TP 4 : Supervision avec Nagios

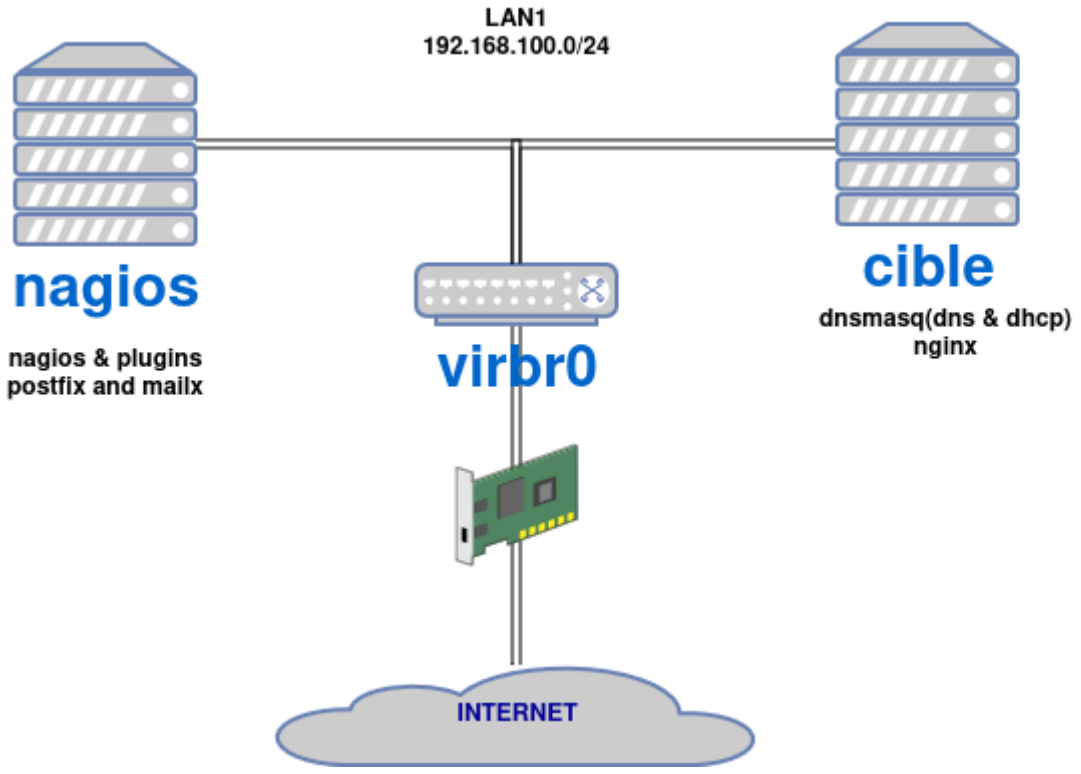
Objectif

Mettre en place une solution de supervision réseau basique avec Nagios pour surveiller les services critiques du réseau, notamment DNS, DHCP, ainsi que les hôtes dans le réseau. Déploiement de base avec l'approche **IaC** (Infrastructure as Code)

Outils

- **Nagios** : Outil de supervision réseau.
- **Libvirt, KVM, cloud-init**, **OpenTofu**: Outils de virtualisation.
- **Linux (Alpine)** : Système d'exploitation pour installer et configurer Nagios et la machine cible.
- **Docs**: https://opentofu.org/docs/language/, https://blog.stephane-robert.info/docs/virtualiser/type1/kvm/

Plan du TP



0. Configuration de base

```
.
├── cible.cfg                   # Contient la config de base cloud-init de la
machine cible
├── cible.tf                    # définitions des ressources(tofu) pour la vm
cible
├── install_tofu.sh             # installation de opentofu
├── main.tf                     # fichier principale des ressources(tofu) de
l'architecture réseau
├── nagios.cfg                  # Contient la config de base cloud-init de la VM
Nagios
├── nagios.tf                   # définitions des ressources(tofu) pour la vm
nagios
├── qemu.conf                   # mettre à jour votre conf /etc/libvirt/qemu.conf
avec les info de ce fichier
├── run.sh                      # déployer l'architecture réseau
```

Voici le fichier main.tf Terraform commenté comme vu lors de la séance de cours TP:

```
# Définition des versions requises pour les providers dans Terraform
# Ici, nous utilisons le provider "libvirt" pour interagir avec KVM/libvirt.
terraform {
  required_providers {
    libvirt = {
      # Le provider "libvirt" est fourni par "dmacvicar".
      source = "dmacvicar/libvirt"
      # Nous utilisons la version 0.8.0 du provider "libvirt".
      version = "0.8.0"
    }
  }
}

# Définition du provider libvirt
# Le provider permet de se connecter à l'hyperviseur local via libvirt.
provider "libvirt" {
  # "qemu:///system" indique que nous nous connectons à l'instance système de
libvirt sur l'hôte local.
  uri = "qemu:///system"
}

# Création d'un réseau virtuel libvirt
# Ce réseau sera utilisé pour connecter des machines virtuelles.
resource "libvirt_network" "lan1" {
  # Nom du réseau libvirt (LAN interne)
  name      = "lan1"

  # Plage d'adresses IP assignée à ce réseau
  # Le réseau utilisera les adresses IP dans la plage 192.168.101.0/24.
  addresses = ["192.168.101.0/24"]
}

# Définition d'un pool de stockage libvirt
```

```
# Un pool de stockage est un espace sur le disque hôte où seront stockés les
volumes des machines virtuelles.
resource "libvirt_pool" "new_pool" {
  # Nom du pool de stockage
  name = "new_pool"

  # Type de pool de stockage : ici, un répertoire (dir).
  type = "dir"

  # Chemin sur l'hôte où le pool sera créé
  path = "/var/lib/libvirt/new_pool"
}

# Définition d'un volume libvirt
# Ce volume représente une image disque (volume qcow2) que nous allons
télécharger depuis Alpine Linux.
resource "libvirt_volume" "alpine_cloud" {
  # Nom du volume dans libvirt (le nom du fichier image disque).
  name = "alpine.img"

  # Source de l'image : ici, nous téléchargeons une image cloud Alpine depuis
l'URL officielle.
  # Cette image est au format "qcow2" et est compatible avec cloud-init.
  source = "https://dl-
cdn.alpinelinux.org/alpine/v3.20/releases/cloud/nocloud_alpine-3.20.3-x86_64-
bios-cloudinit-r0.qcow2"
  #source = "alpine.qcow2"  # Vous pouvez utiliser une image locale si elle est
déjà téléchargée.

  # Format de l'image (qcow2 est un format de disque virtuel souvent utilisé avec
KVM/libvirt).
  format = "qcow2"

  # Pool de stockage où le volume sera créé.
  pool   = libvirt_pool.new_pool.name
}
```

Explications des concepts :

- **Provider** : Un provider dans Terraform permet de gérer des ressources spécifiques. Ici, `libvirt` est le provider qui permet d'interagir avec des machines virtuelles, des réseaux et des volumes via KVM/libvirt.
- **Resource** : Une ressource est un objet que Terraform va gérer. Par exemple, ici, nous créons un réseau virtuel, un pool de stockage, et un volume (image disque).
- **libvirt_network** : Cette ressource crée un réseau virtuel interne pour que les machines puissent se connecter entre elles.
- **libvirt_pool** : Le pool de stockage est un répertoire où les images et volumes des machines seront stockés.
- **libvirt_volume** : Le volume est une image disque, dans ce cas une image Alpine Linux au format qcow2. Il peut être utilisé comme disque pour les machines virtuelles.

Chaque ressource crée une partie de l'infrastructure qui sera utilisée pour héberger des machines virtuelles dans un environnement KVM/libvirt.

Explication détaillée du fichier nagio.tf Terraform, ligne par ligne comme vu au cours TP, pour vous aider à comprendre comment configurer une machine virtuelle avec Nagios en utilisant Terraform et libvirt :

```
# Création d'un volume basé sur l'image Alpine Cloud pour Nagios
resource "libvirt_volume" "nagios_volume" {
  # Nom du volume (fichier image disque) qui sera utilisé pour la machine
virtuelle Nagios
  name = "nagios.img"

  # Ce volume sera basé sur l'image Alpine Cloud que nous avons précédemment
téléchargée.
  base_volume_id = libvirt_volume.alpine_cloud.id

  # Le format du volume est "qcow2", un format de disque virtuel utilisé par
KVM/libvirt.
  format = "qcow2"

  # Le volume sera stocké dans le pool de stockage "new_pool".
  pool = libvirt_pool.new_pool.name
}

# Utilisation d'un fichier de configuration Cloud-init pour la machine Nagios
data "template_file" "user_data_nagios" {
  # Chargement du fichier "nagios.cfg", qui contient la configuration Cloud-init
personnalisée pour la machine Nagios.
  template = file("${path.module}/nagios.cfg")

  # Variable définie pour être utilisée dans le fichier Cloud-init.
  vars = {
    vm_name = "nagios"  # Nom de la machine virtuelle Nagios
  }
}

# Génération de la configuration cloud-init pour la machine Nagios
data "template_cloudinit_config" "config_nagios" {
  # Compression désactivée pour le fichier cloud-init
  gzip = false

  # Encodage Base64 désactivé pour le fichier cloud-init
  base64_encode = false

  # Ajout d'une partie au fichier cloud-init contenant la configuration
utilisateur
  part {
    filename = "init.cfg"  # Nom du fichier Cloud-init généré
    content_type = "text/cloud-config"

    # Le contenu du fichier Cloud-init est généré à partir du fichier de template
"nagios.cfg"
    content = "${data.template_file.user_data_nagios.rendered}"
  }
}

# Création d'un disque cloud-init pour initialiser la machine Nagios
```

```
resource "libvirt_cloudinit_disk" "nagios" {
  # Nom du fichier ISO cloud-init qui sera utilisé pour initialiser la VM
  name = "nagios.iso"

  # Le disque cloud-init sera stocké dans le pool "new_pool"
  pool = libvirt_pool.new_pool.name

  # Contenu du fichier cloud-init généré à partir du template "config_nagios"
  user_data = data.template_cloudinit_config.config_nagios.rendered

  # Optionnel : fichier de configuration réseau à ajouter (désactivé ici)
  # network_config = data.template_file.network_config.rendered
}

# Création de la machine virtuelle Nagios (VM1)
resource "libvirt_domain" "nagios" {
  # Nom de la machine virtuelle (VM)
  name = "nagios"

  # Configuration des ressources : 512 Mo de RAM
  memory = "512"

  # 1 CPU virtuel alloué à la machine virtuelle
  vcpu = 1

  # Utilisation du disque cloud-init pour configurer la machine lors de son
démarrage
  cloudinit = libvirt_cloudinit_disk.nagios.id

  # Configuration de l'interface réseau
  network_interface {
    # Connexion de l'interface réseau au réseau virtuel "lan1"
    network_id = libvirt_network.lan1.id

    # Définition du nom d'hôte pour la machine virtuelle
    hostname = "nagios"
  }

  # Configuration du disque dur de la VM, utilisant le volume créé précédemment
  disk {
    volume_id = libvirt_volume.nagios_volume.id
  }

  # Configuration de la console série pour interagir avec la machine
  console {
    type        = "pty"
    target_port = "0"
    target_type = "serial"
  }

  # Configuration d'un affichage graphique (ici, Spice est utilisé)
  graphics {
    type = "spice"
  }
}
```

Explication des concepts :

1. **libvirt_volume** : Ce bloc crée un volume pour la machine virtuelle Nagios. Le volume est basé sur l'image `alpine_cloud` que nous avons définie précédemment. Il sera stocké dans le pool de stockage `new_pool` avec un format `qcow2`, permettant d'économiser de l'espace disque grâce à sa gestion des snapshots et du stockage dynamique.
2. **data "template_file"** : Utilisé pour charger un fichier de configuration Cloud-init (`nagios.cfg`), qui peut être personnalisé pour la machine Nagios. Ce fichier contient les commandes et les configurations qui seront appliquées lors du démarrage de la machine virtuelle.
3. **data "template_cloudinit_config"** : Génère la configuration Cloud-init à partir du fichier de template. Cette configuration est utilisée pour initialiser la machine virtuelle (par exemple, définir des utilisateurs, installer des packages, configurer les réseaux, etc.).
4. **libvirt_cloudinit_disk** : Crée un disque ISO contenant les données Cloud-init (générées par Terraform) qui sera monté sur la machine virtuelle pour automatiser sa configuration au démarrage.
5. **libvirt_domain** : Ce bloc crée une machine virtuelle (`domain` dans la terminologie libvirt). Elle est configurée avec 512 Mo de RAM et un CPU virtuel, connectée au réseau `lan1`, et initialisée à l'aide de Cloud-init. Elle utilise l'image disque (`nagios.img`) définie plus haut, ainsi qu'une console série et un affichage graphique via Spice.

> La même chose pour le fichier cible.tf

1. Installation de OpenTofu et déploiement de l'architecture réseau

Rappels:

Voici les commandes de base pour **OpenTofu**, une alternative open-source à Terraform, permettant la gestion de l'infrastructure en tant que code. Si vous êtes déjà familier avec Terraform, vous verrez que les commandes d'OpenTofu sont très similaires, car il est basé sur le même concept.

1. Initialiser le projet OpenTofu

Cette commande initialise le répertoire de travail et télécharge les plugins requis pour les providers (comme libvirt, AWS, etc.).

```
tofu init
```

2. Planifier les modifications

Cette commande génère un plan d'exécution pour montrer ce que Tofu ferait sans réellement apporter de modifications à l'infrastructure. Elle est utile pour voir les changements avant de les appliquer.

```
opentofu plan
```

3. Appliquer les modifications

Applique les modifications à l'infrastructure, créant, modifiant ou détruisant les ressources selon les définitions du fichier de configuration.

```
tofu apply
```

4. Détruire les ressources

Cette commande détruit toutes les ressources gérées par OpenTofu dans le projet. Elle est utilisée pour nettoyer les ressources après utilisation.

```
tofu destroy
```

5. Afficher l'état des ressources

Pour voir l'état actuel de l'infrastructure et des ressources gérées par OpenTofu.

```
tofu show
```

6. Vérifier l'état de l'infrastructure

OpenTofu garde une trace de l'état de l'infrastructure dans un fichier `tfstate`. Cette commande affiche l'état enregistré des ressources.

```
tofu state list
```

7. Valider les fichiers de configuration

Cette commande permet de vérifier que les fichiers de configuration Tofu (fichiers `.tf`) sont bien écrits et valides.

```
tofu validate
```

8. Formatage des fichiers

Pour formater automatiquement les fichiers de configuration (par exemple, `.tf`) selon les standards de style d'OpenTofu.

```
tofu fmt
```

9. Afficher des informations détaillées sur une ressource

Pour inspecter une ressource spécifique dans l'état actuel.

```
tofu state show <resource_name>
```

10. Importer une ressource existante dans l'état

Cette commande permet d'importer une ressource existante dans OpenTofu afin qu'elle soit gérée par l'outil.

```
tofu import <resource_name> <resource_id>
```

11. Générer une représentation graphique du plan

Pour générer un graphique de dépendances sous forme de fichier `.dot` que vous pouvez convertir en une image (il suffira d'installer graphviz).

```
tofu graph | dot -Tpng -o archi.png
```

12. Verrouiller ou déverrouiller l'état

Lorsque plusieurs personnes travaillent sur la même infrastructure, il peut être utile de verrouiller l'état pour éviter des conflits. Ces commandes permettent de verrouiller et de déverrouiller l'état.

- Verrouiller :

  ```
  tofu force-unlock <lock-id>
  ```

- Déverrouiller :

  ```
  tofu force-unlock <lock-id>
  ```

Notes

- **`opentofu`** remplace simplement **`terraform`** dans les commandes. Si tu es déjà familier avec Terraform, la transition vers OpenTofu sera très simple, car les commandes et les concepts sont presque identiques.

Ces commandes de base te permettent de gérer et automatiser efficacement votre infrastructure avec OpenTofu.

1.1 **Pré-requis : Installation de OpenTofu**

- Lancer le script :

  ```
  ./install_tofu.sh
  ```

- Vérifier que opentofu est installé :

  ```
  ~/tps/TP4-tofu$ tofu --version
  OpenTofu v1.8.2
  on linux_amd64
  + provider registry.opentofu.org/dmacvicar/libvirt v0.8.0
  + provider registry.opentofu.org/hashicorp/template v2.2.0
  ```

- Le fichier `./install_tofu.sh` permet d'installer opentofu dans votre système

1.2 **Pré-requis : configuration de qemu**

- Lancer le fichier /etc/libvirt/qemu.conf, ajouter les ligne suivantes :

```
security_driver = "none"
user = "libvird-qemu"
group = "kvm"
```

- Cette action permet de donner les droits à libvirt dans le repertoire: /var/lib/libvirt/
- Redémarrer libvirtd

```
sudo systemctl restart libvirtd
```

1.3 Déploiement de l'architecture réseau (exécuter le script run.sh) :

- Lancer le script de déploiement:

```
$ ./run.sh
```

```
    - Vérifier le déploiement:
```

```
$ virsh list
 Id   Name     State
------------------------
 36   nagios   running
 37   cible    running
```

```
$ virsh net-list
 Name      State    Autostart   Persistent
-------------------------------------------
 default   active   yes         yes
 lan1      active   no          yes
```

```
$ virsh pool-list
 Name       State    Autostart
--------------------------------
 default    active   yes
 new_pool   active   yes
```

1.4 **Accès à l'interface web**

- Vous pouvez maintenant accéder à l'interface web de Nagios via l'URL suivante : `http://<adresse_IP_VM>/nagios`. Utilisez les identifiants `nagiosadmin` pour le nom d'utilisateur et `nagiosadmin` pour le mot de passe.

2. Supervision des services DNS et DHCP

1. **Surveillance du service DNS**
   - Dans le fichier de configuration des hôtes et services de Nagios, ajouter un bloc pour l'hôte cible :

     ```
     define host {
         use                     linux-server
         host_name               cible
         alias                   cible
         address                 <address_ip_cible>
     }
     ```

     Attention : vous devez récupérer l'adresse IP de la machine cible. Pour cela, utilisez `virsh console cible` pour vous connecter à la VM cible.
   - Ajouter l'hôte cible au groupe `linux-servers` :

     ```
     define hostgroup {
         hostgroup_name          linux-servers
         alias                   linux-servers
         members                 localhost,cible
     }
     ```

   - Ajouter un bloc pour surveiller le service DNS dans `/etc/objects/localhost.cfg` :

     ```
     define service {
         use                     generic-service
         host_name               cible
         service_description     DNS Service
         check_command           check_dns!google.com!$HOSTADDRESS$
     }
     ```

   - Ajouter une commande `check_dns` dans `/etc/objects/commands.cfg` si elle n'existe pas :

     ```
     define command {
         command_name    check_dns
         command_line    $USER1$/check_dns -H $ARG1$
     }
     ```

   - Relancer le service Nagios pour appliquer la configuration :

     ```
     sudo systemctl restart nagios
     ```

   - Vérifier dans l'interface web de Nagios que le service DNS est au vert. /!\ Attention : il faut attendre environ 5 minutes pour voir la mise à jour. Ce temps peut être changé dans le fichier `template.cfg`, bloc `local-service`, variable `check_interval`.
2. **Surveillance du service DHCP**
   - Ajouter une vérification pour le service DHCP dans `/etc/objects/localhost.cfg` :

```
define service {
    use                     generic-service
    host_name               cible
    service_description     DHCP Service
    check_command           check_dhcp!$HOSTADDRESS$
}
```

- Ajouter une commande pour le service DHCP dans `/etc/objects/commands.cfg` si elle n'existe pas :

```
define command {
    command_name    check_dhcp
    command_line    $USER1$/check_dhcp -s $ARG1$ -i $ARG2$
}
```

- Relancer le service Nagios :

```
sudo systemctl restart nagios
```

- Vérifier dans l'interface web de Nagios que le service DHCP est au vert. /!\ Attention : il faut attendre environ 5 minutes pour voir la mise à jour.

3. Supervision des hôtes

1. **Ajout d'un nouveau service à surveiller (le serveur NGINX déployé sur la cible)**
   - Pour surveiller le service NGINX, ajouter un service dans `/etc/objects/localhost.cfg` :

```
define service {
    use                     generic-service
    host_name               cible
    service_description     CHECK NGINX
    check_command           check_http!$HOSTADDRESS$!5!10
}
```

2. **Redémarrer Nagios pour appliquer la nouvelle configuration** :

```
sudo systemctl restart nagios
```

Vérifier dans l'interface web de Nagios que le service NGINX répond et est au vert. /!\ Attention : il faut attendre environ 5 minutes pour voir la mise à jour.

4. Configuration d'alertes et tableau de bord

1. **Configurer les alertes par e-mail**
   - L'envoi d'e-mails est optionnel. Vous pouvez laisser cette partie. Pour les courageux, dans le fichier `nagios.cfg`, des commandes de configuration de Postfix sont présentes pour l'envoi des mails. Il faudra remplacer les paramètres par les vôtres.
   - Configurer la commande pour l'envoie de mails dans les fichier `/etc/objects/commands.cfg` , vous pouvez remplacer nagios@ares.fr par votre @gmail:

```
define command {
command_name    notify-host-by-email
command_line    /usr/bin/printf "%b" "***** Nagios *****\n\nNotification
Type: $NOTIFICATIONTYPE$\nHost: $HOSTNAME$\nState: $HOSTSTATE$\nAddress:
$HOSTADDRESS$\nInfo: $HOSTOUTPUT$\n\nDate/Time: $LONGDATETIME$\n" |
/usr/bin/mail  -r nagios@ares.fr -s "** $NOTIFICATIONTYPE$ Host Alert:
$HOSTNAME$ is $HOSTSTATE$ **" $CONTACTEMAIL$
}

define command {
command_name    notify-service-by-email
command_line    /usr/bin/printf "%b" "***** Nagios *****\n\nNotification
Type: $NOTIFICATIONTYPE$\n\nService: $SERVICEDESC$\nHost:
$HOSTALIAS$\nAddress: $HOSTADDRESS$\nState: $SERVICESTATE$\n\nDate/Time:
$LONGDATETIME$\n\nAdditional Info:\n\n$SERVICEOUTPUT$\n" |  /usr/bin/mail
-r nagios@ares.fr -s "** $NOTIFICATIONTYPE$ Service Alert:
$HOSTALIAS$/$SERVICEDESC$ is $SERVICESTATE$ **" $CONTACTEMAIL$
}
```

- Configurer l'envoi d'e-mails pour les alertes dans `/etc/objects/contacts.cfg` :

```
define contact {
    contact_name                    nagiosadmin
    use                             generic-contact
    alias                           Nagios Admin
    email                           admin@gmail.com
}
```

  Remplacez `admin@gmail.com` par l'adresse e-mail à laquelle vous souhaitez recevoir les alertes.

- La notification d'un hôte ou d'un service s'obtient en modifiant la période et en activant la notification sur l'hôte ou le service concerné.

```
define host {
    use                     linux-server
    host_name               cible
    alias                   cible
    address                 192.168.100.154
    notifications_enabled   1                # activer la notification
    max_check_attempts      10
    check_period            24x7
    notification_interval   2                # interval d'envoie
    notification_period     24x7
}
```

- **Redémarrer Nagios pour appliquer la nouvelle configuration** :

```
sudo systemctl restart nagios
```

Vérifier dans l'interface web de Nagios que les notificatio sont émises

```
Dans le panneau latéral gauche, cliquez sur Hosts, puis sélectionnez l'hôte
concerné et observez la ligne :
Last Notification:  10-10-2024 07:14:59 (notification ...)
```

2. **Tableau de bord**
    - Vous pouvez accéder au tableau de bord Nagios via le navigateur web à l'adresse `http://<IP_VM>/nagios` pour visualiser en temps réel l'état des hôtes et services supervisés ainsi que les alertes générées.

Suivi post-TP

vous devez fournir :

- La configuration des hôtes et services supervisés dans Nagios.
- Une capture d'écran du tableau de bord Nagios montrant l'état des services et des hôtes.
- Un exemple d'alerte reçue par e-mail (optionnelle).