

Cloud Computing

Rapport du Projet

Encadré par **Pr. Chakir El Amrani**

Réalisé par **EL GHAZI Loubna**

Table de Matières

1- Introduction.....	3
2- CloudSim.....	4
-Objectif	4
-Contexte.....	4
-Spécificités de l'Université Abdelmalek Essaadi	4
-Scénarios d'Architecture Cloud	5
-Choix du Meilleur Scénario	6
-Implémentation avec CloudSim	6
-Conclusion.....	7
3- OpenStack.....	8
-Objectifs.....	8
-Méthodologie d'installation d'OpenStack	8
-Test du Middleware.....	9
-Implémentation de l'IaaS	10
4- Docker	15
-Installation et Configuration de Docker sous Linux	15
-Différences entre Docker et une Machine Virtuelle.....	16
5- Conclusion	17

1- Introduction

Dans un monde où les infrastructures informatiques sont de plus en plus sollicitées par les entreprises et les établissements académiques, le Cloud Computing se présente comme une solution de choix pour répondre aux besoins de flexibilité, de scalabilité et de gestion efficace des ressources. Ce projet vise à explorer et à implémenter plusieurs technologies liées au Cloud Computing, à travers trois grandes parties : CloudSim, OpenStack, et Docker.

La première partie du projet s'intéresse à CloudSim, un simulateur de Cloud Computing, dans lequel nous proposerons une architecture hybride pour l'Université Abdelmalek Essaadi (UAE). Cette architecture hybride combinera à la fois un cloud privé et un cloud public pour répondre aux besoins variés de l'université, en tenant compte des spécificités telles que la sécurité des données sensibles et la scalabilité pour les services éducatifs. Nous explorerons plusieurs scénarios possibles et sélectionnerons le meilleur en fonction des critères identifiés.

La deuxième partie du projet se concentre sur OpenStack, une plateforme open-source permettant de déployer des infrastructures de cloud privé. Nous procéderons à l'installation d'OpenStack sur un système Linux, testerons ses fonctionnalités de middleware et implémenterons un IaaS (Infrastructure as a Service) ainsi qu'un SaaS (Software as a Service). Ce volet vise à mettre en pratique les concepts abordés en cours et à se familiariser avec la gestion d'un cloud privé.

Enfin, la troisième partie du projet portera sur Docker, une technologie de conteneurisation qui permet de créer, déployer et gérer des applications de manière légère et efficace. Nous installerons Docker sur un système Linux, apprendrons à créer et à gérer des conteneurs, et comparerons Docker aux machines virtuelles traditionnelles, en mettant en lumière leurs différences en termes de performances et de gestion des ressources.

Ce projet a pour objectif de fournir une vue d'ensemble sur les technologies modernes de Cloud Computing et de conteneurisation, tout en offrant des applications pratiques pour un usage académique et professionnel.

2- CloudSim

-Objectif

L'objectif de cet exercice est de concevoir une architecture cloud adaptée aux besoins de l'Université Abdelmalek Essaadi (UAE). En tenant compte des spécificités de l'université et de ses exigences en matière d'infrastructure informatique, plusieurs scénarios d'architecture Cloud seront évalués pour déterminer le meilleur choix en fonction des besoins, des coûts, et des performances.

-Contexte

L'Université Abdelmalek Essaadi, comme toute institution académique moderne, doit gérer une infrastructure informatique évolutive et flexible pour :

- Héberger des applications pédagogiques (e-learning, systèmes de gestion de cours).
- Fournir des services aux étudiants, enseignants, et personnels administratifs.
- Assurer la gestion de bases de données, le stockage de documents, et la gestion des ressources informatiques.
- Supporter des plateformes de recherche et des simulations de calcul intensif.

-Spécificités de l'Université Abdelmalek Essaadi

Avant de proposer les scénarios, il est important de prendre en compte plusieurs éléments relatifs à l'Université:

Nombre d'utilisateurs : L'université compte plusieurs milliers d'étudiants, enseignants, et administrateurs, avec des besoins différents en termes d'accès et d'utilisation des services.

