# Projet Fin de Module

CLOUD COMPUTING

Université Abdelmalek Essaâdi

Amine Izougaghen

12 janvier 2026

# Table des matières

# 1 Introduction

Le Cloud Computing s'impose comme pilier de la transformation numérique moderne. Pour l'Université Abdelmalek Essaâdi (UAE), avec ses 128 000 étudiants répartis sur plusieurs campus, le passage au Cloud n'est plus une option mais une nécessité stratégique.

Ce projet présente la conception et le déploiement d'une infrastructure Cloud privée performante et automatisée, structurée selon quatre axes complémentaires :

1. **Simulation** : Validation du dimensionnement via CloudSim
2. **Infrastructure (IaaS)** : Provisionnement avec OpenStack
3. **Automatisation (IaC)** : Orchestration via Terraform et Ansible
4. **Gouvernance** : Monitoring et respect des SLA

Cette approche méthodique garantit une infrastructure robuste, scalable et reproductible, capable de soutenir les ambitions numériques de l'université.

# 2 Modélisation et Simulation

## 2.1 Objectifs de la Simulation

CloudSim permet de valider l'architecture avant tout investissement matériel. Notre simulation modélise le datacenter de Tétouan supportant quatre services critiques : gestion des inscriptions, plateforme e-learning, portail web et outils administratifs.

## 2.2 Architecture Simulée

### 2.2.1 Configuration Matérielle

Le serveur physique simulé dispose de :
— **Processeur** : 10 cœurs à 3000 MIPS
— **Mémoire** : 30 Go RAM
— **Réseau** : 100 Gbit/s

### 2.2.2 Machines Virtuelles

| ID | Service | MIPS | vCPU | RAM |
|---|---|---|---|---|
| VM0 | Inscriptions | 2000 | 2 | 4 Go |
| VM1 | E-Learning | 2000 | 4 | 8 Go |
| VM2 | Portail Web | 1000 | 3 | 8 Go |
| VM3 | Administration | 1000 | 1 | 2 Go |

TABLE 1 – Dimensionnement des machines virtuelles

## 2.3 Résultats de Simulation

Le scénario teste 500 tâches simultanées avec une longueur de calcul aléatoire (2000–7000 instructions) et un ordonnancement TimeShared.

**Observations clés :**

— **Taux de succès** : 100% de traitement

— **Temps de réponse** : 36–71s pour les tâches prioritaires

— **Performance** : Confirmation qu'un serveur bien dimensionné suffit

La simulation valide la viabilité technique et économique de l'architecture proposée.

# 3 Déploiement OpenStack

## 3.1 Installation avec DevStack

DevStack permet un déploiement rapide d'OpenStack pour environnements de test et formation. Dans ce projet, nous avons utilisé MicroStack, une distribution simplifiée d'OpenStack.

