 S. Hérauvill	Langage Web 2	M1 GIL
	Persistance des données (MariaDB / PostgreSQL / MongoDB)	Mars 2021
		page 1 / 7

Objectifs

Si vous êtes ici, je suppose que les parties précédentes se bien déroulées, et que vous êtes prêtes & prêts à poursuivre l'aventure avec SPRING.

Un service REST doit pouvoir répondre à toutes les formes de requêtes (ou presque), et donc être doté de mémoire : Il faut pour cela, mettre en œuvre un mécanisme de persistance de données.

I - Création des services de base de données avec Docker

Je vous propose donc de mettre en place 3 services de base de données avec Docker. Bien évidemment, vous n'aurez besoin que d'un seul de ces services, mais c'est l'occasion de voir (si ce n'est déjà fait), 3 versions distinctes de bases de données représentées ici par : MariaDB, PostgreSQL et MongoDB.

Ce chapitre présente la mise en place et l'utilisation (de base) de chacun de ces services.

Remarques :

- Je vous renvoie au TP Docker si toutefois vous aviez besoin de vous rafraîchir la mémoire à ce sujet
- Docker permet ici d'installer très simplement ces services afin de les tester, sans modifier votre système
- Je suppose que vous disposez d'un service Docker installé et accessible pour la suite du TP, ou d'une machine virtuelle pour réaliser ces tests

II - MariaDB

On ne présente plus ce fork open-source de MySQL (fork créé juste avant la reprise de MySQL par Sun, avant de tomber dans le giron d'Oracle). Ces 2 outils, très similaires, forment la référence des sites web classiques, et sont tout à fait respectables quant à leurs performances, s'ils sont bien configurés et utilisés.

Cette première installation ne devrait pas vous poser de problème, vous êtes en terrain connu.

Remarque : Avant de poursuivre, vous devez désactiver une éventuelle version de mysql active sur votre système

II.1 - Création du conteneur

Suivons la démarche usuelle pour trouver notre conteneur :

- Recherchez une image docker de mariadb
- Comment peut-on identifier l'image officielle ?
Remarque : On peut aussi obtenir des informations depuis le site officiel de MariaDB !
- Chargez (pull) l'image nommée mariadb
- Analysons quand même le contenu de cette image avant de l'exécuter avec la commande :
`docker inspect mariadb`
En déduire le numéro de version, et le port ouvert par ce conteneur
- Après ces vérifications, nous pouvons démarrer notre conteneur :
`docker run --name mariadbtest -e MYSQL_ROOT_PASSWORD=mypass -d mariadb`
- Vérifiez que le conteneur est bien actif

II.2 - Utilisation de mariadb en mode local depuis le conteneur

- Vous pouvez maintenant travailler avec ce SGBD que vous maîtrisez parfaitement. Je vous guide pour la connexion, puis vous gérerez vous-même les opérations :
- Exécutez les commandes suivantes pour prendre la main sur MariaDB :
 - `docker exec -it mariadbtest bash`
 - `mysql -pmypass` *!! Utilisez le mot de passe défini précédemment*
- Et voila, vous disposez d'un serveur SGBD tout propre, dans lequel vous allez réaliser les opérations suivantes :

 S. Hérauvillle	Langage Web 2	M1 GIL
	Persistence des données (MariaDB / PostgreSQL / MongoDB)	Mars 2021
		page 2 / 7

- Affichez la liste des schémas (databases) disponibles
- Créez le schéma `cvtest`, puis connectez vous sur ce schéma
- Créez un table nommée `smallstb`, contenant les champs suivants :
 - `id` entier, clé primaire
 - `nom` chaîne de caractères de longueur variable (50 caractères max), et non nulle
 - `version` entier
 - `date` format date (on prendra la notation par défaut de la forme AAAA-MM-JJ)
 - `descr` chaîne de caractères de longueur variable (64 caractères max)
- Affichez la liste des tables
- Affichez le format de la table `smallcv`
- Insérez dans la table `smallcv`, les 2 enregistrements suivants (fournis ici au format csv) :
 - 1, TURING, 1234, 23/06/1912, Cryptanaliste ;
 - 2, JOHNSON, 534, 25/06/2003, projet SPRING ;
- Affichez le contenu de la table `smallcv`
- Vous allez maintenant créer un utilisateur permettant d'exploiter cette table :
 - Créez un nouvel utilisateur défini ainsi :
 - nom : adminspring
 - mot de passe : password
 - hôte : %
 - Attribuez à cet utilisateur, tous les privilèges sur le schéma `cvtest`
- Fermez votre session administrateur sur MariaDB, et connectez vous en utilisant les identifiants de votre nouvel utilisateur
- Vérifiez que vous pouvez afficher le contenu de la table créée précédemment

