# IKAN**ALM**

# Bimodal and IKAN ALM

One platform for Gartner's Bimodal
Mode 1 and Mode 2

# Table of contents

# Gartner defines Bimodal as follows:

*"Bimodal is a collection of principles, capabilities, methods, behaviors and approaches that enable an organization to differentiate the normal from the abnormal, the evolution from the revolution; the continual improvement from the disruptive innovation — and manage them differently but coherently. It's about the business, not IT, even if in some cases Bimodal starts within IT."*

Bimodal is the practice of managing two separate but coherent styles of work: one focused on predictability; the other on exploration.

## Mode 1: Predictable

Mode 1 focuses on predictability and has a goal of stability. It is best used where requirements are well-understood in advance, and can be identified by a process of analysis. It includes the necessary investment in renovating and opening up the legacy environment.

## Mode 2: Experimental

Mode 2 is exploratory. In this case, the requirements are not well-understood in advance. Mode 2 is best suited for areas where an organization cannot make an accurate, detailed, predefined plan because not enough is known about the area. Mode 2 efforts don't presume to predict the future, but allow the future to reveal itself in small pieces. This work often begins with a hypothesis that is proven, disproven or evolves during a process typically involving short iterations/projects.

Source: Gartner - Deliver on the Promise of Bimodal - Published: 18 February 2016 ID: G00299685.

Our IKAN ALM framework and solutions are the perfect fit for Mode 1, asking for predictability and stability, as well as for Mode 2, requiring short iterations/projects. Using IKAN ALM you can have Mode 1 and Mode 2 projects, each of them having a distinct lifecycle using related IKAN ALM Build and Deploy Phases or scripts.

On the one hand IKAN ALM offers a single platform to renovate and open your legacy environments and on the other hand it allows to experiment with hypotheses using short iterations or small project.

Once your Mode 2 projects become predictable and stable, you can move them to Mode 1 by using one and the same platform, IKAN ALM.

In this white paper, we will describe a Mode 1 and a Mode 2 way of working.

For Mode 1 we will use an IBM z/OS mainframe example and a JAVA example (distributed platform), and for Mode 2 a research project investigating Docker.

ALM solutions consist of an **integrated set of tools** for:

- Requirements Management
- Versioning
- Build & Deploy to test and production environments
- Lifecycle and Approval Management

And, also of major importance, adhere to the relevant process standards and governance rules.

Modern ALM solutions are:

**Methodology-independent:** It does not matter whether you use a linear (waterfall) or iterative (Agile) approach for your development process.

**Repository-neutral:** Cross-platform development is perfectly feasible.

**Tool-independent:** Organizations/departments can continue using their preferred tools in each stage of the process.

**Multi-platform:** mainframe, distributed or mobile, all of them are covered.

**Bimodal:** Support Bimodal, Mode 1 and Mode 2.

**Possible cost savings are:**

- Less time and resources needed for developing, testing and deploying (multi-environment) applications
- Having one ALM platform for Bimodal Mode 1 and Mode 2
- Reduced risk of human errors
- Improved efficiency and productivity
- Improved application quality and release reliability
- Faster time-to-market
- Improved customer satisfaction

In sum, implementing ALM to achieve Bimodal, increases your productivity through standardization, improves the communication between all stakeholders, speeds up the time-to-market, reduces the risks and increases the quality, in a cost-efficient way.

Our IKAN ALM solution offers all that, with as main asset the fully automation of the build and deploy phases for both Bimodal Mode 1 and Mode 2. In the following sections we will give a general overview of the IKAN ALM features, followed by three examples to explain how IKAN ALM covers Bimodal.

# Challenges

## General challenges

- No common ALM platform
- Development teams lose too much time with the build and deploy work, too many avoidable mistakes are being made and too much effort is required to communicate properly.
- Testers lose too much time setting up test environments and, as a consequence, they do not have enough time to test applications in a proper way.
- Operations people don't receive all the needed information or components are missing.
- The communication between the different stakeholders needs to be improved.

## Bimodal challenges

- How to renovate legacy applications?
- How to open up the legacy environments for the future?
- How to support exploration and disruption?
- How to achieve coherence between Bimodal Mode 1 and Mode 2?
- How to better support the business?