### 3.1.1 Prérequis Système

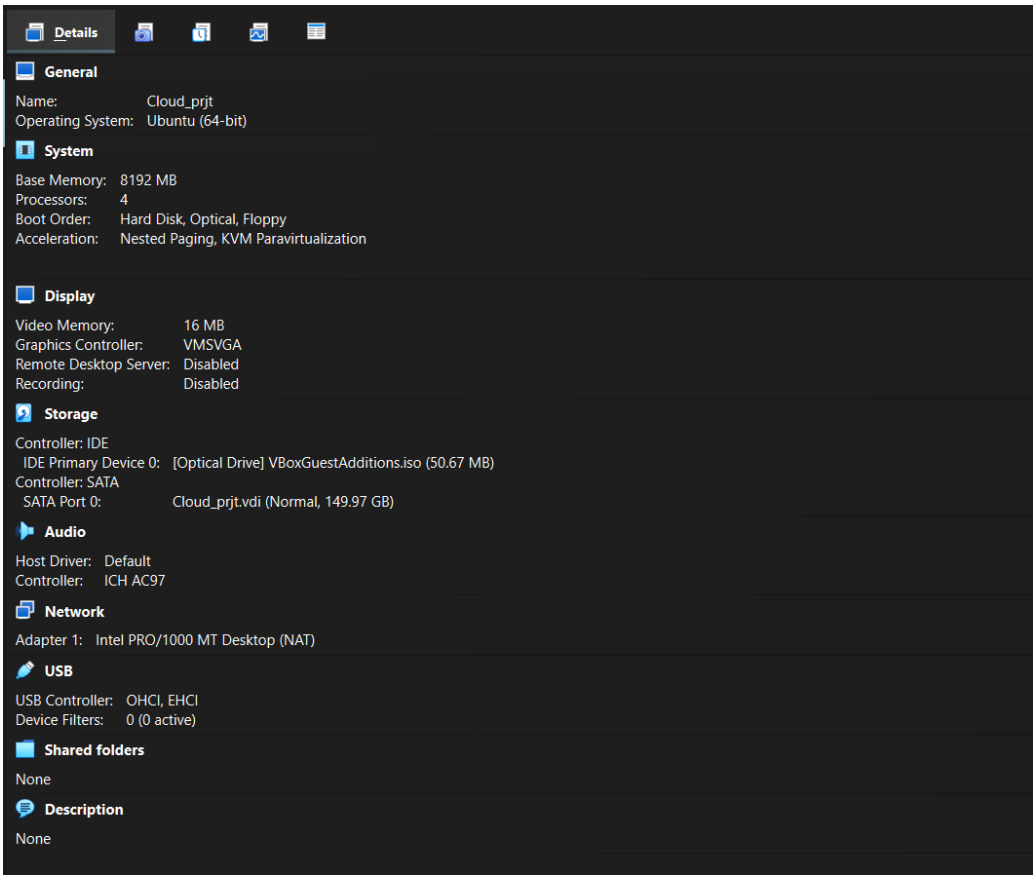


FIGURE 1 – Configuration de la machine virtuelle hôte (Ubuntu 64-bit, 8GB RAM, 4 vCPU)

Configuration matérielle :
— **Système** : Ubuntu 64-bit
— **Mémoire** : 8192 MB (8 GB)
— **Processeurs** : 4 vCPU
— **Virtualisation** : KVM Paravirtualization
— **Stockage** : 149.97 GB
— **Réseau** : Intel PRO/1000 MT (NAT)

### 3.1.2 Procédure d'Installation

```
# Installation de MicroStack via Snap
sudo snap install microstack --beta

# Initialisation de l'environnement
sudo microstack init --auto

# Verification de l'installation
microstack.openstack service list
```

Listing 1 – Installation MicroStack

### 3.1.3 Interface Horizon

Accès à l'interface web de gestion OpenStack :

