IKANALM

# Architecture d'IKAN ALM

Combler l'écart Système de gestion de cycle de vie d’une application (ALM)  à l'échelle de l'organisation

# Tables des matières

# Résumé

IKAN ALM est une application Web serveur-agent pour la gestion du cycle de vie d'une application (ALM en anglais) avec une console d'administration basée sur le Web et constituée de plusieurs services ALM principaux et de plusieurs intégrations avec d'autres outils ALM.

Toutes les actions d'IKAN ALM sont définies et exécutées via l'application Web ou l'interface de ligne de commande (CLI en anglais).

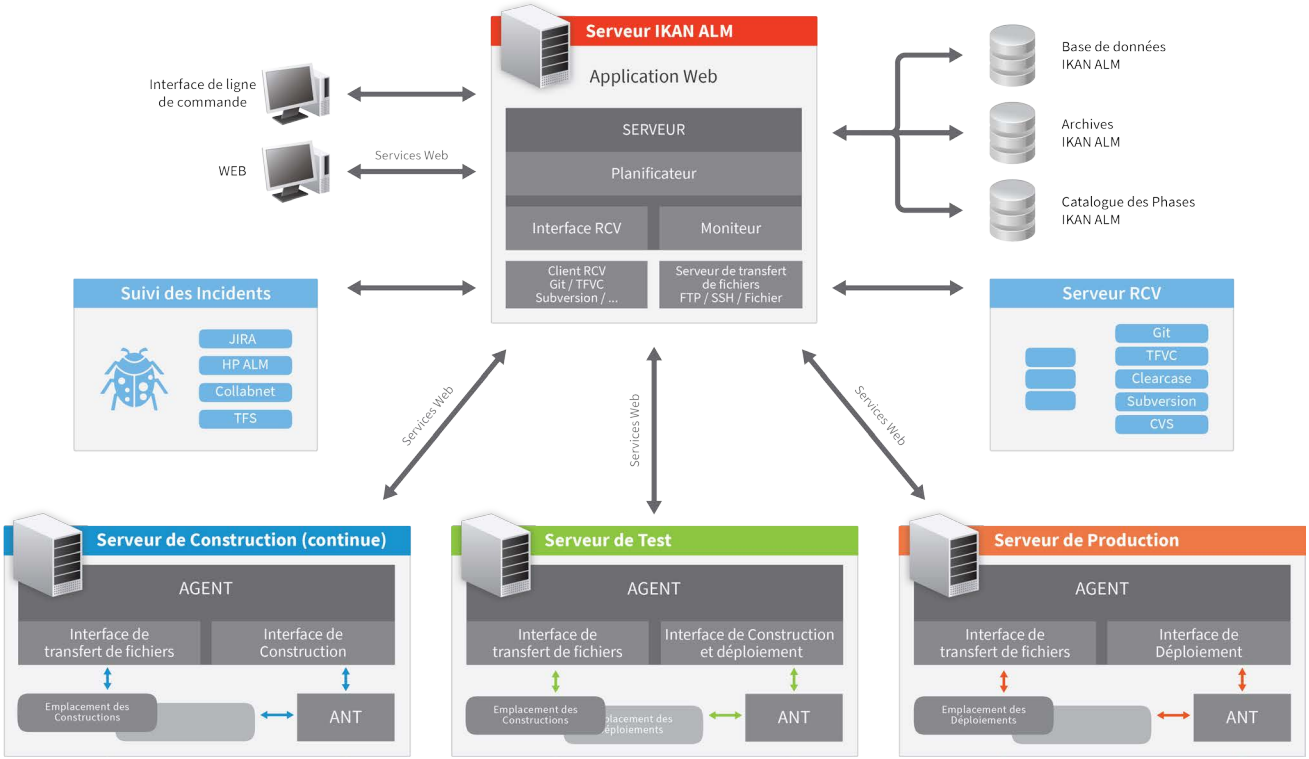
L'application Web est utilisée pour définir les éléments d'administration globale et de projet et le bureau personnel de l'utilisateur.

**Principaux services ALM offerts par IKAN ALM:**

- Cycle de vie
- Construction
- Déploiement
- Approbations
- Notifications

**Intégrations avec d'autres outils ALM:**

- Référentiels de contrôle de version
- Systèmes de Définition des besoins, de Suivi des Incidents ou de Suivi des Defects
- Outils de Construction
- Outils de Déploiement
- Outils de Reporting



La section " Administration globale" est utilisée pour définir les Paramètres Système, l'accès au Référentiel de Contrôle de Version, les Groupes d'utilisateurs, les Machines, les Transporteurs (FTP, FileCopy, Secured Shell), les Outils de script (Ant, Maven or NAnt) et les Systèmes de Suivi des Incidents (JIRA, HP ALM, Collabnet ou autres).

La section "Administration des projets" est utilisée pour définir le Projet, son Cycle de vie et ses Niveaux (un ou plusieurs Niveaux de construction, de tests et de production).

