08921 | keystone    | identity       |
| e1ea3a939e624357858fde854614fdb2 | nova_legacy | compute_legacy |
+----------------------------------+-------------+----------------+
guest@ubuntuservervm:~/devstack$
```

A quick overview of these systems is posted below.

glance: Image service
placement: Resource placement service
neutron: Network service
cinder: Block storage service
cinderv3: Block storage service (version 3)
nova: Compute service
keystone: Identity service
nova_legacy: Compute legacy service

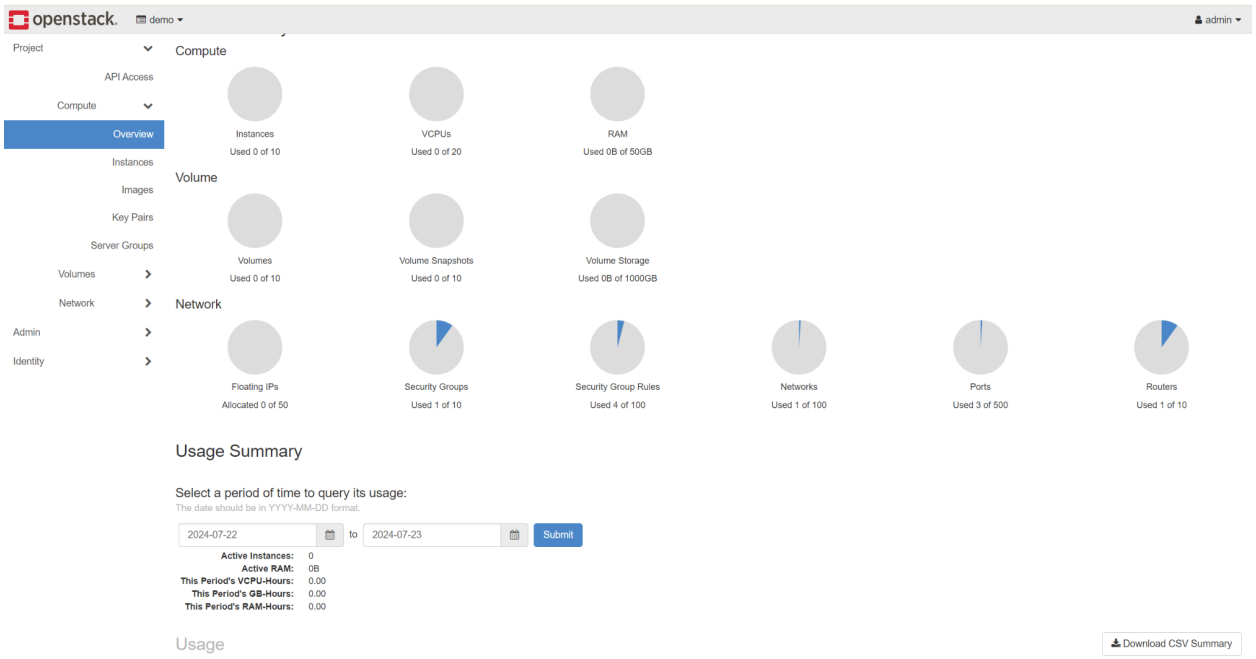
```
  GNU nano 6.2                  /etc/apache2/sites-available/horizon.conf *
<VirtualHost *:80>
    ServerName 10.0.2.15_
    WSGIScriptAlias /dashboard /opt/stack/horizon/openstack_dashboard/wsgi.py
    WSGIDaemonProcess horizon user=guest group=guest processes=3 threads=10 home=/opt/stack/horizon
    WSGIApplicationGroup %{GLOBAL}

    SetEnv APACHE_RUN_USER guest
    SetEnv APACHE_RUN_GROUP guest
    WSGIProcessGroup horizon

    DocumentRoot /opt/stack/horizon/.blackhole/
    Alias /dashboard/media /opt/stack/horizon/openstack_dashboard/static
    Alias /dashboard/static /opt/stack/horizon/static

    RedirectMatch "^/$" "/dashboard/"

    <Directory />
        Options FollowSymLinks
        AllowOverride None
    </Directory>

    <Directory /opt/stack/horizon/>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride None
        # Apache 2.4 uses mod_authz_host for access control now (instead of
        #  "Allow")
        <IfVersion < 2.4>
            Order allow,deny
            Allow from all
        </IfVersion>
        <IfVersion >= 2.4>
            Require all granted
        </IfVersion>

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

Verify that apache is running *sudo systemctl status apache2*. Check if is listening with *sudo netstat -tuln | grep :80*.

If any changes were made to the local.conf file (specifically the IP address) after running ./stack.sh then you'll have to unstack and stack again. Use openstack endpoint list to make sure the URL is consistent.

```
[7]+  Stopped                 sudo systemctl status devstack@keystone
guest@ubuntuservervm:~/devstack$ openstack endpoint list
+----------------+-----------+--------------+---------------+---------+-----------+-----
| ID             | Region    | Service Name | Service Type  | Enabled | Interface | URL
+----------------+-----------+--------------+---------------+---------+-----------+-----
| 028439baa3444  | RegionOne | keystone     | identity      | True    | public    | http
| dee9b624b8136  |           |              |               |         |           | .1.2
| 6ba368         |           |              |               |         |           | y
| 1aade4170b554  | RegionOne | placement    | placement     | True    | public    | http
| 7bc9750347287  |           |              |               |         |           | .1.2
| badd02         |           |              |               |         |           | nt
| 3ea4a0c0b01c4  | RegionOne | neutron      | network       | True    | public    | http
| b349fb004dba9  |           |              |               |         |           | .1.2
| da7555         |           |              |               |         |           | twork
| 6a33567ce5f74  | RegionOne | nova         | compute       | True    | public    | http
| db4a8dd9b27bb  |           |              |               |         |           | .1.2
| 225469         |           |              |               |         |           | /v2.
| 6a3959c727574  | RegionOne | nova_legacy  | compute_legac | True    | public    | http
| 462b1065e078c  |           |              | y             |         |           | .1.2
| 5e48f8         |           |              |               |         |           | /v2/
|                |           |              |               |         |           | id)s
| 8c13f4c9877f4  | RegionOne | cinder       | block-storage | True    | public    | http
| 305841c320933  |           |              |               |         |           | .1.2
| 07d760         |           |              |               |         |           | v3/$
|                |           |              |               |         |           | d)s
| a14f63c9934b4  | RegionOne | glance       | image         | True    | public    | http
| 929bf8f42b886  |           |              |               |         |           | .1.2
| 7a5705         |           |              |               |         |           |
| d2b8e4506e9d4  | RegionOne | cinderv3     | volumev3      | True    | public    | http
| 45eb754d77fa3  |           |              |               |         |           | .1.2
| b172da         |           |              |               |         |           | v3/$
|                |           |              |               |         |           | d)s
+----------------+-----------+--------------+---------------+---------+-----------+-----
guest@ubuntuservervm:~/devstack$ _
```

Finally, navigate to http://your_vm_ip_address/dashboard to see the Openstack Dashboard home page. You can log into Openstack from here using "admin" and the password you chose when setting up.



Powering Off Your VM Correctly

Powering off your VM and then powering it back on can potentially cause issues with the services running in your OpenStack environment, as they may not automatically restart or might not be in the correct state upon reboot. When you power off your VM, ensure you perform a graceful shutdown using the shutdown command to prevent any abrupt termination of services. After powering it back on, verify that the network configuration remains consistent, particularly the IP address if you're using a bridged adapter. You will need to restart Devstack services manually. Navigate to your Devstack directory and run the ./stack.sh script to start all necessary services. Verify the status of these services using the openstack service list command and ensure that Apache and Keystone are running properly by checking their statuses with systemctl. To streamline this process, you can create a startup script that automates the restarting of Devstack services. Create a script file with the necessary commands, make it executable, and run it each time you power on the VM. By following these steps, you can ensure that your OpenStack environment restarts correctly, avoiding any potential issues related to service states and network configurations.

