Documentation d'Architecture Technique (DAT)

# Table des matières

# Table des matières

1. [Introduction](#introduction) 2. [Prérequis](#prérequis) 3. [Installation d'OpenStack (Nova)](#installation-dopenstack-nova) 4. [Configuration de Terraform pour OpenStack](#configuration-de-terraform-pour-openstack) 5. [Installation et configuration d'Authelia avec Docker Compose](#installation-et-configuration-dauthelia-avec-docker-compose) 6. [Installation et configuration de Nginx](#installation-et-configuration-de-nginx) 7. [Intégration d'Authelia avec Nginx](#intégration-dauthelia-avec-nginx) 8. [Validation et tests](#validation-et-tests) 9. [Dépannage](#dépannage) 10. [Références](#références)

# Introduction

Cette documentation technique détaille le processus d'installation et de configuration d'un environnement complet incluant:

* OpenStack avec le service Nova pour la gestion des instances de calcul
* Terraform pour l'Infrastructure as Code (IaC) avec OpenStack
* Authelia pour l'authentification et l'autorisation
* Nginx comme serveur web/proxy inverse

L'objectif est de fournir un guide étape par étape pour configurer un environnement sécurisé et robuste.

# Prérequis

## Matériel recommandé

* CPU: Minimum 4 cœurs (8+ recommandés)
* RAM: Minimum 16 Go (32+ Go recommandés)
* Stockage: Minimum 100 Go d'espace libre

## Logiciels requis

* Système d'exploitation: Ubuntu 22.04 LTS (recommandé)
* Docker et Docker Compose
* Git
* Python 3.8+

## Réseau

* Connexion Internet stable
* Ports à ouvrir:
* 80/443 (HTTP/HTTPS)
* 9091 (Authelia)
* 5000 (Keystone)
* 8774 (Nova)

# Installation d'OpenStack (Nova)

## Préparation du système

# Mise à jour du système  
sudo apt update && sudo apt upgrade -y  
  
# Installation des dépendances  
sudo apt install -y python3-pip python3-dev libffi-dev gcc libssl-dev git

## Installation de DevStack

DevStack est la méthode recommandée pour une installation rapide d'OpenStack en environnement de développement.

# Créer un utilisateur stack  
sudo useradd -s /bin/bash -d /opt/stack -m stack  
sudo chmod +x /opt/stack  
echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack  
  
# Passer à l'utilisateur stack  
sudo su - stack  
  
# Cloner le dépôt DevStack  
git clone https://opendev.org/openstack/devstack  
cd devstack  
  
# Créer le fichier de configuration local.conf  
cat > local.conf << EOF  
[[local|localrc]]  
ADMIN\_PASSWORD=password  
DATABASE\_PASSWORD=password  
RABBIT\_PASSWORD=password  
SERVICE\_PASSWORD=password  
  
# Enable Nova  
enable\_service n-api  
enable\_service n-crt  
enable\_service n-cpu  
enable\_service n-cond  
enable\_service n-sch  
enable\_service n-novnc  
enable\_service n-api-meta  
enable\_service placement-api  
enable\_service placement-client  
  
# Désactiver les services non nécessaires pour cette installation  
disable\_service horizon  
disable\_service swift  
disable\_service cinder  
  
HOST\_IP=127.0.0.1  
EOF  
  
# Lancer l'installation  
./stack.sh

L'installation prendra environ 30-45 minutes selon les performances du système.

## Vérification de l'installation

# Vérifier les services OpenStack  
source openrc admin admin  
openstack service list  
openstack compute service list

Vous devriez voir tous les services Nova listés et en état "up".

## Configuration pour la production

Pour un environnement de production, consultez la documentation officielle d'OpenStack pour l'installation manuelle des services: https://docs.openstack.org/nova/latest/install/

# Configuration de Terraform pour OpenStack

## Installation de Terraform

# Ajouter la clé GPG HashiCorp  
wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg  
  
# Ajouter le dépôt HashiCorp  
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  
  
# Mettre à jour et installer Terraform  
sudo apt update && sudo apt install terraform

## Configuration du provider OpenStack

Créez un répertoire pour votre projet Terraform:

mkdir -p ~/terraform-openstack  
cd ~/terraform-openstack

Créez un fichier `provider.tf`:

terraform {  
 required\_providers {  
 openstack = {  
 source = "terraform-provider-openstack/openstack"  
 version = "~> 1.51.1"  
 }  
 }  
}  
  
provider "openstack" {  
 # Les identifiants seront chargés via les variables d'environnement  
}

Créez un fichier `variables.tf`:

variable "image\_id" {  
 type = string  
 description = "ID de l'image à utiliser pour les instances"  
}  
  
variable "flavor\_id" {  
 type = string  
 description = "ID du type d'instance à utiliser"  
}  
  
variable "network\_id" {  
 type = string  
 description = "ID du réseau à utiliser"  
}  
  
variable "keypair\_name" {  
 type = string  
 description = "Nom de la paire de clés SSH à utiliser"  
}

Créez un fichier `main.tf` pour définir vos ressources:

resource "openstack\_compute\_instance\_v2" "basic" {  
 name = "instance-basic"  
 image\_id = var.image\_id  
 flavor\_id = var.flavor\_id  
 key\_pair = var.keypair\_name  
 security\_groups = ["default"]  
  
 network {  
 uuid = var.network\_id  
 }  
}  
  
output "instance\_ip" {  
 value = openstack\_compute\_instance\_v2.basic.access\_ip\_v4  
}

## Fichier d'environnement pour Terraform

Créez un fichier `openrc.sh` pour stocker les variables d'environnement:

#!/bin/bash  
export OS\_AUTH\_URL=http://localhost:5000/v3  
export OS\_PROJECT\_NAME="admin"  
export OS\_USER\_DOMAIN\_NAME="Default"  
export OS\_PROJECT\_DOMAIN\_ID="default"  
export OS\_USERNAME="admin"  
export OS\_PASSWORD="password"  
export OS\_REGION\_NAME="RegionOne"  
export OS\_INTERFACE=public  
export OS\_IDENTITY\_API\_VERSION=3  
export TF\_VAR\_image\_id="ID\_DE\_VOTRE\_IMAGE"  
export TF\_VAR\_flavor\_id="ID\_DE\_VOTRE\_FLAVOR"  
export TF\_VAR\_network\_id="ID\_DE\_VOTRE\_RESEAU"  
export TF\_VAR\_keypair\_name="NOM\_DE\_VOTRE\_KEYPAIR"

## Utilisation de Terraform avec OpenStack

# Charger les variables d'environnement  
source openrc.sh  
  
# Initialiser Terraform  
terraform init  
  
# Vérifier le plan d'exécution  
terraform plan  
  
# Appliquer la configuration  
terraform apply

# Installation et configuration d'Authelia avec Docker Compose

## Préparation de l'environnement

# Créer un répertoire pour Authelia  
mkdir -p ~/authelia  
cd ~/authelia  
  
# Créer les sous-répertoires nécessaires  
mkdir -p config

## Configuration Docker Compose

Créez un fichier `docker-compose.yml`:

version: '3.8'  
  
services:  
 authelia:  
 image: authelia/authelia:latest  
 container\_name: authelia  
 volumes:  
 - ./config:/config  
 ports:  
 - 9091:9091  
 environment:  
 - TZ=Europe/Paris  
 restart: unless-stopped  
 networks:  
 - proxy  
 labels:  
 - "com.centurylinklabs.watchtower.enable=true"  
  
 redis:  
 image: redis:alpine  
 container\_name: redis  
 volumes:  
 - redis\_data:/data  
 restart: unless-stopped  
 networks:  
 - proxy  
 command: redis-server --requirepass authelia\_redis\_password  
  
networks:  
 proxy:  
 external: true  
  
volumes:  
 redis\_data:

## Configuration d'Authelia

Créez un fichier `config/configuration.yml`:

---  
###############################################################################  
# Authelia Configuration #  
###############################################################################  
  
server:  
 host: 0.0.0.0  
 port: 9091  
  
log:  
 level: info  
   
jwt\_secret: a\_very\_secure\_jwt\_secret\_key\_change\_me  
   
default\_redirection\_url: https://example.com  
  
totp:  
 issuer: authelia.com  
 period: 30  
 skew: 1  
  
# Configuration de la base de données (fichier local pour la simplicité)  
storage:  
 local:  
 path: /config/db.sqlite3  
  
# Configuration Redis pour les sessions  
session:  
 name: authelia\_session  
 domain: example.com  
 secret: a\_very\_secure\_session\_secret\_key\_change\_me  
 expiration: 3600 # 1 heure  
 inactivity: 300 # 5 minutes  
 redis:  
 host: redis  
 port: 6379  
 password: authelia\_redis\_password  
  
# Configuration d'authentification  
authentication\_backend:  
 file:  
 path: /config/users\_database.yml  
 password:  
 algorithm: argon2id  
 iterations: 1  
 key\_length: 32  
 salt\_length: 16  
 memory: 1024  
 parallelism: 8  
  
# Configuration d'accès  
access\_control:  
 default\_policy: deny  
 rules:  
 - domain: public.example.com  
 policy: bypass  
 - domain: "\*.example.com"  
 policy: two\_factor  
  
# Configuration de notification (exemple avec fichier, à remplacer par SMTP en production)  
notifier:  
 filesystem:  
 filename: /config/notification.txt

Créez le fichier `config/users\_database.yml`:

---  
users:  
 admin:  
 displayname: "Admin User"  
 password: "$argon2id$v=19$m=65536,t=3,p=4$ZWlCbS9wTlB5aEovb29mSA$gpQlZvsP/uEoJ5/Mp3igdkUAfvKUxaOPEh9c3lPGR4Y" # mot de passe: admin  
 email: admin@example.com  
 groups:  
 - admins  
 user:  
 displayname: "Regular User"  
 password: "$argon2id$v=19$m=65536,t=3,p=4$cXpuTVpJbmxiWUlLcEtPZw$N/9QOuFzV9KyKLKe0pTt4uu9BHr0v8UwV+XlVyFu/Kg" # mot de passe: password  
 email: user@example.com  
 groups:  
 - users

## Démarrage d'Authelia

# Créer le réseau Docker s'il n'existe pas  
docker network create proxy  
  
# Démarrer les conteneurs  
docker-compose up -d

# Installation et configuration de Nginx

## Installation de Nginx

sudo apt update  
sudo apt install -y nginx

## Configuration de base de Nginx

# Sauvegarde de la configuration par défaut  
sudo cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf.bak  
  
# Créer un répertoire pour les certificats SSL  
sudo mkdir -p /etc/nginx/ssl

Générer des certificats auto-signés pour le développement (remplacer par Let's Encrypt en production):

sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \  
 -keyout /etc/nginx/ssl/nginx.key \  
 -out /etc/nginx/ssl/nginx.crt \  
 -subj "/C=FR/ST=Paris/L=Paris/O=Organization/CN=example.com"

Créez le fichier de configuration principal `/etc/nginx/nginx.conf`:

user www-data;  
worker\_processes auto;  
pid /run/nginx.pid;  
include /etc/nginx/modules-enabled/\*.conf;  
  
events {  
 worker\_connections 1024;  
 multi\_accept on;  
}  
  
http {  
 # Configuration de base  
 sendfile on;  
 tcp\_nopush on;  
 tcp\_nodelay on;  
 keepalive\_timeout 65;  
 types\_hash\_max\_size 2048;  
   
 # MIME types  
 include /etc/nginx/mime.types;  
 default\_type application/octet-stream;  
  
 # Logging  
 access\_log /var/log/nginx/access.log;  
 error\_log /var/log/nginx/error.log;  
  
 # Compression gzip  
 gzip on;  
 gzip\_vary on;  
 gzip\_proxied any;  
 gzip\_comp\_level 6;  
 gzip\_buffers 16 8k;  
 gzip\_http\_version 1.1;  
 gzip\_types text/plain text/css application/json application/javascript text/xml application/xml application/xml+rss text/javascript;  
  
 # Inclure les configurations de sites  
 include /etc/nginx/conf.d/\*.conf;  
 include /etc/nginx/sites-enabled/\*;  
}

# Intégration d'Authelia avec Nginx

## Configuration de Nginx pour l'authentification Authelia

Créez un fichier `/etc/nginx/snippets/authelia-location.conf`:

location /authelia {  
 internal;  
 set $upstream\_authelia http://127.0.0.1:9091/api/verify;  
 proxy\_pass $upstream\_authelia;  
 proxy\_pass\_request\_body off;  
 proxy\_set\_header Content-Length "";  
   
 # Paramètres à transmettre à Authelia  
 proxy\_set\_header X-Original-URL $scheme://$http\_host$request\_uri;  
 proxy\_set\_header X-Forwarded-Proto $scheme;  
 proxy\_set\_header X-Forwarded-Host $http\_host;  
 proxy\_set\_header X-Forwarded-Uri $request\_uri;  
 proxy\_set\_header X-Forwarded-For $remote\_addr;  
 proxy\_set\_header Remote-User $remote\_user;  
 proxy\_set\_header Remote-Name $remote\_user;  
   
 # Configurer le timeout (important pour la validation 2FA)  
 proxy\_connect\_timeout 5s;  
 proxy\_send\_timeout 5s;  
 proxy\_read\_timeout 5s;  
}

Créez un fichier `/etc/nginx/snippets/authelia-authrequest.conf`:

auth\_request /authelia;  
auth\_request\_set $target\_url $scheme://$http\_host$request\_uri;  
auth\_request\_set $user $upstream\_http\_remote\_user;  
auth\_request\_set $name $upstream\_http\_remote\_name;  
auth\_request\_set $groups $upstream\_http\_remote\_groups;  
auth\_request\_set $redirect $upstream\_http\_location;  
  
error\_page 401 =302 https://auth.example.com/?rd=$target\_url;

## Configuration d'un site web protégé par Authelia

Créez un fichier `/etc/nginx/sites-available/protected.example.com.conf`:

server {  
 listen 80;  
 server\_name protected.example.com;  
   
 # Redirection HTTP vers HTTPS  
 return 301 https://$host$request\_uri;  
}  
  
server {  
 listen 443 ssl http2;  
 server\_name protected.example.com;  
   
 # Certificats SSL  
 ssl\_certificate /etc/nginx/ssl/nginx.crt;  
 ssl\_certificate\_key /etc/nginx/ssl/nginx.key;  
 ssl\_protocols TLSv1.2 TLSv1.3;  
 ssl\_prefer\_server\_ciphers on;  
 ssl\_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384;  
   
 # Intégration Authelia  
 include /etc/nginx/snippets/authelia-authrequest.conf;  
   
 # Directives de sécurité  
 add\_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;  
 add\_header X-Content-Type-Options nosniff;  
 add\_header X-Frame-Options SAMEORIGIN;  
 add\_header X-XSS-Protection "1; mode=block";  
  
 location / {  
 # Intégration avec le service protégé (exemple: OpenStack Horizon)  
 proxy\_pass http://localhost:8080;  
 proxy\_set\_header Host $host;  
 proxy\_set\_header X-Real-IP $remote\_addr;  
 proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  
 proxy\_set\_header X-Forwarded-Proto $scheme;  
   
 # Variables transmises par Authelia  
 proxy\_set\_header Remote-User $user;  
 proxy\_set\_header Remote-Name $name;  
 proxy\_set\_header Remote-Groups $groups;  
 }  
   
 # Endpoint d'authentification Authelia  
 include /etc/nginx/snippets/authelia-location.conf;  
}

## Configuration du portail Authelia

Créez un fichier `/etc/nginx/sites-available/auth.example.com.conf`:

server {  
 listen 80;  
 server\_name auth.example.com;  
   
 # Redirection HTTP vers HTTPS  
 return 301 https://$host$request\_uri;  
}  
  
server {  
 listen 443 ssl http2;  
 server\_name auth.example.com;  
   
 # Certificats SSL  
 ssl\_certificate /etc/nginx/ssl/nginx.crt;  
 ssl\_certificate\_key /etc/nginx/ssl/nginx.key;  
 ssl\_protocols TLSv1.2 TLSv1.3;  
 ssl\_prefer\_server\_ciphers on;  
 ssl\_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384;  
   
 # Directives de sécurité  
 add\_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;  
 add\_header X-Content-Type-Options nosniff;  
 add\_header X-Frame-Options SAMEORIGIN;  
 add\_header X-XSS-Protection "1; mode=block";  
  
 # Proxy vers Authelia  
 location / {  
 proxy\_pass http://127.0.0.1:9091;  
 proxy\_set\_header Host $host;  
 proxy\_set\_header X-Real-IP $remote\_addr;  
 proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  
 proxy\_set\_header X-Forwarded-Proto $scheme;  
   
 # Pour WebSocket  
 proxy\_http\_version 1.1;  
 proxy\_set\_header Upgrade $http\_upgrade;  
 proxy\_set\_header Connection "upgrade";  
 }  
}

## Activation des configurations et redémarrage de Nginx

# Créer les liens symboliques pour activer les configurations  
sudo ln -s /etc/nginx/sites-available/protected.example.com.conf /etc/nginx/sites-enabled/  
sudo ln -s /etc/nginx/sites-available/auth.example.com.conf /etc/nginx/sites-enabled/  
  
# Vérifier la configuration  
sudo nginx -t  
  
# Redémarrer Nginx  
sudo systemctl restart nginx

# Validation et tests

## Test d'OpenStack

# Sourcer les variables d'environnement OpenStack  
source ~/devstack/openrc admin admin  
  
# Vérifier que les services fonctionnent  
openstack service list  
openstack compute service list  
  
# Créer une VM de test  
openstack server create --flavor m1.tiny --image cirros --network private test-vm

## Test de Terraform

# Sourcer les variables d'environnement  
source ~/terraform-openstack/openrc.sh  
  
# Exécuter Terraform  
cd ~/terraform-openstack  
terraform apply -auto-approve

## Test d'Authelia et Nginx

1. Ajoutez les entrées dans le fichier `/etc/hosts` pour les tests: ``` 127.0.0.1 auth.example.com 127.0.0.1 protected.example.com ```

2. Accédez à `https://protected.example.com` - Vous devriez être redirigé vers `https://auth.example.com` - Connectez-vous avec les identifiants configurés (admin/admin ou user/password) - Après authentification réussie, vous devriez accéder au service protégé

# Dépannage

## Problèmes avec OpenStack

1. Vérifiez les journaux: ```bash sudo journalctl -u devstack@\* ```

2. Si un service ne démarre pas: ```bash # Redémarrez le service spécifique (exemple avec Nova API) sudo systemctl restart devstack@n-api ```

## Problèmes avec Authelia

1. Vérifiez les logs Docker: ```bash docker logs authelia ```

2. Vérifiez la configuration: ```bash docker exec -it authelia authelia validate-config ```

## Problèmes avec Nginx

1. Vérifiez les logs Nginx: ```bash sudo tail -f /var/log/nginx/error.log ```

2. Testez la configuration: ```bash sudo nginx -t ```

# Références

* [Documentation officielle OpenStack](https://docs.openstack.org/)
* [Documentation officielle Terraform](https://registry.terraform.io/providers/terraform-provider-openstack/openstack/latest/docs)
* [Documentation officielle Authelia](https://www.authelia.com/docs/)
* [Documentation officielle Nginx](https://nginx.org/en/docs/)