DOCUMENT TECHNIQUE

Document technique pour l’entreprise CNRT

Groupe N°2

Table des matières

[Introduction 2](#_Toc169573672)

[I. Configuration des Services Docker 3](#_Toc169573673)

[1. Création des Dockerfiles : 3](#_Toc169573674)

[2. Utilisation de Docker Compose : 4](#_Toc169573675)

[II. Développement du Service Web avec Django 5](#_Toc169573676)

[1. DockerFile pour le Service Web : 5](#_Toc169573677)

[2. Endpoints API : 5](#_Toc169573678)

[III. Configuration de la Base de Données PostgreSQL 6](#_Toc169573679)

[1. DockerFile pour le Service de la base de donnée : 6](#_Toc169573680)

[2. Configuration : 6](#_Toc169573681)

[IV. Configuration du Service MailHog 6](#_Toc169573682)

[1. Docker pour le Service de Mail : 6](#_Toc169573683)

[V. Configuration du Service DNS avec CoreDNS 7](#_Toc169573684)

[1. Docker pour le Service de DNS: 7](#_Toc169573685)

[VI. Configuration du Service Pare-feu avec Iptables 7](#_Toc169573686)

[1. Docker pour le Service de Pare-feu: 7](#_Toc169573687)

[2. Configuration des règles de iptables 8](#_Toc169573688)

[VII. Avis et recommandations sur les technologies choisies : 9](#_Toc169573689)

[1. Docker et Docker Compose 9](#_Toc169573690)

[2. Django (Service Web) 9](#_Toc169573691)

[3. PostgreSQL (Base de Données) 9](#_Toc169573692)

[4. MailHog (Service Mail) 9](#_Toc169573693)

[5. CoreDNS (Service DNS) 10](#_Toc169573694)

[6. Iptables (Service Pare-feu) 10](#_Toc169573695)

[7. Surveillance et Maintenance 10](#_Toc169573696)

[VIII. Intégration des Services et Tests 10](#_Toc169573697)

[1. Tests de Communication : 10](#_Toc169573698)

[2. Tests d'Intégration : 11](#_Toc169573699)

[IX. Déploiement et Maintenance 11](#_Toc169573700)

[1. Déploiement : 11](#_Toc169573701)

[2. Surveillance et Maintenance : 11](#_Toc169573702)

[Conclusion 11](#_Toc169573703)

# Introduction

Ce document technique détaille la mise en place et l'intégration des différents services pour le projet "AgenceConnect" de la Caisse Nationale des Retraités du Togo (CNRT). L'objectif est de créer une infrastructure technologique sécurisée et efficace pour moderniser la gestion des retraités.

# Configuration des Services Docker

Docker sera utilisé pour la containerisation des services, simplifiant ainsi le déploiement et la gestion des applications.

## 1. Création des Dockerfiles :

Chaque service (Service Web, Base de données, Service Mail, Service Pare-feu, Service DNS) aura son propre Dockerfile définissant les dépendances et les configurations spécifiques.

Voici la struture general du projet par reportoire :

project/

│

├── docker-compose.yml

│

├── web/

│ ├── Dockerfile

│ └── ... # Fichiers spécifiques à l’ application web

│

├── db/

│ ├── Dockerfile

│ └── ... # Scripts d'initialisation de la base de données

│

├── mail/

│ ├── Dockerfile

│ └── ... # Configurations spécifiques pour le serveur de messagerie

│

├── dns/

│ ├── Dockerfile

│ └── ... # Configurations spécifiques pour le service DNS (Bind9, par exemple)

│

└── firewall/

├── Dockerfile

└── iptables-rules/

└── ruleset.sh

## 2. Utilisation de Docker Compose :

Tous les services seront configurés dans un fichier `docker-compose.yml` pour faciliter le déploiement et la gestion des conteneurs.