Automated Openstack Launch

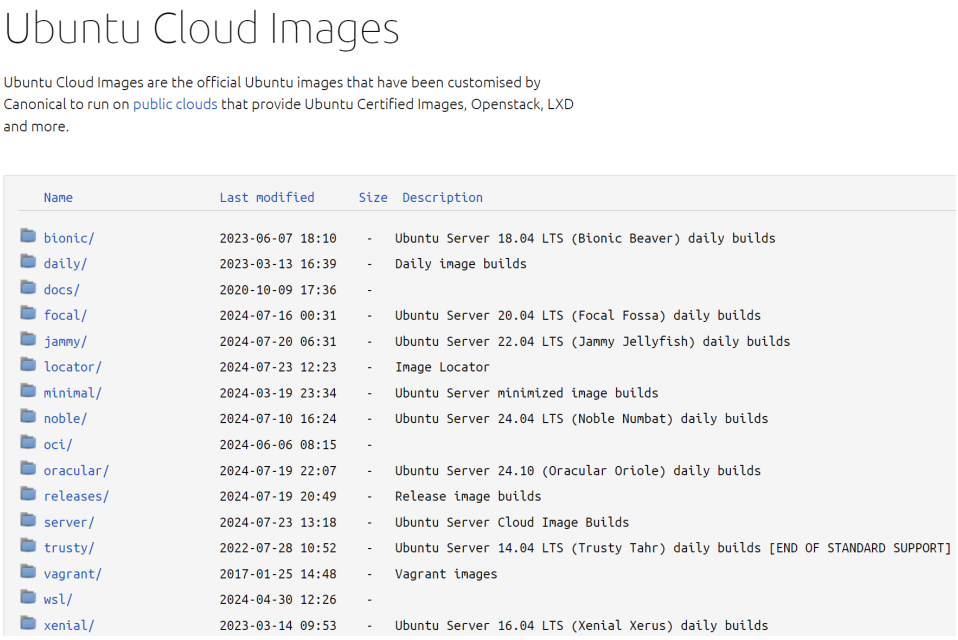
You may find it helpful to automate the openstack launch through the VM by creating a script. If you are using the dynamic IP address make sure your script can get the IP properly before running ./stack.sh.

Creating an Apache Web Server and Hosting a Publicly Accessible Website on Openstack.

Once you are logged into your Openstack Dashboard hosted from your Linux Server VM, you can navigate to Compute > Images. Right now there is only one Image and it is not the correct OS for our Apache Server. We will upload another Ubuntu Image to be used instead.

QCOW2 images are best for cloud resources.

To download a QCOW2 image suitable for your OpenStack instance, it's best to use a current LTS (Long Term Support) release as it is stable and well-supported. The latest LTS release at the time is Ubuntu 22.04 LTS (Jammy Jellyfish).

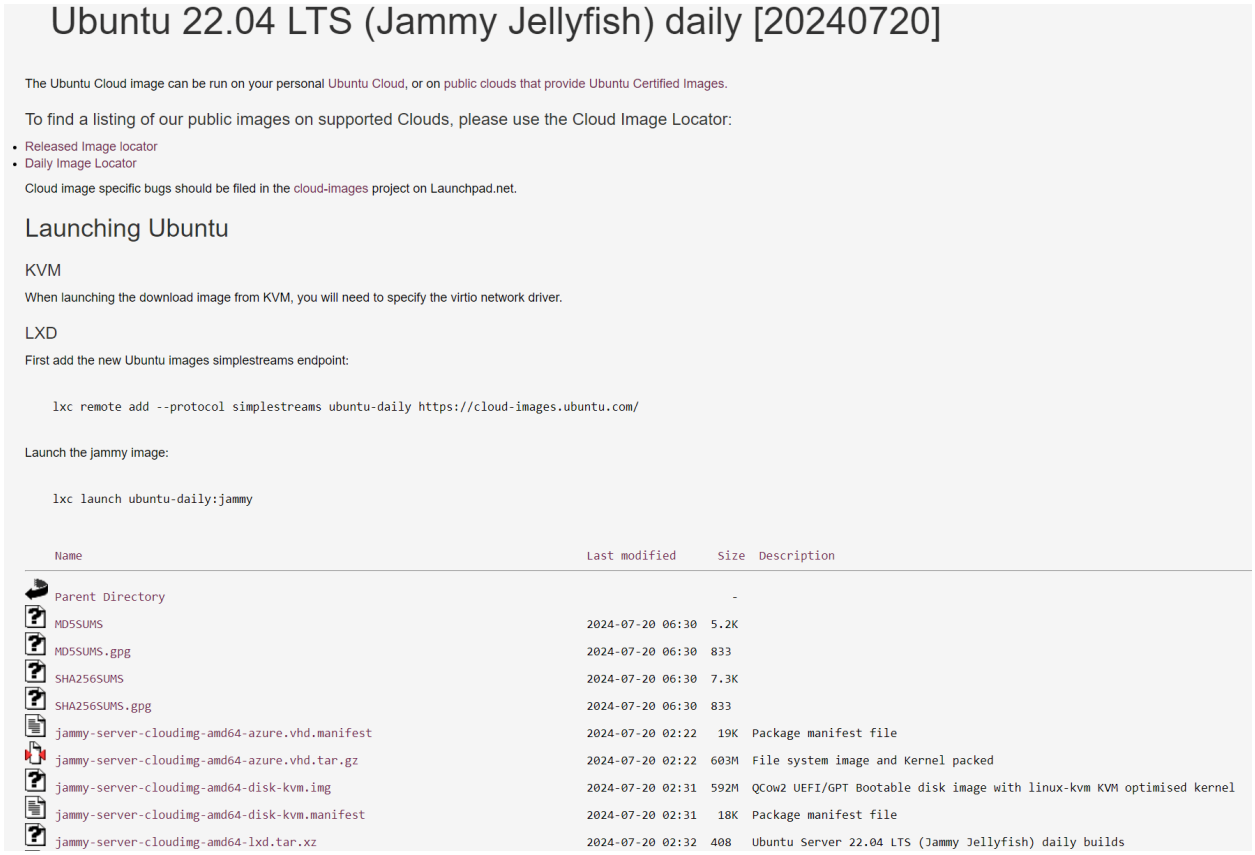


Ubuntu Cloud Images

Ubuntu Cloud Images are the official Ubuntu images that have been customised by Canonical to run on public clouds that provide Ubuntu Certified Images, Openstack, LXD and more.

Name	Last modified	Size	Description
bionic/	2023-06-07 18:10	-	Ubuntu Server 18.04 LTS (Bionic Beaver) daily builds
daily/	2023-03-13 16:39	-	Daily image builds
docs/	2020-10-09 17:36	-	
focal/	2024-07-16 00:31	-	Ubuntu Server 20.04 LTS (Focal Fossa) daily builds
jammy/	2024-07-20 06:31	-	Ubuntu Server 22.04 LTS (Jammy Jellyfish) daily builds
locator/	2024-07-23 12:23	-	Image Locator
minimal/	2024-03-19 23:34	-	Ubuntu Server minimized image builds
noble/	2024-07-10 16:24	-	Ubuntu Server 24.04 LTS (Noble Numbat) daily builds
oci/	2024-06-06 08:15	-	
oracular/	2024-07-19 22:07	-	Ubuntu Server 24.10 (Oracular Oriole) daily builds
releases/	2024-07-19 20:49	-	Release image builds
server/	2024-07-23 13:18	-	Ubuntu Server Cloud Image Builds
trusty/	2022-07-28 10:52	-	Ubuntu Server 14.04 LTS (Trusty Tahr) daily builds [END OF STANDARD SUPPORT]
vagrant/	2017-01-25 14:48	-	Vagrant images
wsl/	2024-04-30 12:26	-	
xenial/	2023-03-14 09:53	-	Ubuntu Server 16.04 LTS (Xenial Xerus) daily builds

Be sure to click the QCow2 img file and your download will begin.



Ubuntu 22.04 LTS (Jammy Jellyfish) daily [20240720]

The Ubuntu Cloud image can be run on your personal Ubuntu Cloud, or on public clouds that provide Ubuntu Certified Images.

To find a listing of our public images on supported Clouds, please use the Cloud Image Locator:

- Released Image locator
- Daily Image Locator

Cloud image specific bugs should be filed in the cloud-images project on Launchpad.net.

Launching Ubuntu

KVM

When launching the download image from KVM, you will need to specify the virtio network driver.

