****

**420-414 - Infonuagique**

**Examen final**

**Rapport d’implémentation**

**DA : 6240201**

**Nom, prénom : Marc-Antoine Mercier**

**/ 25**

## Table des matières

[**1 - Introduction 3**](#_67nisotpon4)

[**2 - Infrastructure 3**](#_j62286n7fjad)

[2.1 - Schéma de l’infrastructure réseau (AWS) 3](#_j6ma99rr0pvo)

[2.2 - Description des composants réseau 3](#_u1iicythxd2)

[2.3 - Groupes de sécurité 4](#_t8x2t6vlje6f)

[2.4 - Configuration de l’instance EC2 4](#_q90eecp77fws)

[**3 - Déploiement des services : Docker et Docker Compose 5**](#_ugbwfa41gfcq)

[3.1 - Architecture Docker 5](#_7jmlpcyqzfaa)

[3.2 - Services déployés 5](#_jrpsv5l9c7e9)

[3.3 - Fichier docker-compose.yml 5](#_b8kaa8lz1lz0)

[**4 - Implémentation d’un proxy inverse (Traefik) 5**](#_rlbo8nkihx44)

[**5 - Nom de domaine 5**](#_qclcglxxb2y8)

[**6 - Haute disponibilité et résilience 5**](#_fhaps1caarg5)

[**7 - Automation avec Terraform (optionnel) 6**](#_2uz4mklkfs7c)

[**8 - Problèmes rencontrés et solutions apportées 6**](#_3uc3rhl53fye)

[**9 - Conclusion 6**](#_89m3hbrc49ea)

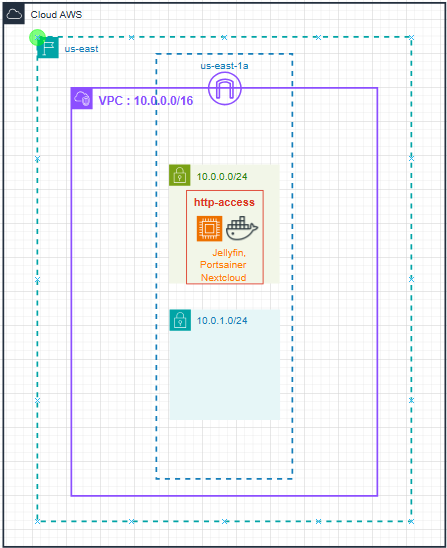
[**Annexes (facultatif) 6**](#_5rgqrmomrf0j)

## 1 – Introduction

Dans le cadre du cours d’infonuagique, nous avons dû concevoir et faire un rapport pour un architecture répondant au besoin d’une infrastructure multi-services. Les services que je vais déployer sont les suivantes Portainers, Nextcloud, Jellyfin, mon frontend et mon backend de mon application web.

## 2 - Infrastructure

##### 2.1 - Schéma de l’infrastructure réseau (AWS)



##### 2.2 - Description des composants réseau *:*

|  |  |
| --- | --- |
| ***Nom du VPC*** | VPC-EpreuveFinale |
| ***Région*** | Us-east-1 |
| ***Plage d’adresses IPv4*** | 10.0.0.0/16 |
| ***Configuration de la connectivité*** | Passerelle internet |

|  |  |
| --- | --- |
| ***Nom de la table de routage*** | Ep\_public\_route\_table |
| ***Règles de routage*** | |
| ***Destination*** | ***Cible*** |
| 0.0.0.0/0 | Passerelle Internet |
| 10.0.0.0/16 | local |

|  |  |
| --- | --- |
| ***Nom de la table de routage*** | Ep\_private\_route\_table |
| ***Règles de routage*** | |
| ***Destination*** | ***Cible*** |
| 10.0.0.0/16 | local |

|  |  |
| --- | --- |
| ***Nom du sous-réseau*** | Epreuve-finale-public-1 |
| ***Zone de disponibilité*** | Us-east-1 |
| ***Plage d’adresse IPv4*** | 10.0.0.0/24 |
| ***Public/Privé*** | Public |
| ***Table de routage associée*** | Ep\_public\_route\_table |

|  |  |
| --- | --- |
| ***Nom du sous-réseau*** | Epreuve-finale-private-1 |
| ***Zone de disponibilité*** | Us-east-1 |
| ***Plage d’adresse IPv4*** | 10.0.1.0/24 |
| ***Public/Privé*** | Privé |
| ***Table de routage associée*** | Ep\_private\_route\_table |

