Utiliser et développer des Phases personnalisées dans IKAN ALM

Version 5.7

Mai 2019

IKAN Development N.V.
Kardinaal Mercierplein 2
2800 Mechelen
BELGIUM

Table des matières

CHAPITRE 1
Introduction

Les Phases personnalisées constituent l'innovation la plus importante de la version 5.5 d'IKAN ALM. Dans la version 5.6 elles ont encore été optimisée (les fonctionnalités de remplacement et de suppression en masse, ...).

En créant des Phases personnalisées, les Utilisateurs peuvent considérablement personnaliser le flux de travail de leurs projets, en utilisant des blocs d'exécution réutilisables. Les Utilisateurs peuvent créer une Phase, charger leurs scripts (Ant, Gradle, NAnt, ...) dans cette Phase, définir les paramètres de Phase utilisés dans leurs scripts et, finalement, utiliser cette Phase dans leurs projets. En utilisant les fonctionnalités d'import/export, les Phases peuvent non seulement être partagées par des projets différents, mais également par des installations IKAN ALM différentes.

Les avantages de créer une Phase pour exécuter une tâche au lieu d'exécuter un énorme script monolithique sont les suivants:

- Réutilisation

  Les Phases peuvent être partagées par des projets, ainsi que par des installations IKAN ALM.

- Journaux améliorés

  En utilisant plusieurs petites Phases, le flux de travail est divisé en fractions plus petites. Cela facilite la compréhension et le contrôle du flux de travail, car, il sera beaucoup plus facile de détecter exactement quel étape dans le flux de travail a échoué.

- Contrôle de Versions

  Une Phase est identifiée par une combinaison unique: nom/version. Si les scripts contenus dans une Phase sont modifiés, le numéro de version changera également. Cela permet aux Utilisateurs de vérifier quelle version d'un script ils sont en train d'exécuter, et leur permettra également d'utiliser des versions différentes d'une même Phase dans une seule installation IKAN ALM.

- Maintenance

  En utilisant plusieurs phases courtes à but restreint, les scripts utilisés auront tendance à être plus petits, ce qui les rend plus simples et plus faciles à maintenir.

- Gestion de paramètres améliorée

  En déclarant formellement les paramètres auxquels réagissent les scripts d'une Phase, il est plus facile d'établir les paramètres requis. En plus, IKAN ALM contient une fonctionnalité pour l'édition en masse des paramètres.

Ce document vise surtout à expliquer comment travailler avec des Phases personnalisées dans IKAN ALM.

D'abord, nous décrirons le concept de Phases en expliquant la configuration par défaut des Niveaux et des Environnements avec les Phases de Noyau, et comment cette configuration peut être améliorée en introduisant des Phases personnalisées spécifiques.

Après, nous expliquerons comment vous pouvez créer et développer une simple Phase personnalisée dans l'Administration globale, comment vous pouvez appliquer une telle Phase dans un Niveau ou un Environnement et comment cette Phase agira sur ce Niveau ou cet Environnement.

Finalement, nous parlerons du Cycle de vie d'une Phase et des meilleurs pratiques à suivre lorsque vous développez ou utilisez différentes versions d'une même Phase.

CHAPITRE 2
Le concept de Phases

Quand IKAN ALM exécute des Requêtes de Niveau, des Constructions et des Déploiements, toutes les actions sont effectuées en exécutant une séquence de Phases. Ces Phases sont définies dans la base de données d'IKAN ALM et peuvent être consultées et modifiées dans la section Phases dans le contexte de l'Administration globale. Une fois définies dans l'Administration globale, les Phases peuvent être associées à des Niveaux, à l'Environnement de construction ou aux Environnements de déploiement dans le contexte de l'Administration des projets.

2.1. Phases de Noyau

Les Phases de "Noyau" IKAN ALM forment la fonctionnalité de base de la gestion du cycle de vie d'une application, telles que la récupération et le balisage de code à partir d'un système de Contrôle de Versions, le transport de ressources et la construction des objets à partir de et vers les Agents locaux et à distance, l'intégration avec les systèmes de Suivi des Incidents, etc. Elles peuvent être affichées, mais pas modifiées ni supprimées. Vous devez les considérer comme faisant partie intégrante d'IKAN ALM. Toutes les versions d'IKAN ALM inférieures à la version 5.5 ne supportent que ces Phases de Noyau.

Initialement, après avoir installé une version IKAN ALM originale (dite "vanilla"), seules les Phases de Noyau sont disponibles dans le Catalogue des Phases. Elles peuvent être vérifiées via *Administration globale* > *Phases* > *Aperçu*:

Aperçu des Phases

Nom	Version	Nom d'affichage	Auteur	Type d'exécution	Noyau	Certifiée	Publiée	N	C	D
com.ikanalm.phases.core.build.transport.packageresults	5.7.0	Transport Package Results	IKAN	BASE	✓				✓	
com.ikanalm.phases.core.scripting.scriptingPhase	5.7.0	Execute Script	IKAN	BASE	✓			✓	✓	✓
com.ikanalm.phases.core.level.link.filerevisions	5.7.0	Link File Revisions	IKAN	BASE	✓			✓		
com.ikanalm.phases.core.level.issue.tracking	5.7.0	Issue Tracking	IKAN	BASE	✓			✓		
com.ikanalm.phases.core.level.cleanup	5.7.0	Cleanup Work Copy	IKAN	BASE	✓			✓		
com.ikanalm.phases.core.level.deploy	5.7.0	Deploy	IKAN	BASE	✓			✓		
com.ikanalm.phases.core.level.tag	5.7.0	Tag Code	IKAN	BASE	✓			✓		
com.ikanalm.phases.core.level.build	5.7.0	Build	IKAN	BASE	✓			✓		
com.ikanalm.phases.core.level.retrieve.source	5.7.0	Retrieve Code	IKAN	BASE	✓			✓		
com.ikanalm.phases.core.deploy.cleanup.buildfiles	5.7.0	Cleanup Build Result	IKAN	BASE	✓					✓

20 occurences trouvées, affichées de 1 à 10

1 2

2.2. Les Phases de Noyau dans le flux de travail par défaut d'un Niveau de construction et d'un Environnement de construction

Pour vous donner un exemple, nous avons configuré un Projet Web auquel est associé un Cycle de vie appelé BASE, qui contient un Niveau de construction et un Niveau de test. Le Niveau de construction contient un Environnement de construction sur lequel est construit le Projet. Après avoir créé ce Niveau, nous pouvons afficher le flux de travail par défaut en sélectionnant *Éditer les Phases* dans la définition du Niveau.

Comme mentionné auparavant, les Phases de Noyau exécutent les actions de base pour un Niveau de construction sur le serveur IKAN ALM: récupérer les sources à partir du système de contrôle des versions, démarrer et faire le suivi de la Construction sur l'Environnement de construction, baliser les sources dans le système de contrôle des versions, vérifier les commentaires de validation pour les Incidents et les associer à la Requête de niveau, et nettoyer l'Emplacement des Copies de travail.

Note: Nous avons optimisé le flux de travail en retirant la Phase de Déploiement vu qu'il n'y a pas d'Environnement de déploiement.

Maintenant, examinons le flux de travail de l'Environnement de construction (en sélectionnant *Éditer les Phases* dans la définition de l'Environnement de Construction):

Ici aussi les différentes Phase de Noyau exécutent les actions de base, mais cette fois-ci sur l'Environnement de construction qui est exécuté sur l'Agent IKAN ALM et qui peut être sur une autre machine que celle du Serveur IKAN ALM.

Le travail le plus important est effectué par la Phase de noyau: Exécution du script. D'abord il peut y avoir un processus de compilation, suivi de tests d'unité et d'une mise en Paquet des sources compilées. Pour les projets plus larges, il peut y avoir plus de tâches, telles que la vérification du code, le débogage, la génération de documentation, etc. Au bout du compte, dans un tel cas, vous pouvez vous retrouver avec un script de construction très complexe qui est difficile à entretenir, qui ne peut pas être réutilisé dans d'autres projets, qui est dirigé par un grand nombre de paramètres de construction et de machine, et pour lequel il est très difficile d'analyser le journal et retrouver la cause exacte d'un échec dans le cas d'une construction échouée.

2.3. Améliorer le flux de travail du Niveau de Construction et de l'Environnement de Construction en utilisant des Phases personnalisées


Aperçu des Phases

Nom	Version	Nom d'affichage	Auteur	Type d'exécution	Noyau	Certifiée	Publiée	N	C	D
com.ikanalm.compilesources	1.0.0	Compile Sources	ikanhowto	ANT					✓	
com.ikanalm.unittests	1.0.0	Unit Tests	ikanhowto	ANT					✓	
com.ikanalm.packagecode	1.0.0	Package Code	ikanhowto	ANT					✓	

3 occurences trouvées et affichées

Dans cet exemple, nous avons créé nos propres Phases personnalisées dans le Catalogue des Phases IKAN ALM. Il y une Phase qui compile les Sources, une autre qui exécute les tests d'unité et une troisième qui met en paquet le code compilé dans une entité à déployer (par exemple, un fichier war, exe ou dll).

Dans le chapitre suivant, nous décrirons comment vous pouvez créer ces Phases, mais d'abord nous expliquerons comment les appliquer.


Aperçu des Phases

Nom de Phase	Version de la Phase	Abandon si erreur	Phase suivante si erreur
Transfert des Sources	5.7.0	Oui	Nettoyage Emplacement Source
Vérification du script de construction	5.7.0	Oui	Nettoyage Emplacement Source
Compilation des Sources	1.0.0	Oui	Nettoyage Emplacement Source
Tests d'unité	1.0.0	Oui	Mise en paquet du code compilé
Mise en paquet du code compilé	1.0.0	Oui	Nettoyage Emplacement Source
Compression de la construction	5.7.0	Oui	Nettoyage Emplacement Source
Archivage Résultat	5.7.0	Oui	Nettoyage Emplacement Source
Nettoyage Emplacement Source	5.7.0	Non	Nettoyage Emplacement Cible
Nettoyage Emplacement Cible	5.7.0	Non	

Insérer une Phase Historique

Dans le flux de travail de l'Environnement de Construction, la Phase *Exécution du script* a été retirée et remplacée par l'insertion de trois nouvelles Phases. Résultat: maintenant nous pouvons clairement voir quand une compilation échoue sans devoir analyser en détail le journal de construction.

