****

**420-414 - Infonuagique**

**Examen final**

**Rapport d’implémentation**

**DA : 6225454**

**Nom, prénom : Djerroud, Takfarinas**

**/ 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)

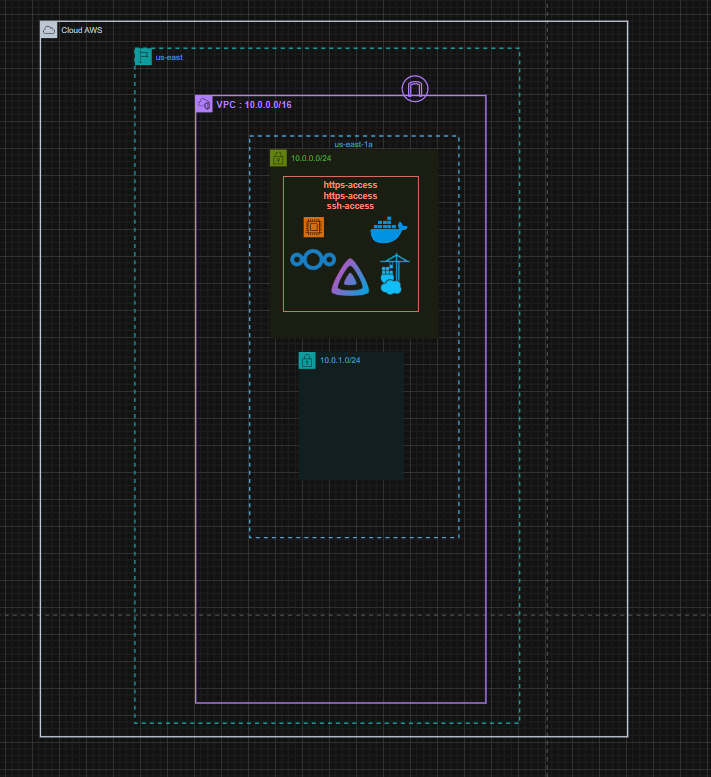
[**Annexes (facultatif) 6**](#_5rgqrmomrf0j)

## 1 - Introduction

Ce projet consiste au déploiement d’un projet grâce à AWS, Docker et Terraform. Les services déployés sont Portainer, Nextcloud, Jellyfin et Traefik.

## 2 - Infrastructure

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



##### 2.2 - Description des composants réseau

|  |  |
| --- | --- |
| ***Nom du VPC*** | VPC-tpfinal |
| ***Région*** | us-east-1 |
| ***Plage d’adresses IPv4*** | 10.0.0.0/16 |
| ***Configuration de la connectivité***  ***(Passerelle Internet ou autre)*** | Passerelle Internet (IGW) |

|  |  |
| --- | --- |
| ***Nom de la table de routage*** | tpfinal-rtb-public |
| ***Règles de routage*** | |
| ***Destination*** | ***Cible*** |
| 0.0.0.0/0 | IGW |

|  |  |
| --- | --- |
| ***Nom de la table de routage*** | tpfinal-rtb-private |
| ***Règles de routage*** | |
| ***Destination*** | ***Cible*** |
| 10.0.0.0/16 | local |

|  |  |
| --- | --- |
| ***Nom du sous-réseau*** | tpfinal-public-1 |
| ***Zone de disponibilité*** | us-east-1a |
| ***Plage d’adresse IPv4*** | 10.0.0.0/24 |
| ***Public/Privé*** | Public |
| ***Table de routage associée*** | tpfinal-rtb-public |

|  |  |
| --- | --- |
| ***Nom du sous-réseau*** | tpfinal-private-1 |
| ***Zone de disponibilité*** | us-east-1a |
| ***Plage d’adresse IPv4*** | 10.0.1.0/24 |
| ***Public/Privé*** | Privé |
| ***Table de routage associée*** | tpfinal-rtb-private |

