Instructions for a Google compute engine installation

Ce guide détaille le processus de déploiement de l'application démo WASPL, packagée avec Docker Compose, sur une machine virtuelle (VM) Google Cloud Compute Engine. La stratégie employée utilise Google Cloud Artifact Registry pour stocker les images Docker pré-construites, évitant ainsi les problèmes de ressources lors de la construction des images directement sur la VM de destination.

Contexte : L'application WASPL est composée de plusieurs services (MongoDB, waspleditor, waspltestrunner, Nginx) orchestrés par Docker Compose. Le défi initial était la saturation des ressources de la VM lors de la phase docker build.

Stratégie adoptée :

1. Construire les images Docker dans un environnement performant (machine locale).
2. Stocker les images construites dans Google Cloud Artifact Registry.
3. Configurer la VM pour télécharger (pull) les images depuis Artifact Registry et lancer l'application avec Docker Compose.
4. Configurer le pare-feu GCP pour autoriser le trafic entrant.

Prérequis :

- Un compte Google Cloud Platform actif.
- Le SDK Google Cloud (gcloud CLI) installé et configuré localement (authentifié avec votre compte et projet par défaut défini).
- Docker Desktop (ou un environnement Docker fonctionnel) installé et lancé sur votre machine locale.
- Votre code source WASPL (incluant le docker-compose.yml, les Dockerfiles, et les fichiers .env) disponible sur votre machine locale, idéalement géré avec Git.
- Une instance Google Cloud Compute Engine (VM) créée et en cours d'exécution (par exemple, wiquid-waspl-machine), de préférence basée sur Debian ou Ubuntu.
- Accès SSH à cette VM (via gcloud compute ssh, Putty, ou la console web).

Phase 1 : Construction des Images Locales et Envoi vers Artifact Registry

Cette phase est réalisée sur votre **machine locale**.

1. **Assurer le fonctionnement de Docker local :**
   - Lancez Docker Desktop sur votre machine locale.
   - Ouvrez votre terminal ou invite de commande.
   - Vérifiez que Docker répond :

```
docker version
docker info
```

Si vous obtenez une erreur de connexion, assurez-vous que Docker Desktop est lancé et complètement démarré.

2. **Créer un dépôt Docker dans Google Cloud Artifact Registry :**
   - Cela se fait via la CLI gcloud. Choisissez une région ([GCP_REGION], ex: europe-west1) de préférence proche de la zone de votre VM ([VM_ZONE], ex: europe-southwest1-b).
   - Exécutez la commande (remplacez [YOUR_PROJECT_ID] par votre ID projet GCP, ex: gen-lang-client-0666666155932) :

```
gcloud artifacts repositories create waspl-demo-repo --repository-format=docker
--location=[GCP_REGION] --description="Docker images for WASPL demo"
```

Si l'API Artifact Registry n'est pas activée, gcloud vous le demandera. Répondez y et attendez qu'elle s'active et que la commande se relance.

3. **Configurer Docker pour s'authentifier auprès d'Artifact Registry :**
   - Cela permet à votre client Docker local de communiquer en toute sécurité avec votre dépôt GCP.
   - Exécutez la commande (remplacez [GCP_REGION]) :

```
gcloud auth configure-docker [GCP_REGION]-docker.pkg.dev
```

Confirmez si nécessaire (y).

4. **Construire vos images Docker localement :**
   - Naviguez dans le répertoire racine de votre projet WASPL sur votre machine locale.
   - Exécutez les commandes de build pour chaque service custom (remplacez ./waspleditor et ./waspltestrunner par les chemins corrects si besoin) :

```
docker build -t waspleditor-local ./waspleditor
docker build -t waspltestrunner-local ./waspltestrunner
```

Cette étape est gourmande en CPU/RAM/Disque, mais elle s'exécute sur votre machine locale.

