

Practical

Virtual Machine Management

Remote in the cloud, local with libvirt

Mathieu Bacou

`mathieu.bacou@telecom-sudparis.eu`

2026 – 2027

Télécom SudParis

Institut Mines-Télécom & Institut Polytechnique de Paris



Part I.

Introduction

1. Overview

Virtual machines (VM) remain at the heart of cloud platforms to provide resources to the users. In this practical, you will discover how easy it became to access cloud resources using VMs. The products of Google Cloud Platform (GCP)¹ were chosen as the application context, but you will find similar concepts in other cloud platforms.

Nonetheless, GCP's interface hides the actual work that is done to run VMs. The lecture showed you that creating a virtual machine (VM) locally (with QEMU/KVM) is a quick operation, but requires studying the manual for a long time to understand what options are needed. Moreover, additional tooling is required for easy network configuration and image management; and VM operations for the cloud are essentially missing: remote VM access, VM lifecycle management, migration, snapshotting...

In this context, libvirt² is a toolkit, hypervisor-agnostic, that offers higher-level management of VMs and access to related features. It also includes bindings to many programming languages to build custom virtualization platforms. This practical focuses on the usage of libvirt as an interactive toolkit for VM administration.

The practical has two parts: in part II, you buy resources from GCP with two VMs, with an overview of its main configuration options; and in part III, you use those two VMs as hypervisor hosts to discover libvirt.

2. Prerequisite

Knowledge of using a terminal is assumed, and an understanding of virtualization notions seen during the lecture is welcome.

The practical makes use of resources from GCP, which is a paid product. For this class, you should have received credits to redeem on GCP using a Google account; on this note, you might prefer creating an account linked to your institutional e-mail address instead of using a personal account.

As for technical requirements, almost all operations take place in VMs from GCP. Thus, the only prerequisite is to install and configure Google Cloud SDK, and more specifically its `gcloud` command. Instructions can be found on Google Cloud Docs: https://cloud.google.com/sdk/docs/install#installation_instructions.

¹*Cloud computing services / Google Cloud*. 2011. URL: <https://cloud.google.com/> (visited on 12/06/2020).

²*libvirt: The virtualization API*. 2005. URL: <https://libvirt.org/> (visited on 12/05/2020).

3. Setup for Google Cloud CLI

Information

It is definitely more comfortable to work on the practical from a local terminal on your computer, so prefer installing the `gcloud` command. However, you can also follow the practical from Google Cloud's console embedded in its website, named Cloud Shell and located in the upper right corner of every Google Cloud page.

Warning

You will be working with paid VMs from the cloud provider Google Cloud Platform. The pricing model is that you pay for your VM resources as long as the VM is up, whatever its activity. Thus: *do not forget to shutdown all your VMs at the end of the practical.*

3. Setup for Google Cloud CLI

You are advised to set default values for the *project* and *zone* when using the Google Cloud CLI. This will allow you to omit them from every command you type. This is shown in listing 1.

Listing 1: Setup GCP CLI.

```
# Replace $GCP_PROJECT_NAME with the project name (from the web interface).
gcloud config set project $GCP_PROJECT_NAME
# europe-west9-b is recommended.
gcloud config set compute/zone europe-west9-b
```

Part II.

Virtual Machines in the cloud with Google Cloud Platform

In this part, you will:

1. create a VM based on a public image;
2. prepare a custom system image;
3. create two VMs for the next part.

First, go to Google Cloud's console at <https://console.cloud.google.com/>.

4. [Basics] VM creation in web interface

Click your way to the menu “Compute Engine”, submenu “VM instances”, as shown in fig. 1: left panel, “Compute Engine”, popup menu, “VM instances”. Then click on the button to “Enable” the API.

Click to “Create an instance” of a new VM, and configure it as shown in figs. 2 and 3:

Name at your convenience;

Region better to pick a region nearby;

Machine configuration there are many choices of series and types:

Family VM's optimization: general usage, computation, memory or GPU,

Series architecture of the underlying physical machine, E2 is a basic Intel processor,

Type of machine machine size and additional resources, choose something small like e2-medium or smaller;

OS and storage (left pane) check the disk size (10 GB is expected) and that the OS is Debian GNU/Linux 12 (bookworm).

Question 1

Take the time to understand all the options that are shown to you. In particular:

- Why are you advised to “pick a region nearby”?
- What are the effects of choosing a small VM configuration?