Exemple de `docker-compose.yml` :

version: '3'

services:

web:

build: ./web

ports:

- "8000:8000"

depends\_on:

- db

- mail

db:

build: ./db

environment:

POSTGRES\_USER: user

POSTGRES\_PASSWORD: password

POSTGRES\_DB: agence\_db

mail:

build: ./mail

environment:

- MAIL\_HOST=smtp.example.com

- MAIL\_PORT=587

- MAIL\_USER=user

- MAIL\_PASSWORD=password

firewall:

build:

context: ./firewall

dockerfile: Dockerfile

volumes:

- ./firewall/iptables-rules:/iptables-rules

privileged: true # Nécessaire pour utiliser iptables dans un conteneur Docker

# Développement du Service Web avec Django

Django sera utilisé pour développer l'application web principale permettant la gestion des retraités et des opérations de pension.

## DockerFile pour le Service Web :

**Nom du fichier :** web/Dockerfile

# Utilisation de l'image Python officielle comme base

FROM python:3.9-slim

# Répertoire de travail dans le conteneur

WORKDIR /app

# Copie des fichiers nécessaires (assurez-vous d'avoir un .dockerignore approprié pour exclure les fichiers non nécessaires)

COPY requirements.txt .

# Installation des dépendances Python

RUN pip install --no-cache-dir -r requirements.txt

# Copie du reste de l'application

COPY . .

# Commande par défaut pour exécuter l'application Django

CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]

## 2. Endpoints API :

Rappelons que notre service web permet la création, la lecture, la mise à jour et la suppression (CRUD) des données des retraités et des opérations de pension et un formulaire de contact pour les messages.

# Configuration de la Base de Données PostgreSQL

PostgreSQL sera utilisé comme système de gestion de base de données pour stocker les informations des retraités et les détails des opérations.

## DockerFile pour le Service de la base de donnée :

**Nom du fichier :** db/Dockerfile

# Utilisation de l'image PostgreSQL officielle

FROM postgres:latest

# Optionnel : Définir des variables d'environnement pour la configuration PostgreSQL

ENV POSTGRES\_USER=user

ENV POSTGRES\_PASSWORD=password

ENV POSTGRES\_DB=agence\_db

# Copie de fichiers d'initialisation ou de configuration si nécessaire

# COPY init.sql /docker-entrypoint-initdb.d/

## Configuration :

Passons à la configuration des variables d'environnement pour définir le nom d'utilisateur, le mot de passe et les autres paramètres nécessaires à PostgreSQL.

# Configuration du Service MailHog

## Docker pour le Service de Mail :

**Nom du fichier :** mail/Dockerfile

# Utilisation de l'image MailHog officielle

FROM mailhog/mailhog:latest

# Optionnel : Exposer le port SMTP de MailHog (par défaut : 1025) et le port HTTP

(par défaut : 8025)

EXPOSE 1025 8025

# Commande par défaut pour démarrer MailHog

CMD ["MailHog"]

# Configuration du Service DNS avec CoreDNS

## Docker pour le Service de DNS:

**Nom du fichier :** dns/Dockerfile

# Utilisation de l'image CoreDNS officielle

FROM coredns/coredns:latest

# Optionnel : Copie des fichiers de configuration CoreDNS

# COPY Corefile /etc/coredns/

# Commande par défaut pour démarrer CoreDNS

CMD ["coredns"]

# Configuration du Service Pare-feu avec Iptables

## Docker pour le Service de Pare-feu:

**Nom du fichier :** firewall/Dockerfile

# Utilisation d'une image basée sur Ubuntu ou Debian

FROM ubuntu:latest

# Installation d'iptables

RUN apt-get update && apt-get install -y iptables

# Commande par défaut pour démarrer le conteneur

CMD ["bash"]

## 2. Configuration des règles de iptables

Une fois que vous avoir une image Docker avec iptables installé, nous pouvons monter un volume contenant les règles iptables personnalisées dans le conteneur.