LXD

First add the new Ubuntu images simplestreams endpoint:

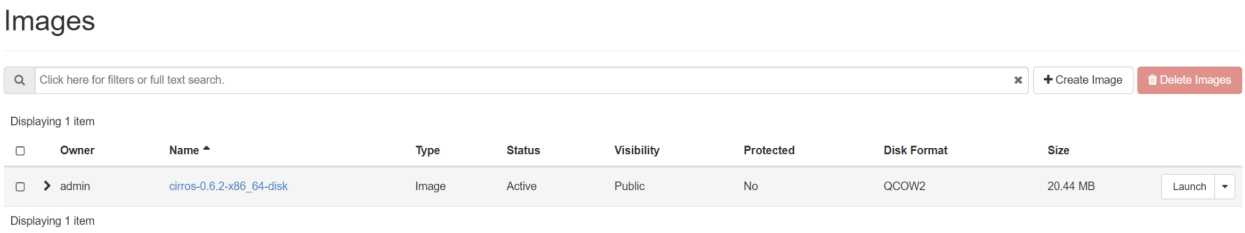
```
lxc remote add --protocol simplestreams ubuntu-daily https://cloud-images.ubuntu.com/
```

Launch the jammy image:

```
lxc launch ubuntu-daily:jammy
```

Name	Last modified	Size	Description
Parent Directory		-	
MD5SUMS	2024-07-20 06:30	5.2K	
MD5SUMS.gpg	2024-07-20 06:30	833	
SHA256SUMS	2024-07-20 06:30	7.3K	
SHA256SUMS.gpg	2024-07-20 06:30	833	
jammy-server-cloudimg-amd64-azure.vhd.manifest	2024-07-20 02:22	19K	Package manifest file
jammy-server-cloudimg-amd64-azure.vhd.tar.gz	2024-07-20 02:22	603M	File system image and Kernel packed
jammy-server-cloudimg-amd64-disk-kvm.img	2024-07-20 02:31	592M	QCow2 UEFI/GPT Bootable disk image with linux-kvm KVM optimised kernel
jammy-server-cloudimg-amd64-disk-kvm.manifest	2024-07-20 02:31	18K	Package manifest file
jammy-server-cloudimg-amd64-lxd.tar.xz	2024-07-20 02:32	408	Ubuntu Server 22.04 LTS (Jammy Jellyfish) daily builds

Once your download has finished, navigate back to your “Images” page on Openstack. Upload the QCow2 image we just downloaded.



Images

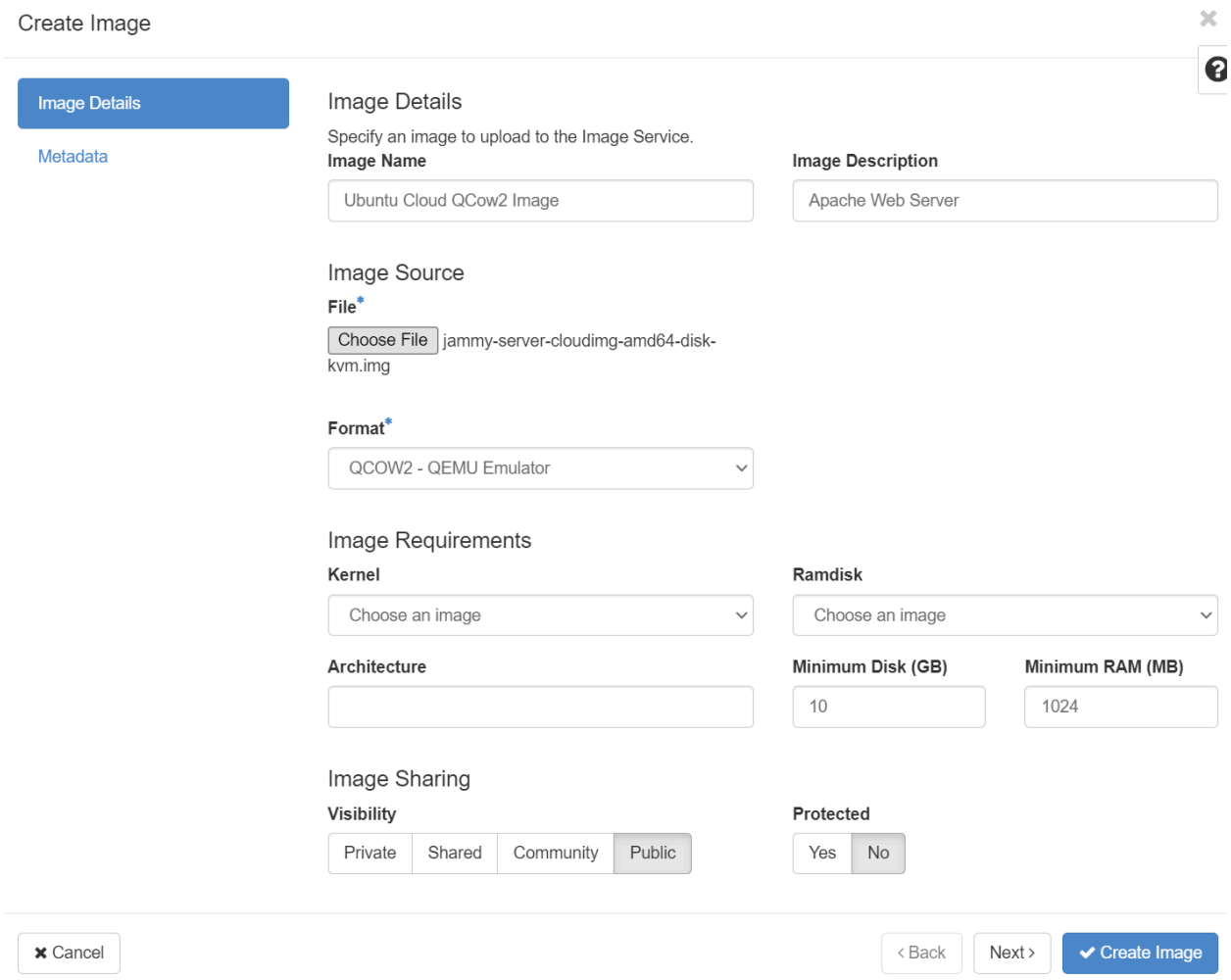
Click here for filters or full text search.

+ Create Image | Delete Images

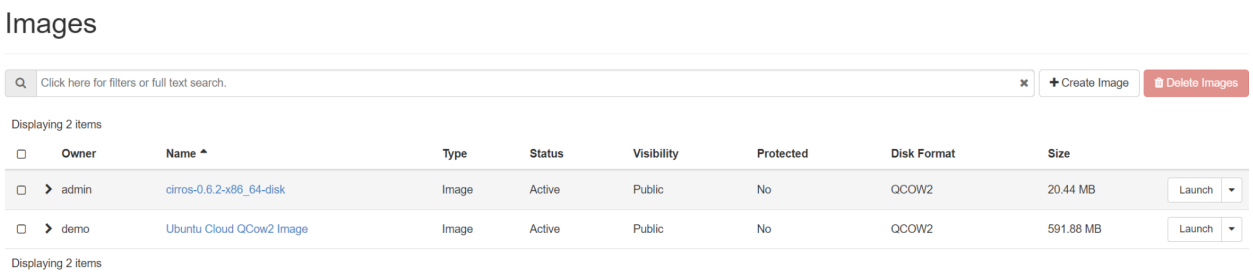
Displaying 1 item

Owner	Name	Type	Status	Visibility	Protected	Disk Format	Size	
admin	cirros-0.6.2-x86_64-disk	Image	Active	Public	No	QCOW2	20.44 MB	Launch

Displaying 1 item



After some time the completed Images list will contain your new image.



Select “Create Image”.

Once the image uploads, we will create a new instance using it. Navigate to “Instances” and click “Launch Instance”. Choose the QCow2 file we just downloaded.

Launch Instance

- Details
- Source *
- Flavor *
- Networks *
- Network Ports
- Security Groups
- Key Pair
- Configuration
- Server Groups
- Scheduler Hints
- Metadata

Please provide the initial hostname for the instance, the availability zone where it will be deployed, and the instance count. Increase the Count to create multiple instances with the same settings.

Project Name

demo

Instance Name *

Apache Web Server Instance

Description

Apache Web Server

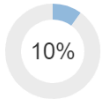
Availability Zone

nova

Count *

1

Total Instances
(10 Max)



0 Current Usage
1 Added
9 Remaining

Cancel | Back | Next > | Launch Instance

- Details
- Source
- Flavor *
- Networks *
- Network Ports
- Security Groups
- Key Pair
- Configuration
- Server Groups
- Scheduler Hints
- Metadata

Instance source is the template used to create an instance. You can use an image, a snapshot of an instance (image snapshot), a volume or a volume snapshot (if enabled). You can also choose to use persistent storage by creating a new volume.