Types de services :

- Services de gestion académique : gestion des cours, des emplois du temps, des notes, etc.
- Plateformes de recherche et simulations pour les étudiants et enseignants en sciences et technologies.
- Plateformes d'enseignement à distance (e-learning).
- Services administratifs (gestion des inscriptions, gestion des ressources humaines et financières, etc.).
- Exigences de sécurité : La confidentialité des données des étudiants et du personnel administratif doit être assurée.

Budget limité : L'université doit optimiser les coûts en évitant des dépenses excessives tout en garantissant des performances adéquates.

-Scénarios d'Architecture Cloud

Voici trois scénarios pour l'architecture cloud de l'UAE.

Scénario 1 : Cloud Public

Dans ce scénario, l'UAE utiliserait des services cloud publics, tels que **Amazon Web Services (AWS)**, **Google Cloud Platform (GCP)** ou **Microsoft Azure**. Ce modèle repose sur une infrastructure partagée avec d'autres organisations, offrant une grande flexibilité et évolutivité.

Avantages :

Évolutivité : possibilité d'adapter rapidement les ressources en fonction des besoins (scalabilité).

Coût réduit : paiement à la consommation, pas d'investissement initial lourd.

Accès à des services avancés : Big Data, IA, Machine Learning.

Inconvénients :

Problèmes de confidentialité : dépendance à un fournisseur externe pour la gestion des données sensibles.

Latence : risques de latence pour certaines applications sensibles.

Scénario 2 : Cloud Privé Hybride

Un cloud privé hybride combinerait un cloud privé pour des applications critiques (gestion des données académiques, administration) et un cloud public pour des services évolutifs comme les plateformes de recherche et e-learning.

Avantages :

Sécurité renforcée : gestion des données sensibles dans un cloud privé.

Flexibilité : utilisation des ressources cloud publiques selon la demande, pour les services moins critiques.

Meilleur contrôle : sur l'infrastructure et les données.

Inconvénients :

Complexité : nécessite une gestion plus complexe des ressources et des configurations.

Coût initial plus élevé : pour la mise en place d'une infrastructure privée.

Scénario 3 : Cloud Privé

Dans ce scénario, l'UAE opérerait pour une solution cloud privée, où toute l'infrastructure serait dédiée et gérée en interne ou avec un fournisseur spécialisé.

Avantages :

Contrôle total : sur les données et les applications.

Sécurité maximale : toutes les données sensibles restent dans un environnement privé.

Personnalisation : possibilité d'adapter l'architecture aux besoins spécifiques de l'université.

Inconvénients :

Coût élevé : pour l'acquisition et la gestion de l'infrastructure.

Évolutivité limitée : moins flexible face à des pics de demande.

-Choix du Meilleur Scénario

Après avoir évalué les avantages et inconvénients des trois scénarios proposés, le scénario hybride (**Cloud Privé Hybride**) semble être le meilleur choix pour l'Université Abdelmalek Essaadi pour les raisons suivantes :

Sécurité : La gestion des données sensibles (étudiants, administration) peut être effectuée sur un cloud privé sécurisé.

Flexibilité et Évolutivité : Les services moins critiques, comme les plateformes de recherche et l'e-learning, peuvent tirer parti du cloud public, offrant ainsi une plus grande flexibilité et réduction des coûts.

Optimisation des coûts : L'utilisation d'un cloud public pour les besoins temporaires ou scalables permet de ne payer que pour les ressources utilisées.