# The IKAN ALM Solution

The ability to have project-based (Mode 1 or Mode 2) life-cycles and automated build and deploy procedures will solve most of the issues mentioned above and, as a result, will speed up the processes and enhance the quality. Integrations with Issue Tracking and Versioning Systems will add even more efficiency and data consistency.

IKAN ALM is a platform and environment-independent Application Lifecycle Management solution aimed at ensuring a Bimodal way of working throughout the whole lifecycle of an application.

Developers can continue to work with the IDE (Integrated Development Environment) of their choice. For example,

Visual Studio for Windows .NET, Eclipse, NetWeaver, JDeveloper or IntelliJ for Java, or 3270 emulators, IBM RDZ, Compuware TOPAZ for mainframes.

Mostly, those IDEs are also integrated with popular and well-accepted version control repositories, such as CVS, Subversion or Git.

With IKAN ALM, developers can stay in their comfort zone as very little will change for them. The only thing asked for is that they version their code in one of the supported Version Control Repositories.
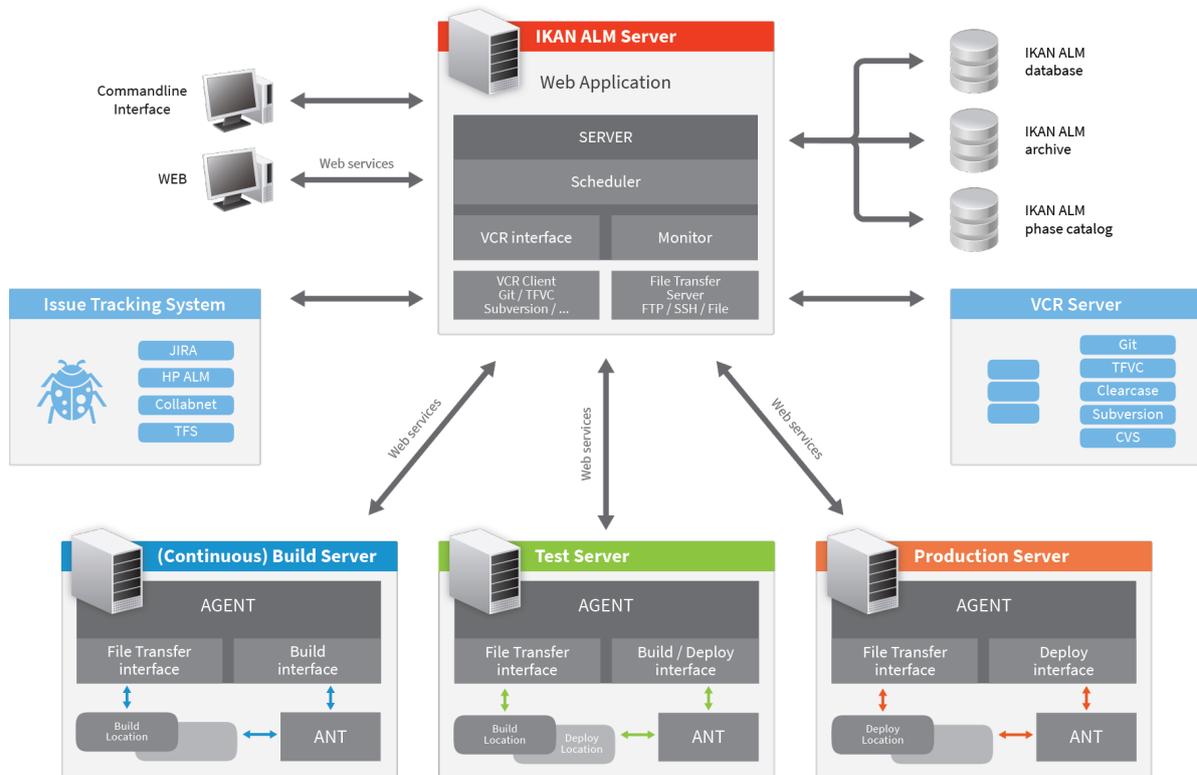


IKAN ALM is a web-based application with a server-agent architecture

For more information on the IKAN ALM architecture, see our IKAN ALM Architecture white paper:
 http://www.ikanalm.com/whitepapers/IKAN_ALM_Architecture.pdf

Next, IKAN ALM will complement the developer's IDE. Apart from integrating with Versioning and Issue Tracking systems, it offers the following main services:

## Life Cycle definition

Ability to define and clone your own project life cycle with a Build, Testing and Production Levels.