Les actions d'IKAN ALM sont appelées des Requêtes de niveau. Une Requête de niveau comprend une Requête de construction ou une Requête de déploiement (vers un environnement de Test ou Production). Les Requêtes de niveau sont créés via l'interface Web ou l'Interface de ligne de commande.

Toutes les informations des sections "Administration globale" et "Administration des projets" sont sauvegardées dans une base de données compatible JDBC, telle que MySQL, Oracle, DB2 ou MSSQL Serveur. Elle contient également le journal des actions des scripts effectuées lors de l'exécution des Requêtes de niveau. Les résultats de construction réels sont sauvegardés dans un format d'archive (*.zip ou *.tar.gz) dans les Archives de construction sur le Serveur IKAN ALM. Ainsi, ils pourront être récupérés pour des Requêtes de niveau de déploiement ultérieures.

L'application Web s'exécute sur un Serveur Web ou un Serveur d'applications (de préférence Apache Tomcat) sur la machine du Serveur IKAN ALM. Le processus "démon" Serveur traitant les Requêtes de niveau de Construction ou Déploiement s'exécute également sur la même machine. L'exécution d'une Requête de niveau consiste en l'exécution des différentes étapes, appelées des Phases, telles que la communication avec les systèmes externes (Contrôle de version, Suivi des Incidents, ...) et l'interaction avec le(s) Agent(s) IKAN ALM via des services Web pour exécuter les actions réelles de Construction et de Déploiement.

Puisque le processus Serveur suit la spécification OSGi (voir Architecture du Serveur IKAN ALM), IKAN ALM peut non seulement exécuter les Phases de Noyau, mais également les Phases personnalisées récupérées du Catalogue des Phases.

L'Agent IKAN ALM est également un processus démon suivant la spécification OSGi (voir Architecture de l'Agent IKAN ALM). Il transfère les Sources (dans le cas d'une Construction) ou un Résultat de construction (dans le cas d'un Déploiement) à partir de la Machine Serveur, il exécute un script utilisant un Outil de script (Ant, Maven, ...) et (dans le cas d'une Construction) il archive le Résultat de construction dans les Archives de construction sur le Serveur. Outre ces Phases de Noyau, il peut également exécuter les Phases personnalisées en récupérant ces Phases du Catalogue des Phases spécifié sur la machine Serveur.

Pour les Utilisateurs, le Catalogue des Phases, contenant les Phases de Noyau et les Phases personnalisées, est une caractéristique principale d'IKAN ALM. Chaque phase exécutée par le Serveur ou par un Agent correspond à une entrée dans le Catalogue des Phases.

IKAN ALM comprend des Phases de Noyau SERVEUR et Agent et tout Utilisateur peut définir et personnaliser des Phases dans le Catalogue des Phases.

Dans ce Catalogue, chaque Phase comprend un fichier d'archive Java (JAR) sur le système de fichiers du Serveur et les métadonnées connexes de la Phase. Les Phases personnalisées peuvent être ajoutées et gérées via l'interface Web.

Lors de l'exécution de la Phase, les métadonnées de la Phase sont utilisées tant par le Serveur que par l'Agent. Le nom et la version de la Phase sont les paramètres principaux pour identifier une Phase. Le numéro de version de la Phase permet d'exécuter des versions différentes d'une même Phase sur le même Serveur ou Agent(s).

