



Lab 1.1. Lab Environment Setup - GCE and VirtualBox VMs

The lab exercises included in this course can be practiced on a system with a minimal hardware profile of up to 2 CPU cores, 1 GB memory, 10 GB disk, and internet connection. We encourage the use of a cloud provider such as Google Cloud Platform (GCP) or Amazon Web Services (AWS) for the provisioning of the lab environment. However, if the cloud is not easily accessible, the lab environment can also be provisioned with a local hypervisor such as VirtualBox. Next we will walk through setting up a [Free Tier](#) eligible cloud Virtual Machine with the Google Compute Engine (GCE) cloud service, followed by a guide to setup a local VirtualBox VM.

Google Compute Engine (GCE) VM

When accessing cloud resources, one of our top concerns is the security of our connection. For that reason we will be using a secure shell client together with a set of keys to establish a secure connection between our host system and the remote VM.

Considering a Ubuntu host machine, we will run the `ssh-keygen` tool to generate a set of keys. Our aim is to create a set of keys to allow for a non-root user login to the remote GCE VM. Our user will be named `student`, and the SSH keys will be generated only for this user. This simplifies the remote login process for the remote `student` user, by removing the necessity of typing passwords at the login prompt. Open a terminal on the host machine, and run the following command, and then press **Enter** for all defaults:

```
user@host:~$ ssh-keygen -C student
```

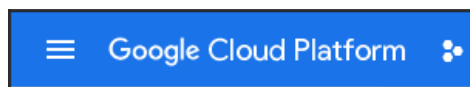
The keys are generated in the the `/home/user/.ssh/` directory:

```
user@host:~$ ls -l /home/user/.ssh/
...
-rw----- 1 user user 2590 Aug  9 17:06 id_rsa
-rw-r--r-- 1 user user  561 Aug  9 17:06 id_rsa.pub
...
```

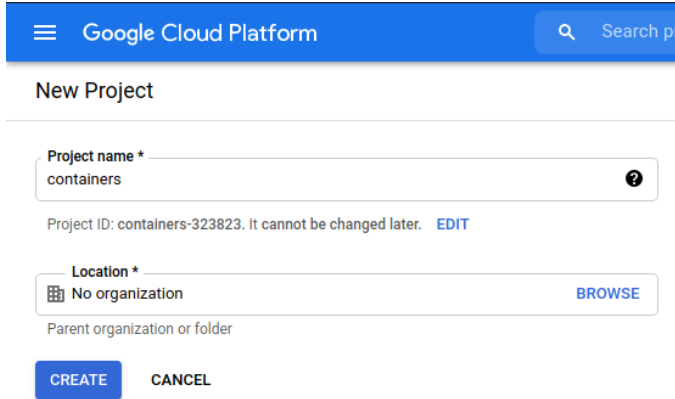
This guide assumes that a Google Cloud Platform (GCP) account has been set up already – although a credit card may be requested during the sign-up process, it should not be charged for the usage of a Free Tier service, as long as the usage limits are not exceeded. The setup is targeting a cloud VM instance which is currently part of Google Cloud Platform’s Free Tier offering – resources that are [free of charge](#) with a specified monthly usage limit. Please check back periodically to stay up to date with the cloud provider’s definition of [Free Tier](#), qualifying services, and [usage limits](#).

NOTE: The following set of screenshots may have a different look on your system, since the cloud services are constantly evolving and features may change without notice.

Once logged into the GCP account, from the Console click the project menu at the right of the GCP **Navigation menu**, select **NEW PROJECT**:



On the new project page provide a new **Project name**, let's name it **containers**, and click **CREATE**:



Google Cloud Platform Search projects

New Project

Project name *
containers ?

Project ID: containers-323823. It cannot be changed later. [EDIT](#)

Location *
No organization [BROWSE](#)

Parent organization or folder

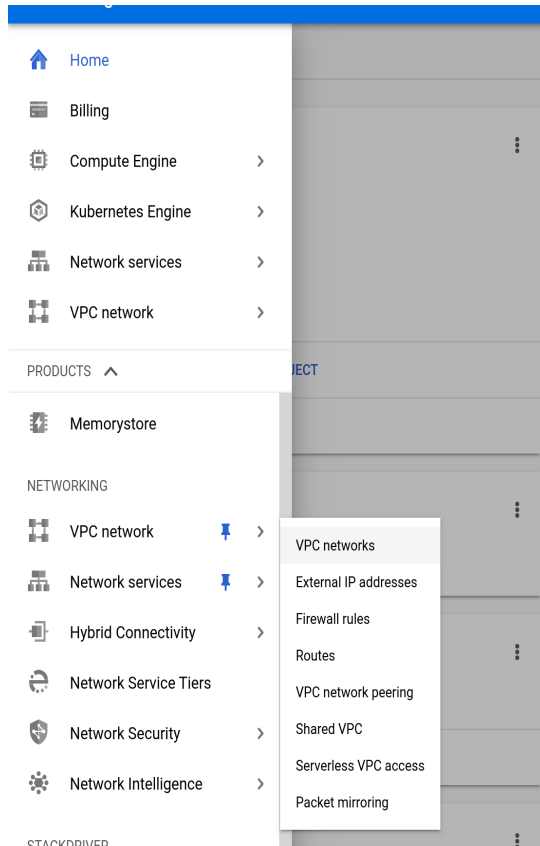
[CREATE](#) [CANCEL](#)