## Build process

Consists of what we call a number of core phases, solution phases and custom phases. Core phases are IKAN ALM phases needed to have IKAN ALM running, solution phases are phases that provide specific build functions for your environment and custom phases are phases built by you.

Examples of solution phases are phases for compiling COBOL, Assembler or PL/1 code on the mainframe, phases to support IBM Websphere or phases to create and deploy Docker containers.

## Deploy process

Consists of what we call a number of core phases, solution phases and custom phases. Core phases are IKAN ALM phases needed to have IKAN ALM running, solution phases are phases that provide specific deploy functions for your environment and custom phases are phases built by you.

Examples of solution phases are phases for stopping and starting application servers, restore Oracle Data Integrator scenarios, DB2 binds on the mainframe, deploying Docker containers, etc.

## Approval process

Next, there is the possibility to make your deployments approval-based and to be notified of any action executed by IKAN ALM.

It is very simple to set up projects with their specific lifecycle and related build and deploy phases for Bimodal Mode 1 and Mode 2. Also, once you have a sample project, a project can be cloned.
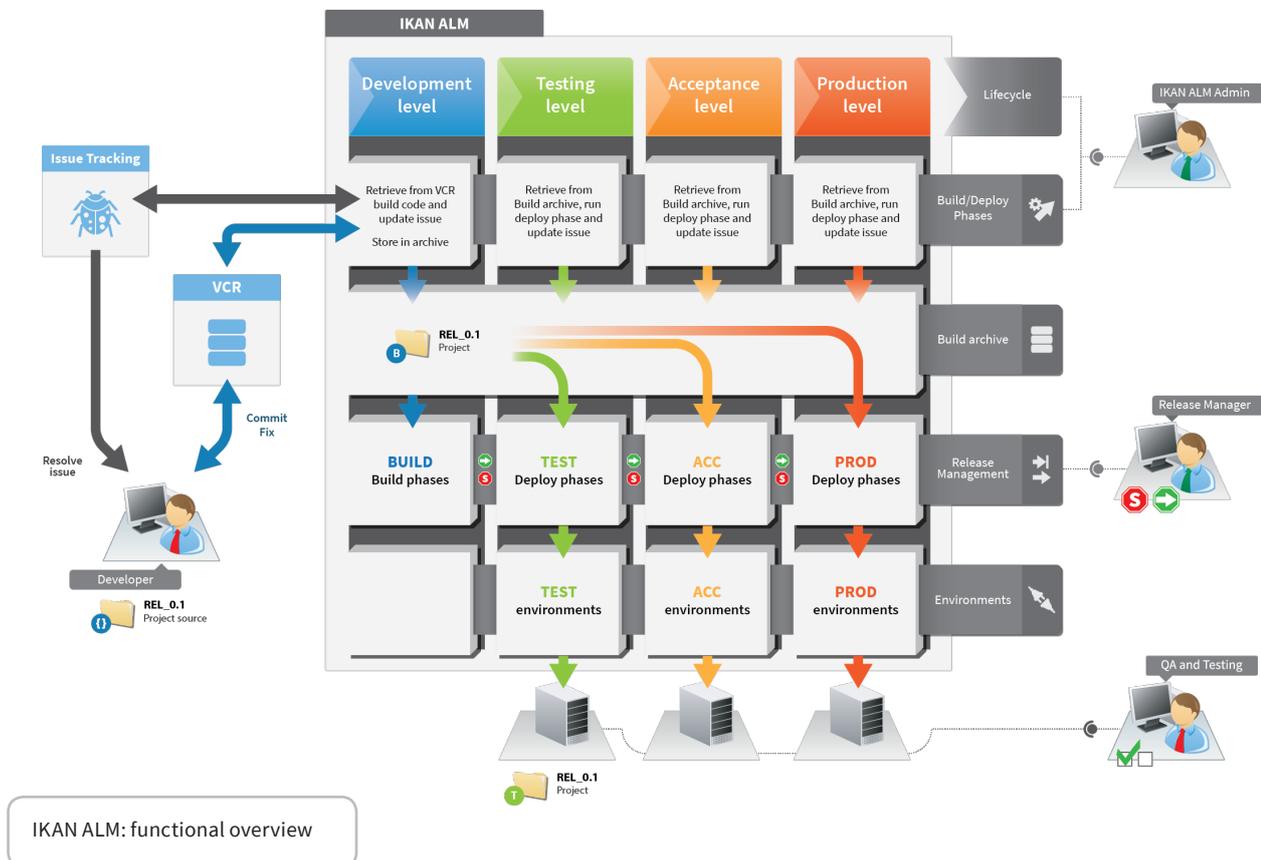