Select Boot Source

Image

Create New Volume

Yes | No

Volume Size (GB) *

10

Delete Volume on Instance Delete

Yes | No

Allocated

Displaying 1 item

Name	Updated	Size	Format	Visibility
> Ubuntu Cloud QCow2 Image	7/23/24 2:01 PM	591.88 MB	QCOW2	Public

Displaying 1 item

Available 1

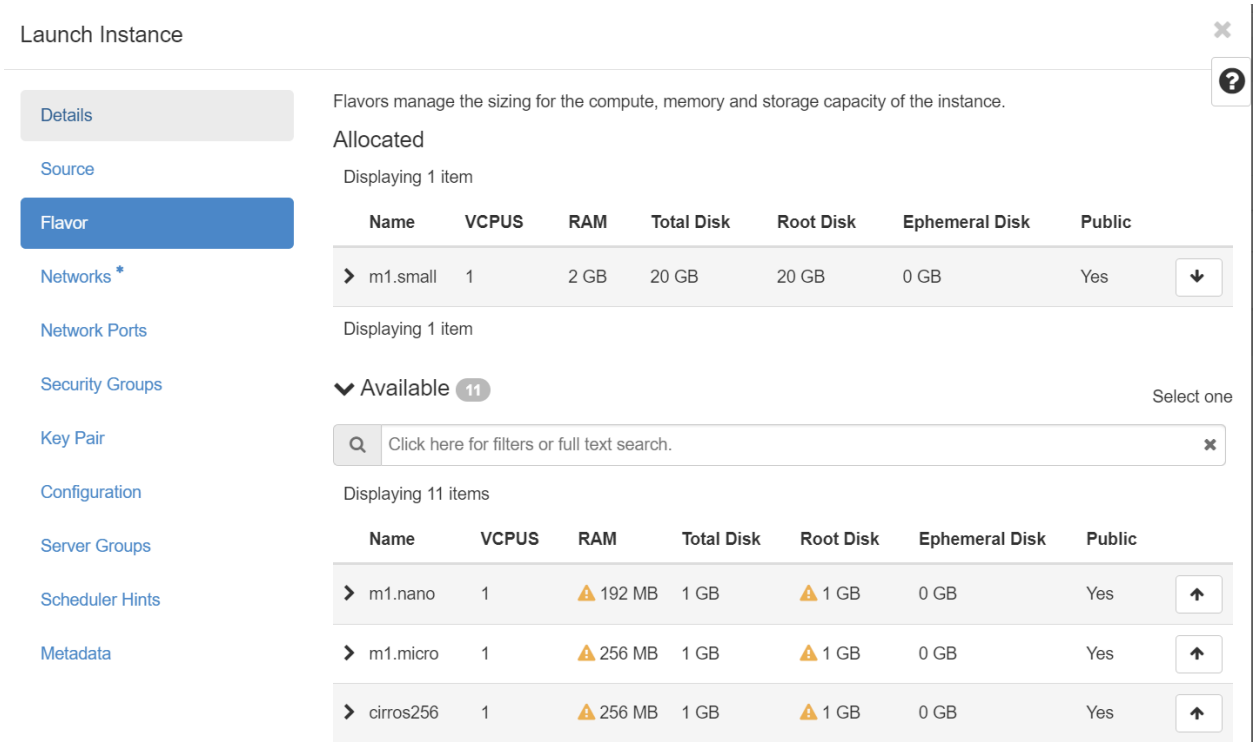
Select one

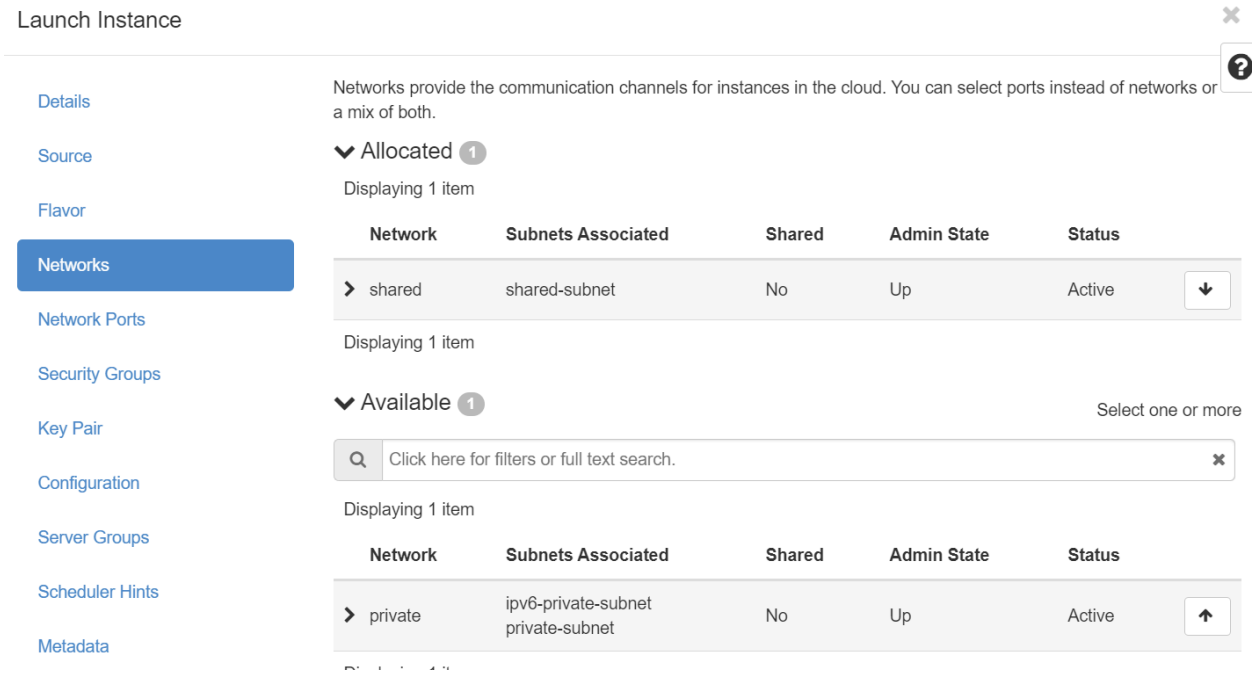
Click here for filters or full text search.

Displaying 1 item

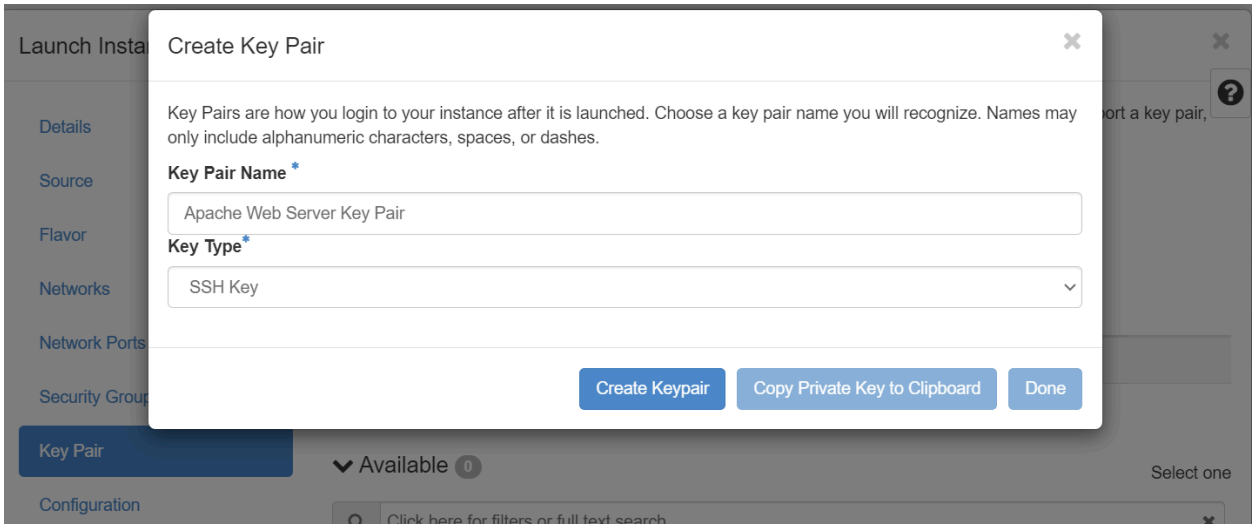
Name	Updated	Size	Format	Visibility
> cirros-0.6.2-x86_64-disk	7/23/24 1:20 AM	20.44 MB	QCOW2	Public

Displaying 1 item





Now we must create a key pair to ssh into our instance once it is running. Copy the newly created key into a .pem file on your local machine.