From the **Notifications** pop-up window click **SELECT PROJECT**, and the newly created **containers** project should show as the current project at the right of the GCP **Navigation menu**:



Google Cloud Platform containers

On the GCP Console select the **Navigation menu** from the top left corner. Scroll down to the **NETWORKING** section, select **VPC network**, then select **VPC networks** from the sub-menu:



If prompted, click **ENABLE**.

On the VPC network screen select **CREATE VPC NETWORK** at the top.
 Provide a name: **container-fundamentals-vpc**.
 Select **Automatic** subnet creation mode.

Create a VPC network

Name *
 container-fundamentals-vpc
Lowercase letters, numbers, hyphens allowed

Description

Subnets
 Subnets let you create your own private cloud topology within GCP. You can create a subnet in each region, or click Custom to create multiple subnets. [Learn more](#)

Subnet creation mode
☐ Custom
☒ Automatic

These IP address ranges will be assigned to each region. If you select Custom for subnet creation, it will be assigned an IP from the appropriate range.

Region	IP address range
asia-east1	10.140.0.0/20
asia-northeast1	10.146.0.0/20

Leave Firewall rules unchecked. We will create a firewall rule later.
 Select **CREATE** to finalize the creation of the VPC network.

Dynamic routing mode ?

☒ **Regional**
 Cloud Routers will learn routes only in the region.

☐ **Global**
 Global routing lets you dynamically learn routes from all regions.

DNS server policies could not be

Maximum transmission unit (MTU)
 1460

CREATE **CANCEL**

EQUIVALENT COMMAND LINE ▾

Select from the menu **Firewall**.

On the Firewall screen select **CREATE FIREWALL RULE**.

Provide a name: **allow-all-ingress-firewall-rule**.

VPC network

← Create a firewall rule

Firewall rules control incoming or outgoing traffic to an instance. By default, incoming traffic from outside your network is blocked. [Learn more](#)

Name *

allow-all-ingress-firewall-rule

Lowercase letters, numbers, hyphens allowed

Description

Logs

Turning on firewall logs can generate a large number of logs which can increase costs in Cloud Logging. [Learn more](#)

☐ On

☒ Off

Select from the Network dropdown list the VPC network created earlier:

container-fundamentals-vpc.

Priority: **1000**.

Direction of traffic: **Ingress**.

Action on match: **Allow**.

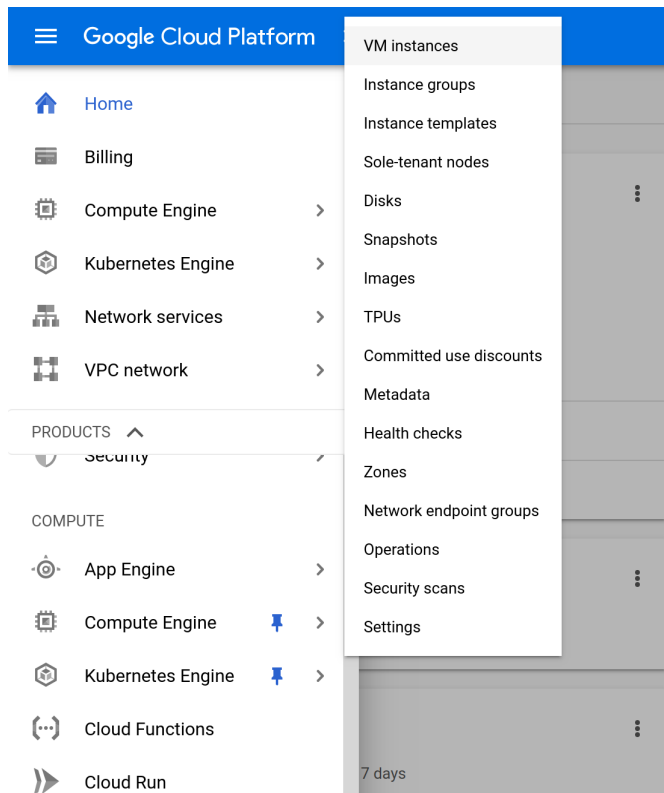
Targets: **All instances in the network**.