II.3 - Utilisation externe

Vérifions maintenant que votre utilisateur peut se connecter correctement depuis un hôte distant :

- Avant de se déconnecter, il est nécessaire d'apporter une modification à votre service MariaDB
- Faites une mise à jour système de votre conteneur, puis installez un éditeur de texte (nano par exemple)
- Éditez le fichier `/etc/mysql/mariadb.conf.d/60-galera.cnf`
- Recherchez la ligne `bind-address 0.0.0.0`, et supprimez le `#` du début de ligne
- Redémarrez le service MariaDB avec les commandes :
 - `mysqladmin -u root -p shutdown`
Remarque : Utilisez le mot de passe de votre utilisateur sur la machine hôte
 - Redémarrez le conteneur avec `docker start mariadbtest`
- Assurez vous que le conteneur est toujours actif
- Quelles informations utiles à la connexion sont fournies par les commandes suivantes :
 - `docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' mariadbtest`
 - `docker inspect mariadbtest | grep -i ipaddress`
 - `docker ps`
- Depuis votre système, connectez vous alors directement au service MariaDB, qui est exécuté dans le conteneur, avec la commande :
`mysql -h <@IP hôte mariadb> -u adminspring -p`
Remarque : le client `mariadb` (ou `mysql`) doit être installé sur votre système
- Vérifiez que vous pouvez visualiser le contenu de la table `smallstb`
- Arrêtez proprement votre conteneur, sans le supprimer !

 S. Hérauvillle	Langage Web 2	M1 GIL
	Persistence des données (MariaDB / PostgreSQL / MongoDB)	Mars 2021
		page 3 / 7

III - PostgreSQL

Nous allons reproduire strictement la même démarche que précédemment, afin d'utiliser cette fois PostgreSQL.

III.1 - Création conteneur

- Recherchez un conteneur permettant d'exécuter postgresql
- Chargez (pull) l'image officielle postgres
- Démarrez un conteneur en utilisant la commande `docker run -it postgres`
- Quel est le message obtenu ?
- En reprenant la même démarche que précédemment (§II.1), démarrez un conteneur nommé `postgresltest`.
- Vérifiez que ce conteneur est actif, et identifiez le numéro du port exposé
- Ouvrez une session sur ce conteneur, et récupérez les informations suivantes :
 - Quelle est la distribution utilisée et sa version ?
 - Quelle est la version du noyau Linux ?
 - Quelles informations sont fournies par les commandes :
 - `psql --version`
 - `psql --help`

III.2 - Utilisation PostgreSQL en local

Vous disposez maintenant d'un service SGBD basé sur PostgreSQL

Remarques :

- *N'hésitez pas à jeter un œil sur la documentation (officielle) de PostgreSQL afin de découvrir ce SGBD open-source, très performant, et offrant des fonctionnalités non disponibles dans MySQL / MariaDB*
- *PostgreSQL reconnaît la syntaxe SQL, néanmoins certaines commandes sont spécifiques, et vous seront alors fournies. N'hésitez pas cependant à consulter la documentation pour les approfondir.*


- Démarrez une session bash sur votre conteneur
- Connectez-vous au service docker, avec la commande :
`psql -h localhost -p 5432 -U postgres --password`

Remarque : Le mot de passe est celui que vous avez spécifié précédemment

- Quelles informations sont fournies par les commandes suivantes :
 - `\c postgres`
 - `create schema cvtest;`
 - `set search_path to cvtest;`
 - `\dn`
- Créez la table `cvtest.smallcv` en reprenant la même syntaxe que pour MariaDB
- Quelles informations obtenez-vous par les commandes suivantes :
 - `\dp`
 - `\d`
 - `\d smallcv`
- Comme précédemment (§I.1.2), insérez 2 enregistrements dans la table `smallcv` (remplacez `mariadb` par `postgresql` dans le champ description du premier enregistrement)
- Affichez le contenu de la table

Vous allez maintenant ajouter un nouvel utilisateur. Exécutez et commentez les commandes suivantes :

- `create user adminspring with password 'password';`
- `grant usage on schema cvtest to adminspring;`