Launch Instance

- Details
- Source
- Flavor
- Networks
- Network Ports
- Security Groups
- Key Pair
- Configuration
- Server Groups
- Scheduler Hints
- Metadata

A key pair allows you to SSH into your newly created instance. You may se
or generate a new key pair.

+ Create Key Pair | Import Key Pair

Allocated

Displaying 1 item

Name	Type	Fingerprint
Apache Web Server Key Pair	ssh	10:f2:31:44:1e:59:fe:

Displaying 1 item

Available 0

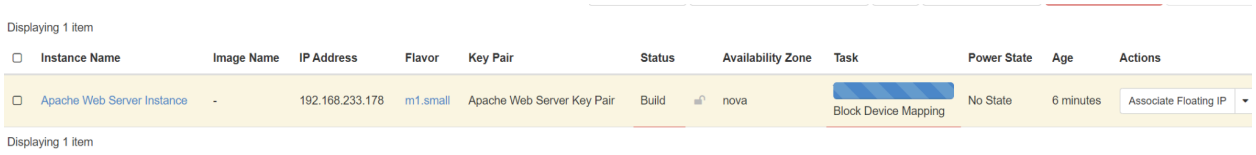
Click here for filters or full text search.

Displaying 0 items

Name	Type	Fingerpri
		No items to display.

Displaying 0 items

Once you launch the instance, it will take some time to create.



Once all the tasks complete, launch the Instance.

If the Instance fails to create, you may need to allocate more resources to it. Also check the status of the Openstack services in our Linux Server VM.

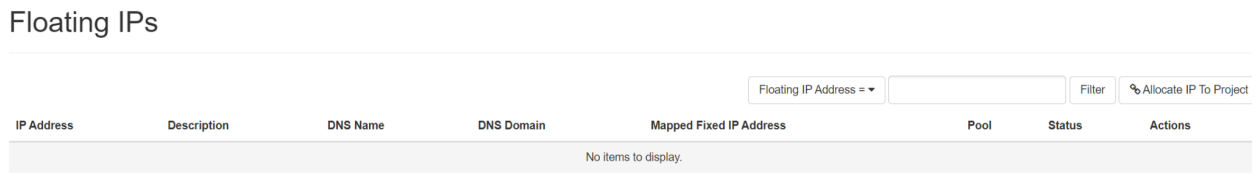
Once the instance is live, we can ssh into the instance. Be sure to reference the path to the key pair .pem file we created.

Run *ssh -i /path/to/your-key.pem ubuntu@<instance_ip>*.
Next, update the package list with *sudo apt update*.
Then install apache with *sudo apt install apache2 -y*.
Use *sudo systemctl start apache2 && sudo systemctl enable apache2* to enable apache.
Open a web browser on the local network and navigate to your instance's IP address. You should see the Apache default page.

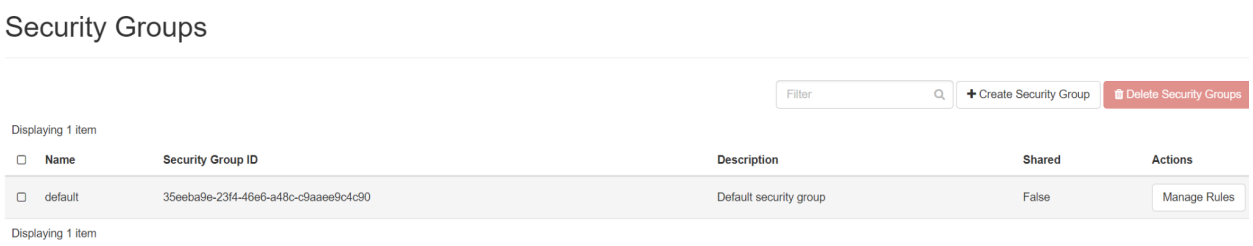
http://<instance_ip>

Make the Instance Accessible from outside the local network.

First we must assign a floating IP address to our network. In Openstack, navigate to Projects > Network > Floating IP. Allocate a new Floating IP and assign it to our new instance.



Next, update the security rules to allow traffic on necessary ports. These are in Project > Network > Security Groups.



Navigate to http://<floating_ip> outside of the local Network to show you can connect to our site online.

Finishing Up: Review and Next Steps

Congratulations, we have completed our Cloud Web Server. Here is an overview of what we have done.

- We created a Virtual Machine using VirtualBoxVM and Linux Server .ISO image.
- We changed our network type to Bridged Adapter
- We installed DevStack, and Openstack on our Linux Server. This automatically installed MariaDB on our server as well.
- We created a new instance on openstack using QCow2 ubuntu image.
- We installed Apache Web Server on our new instance.
- We accessed our Apache Server from outside the local network.

Advantages of Using OpenStack

OpenStack offers significant advantages, primarily due to its scalability, flexibility, and cost efficiency. It can scale horizontally to accommodate growing workloads, allowing you to add more compute nodes, storage, and network resources as needed. Its flexibility is unmatched, supporting a wide range of hypervisors such as KVM, VMware, and Hyper-V, as well as various storage backends like Ceph and Swift, and advanced networking technologies like Neutron and SDN. As an open-source platform, OpenStack eliminates licensing fees, making it a cost-effective solution for building private clouds and reducing dependency on public cloud providers.
Another key advantage of OpenStack is its robust community support. The platform benefits from a large and active community that provides extensive documentation, forums, and regular updates, ensuring it evolves with technological advancements. OpenStack's multi-tenancy features offer isolation and security for multiple projects and users, making it ideal for organizations with different departments or service providers offering cloud services to multiple clients. Additionally, OpenStack integrates seamlessly

with automation and orchestration tools like Ansible, Terraform, and Heat, simplifying the deployment and management of complex applications and services.

Cool Things You Can Do with OpenStack

With OpenStack, you can automate infrastructure management using APIs and orchestration tools, streamlining the provisioning and management of resources. This capability allows you to deploy complex environments using Heat templates or Terraform scripts. OpenStack also enables you to host private cloud environments for internal applications, databases, and services, providing control over data privacy and security while reducing costs compared to public cloud services.
In the realm of DevOps and continuous integration/continuous deployment (CI/CD), OpenStack excels by facilitating the setup of development, testing, and production environments. It integrates with CI/CD tools like Jenkins to automate software development pipelines. For high-performance computing (HPC) needs, OpenStack supports scientific research, simulations, and other HPC workloads, leveraging powerful compute nodes and specialized hardware. Additionally, OpenStack is suitable for big data and analytics applications, allowing you to deploy Hadoop or Spark clusters to process and analyze large datasets efficiently.

Specific Use Cases and Examples

OpenStack is versatile and can be used for a variety of specific use cases. For web hosting, it supports multiple websites and web applications using servers like Apache and Nginx, and can handle traffic spikes with load balancers and auto-scaling features. It also provides isolated development environments for different teams, enabling rapid setup and teardown of test environments. E-commerce platforms benefit from OpenStack's scalability and high availability, ensuring optimal performance during peak shopping seasons.
In the corporate world, OpenStack supports Virtual Desktop Infrastructure (VDI), allowing employees to work remotely and use their own devices (BYOD), while centralizing desktop management and enhancing security. For machine learning and AI applications, OpenStack provides environments for training models, utilizing GPUs for accelerated computing and efficient model training. These use cases illustrate OpenStack's capability to meet diverse business and technical requirements.

Conclusion

In conclusion, OpenStack is a powerful and flexible platform for managing cloud infrastructure, offering both cost and operational efficiencies. By leveraging OpenStack, organizations can build scalable, automated, and secure cloud environments tailored to their specific needs. The platform's extensive possibilities include hosting web applications, implementing advanced AI and big data solutions, and supporting DevOps

practices. With continuous advancements and a strong community backing, OpenStack remains a leading choice for private and hybrid cloud deployments, making it an invaluable tool for modern IT infrastructure management.