-Implémentation avec CloudSim

Description de la simulation

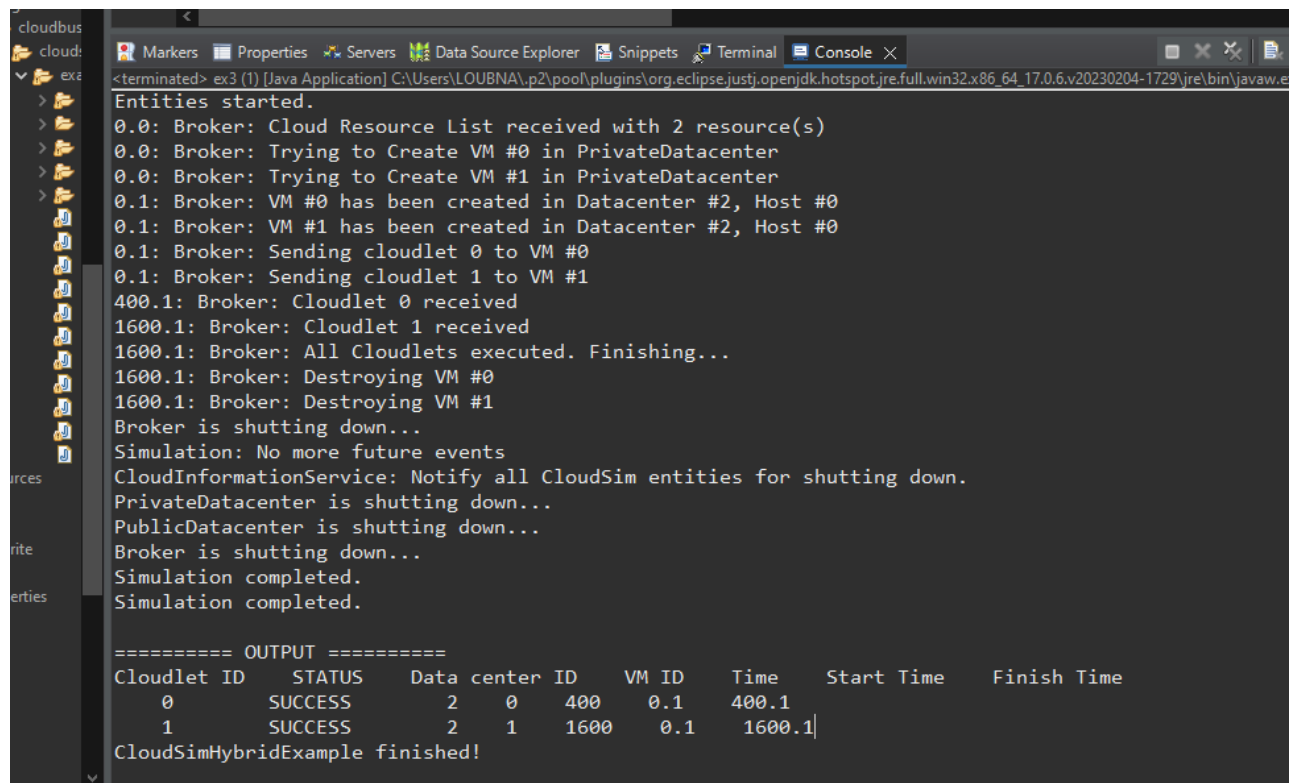
Deux datacenters sont simulés :

PrivateDatacenter : Représente le cloud privé.

PublicDatacenter : Représente le cloud public.

Chaque datacenter héberge des machines virtuelles (VM), qui exécutent des Cloudlets (tâches simulées).

Résultats de la simulation



```
<terminated> ex3 (1) [Java Application] C:\Users\LOUBNA\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.6.v20230204-1729\jre\bin\javaw.exe
Entities started.
0.0: Broker: Cloud Resource List received with 2 resource(s)
0.0: Broker: Trying to Create VM #0 in PrivateDatacenter
0.0: Broker: Trying to Create VM #1 in PrivateDatacenter
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: VM #1 has been created in Datacenter #2, Host #0
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
400.1: Broker: Cloudlet 0 received
1600.1: Broker: Cloudlet 1 received
1600.1: Broker: All Cloudlets executed. Finishing...
1600.1: Broker: Destroying VM #0
1600.1: Broker: Destroying VM #1
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
PrivateDatacenter is shutting down...
PublicDatacenter is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

===== OUTPUT =====
Cloudlet ID   STATUS   Data center ID   VM ID   Time   Start Time   Finish Time
      0      SUCCESS      2      0      400      0.1      400.1
      1      SUCCESS      2      1     1600      0.1     1600.1
CloudSimHybridExample finished!
```

Les Cloudlets ont été exécutés avec succès, répartis entre les deux datacenters selon les ressources disponibles.