Source IP ranges: **0.0.0.0/0**

Protocols and ports: **Allow all**.

Select **CREATE** to finalize the creation of the Firewall rule.

On the GCP Console select the **Navigation menu** from the top left corner. Scroll down to the **COMPUTE** section, select **Compute Engine**, then select **VM instances** from the sub-menu.



On the VM instances screen select **CREATE INSTANCE**.

Provide a name: **ubuntu**.

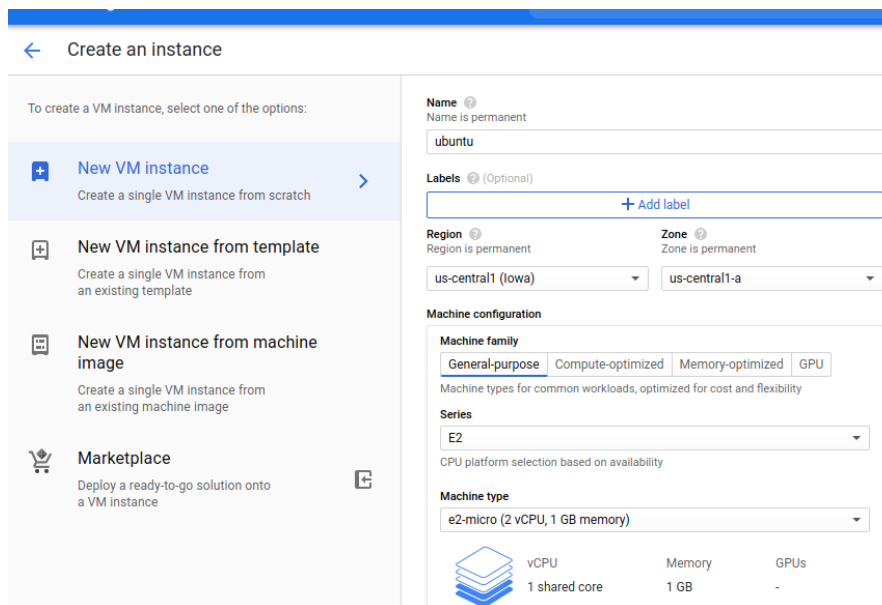
Select from the Region dropdown list the one of the regions supporting Free Tier VM instances (Oregon: us-west1, Iowa: us-central1, South Carolina: us-east1).

Select **General-purpose** Machine family.

Select Series **E2**.

Select from the Machine type dropdown list **e2-micro (2 vCPU, 1 GB memory)**.

Select to **Change** the Boot disk to modify the instance OS image.



← Create an instance

To create a VM instance, select one of the options:

- New VM instance**
Create a single VM instance from scratch
- New VM instance from template
Create a single VM instance from an existing template
- New VM instance from machine image
Create a single VM instance from an existing machine image
- Marketplace
Deploy a ready-to-go solution onto a VM instance

Name ⓘ
Name is permanent
ubuntu

Labels ⓘ (Optional)
+ Add label

Region ⓘ
Region is permanent
us-central1 (Iowa)

Zone ⓘ
Zone is permanent
us-central1-a

Machine configuration

Machine family
General-purpose | Compute-optimized | Memory-optimized | GPU
Machine types for common workloads, optimized for cost and flexibility

Series
E2
CPU platform selection based on availability

Machine type
e2-micro (2 vCPU, 1 GB memory)

vCPU	Memory	GPUs
1 shared core	1 GB	-

From the Boot disk page select the **Public images** tab, Operating system **Ubuntu**, Version **Ubuntu 20.04 LTS**.

Click **Select**.

Boot disk

Select an image or snapshot to create a boot disk, or attach an existing disk. Can't find what you're looking for?

Public images | Custom images | Snapshots | Existing disks

Operating system

Ubuntu

Version

Ubuntu 20.04 LTS

amd64 focal image built on 2021-08-20, supports Shielded VM features ⓘ

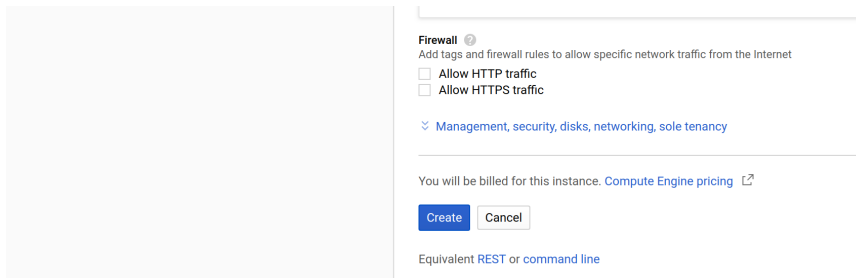
Boot disk type ⓘ

Balanced persistent disk

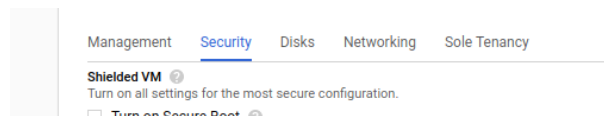
Size (GB) ⓘ

10

Expand the link **Management, security, disks, networking, sole tenancy**.