Note: Même si le test d'unité échoue, nous acceptons que la mise en Paquet du code continue (en établissant la propriété *Abandon si erreur* de la Phase à *Non*), ce qui pourrait être utile dans un flux expérimental instable.

Chacune de ces Phases peut avoir son propre jeu de paramètres qui influencera le script sous-jacent et qui peut être différent en fonction de l'Environnement et du Projet, ce qui facilite la réutilisation de la Phase.


Aperçu des Phases

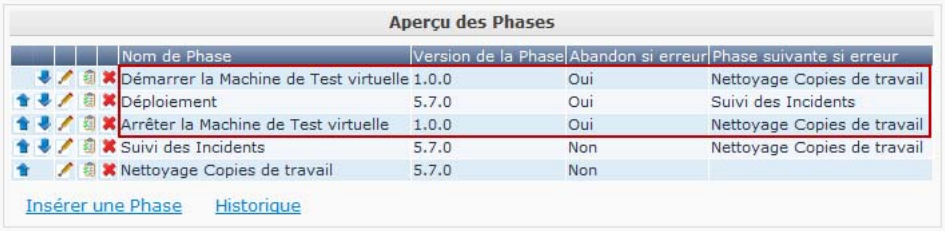
Nom de Phase	Version de la Phase	Abandon si erreur	Phase suivante si erreur
Récupération des Sources	5.7.0	Oui	Nettoyage Copies de travail
Récupération archive	1.0.0	Oui	Nettoyage Copies de travail
Construction	5.7.0	Oui	Nettoyage Copies de travail
Balisage	5.7.0	Oui	Nettoyage Copies de travail
Suivi des Incidents	5.7.0	Oui	Nettoyage Copies de travail
Nettoyage Copies de travail	5.7.0	Non	

Insérer une Phase Historique

Les Phases personnalisées peuvent également être utilisées sur un Niveau, ce qui peut être très utile si vous voulez exécuter des actions spécifiques sur le serveur IKAN ALM. Dans notre exemple nous avons créé une Phase qui récupère des fichiers (par exemple, des composants "prêts à utiliser" ou des librairies, tels que des fichiers dll, jar, exe ou autres) à partir d'une Archive ou d'un Référentiel accessible à partir du Serveur IKAN ALM.

2.4. Améliorer le flux de travail du Niveau de Test et de l'Environnement de Déploiement en utilisant des Phases personnalisées

Afin de montrer les nombreux avantages des Phases personnalisées, nous montrons également le flux de travail adapté sur le Niveau Test et son Environnement de déploiement.


Aperçu des Phases

Nom de Phase	Version de la Phase	Abandon si erreur	Phase suivante si erreur
Démarrer la Machine de Test virtuelle	1.0.0	Oui	Nettoyage Copies de travail
Déploiement	5.7.0	Oui	Suivi des Incidents
Arrêter la Machine de Test virtuelle	1.0.0	Oui	Nettoyage Copies de travail
Suivi des Incidents	5.7.0	Non	Nettoyage Copies de travail
Nettoyage Copies de travail	5.7.0	Non	

Insérer une Phase Historique

Les Phases *Démarrer/arrêter la Machine de Test virtuelle* sur le Niveau Test interagissent avec la Machine du client virtuel sur le Serveur IKAN ALM pour démarrer/arrêter la Machine de Test sur laquelle s'effectuera le Déploiement et sur laquelle seront exécutés les tests automatisés.


Aperçu des Phases

Nom de Phase	Version de la Phase	Abandon si erreur	Phase suivante si erreur
Transfert du fichier de construction archivée	5.7.0	Oui	Nettoyage du fichier de construction archivée
Décompression du fichier de construction	5.7.0	Oui	Nettoyage du fichier de construction archivée
Mise à jour Base de données	1.0.0	Oui	Nettoyage du fichier de construction archivée
Déploiement sur le Serveur Web	1.0.0	Oui	Nettoyage du fichier de construction archivée
Nettoyage du fichier de construction archivée	5.7.0	Non	

Insérer une Phase Historique

Sur l'Environnement de déploiement, les Phases personnalisées effectuent également le travail le plus important: la Phase *Mise à jour Base de données* mettra à jour la Base de données si le script SQL est présent dans le Résultat de construction, la Phase *Deploiement sur le Serveur Web* mettra à jour le serveur Web avec l'archive à déployer (dlls, war, fichiers config, ...) qui a été créée dans l'Archive de construction.

Maintenant que vous comprenez le concept des Phases personnalisées et comment elles enrichissent le flux de travail des Niveaux et des Environnements dans IKAN ALM, nous expliquerons comment vous pouvez développer vos propres Phases.

CHAPITRE 3
Développer une Phase personnalisée

Pour pouvoir créer une Phase personnalisée dans IKAN ALM, vous devez prendre comme point de départ un script existant (Ant, Gradle, Maven et NAnt sont supportés), le charger et créer la Phase, ainsi que les Paramètres de phase obligatoires et optionnels, dans la section de l'Administration globale.

Une fois définie, vous pouvez insérer la Phase personnalisée dans le flux de travail d'un Niveau ou d'un Environnement, établir les valeurs des paramètres et vérifier le résultat lors l'exécution d'une Requête de niveau.

Dans cet exemple, nous développerons une Phase personnalisée qui exécutera une mise à jour d'une Base de données.

Note: Notez qu'il est possible d'utiliser un langage de script autre que ceux supportés, puisque la plupart des Outils de script offrent la possibilité de lancer d'autres scripts et de capturer le journal de sortie (par exemple, en utilisant la tâche exec de Ant (voir http://ant.apache.org/manual/Tasks/exec.html)).

3.1. Créer un script

Nous prenons comme point de départ un script existant, créé selon les meilleures pratiques d'un Outil de script approprié. Vous trouverez la première version du script Ant *UpdateDB.xml* qui se trouve dans la section Le script Ant UpdateDB.xml et ses variables (page 24).

```
3
4       <description>
5           ANT script to update a database.  Currently supports MySQL, MS SQL, Oracle and DB2.
6           Prerequisites: the database driver must be in the ANT lib path.
7       </description>
8
9       <target name="updateDatabase" depends="init,validateRdbmsParams,executeUpdateDatabase"/>
10
```

Pour plus d'informations concernant le développement de scripts Ant, se référer au Manuel de Ant (voir http://ant.apache.org/manual/using.html#buildfile).

Le script *UpdateDB.xml* exécute 3 cibles dans la séquence suivante:

1. init

   Définit le chemin du script SQL qui mettra à jour la base de données à `${source}/update.sql` (plus tard suivront plus d'explications à ce sujet).

2. validateRdbmsParams

   Vérifie la Base de données choisie (MS SQL Server, MySQL, Oracle ou DB2).

3. executeUpdateDatabase

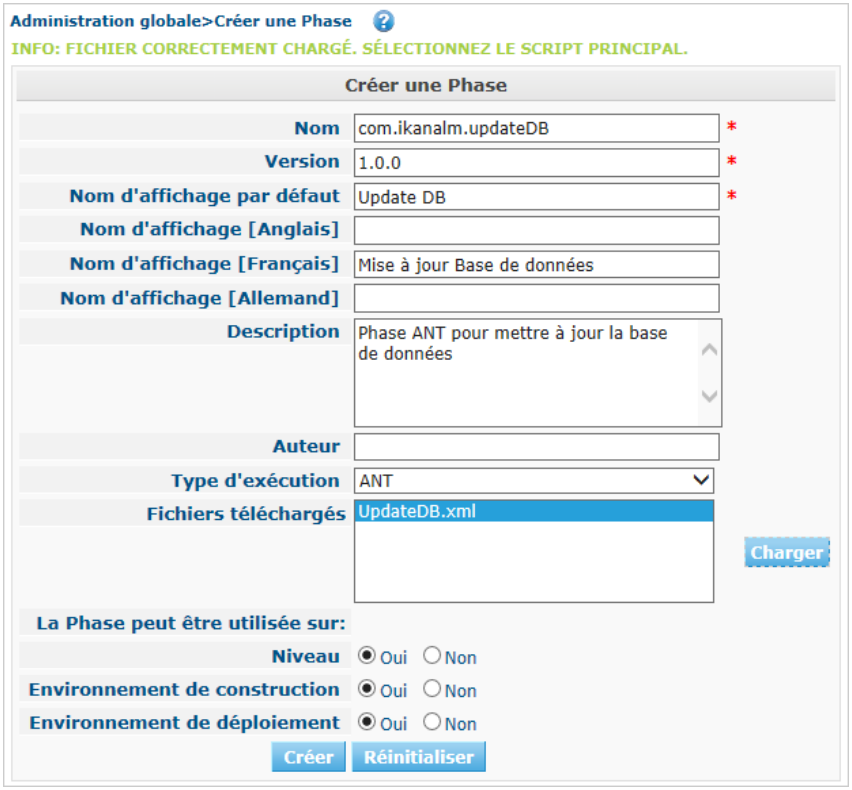
En fonction de la base de données choisie, une sous-cible est appelée pour établir les variables de connexion à la base de données (driver, URL). Ensuite, il vérifie si le script *update.sql* existe dans le chemin indiqué et, finalement, il utilise la tâche SQL Ant (voir http://ant.apache.org/manual/Tasks/sql.html) pour exécuter le script SQL sur la base de données. Des messages *echo* traceront des informations différentes lors de l'exécution de cette Cible.

```
69      <target name="executeUpdateDatabase" depends="paramDb2,paramMssql,paramMysql,paramOracle">
70          <!-- check for existence of base sql script -->
71          <fail message="Update SQL script not found : ${sql.script}">
72              <condition>
73                  <not>
74                      <available file="${sql.script}"></available>
75                  </not>
76              </condition>
77          </fail>
78
79          <echo>Executing Update SQL Script ${sql.script}...</echo>
80          <echo>Database connection parameters :</echo>
81          <echo>driver="${sql.rdbms.driver}"</echo>
82          <echo>url="${sql.rdbms.url}"</echo>
83          <echo>userid="${rdbms.user}"</echo>
84
85          <!-- execute the base sql script -->
86          <sql driver="${sql.rdbms.driver}"
87               url="${sql.rdbms.url}"
88               userid="${rdbms.user}"
89               password="${rdbms.pwd}"
90               src="${sql.script}"
91               delimiter=";"
92               encoding="latin1"
93               print="true"
94               onerror="continue">
95          </sql>
96
97          <echo>Execution of Update SQL Script finished.</echo>
98      </target>
```

Le script contient plusieurs variables (des propriétés Ant), tels que `${rdbms.type}` (le type de base de données, valeurs possibles: MYSQL, MSSQL, DB2 ou ORACLE) et d'autres variables de connexion de base de données, qui sont décrits dans l'appendice et qui doivent être fournis lors de l'exécution du script. Nous avons testé ce script avec un fichier de propriétés sur les bases de données supportées.