Temps d'exécution total des Cloudlets :

Cloudlet 0 (Cloud privé) : 400 ms.

Cloudlet 1 (Cloud public) : 1600 ms.

Interprétation des résultats

Les Cloudlets ont été affectés aux ressources du cloud hybride en fonction des besoins en performance et en confidentialité.

La simulation montre une répartition efficace entre le cloud privé (rapide, sécurisé) et le cloud public (scalable, plus lent).

-Conclusion

Une architecture Cloud Hybride est idéale pour l'Université Abdelmalek Essaadi, car elle permet :

Une gestion optimale des ressources pour répondre aux besoins variés de l'université.

Une amélioration des performances pour les tâches sensibles grâce au cloud privé.

Une flexibilité accrue pour les tâches évolutives grâce au cloud public.

La simulation réalisée dans cet exercice démontre que cette architecture peut être mise en œuvre efficacement.

3- OpenStack

-Objectifs

L'objectif principal de cet exercice est de :

- Installer OpenStack sur un environnement Linux.
- Tester le bon fonctionnement du middleware fourni par OpenStack.
- Implémenter un IaaS basé sur le système Linux fourni avec OpenStack et tester l'accès aux instances via PuTTY.

-Méthodologie d'installation d'OpenStack

Choix de la méthode

Initialement, j'avais prévu d'utiliser DevStack pour l'installation d'OpenStack, mais en raison de divers problèmes de compatibilité, j'ai opté pour MicroStack, une version légère et plus facile à déployer d'OpenStack, conçue pour des environnements de développement ou de test.

Étapes de l'installation

Préparation du système :

-Mise à jour du système d'exploitation avec `apt-get update` et `apt-get upgrade` pour s'assurer que toutes les dépendances sont à jour.

Installation de MicroStack :

-Installation de MicroStack en utilisant la commande suivante :

```
sudo snap install microstack --beta
```

Configuration de MicroStack via la commande :

```
sudo microstack init --auto
```

Une fois l'installation terminée, j'ai vérifié le bon fonctionnement de MicroStack avec la commande suivante :

```
microstack status
```

Cela a permis de vérifier que tous les services nécessaires étaient en cours d'exécution et que l'interface Horizon était accessible via un navigateur web.

-Test du Middleware

-Fonctionnement du Middleware

Une fois OpenStack installé , j'ai testé plusieurs services du middleware pour m'assurer de leur bon fonctionnement. Les services testés incluent :

- **Nova** : pour la gestion des instances virtuelles.
- **Glance** : pour la gestion des images des instances.
- **Keystone** : pour l'authentification et la gestion des utilisateurs.

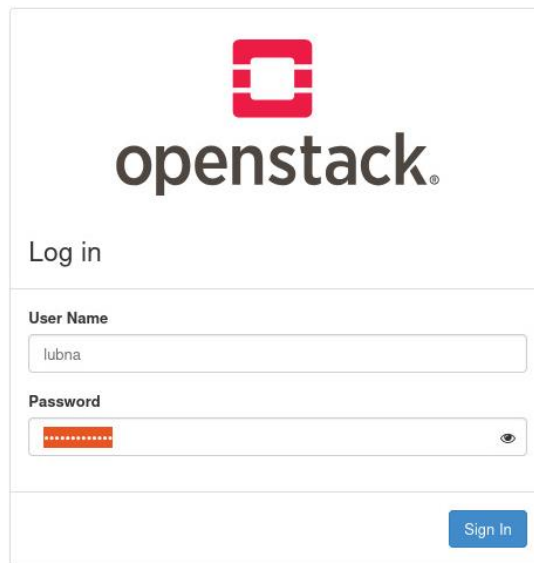
The image shows the OpenStack login interface. At the top is the OpenStack logo, which consists of a red square with a white 'O' inside, followed by the word 'openstack' in a black, lowercase, sans-serif font. Below the logo is the text 'Log in'. Underneath, there are two input fields: 'User Name' with the text 'lubna' entered, and 'Password' with a masked password represented by red dots. To the right of the password field is an eye icon for toggling visibility. At the bottom right of the form is a blue button labeled 'Sign In'.