##### 2.3 - Groupes de sécurité

|  |  |  |  |
| --- | --- | --- | --- |
| ***Nom du groupe de sécurité*** | | ssh-access | |
| ***Description*** | | Autorise accès SSH | |
| ***Règles entrantes*** | | | |
| ***Type*** | ***Protocole*** | ***Plage de ports*** | ***Source*** |
| Entrante | TCP | 22 | 0.0.0.0/0 |
| ***Règles sortantes*** | | | |
| ***Type*** | ***Protocole*** | ***Plage de ports*** | ***Source*** |
| Sortante | Tous | Tous | 0.0.0.0/0 |

**Justification : Accès SSH pour administration.**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Nom du groupe de sécurité*** | | http-access | |
| ***Description*** | | Autorise accès HTTP | |
| ***Règles entrantes*** | | | |
| ***Type*** | ***Protocole*** | ***Plage de ports*** | ***Source*** |
| Entrante | TCP | 80 | 0.0.0.0/0 |
| ***Règles sortantes*** | | | |
| ***Type*** | ***Protocole*** | ***Plage de ports*** | ***Source*** |
| Sortante | Tous | Tous | 0.0.0.0/0 |

**Justification : Accès HTTP pour les services web.**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Nom du groupe de sécurité*** | | https-access | |
| ***Description*** | | Autorise accès HTTPS | |
| ***Règles entrantes*** | | | |
| ***Type*** | ***Protocole*** | ***Plage de ports*** | ***Source*** |
| Entrante | TCP | 443 | 0.0.0.0/0 |
| ***Règles sortantes*** | | | |
| ***Type*** | ***Protocole*** | ***Plage de ports*** | ***Source*** |
| Sortante | Tous | Tous | 0.0.0.0/0 |

***Justification : Accès HTTPS pour les services web.***

##### 2.4 - Configuration de l’instance EC2

|  |  |
| --- | --- |
| ***Nom de l’instance*** | web-server |
| ***Type de l’instance*** | t2.large |
| ***AMI*** | ami-084568db4383264d4 |
| ***VPC*** | VPC-tpfinal |
| ***Sous-réseau*** | tpfinal tpfinal-public-1 |
| ***Groupe(s) de sécurité*** | ssh-access, http-access, https-access |

Dans AWS je vais aller sur mon instance qui a été créer grâce a mon script Terraform. Je vais cliquer dessus puis sélectionner « Se connecter ». Je vais prendre la commande pour rentrer dans mon instance puis l’executer. Une fois dedans, je vais installer docker grâce aux scripts dans le site du cours. Ensuite je vais créer un fichier .env et compose.yml. Finalement une fois que le tout sera configuré a l’aide du site du cours, je vais lancer mes conteneurs avec la commande docker compose up -d.

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

##### 3.1 - Architecture Docker

*A screenshot of a computer

AI-generated content may be incorrect.*

##### 3.2 - Services déployés

* Portainer :
  + Rôle : Interface pour gérer tes conteneurs Docker.
  + Port utilisé : 9000
  + Configuration spécifique :
    - Volumes : dossier de stockage gété par Docker
    - Redémarrage automatique : unless-stopped
* Nextcloud :
  + Rôle : Cloud perso pour gérer ses fichiers.
  + Port utilisé : 80
  + Configuration spécifique :
    - Dépendance : nextclouddb
    - Volumes dossier de stockage gété par Docker
    - Variables d’environnement : personnalisation de ton service
    - Réseau : cloud
    - Redémarrage automatique : unless-stopped
* MariaDB (Nextcloud DB) :
  + Rôle : Base de données pour Nextcloud.
  + Port utilisé : 3306
  + Configuration spécifique :
    - Volumes : dossier de stockage gété par Docker
    - Variables d’environnement : personnalisation de ton service
    - Réseau : cloud
    - Redémarrage automatique : unless-stopped
* Jellyfin :
  + Rôle : Serveur multimédia comme films et séries.
  + Port utilisé : 8096
  + Configuration spécifique :
    - Volumes : dossier de stockage gété par Docker
    - Variables d’environnement : personnalisation de ton service
    - Réseau : cloud
    - Redémarrage automatique : unless-stopped

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

Mon fichier définit séparément chaque service (portainer, etc) dans son propre bloc avec son image Docker. Il relie tous les conteneurs à un même réseau (cloud) pour qu’ils puissent communiquer. Présence de volume en tant que backup qui stockent les données. Il y aussi restart : unless-stopped au cas ou ça plante, le Docker relance tout solo. Pour Nextcloud il y a depends\_on qui attend que la bd soit prête à marcher avant de démarrer l’appli. Traefik lui redirige les requêtes http/s.