3.2. Créer la Phase et ses paramètres

Une fois le script testé et les variables identifiés, vous pouvez les envelopper dans une Phase personnalisée. Assurez-vous que vous avez des droits d'Administration globale et sélectionnez *Phases* > *Créer* dans le contexte de l'*Administration globale*.

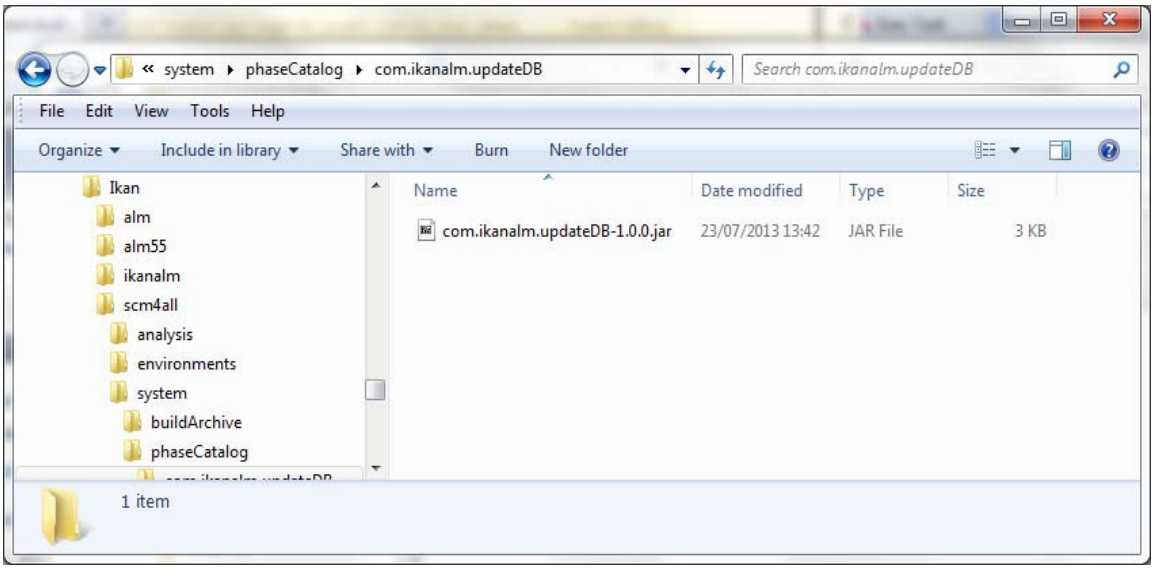


Fournissez les données nécessaires pour créer la nouvelle Phase personnalisée: le nom unique (de préférence un nom DNS inversé), la version (par exemple, major.minor.maintenance) et le nom d'affichage *Mise à jour Base de données* (utilisé lors de l'insertion ou de l'affichage dans les informations détaillées d'une Requête de niveau).

Vu que nous utilisons un script Ant, établissez le type d'exécution à Ant. Chargez le script *UpdateDB.xml* à partir du Système de fichiers. Finalement, spécifiez où la Phase peut être utilisée: sur un Niveau (donc exécutée par le Serveur IKAN ALM) ou sur un Environnement de Construction ou Déploiement (donc exécutée par un Agent IKAN ALM).

Fournissez également une description et un auteur. Pour plus d'informations concernant ces champs, se référer au chapitre *Phases* dans la partie *Administration globale* du *Guide de l'Utilisateur IKAN ALM*.

Après avoir cliqué sur le bouton *Créer*, la Phase sera créée dans le Catalogue des Phases sur le Serveur IKAN ALM (à l'Emplacement du Catalogue des Phases tel que spécifié dans *Administration globale* > *Système* > *Paramètres système*):



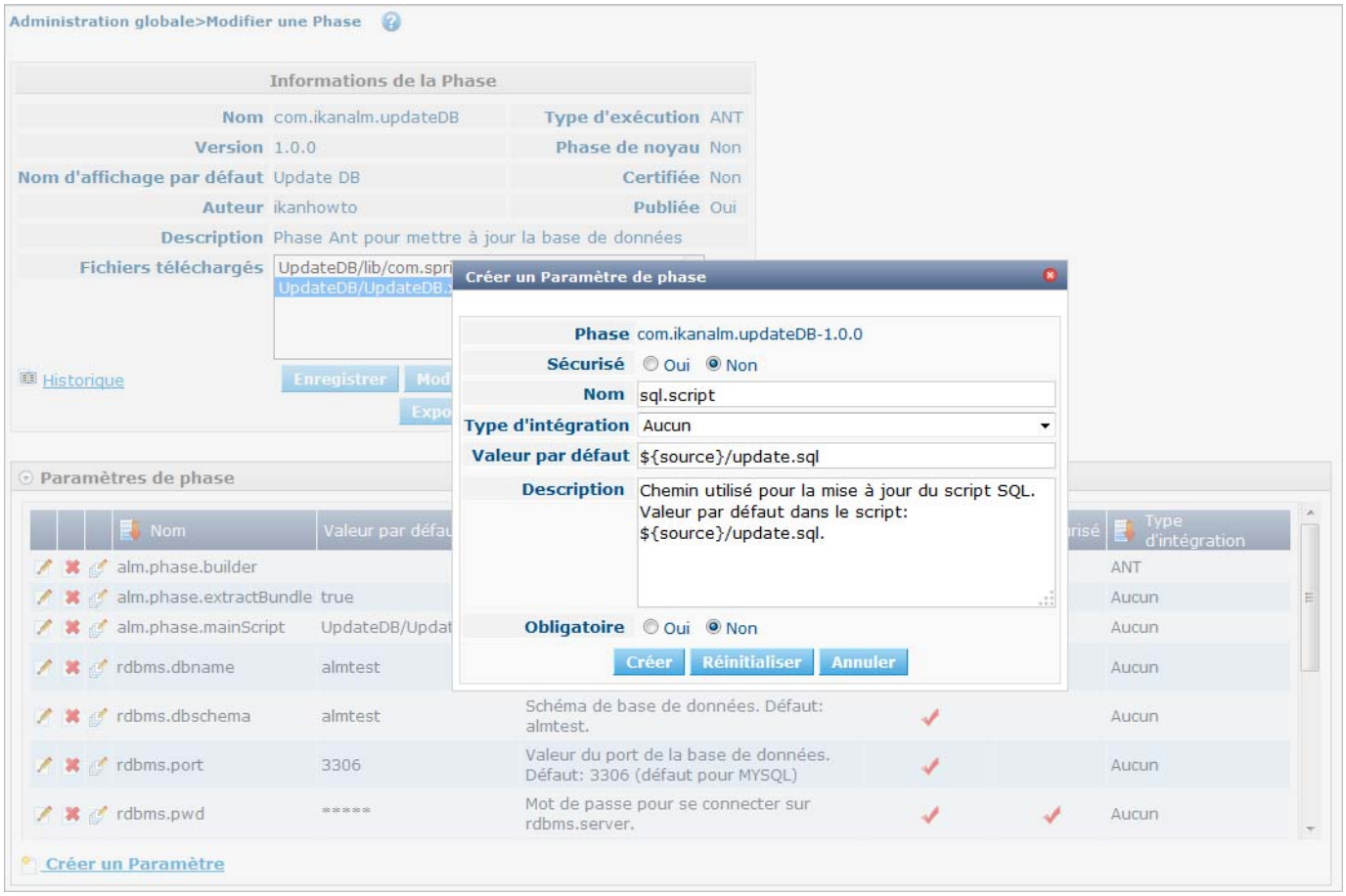
Le fichier d'archive résultant (name-version.jar) contiendra le script et quelques métadonnées et sera automatiquement transporté vers l'environnement du Serveur ou de l'environnement d'exécution de l'Agent IKAN ALM après le traitement de la Requête de niveau (voir plus loin).

Naviguez vers *Administration globale* > *Phases* > *Aperçu* et sélectionnez le lien *Éditer* devant la nouvelle Phase *Mise à jour Base de données* pour pouvoir ajouter les paramètres requis.

Note: Trois paramètres ont été créés automatiquement: alm.phase.builder, alm.phase.mainScript et alm.phase.extractBundle. Ils sont nécessaires pour l'exécution de la Phase et ne peuvent pas être supprimés.

Comme le type d'exécution de la Phase est ANT, le type d'intégration de alm.phase.builder est également ANT. Sa valeur par défaut peut être établie à un des Outils de script Ant qui sont définis dans l'Administration globale.

Créez les paramètres comme spécifiés dans l'Appendice (voir Le script Ant UpdateDB.xml et ses variables (page 24)) en sélectionnant le lien *Créer un Paramètre* en-dessous de la fenêtre d'aperçu *Paramètres de phase*.



Tous les paramètres sont obligatoires, sauf les paramètres sql.script et rdbms.dbschema. Le paramètre sql.script est déjà défini dans le script. En le rendant optionnel, nous utiliserons la valeur par défaut spécifiée dans le script. Vous pouvez toujours écraser cette valeur (nous expliquerons cela plus tard dans la section concernant l'insertion de la Phase dans un Niveau ou un Environnement). Le rdbms.schema n'est nécessaire que pour un rdbms.type DB2; vous pouvez également l'établir après avoir inséré la Phase.