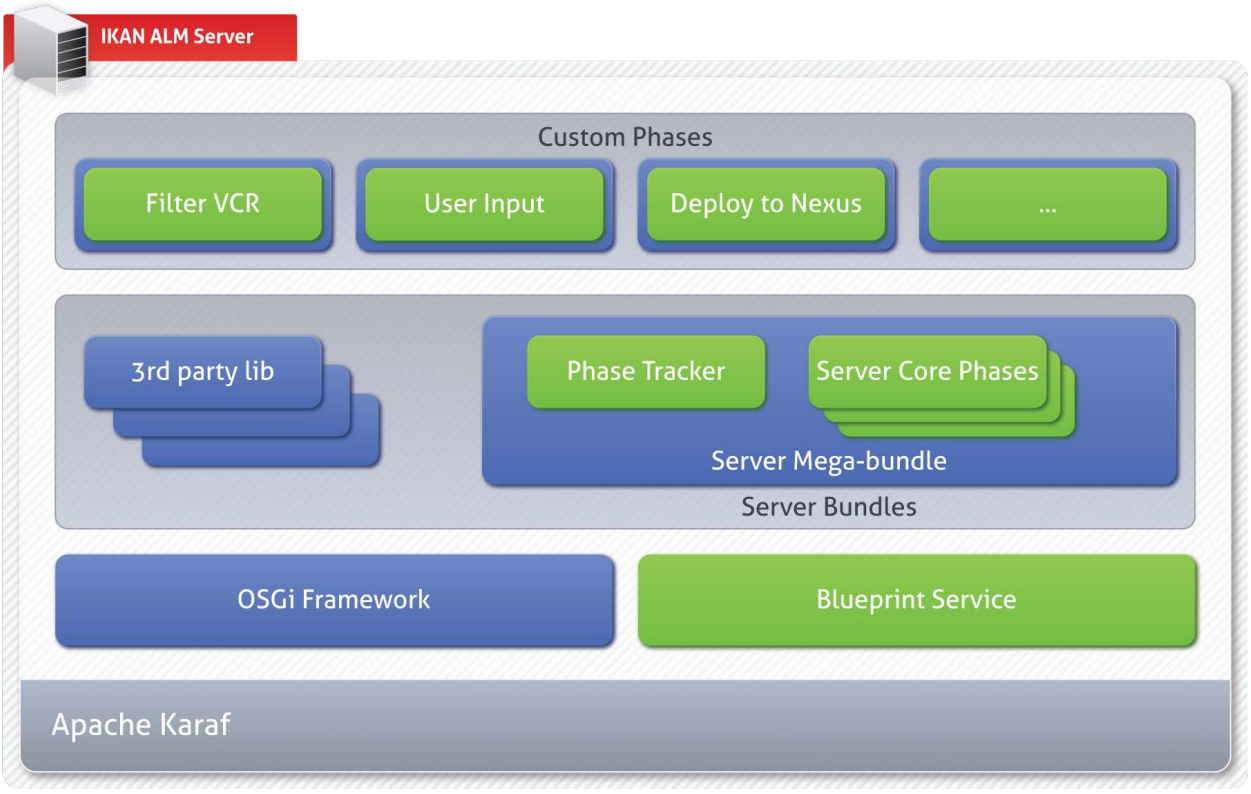
# Architecture du Serveur IKAN ALM

Le Serveur est un processus Java (Démon) contenant les sous-processus Moniteur et Planificateur. Il est exécuté sur une machine Java virtuelle (JVM) et il exécute les Requêtes de niveau lancées par un Utilisateur via l'interface Web ou initialisées par l'Interface de ligne de commande.

Le processus Moniteur dirige l'interaction avec les Référentiels de Contrôle de version lors de l'exécution d'une Requête de niveau (récupération du code Source, balisage du code Source) et il communiquera avec les Agents pour s'assurer que les processus de Construction et de Déploiement soient exécutés correctement (voir Architecture de l'Agent IKAN ALM).

Le support de l'Intégration continue, les Constructions nocturnes et le Déploiement continu est géré par le Planificateur. Le Planificateur appellera le Référentiel de Contrôle de Version pour vérifier si le Code Source a été modifié. Ce déclenchement axé sur les projets peut être effectué à des moments prédéfinis (chaque nuit à 20h00) ou à des intervalles de temps spécifiés (chaque jour, toutes les 5 minutes, ...)[1].

[1] Notez que l'Intégration continue peut également être obtenue par un script hook appelant la ligne de commande IKAN ALM lors de un enregistrement dans le Système de Contrôle de Version




Serveur IKAN ALM OSGi

Le Serveur est de type OSGi et il fonctionne dans Apache KARAF, un conteneur d'exécution OSGi léger. Karaf supporte le Composant Frameworks "Service Blueprint"[2], sélectionné et utilisé par notre solution. Ce Service Blueprint nous permet de rendre nos Phases enfichables.

Les autres avantages du conteneur Karaf sont le cadre de sécurité JAAS, le Service Feature, c.-à-d., le système d'approvisionnement permettant l'installation et la mise à jour automatiques des Phases (voir Interaction entre le Serveur/l'Agent IKAN ALM et le Catalogue des Phases), l'intégration avec le système opérationnel (s'exécute comme un Service Windows ou un démon Linux) et la possibilité de déploiement à chaud de bundles OSGi.

Le code Serveur est empaqueté dans un "Méga-bundle Serveur" OSGi dans lequel plusieurs Composants Blueprint principaux déclarent: le Traqueur de Phases et les Phases de Noyau Serveur. Les Phases de Noyau sont les blocs de construction de base pour le traitement des Requêtes de niveau: la récupération du code Source à partir d'un Référentiel de Contrôle de Version , la notification et la gestion des Agents pour assurer l'exécution correcte des Constructions et des Déploiements, le balisage du code Source dans le RCV.

Chacune de ces Phases est enregistrée comme un Service Blueprint au démarrage du méga-bundle Serveur et reprise par le Traqueur de Phases traçant les services Blueprint des phases.

Ce méga-bundle Serveur importe environ 60 bibliothèques tierces, tous traduits vers les Bundles OSGi appropriés. La plupart d'entre elles provient du Référentiel "SpringSource Enterprise Bundle Repository"[3]; d'autres ont été transformées par nos soins car elles n'étaient pas disponibles dans un référentiel public.

Finalement, le Traqueur de Phases enregistrera également les Phases personnalisées: il s'agit des Phases exécutées par un script particulier (Ant, NAnt ou Maven), qui sont définies dans le Catalogue des Phases par l'Utilisateur même et qui sont utilisées pour exécuter des processus particuliers sur le Serveur (par exemple, pour interagir avec une archive externe telle que Nexus, Jenkins ou TFS) dans le cadre d'une Requête de niveau.

