Pour réaliser ce TP, il est conseillé d'utiliser un environnement Linux avec Maven et Wildfly installés. Pour un système Ubuntu, vous pouvez installer Maven avec le gestionnaire de paquet :

```
sudo apt install maven
```

Vous pouvez installer Wildfly en le téléchargeant depuis le site officiel. Pour extraire Wildfly dans le répertoire bin de votre compte utilisateur :

```
mkdir -p ~/bin
cd ~/bin
wget http://download.jboss.org/wildfly/11.0.0.Final/wildfly-11.0.0.Final.tar.gz
tar -xzf wildfly-11.0.0.Final.tar.gz
```

Pour lancer le serveur, il suffit de lancer le script **standalone.sh** qui se trouve dans le répertoire bin du répertoire d'installation :

```
cd ~/bin/wildfly-11.0.0.Final/bin
./standalone.sh
```

Vous pouvez également réutiliser votre **playbook Ansible** de gestion de serveur wildfly. Il suffit d'ajouter une tâche pour s'assurer que Maven est installé grâce au module Ansible **package**.

Objectifs

- Pouvoir déployer avec une commande Maven une application Web Java EE dans un serveur Wildfly s'exécutant sur la même machine
- Pouvoir réaliser une nouvelle version de l'application avec une commande Maven.

Utilisation de Maven

Maven peut s'exécuter en ligne de commande grâce à la commande **mvn**. Pour cela, le répertoire courant doit contenir un projet Maven, c'est-à-dire le répertoire du projet contenant le fichier **pom.xml**.
On passe en ligne de commande la liste des objectifs (***goals***) que Maven doit réaliser. Par exemple pour compiler les sources :

```
mvn compile
```

Si on veut au préalable que Maven supprime les fichiers de travail (créés dans le répertoire ***target***) :

```
mvn clean compile
```

Les objectifs peuvent correspondre :

- soit à des phases dans le cycle de construction du projet. Dans ce cas, Maven exécute l'ensemble du cycle jusqu'à la phase indiquée. Par exemple, si on exécute :

```
mvn package
```

package correspond à la phase du cycle où Maven doit construire le livrable (par exemple le fichier war pour une application Web Java). Dans ce cas, Maven effectue toutes les phases précédentes (compilation, exécution des tests…) jusqu'à la phase ***package*** incluse.

- soit à un appel à un plugin Maven. Un plugin est identifié par son nom, suivi de : suivi de l'étape propre au plugin. Par exemple, il existe un plugin permettant de versionner son projet : le *maven-release-plugin*. Ce plugin propose ses propres étapes : *prepare* et *perform*. Pour les appeler avec Maven :

```
mvn release:prepare
mvn release:perform
```

Normalement, un plugin définit au moins l'étape ***help*** qui affiche une aide :

```
mvn release:help
```

En outre, chaque plugin dispose généralement de son propre site internet de documentation. Pensez à utiliser un moteur de recherche avec comme termes : « maven plugin » suivi du nom du plugin.

Pour connaître les différentes étapes du cycle de construction défini par Maven, reportez-vous à la documentation :
https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html#Lifecycle_Reference

Configurer des plugins pour un projet Maven

Maven fonctionne entièrement avec un système de plugins qui réalisent toutes les fonctionnalités et sont téléchargés automatiquement avant leur première utilisation. Même la compilation du code source Java est réalisée par un plugin dans Maven : le *maven-compiler-plugin*.
Si les plugins les plus courants sont connus par défaut de Maven, il est possible de préciser leur configuration ou d'en ajouter d'autres. La déclaration et la configuration se font dans le fichier **pom.xml** du projet. Pour cela on utilise la balise **build** qui contient la balise **plugins** qui contient la liste des plugins à configurer ou à ajouter pour le projet.

```
<build>
  <plugins>
    <plugin>
      <!-- Le groupId du plugin -->
      <groupId></groupId>
      <!-- Le nom du plugin -->
      <artifactId></artifactId>
      <!-- Le version du plugin -->
      <version></version>
      <!-- Le section execution est optionnelle. Elle permet de configurer
           une étape (goal) précise d'un plugin et éventuellement de rattacher
           ce plugin à une phase pour l'incorporer au cycle de construction
           du projet
      -->
      <executions>
        <execution>
          <!-- les étapes concernées -->
          <goals>
            <goal></goal>
```

```
          </goals>
          <!-- la phase au cours de laquelle l'exécution doit se faire
          (optionnelle) -->
          <phase></phase>
          <!-- Chaque plugin à des paramètres que l'on peut configurer ici
          pour une configuration relative à une exécution -->
          <configuration>
          </configuration>
        </execution>
      </executions>
      <!-- Chaque plugin à des paramètres que l'on peut configurer ici
           pour une configuration pour l'ensemble du plugin -->
      <configuration>
      </configuration>
    </plugin>
  </plugins>
</build>
```

