Annexe A

Infrastructure Docker OpenZiti N1

Cette annexe est une marche à suivre pour reproduire l'installation de la première infrastructure OpenZiti déployée avec Docker (section 10.1). Le but de cette infrastructure n'est pas que toute la solution soit déployée automatiquement par Docker comme le tutoriel quickstart fournit par la documentation d'OpenZiti, mais plutôt une marche à suivre progressive afin de comprendre comment créer un réseau OpenZiti puis le processus pour y ajouter des machines.

Les différents fichiers sont disponibles à l'adresse suivante :

https://github.com/MehSalhi/TB-ZeroTrust-OpenZiti/tree/master/Network/N1b

A.1 Composants

- contrôleur OpenZiti
- initialisateur du contrôleur
- edge router OpenZiti
- console web OpenZiti
- machine qui stream un flux vidéo
- serveur vidéo qui reçoit le flux et le retransmet à des clients

A.2 Pré-requis

Les logiciels suivants doivent être installés :

- Docker
- Docker compose

A.3 Marche à suivre

A.3.1 Étapes

Les étapes suivantes sont nécessaires :

Environnement Création de l'environnement Docker

Contrôleur Lancement et configuration du contrôleur

Routeur Création de l'identité du routeur et inscription auprès du contrôleur

Console Web Connexion à la console web

Identités Création des identités du streamer et du serveur vidéos

Configurations Création de configuration d'interception du trafic pour le rediriger sur le réseau OpenZiti

Service Création d'un service "video"

Politiques Création de politiques pour permettre au streamer de se connecter au service et au serveur vidéo d'écoute ce service

Inscription Inscription du streamer et du serveur vidéo auprès du contrôleur

A.3.2 Environnement

Créer un fichier suivant (aussi disponible sur Github dans le répertoire 'N1/docker-compose.yaml') :

Code source A.3.1: docker-compose.yaml

```
# Auteur       : Mehdi Salhi
# Auteur       : OpenZiti documentation
# Sujet        : Travail de Bachelor Zero Trust OpenZiti
# But          : Déploie une infrastructure OpenZiti
# No           : n1
# Description : 1 contrôleur, 1 routeur, 1 initialisteur, 1 serveur vidéo, 1
#               streamer. Tous les fichiers sont sauvegardés dans le répertoire
#               local "vol", et monté dans le répertoire "persistent" sur les
#               containers
#
version: '2.4'
services:
  # openziti controller
  ziti-controller:
    image: "${ZITI_IMAGE}:${ZITI_VERSION}"
    env_file:
      - ./.env
    ports:
      - ${ZITI_EDGE_CONTROLLER_PORT:-1280}:${ZITI_EDGE_CONTROLLER_PORT:-1280}
```

```
20      - ${ZITI_CTRL_PORT:-6262}:${ZITI_CTRL_PORT:-6262}
21    environment:
22      -
    ↪ ZITI_EDGE_IDENTITY_ENROLLMENT_DURATION=${ZITI_EDGE_IDENTITY_ENROLLMENT_DURATION}
23      - ZITI_EDGE_ROUTER_ENROLLMENT_DURATION=${ZITI_EDGE_ROUTER_ENROLLMENT_DURATION}
24    volumes:
25      - type: bind
26        source: ./vol
27        target: /persistent
28    entrypoint:
29      - "/var/openziti/scripts/run-controller.sh"
30    networks:
31      A:
32        aliases:
33          - ziti-edge-controller
34      control_server:
35        aliases:
36          - ziti-edge-controller
37      control_stream:
38        aliases:
39          - ziti-edge-controller
40
41  # controller init
42  ziti-controller-init-container:
43    image: "${ZITI_IMAGE}:${ZITI_VERSION}"
44    depends_on:
45      - ziti-controller
46    environment:
47      - ZITI_CONTROLLER_RAWNAME="${ZITI_CONTROLLER_RAWNAME}"
48      - ZITI_EDGE_CONTROLLER_RAWNAME="${ZITI_EDGE_CONTROLLER_RAWNAME}"
49    env_file:
50      - ./.env
51    networks:
52      A:
53        aliases:
54          - ziti-edge-controller-init-container
55    volumes:
56      - type: bind
57        source: ./vol
58        target: /persistent
59    entrypoint:
60      - "/var/openziti/scripts/run-with-ziti-cli.sh"
61    command:
62      - "/var/openziti/scripts/access-control.sh"
63
64  # edge router
65  ziti-edge-router:
66    image: "${ZITI_IMAGE}:${ZITI_VERSION}"
67    hostname: ziti-edge-router
```

```
68      depends_on:
69        - ziti-controller
70      environment:
71        - ZITI_CONTROLLER_RAWNAME="${ZITI_CONTROLLER_RAWNAME}"
72        - ZITI_EDGE_CONTROLLER_RAWNAME="${ZITI_EDGE_CONTROLLER_RAWNAME}"
73        - ZITI_EDGE_ROUTER_RAWNAME=${ZITI_EDGE_ROUTER_RAWNAME:-ziti-edge-router}
74        - ZITI_EDGE_ROUTER_ROLES=public
75      ports:
76        - ${ZITI_EDGE_ROUTER_PORT:-3022}:${ZITI_EDGE_ROUTER_PORT:-3022}
77      networks:
78        A:
79          aliases:
80            - ziti-edge-router
81        B:
82          aliases:
83            - ziti-edge-router
84        C:
85          aliases:
86            - ziti-edge-router
87      volumes:
88        - type: bind
89          source: ./vol
90          target: /persistent
91      entrypoint: ["bash", "-c", sleep 5 && /var/openziti/ziti-bin/ziti-router run
    ↪ /persistent/ziti-edge-router.yaml]
92      #stdin_open: true
93      #tty: true
94
95      # console
96    ziti-console:
97      image: openziti/zac
98      environment:
99        - ZAC_SERVER_CERT_CHAIN=/persistent/pki/${ZITI_EDGE_CONTROLLER_HOSTNAME:-ziti-
    ↪ controller}-intermediate/certs/${ZITI_EDGE_CONTROLLER_HOSTNAME:-ziti-controller}
    ↪ -server.cert
100       -
    ↪ ZAC_SERVER_KEY=/persistent/pki/${ZITI_EDGE_CONTROLLER_HOSTNAME:-ziti-controller}
    ↪ -intermediate/keys/${ZITI_EDGE_CONTROLLER_HOSTNAME:-ziti-controller}-server.key
101       - PORTTLS=8443
102     ports:
103       - 1408:1408
104       - 8443:8443
105     working_dir: /usr/src/app
106     volumes:
107       - type: bind
108         source: ./vol
109         target: /persistent
110     networks:
111       - A
```

```
112
113     ##### machines
114     # video server
115   video-server:
116     image: mehdi/rtmp-hls_server
117     build:
118       dockerfile: ./rtmp-hls-server/Dockerfile
119       context: .
120       network: host
121     depends_on:
122       - ziti-edge-router
123     cap_add:
124       - NET_ADMIN
125     ports:
126       - 8080:8080
127     volumes:
128       - type: bind
129         source: ./vol
130         target: /persistent
131     healthcheck:
132       test: curl --fail http://localhost:8080 || exit 1
133     entrypoint:
134       - "/persistent/tunnel-server.sh"
135     networks:
136       B:
137         aliases:
138           - video-server
139       control_server:
140         aliases:
141           - video-server
142
143   # video streamer
144   video-streamer:
145     image: mehdi-linuxserver/ffmpeg
146     depends_on:
147       video-server:
148         condition: service_healthy
149     cap_add:
150       - NET_ADMIN
151     devices:
152       - /dev/net/tun:/dev/net/tun
153     build:
154       dockerfile: Dockerfile_streamer
155       context: .
156       network: host
157     volumes:
158       - type: bind
159         source: ./vol
160         target: /persistent
```

```
    entrypoint:
      - "/persistent/tunnel-streamers.sh"
    privileged: true
    stdin_open: true
    tty: true
    networks:
      C:
        aliases:
          - video-streamer
      control_stream:
        aliases:
          - video-streacontrol_streamermer

# networks
networks:
  A:
    driver: bridge
  B:
    driver: bridge
  C:
    driver: bridge
  control_server:
    driver: bridge
  control_stream:
    driver: bridge
```

Dans le même répertoire que le fichier docker-compose.yaml, mettre le fichier .env suivant :
(Fichier disponible ici : https://github.com/MehSalhi/TB-ZeroTrust-OpenZiti/blob/master/Network/N1b/.env

Code source A.3.2: .env

```
# OpenZiti Variables
ZITI_IMAGE=openziti/quickstart
ZITI_VERSION=latest

# The duration of the enrollment period (in minutes), default if not set
# shown - 7days
ZITI_EDGE_IDENTITY_ENROLLMENT_DURATION=10080
ZITI_EDGE_ROUTER_ENROLLMENT_DURATION=10080

# controller address/port information
ZITI_CONTROLLER_RAWNAME=ziti-controller
#ZITI_CONTROLLER_HOSTNAME=advertised.address
#ZITI_CTRL_PORT=8440

ZITI_EDGE_CONTROLLER_RAWNAME=ziti-edge-controller
#ZITI_EDGE_CONTROLLER_HOSTNAME=advertised.address
#ZITI_EDGE_CONTROLLER_PORT=8441
#ZITI_EDGE_CONTROLLER_IP_OVERRIDE=172.17.0.1
```

```
19
20 # router address/port information
21 #ZITI_EDGE_ROUTER_RAWNAME=advertised.address
22 #ZITI_EDGE_ROUTER_PORT=8442
23 #ZITI_EDGE_ROUTER_IP_OVERRIDE=172.17.0.1
```

- Commenter la ligne entrypoint de la section ziti-edge-router et décommenter les lignes "stdin_open : true" et "tty : true". Cela fera que le container du routeur ne se fermera pas automatiquement lorsque le script pour lancer le routeur échoue. Cela est normal car pour l'instant le routeur ne possède pas d'identité auprès du controller et ne peux donc pas s'authentifier. On pourra ainsi se connecter au container du routeur pour s'inscrire auprès du contrôleur.

- créer le fichier ziti.env dans "/vol" et s'assurer qu'il y ait les droits de lecture et écriture dessus. Il sera utilisé par le contrôleur et l'initialisateur pour y stocker des données d'environnement

- créer les scripts suivants dans "/vol" qui seront utilisés par le streamer et le serveur vidéo comme point d'entrée :

(Fichier disponible dans le répertoire 'N1b/vol')

Code source A.3.3: tunnel-server.sh

```
1 # Auteur      : Mehdi Salhi
2 # Sujet       : Travail de Bachelor Zero Trust OpenZiti
3 # But         : point d'entrée du serveur vidéo.
4 # No          : n1
5 # Description : Créer l'interface tun, lance nginx et le tunnel OpenZiti
6 #!/bin/bash
7
8 mkdir -p /dev/net
9 mknod /dev/net/tun c 10 200
10 chmod 600 /dev/net/tun
11
12 nginx
13 ziti-edge-tunnel run -i /persistent/video-server.json
```

(Fichier disponible dans le répertoire 'N1b/vol')

Code source A.3.4: tunnel-streamers.sh

```
1 # Auteur      : Mehdi Salhi
2 # Sujet       : Travail de Bachelor Zero Trust OpenZiti
3 # But         : point d'entrée du streamer.
4 # No          : n1
5 # Description : Créée l'interface tun, install une librairie, recrée le fichier
```

```
#               resolv.conf et lance le tunnel OpenZiti puis le flux vidéo

#!/bin/bash

apt install -y /persistent/libssl1.1_1.1.1f-1ubuntu2.17_amd64.deb

mkdir -p /dev/net
mknod /dev/net/tun c 10 200
chmod 600 /dev/net/tun

umount /etc/resolv.conf && echo 'nameserver 127.0.0.11
ptions ndots:0' | tee /etc/resolv.conf

ziti-edge-tunnel run -i /persistent/streamers.json &
sleep 20
ffmpeg -stream_loop -1 -re -i /video/nautilus.mp4 -f flv
  ↪  rtmp://video-server.ziti/live/test &
/bin/bash
```

Construire les différentes images Docker :

```
docker compose build
```

A.3.3 Contrôleur

Lancer l'infrastructure docker, et noter le mot de passe qui s'affiche vers le début du script, il servira à se connecter au contrôleur :

```
docker compose up
```

Le mot de passe est une chaîne de caractère sous la forme suivante, parmi les premières lignes :

```
n1b-ziti-controller-1 | Do you want to keep the generated admin
↪  password '9gAOVAOqVsZ4qOnlWyAhmosSJZim6JoJ'? (Y/n) INFO: using
↪  ZITI_PWD=9gAOVAOqVsZ4qOnlWyAhmosSJZim6JoJ
```

La suite du script initialise un environnement OpenZiti avec une PKI, une base de donnée et divers fichiers.

Une fois terminé, se connecter au contrôleur pour vérifier que tout soit correcte :

```
# ouvrir une console sur le container du contrôleur
docker exec -it n1b-ziti-controller-1 bash
# s'authentifier auprès du contrôleur. Username: admin et mot de passe
↪  noté
# précédemment

ziti edge login
```

A.3.4 Routeur

Depuis le contrôleur, créer la configuration et un token d'identification. Le fichier de configuration permettra aux routeur de s'initialiser et le token de s'authentifier auprès du contrôleur.

```
ziti create config router edge --routerName  ziti-edge-router \
                               --output ziti-edge-router.yaml
ziti edge create edge-router ziti-edge-router --jwt-output-file
ziti-edge-router.jwt --tunneler-enabled -a public
```

Se connecter au routeur et s'inscrire auprès du contrôleur :

```
docker exec -it n1b-ziti-edge-router-1 bash
ziti-router enroll ziti-edge-router.yaml --jwt ziti-edge-router.jwt

# le message de confirmation suivant doit être reçu
[   1.104]    INFO edge/router/enroll.(*RestEnroller).Enroll:
↪  registration complete
```

A ce stade, le fichier docker-compose.yml peut etre modifié pour enlever le tty : true et remettre l'entrypoint, puis etre redémmaré. Le routeur s'authentifiera automatiquement au contrôleur au lancement

Pour vérifier que le routeur soit bien online, depuis le contrôleur :

```
ziti@b3d903d37b29:/persistent$ ziti edge list edge-routers

-----------------------------------------------------------
| ID         | NAME             | ONLINE | ALLOW TRANSIT |
-----------------------------------------------------------
| FPfB6MTD4y | ziti-edge-router | true   | true          |
-----------------------------------------------------------

results: 1-1 of 1
```

A.3.5 Console web

Une console web est disponible dans l'infrastructure Docker. Elle permet d'administrer le réseau OpenZiti. Pour s'y connecter depuis un navigateur web :

Adresse : `http://localhost:1408`

Entrer les informations suivantes :

Controller name : controller

Address : `https://ziti-controller:1280`

FIGURE A.1 – Console ZAC setup contrôleur

FIGURE A.2 – Login console

A.3.6 Identité

Toutes les entitiés qui font partie d'un réseau OpenZiti doivent posséder une identité et s'incrire auprès du contrôleur. Il faut donc créer des identités et des jetons d'inscriptions depuis le contrôleur pour le streamer et le serveur vidéo. Les jetons sont aux format jwt et seront consommés lors de l'inscription auprès du contrôleur.

D

```
docker exec -it n1b-ziti-controller-1 bash
ziti edge login

# créer l'identité du serveur vidéo
ziti edge create identity user video-server.ziti -o
↪  video-server.ziti.jwt

# créer l'identité du streamer
ziti edge create identity user streamers -a 'streamers' -o
↪  streamers.jwt
```

A.3.7 Configurations

Les configuration servent à OpenZiti de savoir comment gérer le trafic. Il est possible d'intercepter le trafic à destination d'une adresse et d'un port, puis de le rediriger sur le réseau OpenZiti afin de le récupérer de l'autre coté.

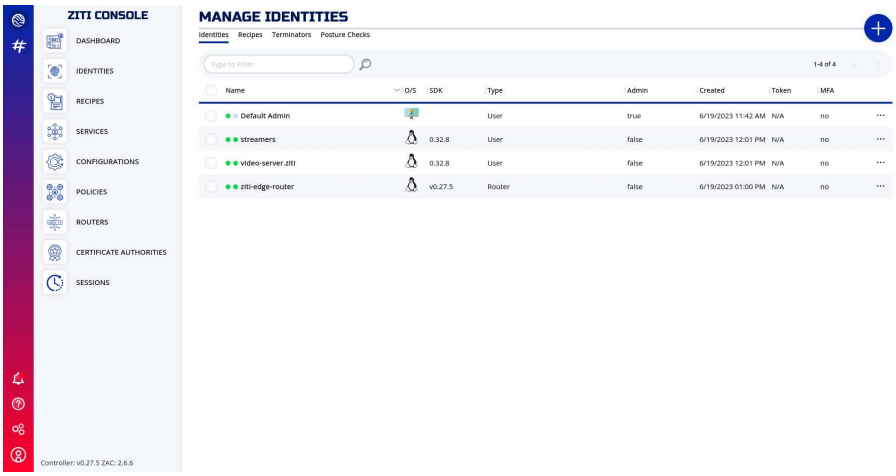

FIGURE A.3 – ZAC console

```
# configuration d'interception pour le serveur
# cela va récupérer le trafic à destination de video-server.ziti
# depuis le réseau OpenZiti et le rediriger sur la machine

ziti edge create config video.host.v1 host.v1 '{"protocol":"tcp",
↪  "address":"'"video-server.ziti"'", "port":1935}'

# configuration du streamer
# cela va intercepter le trafic à destination de video-server.ziti
# vers le port 1935 et l'envoyer sur le réseau OpenZiti
ziti edge create config streamers.intercept.v1 intercept.v1
↪  '{"protocols":["tcp"],"addresses":["video-server.ziti"],
↪  "portRanges":[{"low":1935, "high":1935}]}'
```

A.3.8 Service

Les services permettent d'associer des configurations afin de gérer le traffic. Dans notre cas, nous créer le service "video" qui permet de lier le streamer au serveur vidéo.

```
# service qui permet d'associer la config du server et celle du
↪  streamer
ziti edge create service video.svc --configs
↪  streamers.intercept.v1,video.host.v1
```

A.3.9 Politiques

Les politiques permettent de définir qui à le droit de se connecter et écouter un service. Dans notre cas, nous allons permettre au streamer d'envoyer des données sur le service vidéo, et au serveur d'écouter le service vidéo pour récupérer les données.

```
# politique qui permet au streamer de se connecter au service vidéo
ziti edge create service-policy streamer.policy.dial Dial
↪  --service-roles "@video.svc" --identity-roles '#streamers'

# politique qui permet au serveur vidéo d'écouter sur le service vidéo
ziti edge create service-policy video.policy.bind Bind --service-roles
↪  '@video.svc' --identity-roles "@video-server.ziti"
```

A.3.10 Inscription

La dernière étape consiste à inscrire le streamer et le serveur vidéo auprès du contrôleur. Cela se fera avec l'exécutable "ziti-edge-tunnel" qui va consommer le jeton jwt et produire un fichier de configuration qui sera utilisé lors du lancement du tunnel OpenZiti.

```
# inscrire le serveur vidéo
docker exec -it n1b-video-server-1 bash
cd /persistent
ziti-edge-tunnel enroll --jwt ./video-server.ziti.jwt --identity
↪  ./video-server.json
```

```
# inscrire le streamer
docker exec -it n1b-video-streamer-1 bash
cd /persistent
ziti-edge-tunnel enroll --jwt streamers.jwt --identity streamers.json
```

À ce stade, l'infrastructure est opérationelle et le tout peut être lancé via docker compose. Les différentes machines se connecteront automatiquer au résau OpenZiti, le streamer enverra son flux vidéo au serveur vidéo et ce dernier diffusera le flux aux clients qui s'y connectent.

Le flux vidéo peut être accédé depuis la machine hôte aux adresses suivantes :

`http://localhost:8080/players/hls.html`

`http://localhost:8080/players/hls_hlsjs.html`

Il faut quelques minutes pour que le flux soit disponibles après que le serveur ait été lancé.