services:

portainer:

# Image de portainer officielle (ce=community edition, alpine=légère)

image: portainer/portainer-ce:alpine

# Nom donné au conteneur

container\_name: portainer

# Dans le cas où le conteneur s'arrête/crash, il est automatiquement relancé

restart: unless-stopped

# Réseau docker dans lequel le conteneur va s'exécuter

networks:

- cloud

# Mappage de port (du port 9000 du conteneur vers le port 9000 de la machine hôte)

#ports:

# - 9000:9000

# Volumes

volumes:

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

- portainer\_data:/data

labels:

- 'traefik.enable=true'

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

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

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

- "traefik.http.routers.frontend.service=frontend"

nextclouddb:

# Image mariadb officielle pour entreposer les données de nextcloud

image: mariadb

# Nom donné au conteneur

container\_name: nextcloud-db

# Dans le cas où le conteneur s'arrête/crash, il est automatiquement relancé

restart: unless-stopped

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

# Réseau dans lequel le conteneur va s'exécuter (dans le même réseau que nextcloud)

networks:

- cloud

# Volumes du conteneur: mappe un répertoire du conteneur vers un volume docker pour faire persister les données

volumes:

- nextclouddb:/var/lib/mysql

# Variables d'environnement passés au conteneur lors de son lancement

environment:

- PUID=1000

- PGID=1000

# Timezone

- TZ=America/Montreal

# Indentifiants, mots de passe de la base de données

- MYSQL\_RANDOM\_ROOT\_PASSWORD=true

- MYSQL\_PASSWORD=${DB\_PASSWORD}

- MYSQL\_DATABASE=${DB\_DATABASE}

- MYSQL\_USER=${DB\_USER}

nextcloud:

# Image nextcloud officielle

image: nextcloud

# Nom donné au conteneur

container\_name: nextcloud

# Dans le cas où le conteneur s'arrête/crash, il est automatiquement relancé

restart: unless-stopped

# Réseau dans lequel le conteneur va s'exécuter (dans le même réseau que la BD)

networks:

- cloud

# Attend que la base de données se lance avant de se lancer

depends\_on:

- nextclouddb

# Mappage de port (du port 80 du conteneur vers le port 8081 de la machine hôte)

# ports:

# - 8081:80

# Volumes du conteneur: mappe un répertoire du conteneur vers un répertoire de la machine hôte

volumes:

- ./html:/var/www/html

- ./custom\_apps:/var/www/html/custom\_apps

- ./config:/var/www/html/config

- ./data:/var/www/html/data

# Variables d'environnement passés au conteneur lors de son lancement

environment:

- PUID=1000 # The user ids. Most likely both should be 1000. Incorrectly setting these will led to file permission issues

- PGID=1000 # Set these to whatever your user is.

# Timezone

- TZ=America/Montreal

# Indentifiants, mots de passe de la base de données

- 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'

jellyfin:

# Image jellyfin officielle

image: jellyfin/jellyfin:latest

# Nom donné au conteneur

container\_name: jellyfin

# Dans le cas où le conteneur s'arrête/crash, il est automatiquement relancé

restart: unless-stopped

# Réseau dans lequel le conteneur va s'exécuter

networks:

- cloud

# Mappage de port (du port 8096 du conteneur vers le port 8096 de la machine hôte)

# ports:

# - 8096:8096

environment:

- TZ=America/Montreal

# Volumes du conteneur: mappe des répertoires du conteneur vers des répertoire de la machine hôte

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'

proxy:

image: traefik

container\_name: traefik

restart: unless-stopped

networks:

- cloud

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}"

# Définition des volumes des conteneurs (portainer et bd)

volumes:

portainer\_data:

nextclouddb:

jellyfin\_data:

# Création du réseau cloud (ou les conteneurs vont être lancés)

networks:

cloud:

driver: bridge

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

*Expliquez l’intégration de Traefik : Comment avez-vous configuré les règles de redirection pour chaque service ? Comment avez-vous implémenté le chiffrement SSL/TLS pour mettre en place le protocole HTTPS ?*

Jai mis des labels dans chaque service. Il y a une vérification pour voir si le nom de domaine est bien mien puis après on recevra le certificat quand tout sera correct et passeront tous les services en https.