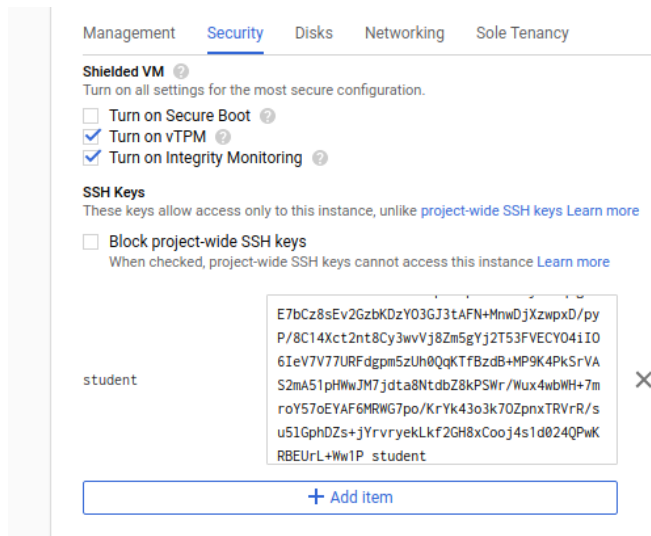
Select the **Security** tab:



On your host machine, locate the key pair generated earlier with the `ssh-keygen` tool, and visualize the content of the `/home/user/.ssh/id_rsa.pub` public key file:

```
user@host:~$ sudo cat /home/user/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDICI94kEU6KCI1b9/
...
KrYk43o3k7OZpnxTRVrR/su5lGphDZs+jYrvryekLkf2GH8xCooj4s1d024QPwKRBEUrL+
Ww1P student
```

Copy the entire content of the file, and paste it into the text box labeled **Enter public SSH key**. The user name **student** will immediately be recognized and extracted by the SSH key management system:



Management **Security** Disks Networking Sole Tenancy

Shielded VM ⓘ
Turn on all settings for the most secure configuration.

☐ Turn on Secure Boot ⓘ
☒ Turn on vTPM ⓘ
☒ Turn on Integrity Monitoring ⓘ

SSH Keys
These keys allow access only to this instance, unlike [project-wide SSH keys](#) [Learn more](#)

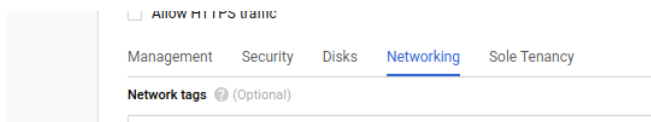
☐ Block project-wide SSH keys
When checked, project-wide SSH keys cannot access this instance [Learn more](#)

student

```
E7bCz8sEv2GzbKDzY03GJ3tAFN+MnwDjXzwpXD/py
P/8C14Xct2nt8Cy3wvVj8Zm5gYj2T53FVECY04110
61eV7V77URFdgpm5zUh0QqKTfBzdB+MP9K4PkSrVA
S2mAS1pHwJM7jda8NtdbZ8kPSWr/Wux4wbWH+7m
roY57oEYAF6MRWG7po/KrYk43o3k70ZpnxTRVrR/s
u51GphDZs+jYrvryekLkf2GH8xCooj4s1d024QPwK
RBEUrL+Ww1P student
```

[+ Add item](#)

Select the **Networking** tab:



☐ ALLOW HTTPS traffic

Management Security Disks **Networking** Sole Tenancy

Network tags ⓘ (Optional)

Select to edit the **Network interfaces** field.

From the Network dropdown list select VPC network created earlier and its associated Subnetwork.

Select **Done**.

The screenshot shows the 'Create an instance' dialog in the Google Cloud Platform console. The 'Network interface' panel is expanded, showing the following configuration:

- Network:** container-fundamentals-vpc
- Subnetwork:** container-fundamentals-vpc (10.128.0.0/20)
- Primary internal IP:** Ephemeral (Automatic)
- External IP:** Ephemeral
- Network Service Tier:** Premium (Current project-level tier, change) (Selected)
- IP forwarding:** Off
- Public DNS PTR Record:** Enable (Selected)

At the bottom of the panel, there is a '+ Add network interface' button and a 'Less' link.