5. **Taguer les images pour Artifact Registry :**
   - Donnez à vos images construites des noms qui incluent le chemin complet de votre dépôt Artifact Registry. Le format est [GCP_REGION]-docker.pkg.dev/[YOUR_PROJECT_ID]/[REPO_NAME]/[IMAGE_NAME]:[TAG].
   - Exécutez les commandes (remplacez les valeurs entre crochets) :

```
docker tag waspleditor-local [GCP_REGION]-
docker.pkg.dev/[YOUR_PROJECT_ID]/waspl-demo-repo/waspleditor:latest
docker tag waspltestrunner-local [GCP_REGION]-
```

```
docker.pkg.dev/[YOUR_PROJECT_ID]/waspl-demo-repo/waspltestrunner:latest
```

6. **Pousser les images vers Artifact Registry :**
   - Uploadez les images taguées vers votre dépôt GCP.
   - Exécutez les commandes (remplacez les valeurs entre crochets) :

     ```
     docker push [GCP_REGION]-docker.pkg.dev/[YOUR_PROJECT_ID]/waspl-demo-repo/wasplеditor:latest
     docker push [GCP_REGION]-docker.pkg.dev/[YOUR_PROJECT_ID]/waspl-demo-repo/waspltestrunner:latest
     ```

     Attendez que ces commandes se terminent. Vous verrez la progression de l'upload.

Phase 2 : Déploiement sur la VM Google Cloud

Cette phase est réalisée sur votre **VM Google Cloud** via une connexion SSH.

1. **Connectez-vous à votre VM :**
   - Utilisez votre méthode préférée (par exemple, gcloud compute ssh wiquid-waspl-machine --zone europe-southwest1-b).
2. **Installer Docker Engine et Docker Compose sur la VM :**
   - Si ce n'est pas déjà fait, installez les prérequis sur la VM.

     ```
     sudo apt-get update
     sudo apt-get install docker-ce docker-ce-cli containerd.io -y
     sudo curl -L "https://github.com/docker/compose/releases/download/v2.20.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose # Vérifiez la dernière version si v2.20.2 est ancienne
     sudo chmod +x /usr/local/bin/docker-compose
     # Pour éviter sudo avec docker (se déconnecter/reconnecter après)
     sudo usermod -aG docker $USER
     ```

     Après sudo usermod -aG docker $USER, déconnectez-vous de la session SSH et reconnectez-vous pour que le changement de groupe prenne effet.
3. **Assurer la présence des fichiers du projet sur la VM :**
   - Vérifiez que le dossier de votre projet (contenant le docker-compose.yml et les fichiers .env) est bien présent sur la VM (par exemple, via un clone Git précédent ou en utilisant gcloud compute scp si nécessaire).
   - Naviguez dans ce dossier :

     ```
     cd ~/WASPL-GVM/ # Ou le chemin de votre projet
     ```

4. **Modifier le fichier docker-compose.yml pour utiliser les images d'Artifact Registry :**

- Éditez le fichier docker-compose.yml sur la VM :

```
nano docker-compose.yml
```

- Pour les services waspleditor et waspltestrunner, **remplacez la ligne build: ... par une ligne image: ...** qui pointe vers les images dans Artifact Registry.

```
# Exemple pour waspleditor
waspleditor:
  # build: ./waspleditor # Commenter ou supprimer
  image: [GCP_REGION]-docker.pkg.dev/[YOUR_PROJECT_ID]/waspl-demo-
repo/waspleditor:latest # Ajouter
  env_file:
    - ./waspleditor/.env.docker
  # ... autres configurations ...

# Exemple pour waspltestrunner
waspltestrunner:
  # build: ... # Commenter ou supprimer la section build
  image: [GCP_REGION]-docker.pkg.dev/[YOUR_PROJECT_ID]/waspl-demo-
repo/waspltestrunner:latest # Ajouter
  env_file:
    - ./waspltestrunner/.env.production.docker
  # ... autres configurations ...
```

- Enregistrez et quittez l'éditeur.

5. **Configurer Docker sur la VM pour puller depuis Artifact Registry :**
   - Permettez au démon Docker sur la VM de télécharger des images depuis votre dépôt GCP.

     ```
     gcloud auth configure-docker [GCP_REGION]-docker.pkg.dev
     ```

     Confirmez si nécessaire (y). Le compte de service de la VM doit avoir les permissions nécessaires (rôle Artifact Registry Reader). L'option "Allow full access..." lors de la création de la VM est souvent suffisante pour une démo.
6. **Lancer l'application avec Docker Compose sur la VM :**
   - Assurez-vous d'être dans le répertoire de votre projet contenant le docker-compose.yml modifié.
   - Exécutez :

     ```
     docker-compose up -d
     ```

Docker Compose va maintenant télécharger les images depuis Artifact Registry et démarrer les conteneurs. Cela devrait être beaucoup moins gourmand en ressources que le build.