Note: Vous pouvez utiliser le flag *Sécurisé* pour le paramètre rdbms.pwd pour que sa valeur ne soit jamais montrée à d'autres Utilisateurs. Dans la capture d'écran ci-dessus, vous verrez également que nous avons établi des valeurs par défaut pour les paramètres pour établir la connexion, dans ce cas, avec une base de données MySQL nommée "almtest" sur localhost. Ils peuvent être écrasés au moment de leur application dans un Niveau ou un Environnement.

3.3. Insérer la Phase dans le flux de travail d'un Niveau et/ou d'un Environnement

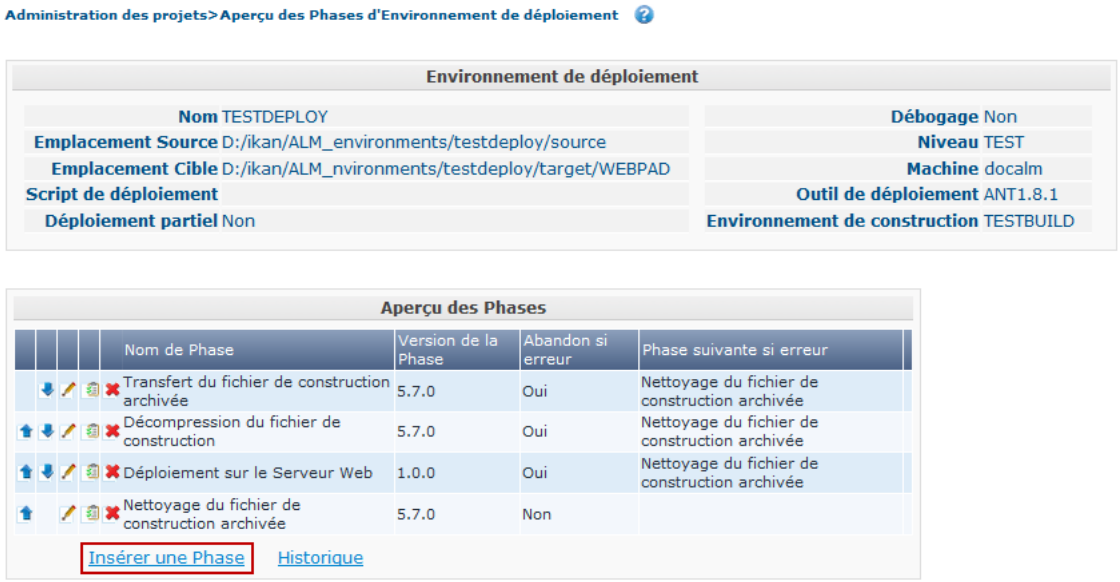
Astuce pour faciliter les choses: insérez la Phase dans un Environnement de construction ou de déploiement auquel est déjà associé un Outil de script Ant.

Une fois que la Phase et ses paramètres ont été définis dans l'Administration globale, vous pouvez l'insérer dans le flux de travail d'un Niveau ou d'un Environnement d'un de vos projets (suivant ce que vous avez spécifié dans la définition concernant l'endroit où elle peut être utilisée).

Assurez-vous que vous avez des droits d'Administration des projets, naviguez vers le Niveau ou l'Environnement Cible et sélectionnez le lien *Éditer les Phases*.

Note: Pour un Niveau, sélectionnez le lien *Éditer les Phases* dans la fenêtre *Aperçu des Niveaux*. Pour un Environnement de construction/déploiement, cliquez sur le lien *Éditer les Phases* dans l'*Aperçu des Environnements de construction/déploiement*.

Ensuite, cliquez sur le lien *Insérer une Phase* en bas de la fenêtre *Aperçu des Phases*:



Dans ce document, nous supposons que la Phase est insérée dans un Environnement de déploiement. Sélectionnez la Phase *Mise à jour Base de données* parmi les Phases disponibles, établissez l'attribut *Abandon si erreur* à *Oui* ou à *Non*, spécifiez sa Position d'insertion, indiquez quelle Phase devra être exécutée en cas d'échec et, finalement, cliquez sur le bouton *Insérer*:

Dans l'aperçu des Phases, sélectionnez le lien *Voir les Paramètres* à côté de la Phase *Mise à jour Base de données* qui vient d'être insérée, pour pouvoir vérifier tous les paramètres de Phase que nous avons définis dans l'Administration globale:

Tous les paramètres obligatoires sont automatiquement créés au moment où la Phase est insérée dans le flux de travail d'un Niveau ou d'un Environnement, et leurs valeurs sont copiées à partir des valeurs par défaut spécifiées dans l'Administration globale. Si vous voulez que cette Phase mette à jour une Base de données DB2, vous devez écraser les valeurs par défaut en cliquant sur le lien *Éditer le Paramètre* à côté des paramètres. Cependant, les paramètres optionnels doivent être créés si vous voulez les fournir lors de l'exécution du script. Donc, pour mettre à jour une base de données DB2 il vous faut le paramètre optionnel rdbms.dbschema nécessaire à l'URL jdbc.

Cliquez sur le lien *Créer un Paramètre* à côté du paramètre requis.

Utilisez le lien *Éditer un Paramètre de phase global* (uniquement disponible si vous avez des droits d'Administration globale) à côté du paramètre pour afficher la fenêtre *Éditer un Paramètre de phase* dans l'Administration globale. Là, vous verrez que votre Phase est maintenant connectée à un Environnement de déploiement, et que vous pouvez retourner dans le contexte du Projet en cliquant sur le même icône de lien *Éditer un Paramètre de phase d'environnement.*

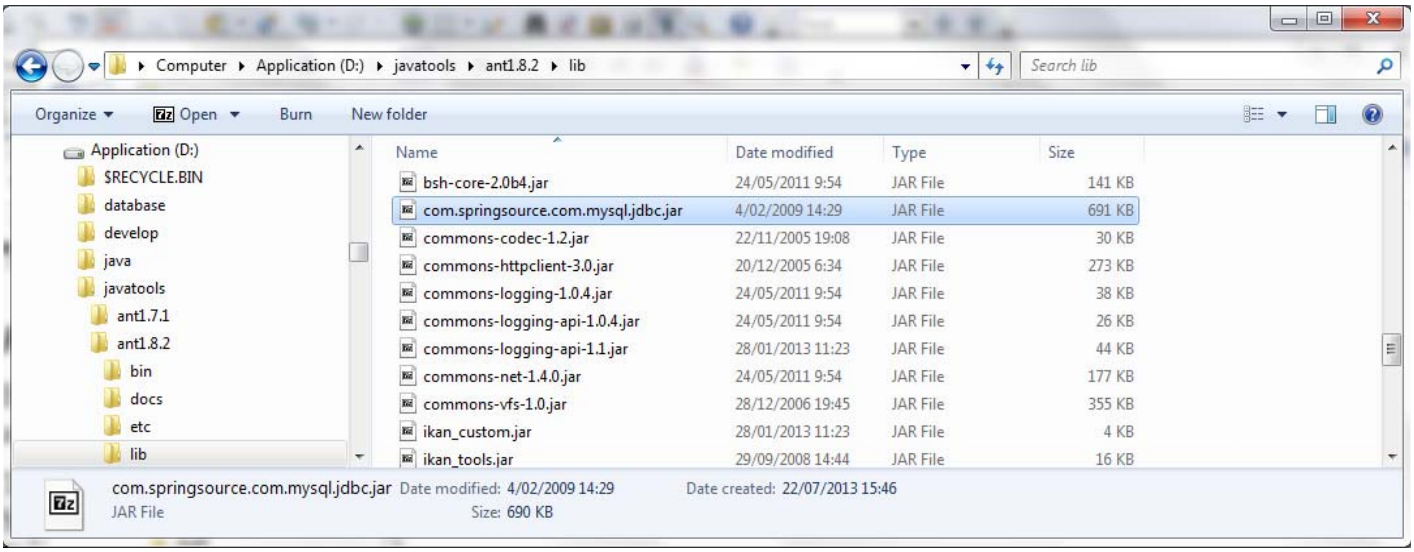
Note: Si vous avez ignoré l'astuce au début de cette section et que vous avez inséré le lien dans un Niveau ou dans un Environnement de construction/déploiement non associé avec une définition Ant, vous devez vous assurer que le paramètre alm.phase.builder reçoive la valeur d'une définition Ant qui existe soit a) sur le Serveur IKAN ALM dans le cas d'un Niveau; b) sur l'agent IKAN ALM (identifié par la Machine associée) dans le cas d'un Environnement de construction ou de déploiement.

Parce que vous avez changé le flux de travail, vous devez sélectionner *Auditer le Projet* à partir du menu *Administration des projets* et cliquer sur le bouton *Déverrouiller* avant de pouvoir créer une Requête de niveau.