##### 2.3 - Groupes de sécurité

|  |  |  |  |
| --- | --- | --- | --- |
| ***Nom du groupe de sécurité*** | | HTTPS | |
| ***Description*** | | Allow ssh and http and https | |
| ***Règles entrantes*** | | | |
| ***Type*** | ***Protocole*** | ***Plage de ports*** | ***Source*** |
| SSH | TCP | 22 | 22 |
| ***Type*** | ***Protocole*** | ***Plage de ports*** | ***Source*** |
| HTTP | TCP | 80 | 80 |
| ***Type*** | ***Protocole*** | ***Plage de ports*** | ***Source*** |
| HTTPS | TCP | 443 | 443 |
| ***Règles sortantes*** | | | |
| ***Type*** | ***Protocole*** | ***Plage de ports*** | ***Source*** |
| All | All | All | 0.0.0.0/0 (Any) |

***Les groupes sécurités dans le cloud AWS sont dit « stateful ». Le terme « stateful » pour les groupes de sécurité signifie qu’on définie seulement les règles d’entré, car les règles de sortie accepte tout (le trafic sortant est toujours autorisé).***

2.4 - Configuration de l’instance EC2

|  |  |
| --- | --- |
| ***Nom de l’instance*** | Ep\_instance |
| ***Type de l’instance*** | t2.Large |
| ***AMI*** | Ubuntu 24.04 LTS |
| ***VPC*** | VPC-EpreuveFinale |
| ***Sous-réseau*** | Epreuve-finale-public-1 |
| ***Groupe(s) de sécurité*** | HTTPS |

*Je compte configurer l’instance par ssh, car cela n’est pas une bonne pratique de faire un userdata file, car il peut avoir des modifications des les dépendances de l’instance, donc la meilleur pratique que je n’ai pas fait serait de faire « un golden images » de l’instance pour respecter le principe de l’*idempotence *(input A = toujours output B) « fonction pur en programmation fonctionnel ». Les étapes que j’ai fait pour configurer mon instance sont les suivantes, me connecter ssh avec la clé ssh gérer quand j’ai déployé mon infrastructure en tant que Code (IaC). Quand je me suis connecter sur l’instance j’ai du clone mon repository github avec la commande suivante (git clone https://* [*https://github.com/Marc-AntoineMercier/Session4.Infonuagique.EF*](https://github.com/Marc-AntoineMercier/Session4.Infonuagique.EF)*), après je me suit déplacer avec la commande (cd Session4.Infonuagique.EF) et après j’ai du lancer mon docker compose file qui me permet de lancer les services que j’ai choisie.*

## 3 - Déploiement des services : Docker et Docker Compose

##### 3.1 - Architecture Docker

##### 3.2 - Services déployés

Jellyfin est un service qui permet de stocker des films, séries, musiques et autre type de consumable. Jellyfin utilise le port 8096. Comme variable d’environement spécial j’ai mis le time zone à Montréal pour avoir la bonne heure de plus j’ai mis des volumes pour garder les informations de mon Jellyfin même si le container a une erreur.

Nextcloud est un service qui permet de faire comme teams, donc un drive personnel ou on peut stocker les fichiers et utiliser un système de messagerie. Je l’ai déployé sur le port 8081. Les variables d’environnement sont le time zone, mettre un mot de passe pour se connecter à la base de données qu’il a besoin pour fonctionner et mettre les informations de connexion a cette base de données, à cause que j’ai fait le service de Next cloud j’ai dû implémenter une base de données mariadb, j’ai mis les même variables d’environnement. De plus, j’ai mis des volumes pour garder les informations même si le container à un problème.

Portainer est un service qui permet de gérer les containers dockers avec un Ui pour ne pas le faire en ligne de commande. Dans ce service je n’ai pas utiliser de volume, car il n’y a pas d’information que je veux garder si il y a un problème avec le container.