Plugin Maven pour Wildfly

groupId : org.wildfly.plugins
packageId : wildfly-maven-plugin
version : 1.2.1.Final
documentation en ligne : https://docs.jboss.org/wildfly/plugins/maven/latest/

Le plugin Maven pour Wildfly permet, entre autres, de déployer l'application Web créée dans une instance d'un serveur Wildfly en cours d'exécution.
Reportez-vous à la documentation : https://docs.jboss.org/wildfly/plugins/maven/latest/deploy-mojo.html
Configurez le projet Maven de manière à ce que le déploiement soit réalisé sur une phase bien précise de construction du projet. Vous devez choisir la phase qui vous semble la plus pertinente. **Attention**, la phase ***deploy*** est un faux-ami : cette phase est réservée au déploiement des artefacts dans les dépôts Maven.

Dans le cadre de ce TP, nous nous limiterons à déployer une application sur un serveur Wildfly s'exécutant sur la même machine. Pour des raisons de sécurité, le serveur Wildfly n'autorise pas par défaut le déploiement depuis une machine distante.

Plugin Maven pour le versionnage

groupId : org.apache.maven.plugins
packageId : maven-release-plugin
version : 2.5.3
documentation en ligne : https://maven.apache.org/maven-release/maven-release-plugin/

Le plugin de versionnage est un plugin simple d'usage mais assez complexe à appréhender du fait de l'ensemble des opérations qu'il réalise. Il permet de créer une nouvelle version d'un projet pour une livraison par exemple. Son usage se découpe en deux étapes :

La préparation

À cette étape, le plugin demande des informations sur le numéro de version à réaliser et sur le prochain numéro de version pour la suite du développement. Ensuite, il vérifie que le projet se construit correctement. Il exécute pour cela toutes les étapes de construction du projet jusqu'à l'étape ***verify***. Enfin, il crée un libellé (***tag***) de version dans le dépôt des sources du projet (par exemple un dépôt Git).
Pour exécuter cette étape de préparation, il faut utiliser la commande :

```
mvn release:prepare
```

La réalisation

À cette étape, le plugin récupère l'intégralité du projet depuis le dépôt des sources correspondant au libellé (***tag***) créé à l'étape précédente. Et il exécute pour cela toutes les étapes de construction du projet jusqu'à l'étape ***deploy***.
Pour exécuter cette étape de réalisation, il faut utiliser la commande :

```
mvn release:perform
```

Pour que le plugin fonctionne correctement, il faut que le fichier ***pom.xml*** contienne la balise **scm** (*source control management*) qui indique l'URL du dépôt des sources du projet. Pour un projet dans un dépôt Git, il sera de la forme :

```
<scm>
  <developerConnection>scm:git:[URL du dépôt Git]</developerConnection>
</scm>
```

Attention, un appel à ***release:perform*** réalise toutes les étapes du cycle de construction du projet jusqu'à ***deploy*** (inclus). Pour Maven, ***deploy*** signifie envoyer le projet construit dans un dépôt Maven de manière à pouvoir le partager avec d'autres projets. Si cela est très utile quand on crée une bibliothèque Java que l'on veut réutiliser dans d'autres projets, cela a moins de sens pour une application Web qui doit être déployée dans un serveur d'application. De plus, vous ne disposez pas des droits nécessaires pour stocker le résultat de votre projet dans les dépôts officiels de Maven. Donc il va falloir reconfigurer le plugin de déploiement afin qu'il s'arrête à la phase qui précède la phase ***deploy***. Sinon votre versionnage se terminera systématiquement en erreur.

Le rollback

Le plugin de versionnage permet d'effectuer un rollback, c'est-à-dire d'annuler la phase de préparation. Cela est très pratique si la phase de réalisation échoue. Le rollback supprime notamment le libellé (***tag***) de version ajouté dans le dépôt de source.

```
mvn release:rollback
```

Livrable attendu

Vous devez fournir une URL vers un dépôt Git contenant un projet Maven d'une application Web Java EE. Il doit être possible de réaliser avec succès la commandes Maven suivante sur le projet :

```
mvn release:prepare release:perform
```

Le résultat de cette commande doit créer une nouvelle version du projet (ainsi qu'un libellé dans le dépôt Git) et l'application doit être déployée dans un serveur Wildfly s'exécutant sur la même machine.