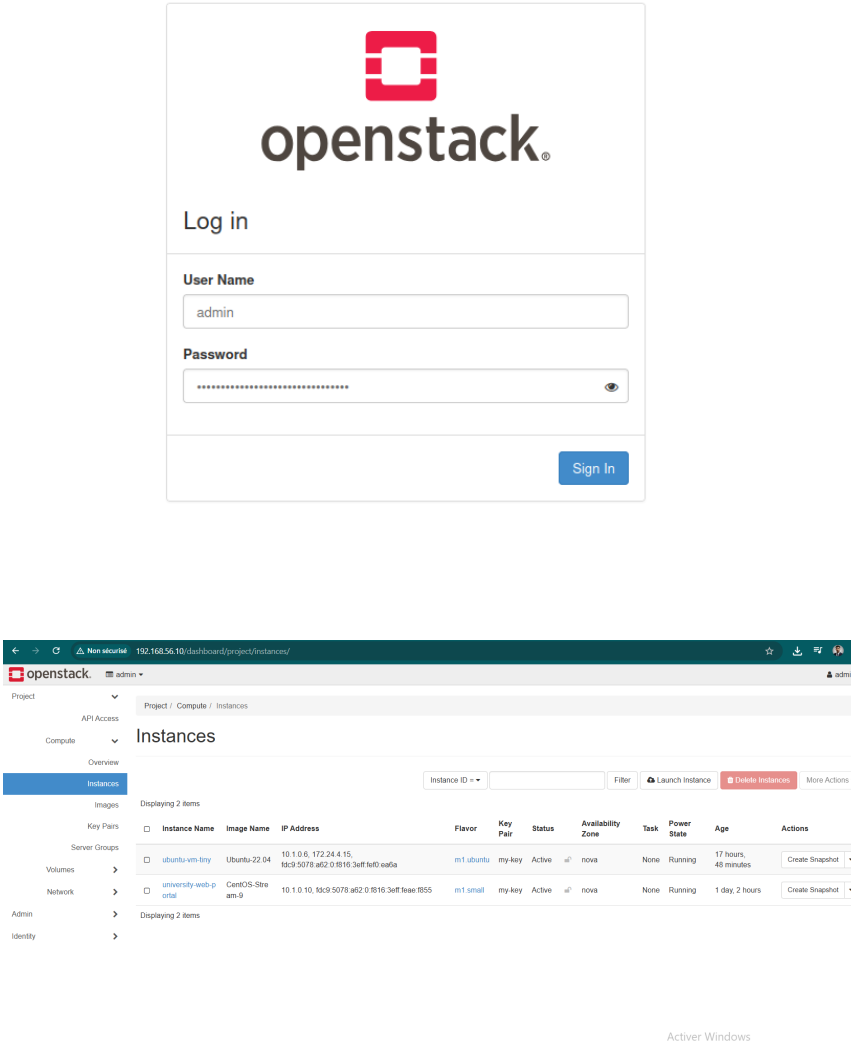


FIGURE 2 – Page de connexion Horizon (Dashboard OpenStack)

L'interface Horizon permet la gestion graphique complète de l'infrastructure :
— Gestion des instances (création, suppression, snapshots)
— Configuration réseau (réseaux virtuels, routeurs, IP flottantes)
— Gestion du stockage (volumes, images)
— Administration des utilisateurs et projets

## 3.2 Vérification des Services

Après installation, vérification des composants essentiels :

```
microstack.openstack service list
microstack.openstack hypervisor list
```

Listing 2 – Vérification des services

```
cisco@cisco:~$ microstack.openstack service list
+----------------------------------+-----------+-----------+
| ID                               | Name      | Type      |
+----------------------------------+-----------+-----------+
| 01abf291a9f74a908bb8ed8d80f18a4a | neutron   | network   |
| 087988ba21b34306b59cc7e9fe9c9f42 | placement | placement |
| 47328e8fca174c058af41e322279d863 | keystone  | identity  |
| 52a6ba94f677406d99d4524ff47feb38 | cinderv2  | volumev2  |
| 7909f77f1368406db5a30eae981853ea | glance    | image     |
| 8628e3d4805b4623a1555f3829eaac18 | nova      | compute   |
| d29a9e600e004a7583de595fa97136f3 | cinderv3  | volumev3  |
+----------------------------------+-----------+-----------+
```

FIGURE 3 – Liste complète des services OpenStack (Neutron, Nova, Glance, Keystone, Cinder)

```
cisco@cisco:~$ microstack.openstack hypervisor list
+----+---------------------+-----------------+-----------+-------+
| ID | Hypervisor Hostname | Hypervisor Type | Host IP   | State |
+----+---------------------+-----------------+-----------+-------+
|  1 | cisco               | QEMU            | 10.0.2.15 | up    |
+----+---------------------+-----------------+-----------+-------+
```

FIGURE 4 – Vérification de l'hyperviseur QEMU (État : UP)

Tous les services (Nova, Neutron, Glance, Horizon, Keystone, Cinder) sont opérationnels.

## 3.3 Configuration Réseau et Sécurité

### 3.3.1 Réseau Virtuel

Configuration d'un réseau privé avec routeur et passerelle publique pour permettre l'accès externe aux instances.

### 3.3.2 Security Groups

Configuration des règles de pare-feu pour autoriser les flux critiques :

```
cisco@cisco:~$ microstack.openstack security group rule list default
+--------------------------------------+-------------+-----------+-----------+------------+--------------------------------------+
| ID                                   | IP Protocol | Ethertype | IP Range  | Port Range | Remote Security Group                |
+--------------------------------------+-------------+-----------+-----------+------------+--------------------------------------+
| 0f501823-e81c-43ee-bdbc-51f6571922ff | None        | IPv4      | 0.0.0.0/0 |            | None                                 |
| 1c3cb567-1c16-4fa5-9600-55583be65b14 | tcp         | IPv4      | 0.0.0.0/0 | 22:22      | None                                 |
| a9b9d4b4-40ec-410d-8c40-294a8ad0ca20 | None        | IPv6      | ::/0      |            | None                                 |
| bdd52b15-77cf-4dcf-9f8a-881dbfd50ea7 | icmp        | IPv4      | 0.0.0.0/0 |            | None                                 |
| d40b84fd-2dc7-49cf-9ae6-46a6447bd437 | None        | IPv6      | ::/0      |            | 3563a6a0-45a8-48f8-9209-8e50a4c98c21 |
| f46dbfff-0415-4ebf-8c9a-8c36eb77fab4 | None        | IPv4      | 0.0.0.0/0 |            | 3563a6a0-45a8-48f8-9209-8e50a4c98c21 |
+--------------------------------------+-------------+-----------+-----------+------------+--------------------------------------+
cisco@cisco:~$
```