After one minute, the VM instance is up and running.

4. [Basics] VM creation in web interface

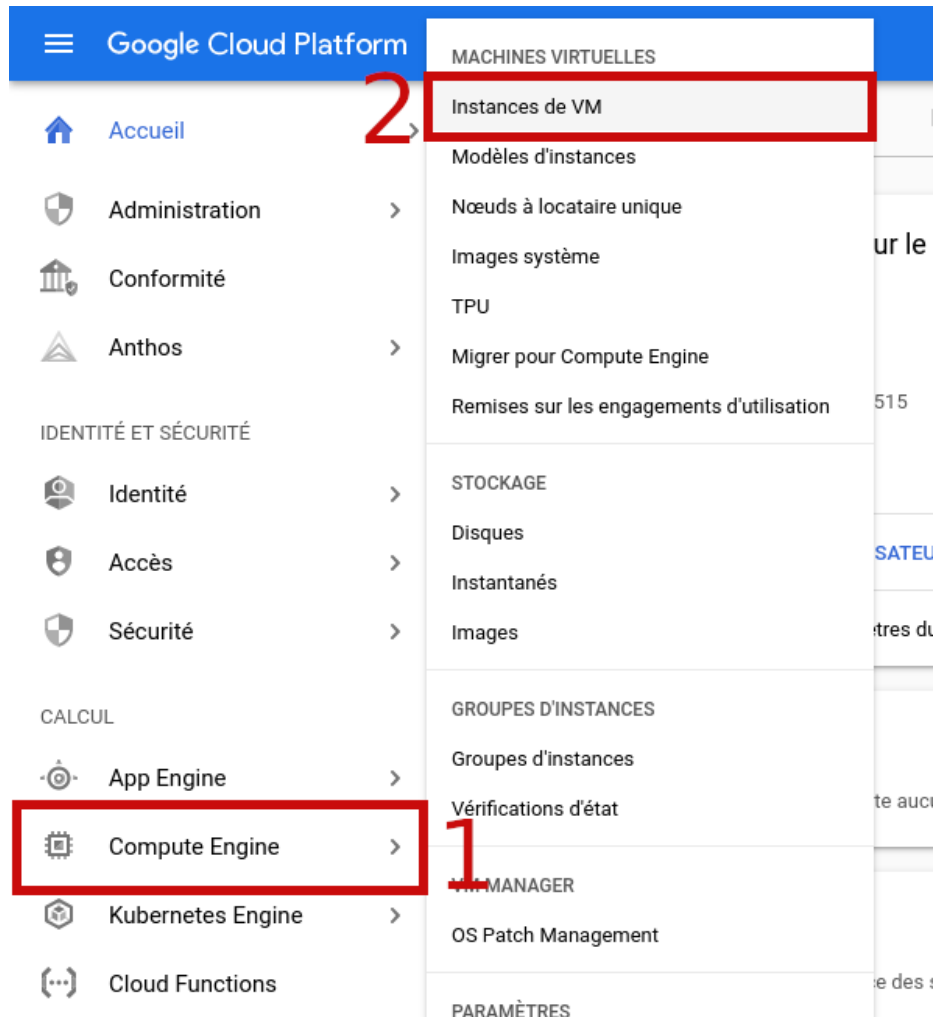


Figure 1: Menu path to VM instances.

4. [Basics] VM creation in web interface

Nom *
my-vm

Libellés ?
[+ AJOUTER DES LIBELLÉS](#)

Région *
europe-west1 (Belgique)

Zone *
europe-west1-b

La sélection d'une région est définitive

La zone est définitive

Configuration de la machine

Famille de machines

USAGE GÉNÉRAL OPTIMISÉE POUR LE CALCUL MÉMOIRE OPTIMISÉE GPU

Types de machines pour les charges de travail courantes permettant d'optimiser les coûts et la flexibilité

Série
E2

Sélection de la plate-forme de processeur en fonction de la disponibilité

Type de machine
e2-medium (2 processeurs virtuels, 4 Go de mémoire)



vCPU
1 à 2 vCPU (1 cœur partagé)

Memory
4 Go

Figure 2: Instance configuration: name, location, processor and memory.