**Exemple de règles iptables :** Création d’un fichier ruleset.sh dans firewall/iptables-rules/ avec vos règles iptables :

#!/bin/bash

# Effacer toutes les règles existantes

iptables -F

# Autoriser les connexions entrantes sur le port 8000 (exemple)

iptables -A INPUT -p tcp --dport 8000 -j ACCEPT

# Autoriser le trafic sortant

iptables -P OUTPUT ACCEPT

# Bloquer tout le trafic entrant par défaut

iptables -P INPUT DROP

# Autoriser les connexions établies

iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

# Avis et recommandations sur les technologies choisies

## Docker et Docker Compose

Avantages : Utilisation de Docker pour la containerisation des services est une décision judicieuse. Cela simplifie le déploiement, la gestion et la scalabilité des applications. Docker Compose permet de définir et de gérer facilement plusieurs conteneurs interconnectés à l'aide d'un fichier YAML unique.

Recommandation : Assurons-nous de bien gérer les versions des images Docker utilisées et de maintenir les Dockerfiles à jour pour inclure toutes les dépendances nécessaires.

## Django (Service Web)

Avantages : Django est un framework robuste et bien documenté pour le développement d'applications web. Il offre une sécurité intégrée et des fonctionnalités avancées pour la gestion des utilisateurs et des données.

Recommandation : Veillons à suivre les bonnes pratiques de développement Django, notamment en matière de sécurité et de performances.

## PostgreSQL (Base de Données)

Avantages : PostgreSQL est un système de gestion de base de données relationnelle fiable et performant. Il est connu pour sa conformité ACID et ses capacités avancées.

Recommandation : Assurons-nous de configurer correctement les paramètres de sécurité et de performance de PostgreSQL dans notre environnement Docker.

## MailHog (Service Mail)

Avantages : MailHog est utile pour le développement et les tests en capturant les e-mails sortants et en les affichant dans une interface web. Cela évite d'envoyer des e-mails réels pendant les phases de développement.

Recommandation : Configuration de MailHog pour qu'il soit facilement accessible aux développeurs et aux testeurs lors des tests d'intégration.

## CoreDNS (Service DNS)

Avantages : CoreDNS est un serveur DNS flexible et extensible, idéal pour gérer la résolution de noms de domaine dans des environnements Dockerisés.

Recommandation : Assurons-nous de bien configurer les zones DNS pertinentes pour répondre aux besoins spécifiques de votre application.

## Iptables (Service Pare-feu)

Avantages : Iptables est un outil puissant pour la gestion des règles de pare-feu sous Linux, offrant un contrôle fin sur le trafic réseau entrant et sortant.

Recommandation : Utilisation des scripts de configuration (comme les règles iptables) montés via des volumes Docker pour une gestion plus flexible et sécurisée.

## Surveillance et Maintenance

Avantages : La mise en place d'outils de surveillance pour les conteneurs Docker et les services est essentielle pour assurer la disponibilité et les performances de l'application.

Recommandation : Utilisation des outils comme Prometheus pour la surveillance des métriques et Grafana pour la visualisation, intégrés à votre architecture Docker.

# Intégration des Services et Tests

Assurons-nous que tous les services fonctionnent ensemble de manière sécurisée et efficace.

## 1. Tests de Communication :

Test de la communication entre tous les services (Service Web, Base de Données, Service Mail, Service DNS, Service Pare-feu) pour vérifier leur bon fonctionnement.

## 2. Tests d'Intégration :

Effectuons ensuite des tests d'intégration pour garantir que toutes les parties de l'application fonctionnent correctement ensemble.

# Déploiement et Maintenance

Déploiement de l'application sur notre infrastructure et assurons sa maintenance continue.

## 1. Déploiement :

Déploiement des conteneurs sur notre serveur Atlas.

## 2. Surveillance et Maintenance :

Metons en place des outils de surveillance pour surveiller la santé des conteneurs et des services.

Planifions des maintenances régulières pour appliquer les mises à jour de sécurité et optimiser les performances de l'infrastructure.

# Conclusion

Ce document technique fournit un guide détaillé pour la mise en place et l'intégration des services nécessaires à ce projet. En suivant ces étapes, nous allons assurer une infrastructure robuste et sécurisée pour la gestion moderne de nos retraités. Merci de respectez toutes les étapes decrtes et bon travail à tous !