Figure1 : Utilisateur crée : lubna pour le projet Testproject

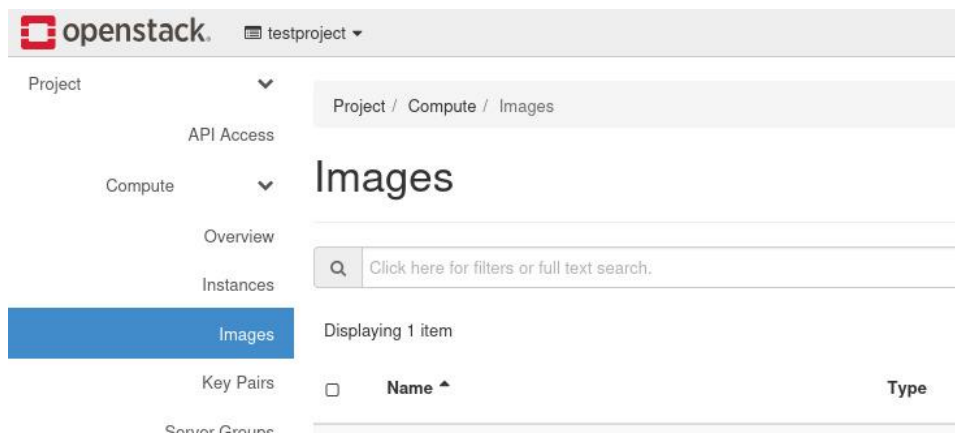


Figure2 : Interface horizon pour le projet crée : testProject

-Résultats des tests

- J'ai vérifié la création et le lancement d'instances avec Nova via l'interface Horizon et la ligne de commande. Toutes les instances se sont lancées sans problème.
- Les images stockées via Glance ont été récupérées et utilisées pour lancer des machines virtuelles avec succès.
- Le service Keystone a permis d'ajouter de nouveaux utilisateurs et de créer des tokens d'authentification pour les services.

-Implémentation de l'IaaS

Création de l'IaaS

Pour implémenter l'IaaS, j'ai suivi plusieurs étapes, incluant la configuration du réseau et la création de l'instance :

1. Création du réseau et du routeur :

- Un réseau a été créé pour permettre la communication entre les instances.
- Un routeur a été configuré pour connecter le réseau privé au réseau public, permettant ainsi à l'instance d'avoir un accès à Internet.

2. Configuration des adresses IP :

- J'ai configuré une adresse IP statique pour le réseau, afin de garantir que les instances créées obtiennent une adresse IP stable.

3. Création et configuration de la clé SSH :

- J'ai généré une clé SSH pour permettre une connexion sécurisée aux instances créées. Cette clé a été associée à l'instance lors de sa création.

4. Création de l'instance :

- Une fois le réseau, le routeur, et la configuration de l'IP et de la clé SSH terminés, une instance virtuelle a été lancée sur OpenStack en utilisant une image Linux (préexistante sur Glance :Cirros).

Images représentatives :

Project

Admin

Overview

Compute

Volume

Network

Networks

Routers

Floating IPs

RBAC Policies

System

Identity

Create Network

NetworkSubnetSubnet Details

☒ Enable DHCP

Specify additional attributes for the subnet.