Boot disk ?		Monthly estimate	
		\$29.53	
		That's about \$0.04 hourly	
		Pay for what you use: no upfront costs and per second billing	
Name	test-ubuntu-vm-2	Item	Monthly estimate
Type	New balanced persistent disk	2 vCPU + 4 GB memory	\$28.37
Size	10 GB	10 GB balanced persistent disk	\$1.16
Snapshot schedule ?	No schedule selected	Total	\$29.53
License type ?	Free		
Image	Ubuntu 24.04 LTS		
CHANGE			

Figure 3: Instance configuration: disk, estimated bill.

5. [Medium] VM image customization: enabling nested virtualization

Question 2

What happened during this minute? There are two levels to consider:

1. what did the cloud platform do?
2. what did the hypervisor do?

From your shell (own terminal on your computer or Google Cloud Shell), SSH into it with the command in listing 2. It takes care of SSH key management and configuration to reach GCP's VMs.

Listing 2: SSH into a GCP VM.

```
gcloud compute ssh my-vm
```

Congratulations! You created your first VM in the cloud and jumped into it. Using the commands `lscpu`, `free -h` and `lsblk`, you can check the actual resources that were given to your VM (respectively, CPU, memory and disk).

We are not going to do anything with this instance though, so leave it (press `Ctrl-D` or type `exit`), and then shut it down. You can do this either via:

Web interface – select the VM in the instance list and click the “Stop” button;

CLI – type: `gcloud compute instances stop my-vm`.

5. [Medium] VM image customization: enabling nested virtualization

In this section, we prepare GCP VMs to run `libvirt`, and then run our own VMs inside them! This is called *nested virtualization*³.

Unless you can endure the poor performances of full virtualization for your nested VMs, you want to enable hardware assistance for virtualization *inside* the GCP VM, so that your nested VMs can themselves be virtualized with hardware assistance. This serves as an example of VM image customization.

On GCP, this setup requires to explicitly enable nested virtualization when creating a VM instance, and to select a type of VM that supports nested virtualization (one with Intel CPUs, for VMX support).

Information

On your own hardware, a hardware-assisted VM will also be able to run a hardware-assisted nested VM, as long as the feature is enabled in the emulated CPU.

³The term is also applicable to the method of deploying containers inside VMs.

5. [Medium] VM image customization: enabling nested virtualization

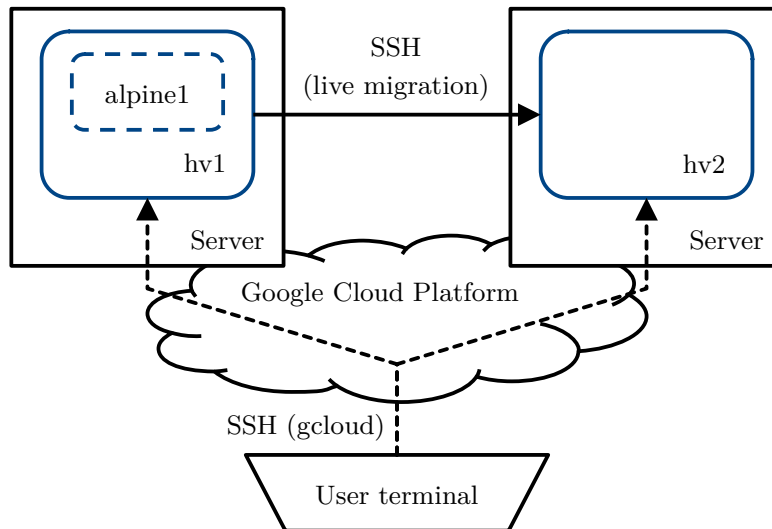


Figure 4: Illustration of the setup for nested virtualization on GCP.

In this practical, we need two VMX-enabled VMs (that we call “hypervisor VMs”), so we will prepare one, then create a system image from it in order to clone it. Finally, we will set the first one to be able to SSH to the other, which is required for VM live migration with libvirt (section 8.1). This is illustrated in fig. 4.

Question 3

Without hardware-assisted virtualization, what is the most performant virtualization mode you could use for your nested VMs?

5.1. VMX-enabled VM instance

Create a VM instance that has nested virtualization enabled (listing 3), called `hv1`. You need to spawn it on a compatible machine type, here `n2-standard-2` (i.e., an Intel processor).

Information

This step cannot be done through the web interface, because the setting is not available to it.