Select **Create** to finalize the creation of the VM instance.

← Create an instance

Access scopes ⓘ

- ☒ Allow default access
- ☐ Allow full access to all Cloud APIs
- ☐ Set access for each API

Firewall ⓘ

Add tags and firewall rules to allow specific network traffic from the Internet

- ☐ Allow HTTP traffic
- ☐ Allow HTTPS traffic

Management Security Disks **Networking** Sole Tenancy

Network tags ⓘ (Optional)

Hostname ⓘ

Set a custom hostname for this instance or leave it default. Choice is permanent

container-fundamentals-instance-us-central1-a.c.stately-arbor-258201.internal

Network interfaces ⓘ

Network interface is permanent

container-fundamentals-vpc container-fundamentals-vp...

+ Add network interface

⤴ Less

You will be billed for this instance. [Compute Engine pricing](#) ⓘ

Create Cancel

Equivalent REST or command line

From the VM instances page we can select the newly created instance followed by the desired action from the top menu: Start, Stop, Reset, Delete, Suspend, Resume.

Running VM instance can be accessed directly by selecting **SSH**, which will open an in-browser terminal:

Compute Engine

VM instances CREATE INSTANCE IMPORT VM REFRESH START STOP RESET DELETE

Filter VM instances ⓘ Columns ▾

Name ^	In use by	Internal IP	External IP	Connect
container-fundamentals-instance-us-central1-a		10.128.0.3 (nic0)	35.235.235.235	SSH ▾ ⋮
container-fundamentals-instance-us-central1-b		10.128.0.4 (nic0)	35.235.235.236	SSH ▾ ⋮
ubuntu		10.128.0.3 (nic0)	35.235.235.235	SSH ▾ ⋮

However, the in-browser terminal may not be as feature rich as our favorite terminal application, therefore we will access our remote VM by running the `ssh` command from our favorite terminal instead. From our desired terminal the `ssh` command will allow us to access the remote VM without the need to type in a password at the login prompt, thanks to the keys we have

generated earlier. The External IP will be visible on your VM instances page and must be used together with the desired user `student` in the `ssh` command below:

```
user@host:~$ ssh student@<External-IP>
```

The sample External IP below is for illustration purposes only, and should not be used for lab exercises. Type `yes` when prompted to confirm connection intent:

```
user@host:~$ ssh student@34.120.187.202
The authenticity of host '34.120.187.202 (34.120.187.202)' can't be
established.
ECDSA key fingerprint is
SHA256:U5+A5MXV92hVR4hZGtVlOWvSwYYpM/fhujpAnd3Y6fQ.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

```
yes
```

```
Warning: Permanently added '34.120.187.202' (ECDSA) to the list of
known hosts.
```

```
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.8.0-1038-gcp x86_64)
```

```
...
```

```
Last login: Tue Aug 10 00:02:33 2021 from ...
```

After a successful login, our prompt from the VM should look like this:

```
student@ubuntu:~$
```

At this point we are ready for lab exercises. Keep in mind that before remotely accessing the GCE `ubuntu` VM, the VM needs to be started from the Console, and needs to remain in a **Running** state throughout the exercise. Once an exercise is completed, the VM can be stopped or shut down, without any loss of work, and then started back up when needed again. Chances are very high that the VM's public IP address will change across VM restarts. Consult the VM management Console to retrieve the latest public IP address of the GCE VM prior to running the `ssh student@<External-IP>` command.

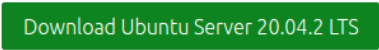
VirtualBox VM

While we encourage learners to explore cloud infrastructure providers to run the lab exercises, access to cloud services may not always be possible in all situations. With that in mind, we are going to present the provisioning of a VM utilizing a popular hypervisor, VirtualBox.

Let's download and install the latest VirtualBox release matching our host operating system. VirtualBox is supported by Linux, OS X, Windows, and Solaris hosts. This installation will assume a native Ubuntu (18.04 LTS or 20.04 LTS) Linux host operating system. Navigate to <https://www.virtualbox.org/wiki/Downloads> and click on the desired host platform to download and install the desired VirtualBox package. Also, download the VirtualBox Extension Pack, which may be known as “guest additions” – a software pack to be used in a later step. Keep in mind that the versions of the VirtualBox package and extension pack should match:

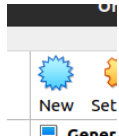


In preparation, we should also download a desired guest operating system image. For the purpose of this course, we are downloading the Ubuntu Server 20.04.2 LTS image file. Navigate to <https://ubuntu.com/download/server>, select **Option 2 – Manual server installation**, then click on the green download button:

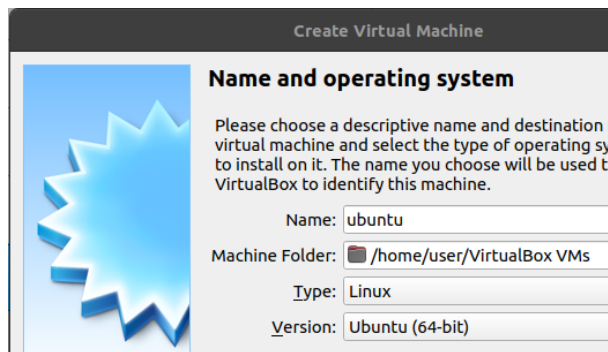
A green rectangular button with the text "Download Ubuntu Server 20.04.2 LTS" in white.

Save the **iso** image file to a desired and easily accessible location on your host system.

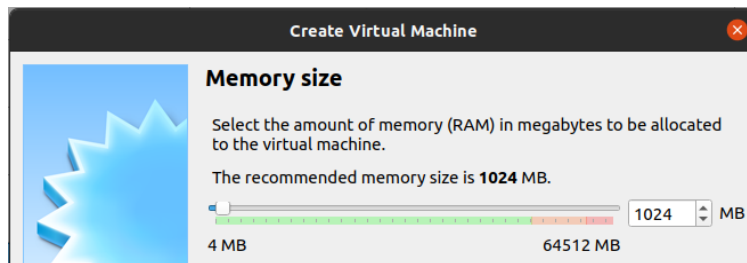
Assuming that we have successfully downloaded and installed VirtualBox, successfully downloaded the VirtualBox guest additions and the Ubuntu iso, let's provision our Virtual Machine. Start VirtualBox, and from the **Oracle VM VirtualBox Manager** interface select **New**, or press simultaneously **CTRL + N**, or expand the **Machine** menu and select **New**:



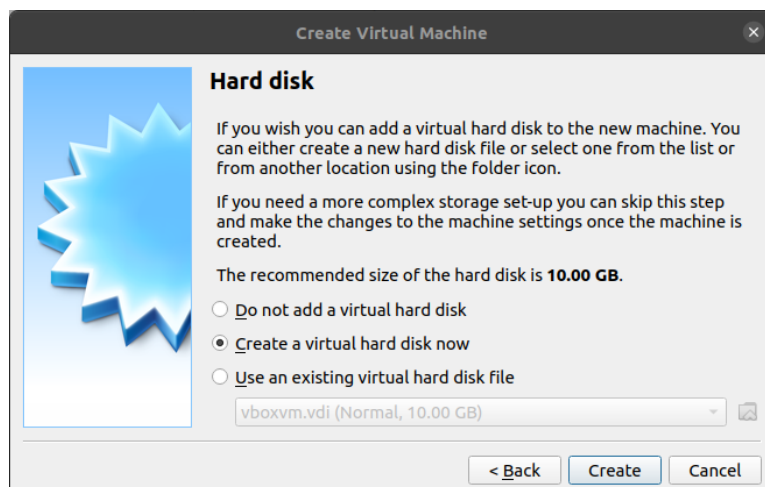
On the **Create Virtual Machine** interface provide the VM Name **ubuntu**, select Type **Linux**, select Version **Ubuntu (64-bit)**, then click **Next**:



On the following step set the Memory size to at least the minimum recommended **1024 MB** and click **Next**:



Then select **Create a virtual hard disk now** and click **Create**:



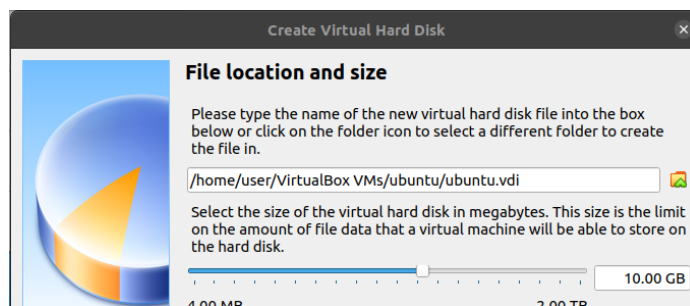
Select the **VDI** virtual hard disk file type, click **Next**:



Select **Dynamically allocated**, click **Next**:



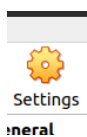
Leave the default **10 GB** size or adjust based your host's available free disk space, then click **Create**:



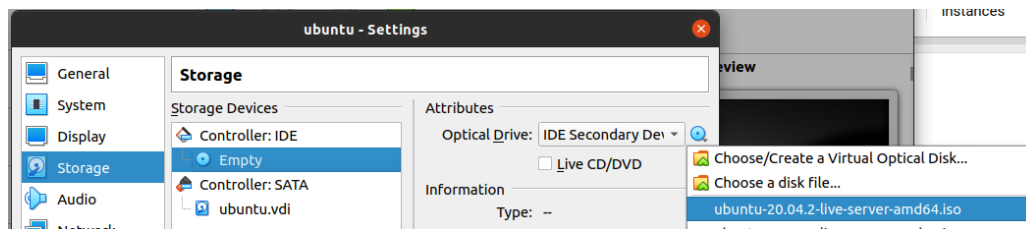
The ubuntu VM has been provisioned, and it is currently powered off.



Before attempting to boot the VM, let's make a few changes in its settings:

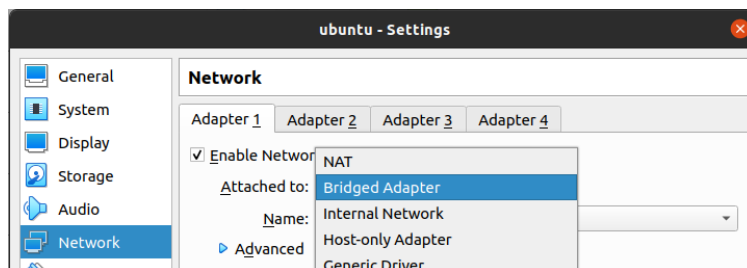


Select the **Storage** category, select the **Empty** IDE storage device, and by clicking on the small disk icon select the desired **iso** image from the drop down, or navigate to the desired **iso** image file with **Choose a disk file...** :

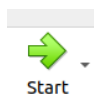


The VM is provisioned by default with a networking device operating in **NAT** mode. Since **NAT** may require additional configuration to support certain aspects of container operations, we will change it to a **Bridged Adapter** mode instead.

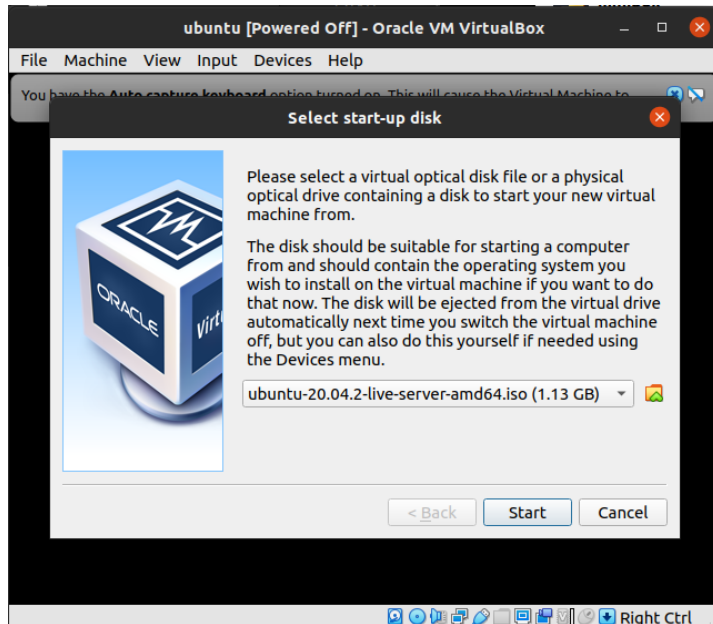
Select the **Network** category, then select Attached to: Bridged Adapter, and the Name: to be the physical device that connects your host to the internet (could be an ethernet physical adapter – **ensx / enpx...**, or a wireless lan physical adapter – **wlpxxx...**):



Now we are ready to start the ubuntu VM:



Verify the desired startup disk is selected, which should be the desired **iso** image file of the **Ubuntu Server 20.04.2 LTS**, then click **Start**:



Using your keyboard arrows proceed through the following configuration screens:

Select your desired language then press **Enter**.

For keyboard configuration press **Enter** for default.

For Network connections press **Enter** for defaults.

For Proxy press **Enter** for defaults.

For Ubuntu archive mirror press **Enter** for defaults.

For Guided storage press **Enter** for defaults.

For Storage configuration select **Continue** and press **Enter** for defaults.

For Profile setup: Your name: **student**, Server's name: **ubuntu**, Username: **student**, Password: any password of your choice that is easy for you to remember and use when logging into the VirtualBox ubuntu VM. Use the arrows or TAB to navigate between fields. Select **Done** and press **Enter**.

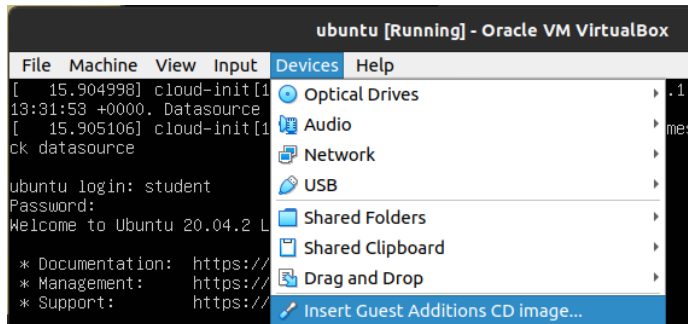
For SSH Setup select **Install OpenSSH server**, then select **Done** and press **Enter**.

Skip the Featured Server Snaps, just select **Done** and press **Enter**.

After the installation is completed, the **Reboot Now** option will become available, select it and press **Enter**. At the Failed unmounting error message just press **Enter**.

After the reboot is completed, press **Enter** to see the **ubuntu login:** prompt. This is where we type the user name **student**, and the password we chose in an earlier step.

There are a few additional steps we need to complete before we can access the local VirtualBox VM from our favorite terminal application. First, we will insert the guest additions image:



Then, at the prompt, we will run the following two commands:

```
student@ubuntu:~$ lsmod | grep -io vboxguest
vboxguest
```

```
student@ubuntu:~$ modinfo vboxguest
```

Let's retrieve the private IP address of the ubuntu VM, by running the following command. The IP address displayed can then be used from our favorite terminal running on the host system:

```
student@ubuntu:~$ hostname -I
192.168.0.199
```

On the host system, run the `ssh` command from a terminal, type `yes` and then the student's password when prompted. Keep in mind that the password will remain hidden as it is typed:

```
user@host:~$ ssh student@192.168.0.199
The authenticity of host '192.168.0.199 (192.168.0.199)' can't be
established.
ECDSA key fingerprint is
SHA256:PxHIiogt1ChpOT34F1OZgMSIay/+gr2oG+NzZI6nC9g.
Are you sure you want to continue connecting (yes/no/[fingerprint])?

yes
Warning: Permanently added '192.168.0.199' (ECDSA) to the list of
known hosts.
student@192.168.0.199's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-81-generic x86_64)
```

...

Last login: Tue Aug 10 13:35:27 2021

One last step is to ensure that the **student** user is able to run sudo on the guest VM without being prompted for a password. Run the following command and type the student's password when prompted, which will remain hidden as it is typed:

```
student@ubuntu:~$ sudo -i
```

```
[sudo] password for student:
```

```
root@ubuntu:~# cd /etc/sudoers.d/
```

```
root@ubuntu:/etc/sudoers.d# ls -la
```

```
total 12
```

```
drwxr-xr-x  2 root root 4096 Feb  1  2021 ./
drwxr-xr-x 95 root root 4096 Aug 10 18:40 ../
-r--r----- 1 root root  958 Feb  3  2020 README
```

```
root@ubuntu:/etc/sudoers.d# touch student
```

```
root@ubuntu:/etc/sudoers.d# echo "student ALL=(ALL:ALL) NOPASSWD:ALL"
> student
```

```
root@ubuntu:/etc/sudoers.d# cat student
```

```
student ALL=(ALL:ALL) NOPASSWD:ALL
```

```
root@ubuntu:/etc/sudoers.d# ls -la
```

```
total 16
```

```
drwxr-xr-x  2 root root 4096 Aug 10 14:16 .
drwxr-xr-x 95 root root 4096 Aug 10 13:31 ..
-r--r----- 1 root root  958 Feb  3  2020 README
-rw-r--r--  1 root root   35 Aug 10 14:16 student
```

```
root@ubuntu:/etc/sudoers.d# chmod 440 student
```

```
root@ubuntu:/etc/sudoers.d# ls -la
```

```
total 16
```

```
drwxr-xr-x  2 root root 4096 Aug 10 14:16 .
drwxr-xr-x 95 root root 4096 Aug 10 13:31 ..
-r--r----- 1 root root  958 Feb  3  2020 README
-r--r----- 1 root root   35 Aug 10 14:16 student
```

At this point we are ready for lab exercises. Keep in mind that before remotely accessing the VirtualBox **ubuntu** VM, the VM needs to be started from the **Oracle VM VirtualBox Manager**, and needs to remain in a **Running** state throughout the exercise. Once an exercise is completed, the VM can be stopped or shut down, without any loss of work, and then started back up when needed again. Chances are very high that the VM's private IP address will be preserved across restarts, and the `ssh student@192.x.x.x` command to remain unchanged. In the case the `ssh` command stops working, revisit earlier steps to log directly into the **ubuntu** VM and retrieve its IP address, and then update the `ssh` command.