FIGURE 5 – Règles de sécurité configurées (ICMP, SSH port 22, IPv4/IPv6)

Flux autorisés :
— ICMP (ping) - IPv4 et IPv6
— SSH (port 22) - TCP
— Tout trafic IPv4/IPv6 sortant

## 3.4 Déploiement SaaS

### 3.4.1 Instance CirrOS de Test

Déploiement d'une première instance pour validation de l'infrastructure :

```
cisco@cisco:~$ microstack.openstack server list
+--------------------------------------+-------------+--------+---------------------------------+--------+---------+
| ID                                   | Name        | Status | Networks                        | Image  | Flavor  |
+--------------------------------------+-------------+--------+---------------------------------+--------+---------+
| cfa7508c-dfa2-4dd5-a98e-51cc07f414e0 | my-cirros-vm | ACTIVE | test=192.168.222.133, 10.20.20.82 | cirros | m1.tiny |
+--------------------------------------+-------------+--------+---------------------------------+--------+---------+
cisco@cisco:~$
```

FIGURE 6 – Instance CirrOS active avec double interface réseau (test=192.168.222.133, 10.20.20.82)

**Validation de la connectivité réseau** :

```
cisco@cisco:~$ ping -c 4 10.20.20.82
PING 10.20.20.82 (10.20.20.82) 56(84) bytes of data.
64 bytes from 10.20.20.82: icmp_seq=1 ttl=63 time=4.84 ms
64 bytes from 10.20.20.82: icmp_seq=2 ttl=63 time=15.0 ms
64 bytes from 10.20.20.82: icmp_seq=3 ttl=63 time=2.16 ms
64 bytes from 10.20.20.82: icmp_seq=4 ttl=63 time=1.91 ms

--- 10.20.20.82 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3433ms
rtt min/avg/max/mdev = 1.907/5.966/14.962/5.319 ms
cisco@cisco:~$
```

FIGURE 7 – Test de connectivité ICMP réussi (0% packet loss, RTT moyen : 5.966ms)

**Accès SSH et vérification système** :

```
cisco@cisco:~$ ssh -i mykey.pem cirros@10.20.20.82
sign_and_send_pubkey: no mutual signature supported
cirros@10.20.20.82's password:
$ ls
$ hostname
cirros
$ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1442 qdisc pfifo_fast qlen 1000
    link/ether fa:16:3e:b9:b9:a7 brd ff:ff:ff:ff:ff:ff
    inet 192.168.222.133/24 brd 192.168.222.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:feb9:b9a7/64 scope link
       valid_lft forever preferred_lft forever
$
```

FIGURE 8 – Connexion SSH réussie et affichage de la configuration réseau de l'instance

Paramètres de l'instance :
— **Image** : CirrOS
— **Flavor** : m1.tiny

— **Réseaux** : test (192.168.222.133) et 10.20.20.82
— **État** : ACTIVE

### 3.4.2 Service Web Production

Pour la partie SaaS (Software as a Service), déploiement d'un service web fonctionnel.

1. **Téléchargement** : Image CentOS Stream 7
2. **Intégration** : Ajout dans Glance
3. **Configuration** : Démarrage automatique du service

```
# Creation du repertoire et contenu HTML
mkdir -p /var/www/html
echo "<h1>UAE Cloud - Portail SaaS</h1>" > /var/www/html/index.html

# Lancement du serveur web
cd /var/www/html
python -m SimpleHTTPServer 80
```

Listing 3 – Déploiement du serveur web

**Validation** : Service web accessible et opérationnel via l'IP flottante assignée.

# 4 Infrastructure as Code

L'approche IaC garantit reproductibilité et élimine les erreurs manuelles via l'automatisation complète.