Listing 3: Spawn hypervisor VM for installation.

```
gcloud compute instances create hv1 \
  --enable-nested-virtualization \
  --zone=europe-west9-b --machine-type=n2-standard-2
```

5. [Medium] VM image customization: enabling nested virtualization

When the creation is complete, SSH into `hv1` with `gcloud compute ssh hv1`. Check that VMX is enabled inside `hv1` by running command in listing 4: you should see the output `Virtualization: VT-x`.

Listing 4: Check VMX in hypervisor VM.

```
# should output "Virtualization: VT-x"
lscpu | grep Virtualization
```

5.2. Installation of hypervisor VM

Install `libvirt` and co. with the commands in listing 5. `adduser` will let you use `libvirt` as the normal user.

Listing 5: Install and configure `libvirt` and others.

```
sudo apt update
sudo apt install -y qemu-kvm libvirt-daemon-system virt-manager libguestfs-tools

# Add your user to the groups libvirt and kvm to use virsh without sudo.
sudo usermod -aG libvirt,kvm $USER
# Also give libvirt's user rights to search your home folder to access the VM disk image.
chmod o+rx $HOME
```

Here are the installed dependencies:

qemu-kvm hypervisor;
libvirt-daemon core `libvirt` software (management daemon);
virt-manager interactive interface to `libvirt`;
libguestfs-tools more tools to use `libvirt`.

Then, shutdown the VM with `sudo shutdown now` to be able to create a clean system image from it.

Question 4

Why do you have to shutdown the VM before creating a system image to clone it? What is going to be cloned exactly (excluding GCP specifics)?

5.3. Creation of system image

A system image in GCP is a clone of a full VM instance, i.e. a VM image (disk) with an instance model.

You can create the system image `hypervisor-kvm` either via:

Web interface – see fig. 5;

CLI – see listing 6.

Listing 6: Create system image from VM instance.

```
gcloud compute machine-images create hypervisor-kvm --source-instance hv1
```

5. [Medium] VM image customization: enabling nested virtualization

INSTANCES

PLANIFICATIONS D'INSTANCES

Les instances de VM sont des machines virtuelles hautement configurables permettant d'exécuter des charges de travail sur l'infrastructure Google. [En savoir plus](#)

Filtre

Saisissez le nom ou la valeur de la propriété

☐	État	Nom ↑	Zone	Recommandations	Utilisé par	Adresse IP interne	Adresse IP externe	Connecter
☐	●	hv1	europa-west1-b			10.132.0.3 (nic0)		SSH ▾
☐	●	hv1-console	europa-west1-b			10.132.0.4 (nic0)		SSH ▾
☐	●	hv2	europa-west1-b			10.132.0.5 (nic0)		SSH ▾
☐	●	my-vm	europa-west1-b			10.132.0.2 (nic0)		SSH ▾

Actions associées

🔍 Découvrir la sauvegarde et la reprise après sinistre NOUVEAU

Sauvegardez vos VM et configurez la reprise après sinistre

📄 Afficher le rapport de facturation

Affichez et gérez votre facturation Compute Engine

🛡️ Surveiller les VM

Identifiez les anomalies des VM à l'aide de métriques liées au CPU et au réseau

🔗 Explorer

Affichez, recherchez et téléchargez des journaux

⋮

Démarrer/Reprendre

Arrêter

Suspendre

Réinitialiser

Supprimer

Afficher les détails du réseau

Créer une image système

Afficher les journaux

Afficher les graphiques de surveillance

Figure 5: Hypervisor system image configuration.