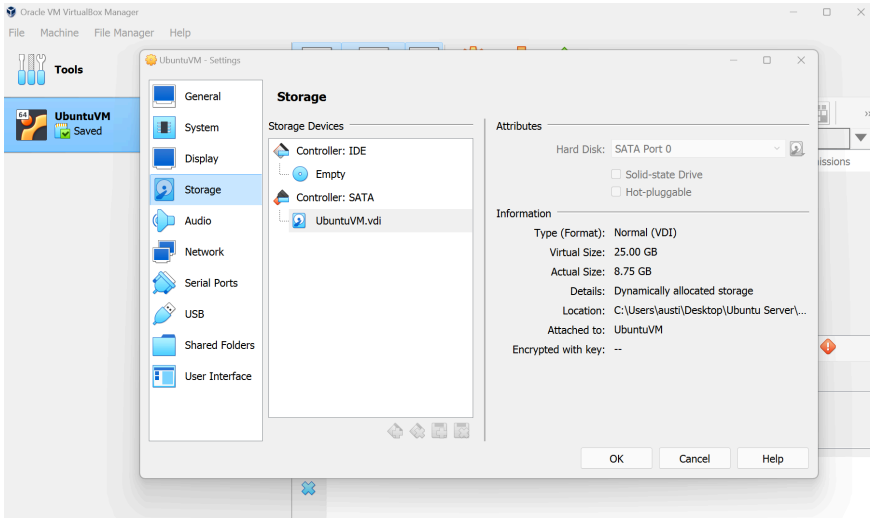
How to resize Openstack server on a LinuxServer VM: DO NOT DO UNLESS CALLED FOR!

I Decided to include this section for education purposes. Resizing your VM can be necessary for many reasons, but notably it is used to allow the VM to fit different hardware configurations and therefore take advantage of the hardware.

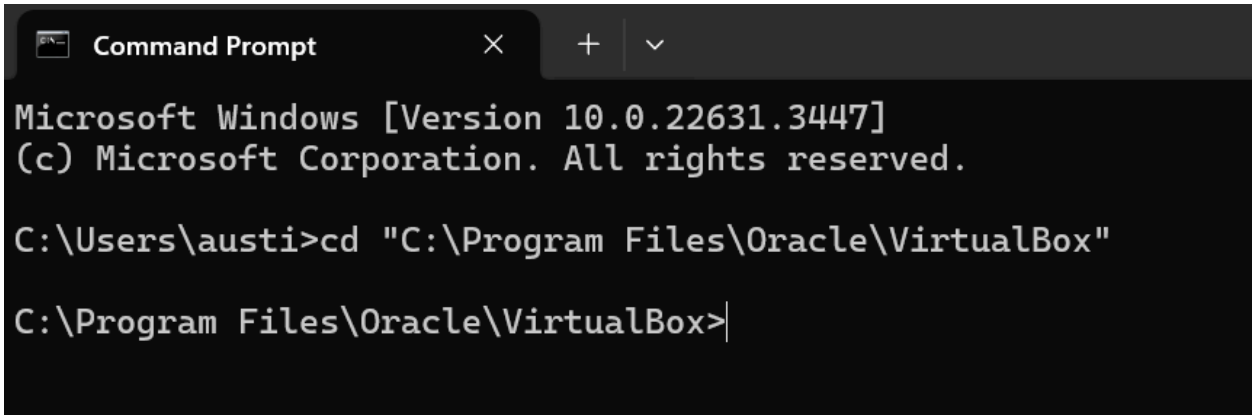
Once you open the VirtualBox GUI, you can navigate to the storage settings of your LinuxServer VM.

Copy the Location of the file on your system.

Here we see the “Virtual Size” is 25GB and “Actual Size” is 8.75GB.



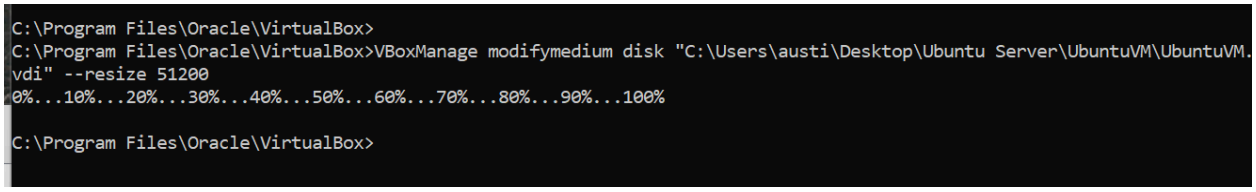
Open the command prompt as Administrator and navigate to the location of the VM.



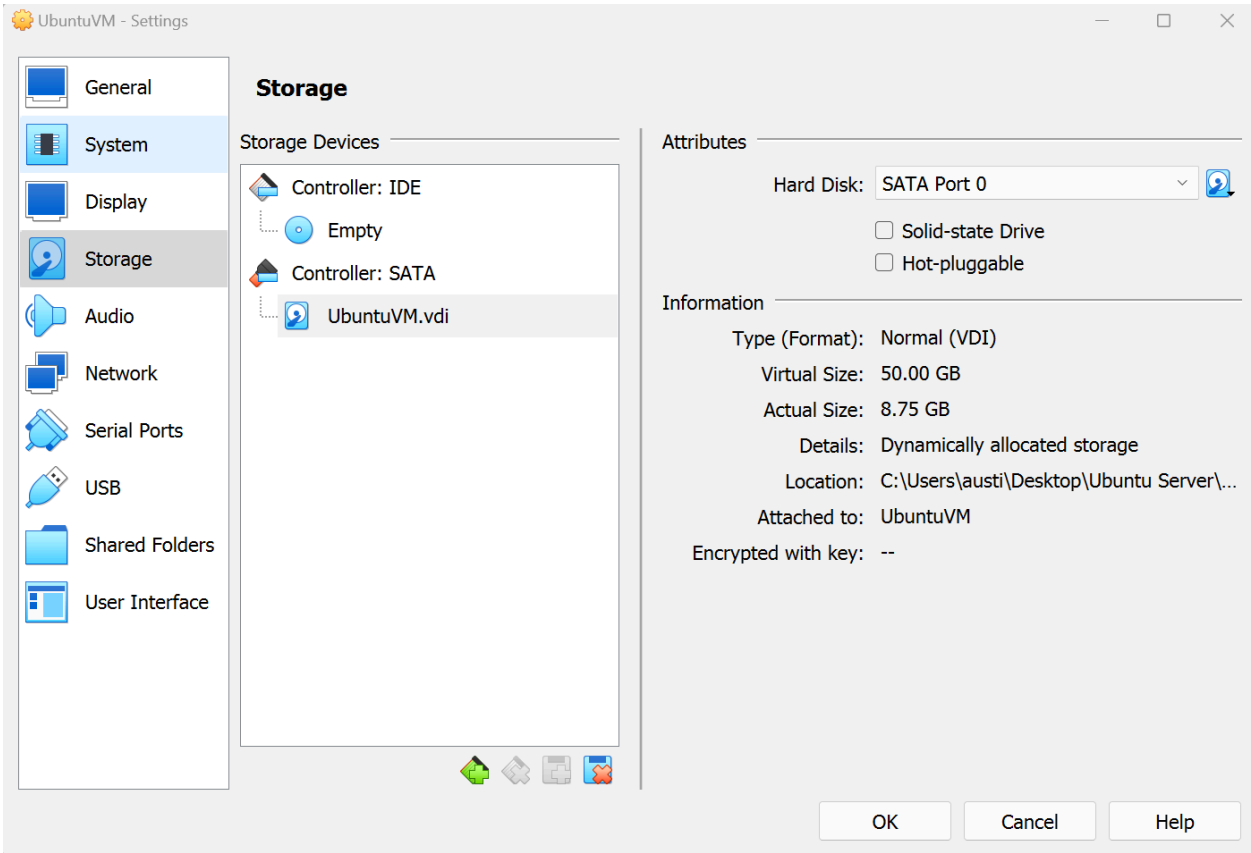
Use the following command to resize the VM. Specify the new size in megabytes. For example, if you want to resize the disk to 50 GB, you would enter 51200 MB (50 x 1024).

Command used:

VBoxManage modifymedium disk "C:\Users\austi\Desktop\Ubuntu Server\UbuntuVM\UbuntuVM.vdi" --resize 51200



The changes will then be reflected in the VirtualBox GUI:



To modify the partitions safely, it's advisable to boot your VM using a live CD tool like GParted Live.