Allocation Pools192.168.1.2,192.168.22

DNS Name Servers8.8.8.8

Host Routes

CancelBackCreate

Project

API Access

Compute

Volumes

Network

Network Topology

Networks

Routers

Security Groups

Floating IPs

Admin

Identity

Project / Network / Security Groups / Manage Security Group Rules

Manage Security Group Rules: default (92ffc908-ebda-4cd4-aeb7-47f4e5f22b97)

+ Add Rule

Delete Rules

Displaying 6 items

<input type="checkbox"/>	Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Description	Actions
<input type="checkbox"/>	Egress	IPv4	ICMP	Any	0.0.0.0/0	-	-	Delete Rule
<input type="checkbox"/>	Egress	IPv4	TCP	Any	0.0.0.0/0	-	-	Delete Rule
<input type="checkbox"/>	Egress	IPv4	UDP	Any	0.0.0.0/0	-	-	Delete Rule
<input type="checkbox"/>	Ingress	IPv4	ICMP	Any	0.0.0.0/0	-	-	Delete Rule
<input type="checkbox"/>	Ingress	IPv4	TCP	Any	0.0.0.0/0	-	-	Delete Rule
<input type="checkbox"/>	Ingress	IPv4	UDP	Any	0.0.0.0/0	-	-	Delete Rule

Displaying 6 items

Compute >

Volumes >

Network >

Network Topology

Networks

Routers

Security Groups

Floating IPs

Floating IPs

Floating IP Address = Filter [🔗 Allocate IP To Project](#) [🔥 Release Floating IPs](#)

Displaying 1 item

<input type="checkbox"/>	IP Address	Description	Mapped Fixed IP Address	Pool	Status	Actions
<input type="checkbox"/>	10.20.20.29		-	external	Down	Associate <input type="button" value="▼"/>

Displaying 1 item

Project >

Admin >

Overview

Compute >

Volume >

Network >

Networks

Routers

Floating IPs

RBAC Policies

System >

Admin / Network / Networks / mon_reseau

mon_reseau

[Edit Network](#)

[Overview](#) [Subnets](#) [Ports](#) [DHCP Agents](#)

Subnets

Filter [+ Create Subnet](#) [🗑 Delete Subnets](#)

Displaying 1 item

<input type="checkbox"/>	Name	CIDR	IP Version	Gateway IP	Used IPs	Free IPs	Actions
<input type="checkbox"/>	mon_subnet	192.168.1.0/24	IPv4	192.168.1.1	1	20	Edit Subnet <input type="button" value="▼"/>

Displaying 1 item

API Access

Compute >

Overview

Instances

Images

Key Pairs

Server Groups

Volumes >

Network >

Images

[Click here for filters or full text search.](#) [+ Create Image](#) [🗑 Delete Images](#)

Displaying 1 item

<input type="checkbox"/>	Name	Type	Status	Visibility	Protected	Disk Format	Size	Actions
<input type="checkbox"/>	> cirros	Image	Active	Public	No	QCOW2	12.13 MB	Launch <input type="button" value="▼"/>

Displaying 1 item

Instances

Displaying 1 item

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
mon_instance	cirros	192.168.1.5	m1.tiny	ma-key	Active	nova	None	Running	0 minutes	Create Snapshot

Network Topology

Small Normal

external 10.20.20.0/24

mon_route.. Router

mon_reseau 192.168.1.0/24

mon_insta... Instance

Launch Instance Create Network Create Router

Ensuite, j'ai converti le fichier de clé SSH en format **PPK** à l'aide de **PuTTYgen** pour que PuTTY puisse le lire et l'utiliser lors de la connexion à l'instance.

Test avec PuTTY

Une fois l'instance créée, j'ai utilisé PuTTY pour me connecter à distance à la machine virtuelle via SSH. Pour cela :

