Un pipeline
Jee WebApp, CRUD / BDD

Objectif

Arrvier à monter une application Web Jee comprenant:

- un formulaire
- agissant sur une unique table de BDD (table "Abonnés", par exemple: 4 champs Prénom, Nom, Âge, et une clé primaire) . Une seule table dans la BDD.
- Le formulaire doit permettre les opértions CRUD sur les enregistrements de la table de la BDD.

BOM:

- Tomcat,
- mariaDB,
- client HeidiSQL https://www.heidisql.com/ (client graphique SQL)
- accès à l'hôte docker (donc la VM) par le dévelopeur, avec le fullstack-maven-plugin:
  - - Bastien:
  - username: "bastien-fullstack-mvn"
  - mot de passe: "mdp@b@sti1"
  - - Florian:
  - username: "florian-fullstack-mvn"
  - mot de passe: "mdp@flor1@n"
  - - Hamza:
  - username: "hamza-fullstack-mvn"
  - mot de passe: "mdp@6@mz@"
  - - Nicolas:
  - username: "nico-fullstack-mvn"
  - mot de passe: "mdp@n1col@s"
  - - Abraham:
  - username: "bra-fullstack-mvn"
  - mot de passe: "mdp@l1ncoln"
  - etc...
- accès "management" BDD avec le user "romanov/devientp@asvert"
- accès BDD par l'application Web Java Jee avec l'utilisateur:
  - username: "appli-de-lauriane"
  - mot de passe: "mdp@ppli-l@urian3"

Autres installations, sur l'hôte de virtualisation (la machine physique du poste de dev) :

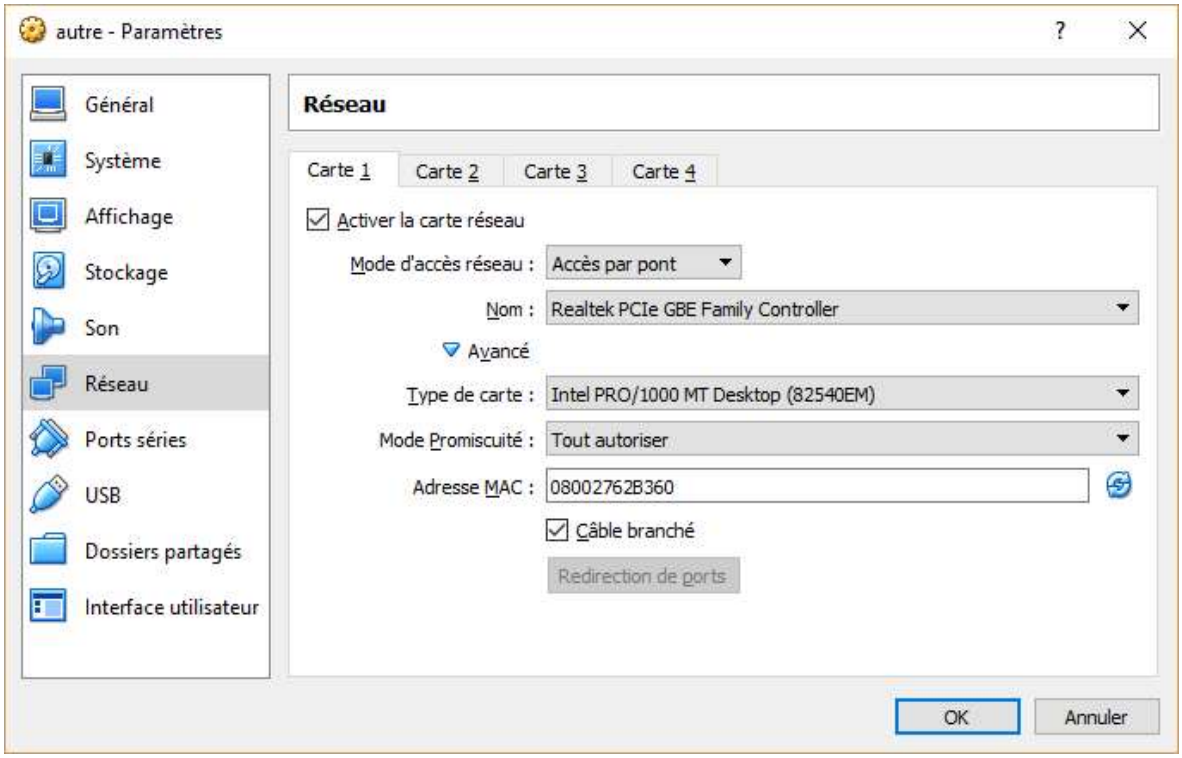
- eclipse
- HeidiSQL

Mode d'emploi:

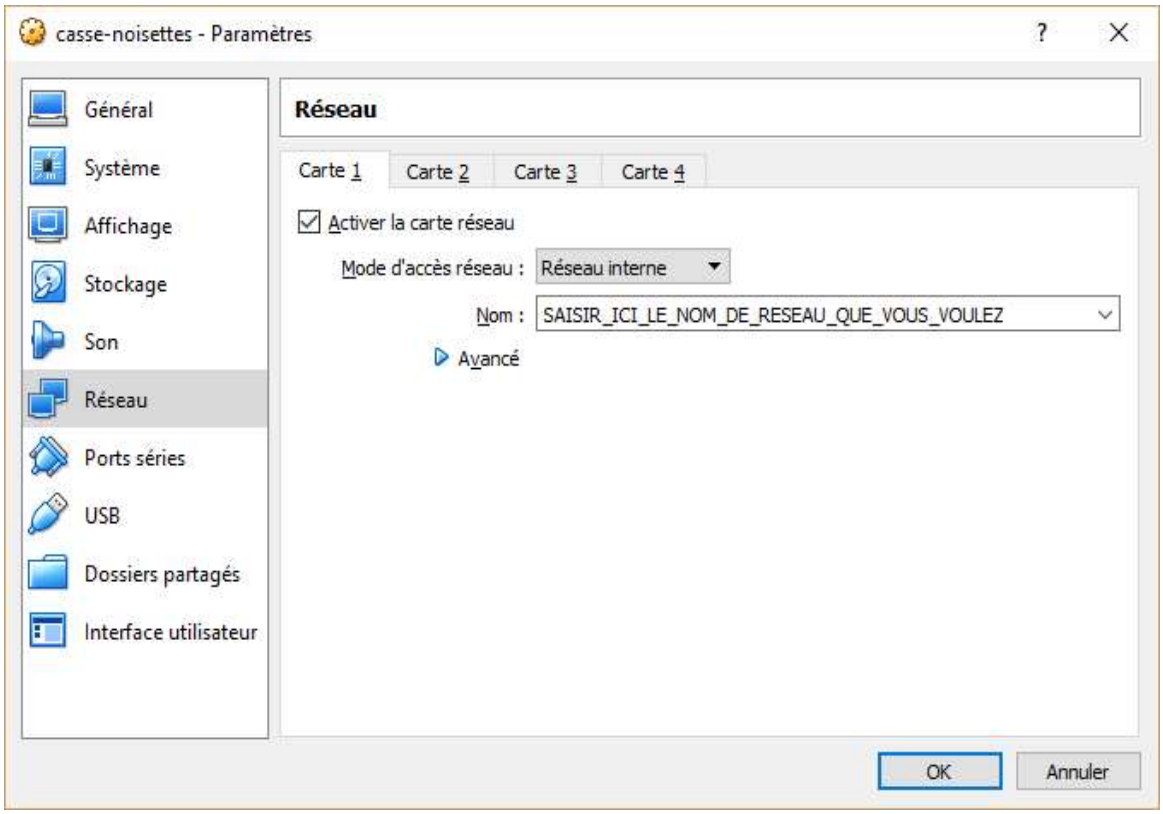
- Nous allons créer une nouvelle VM Ubuntu. Pour pouvoir réaliser l'ensemble des opérations suivantes, la VM Ubuntu que tu vas créer:
  - *doit* avoir accès à Internet.

    (ou à un réseau IP, et dans ce réseau pouvoir accéder par le protocole TCP/IP, à ce que l'on appelle des "repository" ("dépôt") valides pour le logiciel "apt-get", mais ceci est une autre histoire).

    Un test simple: si dans la VM, il est possible de faire un "sudo yum update -y" avec succès, vous savez que la configuration réseau permet l'accès à internet depuis la VM..
- Créez une nouvelle VM: vous y installerez donc CentOS (en date d'écriture de ce document, la dernière version majeure de CentOS est la version 7). Vous activerez 4 cartes réseau à la VM, dont:
  - 2 doivent être configurées avec un Mode d'accès réseau de type "Bridge Network" ("Accès par pont"),
  - Et 2 autres doivent être configurées avec un Mode d'accès réseau de type "Internal Network" ("Réseau interne"). Pour l'une des cartes, on saisira le nom "**RESAU_USINE_LOGICIELLE**", et pour l'autre , le nom "**RESEAU_OPS_USINE_LOGIGICIELLE**".