3.4. Exécuter la Phase avec Créer une Requête de niveau

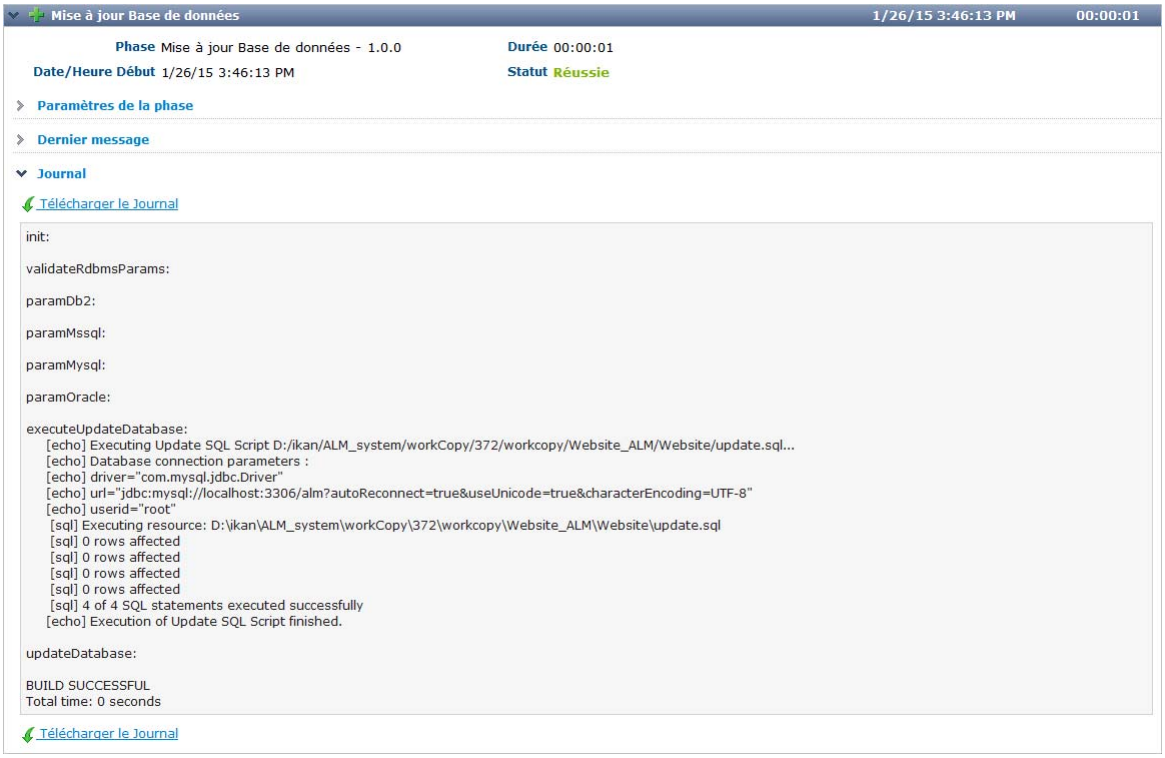
Avant de pouvoir exécuter la Phase, vous devez vous assurer de fournir a) le script *update.sql* et b) le driver jdbc, sinon la Phase échouera. De préférence, le script *update.sql* est chargé dans le Système de Contrôle de Versions connecté à votre projet. Parce que nous avons établi son emplacement par défaut à `${source}/update.sql`, celui-ci sera fourni pour un Environnement de construction si vous l'enregistrez (commit) dans le répertoire Racine de la branche ou du "trunk" que vous utilisez dans le projet. Si vous voulez qu'il soit disponible dans l'Environnement de déploiement (ce qui est notre cas vu que nous avons inséré la Phase *Mise à jour Base de données* dans un Environnement de déploiement), assurez-vous que vous le copiez à partir de ${source} vers ${target} lors de la création de la Construction qui sera déployée.

Notez que vous pouvez également optez pour établir la valeur du paramètre sql.script, en le créant comme un paramètre d'environnement éditable sur l'Environnement de déploiement. Ainsi, vous pouvez toujours modifier sa valeur lors de la création de la Requête de niveau. Le driver jdbc doit être présent dans le chemin de classe au moment de l'exécution de la Phase. Une manière de procéder est de le copier vers le répertoire ANT_Home/lib de l'installation Ant sur l'Agent IKAN ALM qui exécutera la Phase.



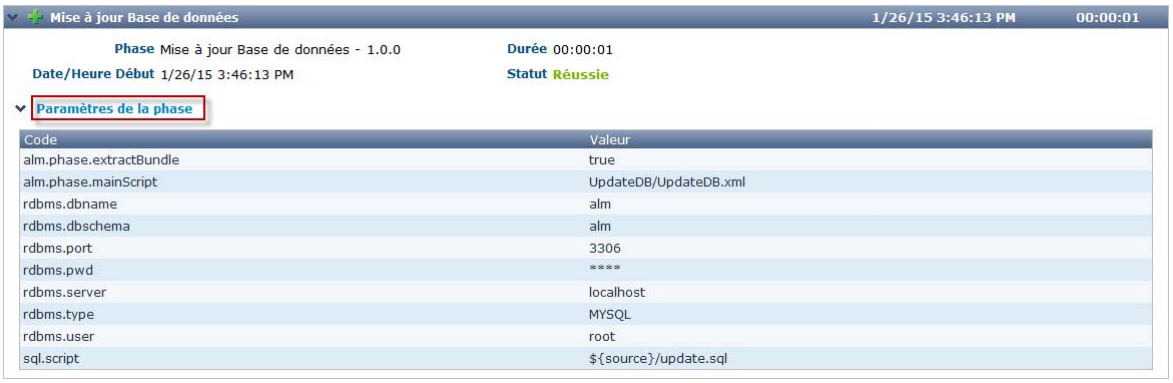
Nous optimiserons la configuration de ce driver lorsque nous traitons le Cycle de vie de la Phase dans le chapitre suivant. Maintenant que le script *update.sql* et le driver jdbc ont été correctement distribués, nous pouvons exécuter une Requête de niveau pour le Niveau qui contient l'Environnement de déploiement contenant notre Phase *Mise à jour Base de données*. Pour afficher le résultat du script, consultez les Journaux de Phase de la Requête de niveau (en sélectionnant l'onglet *Journaux de Phase* sur l'écran *Informations détaillées*). Là, vous retrouverez le journal de la Phase *Mise à jour Base de données*.

Nous optimiserons la configuration de ce driver lorsque nous traitons le Cycle de vie de la Phase dans le chapitre suivant. Maintenant que le script *update.sql* et le driver jdbc ont été correctement distribués, nous pouvons exécuter une Requête de niveau pour le Niveau qui contient l'Environnement de déploiement contenant notre Phase *Mise à jour Base de données*. Pour vérifier le résultat du script, consultez le log de la Phase de déploiement de Requête de niveau (en cliquant sur *Aperçu des détails* dans la fenêtre *Aperçu des déploiements* dans l'Aperçu détaillé de la Requête de niveau). Là, vous retrouverez le journal de la Phase *Mise à jour Base de données*.



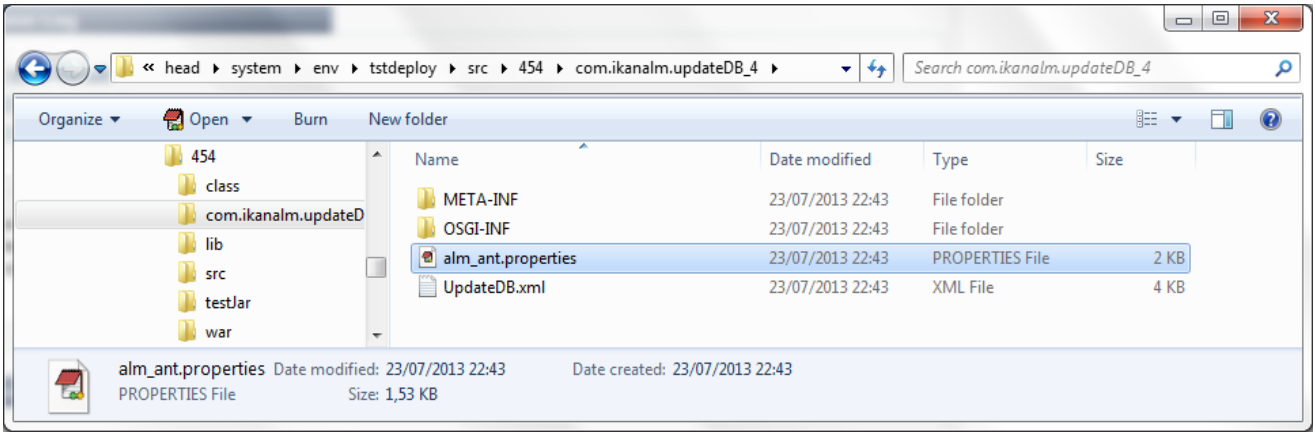
Vous reconnaîtrez les instructions cible et *echo* comme mentionnées dans la section Créer un script (page 6).

Cliquez sur le lien *Paramètres de la phase* dans le journal de la Phases Mise à jour Base de données pour en afficher les propriétés:



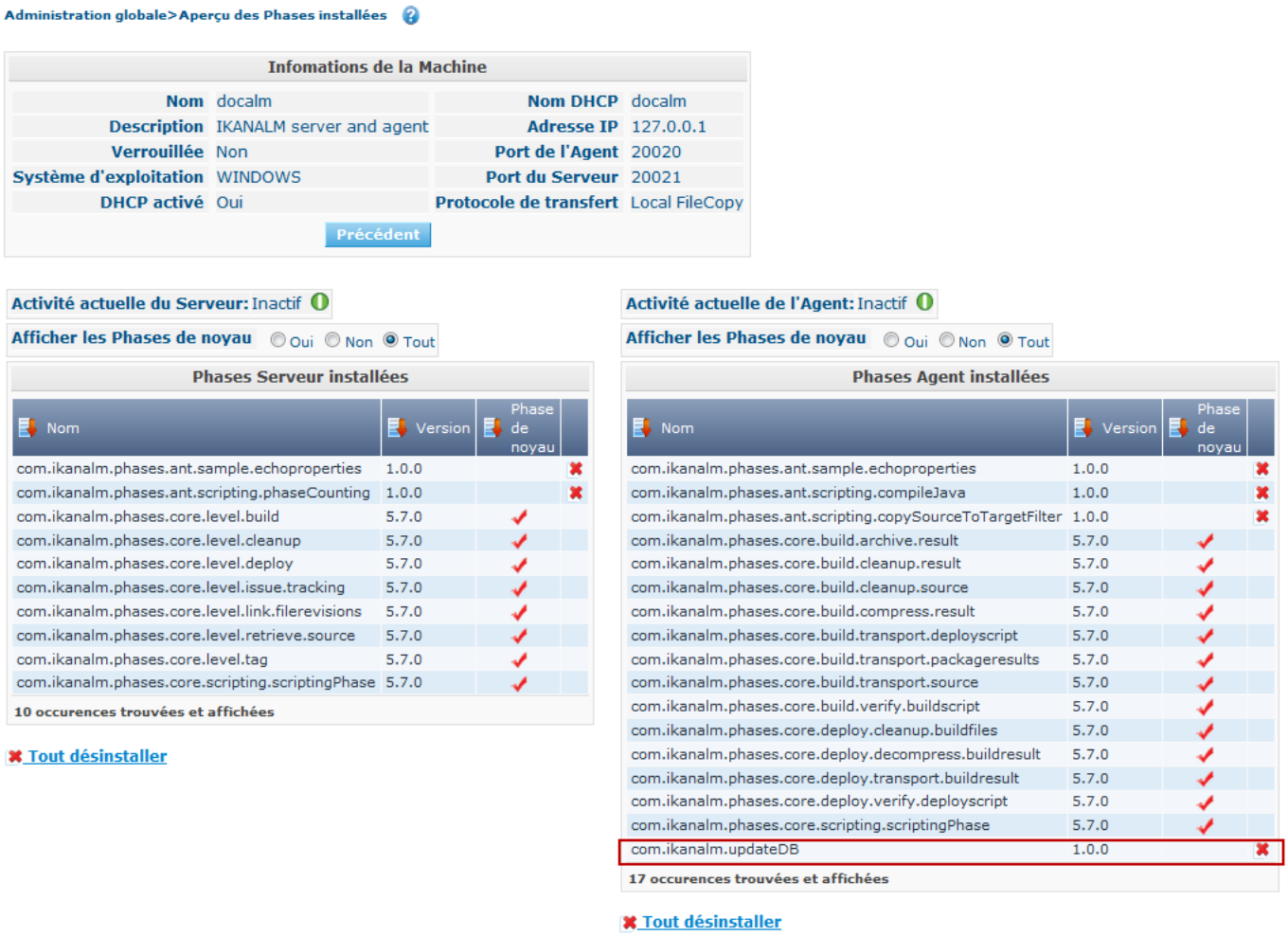
En plus de ces Paramètres de phase, le script peut également utiliser les Paramètres de Déploiement (affichés dans le panneau "Paramètres de Déploiement" près du sommet du Journal des "Actions de Déploiement"). Il s'agit là des Paramètres de déploiement prédéfinis (voir l'appendice à ce sujet dans le *Guide de l'Utilisateur IKAN ALM*) ainsi que des Paramètres d'Environnement de déploiement et des Paramètres de machine

