CLOUD COMPUTING - PROJET

SAMIR TAOUS **ENCADREMENT : PR. CHAKER AMRANI**

Phase I : Cloudsim

Ce programme permet de simuler **numériquement** l'infrastructure Cloud de l'Université Abdelmalek Essaadi pour tester sa viabilité sans coût matériel

```
package org.cloudbus.cloudsim.examples;

import org.cloudbus.cloudsim.*;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.*;
import java.util.*;

public class UAESimulation {

   public static void main(String[] args) {
       try {
           // 1. Initialisation de CloudSim
           CloudSim.init(1, Calendar.getInstance(), false);

           // 2. Création du Datacenter UAE (Infrastructure physique)
           Datacenter uaeDatacenter = createDatacenter("UAE_Datacenter");

           // 3. Création du Broker (Gestionnaire de requêtes)
           DatacenterBroker uaeBroker = new
DatacenterBroker("UAE_Broker");
           int brokerId = uaeBroker.getId();

           // 4. Création des VMs pour les services de l'Université
           List<Vm> vms = new ArrayList<Vm>();

           // VM Moodle (E-learning) : 4GB RAM, 2 CPUs
           vms.add(new Vm(0, brokerId, 1000, 2, 4096, 1000, 10000, "Xen",
new CloudletSchedulerTimeShared()));

           // VM Scolarité (Inscriptions) : 2GB RAM, 1 CPU
           vms.add(new Vm(1, brokerId, 1000, 1, 2048, 1000, 10000, "Xen",
new CloudletSchedulerTimeShared()));


           uaeBroker.submitGuestList(vms);

           // 5. Création des Cloudlets (Charges de travail des
étudiants)
           List<Cloudlet> cloudlets = new ArrayList<Cloudlet>();
           UtilizationModel model = new UtilizationModelFull();

           for (int i = 0; i < 10; i++) {
               Cloudlet cl = new Cloudlet(i, 20000, 1, 300, 300, model,
model, model);
               cl.setUserId(brokerId);
               cl.setVmId(i % 2); // Alterne entre Moodle et Scolarité
               cloudlets.add(cl);
           }

           // Commande exacte pour les Cloudlets
           uaeBroker.submitCloudletList(cloudlets);

           // 6. Démarrage de la simulation
```

```
            CloudSim.startSimulation();
            List<Cloudlet> results = uaeBroker.getCloudletReceivedList();
            CloudSim.stopSimulation();

            // 7. Affichage des résultats
            System.out.println("\n--- RESULTATS UAE CLOUD (120 000
ETUDIANTS SIMULÉS) ---");
            for (Cloudlet c : results) {
                String service = (c.getVmId() == 0) ? "MOODLE" :
"SCOLARITE";
                System.out.println("ID: " + c.getCloudletId() + " | Service:
" + service + " | Status: SUCCESS | Finish: " + c.getFinishTime());
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static Datacenter createDatacenter(String name) throws Exception
{
        List<Host> hostList = new ArrayList<Host>();
        List<Pe> peList = new ArrayList<Pe>();
        peList.add(new Pe(0, new PeProvisionerSimple(3000)));

        // Simuler un serveur de l'Université (16GB RAM)
        hostList.add(new Host(0, new RamProvisionerSimple(16384), new
BwProvisionerSimple(10000), 1000000, peList, new
VmSchedulerTimeShared(peList)));

        DatacenterCharacteristics characteristics = new
DatacenterCharacteristics("x86", "Linux", "Xen", hostList, 10.0, 3.0, 0.05,
0.001, 0.0);
        return new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), new LinkedList<Storage>(), 0);
    }
}
```

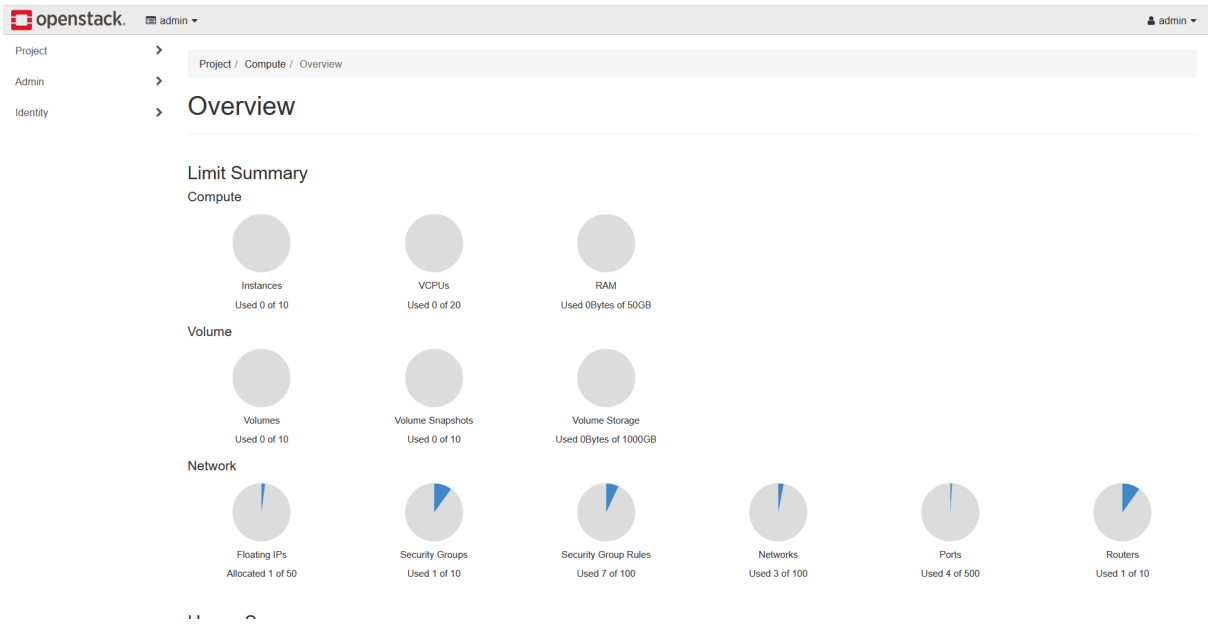
Phase II : OpenStack

1. Installation de Openstack

```
Post-stack database query stats:
+------------+-----------+-------+
| db         | op        | count |
+------------+-----------+-------+
| keystone   | SELECT    | 20515 |
| keystone   | UPDATE    |     7 |
| keystone   | INSERT    |    93 |
| neutron    | DESCRIBE  |     2 |
| neutron    | CREATE    |     1 |
| neutron    | SHOW      |     4 |
| neutron    | SELECT    |  5972 |
| neutron    | INSERT    |  4121 |
| neutron    | DELETE    |    28 |
| neutron    | UPDATE    |   134 |
| placement  | SELECT    |    26 |
| placement  | INSERT    |    69 |
| placement  | SET       |     2 |
| nova_api   | SELECT    |    50 |
| nova_cell0 | SELECT    |    27 |
| nova_cell1 | SELECT    |    95 |
| nova_cell0 | INSERT    |     4 |
| nova_cell0 | UPDATE    |     6 |
| placement  | UPDATE    |     3 |
| nova_cell1 | INSERT    |     4 |
| nova_cell1 | UPDATE    |    43 |
| cinder     | SELECT    |    57 |
| cinder     | INSERT    |     5 |
| cinder     | UPDATE    |     5 |
| glance     | INSERT    |    14 |
| glance     | SELECT    |    28 |
| glance     | UPDATE    |     2 |
| nova_api   | INSERT    |    20 |
| nova_api   | SAVEPOINT |    10 |
| nova_api   | RELEASE   |    10 |
| cinder     | DELETE    |     1 |
+------------+-----------+-------+
```

2. Test des fonctionnalités du Middleware

Le "middleware" regroupe les services d'OpenStack (Nova, Neutron, Keystone, etc.).



Liste des Services

```
samirtaous@samirtaous:~/devstack$ openstack service list
+----------------------------------+-------------+----------------+
| ID                               | Name        | Type           |
+----------------------------------+-------------+----------------+
| 31cd11f7037f4eafaf8d4ae670691d2a | cinder      | block-storage  |
| 37771616b732418ba87b90fd6f06226a | neutron     | network        |
| 4431233a134f48dcb204c831f1c2f717 | cinderv3    | volumev3       |
| a2c88de5a4bc4b519750b15fa82f72e6 | keystone    | identity       |
| a7b6b974c61f4cd6961f4afa6a4cf679 | nova_legacy | compute_legacy |
| b1445e3b332c4ab8b57aa99a426533ec | glance      | image          |
| b766a5b910424b48976139a5f88e5781 | nova        | compute        |
| fb0cda7c8ecb4328bfa422c7cea5c68a | placement   | placement      |
+----------------------------------+-------------+----------------+
```

Liste des Services Compute

```
samirtaous@samirtaous:~/devstack$ openstack compute service list
+--------------------+----------------+------------+----------+---------+-------+----------------------------+
| ID                 | Binary         | Host       | Zone     | Status  | State | Updated At                 |
+--------------------+----------------+------------+----------+---------+-------+----------------------------+
| 1a5b0e1c-e744-     | nova-conductor | samirtaous | internal | enabled | up    | 2026-01-                   |
| 4f6a-bbed-         |                |            |          |         |       | 11T00:14:03.00000          |
| e16829110195       |                |            |          |         |       | 0                          |
| 9a8ca9da-a114-     | nova-compute   | samirtaous | nova     | enabled | up    | 2026-01-                   |
| 45b4-a294-         |                |            |          |         |       | 11T00:14:05.00000          |
| 0cebe9ab90fc       |                |            |          |         |       | 0                          |
| b9e5b1b6-1c04-     | nova-scheduler | samirtaous | internal | enabled | up    | 2026-01-                   |
| 4576-86db-         |                |            |          |         |       | 11T00:13:21.00000          |
| 6000221acecc       |                |            |          |         |       | 0                          |
| 753c528c-3e58-     | nova-conductor | samirtaous | internal | enabled | up    | 2026-01-                   |
| 4b89-9850-         |                |            |          |         |       | 11T00:13:39.00000          |
| 9feb981de75f       |                |            |          |         |       | 0                          |
+--------------------+----------------+------------+----------+---------+-------+----------------------------+
```

Liste des Agents Network

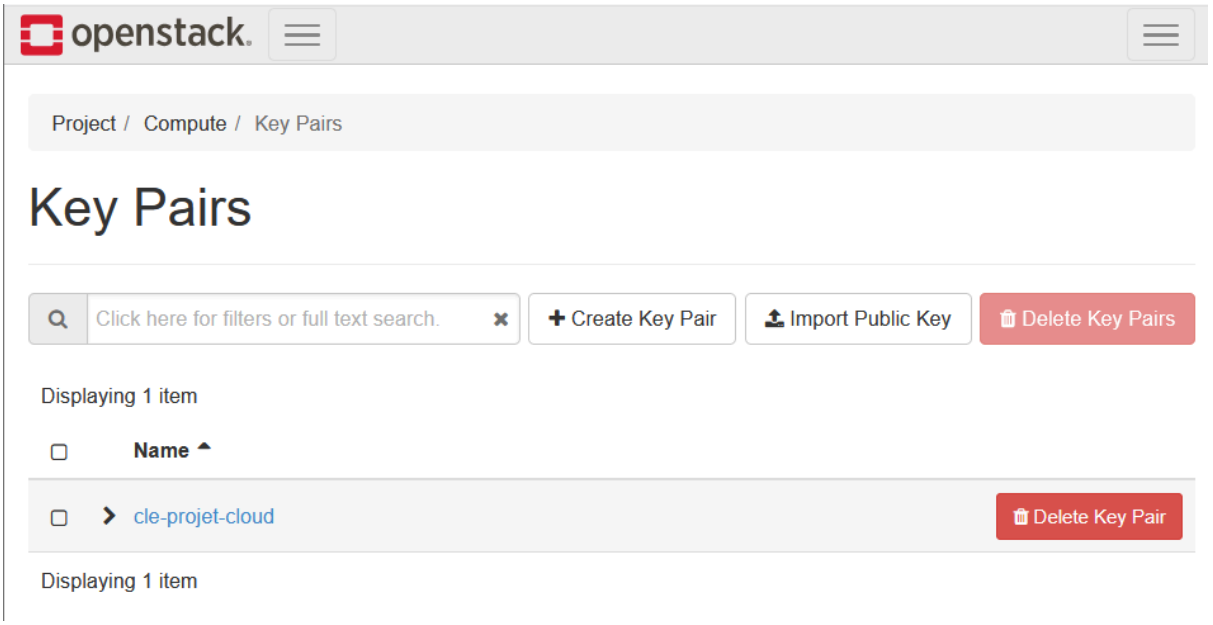
```
samirtaous@samirtaous:~/devstack$ openstack network agent list
+----------------+----------------+------------+-------------------+-------+-------+-----------------
-+
| ID             | Agent Type     | Host       | Availability Zone | Alive | State | Binary
 |
+----------------+----------------+------------+-------------------+-------+-------+-----------------
-+
| 839502f7-8c47- | OVN Controller | samirtaous |                   | :-)   | UP    | ovn-controller
 |
| 45a4-b68d-     | Gateway agent  |            |                   |       |       |
 |
| bb1376a9d17d   |                |            |                   |       |       |
 |
| 0ef893c9-9eb8- | OVN Metadata   | samirtaous |                   | :-)   | UP    | neutron-ovn-
 |
| 5128-becc-     | agent          |            |                   |       |       | metadata-agent
 |
| 0f87c1ad1534   |                |            |                   |       |       |
 |
+----------------+----------------+------------+-------------------+-------+-------+-----------------
```

Liste des images (CirrOS active)

```
samirtaous@samirtaous:~/devstack$ openstack image list
+--------------------------------------+---------------------------+--------+
| ID                                   | Name                      | Status |
+--------------------------------------+---------------------------+--------+
| c9351551-87a0-482b-bc18-c9e0e412c1bd | cirros-0.6.2-x86_64-disk | active |
+--------------------------------------+---------------------------+--------+
```

3. Implémentation IaaS basé sur cirrOS

3.1- Création key-pair



3.2- Configuration de la Sécurité

Par défaut, OpenStack bloque tout le trafic. On ouvre les "portes" pour le Ping et le SSH.

Manage Security Group Rules: default (146c51d9-ccd8-4f6d-ba4c-1509e912ff5a)

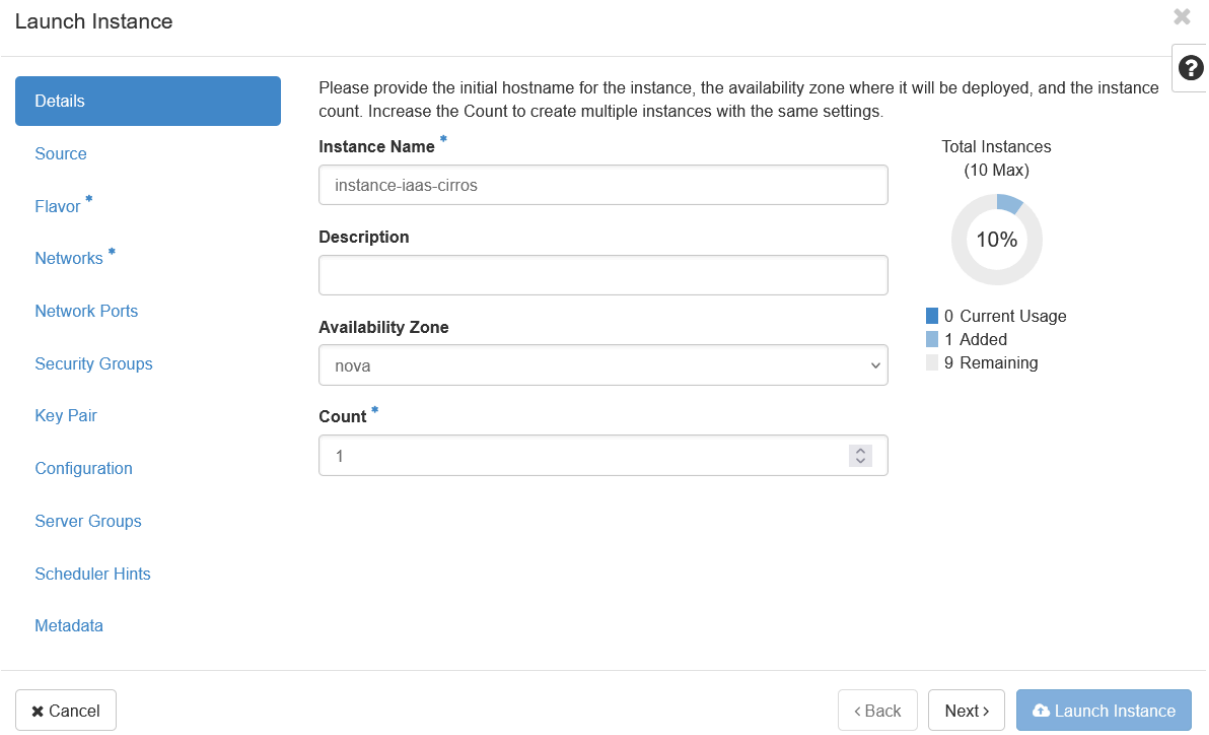
+ Add Rule | Delete Rules

Displaying 7 items

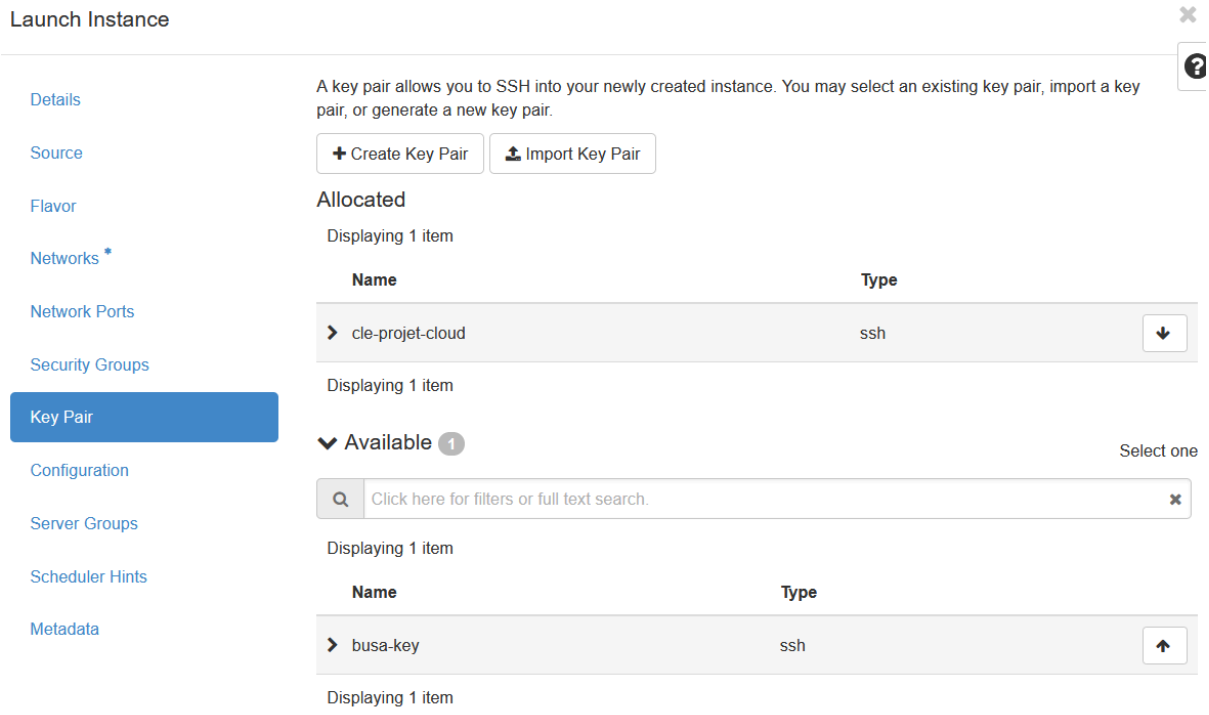
Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Description	Actions
Egress	IPv4	Any	Any	0.0.0.0/0	-	-	Delete Rule
Egress	IPv6	Any	Any	::/0	-	-	Delete Rule
Ingress	IPv4	Any	Any	-	default	-	Delete Rule
Ingress	IPv4	ICMP	Any	0.0.0.0/0	-	-	Delete Rule
Ingress	IPv4	TCP	Any	0.0.0.0/0	-	-	Delete Rule
Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0	-	-	Delete Rule
Ingress	IPv6	Any	Any	-	default	-	Delete Rule

Displaying 7 items

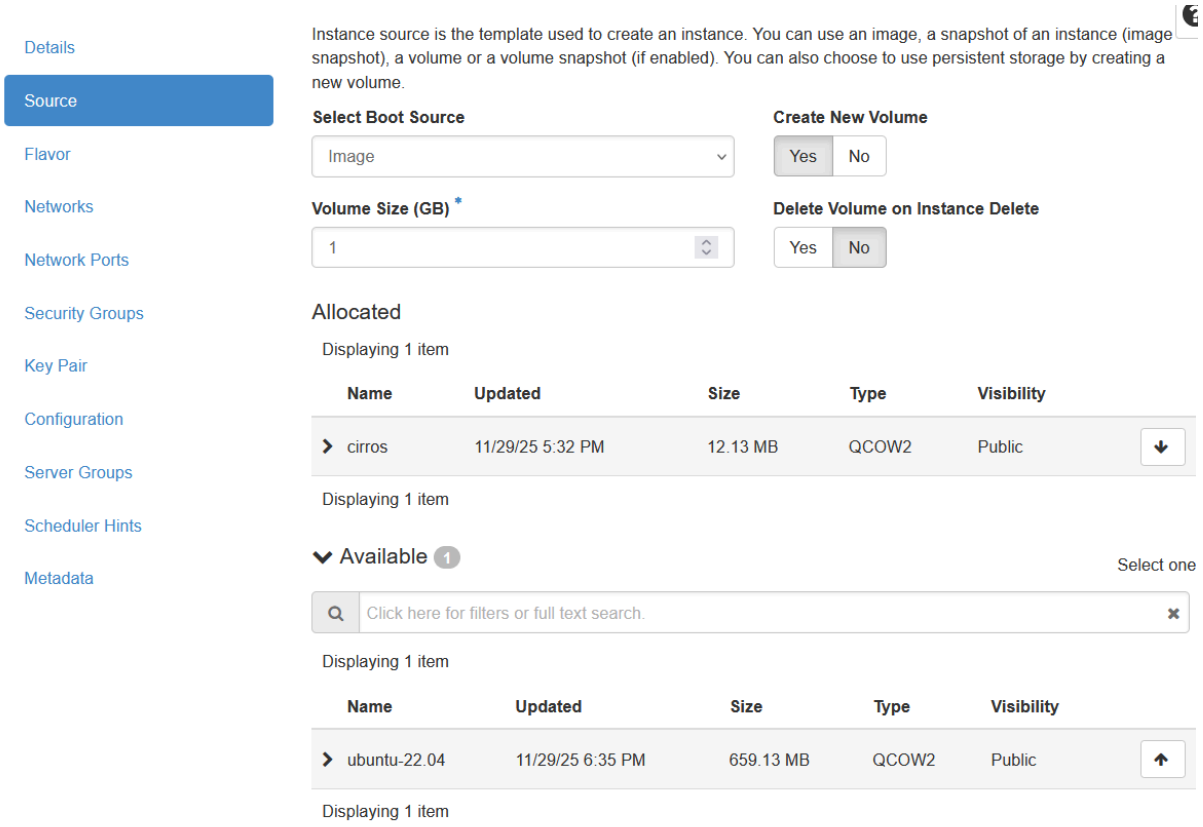
3.3- Lancement de l'instance CirrOS



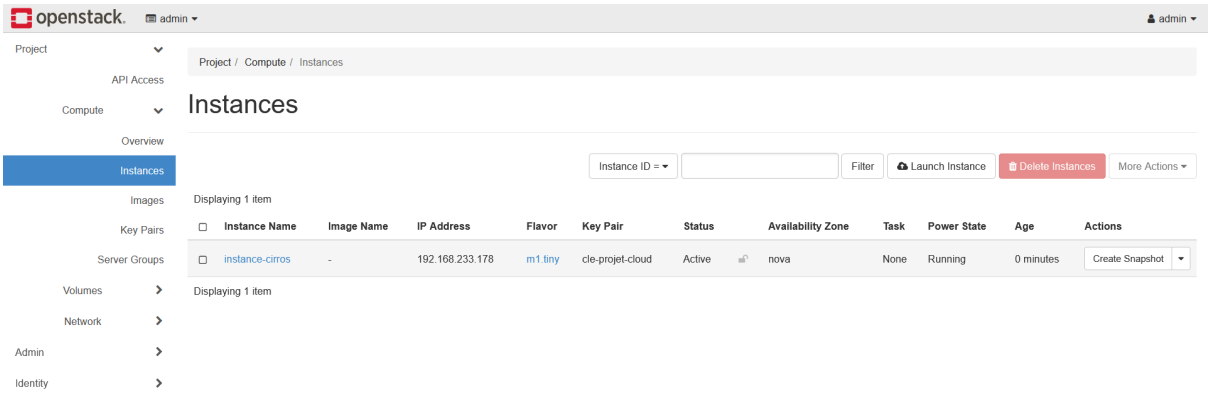
Selection de notre keypair



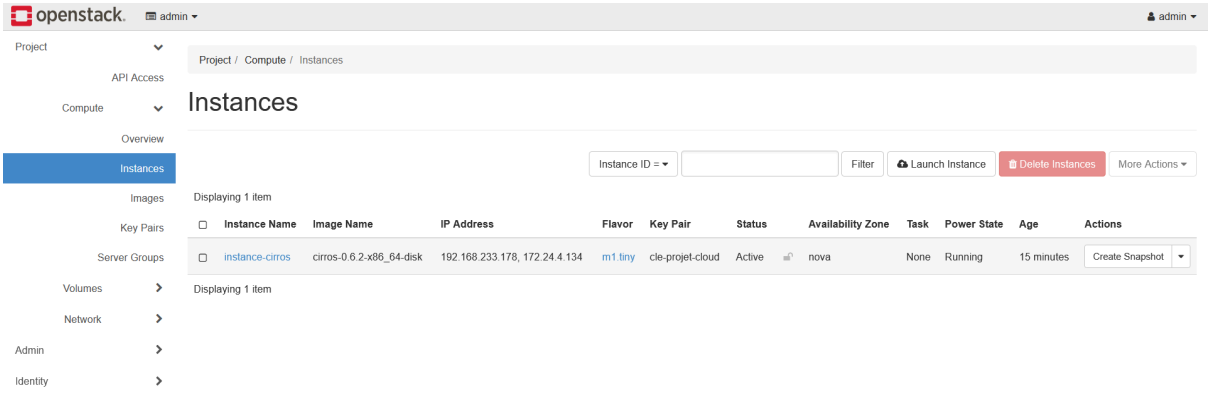
Source



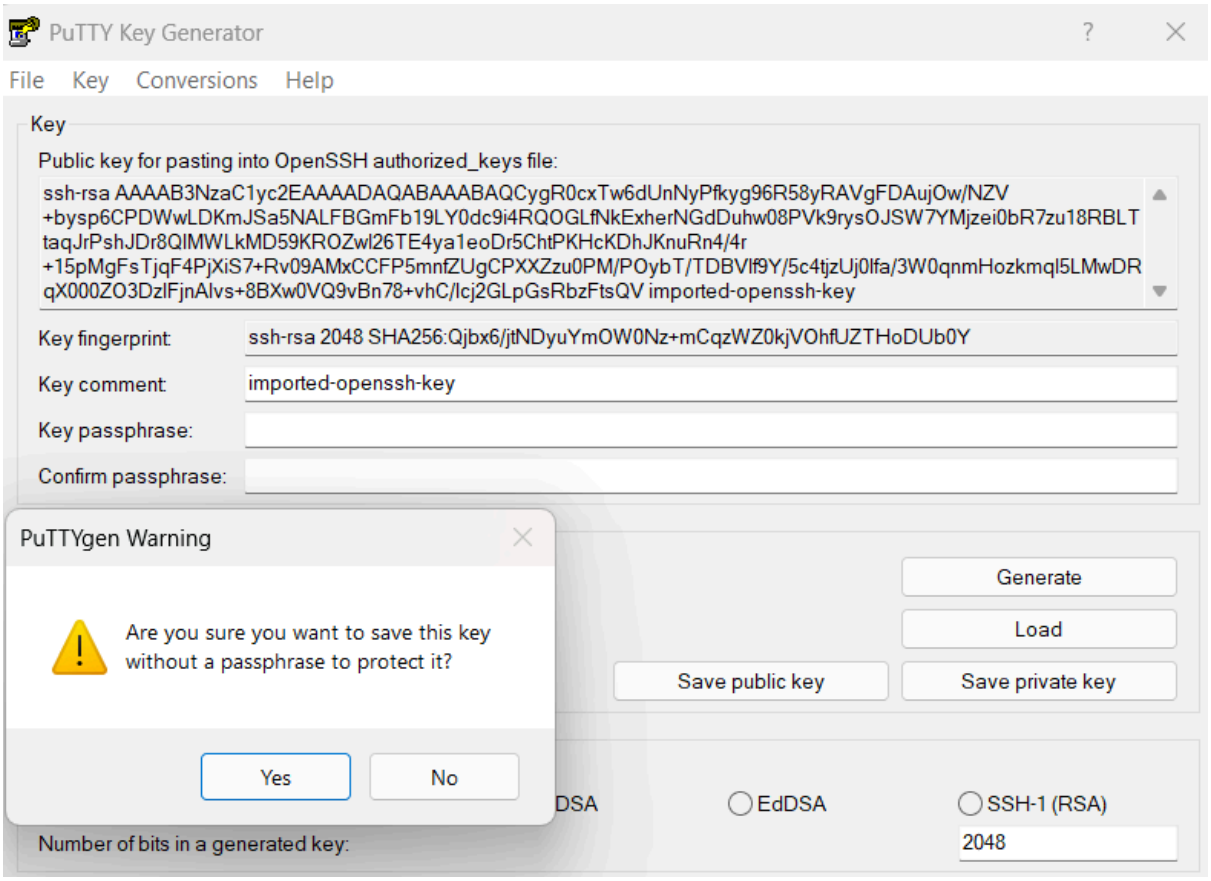
On vérifie la création de l'instance



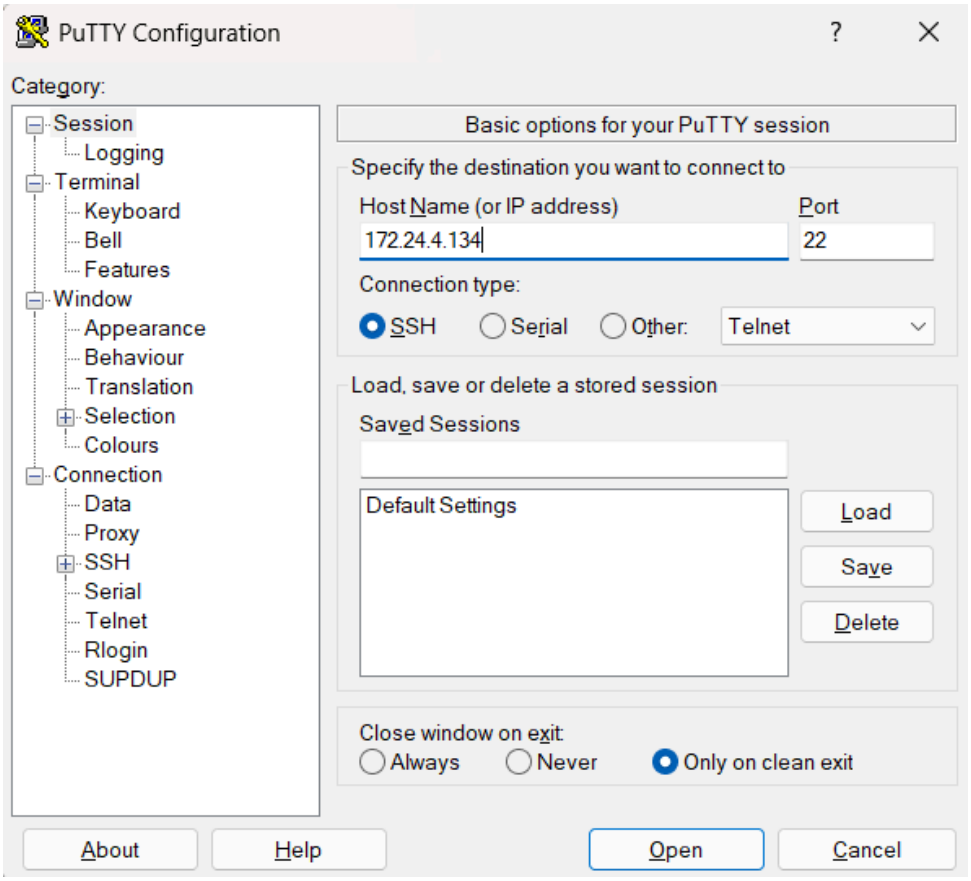
Finalement pour tester avec Putty, on associe l'instance avec une ip flottante



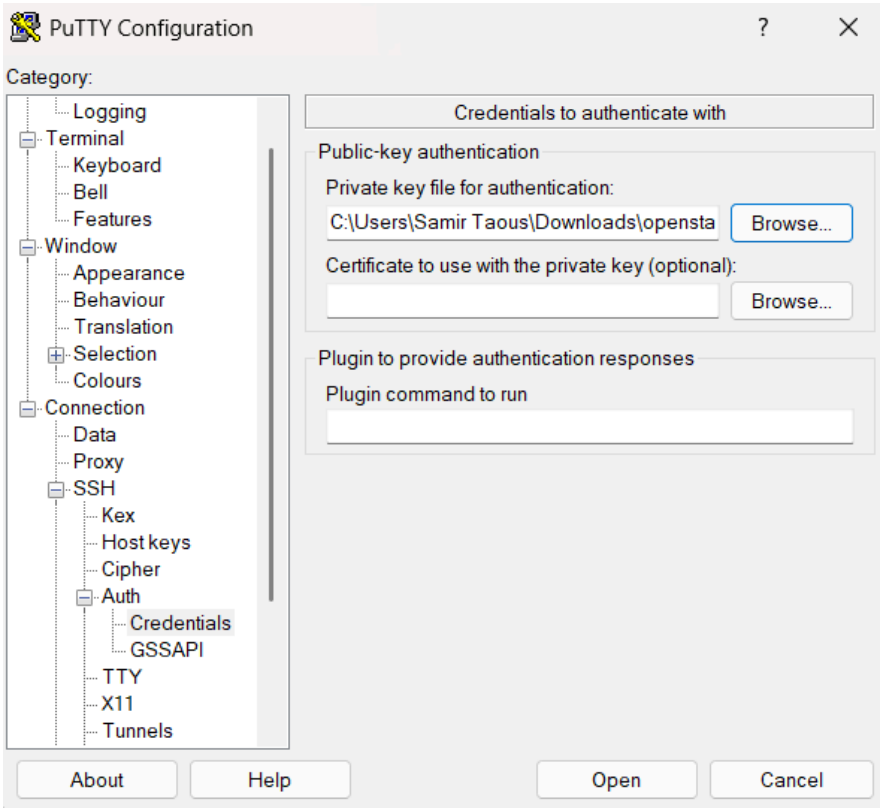
3.4. Convertissement pem to ppk



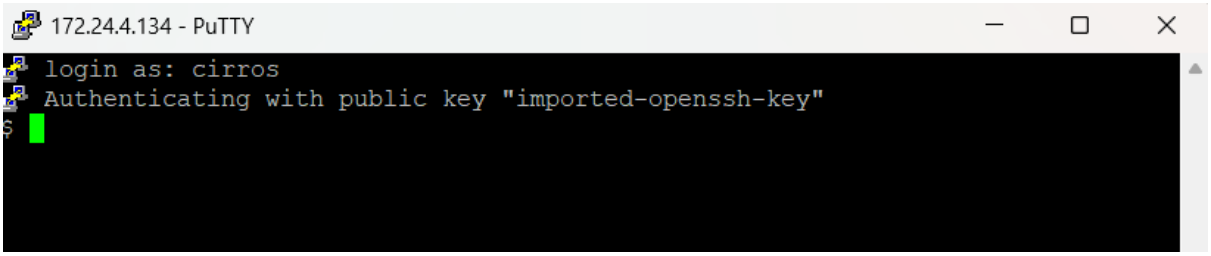
3.5. Configuration de connexion Putty



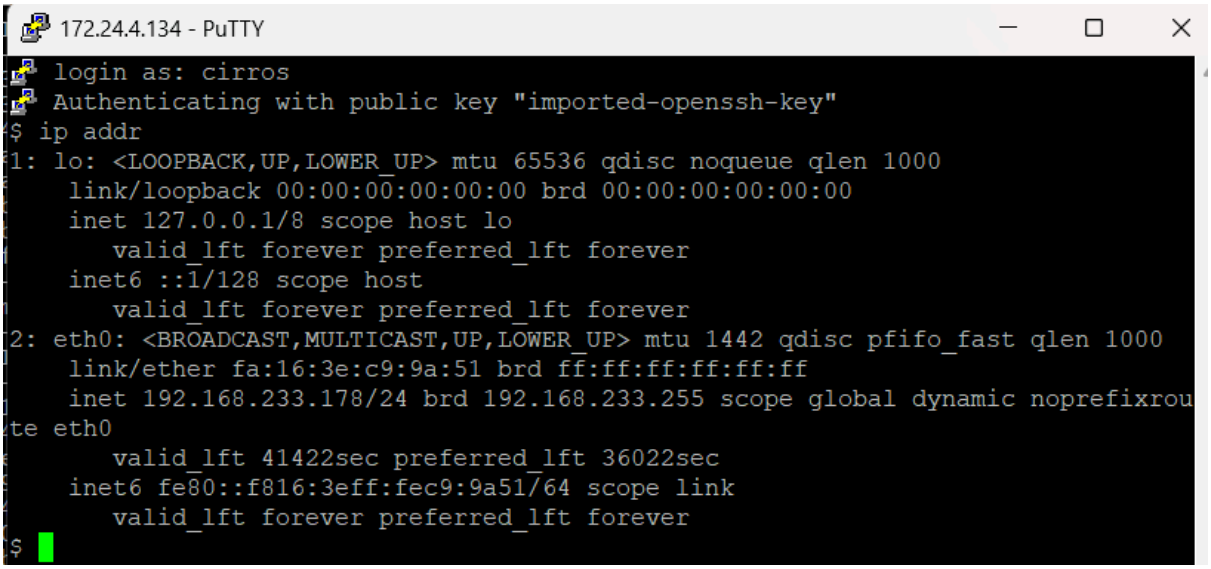
On importe la clé privée téléchargée



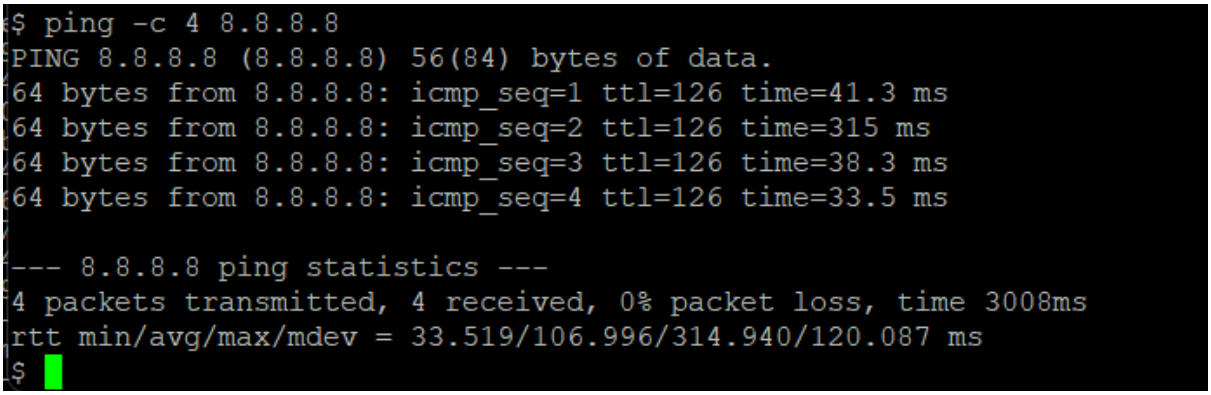
3.6. Test de connexion avec CirrOS



Ip interne assigné par OpenStack pour vérifier que le bridge est actif

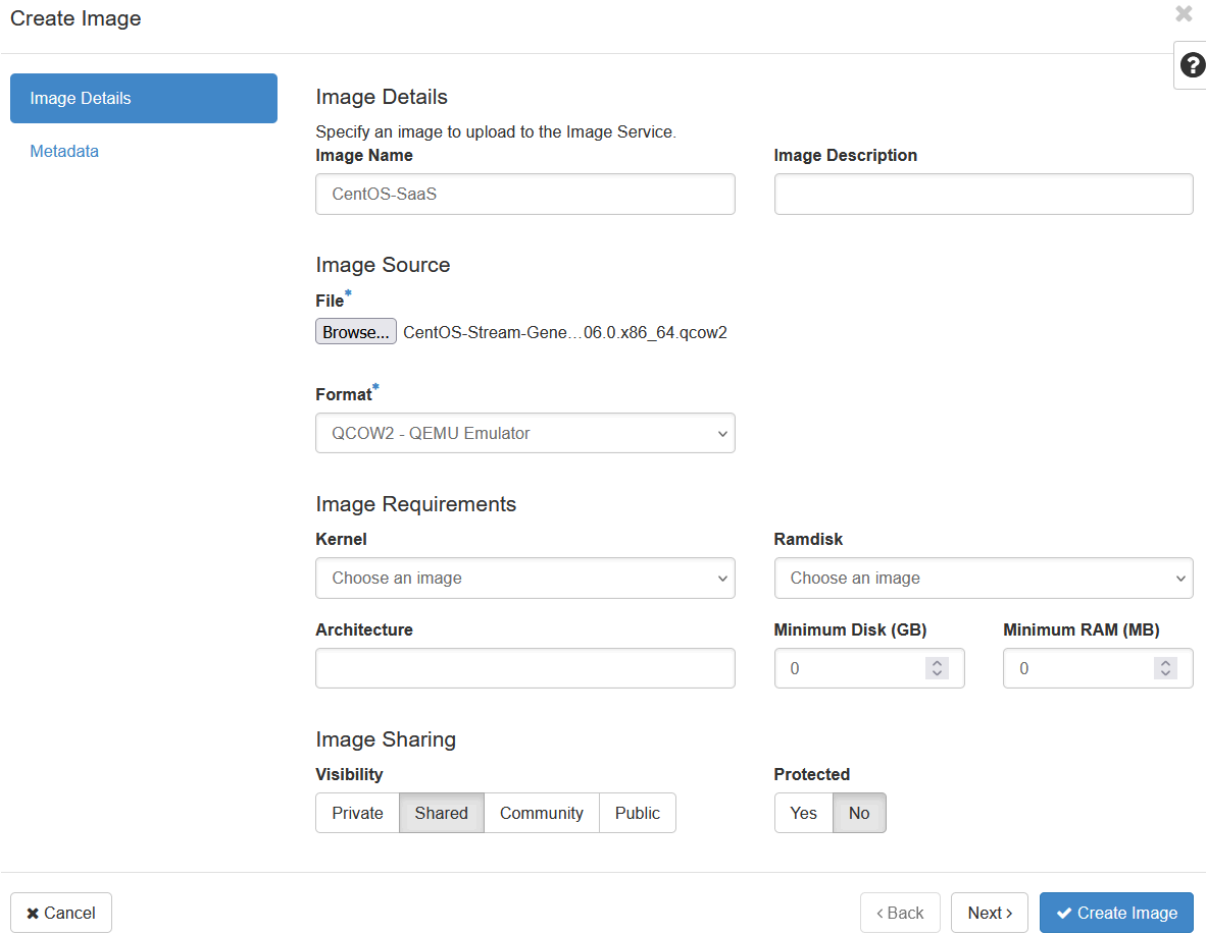


Ping pour tester que la configuration network est correcte

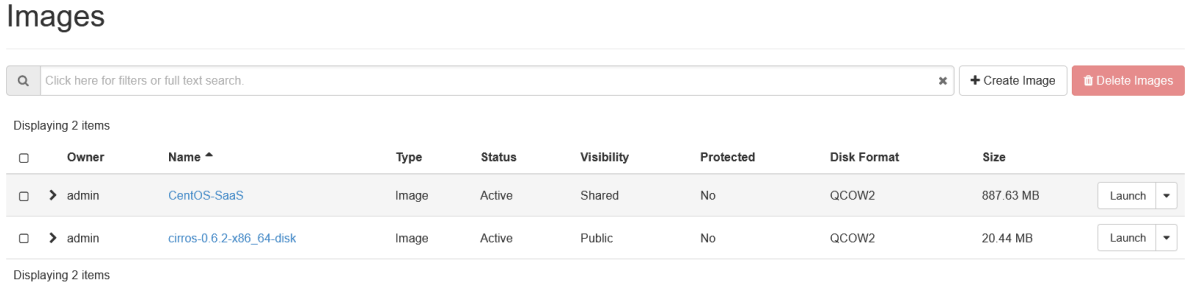


4. Implémentation d’un SaaS dans une VM CentOS

4.1. Téléversement de l’image CentOS dans Glance

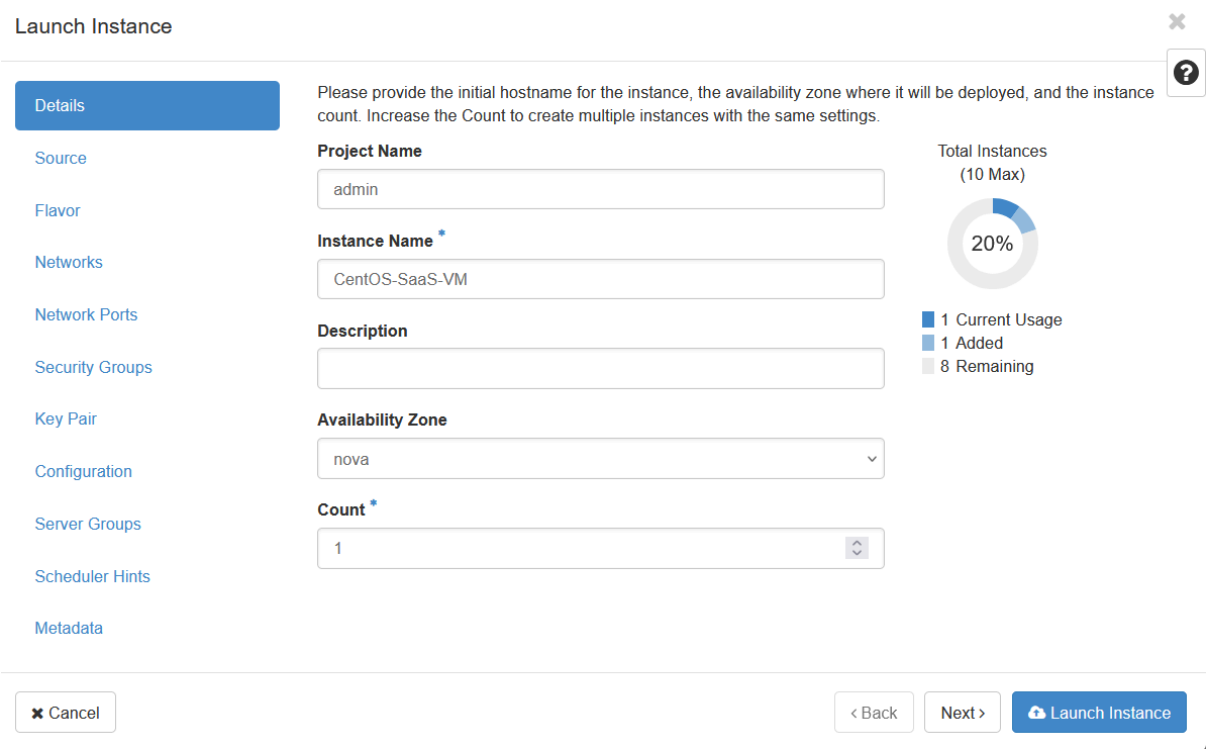


On vérifie la création de l’image CentOS-SaaS

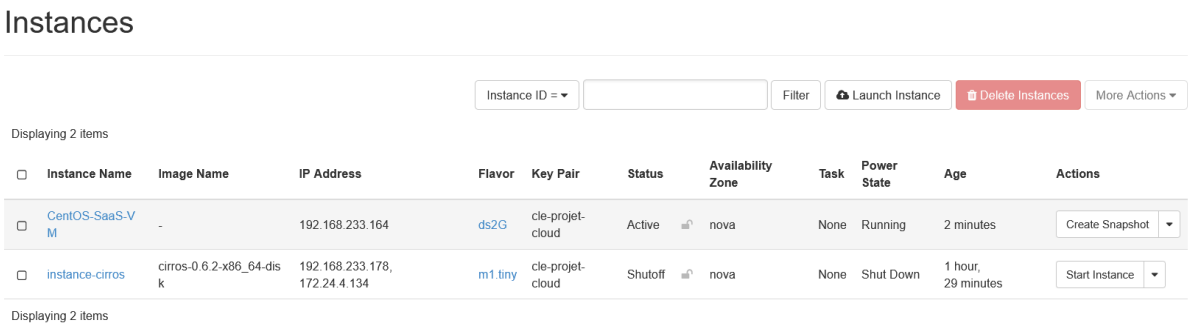


4.2. Création de l’instance CentOS (VM)

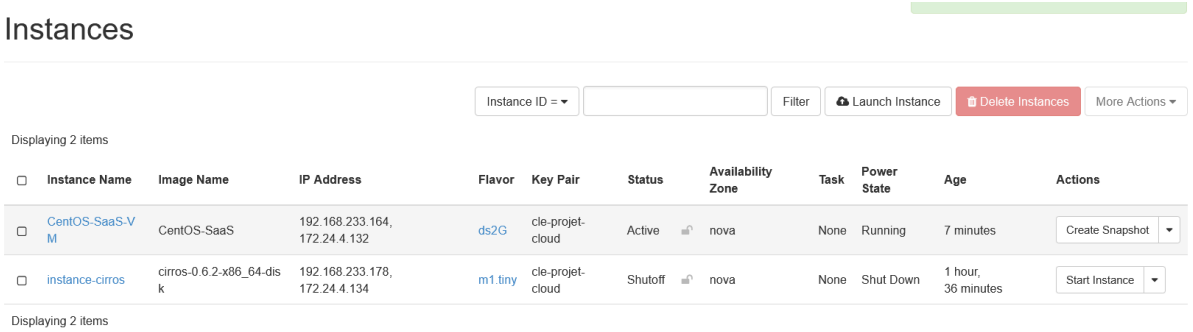
On utilise le flavor ds2G (10 gb Disk Space - 2 gb ram)



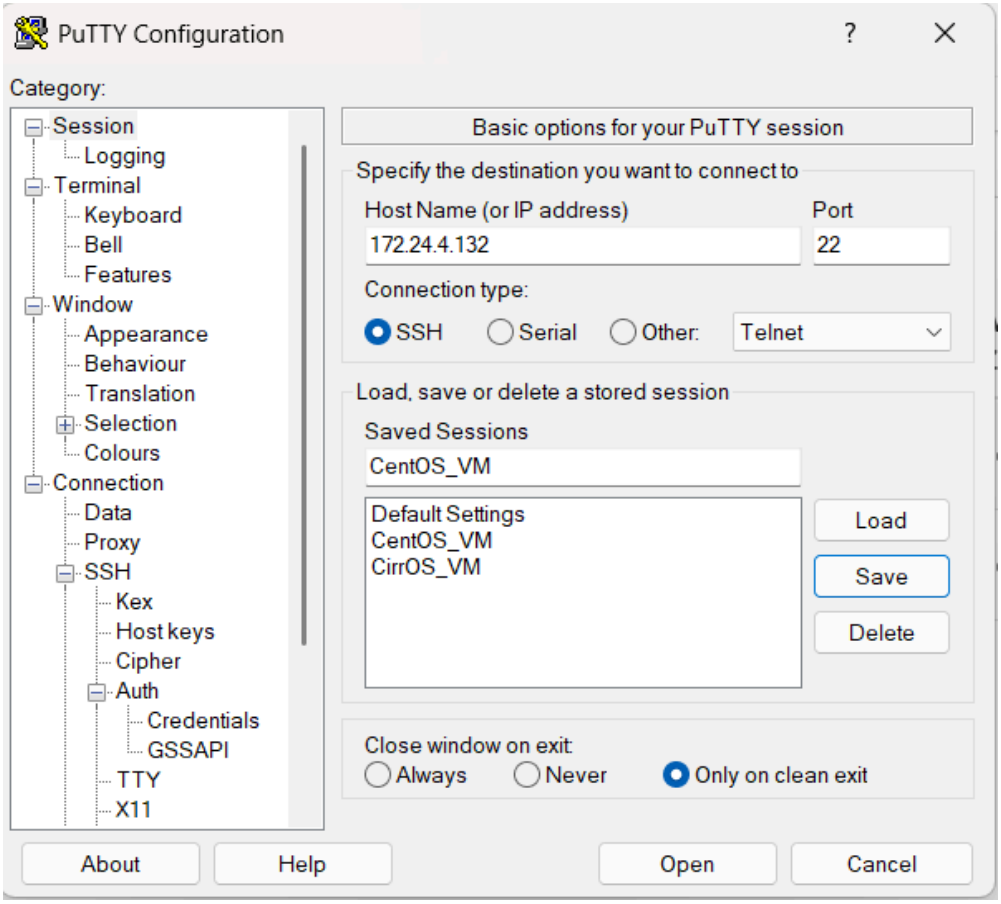
On vérifie la création de l’instance



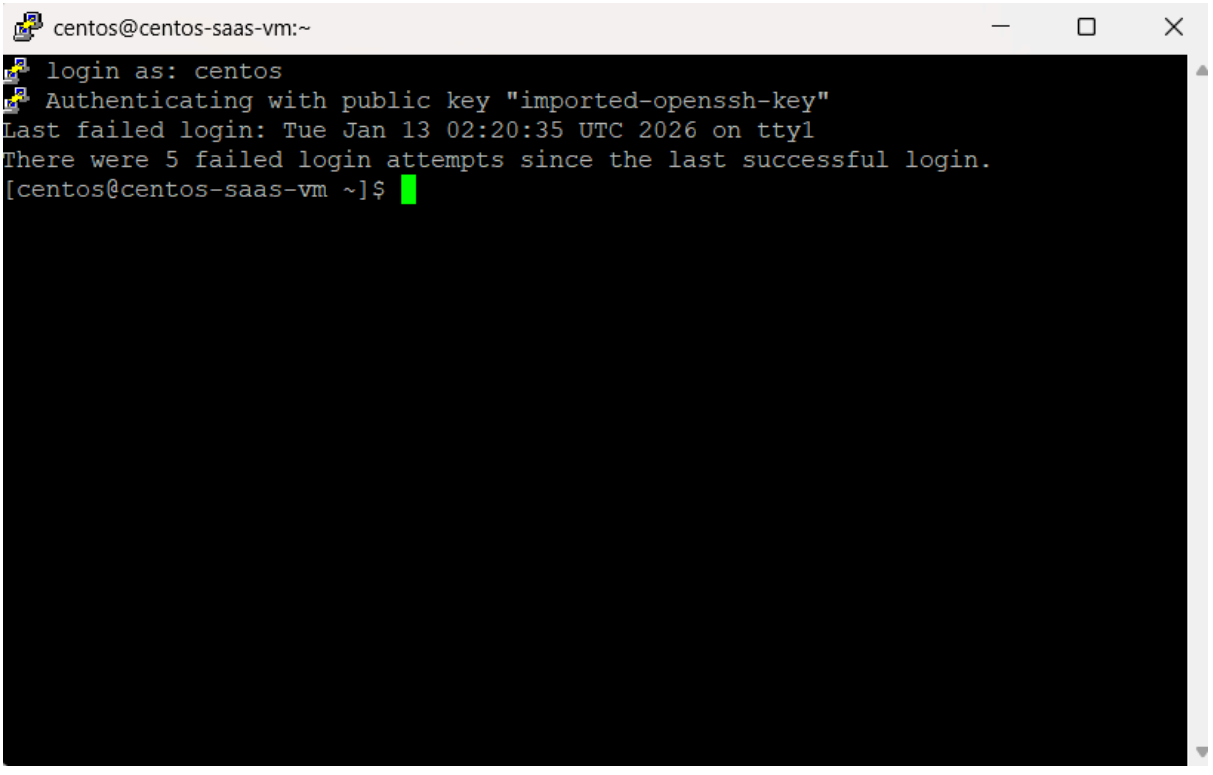
Comme pour cirrOS, on associe une ip flottante à l’instance crée



4.3. Configuration et Test avec Putty



On vérifie la connexion avec putty



4.4. Installation du Serveur Web

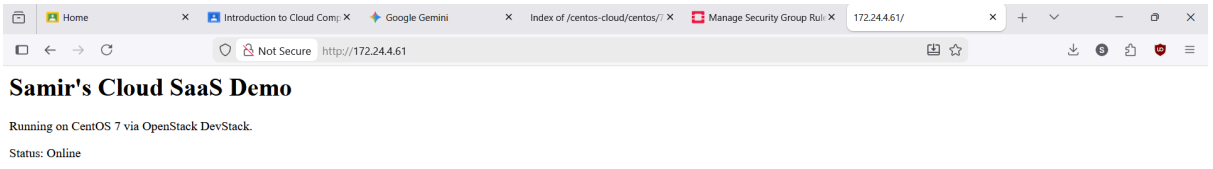
```
Installed:
  httpd.x86_64 0:2.4.6-99.el7.centos.1

Dependency Installed:
  apr.x86_64 0:1.4.8-7.el7                    apr-util.x86_64 0:1.5.2-6.el7_9.1
  httpd-tools.x86_64 0:2.4.6-99.el7.centos.1  mailcap.noarch 0:2.1.41-2.el7

Complete!
[centos@centos-saas-vm yum.repos.d]$ ~~
```

4.5. Création du Contenu SaaS

```
[centos@centos-saas-vm yum.repos.d]$ echo "<h1>Samir's Cloud SaaS Demo</h1><p>Ru
nning on CentOS 7 via OpenStack DevStack.</p><p>Status: Online</p>" | sudo tee /
var/www/html/index.html
<h1>Samir's Cloud SaaS Demo</h1><p>Running on CentOS 7 via OpenStack DevStack.</
p><p>Status: Online</p>
[centos@centos-saas-vm yum.repos.d]$
```

4.6. Test du SaaS

On ouvre le port 80 sur la VM et on teste sur la machine hôte

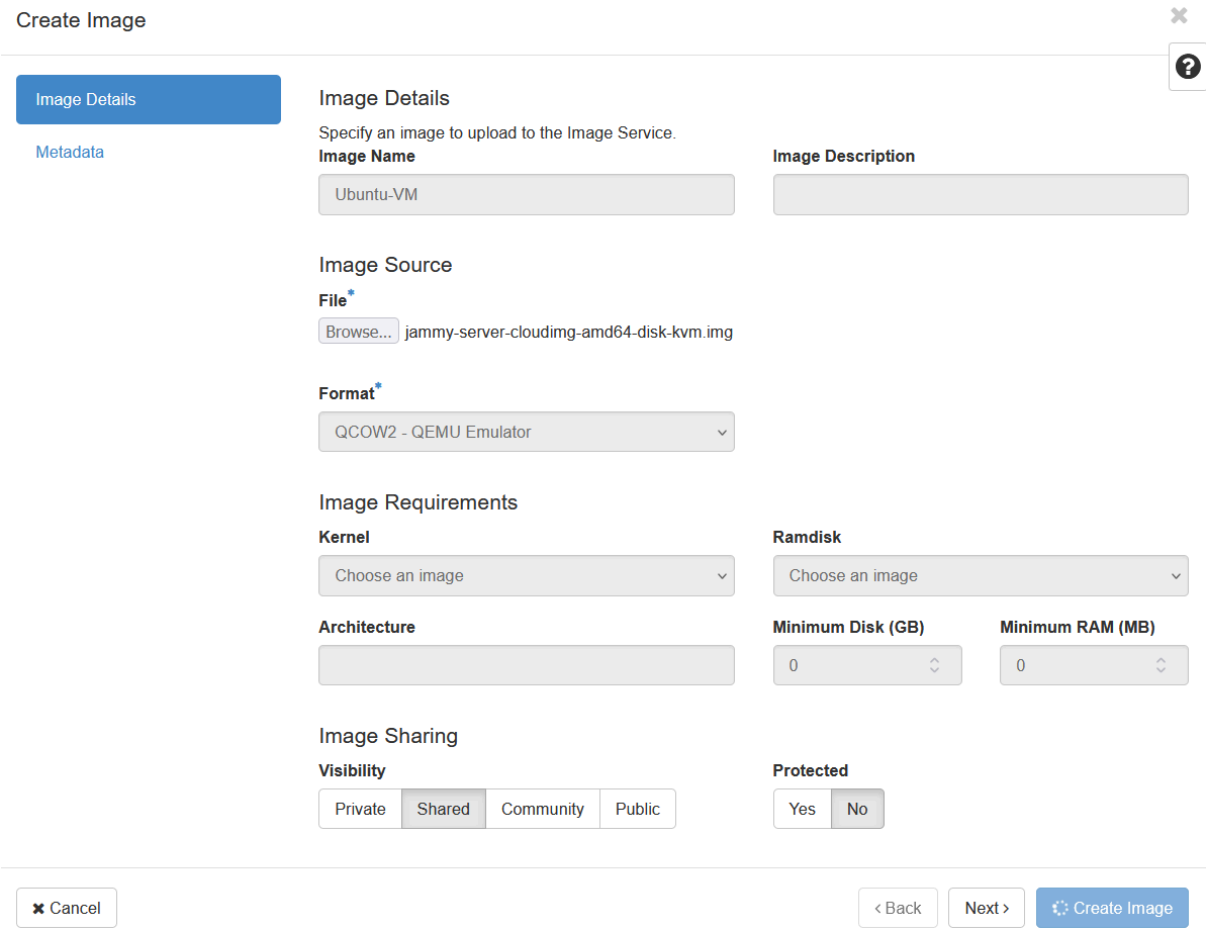


Phase III : Terraform et Ansible

1. Installation Terraform et Ansible

```
samirtaous@samirtaous:~/devstack$ # Installer Terraform
sudo apt-get update && sudo apt-get install -y gnupg software-properties-common
wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor | sudo tee /usr/share/keyrings/has
hicorp-archive-keyring.gpg > /dev/null
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.
com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list
sudo apt-get update && sudo apt-get install terraform -y

# Installer Ansible
sudo apt-get install ansible -y
Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:2 http://ma.archive.ubuntu.com/ubuntu noble InRelease
Hit:3 http://ma.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://ma.archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... 23%
```

2. Création de l'image et instance Ubuntu



Instances

Instance ID = | Filter | Launch Instance | Delete Instances | More Actions

Displaying 3 items

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
Ubuntu-instance	-	192.168.233.134	m1.tiny.ubuntu	cle-projet-cloud	Active	nova	None	Running	1 minute	Create Snapshot
CentOS-SaaS-VM	CentOS-SaaS	192.168.233.92, 172.24.4.61	ds2G	cle-projet-cloud	Shutoff	nova	None	Shut Down	55 minutes	Start Instance
instance-iaas-cirros	cirros-0.6.2-x86_64-disk	192.168.233.235, 172.24.4.237	m1.tiny	cle-projet-cloud	Shutoff	nova	None	Shut Down	1 hour, 16 minutes	Start Instance

Displaying 3 items

3. Configuration Terraform

```
# 1. Définition du fournisseur OpenStack
terraform {
  required_providers {
    openstack = {
      source  = "terraform-provider-openstack/openstack"
      version = "~> 1.50.0"
    }
  }
}

provider "openstack" {
  # Les identifiants sont récupérés automatiquement depuis votre fichier openrc
}

# 2. Création de l'instance Ubuntu
resource "openstack_compute_instance_v2" "vm_nginx" {
  name            = "ubuntu-nginx-terraform"
  image_name      = "ubuntu-jammy"
  flavor_name     = "m1.tiny.ubuntu"
  key_pair        = "cle-projet-cloud"
  security_groups = ["default"]

  network {
    name = "private" # Nom du réseau interne de votre DevStack
  }
}

# 3. Création d'une IP flottante
resource "openstack_networking_floatingip_v2" "fip_ubuntu" {
  pool = "public" # Pool d'IPs externes de DevStack
}

# 4. Association de l'IP flottante à la VM
resource "openstack_compute_floatingip_associate_v2" "fip_assoc" {
  floating_ip = openstack_networking_floatingip_v2.fip_ubuntu.address
  instance_id = openstack_compute_instance_v2.vm_nginx.id
}

# 5. Affichage de l'IP à la fin (utile pour Ansible)
output "ip_de_la_vm" {
  value = openstack_networking_floatingip_v2.fip_ubuntu.address
}
```

4. Execution de Terraform

4.1. Initialisation

```
samirtaous@samirtaous:~/terraform-deploy$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding terraform-provider-openstack/openstack versions matching "~> 1.50.0"...
- Installing terraform-provider-openstack/openstack v1.50.0...
- Installed terraform-provider-openstack/openstack v1.50.0 (self-signed, key ID 4F80527A391BEFD2)
Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://developer.hashicorp.com/terraform/cli/plugins/signing
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
samirtaous@samirtaous:~/terraform-deploy$
```

4.2. Vérification

```
samirtaous@samirtaous:~/terraform-deploy$ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are
indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # openstack_compute_floatingip_associate_v2.fip_assoc will be created
  + resource "openstack_compute_floatingip_associate_v2" "fip_assoc" {
      + floating_ip = (known after apply)
      + id          = (known after apply)
      + instance_id = (known after apply)
      + region      = (known after apply)
    }

  # openstack_compute_instance_v2.vm_nginx will be created
  + resource "openstack_compute_instance_v2" "vm_nginx" {
      + access_ip_v4        = (known after apply)
      + access_ip_v6        = (known after apply)
      + all_metadata        = (known after apply)
      + all_tags            = (known after apply)
      + availability_zone   = (known after apply)
      + created             = (known after apply)
      + flavor_id           = (known after apply)
      + flavor_name         = "m1.tiny.ubuntu"
      + force_delete        = false
      + id                  = (known after apply)
      + image_id            = (known after apply)
      + image_name          = "ubuntu-jammy"
      + key_pair            = "cle-projet-cloud"
      + name                = "ubuntu-nginx-terraform"
      + power_state         = "active"
      + region              = (known after apply)
      + security_groups     = [
          + "default",
        ]
      + stop_before_destroy = false
      + updated             = (known after apply)

      + network {
          + access_network = false
          + fixed_ip_v4    = (known after apply)
          + fixed_ip_v6    = (known after apply)
          + floating_ip    = (known after apply)
          + mac            = (known after apply)
          + name           = "private"
          + port           = (known after apply)
          + uuid           = (known after apply)
        }
    }

  # openstack_networking_floatingip_v2.fip_ubuntu will be created
  + resource "openstack_networking_floatingip_v2" "fip_ubuntu" {
      + address    = (known after apply)
      + all_tags   = (known after apply)
      + dns_domain = (known after apply)
      + dns_name   = (known after apply)
      + fixed_ip   = (known after apply)
      + id         = (known after apply)
      + pool       = "public"
```

4.3. Execution

```
samirtaous@samirtaous:~/terraform-deploy$ terraform apply -auto-approve

Terraform used the selected providers to generate the following execution plan. Resource actions are
indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # openstack_compute_floatingip_associate_v2.fip_assoc will be created
  + resource "openstack_compute_floatingip_associate_v2" "fip_assoc" {
      + floating_ip = (known after apply)
      + id          = (known after apply)
      + instance_id = (known after apply)
      + region      = (known after apply)
    }

  # openstack_compute_instance_v2.vm_nginx will be created
  + resource "openstack_compute_instance_v2" "vm_nginx" {
      + access_ip_v4        = (known after apply)
      + access_ip_v6        = (known after apply)
      + all_metadata        = (known after apply)
      + all_tags            = (known after apply)
      + availability_zone   = (known after apply)
      + created             = (known after apply)
      + flavor_id           = (known after apply)
      + flavor_name         = "m1.tiny.ubuntu"
      + force_delete        = false
      + id                  = (known after apply)
      + image_id            = (known after apply)
```

Résultat final de l'exécution

```
Changes to Outputs:
  + ip_de_la_vm = "172.24.4.15"
openstack_compute_instance_v2.vm_nginx: Creating...
openstack_compute_instance_v2.vm_nginx: Still creating... [00m10s elapsed]
openstack_compute_instance_v2.vm_nginx: Still creating... [00m20s elapsed]
openstack_compute_instance_v2.vm_nginx: Still creating... [00m30s elapsed]
openstack_compute_instance_v2.vm_nginx: Still creating... [00m40s elapsed]
openstack_compute_instance_v2.vm_nginx: Still creating... [00m50s elapsed]
openstack_compute_instance_v2.vm_nginx: Creation complete after 53s [id=d7b1a33f-738d-4886-b679-f875313
078be]
openstack_compute_floatingip_associate_v2.fip_assoc: Creating...
openstack_compute_floatingip_associate_v2.fip_assoc: Creation complete after 3s [id=172.24.4.15/d7b1a33
f-738d-4886-b679-f875313078be/]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

ip_de_la_vm = "172.24.4.15"
samirtaous@samirtaous:~/terraform-deploy$
```

5. Déploiement avec Ansible

On mets à jour le hosts.ini avec la nouvelle ip et on teste la configuration

```
 GNU nano 7.2                                  hosts.ini
[webserver]
172.24.4.15

[webserver:vars]
ansible_user=ubuntu
ansible_ssh_private_key_file=~/.ssh/id_rsa
ansible_ssh_common_args='-o StrictHostKeyChecking=no'
```

Phase IV : SLA

1. Fichier sla.txt

```
============================================================
      SERVICE LEVEL AGREEMENT (SLA) - PROJET UAE
============================================================
Client : Université Abdelmalek Essaadi (Département Informatique)
Fournisseur : Infrastructure OpenStack (Samir)

OBJET DU SLA :
Surveillance de la disponibilité des instances Terraform/OpenStack.

OBJECTIF DE PERFORMANCE :
- Taux de disponibilité cible : 99.5%
- Fréquence de contrôle : Toutes les 5 minutes
- Période d'évaluation : Quotidienne

PÉNALITÉS :
En cas de disponibilité < 99.5%, une maintenance corrective
prioritaire doit être déclenchée sur les services Nova/Neutron.

------------------------------------------------------------
LOGS DE SURVEILLANCE QUOTIDIENNE :
------------------------------------------------------------
```

2. Fichier monitor_sla.py

```
 GNU nano 7.2                                  monitor_sla.py *
import openstack
import time
from datetime import datetime

# Connexion à OpenStack (utilise vos identifiants source ~/devstack/openrc)
conn = openstack.connect(cloud='openstack')

def calculate_uptime():
    """
    Vérifie l'état des instances créées via Terraform.
    Utilise le module Nova (compute) pour obtenir le statut.
    """
    try:
        # Récupère toutes les instances du projet
        servers = list(conn.compute.servers())
        if not servers:
            return 100.0, 0

        total = len(servers)
        # On compte comme 'disponible' uniquement les serveurs au statut 'ACTIVE'
        active = sum(1 for s in servers if s.status == 'ACTIVE')

        availability = (active / total) * 100
        return availability, total
    except Exception as e:
        print(f"Erreur lors de la lecture des données Nova : {e}")
        return 0.0, 0

def report_to_sla(availability, count):
    """Écrit le rapport de surveillance dans le fichier sla.txt"""
    now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    threshold = 99.5
    status = "OK" if availability >= threshold else "ALERTE : HORS SLA"

    log_entry = f"[{now}] Instances: {count} | Disponibilité: {availability:.2f}% | Statut: {status}\n"

    with open("sla.txt", "a") as f:
        f.write(log_entry)
    print(log_entry.strip())

# Boucle de surveillance infinie
if __name__ == "__main__":
    print("--- Surveillance SLA OpenStack activée (Ctrl+C pour arrêter) ---")
    try:
        while True:
            avail, nb_vm = calculate_uptime()
            report_to_sla(avail, nb_vm)

            # Attente de 5 minutes (300 secondes)
            time.sleep(300)
    except KeyboardInterrupt:
        print("\nSurveillance arrêtée.")
```

3. Test

Sans aucune instance activé la disponibilité est à 0%

```
(venv) samirtaous@samirtaous:~/openstack-monitoring$ python monitor_sla.py
--- Surveillance SLA OpenStack activée (Ctrl+C pour arrêter) ---
[2026-01-13 10:06:07] Instances: 3 | Disponibilité: 0.00% | Statut: ALERTE : HORS SLA
^X^C
Surveillance arrêtée.
(venv) samirtaous@samirtaous:~/openstack-monitoring$
```

On active CentOS et on relance le script python

```
(venv) samirtaous@samirtaous:~/openstack-monitoring$ python monitor_sla.py
--- Surveillance SLA OpenStack activée (Ctrl+C pour arrêter) ---
[2026-01-13 10:07:32] Instances: 3 | Disponibilité: 33.33% | Statut: ALERTE : HORS SLA
```

Finalement le fichier sla.txt enregistre tous les résultats de surveillance

```
(venv) samirtaous@samirtaous:~/openstack-monitoring$ cat sla.txt
============================================================
      SERVICE LEVEL AGREEMENT (SLA) - PROJET UAE
============================================================
Client : Université Abdelmalek Essaadi (Département Informatique)
Fournisseur : Infrastructure OpenStack (Samir)

OBJET DU SLA :
Surveillance de la disponibilité des instances Terraform/OpenStack.

OBJECTIF DE PERFORMANCE :
- Taux de disponibilité cible : 99.5%
- Fréquence de contrôle : Toutes les 5 minutes
- Période d'évaluation : Quotidienne

PÉNALITÉS :
En cas de disponibilité < 99.5%, une maintenance corrective
prioritaire doit être déclenchée sur les services Nova/Neutron.

------------------------------------------------------------
LOGS DE SURVEILLANCE QUOTIDIENNE :
------------------------------------------------------------
[2026-01-13 10:06:07] Instances: 3 | Disponibilité: 0.00% | Statut: ALERTE : HORS SLA
[2026-01-13 10:07:32] Instances: 3 | Disponibilité: 33.33% | Statut: ALERTE : HORS SLA
(venv) samirtaous@samirtaous:~/openstack-monitoring$
```