Chacune de ces Phases se trouve dans un bundle OSGi séparé déclarant un seul Service Blueprint. Pendant l'installation ou la mise à jour, le Traqueur de Phases le détectera et le méga-bundle Serveur l'incorporera lors de l'exécution de la Requête de niveau utilisant cette Phase personnalisée spécifique.

Pour l'installation et la mise à jour des Phases personnalisées (ainsi que des Phases de Noyau) provenant du Catalogue des Phases, la fonctionnalité Karaf est utilisée (voir Interaction entre le Serveur/l'Agent IKAN ALM et le Catalogue des Phases).

## Architecture de l'Agent IKAN ALM

L'Agent est un processus Java (Démon) contenant un sous-processus de Construction et de Déploiement. Ce processus est exécuté sur une machine Java virtuelle (JVM) et il traite les Constructions et les Déploiements initialisées par le Serveur.

Il s'agit d'un processus dit "local" s'il est exécuté sur la même machine physique que le Serveur. S'il est exécuté sur une autre machine physique, on parle d'un Agent "distant". L'Agent interagit à distance avec le processus Moniteur du Serveur (via les services WEB) et localement avec un Transporteur (FileCopy, FTP ou SSH) et un Outil de script (Ant, NAnt ou Maven). À cet effet, ces Outils de script doivent être configurés correctement dans l'application Web IKAN ALM et sur la machine Agent.

[2] http://wiki.osgi.org/wiki/Blueprint
[3] http://ebr.springsource.com/

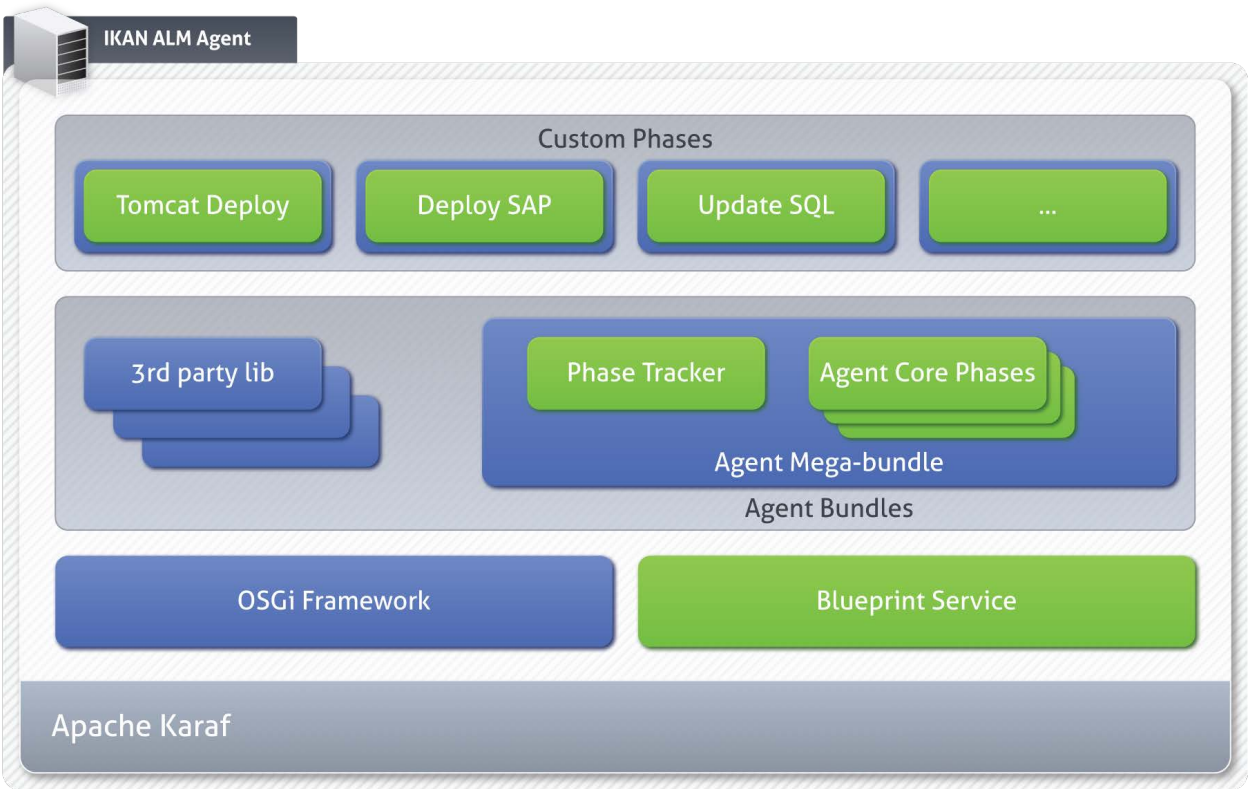
Tout comme le Serveur, l'Agent suit la spécification OSGi. L'Agent s'exécute dans Apache KARAF, offrant les mêmes avantages à l'Agent qu'il n'offre au Serveur. Le code de l'Agent est empaqueté dans un "Méga-bundle Agent" OSGi dans lequel plusieurs Composants Blueprint clé sont déclarés: le Traqueur de Phases et les Phases de Noyau Agent. Les Phases de Noyau sont les blocs de construction de base pour l'exécution des Requêtes de niveau: le transport de la source (Construction) ou la vérification d'un Résultat de construction (Déploiement) par un script de construction ou de déploiement, l'exécution de Constructions ou de Déploiements.