## 5 - Nom de domaine

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

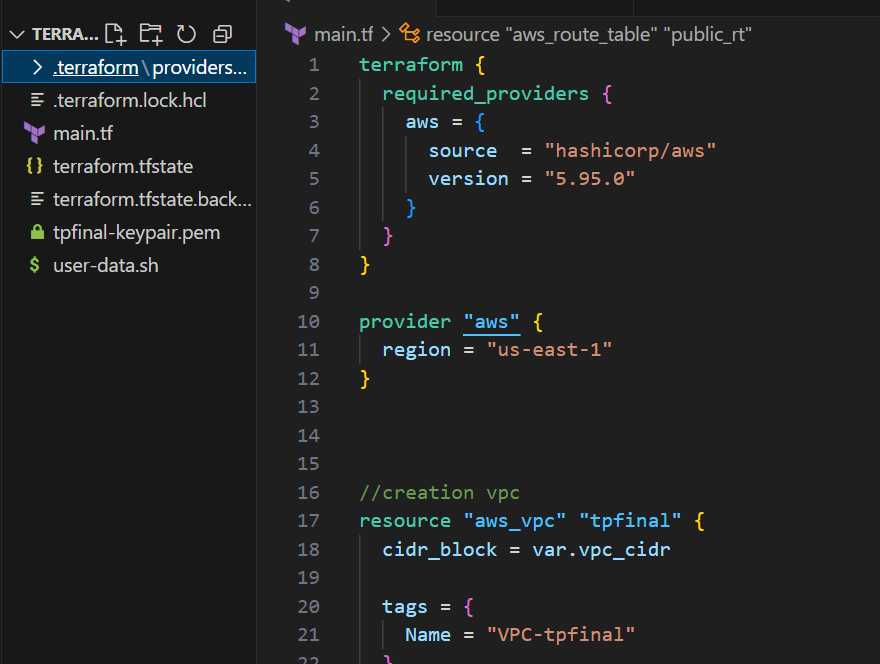
|  |  |
| --- | --- |
| ***Service*** | ***Adresse complète*** |
| Portainer | <https://portainer.takfatpfinal.duckdns.org> |
| Nextcloud | <https://nextcloud.takfatpfinal.duckdns.org> |
| Jellyfin | <https://jellyfin.takfatpfinal.duckdns.org> |

## 6 - Haute disponibilité et résilience

Redémarrage automatique des conteneurs avec Docker Compose (restart: unless-stopped).

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

J’ai utilisé Terraform pour automatiser le déploiement de l'infrastructure AWS. Ça rend la tâche moins longue car en quelques commandes tout se lance. C’est plus rapide, plus précis et stable.

**

## 8 - Problèmes rencontrés et solutions apportées

J’ai eu un problème quand je voulais docker compose up -d mes services. En effet, Nextcloud ne voulait pas se terminer et d’un coup ça me mettait le message d’erreur : no space left on device. J’ai donc cherché cette erreur sur google et je suis tombé sur plusieurs personnes offrant une solution similaire donc j’ai essayé et ça a marché. J’ai dû ajouter ce code dans mon main.tf pour mon instance :

 root\_block\_device {

    volume\_size = 64

    volume\_type = "gp3"

  }

Voilà le lien : https://stackoverflow.com/questions/67210801/aws-instance-changing-volume-size

## 9 - Conclusion

Ce projet a renforcé mes compétences en gestion d’infrastructure infonuagique, Docker et Terraform. J’ai aimé apprendre Terraform car ça facilite grandement la création d’un VPC et de tous ce qui s’en suit puisque ça sauve beaucoup de temps. Ça m’a aussi permis d’apprendre à utiliser un DNS pour des services ce que je pourrais faire dans le futur avec mes sites si je veux. En bref, ce travail final m’a beaucoup appris en infonuagique et m’a surtout aidé à perfectionner mes pratiques.