IKAN ALM: functional overview

# IKAN ALM ensuring Bimodal

In the following section, we will briefly describe three possible Bimodal implementations. All three have the following in common: a distinct lifecycle, automation of builds and deployments, improved traceability and enhanced communication between all stakeholders, which are the fundamentals of a professional way of working.

## Bimodal, mode 1: IKAN ALM and mainframe

A standard development process on mainframe looks similar to the one below:

### Requirements and Issue Tracking
Application requirements and issues are stored in Collabnet's Teamforge, BMC's Remedy or Atlassian's JIRA, …

### Development
Mainframe developers use to work with 3270 screens and store their code in mainframe PDSs (Partitioned Data Sets) instead of in standard Windows directories. In a PDS you can't version your files or members. That is why library management systems like Librarian or Panvalet, and, later on, more sophisticated systems with additional functionality like CA's Endevor or Serena's Change Man, where developed.

Nowadays, mainframes are often no longer seen as development environments, but mainly as machines able to cope with massive amounts of data and transactions. A major challenge is to make them part of today's new ecosystem where distributed environments and mobile play an important role.

All of this requires support for renovation of legacy applications and opening up the legacy for the future.

Another difficulty for the mainframe is to find young graduates that still want to code using a 3270 terminal. That is why companies like Compuware, IBM and EASIRUN have refreshed the mainframe development IDEs and offer TOPAZ , RDZ and COBEE respectively. Those IDEs are Eclipse-based environments with a lot of productivity enhancements for developers and which are attractive for young graduates.

### Versioning
Once the development is done, developers "commit" their code, just like .NET or Java developers, into a common version control repository like CVS, Subversion, Git or Team Foundation Server (TFS).

At this point, IKAN ALM takes over.

IKAN ALM has developed specific mainframe solution phases that will allow you to take the development of the mainframe and to steer the mainframe compile and deploy process from within IKAN ALM using non mainframe technology, but serving the mainframe.

### Automated build and compile
A sample compile process using the IKAN ALM build phases, runs like this:

- From your versioning system, you select the components you want to compile in a package.

- Based on the content of that package, a JCL (compile script for the mainframe) will be automatically generated. Possible criteria for that JCL can be: the language used (Assembler, COBOL, PL/1), databases used (VSAM, DB2, IMS, Datacom, IDMS,…) and whether CICS is used or not.

- That package (sources, copybooks,.. and JCL) is transmitted to the z/OS mainframe machine and the compile jobs are submitted.

Once the compile jobs have been executed, the generated files are loaded and the results are brought back to IKAN ALM and you move to the next step.

## Automated deployment to Test or Production environments

The next step is the deployment to a Test or Production environment.

During the deployment you can, if required, recompile or apply the necessary rules, such as specifying the correct DB2 database or CICS system or Debugger, needed to run in your test or production environments.

The process is similar to the compile process:

- The package (now including the load modules) is transmitted to the z/OS mainframe machine and the deploy or promote job is submitted.

- Once the job has been executed, the results are brought back to IKAN ALM.

More information on how IKAN ALM works with IBM mainframe can be found here:

- On modern mainframe development: http://www. IKAN ALM.com/whitepapers/Modern_mainframe_ development_and_ALM.pdf
- On IKAN ALM and the mainframe: http://www.IKAN ALM.com/whitepapers/Integrating_IKAN_ALM_ and_Mainframe.pdf


## Enhanced communication between all stakeholders

Pre- or post-approvals can be set on Test and Production levels. Those approvals enable a verification moment before or after the execution of the Level Request to Test or Production. On top of that, notifications can be sent out to the people involved.

This enhances the communication between the different stakeholders and improves traceability and quality control.

## Bimodal, mode 1: IKAN ALM and a standard distributed development

IKAN ALM itself is a Java application and IKAN ALM's development process is Agile and DevOps-based.

As an example, we refer to ourselves: all IKAN ALM development and release management is done using our own IKAN ALM solution.