## 4.1 Provisionnement avec Terraform

### 4.1.1 Architecture des Fichiers

— `provider.tf` : Connexion API OpenStack
— `variables.tf` : Paramètres configurables
— `main.tf` : Définition des ressources

```
resource "openstack_compute_instance_v2" "ubuntu_instance" {
  name            = var.instance_name
  image_name      = var.image_name
  flavor_name     = var.flavor_name
  key_pair        = openstack_compute_keypair_v2.terraform_key.name
  config_drive    = true
  security_groups = ["default", openstack_networking_secgroup_v2.
     terraform_sg.name]

  network {
    uuid = data.openstack_networking_network_v2.network.id
  }
}
```

Listing 4 – Définition de l'instance

### 4.1.2 Déploiement

```
terraform init
terraform plan
terraform apply
```

Listing 5 – Application Terraform

**Résultat** : 10 ressources créées (réseau, sécurité, clés SSH, instance) en quelques secondes.

## 4.2 Configuration avec Ansible

### 4.2.1 Inventaire Dynamique

Fichier `inventory.ini` généré automatiquement avec l'IP flottante :

```
[webservers]
172.24.4.154 ansible_user=ubuntu ansible_ssh_private_key_file=~/.
   ssh/terraform_key
```

Listing 6 – Inventaire Ansible

### 4.2.2 Playbook de Déploiement

```
---
- name: Configuration serveur web
  hosts: webservers
  become: yes
  tasks:
    - name: Mise a jour systeme
      apt:
        update_cache: yes

    - name: Installation Nginx
      apt:
        name: nginx
        state: present

    - name: Demarrage Nginx
      service:
        name: nginx
        state: started
        enabled: yes
```

Listing 7 – Playbook Nginx

### 4.2.3 Exécution et Validation

```
ansible-playbook -i inventory.ini playbook.yml
```

Listing 8 – Exécution du playbook

**Résultat** : Service web accessible et fonctionnel, validation via navigateur à l'adresse IP flottante.

# 5 Monitoring et SLA

## 5.1 Script de Surveillance

Développement d'un outil Python pour vérifier la disponibilité en temps réel :

```
import requests
import time

def check_availability(url):
    try:
        response = requests.get(url, timeout=5)
        return response.status_code == 200
    except:
        return False

# Surveillance toutes les 5 minutes
while True:
    status = check_availability("http://172.24.4.154")
    print(f"Disponibilite: {status}")
    time.sleep(300)
```

Listing 9 – Monitoring SLA

## 5.2 Résultats

**Métriques obtenues :**

— **Disponibilité actuelle** : 100.00%
— **Moyenne historique** : Au-dessus du seuil SLA
— **Statut opérationnel** : OK

Cette couche de gouvernance complète le cycle de vie du Cloud en assurant la qualité de service.

# 6 Défis et Solutions

## 6.1 Phase Simulation

**Problème** : Erreurs d'allocation ressources (Host/VM mismatch)
**Solution** : Redimensionnement du serveur à 10 cœurs @ 3000 MIPS

## 6.2 Infrastructure OpenStack

**Problème** : Instabilité DevStack lors de l'installation
**Solution** : Allocation de 8 Go RAM minimum et utilisation de `unstack.sh` pour nettoyage

## 6.3 Automatisation IaC

**Problème** : Connectivité SSH entre Terraform et Ansible
**Solution** : Activation de `config_drive = true` et désactivation de StrictHostKeyChecking
**Problème** : Gestion dynamique des IP flottantes
**Solution** : Génération automatique de l'inventaire Ansible via template Terraform

## 6.4 Monitoring

**Problème** : Erreurs d'authentification API OpenStack (401)
**Solution** : Chargement systématique du fichier `openrc` avant exécution du script

# 7 Conclusion

Ce projet démontre la maîtrise complète du cycle de vie d'une infrastructure Cloud moderne. De la simulation théorique au monitoring opérationnel, chaque phase apporte une valeur ajoutée mesurable.

**Contributions principales :**
— Validation économique via simulation CloudSim
— Infrastructure IaaS fonctionnelle avec OpenStack
— Automatisation complète via IaC (Terraform + Ansible)
— Gouvernance et respect des SLA

**Gains quantifiables :**
— Réduction du temps de déploiement à quelques secondes
— Élimination des erreurs de configuration manuelle
— Disponibilité de service à 100%

Ce travail constitue une base solide pour aborder les défis modernes du Cloud-Native et de la culture DevOps, tout en répondant concrètement aux besoins de l'Université Abdelmalek Essaâdi.