5. [Medium] VM image customization: enabling nested virtualization

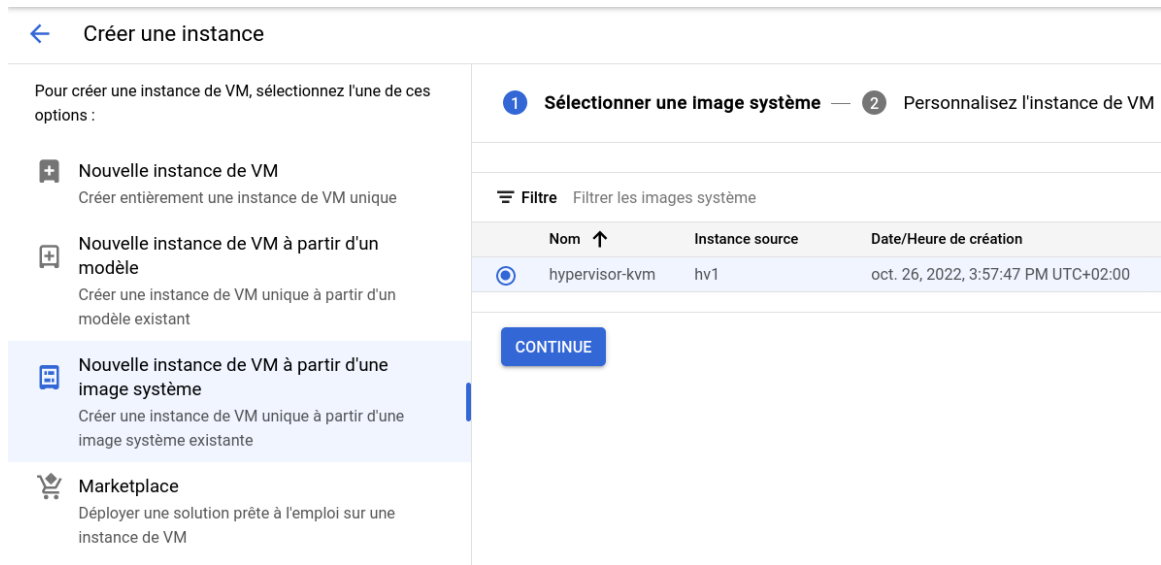


Figure 6: Hypervisor instance creation from system image.

5.4. Setup of two hypervisor VM instances

Create and start a new VM instance `hv2` from this system image, either via:

Web interface – see fig. 6;

CLI – see listing 7.

Listing 7: Create VM instance from system image.

```
gcloud compute instances create hv2 --source-machine-image hypervisor-kvm
```

When `hv2` is up, start again the first hypervisor VM `hv1` with the web interface or with `gcloud compute instances start hv1`, and then SSH into it with `gcloud compute ssh hv1`. We need to configure SSH *inside* `hv1` so that you can SSH from `hv1` to `hv2` during live migration (see section 8). So, SSH into `hv1`, and then follow the commands in listing 8 to do so.

Listing 8: Configure SSH in `hv1`.

```
# setup gcloud in hv1 (accept and follow the instructions)
gcloud auth login
# SSH into hv2 from hv1 to configure publickey authentication from hv1 to hv2
# (follow the instructions, you can keep the default choices)
gcloud compute ssh hv2
# at this point, you are on hv2, just exit to go back to hv1
```

Once you are back to `hv1`, configure its SSH client with the correct identity to reach `hv2`. To achieve this, use the command in listing 9 to write the SSH configuration.

5. [Medium] VM image customization: enabling nested virtualization

Listing 9: Write SSH config to reach hv2 with publickey authentication.

```
# in hv1
echo -e "Host hv2\n\tIdentityFile ~/.ssh/google_compute_engine" >> .ssh/config
```

Check that you can SSH to hv2 manually (i.e. without `gcloud`) with `ssh hv2`, and exit to go back to hv1.

Question 5

Why is it necessary to setup a direct connection between hv1 and hv2 for live migration of VMs?

Well done! You have set up two hypervisor VM instances with SSH access from the first to the second, that will be used for live VM migration at the end of this practical. The setup was illustrated in fig. 4.

This is the end of part II, giving you an overview of VM administration in a cloud platform. In the next part, we focus on manual, local VM management using the hypervisor VM instances you just set up.

Part III.

Manual, local VM management with libvirt

In this part, you will:

1. create a VM on your own hypervisor by importing an existing, pre-installed VM image;
2. create a snapshot of a VM and rollback a VM image;
3. live migrate a VM from one hypervisor to the other.

6. [Basics] Virtual machine creation

In the context of this practical, it is inconvenient to guide you into installing a VM from scratch because it takes too much time, and because doing the installation in the GCP VMs requires extraneous network configuration that brings nothing of interest. Thus, you will create a VM by importing an existing, pre-installed image.

Warning

All this section takes places in your GCP hypervisor instance `hv1`.

6.1. Introduction to libvirt

In this lab, you will use *libvirt* as an interface to the actual hypervisor, QEMU/KVM. libvirt abstracts the operations of managing virtual machines: image management (with the concepts of storage pools and volumes), network administration, life-cycle of *domains* (i.e., virtual machines), access... This abstraction makes it easier to use higher-level features such as migration, replication, scaling, snapshotting, etc.