Chacune de ces Phases est enregistrée comme un service Blueprint au moment où le méga-bundle Agent est démarré et elle sera retenue par le Traqueur de Phases traçant les services Blueprint des phases.

Finalement, le Traqueur de Phases enregistrera également les Phases personnalisées: il s'agit de Phases exécutées par un script particulier (par exemple, pour mettre à jour une Base de données ou pour exécuter un Déploiement vers un serveur Web), définies dans le Catalogue des Phases par l'Utilisateur même et utilisées dans un Environnement de construction ou de déploiement associé à un Agent.

Chacune de ces Phases se trouve dans un bundle OSGi séparé déclarant chacun un seul Service Blueprint.

La fonctionnalité Karaf est utilisée pour l'installation et la mise à jour des Phases personnalisées (ainsi que des Phases de Noyau) à partir du Catalogue des Phases, (voir  Interaction entre le Serveur/l'Agent IKAN ALM et le Catalogue des Phases).




Agent IKAN ALM OSGi

# Interaction entre le Serveur/l'Agent IKAN ALM et le Catalogue des Phases

Le schéma suivant montre comment l'Agent et le Serveur interagissent avec le Catalogue des Phases lors de l'exécution d'une Requête de Construction et de Déploiement. L'objectif est la distribution d'une Phase personnalisée manquante lors de l'exécution d'une Construction par un Agent. Ce processus comprend les étapes essentielles suivantes

**Les étapes suivantes sont exécutées avant ce qui est affiché dans la figure suivante:**

1. Le processus Moniteur Serveur démarre la Requête de Niveau.

2. Le Serveur établit la liste des Phases de la Base de données devant être exécutées.

3. Cette liste contient les Phases de Noyau Serveur et les Phases personnalisées (par exemple: A, B).

4. Le Serveur consulte le Traqueur de Phases pour trouver les objets du service OSGi à partir des Phases Serveur. (Nous supposons que toutes les Phases de la liste établie par le Traqueur de Phases sont enregistrées. Si cela n'est pas le cas, le même processus que celui pour l'Agent est appliqué).

5. Le Serveur démarre l'exécution séquentielle des Phases. Une des Phases de Noyau est une Phase de construction: Via un service WEB, le Serveur notifie l'Agent qu'il doit exécuter une Construction.

**Alors intervient la figure ci-dessus:**

6. Le Processus de Construction Agent détecte la Requête de Construction.

7. L'Agent demande au Serveur la liste des Phases à exécuter.

8. L'Agent reçoit la liste des Phases à exécuter. Cette liste peut contenir des Phases de Noyau Agent ainsi que des Phases personnalisées.

9. L'Agent consulte le composant "Traqueur de Phases" pour récupérer les services OSGi à partir des Phases. Toutes les Phases de Noyau sont trouvées, la Phase personnalisée W n'est pas encore enregistrée. En conséquence, le Processus de Construction est arrêté temporairement.