7. **Vérifier l'état des conteneurs :**
   - Après le démarrage, vérifiez que tout fonctionne :

     ```
     docker-compose ps
     docker-compose logs
     ```

   - Tous les services devraient être en état Up.

Phase 3 : Configuration du Pare-feu Google Cloud et Accès

Cette phase est réalisée dans la **console web Google Cloud**.

1. **Accéder aux règles de Pare-feu du réseau VPC :**
   - Dans le menu de gauche, naviguez vers **"Mise en réseau VPC" -> "Pare-feu"**.
2. **Vérifier ou créer une règle autorisant le trafic HTTP (port 80) :**
   - Recherchez une règle existante qui autorise le trafic entrant sur le port TCP 80 (par exemple, default-allow-http).
   - Si elle existe, notez le ou les "Tags de destination" qu'elle cible (par exemple, http-server).
   - Si elle n'existe pas ou ne convient pas, créez une nouvelle règle :
     - Cliquez sur "Créer une règle de pare-feu".
     - Configurez-la pour autoriser l'entrée (Ingress), le trafic TCP sur le port 80, depuis les plages d'adresses IP sources souhaitées (0.0.0.0/0 pour tout internet), et ciblez-la par un tag (ex: waspl-web-server).
3. **Appliquer le tag réseau à votre instance de VM :**
   - Naviguez vers "Compute Engine" -> "Instances de VM".
   - Cliquez sur le nom de votre VM (wiquid-waspl-machine).
   - Cliquez sur "Modifier".
   - Dans la section "Mise en réseau", trouvez "Tags réseau" et **ajoutez le tag exact** qui est ciblé par la règle de pare-feu qui autorise le port 80 (par exemple, http-server ou waspl-web-server).
   - Cliquez sur "Enregistrer".
4. **Trouver l'Adresse IP externe de la VM :**
   - Retournez à la liste des instances de VM.
   - Notez l'adresse IP listée dans la colonne "Adresse IP externe" pour votre VM.
5. **Accéder à votre site de démo :**
   - Ouvrez un navigateur web et entrez l'Adresse IP externe de la VM dans la barre d'adresse.

Votre application WASPL devrait maintenant être accessible publiquement.

Dépannage rapide :

- **Impossible de me connecter en SSH à la VM :** Vérifiez que la VM est démarrée, que le pare-feu GCP autorise le trafic TCP sur le port 22 (règle default-allow-ssh et tag ssh sur la VM), et que vos clés SSH sont correctement configurées (si vous utilisez gcloud compute ssh, il le gère).
- **docker build local échoue avec une erreur de connexion :** Docker Desktop n'est pas lancé ou ne fonctionne pas correctement sur votre machine locale.
- **docker-compose up -d sur la VM semble bloqué ou échoue :**
  - Vérifiez les logs (docker-compose logs).
  - Assurez-vous que vous avez bien remplacé build: par image: dans le docker-compose.yml sur la VM.
  - Vérifiez que Docker sur la VM est authentifié auprès d'Artifact Registry (gcloud auth configure-docker ...) et que le compte de service de la VM a les permissions de pull.
  - Si la VM est toujours surchargée (peu probable avec le pull), redémarrez-la depuis la console GCP.
- **Impossible d'accéder à l'application via l'IP externe :** Le problème est presque certainement le pare-feu GCP. Vérifiez que la règle autorisant TCP 80 est active et que le tag correspondant est appliqué à votre VM.

Étapes supplémentaires (pour une démo plus robuste) :

- **HTTPS :** Configurez Nginx pour utiliser un certificat SSL (par exemple, via Let's Encrypt en utilisant Certbot) et ouvrez le port 443 dans le pare-feu GCP (default-allow-https + tag https-server sur la VM).
- **Nom de domaine :** Utilisez un nom de domaine pointant vers l'IP externe de votre VM (via Google Cloud DNS ou un autre fournisseur).
- **Persistance MongoDB :** Pour éviter de perdre vos données si la VM est supprimée, utilisez un disque persistant Google Cloud séparé et montez-le sur la VM, puis configurez le volume de MongoDB dans docker-compose.yml pour pointer vers un sous-répertoire de ce point de montage.
- **Surveillance :** Utilisez Cloud Monitoring pour surveiller les ressources de la VM et les logs des applications.

Revision #3
Created 21 May 2025 07:36:43 by Janfix
Updated 21 May 2025 07:41:14 by Janfix