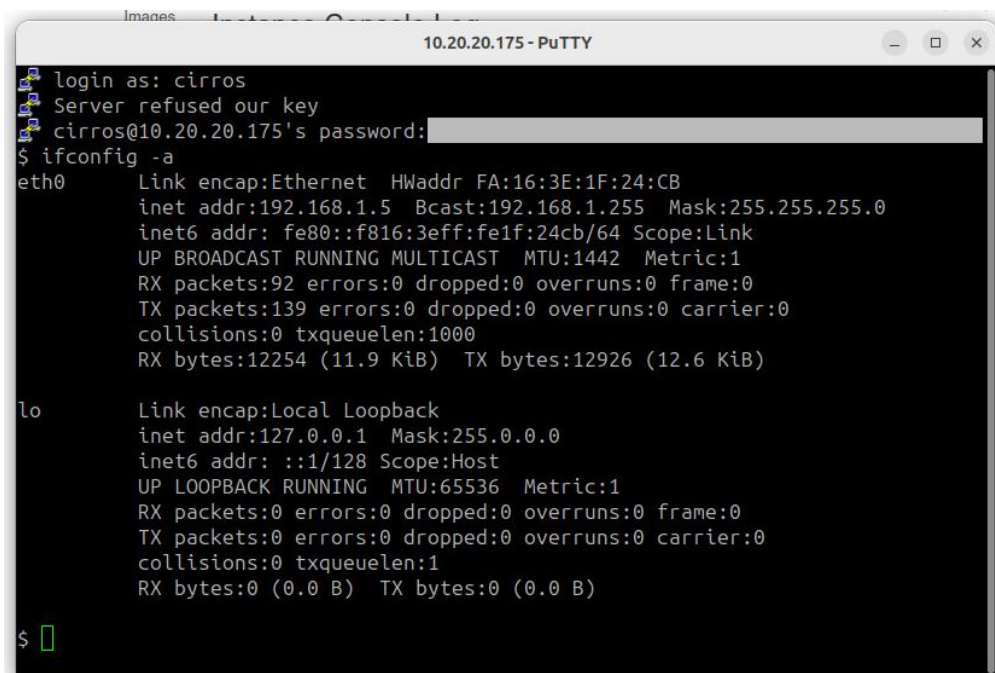
- J'ai récupéré l'adresse IP de l'instance via l'interface Horizon.
- Une fois connecté avec les identifiants SSH et le fichier PPK, j'ai testé diverses commandes Linux pour m'assurer du bon fonctionnement de l'instance.

```
/dev/root resized successfully [took 4.27s]
=== cirros: current=0.4.0 uptime=282.62 ===

  _ _ _ _ _ _ _ _ _ _
 / _ / _ / _ / _ / _ \
/_/_/_/_/_/_/_/_/_/_\
\_/_/_/_/_/_/_/_/_/_\
  http://cirros-cloud.net

login as 'cirros' user, default password: 'gocubsgo', use 'sudo' for root.
cirros login: [ 745.017013] random: nonblocking pool is initialized
```

Recuperation du mot de passe de ciros login pour tester Avec putty



```
10.20.20.175 - PuTTY
login as: cirros
Server refused our key
cirros@10.20.20.175's password: 
$ ifconfig -a
eth0      Link encap:Ethernet  HWaddr FA:16:3E:1F:24:CB
          inet addr:192.168.1.5  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::f816:3eff:fe1f:24cb/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1442  Metric:1
          RX packets:92 errors:0 dropped:0 overruns:0 frame:0
          TX packets:139 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:12254 (11.9 KiB)  TX bytes:12926 (12.6 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

$
```

J'ai exécuté la commande **ifconfig -a**, qui affiche les interfaces réseau de l'instance, confirmant que la connexion à l'instance est bien établie.

4- Docker

Cet exercice présente les étapes d'installation et de configuration de Docker sous Linux, l'utilisation des commandes Docker pour gérer des conteneurs, et les différences clés entre Docker et une machine virtuelle (VM). Docker est une solution incontournable pour la conteneurisation, utilisée largement dans le développement, le déploiement et l'orchestration d'applications modernes.