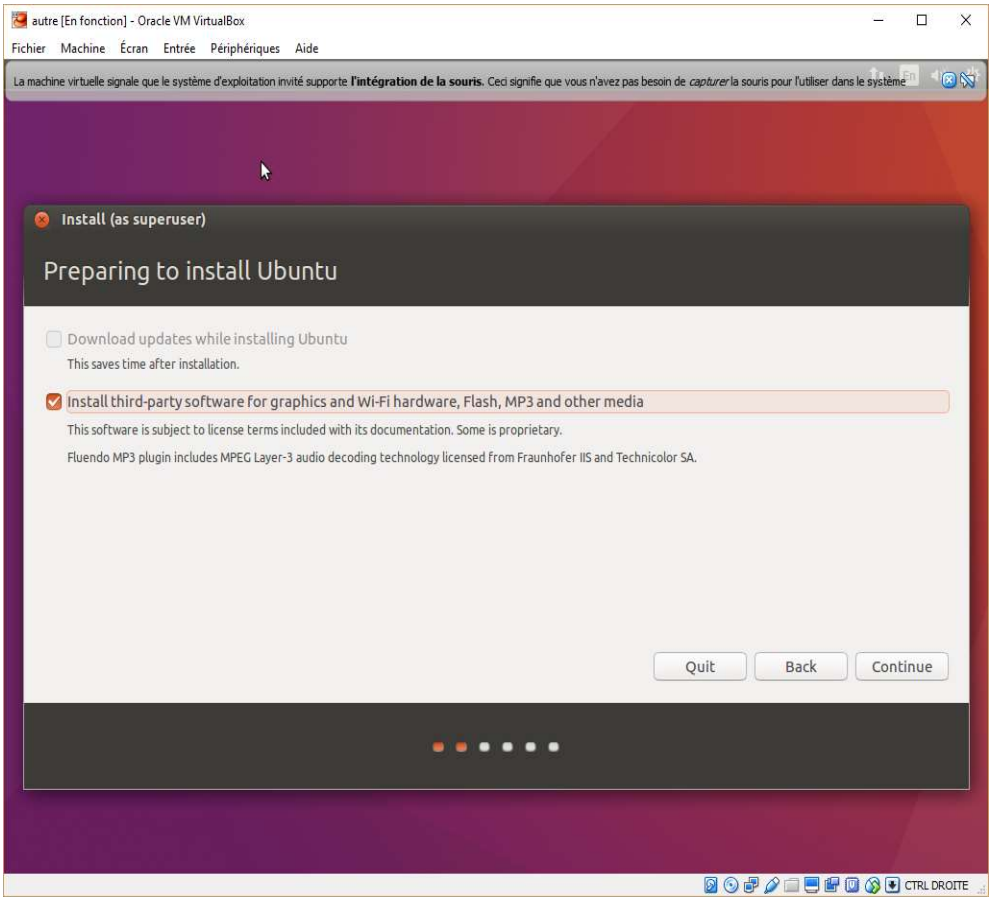


Ci-dessus, on voit que j'ai configuré le mode d'accès réseau "Accès par pont", pour l'une des cartes réseau virtuelles, d'une VM Virtual Box que j'ai créée



Ci-dessus, on voit que j'ai configuré le mode d'accès réseau "Réseau Interne", pour l'une des cartes réseau virtuelles, d'une VM Virtual Box que j'ai créée. On remarquera le champs "Nom", qui est libre de saisie. Pour que deux VM virutalbox soient dans le même "Internal Network", il suffira alors qu'elle aient chacune une carete réseau en mode "Réseau interne", avec l'exact même nom de réseau interne saisit dans le champs "Nom", ci-dessus visible.

- Téléchargez une image ISO d'installation de CentOS 7: ftp://ftp.free.fr/mirrors/ftp.centos.org/7/isos/x86_64/CentOS-7-x86_64-Minimal-1804.iso
- Si avec le PC/MAC avec lequel vous travaillez est connecté à internet via wifi, alors, pendant l'installation de l'OS, (ou après son installation), il faudra installer un pilote réseau de carte wifi, car alors vos cartes virtuelles réseau virutalisent un device physique qui est une carte de type Wifi. Consultez éventuellement votre archtiecte / lead dev pour cette question. Par exemple, pour Ubuntu, à l'étape suivante de l'installation:



il faudrait cocher l'option "Install third-party software for graphics and Wifi hardware [...]": lorsque cette option est cochée, un pilote ("driver") de carte WIFI sera installé, ce qui permettra à la VM de se connecter en Wifi.

- Une fois l'installation CentOS 7 terminée, vous pouvez suivre les instructions du README.md à la racine du repository https://github.com/Jean-Baptiste-Lasselle/predator-netboot-centos-7-host

- Une fois la configuration réseau de votre CentOS 7 terminée, vous pouvez suivre les instructions du README.md à la racine de ce repository, afin de provisionner un pipeline complet avec cible de déploiement tomcat/sgbdr (tomcat:mariadb, tomcat/postgresql)

ANNEXE I. Configuration d'un data source pour applications Jee

- ## Pour Tomcat

  - Pour le cas datasource au niveau du serveur:

- [docker cp dans le conteneur] vérifer la présence de **`$CATALINA_HOME/lib/tomcat-dbcp.jar`** , le déployer si nécessaire
- [docker cp dans le conteneur] déployer dans **`$CATALINE_HOME/lib`** le jar du driver JDBC
- [docker cp dans le conteneur] configuration à appliquer dans le [**`$TOMCAT_HOME/conf/context.xml`**] (à vérifier par META-INF/context.xml du projet):

vierge, ce fichier a pour contenu:

```
<?xml version='1.0' encoding='utf-8'?>
<!-- Licensed to the Apache Software Foundation (ASF) under one or more contributor
        license agreements. See the NOTICE file distributed with this work for additional
        information regarding copyright ownership. The ASF licenses this file to
        You under the Apache License, Version 2.0 (the "License"); you may not use
        this file except in compliance with the License. You may obtain a copy of
        the License at http://www.apache.org/licenses/LICENSE-2.0 Unless required
        by applicable law or agreed to in writing, software distributed under the
        License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
        OF ANY KIND, either express or implied. See the License for the specific
        language governing permissions and limitations under the License. -->
<!-- The contents of this file will be loaded for each web application -->
<Context>

        <!-- Default set of monitored resources. If one of these changes, the -->
        <!-- web application will be reloaded. -->
        <WatchedResource>WEB-INF/web.xml</WatchedResource>
        <WatchedResource>${catalina.base}/conf/web.xml</WatchedResource>

        <!-- Uncomment this to disable session persistence across Tomcat restarts -->
        <!-- <Manager pathname="" /> -->

        <!-- Uncomment this to enable Comet connection tacking (provides events
                on session expiration as well as webapp lifecycle) -->
        <!-- <Valve className="org.apache.catalina.valves.CometConnectionManagerValve"
                /> -->
</Context>
```

et on y ajoute la balise <Resource>:

```
<?xml version='1.0' encoding='utf-8'?>
<!-- Licensed to the Apache Software Foundation (ASF) under one or more contributor
        license agreements. See the NOTICE file distributed with this work for additional
        information regarding copyright ownership. The ASF licenses this file to
        You under the Apache License, Version 2.0 (the "License"); you may not use
        this file except in compliance with the License. You may obtain a copy of
        the License at http://www.apache.org/licenses/LICENSE-2.0 Unless required
        by applicable law or agreed to in writing, software distributed under the
        License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
        OF ANY KIND, either express or implied. See the License for the specific
        language governing permissions and limitations under the License. -->
<!-- The contents of this file will be loaded for each web application -->
<Context>

        <!-- Default set of monitored resources. If one of these changes, the -->
        <!-- web application will be reloaded. -->
        <WatchedResource>WEB-INF/web.xml</WatchedResource>
        <WatchedResource>${catalina.base}/conf/web.xml</WatchedResource>

        <!-- Uncomment this to disable session persistence across Tomcat restarts -->
        <!-- <Manager pathname="" /> -->

        <!-- Uncomment this to enable Comet connection tacking (provides events
                on session expiration as well as webapp lifecycle) -->
        <!-- <Valve className="org.apache.catalina.valves.CometConnectionManagerValve"
                /> -->
```

```
        <!-- Configuration du datasource -->

<Resource name="organisaction/SourceDeDonnees" auth="Container" type="javax.sql.DataSource"
                                                                maxActive="20"
                                                                maxIdle="10"
                                                                maxWait="10000"
                                                                username="lauriane"
                                                                password="lauriane"
                                                                driverClassName="org.mariadb.jdbc.Driver"
                                                                url="jdbc:mariadb://localhost:8456/bdd_oraganisaction"
                                                                />

</Context>
```

et pour finir [docker restart du conteneur]

[Donc il me faut comme paramètres supplémentaires du plugin]

Ci-dessous, une correspondance entre les balises d'un pom.xml utilisant le plugin, et les variables d'environneemnt utilsiées dans les scripts de déploiement du datasource:

- MARIADB_JDBC_DRIVER_CLASS_NAME=
  <jdbc-driver-classname>org.mariadb.jdbc.Driver</jdbc-driver-classname>
- JEE_DATASOURCE_NAME=
  <jee-datasource-name>organisaction/SourceDeDonnees</jee-datasource-name>
- JEE_DATASOURCE_AUTH_USERNAME=<lx-user>lauriane</lx-user>
- JEE_DATASOURCE_AUTH_USERPWD=<lx-pwd>lauriane</lx-pwd>
- JEE_DATASOURCE_MAX_ACTIVE=
  <jee-datasource-max-active>20</jee-datasource-max-active>
- JEE_DATASOURCE_MAX_IDLE=
  <jee-datasource-max-idle>10</jee-datasource-max-idle>
- JEE_DATASOURCE_MAX_WAIT=
  <jee-datasource-max-wait>10000</jee-datasource-max-wait>
- JEE_DATASOURCE_URL_PREFIX=
  <!-- ici il y a le numéro de port et adresse IP SGBDR-->
  <jee-datasource-url-prefix>jdbc:mariadb</jee-datasource-url-prefix>

- JEE_DATASOURCE_URL=
  <!-- Non, inutile dans la configuration, l'URL peut
       être formée à partir:
        ¤ de l'adresse IP utilisée par le SGBDR
        ¤ du numéro de port utilisé par le SGBDR
        ¤ de la valeur de $JEE_DATASOURCE_URL_PREFIX <jdbc-datasource-url-prefix>
        ¤ de la valeur de $NOM_BDD_APPLI <jdbc-datasource-url-prefix>

  -->
  <jee-datasource-url>jdbc:mariadb://192.168.1.149:3306/db</jee-datasource-url>

- dans [WEB-INF/web.xml], ajouter une référence au datasource configuré dans le [$TOMCAT_HOME/conf/context.xml] ccc

```
  <resource-ref>
      <description>DB Connection</description>
      <res-ref-name>jdbc/TestDB</res-ref-name>
      <res-type>javax.sql.DataSource</res-type>
      <res-auth>Container</res-auth>
  </resource-ref>
```

- Enfin, pour tester le datasource, il faut:

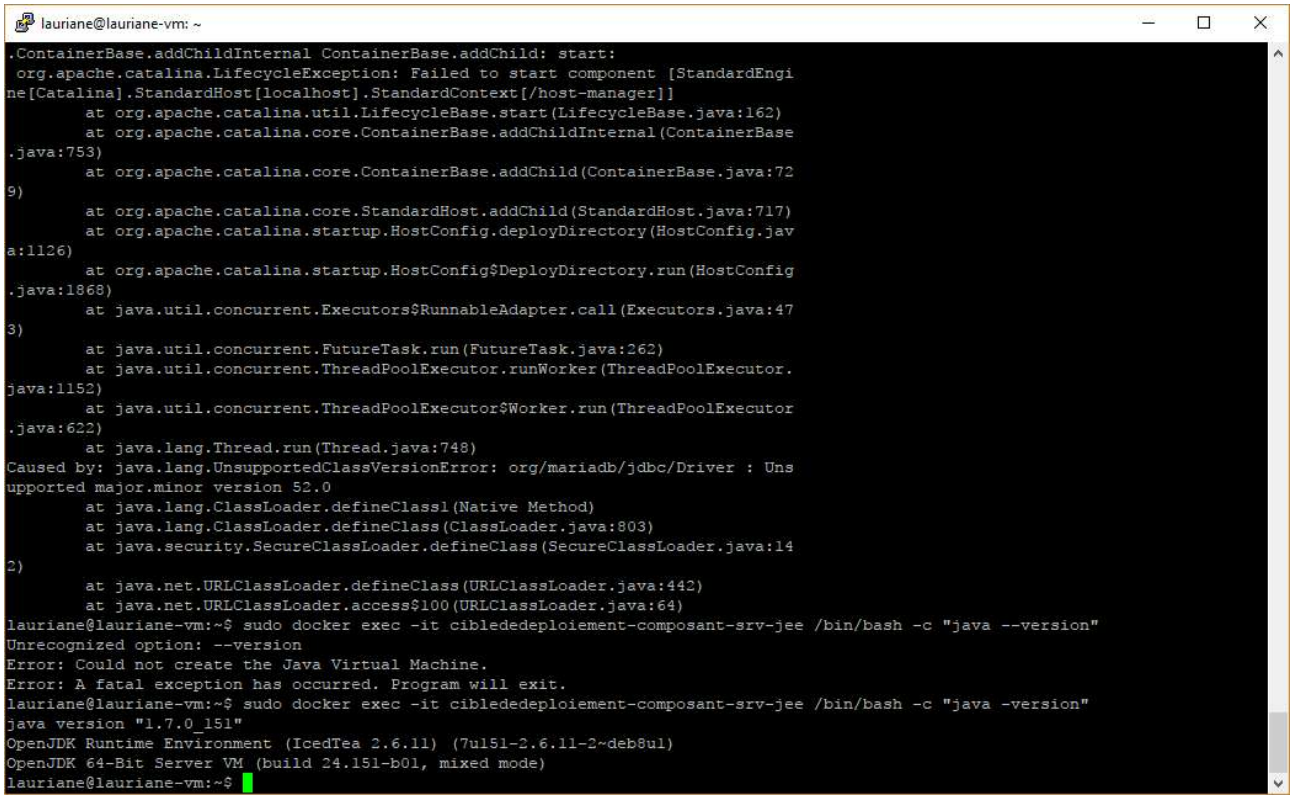
- créer une table de tests "MembresAssos" dans la BDD, et y metttre quelques enregistrement. Exemple de table:

MembresAssos
prenom
nom
username
email
age

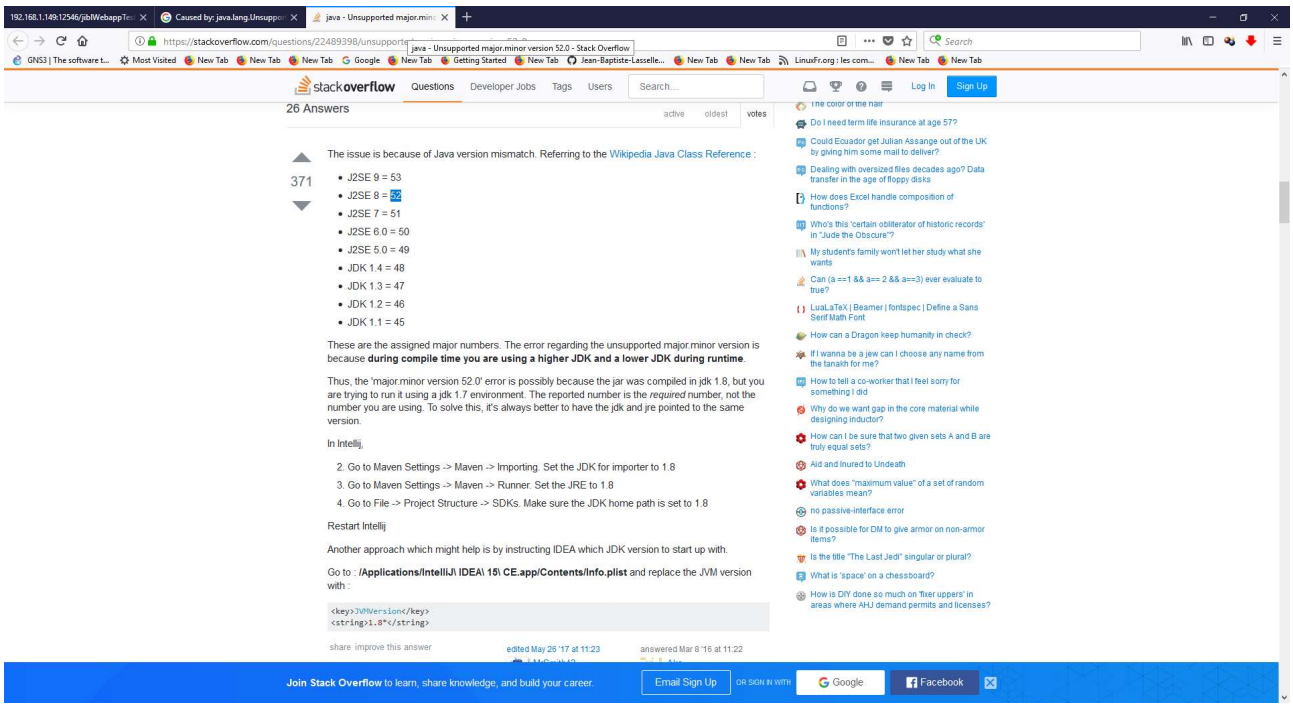
-
  - Pour le cas datasource au niveau de l'application:

// à préciser, mais:

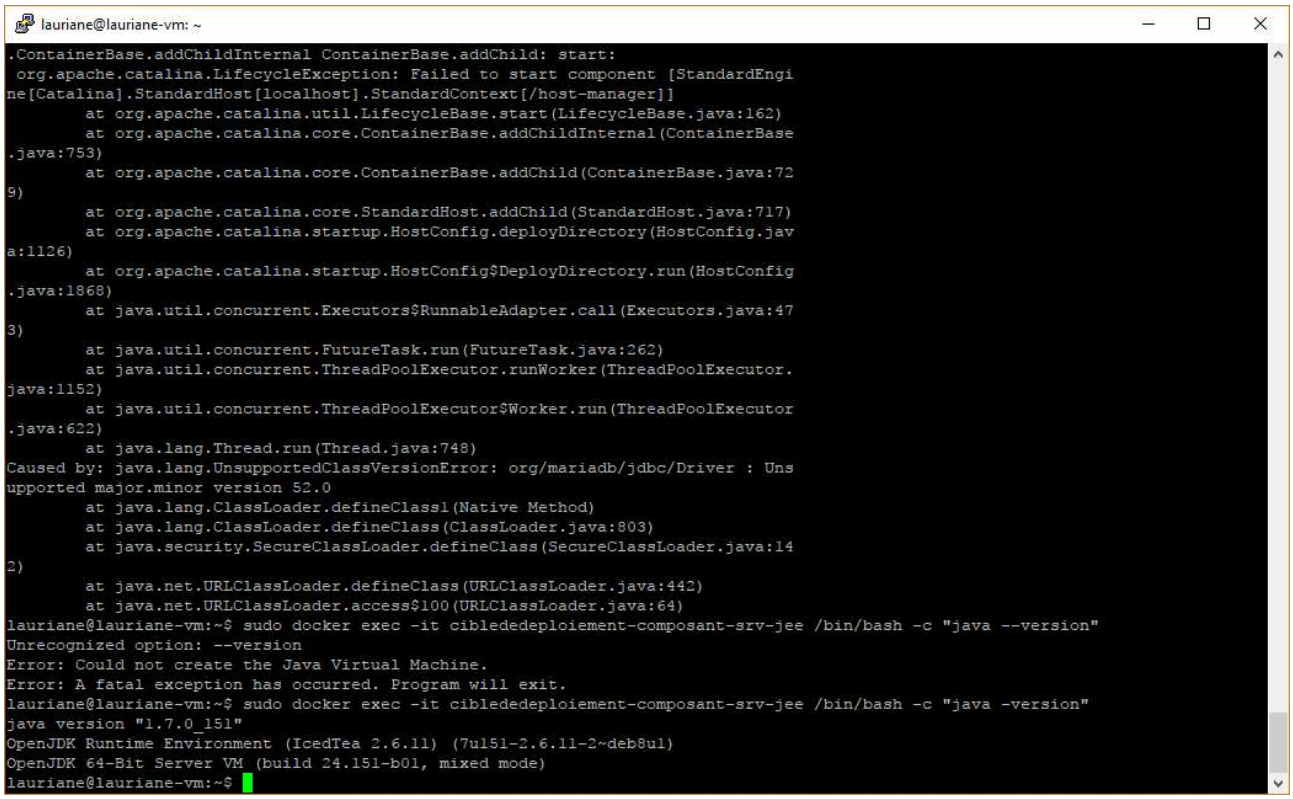
Donc j'ai quasiment finis d'automatiser le déploiment du datasource pour l'application
L'ereur sur laquelle je tombe, indiqe entre autre une erreur de compatibilité entre la version de JRE, et le driver MariaDB/JDBC:



Ci-dessus, ce sont les logs de tomcat dans le conteneur, et on voit une ligne :
"Caused by: java.lang.UnsupportedClassVersionError: org/maraidb/jdbc/Driver : Unsupported major.minor version 52.0"
Hors, le code 52 est associé au JDK 8 pour ce type d'exceptions natives Java, et elle est émise par la classe Pilote JDBC du pilote MariaDB/JDBC:

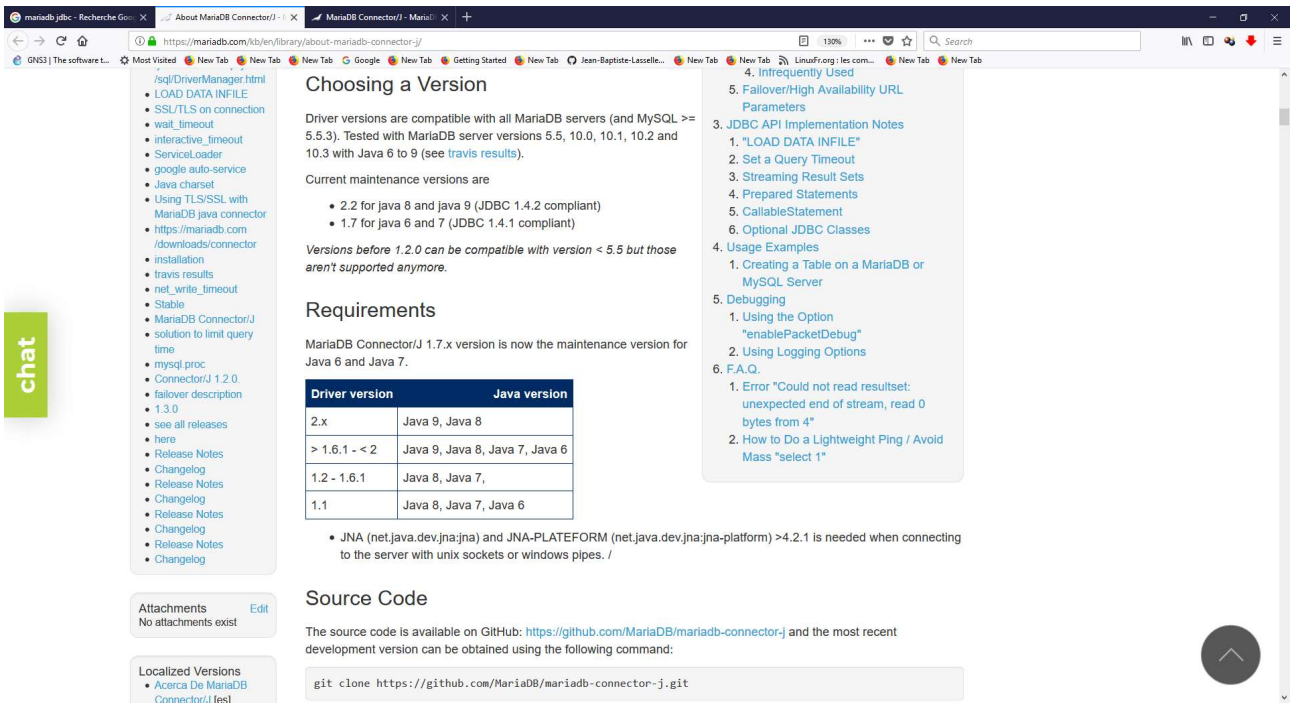


Bon, et de plus on voit que la versiond e java dans le conteneur qui edxécute tomcat est:



Donc il va falloir que je customise le dockerfile de la construction de tomcat. Dans ce dockerfile, je ferai l'installation de la JRE 8 au lieu de la JRE 7.

la dc officielle mariadb JDBC confirme l'incompatibilité (sachant que c'est la version 2.2.1 que je déployais pour le test) :



J'ai eut une autre déclinaison de mes problèmes d'incompatibilité avec la versiond e JRE dans le conteneur docker:
mes prores classes étaient compilées pour une cible JRE 8 (avec un source en Java 8), et ça re-levait une exception d'incompatibilité (et sans aucun log des exceptions de la webapp dans les logs serveurs):

J'ai testé ça en écrivant une simple Servlet faisant une réponse par une page HTML: en l'appelant directement avec son <url-pattern>, j'ai obtenu cette erreur d'incompatibilité entre la version de JRE, et la version de JDK que j'ai utilisé pour compiler mon code, avec les deux paramètres (source/target). Bref, j'ai changé la configuration du "maven-compiler-plugin", pour que mon code soit compilé en 1,7 en source, et en 1,7 en target, et le problème a été résolu, ma Servlet fonctionne (et permet de gérer les exceptions survenues dans l'application).

Arrivé là, j'ai modifié mon application pour qu'elle fasse un traitement des exceptions par config <error-page> dans le web.xml. Les exceptions sont attrapées par une Servlet, qui fait l'affichage d'autant d'infos que possible à propos de l'exception survenue. Une page jsp de test des exceptions permet de vérifier qu'une NullPointerException Levée, est bien traitée par ma servlet {@see `GestionnaireDexceptions` }. Tandis qu'aucune exception n'est attrapée lorsque j'invoque la page faisant usage de mon datasource configuré dans le web.xml

Oui, là, j'ai un problème de complexité pour mes tests:

version du driver jdbc /mariadb
version de tomcat
version de dbcp
version de Java dans le conteneur
version de Java en source et target de compilation (pom.xml)

fichier de configuration $CATALINA_BASE/conf/context.xml
fichier de configuration $CATALINA_BASE/conf/server.xml
fichier de configuration $MONPROJET/WEB-INF/context.xml

Si j'ai seulement 2 versions de chaque, j'ai un gros nombre de combinaisons à tester"

Je vais traiter ce problème plsu en profondeur, et en prenant en compte nos contraintes de projet en temps, à partir du tag: POINT_DE_RETOUR_COMPLEXITE_TESTS_TROP_LENTS

- **Pour Wildfly**