You will type commands using `virsh`, a CLI client to libvirt's daemons. Typing `virsh` in your shell gives you an interactive prompt to libvirt, but you can also type `virsh <virsh cmd>` directly; for example, `virsh start my-domain` will start the domain `my-domain`. You can explore commands with `virsh help`.

Information

You can use Tab-completion on `virsh` commands from Bash.

6.2. Importing an existing VM image

First, download the pre-installed VM image with `wget` as shown in listing 10. This is an installation of Alpine Linux⁴, a very lightweight Linux distribution.

Listing 10: Download pre-installed VM image.

```
wget https://www-public.imtbs-tsp.eu/~mbacou/cours/practicals/\
vm-management/alpine-3.12.1-installed.qcow2.xz -O - | tar -xJ
```

To import the image, first create a libvirt storage pool – a collection of volumes – and a libvirt volume – a disk image. To finish, “upload” the image to the volume, with the commands from listing 11.

Listing 11: Create a volume and upload a VM image to it.

```
mkdir --parents "$HOME/libvirt/images"
virsh pool-create-as --name default --type dir --target "$HOME/libvirt/images"
virsh vol-create-as --pool default --name alpine1 --capacity 1G --format qcow2
virsh vol-upload --pool default --vol alpine1 --file alpine-3.12.1-installed.qcow2
```

Here is an explanation of the commands:

- `pool-create-as` creates a new storage pool of type `directory` at the given path
- `vol-create-as` creates a new volume from the given configuration:
 - `--pool default` is the name of the volume storage pool,
 - `--name alpine1` is the name of the new volume,
 - `--capacity 1G` is the size of the volume,
 - `--format qcow2` defines the file format of the volume (see below for `qcow2`);
- `vol-upload` copies the content of a file (e.g. an existing, non libvirt image) to a volume.

Information

The image is in `qcow2` format: this is a file format optimized to store VM images because it can grow on-demand (such that an apparent 10 GB image can be stored as a much smaller file), and it handles internal snapshotting via copy-on-write (thus “cow” in the name); see section 7 for snapshotting with `qcow2`.

Question 6

Given the information above, why is `qcow2` so useful in a virtualized cloud context?

⁴*Alpine Linux*. 2005. URL: <https://alpinelinux.org/> (visited on 12/06/2020).

Information

In general, `virsh` subcommands that end with `-as` create new libvirt objects from their description given on the command line. On the opposite side, commands without `-as` can read the object definition from an XML file.^a

^a*libvirt: Domain XML format.* URL: <https://libvirt.org/format.html> (visited on 12/06/2020).

6.3. Creation of a new VM**Warning**

Before creating the new VM, which will start it and put you in its console, remember this: *you can leave the VM running and go back to your shell in `hv1` by pressing `Ctrl-5`; on a French AZERTY keyboard, this actually requires pressing `Ctrl-Shift-5`.*^a

^aThe actual keycode to send is `^]`, a standard code to escape a terminal.

The command to create the new domain – to create the new VM – `alpine1` from the imported image, which also starts it and puts you in its console, is given in listing 12. It calls `virt-install`, which is a `virt-manager` command to “install” a new libvirt domain. It has capabilities to automatically fetch installation media of mainstream Linux distributions and configure a domain from it, but here we use it with the `--import` flag to create the new domain from an existing volume. The installation takes some time, *so read the command details below*.

Listing 12: Create and start new VM from imported image.

```
virt-install --import --name alpine1 \
  --os-variant alpinelinux3.12 \
  --memory 2048 --vcpus=2 --cpu host \
  --graphics none \
  --disk vol=default/alpine1,format=qcow2,bus=virtio
```

Here is the detail of the command:

- `--import` skips the installation process and creates a domain from a pre-installed volume;
- `--name` sets the name of the domain;
- `--os-variant` gives information to libvirt to set virtualization options optimized for the guest OS;
- `--memory` and `--vcpus` set the resources of the domain;
- `--cpu host` exposes the exact CPU model of the hypervisor host;
- `--graphics none` disables graphics for the domain (i.e. only a serial console is available);
- `--disk` sets the disk of the domain:

6. [Basics] Virtual machine creation

- `path=alpine-3.12.1-installed.qcow2` gives the path to the disk image,
- `format=qcow2` makes explicit the format of the disk image,
- `bus=virtio` creates the disk with paravirtualized drivers `virtio`.