10. L'Agent/le Traqueur de Phase demande au Serveur d'installer la Phase personnalisée manquante W.

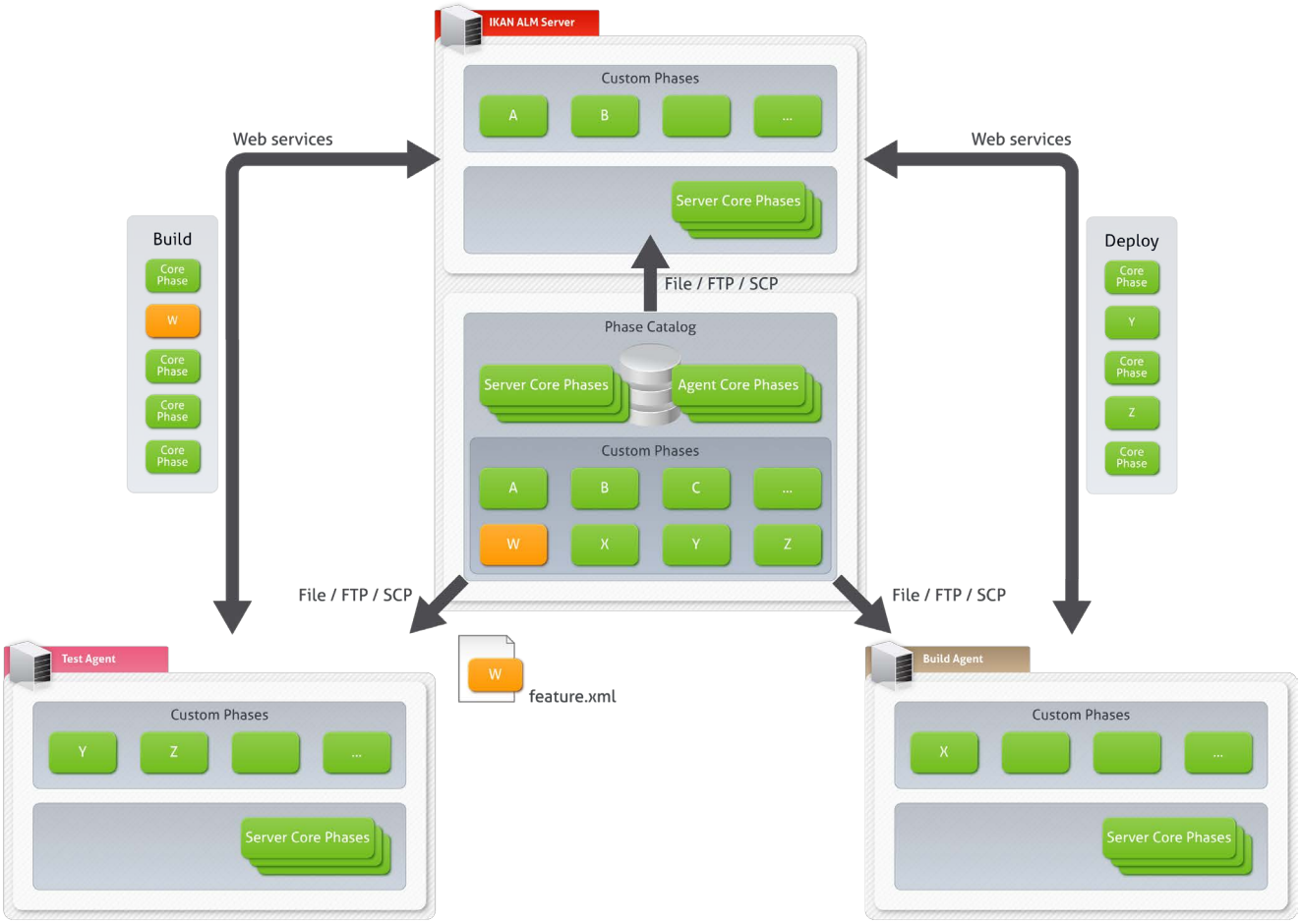
11. Le Serveur réagit par l'envoi du fichier XML contenant la description de la Phase et de l'URL que le Traqueur de Phases peut utiliser pour installer la phase manquante dans le conteneur Karaf Agent. Ce fichier XML est un "fichier de caractéristiques Karaf4", contenant l'URL référant l'emplacement de la version correcte de la Phase manquante W dans le Catalogue des Phases. L'URL est de type FTP ou SSH, en fonction du Transporteur spécifique défini pour l'Agent (Machine).

12. L'Agent transmet le "fichier de caractéristiques" au Feature Service Karaf. Le Feature Service récupère le bundle des Phases personnalisées à partir du Catalogue des Phases en utilisant l'URL fourni et l'installe dans le framework OSGi. Le composant Blueprint détecte que le bundle des Phases personnalisées contient un descripteur Blueprint et enregistre les Phases comme un service OSGi. Par conséquent, la Phase est enregistrée dans le Traqueur de Phases.

13. Quand le processus de Construction redémarrera, l'Agent consultera à nouveau le Traqueur de Phases pour trouver les objets du service OSGi pour cette Phase. Toutes les Phases de Noyau sont trouvées, y compris la Phase personnalisée manquante W.

14. L'Agent démarre l'exécution séquentielle des Phases pour cette Construction spécifique.

[4] https://cwiki.apache.org/confluence/display/KARAF/4.6.+Provisioning




Interaction entre le Serveur/Agent IKAN ALM et le Catalogue des Phases

# Glossaire

| | |
|---|---|
| ***Construction*** | Une Construction est une action sur l'Environnement de construction qui comprend plusieurs sous-processus, appelés des Phases de Construction. Elle fait toujours partie d'une Requête de niveau, qui peut également contenir d'autres Constructions ou Déploiements. [Une Construction est exécutée par l'Agent de construction IKAN ALM.] La Construction se base sur des codes Source récupérés à partir du RCV vers l'Environnement de Construction. Un Outil de script exécute un script de construction sur ces codes Source et en obtient un résultat de construction qui sera transféré vers les Archives de construction. |
| ***Archives de construction*** | L'Emplacement physique (chemin) sur le Serveur IKAN ALM où les résultats de construction sont sauvegardés dans un format compressé et archivé (*.zip or *.tar.gz). Les résultats de construction sont organisés par Projet et par Branche. |
| ***Phase de construction*** | Une Phase de construction est un sous-processus qui doit être exécuté pour compléter l'Action de construction. Les différentes Phases de construction constituent le flux de travail d'une Construction et elles sont insérées dans un Environnement de construction. Elles sont exécutées par le "IKAN ALM Builder Thread" de l'Agent IKAN ALM. Une Phase de construction peut être une Phase de noyau (par exemple, la Phase Vérification du script de construction) ou une Phase de construction personnalisée créée ou importée par l'Utilisateur dans le Catalogue des Phases. |