First log into your VM and run "uname -m" to determine your system architecture. Mine shows 64bit.

```
Last login: Thu Apr 11 18:15:18 UTC 2024 on tty1
kingsland45@ubuntuvm:~$ uname -m
x86_64
kingsland45@ubuntuvm:~$ _
```

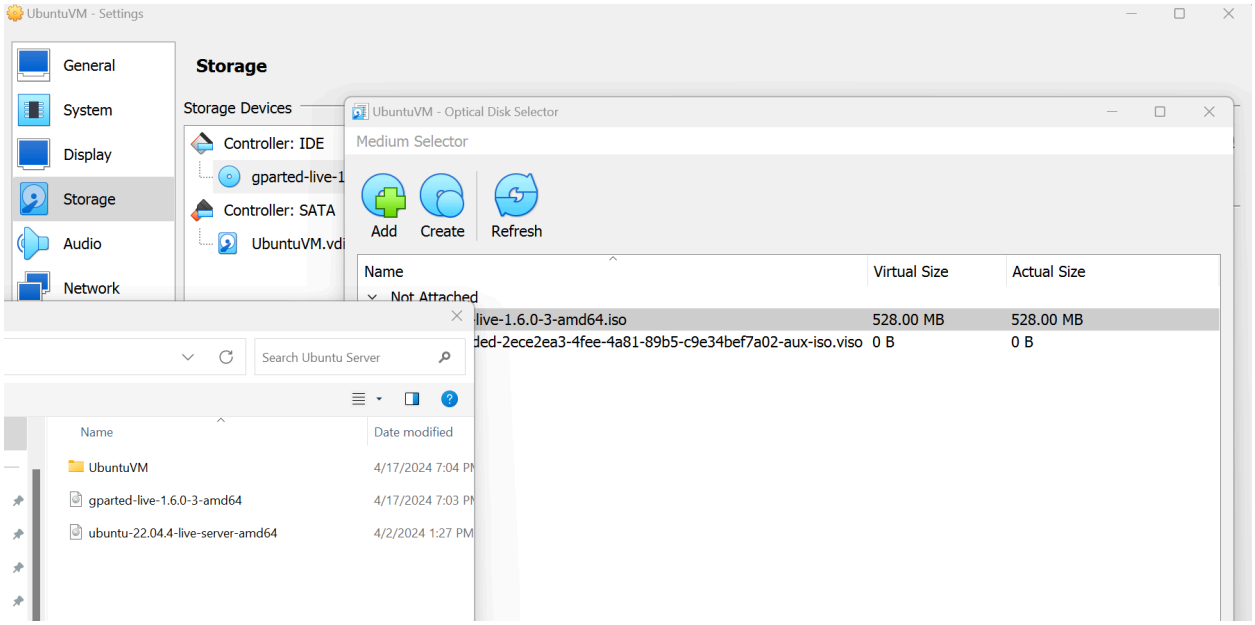
Download GParted Live ISO which matches your system architecture.

https://gparted.org/download.php

Attach the ISO to the VM:

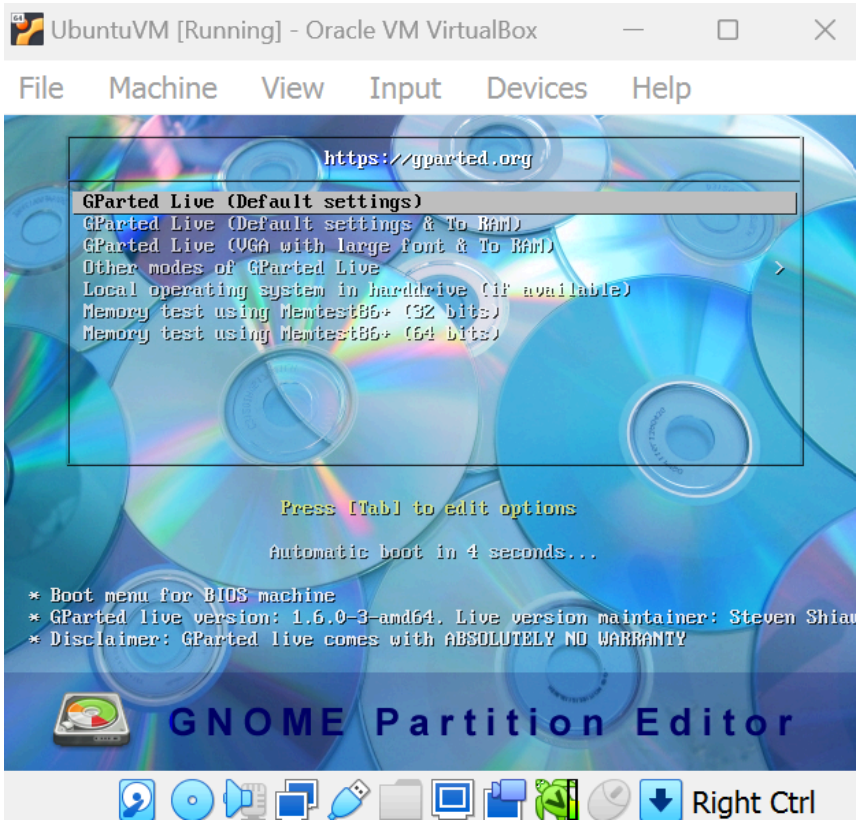
Open VirtualBox Manager.
Select your Ubuntu VM.
Go to "Settings" → "Storage".

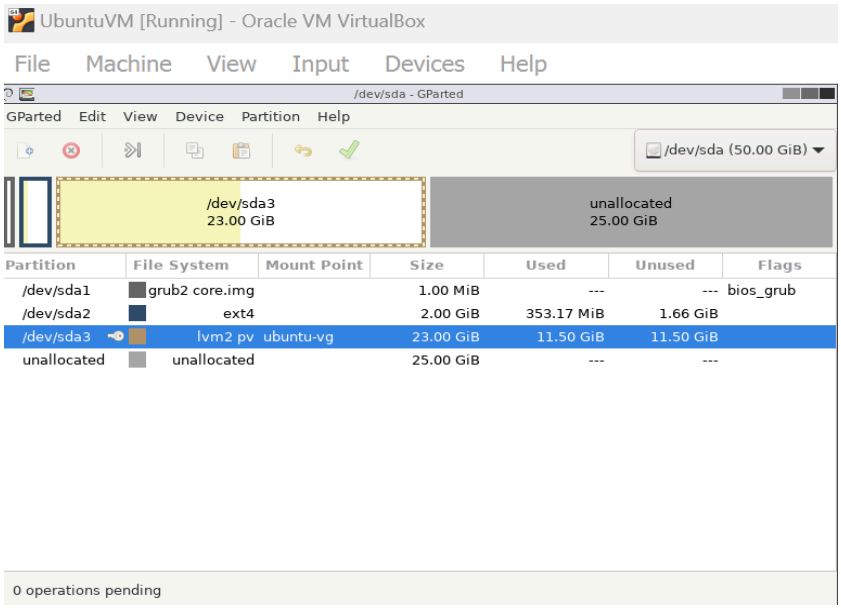
Under "Controller: IDE" (or SATA), click on the "Empty" CD icon.
On the right side, next to "Optical Drive", click the CD icon and choose "Choose a disk file".
It is usually best to mount to “Secondary Device 0”, if it is available.
Select the GParted Live ISO file you downloaded and click "OK".



Boot the VM Using GParted Live:

Start the VM. You might need to press a key (F12) to bring up the boot menu and select the CD/DVD drive to boot from the GParted Live ISO.



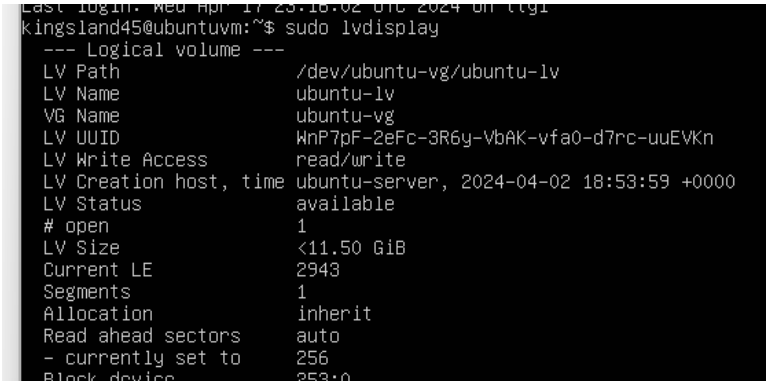


Once you use GParted to resize the lvm2 ubuntu-vg, the unallocated space will go to 1 mb.

Now we need to change the logical size in the VM. Start the VM using its normal ISO.