In addition, libvirt will set the network up for the domain using a bridge (for local networking) and NAT routing rules (for access to the Internet).

Information

In the case of your GCP hypervisor VM, your nested VM will not have access to the Internet for security reasons from the cloud platform point of view.

Now that you have a console to your nested VM, login as `root` (no password required). Congratulations! You created your first VM using libvirt, nested in a GCP hypervisor VM.

Shut it down by typing `poweroff` to end the “installation” process.

Now that the domain is created, you can start it again with the command in listing 13. The flag `--console` immediately puts you in the nested VM’s console.

Listing 13: Start nested VM.

```
virsh start --console alpine1
```

Login as `root` again (it takes a few seconds to get to the console in `alpine1`). Then, create a marker file, for instance with command in listing 14.

Listing 14: Create a marker file.

```
echo v1 > version
```

This marker file will be used in the following steps to demonstrate rollback and live migration features. Finally, detach from the nested VM with `Ctrl-5` (or `Ctrl-Shift-5`, or use Google Cloud Shell’s virtual keyboard); the nested VM `alpine1` is still running, as can be seen with `virsh list`.

6.4. Libvirt domain configuration

All objects in libvirt (domains, volumes...) are defined by XML files.

For instance, you can obtain the definition of your domain `alpine1` with the command in listing 15.

Listing 15: Get libvirt domain definition file.

```
virsh dumpxml alpine1 > alpine1.xml
```

Question 7

Display it using `vim` (or anything else). Libvirt generated much more configuration than what you specified on the `virt-install` command. Everything is automatically fine-tuned for this particular guestOS. Imagine the length of the

equivalent `qemu-system-x86_64` invocation!

7. [Medium] Virtual machine snapshot and rollback

A first useful feature of libvirt when using VMs in a cloud computing environment is the capability to take snapshots of domains, and then roll them back to a previous state.

7.1. Snapshot

Type the command in listing 16 to take a snapshot of your nested VM while it is running.

Listing 16: Take a snapshot of a running domain.

```
virsh snapshot-create-as alpine1 --name alpine1_v1
```

You can list the snapshots with `virsh snapshot-list alpine1`.

Get information about this last snapshot with the command in listing 17. In this command, `--current` is a shortcut to the last snapshot taken of the given domain.

Listing 17: Get information about the last snapshot.

```
virsh snapshot-info alpine1 --current
```

In the output, note that the location of the snapshot is **internal**. This is because the image format of the domain `alpine1` is `qcow2`, which is capable of storing snapshots internally.

Information

Despite the snapshot being stored internally in the `qcow2` disk image, it is a complete snapshot that includes the VM's memory.

7.2. Rollback

To demonstrate the rollback mechanism, first jump back into the nested VM `alpine1` with the command in listing 18.

Listing 18: Get a console to a guest VM.

```
virsh console alpine1
```

Your terminal is blocked waiting on the command prompt of the nested VM: press **Enter** to get a new prompt.

Update your marker file, for example to `v2`. Then, confirm the “update” with `cat version`. Finally, leave `alpine1` running with `Ctrl-5` (or `Ctrl-Shift-5`).

To rollback to version 1 of `alpine1`, type the command in listing 19.

8. [Medium] Virtual machine live migration

Listing 19: Rollback a domain.

```
virsh snapshot-revert alpine1 alpine1_v1
```

Question 8

How can you confirm the rollback? What state do you expect to find in **alpine1**?
Confirm that you rolled the VM back.

To set up the next section (VM live migration), leave **alpine1** running, and delete the snapshot by running the command in listing 20.

Information

We delete the snapshot before trying live migration because snapshots are objects that must be accessible on the new host after the migration. In a real cloud platform, snapshots are managed at the platform level on shared storage, and thus do not need to be specially managed: disk images and snapshots are always available platform-wide.

Listing 20: Delete snapshot.

```
virsh snapshot-delete alpine1 alpine1_v1
```

8. [Medium] Virtual machine live migration

Recall that we have two hypervisor instances running on GCP: **hv1** and **hv2**. The goal of this section is to migrate live – without service suspension – the nested VM **alpine1** from **hv1** to **hv2**, as shown in fig. 4.

8.1. Migration

To demonstrate the live migration, jump back into **alpine1**, run the command in listing 21, and detach from **alpine1**'s console with the command running, by pressing **Ctrl-5** (or **Ctrl-Shift-5**). It writes the date and time every two seconds to a file.