| | |
|---|---|
| ***Outil de construction*** | Un Outil de script installé dans un Environnement de Construction. |
| ***Phase de noyau*** | Les Phases de noyau constituent la fonctionnalité "de noyau" d'IKAN ALM. Elles peuvent uniquement être affichées, et ne peuvent pas être modifiées ni supprimées. Vous devez les considérer comme faisant partie intégrante d'IKAN ALM. Quand un nouveau Niveau, un nouvel Environnement de construction ou de déploiement est créé, son flux de travail par défaut sera créé et consistera entièrement d'une séquence de Phases de noyau. Ce flux de travail par défaut peut être modifié en supprimant des Phases de noyau, en modifiant la séquence des Phases ou en ajoutant des Phases personnalisées. |
| ***Phase personnalisée*** | Une Phase ajoutée par l'Utilisateur est également appelée une Phase "personnalisée". Elle peut être créée à partir de zéro dans l'Administration globale sur la base d'un ou plusieurs scripts ou ressources existants, ou elle peut être importée en utilisant la fonctionnalité "Importer une Phase". Une fois définie dans l'Administration globale, une Phase personnalisée peut être insérée dans le flux de travail par défaut d'un Niveau ou d'un Environnement de construction ou de déploiement (et, par conséquent, modifier ce flux de travail). Toutes les Phases personnalisées sont sauvegardées dans le Catalogue des Phases sur le Serveur IKAN ALM. Elles seront automatiquement transportées vers le Serveur IKAN ALM (Phase de niveau) ou l'Agent IKAN ALM (Phase de construction ou de déploiement) au moment de leur exécution. |
| ***Déploiement*** | Un Déploiement est une action sur l'Environnement de déploiement qui comprend plusieurs sous-processus, appelés des Phases de déploiement. Elle fait toujours partie d'une Requête de Niveau, qui peut également contenir une (ou plusieurs) autre(s) Construction(s) ou un (ou plusieurs) autres Déploiement(s). [Un Déploiement est exécuté par l'Agent de déploiement IKAN ALM.] Le Déploiement se base sur un résultat de construction récupéré à partir des Archives de Construction. Un Outil de script exécute un script de déploiement sur ce résultat de construction. |
| ***Phase de Déploiement*** | Une Phase de déploiement est un sous-processus qui doit être exécuté pour compléter l'Action de déploiement. Les différentes Phases de déploiement constituent le flux de travail d'un Déploiement et elles sont insérées dans un Environnement de déploiement. Elles sont exécutées par le "IKAN ALM Deployer Thread" de l'Agent IKAN ALM. Une Phase de déploiement peut être une Phase de noyau (par exemple, la Phase Transfert du fichier de construction archivée ou une Phase de déploiement personnalisée créée ou importée par l'Utilisateur dans le Catalogue des Phases. |
| ***Outil de déploiement*** | Un Outil de script installé dans un Environnement de Déploiement. |
| ***Agent IKAN ALM*** | Un processus (démon) se trouvant sur une Machine ayant des sous-processus pour l'exécution des Constructions ou des Déploiements. Si l'Agent est sur la même Machine que le Serveur IKAN ALM on parle d'un Agent "local", s'il est sur une autre Machine, on parle d'un Agent "distant". Pendant le processus de Construction ou de Déploiement, l'Agent IKAN ALM interagit à distance avec le processus Moniteur IKAN ALM et localement avec un Transporteur et un Outil de script qui doivent être correctement configurés sur la Machine. |
| ***Serveur IKAN ALM*** | La Machine hébergeant l'application Web d'IKAN ALM, ainsi que les processus Moniteur et Planificateur IKAN ALM. |