(optionnels). Si vous activez le flag de *Débogage* pour l'Environnement de déploiement, vous pouvez vérifier les paramètres dans le fichier *alm_ant.properties* qui contient toutes les propriétés disponibles que vous pouvez utiliser dans le script. Il se trouve dans le sous-répertoire de la Phase extraite sous le répertoire des sources de l'Environnement de déploiement.



Dans ce répertoire vous trouverez également le script *UpdateDB.xml*. Avant l'exécution de toute Phase de déploiement, la *Mise à jour Base de données* a été transportée automatiquement à partir de l'Emplacement du Catalogue des Phases sur le Serveur IKAN ALM et installée dans l'Agent IKAN ALM, en utilisant le Transporteur (FileCopy, FTP ou SCP) connecté à la Machine représentant l'Agent. Tant que votre Phase se

trouve dans l'état non-publié (ce qui est l'état par défaut pour une Phase qui vient d'être créée), ce processus sera répété avant chaque action de Déploiement. Nous expliquerons plus en détails le Cycle de vie de la Phase dans le chapitre suivant. Vous pouvez voir quelles Phases sont actuellement installées sur la Machine Agent en sélectionnant le lien *Phases installées* à côté de la Machine Agent dans la fenêtre *Aperçu des Machines*:



CHAPITRE 4

Le Cycle de vie de la Phase

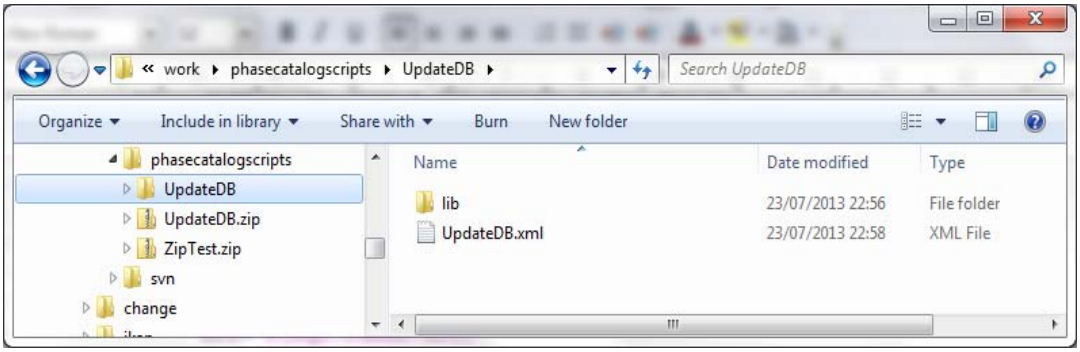
4.1. Améliorer la Phase: Une nouvelle version du script:

La Phase *Mise à jour Base de données* (Update DB) qui vient d'être créée a été testée et son fonctionnement a été prouvé lors du Déploiement. Cependant, il reste un problème avec la configuration du driver de la base de données, que nous devons copier manuellement vers le répertoire `ANT_Home/lib` de l'installation Ant sur l'Agent IKAN ALM sans quoi il ne peut pas être chargé et l'exécution de la Phase échouera.

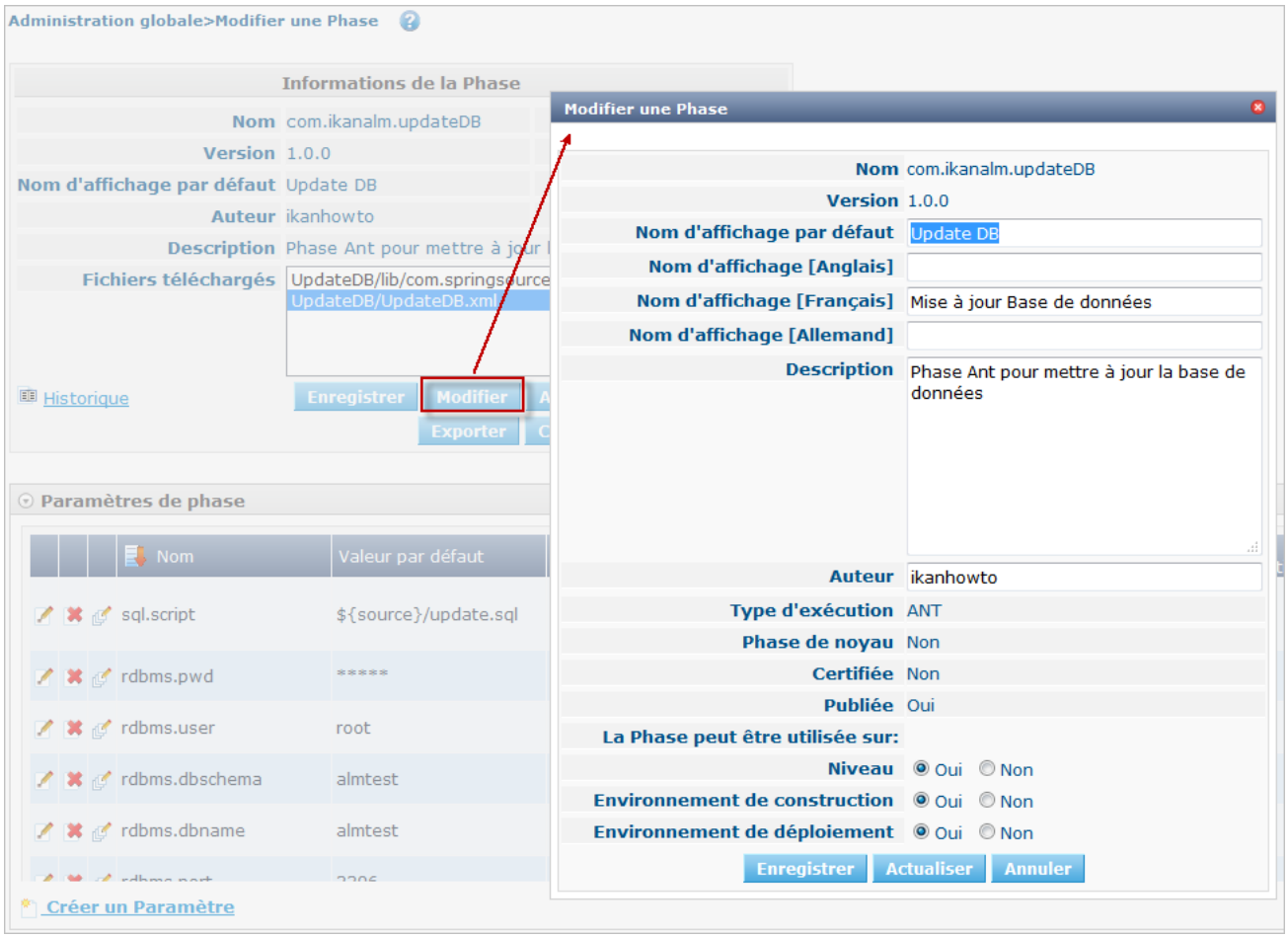
Nous pouvons résoudre cela en spécifiant une référence au chemin de classe dans la tâche SQL du script *UpdateDB.xml*.

```
61  <target name="executeUpdateDatabase" depends="paramDb2,paramMssql,paramMysql,paramOracle">
62    <!-- check for existence of base sql script -->
63    <fail message="Update SQL script not found : ${sql.script}">
64      <condition>
65        <not>
66          <available file="${sql.script}"></available>
67        </not>
68      </condition>
69    </fail>
70
71    <echo>Executing Update SQL Script ${sql.script}...</echo>
72    <echo>Database connection parameters :</echo>
73    <echo>driver="${sql.rdbms.driver}"</echo>
74    <echo>url="${sql.rdbms.url}"</echo>
75    <echo>userid="${rdbms.user}"</echo>
76
77    <!-- execute the base sql script -->
78    <sql driver="${sql.rdbms.driver}"
79      url="${sql.rdbms.url}"
80      userid="${rdbms.user}"
81      password="${rdbms.pwd}"
82      src="${sql.script}"
83      delimiter=";"
84      encoding="latin1"
85      print="true"
86      onerror="continue">
87      <classpath>
88          <fileset dir="${basedir}/lib" includes="*.jar"/>
89      </classpath>
90    </sql>
91
92    <echo>Execution of Update SQL Script finished.</echo>
93  </target>
```

Parce que le répertoire de base est établi à "." (la racine) dans la définition de projet du script Ant, cette ligne spécifie que le driver peut être trouvé dans un sous-répertoire de notre script nommé "lib". Donc, créez la structure de répertoires suivante dans un Environnement temporaire: un répertoire racine *UpdateDB*, contenant la nouvelle version du script et un répertoire lib, vers lequel vous copiez les drivers pour toutes les bases de données que vous voulez supporter avec cette Phase. Zippez le répertoire *UpdateDB* pour générer le fichier *UpdateDB.zip*.