*Listez et décrivez les services choisis parmi ceux proposés.*

##### 3.3 - Fichier *docker-compose.yml*

# Définit les conteneurs à déployer

services:

  # Interface web de gestion Docker

  portainer:

    # Image Docker à utiliser

    image: portainer/portainer-ce:alpine

    # Nom personnalisé du conteneur

    container\_name: portainer

    # Redémarre sauf si arrêté manuellement

    restart: unless-stopped

    # Réseau(s) auquel le conteneur appartient

    networks:

      - cloud

    # Redirige les ports hôte vers le conteneur

    # ports:

    #   - 9000:9000

    # Monte des répertoires ou fichiers dans le conteneur

    volumes:

      - /var/run/docker.sock:/var/run/docker.sock

      - portainer\_data:/data

    # Ajoute des métadonnées, souvent pour Traefik

    labels:

      - 'traefik.enable=true'

      - 'traefik.http.routers.portainer.rule=Host(`portainer.${MY\_DOMAIN}`)'

      - 'traefik.http.routers.portainer.entryPoints=websecure'

      - "traefik.http.services.portainer.loadbalancer.server.port=9000"

    # Définit des variables d’environnement

    environment:

      - TZ=America/Montreal

  # Base de données pour Nextcloud

  nextclouddb:

    image: mariadb

    container\_name: nextcloud-db

    restart: unless-stopped

    # Commandes spécifiques au démarrage

    command: --transaction-isolation=READ-COMMITTED --binlog-format=ROW

    networks:

      - cloud

    volumes:

      - nextclouddb:/var/lib/mysql

    environment:

      - PUID=1000

      - PGID=1000

      - TZ=America/Montreal

      - MYSQL\_RANDOM\_ROOT\_PASSWORD=true

      - MYSQL\_PASSWORD=${DB\_PASSWORD}

      - MYSQL\_DATABASE=${DB\_DATABASE}

      - MYSQL\_USER=${DB\_USER}

  # Application Nextcloud (cloud personnel)

  nextcloud:

    image: nextcloud

    container\_name: nextcloud

    restart: unless-stopped

    networks:

      - cloud

    # Démarre après la base de données

    depends\_on:

      - nextclouddb

    # ports:

    #   - 8081:80

    volumes:

      - /local/nextcloud:/var/www/html

    environment:

      - PUID=1000

      - PGID=1000

      - TZ=America/Montreal

      - MYSQL\_PASSWORD=${DB\_PASSWORD}

      - MYSQL\_DATABASE=${DB\_DATABASE}

      - MYSQL\_USER=${DB\_USER}

    labels:

      - 'traefik.enable=true'

      - 'traefik.http.routers.nextcloud.rule=Host(`nextcloud.${MY\_DOMAIN}`)'

      - 'traefik.http.routers.nextcloud.entryPoints=websecure'

  # Serveur multimédia Jellyfin

  jellyfin:

    image: jellyfin/jellyfin:latest

    container\_name: jellyfin

    restart: unless-stopped

    networks:

      - cloud

    # ports:

    #   - 8096:8096

    environment:

      - TZ=America/Montreal

    volumes:

      - jellyfin\_data:/config

      - /mnt/media/Movies:/movies

      - /mnt/media/TV:/tv

    labels:

      - 'traefik.enable=true'

      - 'traefik.http.routers.jellyfin.rule=Host(`jellyfin.${MY\_DOMAIN}`)'

      - 'traefik.http.routers.jellyfin.entryPoints=websecure'

  # Reverse proxy Traefik

  proxy:

    image: traefik

    container\_name: traefik

    restart: unless-stopped

    networks:

      - cloud

    # Options de configuration en ligne de commande

    command:

      - "--log.level=DEBUG"

      - "--api.insecure=true"

      - "--providers.docker=true"

      - "--providers.docker.exposedbydefault=false"

      - "--certificatesresolvers.letsencrypt.acme.dnschallenge=true"

      - "--certificatesresolvers.letsencrypt.acme.dnschallenge.provider=duckdns"

      - "--certificatesresolvers.letsencrypt.acme.email=mail@mail.com"

      - "--certificatesresolvers.letsencrypt.acme.dnschallenge.disablePropagationCheck=true"

      - "--certificatesresolvers.letsencrypt.acme.dnschallenge.resolvers=1.1.1.1:53,8.8.8.8:53"

      - "--certificatesresolvers.letsencrypt.acme.storage=/letsencrypt/acme.json"

      - "--entrypoints.web.address=:80"

      - "--entrypoints.web.http.redirections.entrypoint.to=websecure"

      - "--entrypoints.web.http.redirections.entrypoint.scheme=https"

      - "--entrypoints.websecure.address=:443"

      - "--entrypoints.websecure.http.tls=true"

      - "--entrypoints.websecure.http.tls.certResolver=letsencrypt"