-Installation et Configuration de Docker sous Linux

Étapes d'installation

1-Mise à jour du système :

```
loubna@loubna-VirtualBox: ~/Desktop
loubna@loubna-VirtualBox:~/Desktop$ sudo apt-get update && sudo apt-get upgrade
-y
[sudo] password for loubna:
Get:1 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Hit:2 http://ma.archive.ubuntu.com/ubuntu noble InRelease
Get:3 http://ma.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
```

2-Installation de Docker :

- Ajout de la clé GPG et dépôt officiel Docker.
- Installation des packages nécessaires :

Cmd :

sudo apt-get install -y docker-ce docker-ce-cli containerd.io

3- Vérification de l'installation :

```
Processing triggers for libc-bin (2.39-0ubuntu0.3) ...
loubna@loubna-VirtualBox:~/Desktop$ docker --version
Docker version 27.3.1, build ce12230
loubna@loubna-VirtualBox:~/Desktop$
```

4-Gestion des Conteneurs avec Docker :

Pour télécharger une image Docker, on utilise la commande :

docker pull <nom_image>

Exemple : Téléchargement de l'image nginx et l'exécuter

```

773c63cd62e4: Pull complete
1d2712910bdf: Pull complete
4b0adc47c460: Pull complete
171eebbdf235: Pull complete
Digest: sha256:bc5eac5eafc581aeda3008b4b1f07ebba230de2f27d47767129a6a905c84f470
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
loubna@loubna-VirtualBox:~/Desktop$ sudo docker run -d -p 8080:80 --name my_nginx nginx
c297fb36c50626a410862bf9ae8fe704985ba0b9749f34ebfcbd3c4904fec479
loubna@loubna-VirtualBox:~/Desktop$

```

5-Lister les conteneurs en cours :

docker ps

```

Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
loubna@loubna-VirtualBox:~/Desktop$ sudo docker run -d -p 8080:80 --name my_nginx nginx
c297fb36c50626a410862bf9ae8fe704985ba0b9749f34ebfcbd3c4904fec479
loubna@loubna-VirtualBox:~/Desktop$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
c297fb36c506   nginx     "/docker-entrypoint...." About a minute ago Up About a minute 0.0.0.0:8080->80/tcp, [::]:8080->80/tcp
loubna@loubna-VirtualBox:~/Desktop$

```

6- Accéder à un conteneur en cours d'exécution :

docker exec -it my_nginx bash

7-On peut aussi arrêter, supprimer des conteneurs en utilisant les commandes suivantes (exps) :

docker stop my_nginx

docker rm my_nginx

-Différences entre Docker et une Machine Virtuelle

Critères	Docker (Conteneurs)	Machine Virtuelle (VM)	
Virtualisation	Virtualisation au niveau des processus.	Virtualisation complète du matériel.	
Taille	Léger (quelques MB).	Plus lourd (quelques GB).	
Démarrage	Rapide (quelques secondes).	Plus lent (plusieurs minutes).	
Isolation	Partage le noyau avec le système hôte.	Complètement isolé (OS indépendant).	
Performances	Plus performant grâce à une faible surcharge.	Moins performant à cause des couches VM.	

	Utilisation	Idéal pour les microservices et le déploiement.	Idéal pour des environnements complexes.
--	--------------------	--	---

Docker simplifie la gestion des applications en fournissant une alternative légère et rapide aux machines virtuelles. Grâce à sa modularité et ses performances élevées, il est idéal pour le développement moderne.

5- Conclusion

Ce projet a été une opportunité d'explorer en profondeur plusieurs aspects essentiels du cloud computing, en mettant en œuvre des outils et des plateformes populaires tels que CloudSim, OpenStack et Docker. À travers ces exercices, nous avons pu appréhender les concepts clés et les techniques associées à la virtualisation, aux infrastructures en tant que service (IaaS), et aux environnements conteneurisés.

En conclusion, ce projet nous a permis de comprendre comment ces technologies cloud se complètent pour offrir des solutions flexibles et évolutives. CloudSim nous a aidé à concevoir et simuler des architectures, OpenStack nous a offert une plateforme pour déployer des infrastructures cloud, et Docker a montré comment les conteneurs peuvent simplifier le déploiement d'applications. Ensemble, ces outils et plateformes constituent un socle technologique puissant pour répondre aux besoins modernes en matière de cloud computing, et ils ont renforcé nos compétences dans ce domaine en constante évolution.