Maintenant vous pouvez mettre à jour la Phase dans l'Administration globale. Dans le menu principal, sélectionnez *Phases* > *Aperçu* et cliquez sur le lien *Éditer* à côté de la Phase *Mise à jour Base de données*:

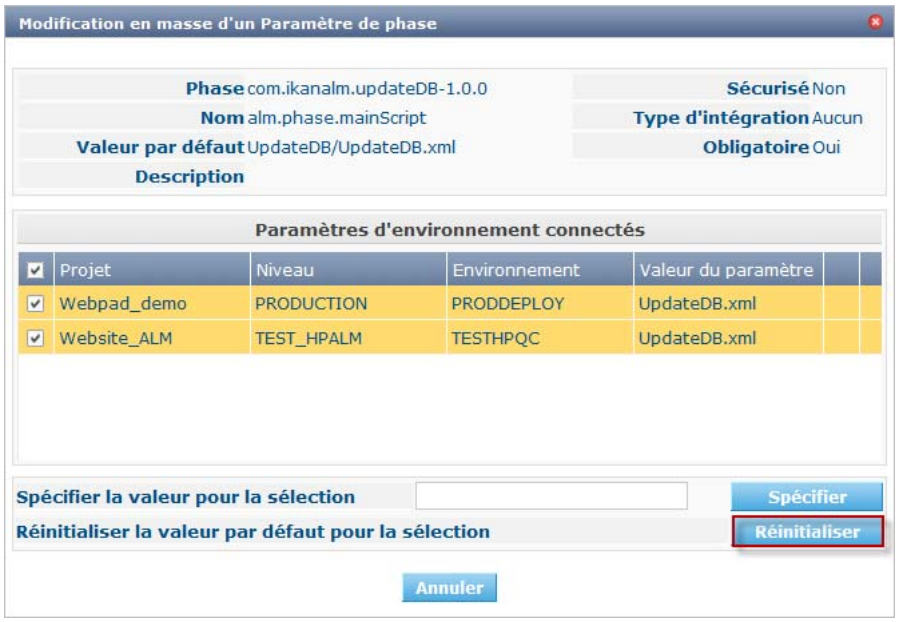


Note: À part le nom et la version de la Phase, la plupart des attributs peuvent toujours être modifiés vu que la Phase n'est pas encore publiée.

Cliquez sur le bouton *Charger* pour charger le fichier *UpdateDB.zip* que nous avons généré. Remarquez la nouvelle structure des Fichiers téléchargés: les drivers dans le sous-répertoire lib (un driver MySQL dans l'exemple ci-dessus) et le script mis à jour se trouvent en-dessous du répertoire *UpdateDB*. Assurez-vous que vous sélectionnez le fichier *UpdateDB/UpdateDB.xml* comme fichier principal parmi les Fichiers téléchargés avant de cliquer sur le bouton *Enregistrer*.

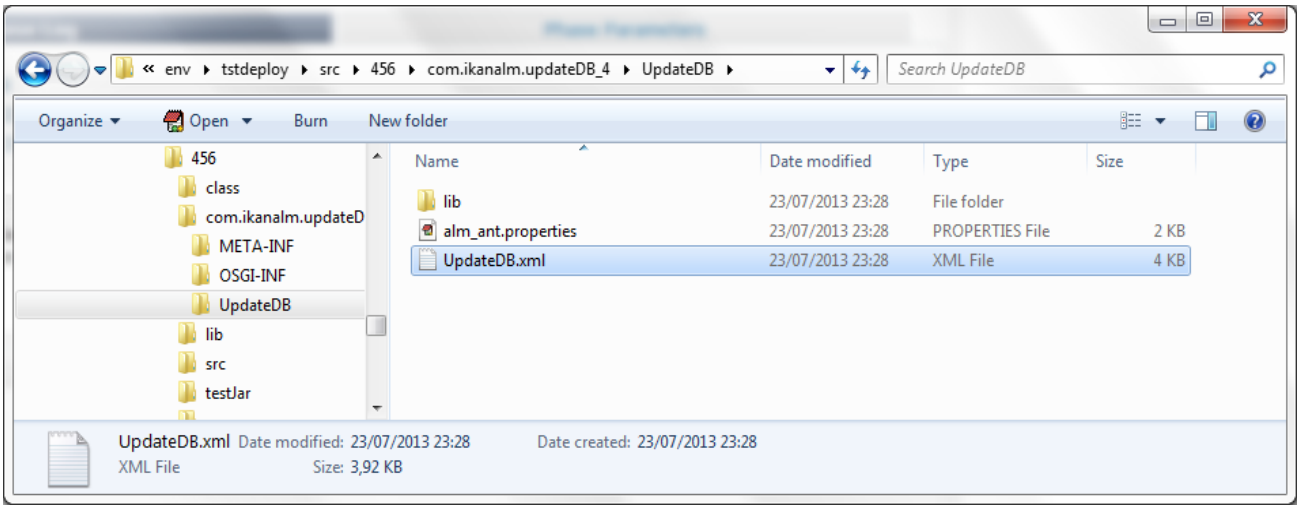
Si vous modifiez la Phase, vous verrez que le paramètre alm.phase.mainscript de la Phase a été établi à *UpdateDB/UpdateDB.xml* lors de la mise à jour. Avant de pouvoir tester notre nouvelle Phase, nous devons nous assurer que ce paramètre est également mis à jour dans l'Environnement de déploiement où il est utilisé.

Pour ce faire, cliquez sur le lien *Éditer en masse* à côté du paramètre alm.phase.mainscript.



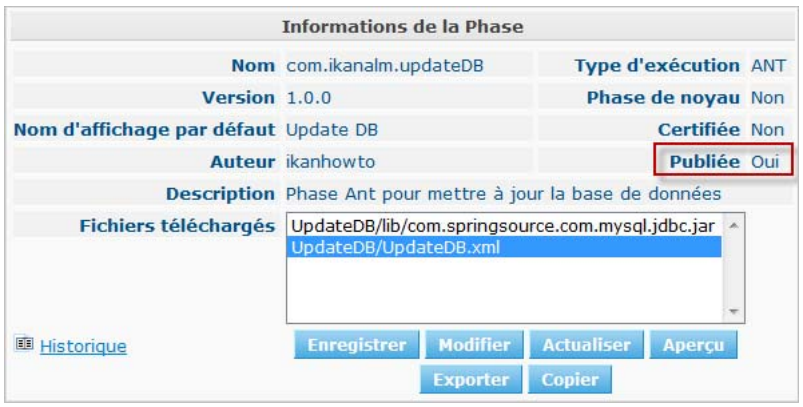
Dans la fenêtre des Paramètres d'environnement connectés, vous verrez tous les Niveaux et Environnements où la Phase *Mise à jour Base de données* a été insérée et où, par conséquent, le paramètre alm.phase.mainScript a été créé. Vous verrez également que ces paramètres ont la valeur ancienne *UpdateDB.xml*. Sélectionnez tous les paramètres en sélectionnant la case à cocher dans l'en-tête, et cliquez sur le bouton Réinitialiser pour changer toutes les valeurs de paramètre en *UpdateDB/UpdateDB.xml*.

Maintenant vous pouvez retirer le driver que vous avez fourni précédemment à partir du répertoire `ANT_HOME/lib` (Exécuter la Phase avec Créer une Requête de niveau (page 13)) et exécuter à nouveau la Requête de niveau. Parce que notre Phase n'est toujours pas publiée, elle sera à nouveau distribuée et installée sur l'Agent IKAN ALM avant l'exécution du Déploiement. Quand la Requête de niveau a terminé, et si vous avez activé l'option de Débogage pour l'Environnement de déploiement, vous verrez dans le répertoire source de l'Environnement de déploiement que le driver est maintenant fourni dans le répertoire lib de la Phase extraite:



4.2. Phase prête pour la Production: Publier la Phase

Maintenant que nous avons résolu le problème de l'attribution du driver et que nous avons testé la Phase *Mise à jour Base de données* avec plusieurs bases de données, elle est prête à être utilisée dans un environnement de production. À ce point, vous voulez que votre Phase soit protégée pour que le script ne puisse plus être modifié. Dans l'interface *Éditer une Phase*, cliquez sur le bouton *Publier* en bas du panneau *Éditer une Phase*. Cette action ne pouvant pas être annulée, vous devez la confirmez. Maintenant, réessayez d'éditer la Phase:

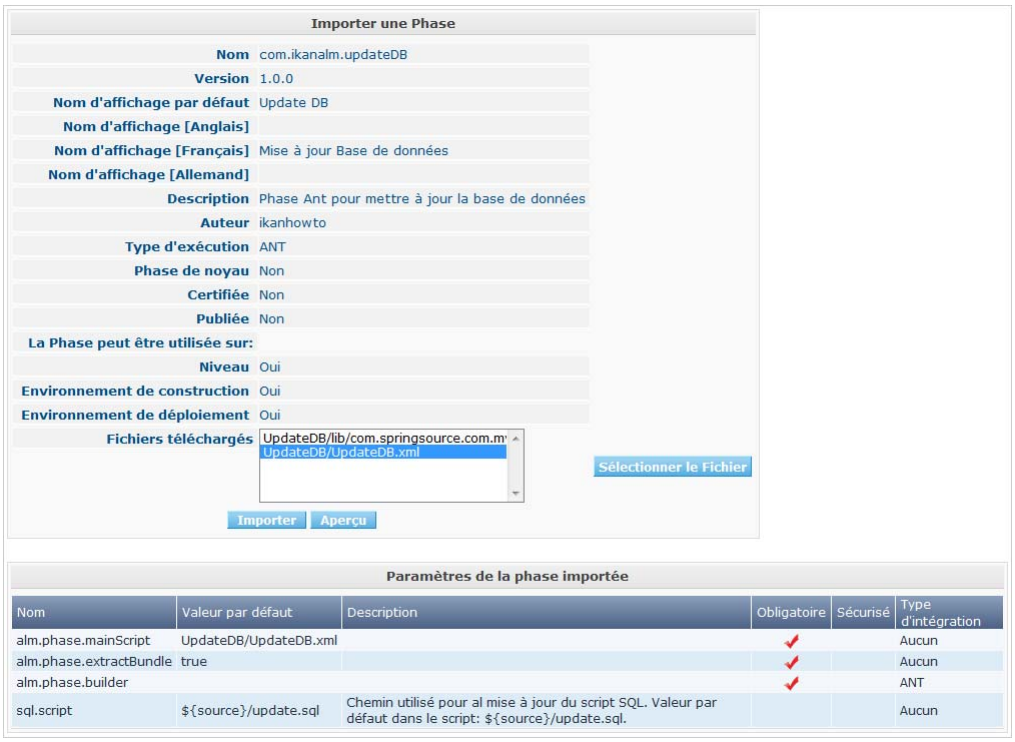


Parce que la Phase est Publiée, vous ne pouvez plus modifier les scripts. Par conséquent, le bouton Charger à côté des Fichiers téléchargés a disparu. Une autre conséquence est que la Phase n'est plus distribuée et installée sur le Serveur ou l'Agent IKAN ALM chaque fois qu'une Requête de niveau de construction/déploiement utilisant cette Phase est exécutée. Le seul moyen pour la redistribuer est de faire une désinstallation manuelle. Vous pouvez faire cela dans l'*Aperçu des Phases installées* en cliquant sur l'icône *Supprimer* à côté de la Phase *Mise à jour Base de données*.