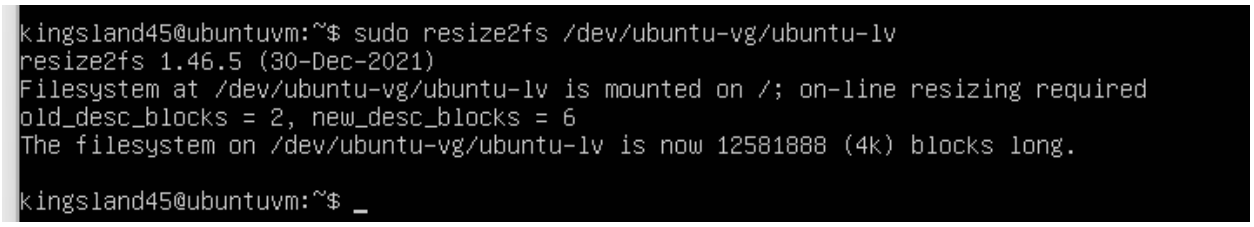
You can change this in the VM settings under the controller IDE options when you click the icon to the right of the optical drive.

Once logged in, use “lvdisplay” to show the name and size.



Next, use “sudo lvextend -l +100%FREE /dev/ubuntu-vg/ubuntu-lv” to extend the logical size.

Then use “sudo resize2fs /dev/ubuntu-vg/ubuntu-lv” to resize the file system.



Run “df -h” to look at the details.

```
kingsland45@ubuntuvm:~$ sudo lvdisplay
  --- Logical volume ---
  LV Path                /dev/ubuntu-vg/ubuntu-lv
  LV Name                ubuntu-lv
  VG Name                ubuntu-vg
  LV UUID                WnP7pF-2eFc-3R6y-VbAK-vfaO-d7rc-uuEVKn
  LV Write Access        read/write
  LV Creation host, time ubuntu-server, 2024-04-02 18:53:59 +0000
  LV Status              available
  # open                 1
  LV Size                <48.00 GiB
  Current LE             12287
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:0
```

```
kingsland45@ubuntuvm:~$ df -h
Filesystem                         Size  Used Avail Use% Mounted on
tmpfs                              197M  1.3M  196M   1% /run
/dev/mapper/ubuntu--vg-ubuntu--lv   48G   11G   35G  24% /
tmpfs                              982M     0  982M   0% /dev/shm
tmpfs                              5.0M     0  5.0M   0% /run/lock
tmpfs                              982M     0  982M   0% /run/qemu
/dev/sda2                          2.0G  252M  1.6G  14% /boot
tmpfs                              197M  4.0K  197M   1% /run/user/1000
kingsland45@ubuntuvm:~$ _
```

Apache: Reliably Determine the Server's Fully Qualified IP Using Port Forwarding (No Bridged Adapter).

This is not needed for our set up.

Port Forwarding and IP Address Change (Router Configuration)

Complete Guide to Setting Up a Static IP and Remote Access for OpenStack

Step 1: Change IP Address to Static

1. Open the Network Interfaces Configuration File:

```bash
sudo nano /etc/network/interfaces
```

2. Add the Static IP Configuration:

```ini
auto enp0s3
iface enp0s3 inet static
    address 192.168.1.101
    netmask 255.255.255.0
    gateway 192.168.1.1
    dns-nameservers 8.8.8.8 8.8.4.4
```

3. Save and Exit:

4. **Restart the Network Interface:**

```bash
sudo systemctl restart networking
```

If this command fails, use:

```bash
sudo systemctl restart NetworkManager
```

5. **Verify the IP Address:**

```bash
ip addr show enp0s3
```

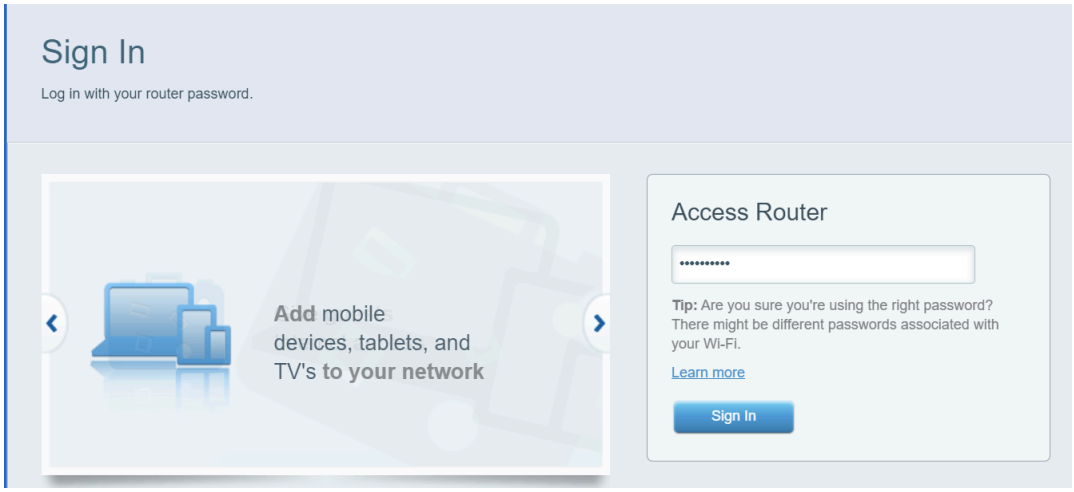
Ensure that `enp0s3` has the IP address `192.168.1.101`.

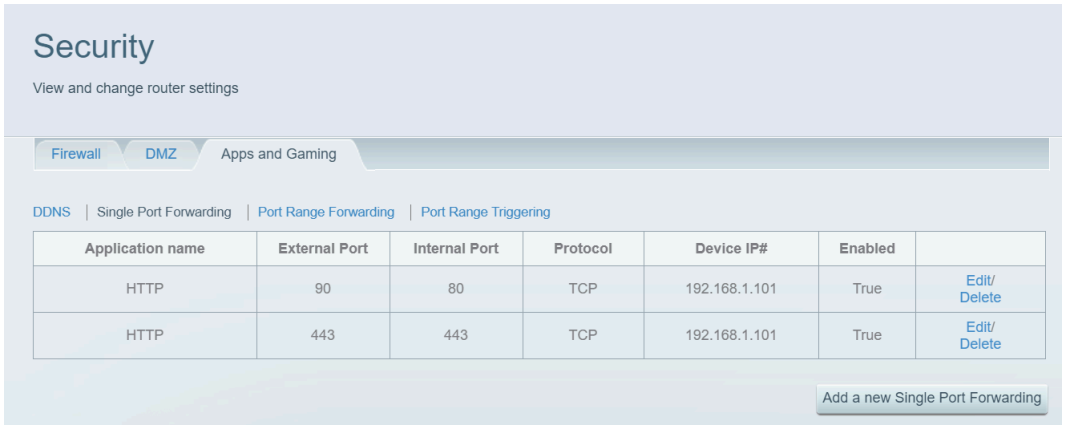
Router passwords are typically “admin”.

When setting up port forwarding

Step 2: Configure Port Forwarding on the Router

1. **Log in to Your Router’s Configuration Page:**
   - Open a web browser and navigate to your router's IP address (usually `192.168.1.1` or `192.168.0.1`).
   - Enter your username and password to log in.
2. **Navigate to Port Forwarding Section:**
   - Look for sections named `Security`, `Advanced Settings`, `Applications & Gaming`, or similar.
3. **Add Port Forwarding Rules:**





Step 3: Verify Firewall Rules on the VM

1. **Check UFW Status:**

```bash
sudo ufw status
```

2. **Allow HTTP and HTTPS Traffic:**

```bash
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw enable
```

Step 4: Verify Apache Configuration

1. Ensure Apache is Configured to Listen on Ports 80 and 443:

   ```bash
   sudo nano /etc/apache2/ports.conf
   ```

   Ensure it contains:

   ```apache
   Listen 80
   Listen 443
   ```

2. Restart Apache:

   ```bash
   sudo systemctl restart apache2
   ```

Step 5: Test Local Access

1. Open a Web Browser on a Device Within Your Local Network:
   - Navigate to `http://192.168.1.101`.

Step 6: Test Remote Access

1. Find Your Public IP Address:

   ```bash
   curl ifconfig.me
   ```

2. Open a Web Browser on a Device Outside Your Local Network:
   - Navigate to `http://your_public_ip_address` (replace with your actual public IP).

3. Troubleshooting Remote Access (if necessary):
   - Check Firewall Rules:

     ```bash
     sudo ufw status
     ```

   - Verify Router Port Forwarding Rules:
     - Ensure rules are correctly forwarding ports 80 and 443 to `192.168.1.101`.
   - Check for ISP Restrictions: