Rapport d'installation

Nom prénom

26 avril 2021

Contents

1 Introduction

Dans ce documents, nous allons expliquer les choix effectués après notre analyse, une explication détaillé des aspects techniques, les étapes à suivre pour un déploiement sécurisé et nous allons voir quelque commande utile pour la maintenance.

2 Choix d'implémentation

Commençons par décrire les choix effectuer pour cette implémentation. Comme expliqué dans notre rapport d'analyse, nous avons décider d'opter pour la suite de programme de caméra de surveillance Kerberos.io, plus spécifiquement les modules Kerberos Agent et Kerberos Hub. Nous avons aussi choisis le service Shinobi Video comme notre solution d'urgence, ce dernier n'est pas une suite de solution mais est un programme *all-in-one* qui aura pour but d'être uniquement utilisé lorsque Kerberos Hub est hors-service. Tous ces services ont une images Docker, ce qui va énormément nous faciliter la tâche car nous pourrons déployer, maintenir et mettre à jour notre implémentation plus aisément. Nous avons décider de faire tourner Docker dans une machine Ubuntu Server 22.04 qui sera muni d'un processeur assez puissant pour faire tourner plusieurs containers en même temps et des des disque durs à hautes capacités pour l'enregistrement vidéo. Veuillez vous référer au rapport d'analyse précédant pour se familiariser au sujet des raisons de ces choix.

3 Aspect technique

3.1 Scripts de configuration

Nous avons produit ces fichier d'installations suivantes:

1. `install.sh`: Ce fichier `bash` va être notre configurateur principale qui va nous permettre d'effectuer une installation simple, tout en ayant un grand contrôle sur notre déploiement. Il s'agit de l'unique fichier à lancer lors d'une installation initiale.
2. `agent_generator.py`: Ce script `python`, qui se fera appelé par `install.sh`, a pour but de générer le fichier de configuration `docker-compose.yml` des instances de Kerberos Agent pour Docker. Ce script va demander à l'utilisateur de taper le nombre d'agent qu'il souhaite mettre en place, la créations des identifiants de connexion à Kerberos Agent, et les informations de connexion à la plateforme Kerberos Hub crée auparavant (*username; public key; private key*). Plus d'explication sur sont fonctionnement et son utilisation seront données lors de l'explication de l'implémentation.
3. `docker-compose.yml`: Il ne sera que générer après que l'utilisateur ait lancé le script d'installation, de ce fait il ne sera pas situé dans le fichier `.zip` donné au départ. Ce fichier va être appelé par la commande `docker compose` pour mettre en place tous les agents qui ont été *pré-configuré* automatiquement par l'utilisateur.
4. `agent_list.txt`: Celui-ci ne sera aussi générer à la qu'à la fin de l'installation et à pour but d'aider l'utilisateur à connaître les ports qui ont été allouer à chaque instance de Kerberos Agent. Il sera primordiale de les connaître du au fait que l'utilisateur devra se connecter à l'interface web ces instances pour plus de configurations au biais de l'adresse IP et du port alloué (`http://IPofDockerHost:portOfTheInstance`). Ce fichier donne aussi le chemin absolue des dossier d'enregistrement vidéo.

3.2 Commandes utiles

Il n'y pas un grand nombre de commande qui pourrait être utiles dû au fait que la suite Kerberos et Shinobi n'ont que leurs interfaces web pour leur gestion. Nous allons dès lors énumérer certaines commandes Docker. Ces commandes supposent que l'utilisateur a ajouté son compte *user* au *group* `docker`, si cela n'a pas été fait, veuillez vous référer aux instructions d'implémentation pour la bonne configuration. (voir 4.1.2)

1. `docker ps`: Affiche les *containers* qui sont en cours d'exécution, l'option `--all` permet d'afficher tous les `containers` y compris ceux qui ne sont pas en train de tourner.
2. `docker compose`: Cette commande permet de créer des *containers* à partir d'un fichier `.yml` (spécifier ou non), les options les plus importantes sont:
   (a) `-p <nom-du-projet>`: pour donner un nom de projet à l'ensemble des *containers* générer
   (b) `-f <fichier-à-compose>`: le fichier `.yml` à lire pour créer les containers, s'il n'est pas appelé, `docker compose` va automatiquement lire le fichier `docker-compose.yml`
   (c) `up`: pour lancer les containers après qu'il été créer

   De ce fait une commande typique: `docker compose -p mon-projet -f ficher-de-conf.yml up`
3. `docker run`: C'est une commande qui permet de créer et de lancer un seul container, il ne devrait qu'être utiliser par l'utilisateur s'il souhaite d'ajouter un Agent de manière manuel. Ces options majeurs sont:
   (a) `-p`: assigner un port au container
   (b) `-e`: les variables d'environnement prisent en charge par le container
   (c) `-v`: attacher un dossier du *host* au container
   (d) `-d`: lancer le container comme tâche de fond à la fin de sa création
4. `docker start <nom-du-conainter>`: Cette commande permet à l'utilisateur de lancer un *container* existant qui est à l'arrêt.
5. `docker stop <nom-du-container>`: Celle-ci permet d'arrêter de manières dites *gracieuse*, peut prendre du temps à s'arrêter.
6. `docker kill <nom-du-container>`: Elle permet d'arrêter immédiatement un container
7. `docker rm <nom-du-container>`: Permet de supprimer un container, l'option `prune` donné sans container, permet de supprimer tous les containers qui ne sont pas en cours d'utilisation

3.3 Ports utilisés

Pour ce qui est des ports utilisées, nous ne pouvons pas tous les listées car cela dépendra du nombre d'Agent que l'administrateur souhaite installer. Nous allouons néanmoins le port 8080 pour le service de secours, Shinobi Video et nous donnons la possibilités à l'utilisateur d'accéder aux ports 8081 jusqu'aux ports 9091 pour les Agents. L'administrateur peut facilement voir quels ports ont été alloué au Agents créer après qu'il est lancées le scripts d'installation dans le fichier `agent_list.txt`.

3.4 Démarrage des services

Par défaut, les services Docker ne se lancerons pas au démarrage de la machine sans que l'administrateur ait configuré cela. Les détails de la configuration seront données lors de l'installation des services. Pour ce qui est des Kerberos Agents, ils ont été réglé à se lancer lors du démarrage de Docker, donc l'administrateur ne doit pas les lancer un à un manuellement. Shinboi quant à lui, ne se lancera jamais automatiquement car il s'agit d'une solution *backup* et que comme il sera uniquement utilisé lorsque Kerberos Hub est hors-service, il ne fera que prendre des ressources en plus pour rien.

4 Étapes d'implémentations des services

4.1 Installation et configuration des services Docker

4.1.1 Installation de Docker

Commençons par installer Docker, la première chose à vérifier est que le système d'exploitation ne possède pas déjà une version `snap` de Docker. Nous n'allons pas utilisé cette version car elle ne possède pas tous les pouvoirs d'accès comparé à la version `apt` de Docker. En plus de cela avoir deux version dans une même machine peut poser énormément de problème au nivaux de la gestion. Pour désinstaller la version `snap`, nous allons faire la commande: `sudo snap remove docker`. Après cette opération effectuer, nous pouvons commencer l'installation de Docker.

1. Désinstallons d'ancienne version de Docker: `sudo apt-get remove docker docker-engine docker.io containerd runc`
2. Mettons à jour nos packages via `sudo apt-get update`
3. Installons les packages qui vont nous permettre d'installer les certificats: `sudo apt-get install ca-certificates curl gnupg`
4. Ajoutons la clé officielle GPG:
```
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | \
sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
```
5. Ajoutons le repository officielle de Docker à notre machine:
```
echo \
"deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] \
https://download.docker.com/linux/ubuntu \
"$(.  /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```
6. Maintenant que ne nous avons ajouter Docker à notre liste des package, mettons le à jour: `sudo apt-get update`
7. Installons la version la plus récente de Docker:
```
sudo apt-get install docker-ce docker-ce-cli \
containerd.io docker-buildx-plugin docker-compose-plugin
```
8. Comme tous les parties de Docker ont été installés, nous pouvons lancer un container *test* pour voir si tout s'est bien déroulé: `sudo docker run hello-world`, si il affiche une quelconque message d'erreurs veuillez recommencer l'étape d'installation de Docker.
9. Pour finir, nous allons installer la commande `docker-compose` qui est une version *legacy* de la nouvelle commande `docker compose`. Elle est utilisé uniquement pour l'installation de Shinobi Video. Installons le avec la commande `sudo apt-get install docker-compose`

4.1.2 Configuration de Docker

Un Docker non configuré ne se lancera pas lors du lancement d'une machine, nous devons le configurer manuellement. Nous devons aussi ajouter notre compte `$USER` à un groupe `docker` pour que nous ne devions pas taper `sudo docker` (e.g.: `docker run hello-world` au lieu de `sudo docker run hello-world`) à chaque fois. De ce fait, nous supposerons que notre serveur n'a qu'un utilisateur créer.
Pour lancer automatiquement les services Docker au *boot*:

1. Ajoutons au *boot*: `sudo systemctl enable docker`
2. Après que les services aient étés configuré, nous pouvons faire de tel sortes que Docker se lance directement au démarrage: `sudo systemctl start docker`

Ajoutons notre `$USER` au groupe `docker` que nous allons créer:

1. Créons le groupe `docker`: `sudo groupadd docker`
2. Intégrons notre utilisateur à ce groupe: `sudo usermod -aG docker $USER`

4.2 Installation des solutions Kerberos et Shinobi Video

4.2.1 Création d'un compte Kerberos Hub et configuration des sous-compte

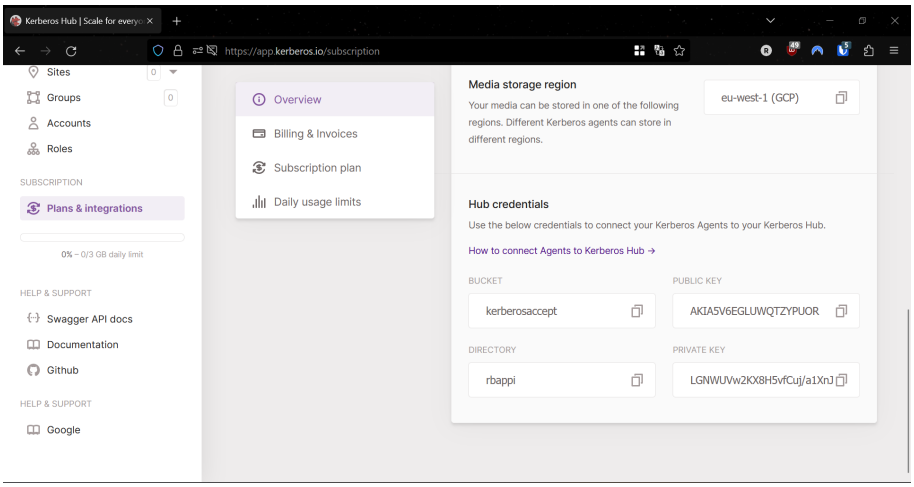
Avant de mettre en ligne les Agents, nous allons devoir créer un compte Kerberos Hub qui va nous permettre de visualiser en direct le flux de tous nos Agents, donc nos caméras. Nous n'allons pas utilisé la version Kubernetes d'Hub car Kerberos.io nous donnes déjà un accès simple à ces services moyennant un frais mensuelle fixe.

1. Allons sur le site de Kerberos Hub.
2. Créons un compte avec un mot de passe robuste, nous recommandons aussi de modifier ce mot de passe régulièrement pour des raison de sécurités.
3. Souscrivons à un abonnement qui nous permettra de visualiser en direct les flux donc tous les *plans* à partir de *Pro*.
4. (Optionnel) A partir du *dashboard*, nous pouvons facilement assigner des pouvoirs en créons des comptes et des groupes (appelé *roles* ici). Pour faire cela, nous devons aller sur *Roles*, il y aura déjà quelque groupes de créer mais nous pouvons en ajouter d'avantages. Pour ensuite déléguer certain pouvoir, nous devons aller dans *Accounts* où nous pouvons créer d'autre utilisateur en leur donnant accés ou non à des fonctionnalités grâce au *roles* créer. Une bonne pratique serait de donner que le pouvoirs de visionnage aux agents de sécurités et de garder les pouvoirs de gestion et entretiens aux administrateurs système.

4.2.2 Création et mise en place des solutions de surveillance

Comme expliqué plus haut, nous avons développer des fichier de qui permettra de facilement créer des containers d'Agents et de créer un unique container de Shinobi Video.

1. Avant de lancer notre programme d'installation, nous devons avoir en main notre nom d'utilisateur créer pour Kerberos Hub, notre clé public et notre clé privé. Ils sont visible dans l'onglet `Plans & Integrations` de l'interface web d'Hub.



2. Commençons par mettre à jour nos packets: `sudo apt-get update`
3. Retournons sur notre machine Ubuntu Server, dézippons le fichier `install.zip` par la commande `unzip install.zip`. Si la commande n'est pas disponible, nous pouvons installer celle-ci grâce à `sudo apt-get install unzip`.
4. Ouvrons le dossier qui a été créer, `install`, et lançons la commande `./install.sh`. S'il y une erreur de permission, nous pouvons la modifier avec `sudo chmod 777 install.sh`. Il faut faire attention aux instructions données lors de la création des containers Kerberos Agents car le nom d'utilisateur et le mot de passe créer ne pourront pas être modifier dans le future. Ceux-ci seront utilisées pour accéder l'interface web.

Nous pouvons aussi installer une unique instance de Kerberos Agents grâce à la commande:

```
docker run -p [port souhaité]:80 --name [nom de caméra désiré] \
-v $(pwd)/agent/config:/home/agent/data/config \
-v $(pwd)/agent/recordings:/home/agent/data/recordings\
-e AGENT_NAME=[nom de caméra désiré]\
-e AGENT_USERNAME=[pseudo de connexion]\
-e AGENT_PASSWORD=[mot de passe de connexion]\
-e AGENT_HUB_KEY=[clé public de Hub]\
-e AGENT_HUB_PRIVATE_KEY=[clé privé de Hub]\
-e AGENT_HUB_USERNAME=[pseudo de Hub]\
-d --restart=alwayskerberos/agent:latest
```

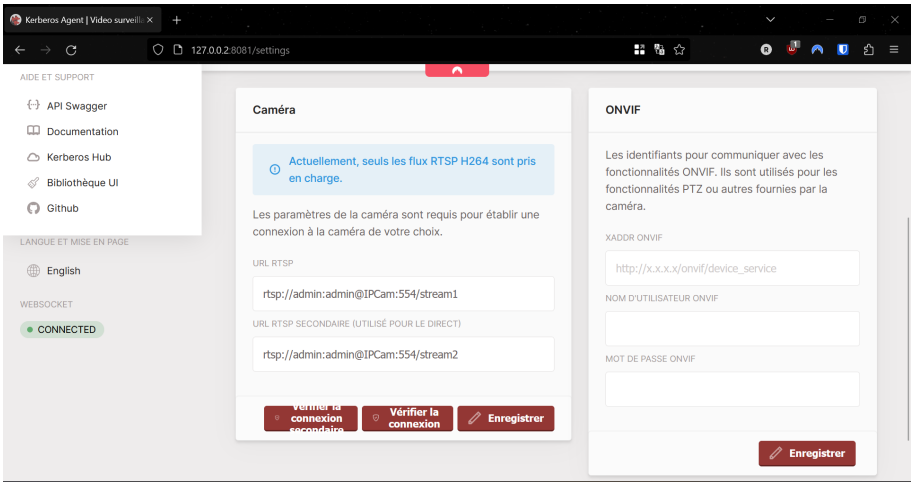
4.2.3 Configuration des Agents et ajout de caméra

Le script d'installation va nous installer les Agents avec toutes les configurations de bases pour l'intégration avec Kerberos Hub. Cela dit, l'administrateur devra ajouter les liens de connexion entre l'Agent et les caméra IP grâce au protocole RTSP dont la majorité des caméra et DVR de surveillance prennent en charge. Nous devons aussi configurer l'enregistrement de flux vidéo.

1. Avant de commencer à configurer, nous devons nous familiariser avec l'accès à l'interface web. C'est ici que vient jouer le fichier `agent_list.txt`, nous pouvons retrouver les Agents et leur ports (@port) qui leurs ont été allouées. Pour accéder à l'interface, nous devons aller sur un navigateur internet et taper l'adresse IP du serveur tournant Docker suivis du port qui a été alloué à au container que nous souhaitons accéder. (e.g.: `http://127.0.0.2:8081`)

2. Connectons nous avec le nom d'utilisateur et mot de passe créer lors de l'installation aux étapes précédentes.

3. Arrivé au *dashboard*, nous allons nous dirigé vers *Configure*, cliquons sur l'onglet *Camera*. Ici, nous pouvons ajouter le flux RTSP d'une caméra *divisé* en deux. Il faut savoir que la plupart des caméra de surveillance transmettent deux flux RTSP en même temps, un de qualité *supérieur*, et le second de qualité *inférieur* qui sera donc un backup. Nous devons donc taper le flux de meilleur qualité dans le premier champs et flux de qualité moindre dans le second. Si la caméra de surveillance ne supporte qu'un flux, nous allons remplir uniquement le premier champs. Après avoir appuyer su *Save*, nous pouvons retourner sur le *dashboard* pour afficher le flux.



4. Dans l'onglet *Recordings*, nous allons activé l'option *Continous recordings* et augmenter la durée des enregistrements, car nous aurons à notre disposition beaucoup de disque dur pour les sauvegarder. Nous recommandons aussi d'activer l'option *Enable auto-clean* et augmenter la taille de dossier, pour que nous puisons accéder les enregistrement plus longtemps. Pour accéder au fichier de ces enregistrement, l'utilisateur doit aller sur le chemin donner par `agent_list.txt` (@path of recordings).

5. Nous devons maintenant refaire les étapes 1 à 4 pour les autres restants.

6. Dès que nous avons terminer la configurations des Agents, nous pouvons nous diriger sur le site de Kerberos Hub pour voir le flux de tous nos Agents dans l'onglet *Live View*.

4.2.4 Configuration de Shinobi Video et ajout de caméra

Comme indiqué précédemment, Shinobi à été installé au port 8080 de l'hôte. La configuration est bien évidemment différentes de Kerberos car c'est une solution *all-in-one*. Nous supposerons ici aussi que nous avons un flux RTSP.

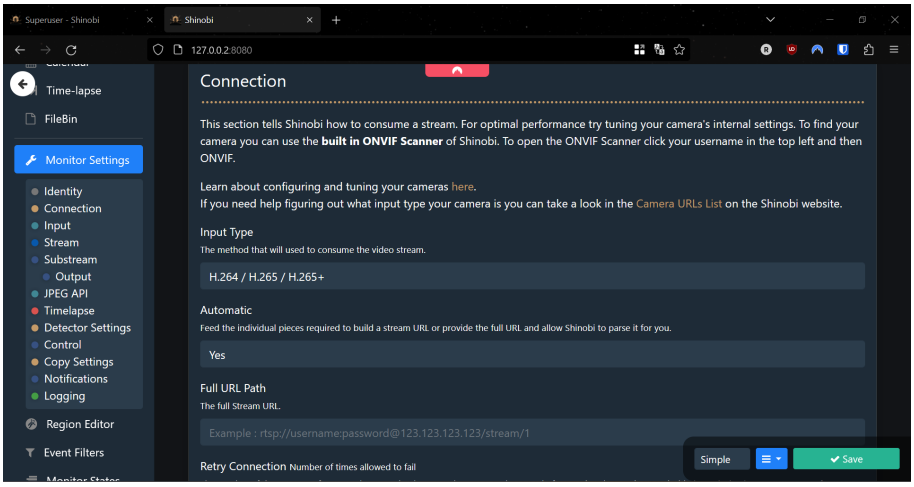
1. Nous devons commencer par créer un compte administrateur par le biais de l'interface web. Nous devons aller sur `http://YOUR_SHINOBI_SERVER_IP_ADDRESS:8080/super` qui ne gère que les données d'identification pour accéder aux services et non au *dashboard* de visionnage. Arrivé sur la page de connexion, le nom d'utilisateur par défaut est `admin@shinobi.video` et le mot de passe est `admin`.

2. Dans le *dashboard*, nous allons d'abord changer les identifications dans l'onglet *Preferences*.

3. Pour nous connecter au *dashboard* de visionnage, nous devons créer des comptes dans l'onglet *Accounts*. Les paramètres les plus importants sont *Email*, *Password* et *Permissions*.

4. Nous pouvons maintenant accéder au visionnage grâce au site `http://YOUR_SHINOBI_SERVER_IP_ADDRESS:8080` avec les identifications que nous avons créer.

5. Pour ajouter une caméra, nous allons sur l'onglet *Monitor Settings* sur la gauche. Nous allons nous diriger dans la catégorie *Connection*. Nous allons remplir le lien *Full URL path* par notre lien de flux RTSP.



6. Finalement, nous pouvons visualiser en direct notre flux sur l'onglet *Live Grid* pour voir si tout s'est déroulé correctement.

5 Maintenance et contrôle des services

5.1 Maintenance

5.1.1 Maintenance des services Docker

Comme nous avons ajouter le *repository* de Docker à notre machine, nous ne devons simplement faire les commandes: `sudo apt-get update` et `sudo apt-get upgrade`.

5.1.2 Maintenance des services Kerberos.io et Shinobi Video

Comme ils s'agient de containers, les administrateurs ne deront pas faire énormément de maintenance. Nous pouvons néanmoins mettre à jour les images Docker de ces containers avec `docker images --format ".Repository:.Tag" | xargs -L1 docker pull`. Nous recommandons fortement aussi de mettre à jour régulièrement les identifications d'accès au solutions de vidéo-surveillance, pour renforcer la sécurité.

5.2 Contrôle des services Docker

Nous avons déjà mentionné quelques commandes intéressente dans la section 3.2 qui ne sont que lier à Docker. Nous allons ici décrire quelque commande recommandées pour le contrôle des services.

1. Mise en pause temporaires des services Docker: `sudo systemctl stop docker`
2. Réactiver les services Docker: `sudo systemctl start docker`
3. Recharger les services Docker: `sudo systemctl reload docker` ou `sudo systemctl restart docker`