 <i>S. Hérauville</i>	Langage Web 2	M1 GIL
	Persistance des données	Mars 2021
	(MariaDB / PostgreSQL / MongoDB)	page 4 / 7

- `grant all privileges on database cvtest to adminspring;`
- `grant select, insert, delete, update`
on all tables in schema cvtest to adminspring;
- `\du` ↔ `select * from pg_user;` → Liste des utilisateurs
- `\dp cvtest.*`
- `\q`

Vérifiez maintenant que votre utilisateur peut accéder à la table :

- `psql -h localhost -d cvtest -U adminspring` → mdp = password
- Affichez le contenu de la table `smallcv`
- Fermez la session PostgreSQL

III.3 - Utilisation externe

Comme précédemment, vous devez identifier l'adresse IP de votre conteneur, ainsi que le port d'accès au service PostgreSQL

- Pour se connecter au service, vous avez besoin d'installer le client sur votre système. Ceux-ci sont disponibles dans les packages `postgresql-client-common` & `postgresql-client`
- Connectez vous au service PostgreSQL en utilisant la même syntaxe suivante :
`psql -h <@IP_postgresql> -d <nom_base> -U <nom_user>`
- Fermez la session PostgreSQL
- Arrêtez proprement le conteneur sans le supprimer

IV - MongoDB

Attaquons nous maintenant à ce troisième gestion de base de données :

Pourquoi n'ai je pas utilisé l'acronyme SGBD concernant mongoDB ?

Avant de rentrer dans le vif du sujet, consultez rapidement la documentation pour comprendre la philosophie de MongoDB, autre système largement répandu en entreprise


IV.1 - Création conteneur

Vous connaissez maintenant la démarche :

- Recherchez un conteneur pour `mongodb`
- Chargez l'image (je vous aide → `mongo`)
- Analysez cette image
- Démarrez le conteneur en mode démon, sous le nom de `mongotest`
- Et voilà !

IV.2 - Utilisation de `mongodb` en local

- Démarrez une session `bash` sur votre conteneur
 - Comme précédemment, identifiez les versions de votre système
 - Distribution et version
 - Version du noyau
 - Version MongoDB
- Et maintenant nous allons répéter les mêmes opérations que précédemment. Exécutez, et commentez les commandes suivantes :
 - `mongo localhost`
 - `use cvtest`
 - `db`
 - `show dbs`
 - `db.smallcv.insert({ id:1, name:'TURING', version:1234, date: Date("1912-06-23"), descr:'Cryptanaliste'})` \

 <i>S. Hérauville</i>	Langage Web 2	M1 GIL
	Persistance des données	Mars 2021
	(MariaDB / PostgreSQL / MongoDB)	page 5 / 7

- `db.smallcv.insert({ id:2, name:'JOHNSON', version:534, date:Date("2003-06-25"), descr:'projet SPRING'})` \
- `show dbs`
- `show tables`
- `db.smallcv.find()`
- `db.createUser({ user : "adminspring", pwd : passwordPrompt(), roles : [{ role : "readWrite", db : "cvtest" }] })` \
- `show users`
- `exit`
- `mongo mongodb://localhost/cvtest --username adminspring --password`
- Affichez le contenu de la table smallcv
- Mettez fin à la session mongodB

IV.3 - Utilisation externe

- A nouveau, identifiez l'adresse IP et le port de connexion à votre conteneur mongotest
- Vous allez également avoir besoin du client mongo, disponible dans le package `mongodb-clients`
- Connectez vous au service mongodb à l'aide de l'utilisateur précédemment créé. Vous utiliserez la même commande en adaptant bien sûr les valeurs des paramètres de connexion.
- Mettez fin à la session mongodB
- Arrêtez proprement le conteneur sans le supprimer

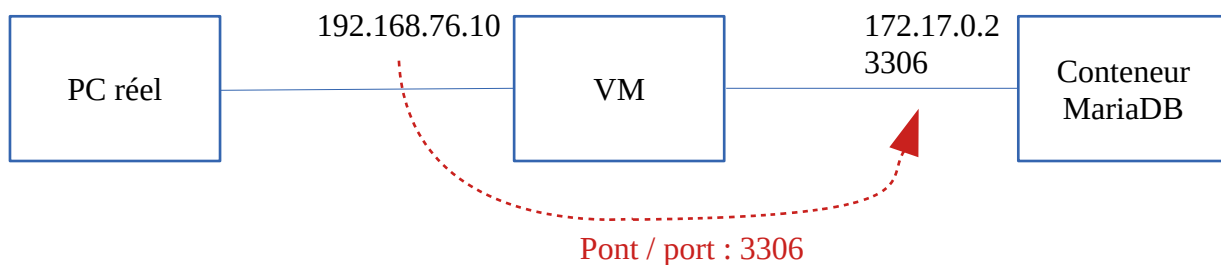
V - Intégration des machines virtuelles dans le projet

Vous disposez maintenant de 3 services prêts à l'emploi pour la gestion de notre `smallcv` qui sera utilisé dans notre service `SPRING REST`.