      - "--entrypoints.websecure.http.tls.domains[0].main=${MY\_DOMAIN}"

      - "--entrypoints.websecure.http.tls.domains[0].sans=\*.${MY\_DOMAIN}"

    volumes:

      - "../data/traefik/letsencrypt:/letsencrypt"

      - "/var/run/docker.sock:/var/run/docker.sock:ro"

    labels:

      - 'traefik.enable=true'

      - 'traefik.http.routers.api.rule=Host(`${MY\_DOMAIN}`)'

      - 'traefik.http.routers.api.entryPoints=websecure'

      - 'traefik.http.routers.api.service=api@internal'

    ports:

      - "443:443"

      - "80:80"

    environment:

      - "DUCKDNS\_TOKEN=${DUCKDNS\_TOKEN}"

# Définit les volumes Docker persistants

volumes:

  portainer\_data:

  nextclouddb:

  jellyfin\_data:

# Définit les réseaux Docker

networks:

  cloud:

    # Utilise le réseau bridge (par défaut)

    driver: bridge

lien repo github : [*https://github.com/Marc-AntoineMercier/Session4.Infonuagique.EF*](https://github.com/Marc-AntoineMercier/Session4.Infonuagique.EF)

J’ai organisé les services de pour que ceux-ci redémarrent s’il y a un problème. En gros, ils vont rouler tant que si long temp que je n’arrête pas les containers manuellement avec docker compose down, la ligne qui permet de faire cela est « restart : unless-stopped ». J’utilise pour des volumes pour garder les informations même si le container crash pour les services de nextclouddb, nextcloud et jellyfin. Je n’ai pas mis de depends\_on, car je n’en ai pas besoin, car ils vont rouler en même temps. En plus, quand j’utilise networks je les mets sur le même réseau pour que traefik permet de faire le reverse proxy.

## 4 - Implémentation d’un proxy inverse (*Traefik*)

J’ai implémenté Traefik en mettant des labels sur chaque container pour définir la facon d’y accéder et mettre que cela fonctionne avec https. J’ai implémenté le chiffrement ssl/tls avec duckdns en utilisant le token qui me donne. Il y a tout un fonctionnement en arrière avec let encrypt qui ca confirmé que nous somme bien le propriétaire de ce nom de domaine.

## 5 - Nom de domaine

|  |  |
| --- | --- |
| ***Fournisseur de nom de domaine utilisé*** | Duckdns |

|  |  |
| --- | --- |
| ***Service*** | ***Adresse complète*** |
| Jellyfin | https://jellyfin.duckdns.org |
| Portainer | https://portainer.duckdns.org |
| Nextcloud | https://nextcloud.duckdns.org |

## 6 - Haute disponibilité et résilience

Le mécanisme pour assurer la disponibilité de mes services sont de redémarrer mes containers s’il y a un problème avec docker. Une autre stratégie pour assurer la haute disponibilité serait de déployer sur plusieurs zones de disponibilité grâce à un load balancer qui gérais les instances dans chaque zone avec un « golden images »

## 7 - Automation avec Terraform (*optionnel*)

J’ai structuré mes fichiers .tf avec la logique suivante chaque chose a son fichier provider pour gérer qui est le provider, network qui gère tout le coter network, security qui gère les groupes de sécurité et le fichier instance pour gère la création des instances, chaque fichier a un fichier de variable pour voir les variables que le fichier de configuration utilise. L’avantage de faire une architecture IaC est de pouvoir le répèter l’infrastructure à tout moment, de faire du versionnage et réduit la probabilité de bug, car on fait du code et on va le voir plus facilement comparer a des clics dans la console aws.

## 8 - Problèmes rencontrés et solutions apportées

Apprendre terraform comment il fonctionne avec certain type comme object.

Tous ces biens passés, car j’ai suivi la logique de tous les travaux pratique plus tous les ateliers faits en classe avec vous. En plus dans les travaux pratiques j’ai toujours faite la partie optionnelle se qui a facilité l’épreuve finale.

## 9 – Conclusion

J’ai beaucoup aimé faire les cours avec vous, mais durant le cours d’infonuagique j’ai appris les bonnes pratiques pour faire du DevOp avec les différents types d’infrastructure. Les seules chose que je ferait différemment sont d’apprendre terraform plus tôt, car vraiment plus simple que la console aws.

## Annexes (facultatif)