Listing 21: Heartbeat command.

```
while true; do
    echo $(date) >> heartbeats
    sleep 2
done
```

As long as communication between both hypervisor hosts is set up (as you already did in section 5.4), libvirt takes care of almost everything about live migration.

First, prepare the migration by creating **alpine1**'s future volume on **hv2**. This is the same command as in listing 11, and it is recalled in listing 22.

8. [Medium] Virtual machine live migration

Listing 22: Create destination volume on hv2 for migration.

```
# on hv2
mkdir --parents "$HOME/libvirt/images"
virsh pool-create-as --name default --type dir --target "$HOME/libvirt/images"
virsh vol-create-as --pool default --name alpine1 --capacity 1G --format qcow2
```

Now back on hv1, type the command in listing 23 to launch the migration. You should keep note of the time when you launch the migration, that will be used afterwards to confirm the liveness of the VM. The migration will take some time, so read the explanation of the command below!

Listing 23: Live migrate a VM.

```
virsh migrate --verbose \
  --live --persistent --undefinesource --copy-storage-all \
  --domain alpine1 --desturi "qemu+ssh://hv2/session"
```

Here is an explanation of the command line:

- **--live** instructs libvirt to perform a live migration, i.e. the guest is not suspended during migration;
- **--persistent** tells libvirt to make the migrated domain persistent on the destination hypervisor;
- **--undefinesource** tells libvirt to undefine – remove – the migrated domain on the source hypervisor;
- **--copy-storage-all** instructs libvirt to copy the domain storage – the disk image of the VM – to the destination hypervisor
- **--domain alpine1** is the name of the migrated domain;
- **--desturi "qemu+ssh://hv2/session"** is a libvirt URI pointing to the destination hypervisor (see below).

Question 9

Why do you need to explicitly tell libvirt to copy the disk image of the VM (think of what data is moved during live migration according to the lecture)? How is the disk image handled during live migration by a cloud platform?

The destination is given in **--desturi** as **hv2**, which is the hostname of the other hypervisor VM (GCP handles hostname resolution of the VMs). The **qemu+ssh** part means that the destination hypervisor is QEMU, and that the data transfer shall be done via SSH. The **/session** part is irrelevant but mandatory.

Because we had to copy the VM disk image, the migration was a bit long, however it was live, so the nested VM was not affected by this.

Question 10

What does it mean that the migration was live (and seamless), for:

8. [Medium] Virtual machine live migration

- any user of the VM?
- the migrated VM?
- the hypervisor?

Information

We had to create the *volume* on hv2 beforehand: libvirt was instructed to copy the *image data* with `--copy-storage-all`, but needed a matching volume definition on the destination hypervisor.

8.2. Confirmation of the live migration

Question 11

How can you confirm the migration? What do you expect to find on hv1 and hv2? With the preparation you did before the migration, what state do you expect to find in `alpine1` to confirm the liveness of the migration?

Once the migration is over, confirm it:

- on hv1: `virsh list --all` shows that `alpine1` is gone;
- on hv2: `virsh list` shows that `alpine1` is here, up and running!

On hv2, jump into `alpine1` with `virsh console alpine1`. Remember that you left it in an infinite `while` loop, so interrupt it with `Ctrl-C`. Display the heartbeats file: no date was missed, which confirms the liveness of the migration.

Congratulations! You performed your first live VM migration.

Part IV.

Conclusion

Clean up your VMs: properly shut down `alpine1` with `poweroff`, then stop your GCP hypervisor VM `hv1` and `hv2`.

Warning

Confirm that they are stopped, otherwise they will burn through your credits!

In this practical, you discovered Google Cloud Platform to run your virtual machines in the cloud. Using cloud resources, you trained using `libvirt` to manually manage your local virtual machines, with snapshots and live migration.

To go further, you can explore GCP autoscaling functionalities by creating groups of instances, and configure your VMs for public access to provide a web service. About `libvirt`, you can install it on your own machine and try virtual machines with graphical interfaces, and explore all the fine-tuning capabilities of its domain configuration⁵ or discover the many network configurations⁶.

⁵*libvirt: Domain XML format*. URL: <https://libvirt.org/formatdomain.html> (visited on 12/06/2020).

⁶*Networking - Libvirt Wiki*. June 8, 2020. URL: <https://wiki.libvirt.org/page/Networking> (visited on 12/06/2020).