4.3. Exporter/importer une Phase

Une fois que votre Phase est stable, vous pouvez l'exporter avec ses métadonnées (tous les paramètres définis) pour qu'elle puisse être réutilisée dans d'autres installations IKAN ALM. Cela peut vous aider si vous avez une configuration IKAN ALM sur un système de test (en parallèle avec votre configuration de production IKAN ALM), sur lequel vous expérimentez avec des Cycles de vie, des scripts et donc probablement aussi avec la création de Phases.

Dans la section Administration globale, sélectionnez l'icône *Exporter* dans l'Aperçu des Phases, ou utilisez le bouton *Exporter* dans l'interface *Éditer les Phases*. Un nouveau fichier d'archive sera chargé (name-version.jar) que vous pouvez importer dans une autre installation IKAN ALM via le menu *Phases* > *Importer*.



Après avoir sélectionné le fichier d'archive exporté, vous remarquerez que toutes les métadonnées, y-compris les Fichiers téléchargés et les paramètres, sont importées en même temps que le script. Seules les valeurs des paramètres sécurisés doivent être établies pour pouvoir commencer à travailler avec la Phase importée.

Note: Si vous avez établi une valeur par défaut pour alm.phase.builder, celle-ci ne sera pas non plus établie lorsque vous l'exportez et l'importez. Ceci est dû au fait qu'il n'est pas certain que la définition de l'outil de construction (Ant, Maven, ...) existe dans l'installation IKAN ALM dans laquelle vous l'importez.

4.4. Créer une nouvelle version de la Phase

Il est logique que les Phases puissent évoluer. Supposons, par exemple, qu'il y a une nouvelle version pour l'intégration que vous avez résolue avec la Phase, ou que vous constatez un problème avec le traitement d'une Phase qui est publiée. Dans ces deux cas, vous devez avoir la possibilité de modifier les scripts, mais cela n'est plus possible pour une Phase publiée. Dans le cas de notre Phase *Mise à jour Base de données*, un exemple pourrait être le support d'une base de données additionnelle. La solution dans ce cas serait de créer une nouvelle version de la Phase. Cela vous permettra de modifier les scripts et les paramètres.

Vous pouvez la créer à partir de zéro, mais le moyen le plus facile est de copier la Phase existante que vous voulez mettre à jour. Dans *Administration globale* > *Aperçu des Phases*, sélectionnez le lien *Copier* à côté de la Phase que vous voulez utiliser pour créer une nouvelle version. Modifier la version, le nom d'affichage par défaut et la description, et cliquez sur le bouton *Copier*. Une fois la Phase copiée, vous pouvez charger une nouvelle version du script et les autres fichiers qui doivent être distribués en même temps que la Phase. Tous les paramètres de la Version originale de la Phase sont également copiés et peuvent être entièrement adaptés (modifiés, supprimés, ajoutés). Une fois que votre Phase est prête, vous pouvez commencer à l'utiliser en l'insérant (en la remplaçant après avoir retiré la version précédente d'abord) dans le flux de travail des Niveaux et de l'Environnement. Notez que l'architecture de l'Agent et du Serveur IKAN ALM permet que différentes versions d'une seule Phase soit installées et exécutées sur la même Machine.

APPENDICE A

Le script Ant UpdateDB.xml et ses variables

A.1. UpdateDB.xml ANT script

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="updateDatabase" default="updateDatabase" basedir=".">

 <description>
 script Ant pour mettre à jour une base de données. Actuellement les bases de don-
nées MySQl, MS SQL, Oracle et DB2 sont supportées.
Prérequis le driver de la base de données doit être fourni dans le chemin lib Ant.
</description>

 <target name="updateDatabase" depends="init,validateRdbmsParams,executeUpdateDa-
tabase"/>

 <!-- get properties and set conditions :-->
 <target name="init">
 <!-- default location of the update SQL script, you may overwrite this as a Phase
Param -->
 <property name="sql.script" value="${source}/update.sql"/>
 </target>

 <!-- validate Database type and set is<DBTYPE> property : -->
 <target name="validateRdbmsParams" description="Validate Database Parameters">
 <fail message="Invalid database type : ${rdbms.type}">
 <condition>
 <not><or>
 <equals arg1="${rdbms.type}" arg2="MYSQL" trim="true" />
 <equals arg1="${rdbms.type}" arg2="MSSQL" trim="true" />
 <equals arg1="${rdbms.type}" arg2="ORACLE" trim="true" />
 <equals arg1="${rdbms.type}" arg2="DB2" trim="true" />
 </or></not>
 </condition>
 </fail>
 <condition property="isMYSQL" >
 <equals arg1="${rdbms.type}" arg2="MYSQL" trim="true" />
 </condition>
 <condition property="isMSSQL" >
 <equals arg1="${rdbms.type}" arg2="MSSQL" trim="true" />
 </condition>
 <condition property="isORACLE" >
 <equals arg1="${rdbms.type}" arg2="ORACLE" trim="true" />
 </condition>
```

```
 <condition property="isDB2" >
 <equals arg1="${rdbms.type}" arg2="DB2" trim="true" />
 </condition>
 </target>

 <!-- Set properties depending on database type -->
 <target name="paramDb2" if="isDB2">
 <property name="sql.rdbms.driver" value="com.ibm.db2.jcc.DB2Driver"/>
  <property name="sql.rdbms.url" value="jdbc:db2://${rdbms.server}:${rdbms.port}/
${rdbms.dbname}:currentSchema=${rdbms.dbschema};"/>
 </target>
 <target name="paramMssql" if="isMSSQL">
 <property name="sql.rdbms.driver" value="net.sourceforge.jtds.jdbc.Driver"/>
    <property    name="sql.rdbms.url"    value="jdbc:jtds:sqlserver://${rdbms.ser-
ver}:${rdbms.port}/${rdbms.dbname}"/>
 </target>
 <target name="paramMysql" if="isMYSQL">
 <property name="sql.rdbms.driver" value="com.mysql.jdbc.Driver"/>
      <property      name="sql.rdbms.url"      value="jdbc:mysql://${rdbms.ser-
ver}:${rdbms.port}/${rdbms.dbname}?autoReconnect=true&amp;useUni-
code=true&amp;characterEncoding=UTF-8"/>
 </target>
 <target name="paramOracle" if="isORACLE">
 <property name="sql.rdbms.driver" value="oracle.jdbc.driver.OracleDriver"/>
     <property     name="sql.rdbms.url"     value="jdbc:oracle:thin:@${rdbms.ser-
ver}:${rdbms.port}:${rdbms.dbname}"/>
 </target>
 <target name="executeUpdateDatabase" depends="paramDb2,paramMssql,paramMysql,pa-
ramOracle">
 <!-- check for existence of base sql script -->
 <fail message="Update SQL script not found : ${sql.script}">
 <condition>
 <not>
 <available file="${sql.script}"></available>
 </not>
 </condition>
 </fail>

 <echo>Executing Update SQL Script ${sql.script}...</echo>
 <echo>Database connection parameters :</echo>
 <echo>driver="${sql.rdbms.driver}"</echo>
 <echo>url="${sql.rdbms.url}"</echo>
 <echo>userid="${rdbms.user}"</echo>

 <!-- execute the base sql script -->
 <sql driver="${sql.rdbms.driver}"
 url="${sql.rdbms.url}"
 userid="${rdbms.user}"
 password="${rdbms.pwd}"
 src="${sql.script}"
 delimiter=";"
 encoding="latin1"
 print="true"
 onerror="continue">
 </sql>
 <echo>Execution of Update SQL Script finished.</echo>
 </target>
</project>
```

A.2. Les variables du script UpdateDB.xml (peuvent être définies comme des paramètres de Phase)

Variable	Description
rdbms.type	Type de Base de données. Actuellement, MYSQL, MSSQL, ORACLE et DB2 sont supportés.
rdbms.server	Le nom de la Machine ou l'Adresse IP du Serveur de Base de données, utilisés dans l'URL de connexion jdbc.
rdbms.port	Port de connexion du serveur de la Base de données, utilisé dans l'URL jdbc (par exemple, 3306 pour MySQL, 1433 pour MS SQL, 1521 pour Oracle, 50000 pour DB2).
rdbms.dbname	Nom de la Base de données qui sera mise à jour, utilisé dans l'URL de connexion jdbc.
rdbms.dbschema	Schéma de Base de données, utilisé dans l'URL de connexion jdbc pour DB2.
rdbms.user	L'utilisateur pour la configuration de la connexion jdbc; cet utilisateur doit avoir des droits de mise à jour.
rdbms.pwd	Le mot de passe de l'utilisateur de la base de données, utilisé pour la configuration de la connexion jdbc.
sql.script	Le chemin vers le script SQL qui sera exécuté sur la base de données.