Si vous avez travaillé sur la même machine que celle utilisée pour le déploiement de `spring`, vous disposez d'un lien direct, mais si vos services sont hébergés sur une autre machine ou sur une machine virtuelle, nous avons une étape intermédiaire à résoudre.

En effet, l'adresse IP du conteneur `172.17.0.2` est celle fournie par l'interface du conteneur, disponible uniquement sur son hôte. Qu'en est-il si je souhaite accéder à ce service depuis une machine externe ?

Je vous propose de résoudre le cas de `MariaDB` (et de résoudre par la même occasion les 2 autres cas)
Nous disposons des informations suivantes :



Nous devons donc établir une connexion entre l'interface de la VM et le conteneur.

Remarque : Je suppose bien sûr que votre VM ne dispose pas d'un service `mysql` déjà installé, sinon il y aura conflit entre les 2 services. Il faudra alors modifier le numéro de port exposé sur la VM (utiliser `13306` par exemple au §1.4.2)

V.1 - Préparation

Commençons par récupérer les informations nécessaires depuis la VM

- `ip a` Récupérez l'adresse IP et le nom de l'interface de la VM
→ Notons ces valeurs `<@IP-VM>` `<intf-VM>`
- Depuis la machine réelle, faire un ping vers la vm pour valider cette adresse IP, et donc l'interface utilisée
- `docker start mariadbtest`
- `docker inspect mariadbtest | grep -i ipaddress`
Relevez l'adresse ip du conteneur `<@IP-cont>`
- `docker ps` Relevez le numéro de port exposé `<port-cont>`

V.2 - Création du pont

Mettons en place l'infra-structure qui effectuera un pont entre l'interface de la VM et le conteneur

- `docker run --rm --name pontdb -d -p <port-ext>:1234 \`
`verb/socat TCP-LISTEN:1234,fork TCP-CONNECT:<@ip-cont>:<port-cont>`

Remarques :

- `<port-ext>` est le port qui sera accessible depuis la MR. `<port-ext> = <port-cont>` par défaut
- Respectez scrupuleusement la syntaxe, notamment les espaces (pas d'espace avant `fork` par exemple)
- Cette commande va charger une nouvelle image qui servira à réaliser le pont

Nous sommes maintenant confronté à 2 cas :

1. Votre VM ne dispose que d'une seule interface réseau → Votre conteneur doit être accessible. Effectuez

 S. Hérauville	Langage Web 2	M1 GIL
	Persistance des données (MariaDB / PostgreSQL / MongoDB)	Mars 2021
		page 7 / 7

les tests suivants :

- Depuis la VM, connectez vous comme précédemment à votre service MariaDB pour valider le fonctionnement, en utilisant l'adresse IP du conteneur, et l'utilisateur adminspring
Vérifiez le bon fonctionnement puis fermez la session MariaDB
- Depuis la VM, relancez la connexion en utilisant cette fois l'adresse IP de votre VM
- Depuis la MR, réalisez une connexion en utilisant l'adresse IP de votre VM

Remarque : Si la connexion n'est pas opérationnelle, vous pouvez également ajouter l'étape suivante

2. Votre VM dispose de plusieurs interfaces réseau. Il est possible que la connexion ne se soit pas réalisée sur la bonne interface. Nous allons tenter d'y remédier :

- Comme précédemment, identifiez l'adresse IP de votre conteneur nommé pontdb
Cette valeur sera notée <@IP-pontdb>
- Dans votre VM, ajoutez la règle de filtrage suivante :

```
sudo iptables -A PREROUTING -t nat -i <intf-VM> -p tcp \
--dport <port-ext> -j DNAT --to <@IP-pontdb>:1234
```
- Refaites les tests : votre service est normalement opérationnel depuis la MR

Remarques :

- Si vous arrêtez votre VM, il sera nécessaire de répéter ces opérations
- Reproduire cette démarche également pour les autres services, en adaptant les paramètres