| | |
|---|---|
| ***Suivi des Incidents*** | Un système externe à IKAN ALM dans lesquels sont identifiés des Incidents (défauts, améliorations, tâches, ...) pour un Projet. Quelques exemples sont: Atlassian JIRA, HP Quality Center, Collabnet TeamForge, Bugzilla ou Trac. IKAN ALM peut être relié à un tel système et faire le suivi des Incidents concernant une Requête de Niveau. L'intégration avec JIRA et HP Quality Center est plus avancée: les Incidents sont automatiquement synchronisés à travers le Cycle de Vie et il est possible d'ajouter un lien vers la Requête de Niveau à l'Incident JIRA, au Défaut HP Quality Center ou à l'Artefact TeamForge. |
| ***Requête de niveau*** | Une Requête de niveau est une action sur un Niveau qui comprend plusieurs sous-processus, appelés des Phases de niveau. Dans la plupart des cas, une Requête de Niveau contiendra au moins une action de Construction ou de Déploiement qui sera exécutée sur des Machines locales ou distantes. Une Requête de Niveau peut être créée manuellement par l'Utilisateur, via l'Interface Web ou via la Ligne de Commande, ou automatiquement par le processus Planificateur du Serveur IKAN ALM. Une Requête de Niveau est traitée par le processus Moniteur du Serveur IKAN ALM. |
| ***Cycle de vie*** | Un Cycle de Vie est une séquence de Niveaux associés à une Branche. Il permet de définir chaque étape dans le processus de promotion des codes Source et des résultats de construction incluant le développement, les procédures de test, de qualité et la production. Un seul Projet peut avoir plusieurs Cycles de Vie, par exemple pour la prochaine réalisation, pour la maintenance et les corrections urgentes sur la version actuelle, pour les développements en parallèle, ... Un Cycle de Vie peut être réutilisé dans plusieurs Branches du Projet. |
| ***Catalogue des Phases*** | L'emplacement physique (le chemin) sur le Serveur IKAN ALM où les Phases personnalisées (créées à partir de zéro ou importées) sont sauvegardées dans un format d'archivage (Phase. name-Phase.version.jar, par exemple, com.ikanalm.echoproperties-1.0.0.jar). Si un Serveur ou un Agent IKAN ALM doit installer une Phase personnalisée manquante, elle sera récupérée à partir de cet emplacement. Cela sera fait en utilisant le Transporteur associé à la Machine de l'Agent ou du Serveur. |
| ***Projet*** | Un Projet IKAN ALM est associé à un Projet ou à un Sous-projet dans un Système de Contrôle de Versions (RCV) qui rassemble les codes Sources reliés. Un Projet IKAN ALM est une structure pour une ou plusieurs Branches pour lesquelles des actions réelles, telles que des Requêtes de Niveau, des Constructions ou des Déploiements, sont exécutées. Il est possible de définir des dépendances entres différents Projets, et à travers des Branches. Il existe deux types de projets: Projets de type "Édition Versions": IKAN ALM utilisera la structure existante dans le RCV pour que les objets à extraire soient récupérés automatiquement au moment de la Construction. Projets de type "Paquets": ce concept permet de travailler avec des fichiers isolés du système RCV. Les objets doivent être ajoutés manuellement dans une structure de paquet créée dans IKAN ALM avant le lancement du processus de Construction. |
| ***Outil de script*** | Un système externe à IKAN ALM installé sur une Machine et capable d'exécuter des scripts créés par l'Utilisateur. IKAN ALM s'intègre avec ANT, NAnt et Maven2. Si l'Outil de script est associé à un Environnement de Construction (de Déploiement) il est appelé Outil de Construction (de Déploiement). Le script pour l'exécution d'une Construction ou d'un Déploiement doit être sauvegardé dans le RCV (ensemble avec les codes Source) ou dans l'Emplacement des Scripts sur le Serveur IKAN ALM. |

| | |
|---|---|
| ***Transporteur*** | Un Transporteur est utilisé pour le transfert de fichiers et de répertoires entre le Serveur IKAN ALM et un Agent local ou distant qui exécute les processus de Construction ou de Déploiement. Par conséquent, un Transporteur doit être défini pour une Machine spécifique associée à un Environnement de Construction ou de Déploiement. IKAN ALM supporte les Transporteurs FileCopy, remote FileCopy, SecureCopy et FTP. Un Transporteur peut transporter les Sources extraites à partir du Système de Contrôle de Versions et le Résultat de construction à partir des Archives de construction, mais il peut également extraire les Phases personnalisées à partir du Catalogue des Phases. |
| ***Référentiel de Contrôle de version (RCV)*** | Un Système de Contrôle des Versions contenant les différentes versions des codes Source. Les codes Source reliés sont regroupés dans un Projet ou un Sous-projet (parfois également appelé un Module). Un Projet RCV peut contenir différents flux de développement, appelé “head” (= main ou trunk), ou Branches. IKAN ALM s’intègre avec les RCVs suivants: CVS, Subversion, Microsoft Visual SourceSafe, IBM ClearCase et Serena PVCS. Pour pouvoir se connecter au RCV, le client RCV doit être correctement installé sur le Serveur IKAN ALM. Le processus Moniteur IKAN ALM interagit avec le RCV en récupérant ou en balisant les codes Source. L’Interface Web interagit avec le RCV pour afficher les numéros de révision, les codes Source modifiés, ... d’une Requête de Niveau. |

# Pour plus d'informations.

Pour en savoir plus, visitez notre site http://www.ikanalm.com
Contactez IKAN Development: info@ikanalm.com

**IKAN Development (Belgium)**
Kardinaal Mercierplein 2
2800 Mechelen, Belgium
Tel. +32 15 797306

**IKAN Development (France)**
3, Rue du Général De Gaulle
28700 Aunay-Sous-Auneau, France
Tél: +33 2 37 25 31 22

info@ikan.be
www.ikan.be