### Development, Issue Tracking and Versioning

A standard Java development environment could look like this: an Eclipse-based IDE for development, Atlassian JIRA for requirements gathering, sprint and release planning, and Subversion or GIT as version control repository.

Once the development is done and the code has been versioned, IKAN ALM will take care of automating the build and deploy processes ensuring a reliable, controlled and fast delivery of your applications.

Let's take as an example our own internal IKAN ALM lifecycle which contains a Build process and processes ensuring the deployment to Test and Production environments.

### Automated build

The "Build" is not just "compiling the source code", it is a process that covers all the steps required to create a "deliverable" of our software. In the Java world, this typically includes:

- Generating and/or compiling sources.
- Compiling test sources.
- Executing tests, e.g., unit tests, integration tests, …
- Packaging (into jar, war, ejb-jar, ear).
- Running health checks (using static analyzers like Checkstyle, Findbugs, PMD, test coverage, etc.).
- Generating reports.

Best Build practices in a Bimodal environment require that all those steps are fully automated and run automatically and continuously. That is known as Continuous Integration, what we apply.

Using the IKAN ALM solution, this can be easily achieved by setting up the lifecycle and by defining the Build environment within that lifecycle using the IKAN ALM Phases. Those phases will take care of building the code, running tests, creating the jars and wars, running health checks and much more.

### Automated deployment to Test and Production environments

The next step, the deployment of the Java application, can be a stressful and long process. Errors during the deployment process will not only impact the Development and Operations department, but also the business side as failing and time consuming or unpredictable deployment processes result in a longer time-to-market and a bad perception of the software quality.

Sound Deployment practices include- amongst others:

- Support (your) best practices
- Middleware support
- Cross-platform support.
- Role-based security.
- Open architecture.
- Logging of deployment activities

Automating this process does not only speed up the deployment process. The deployments are more predictable, controllable and reliable, which leads to saving time and money.

To automate the deployment process, we use dedicated parameter-driven phases to set up the Test and Production environments. Those phases will automatically execute the deployment of the code and update the databases. Basically, we just need to adapt the parameters used for our test or production environment.

### Enhanced communication between all stakeholders

An important functionality to improve the communication between the different stakeholders are pre- or post-approvals. Such approvals enable a verification moment before or after the execution of the deployment to the Test or Production environment. On top of that, notifications can be sent out to the people involved. This enhances the communication between all stakeholders and improves traceability and quality control.

Through the approvals, both project managers, developers, marketing and management are kept informed of the actual status of the builds and deployments.

## Bimodal, mode 2: IKAN ALM and Docker experiment

To illustrate how IKAN ALM can be used for Bimodal Mode 2, a pilot project was created to see how Docker and microservices work.

An existing JAVA project running in Bimodal Mode 1 was used to start an experiment with Docker.
The initial project works with JAVA, an HSQLDB database and a Tomcat web server.

Changes made are:
- adapting the Build process by adding the Docker-specific files
- adapting the Deploy process by adding DockerCompose facilities
- building a Docker Container for HSQLDB and TOMCAT

### Development

From a developer's point of view, nothing will change. The developer can continue to use his favorite IDE for developing his JAVA application and, once his work is done, he can version the code in a versioning system like Subversion or Git. The version control repository structure needs to be adapted.



- etc
- hsqldb-database
- lib
- sql
- src
- tomcat-webserver
- build.xml
- deployProdLevel.xml
- deployTestLevel.xml
- docker-compose.yml
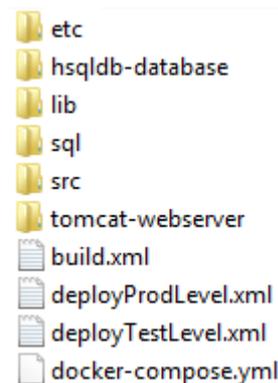
Subversion Project for Docker Phase Research

## Automated Build and Deploy phases

For the Build process, the current Build script needs to be modified to include Docker elements like yml and Docker Files needed for deployment to Test.

IKAN ALM will show a Build result that contains the Java WAR and the needed Docker files.

```
<!-- Copy Docker elements (yaml files, docker files, ...) to the target location -->
<echo message="copy docker elements to sourceroot = ${source}" />
<copy todir="${target}" includeEmptyDirs="true">
  <fileset dir="${source}" includes="*.yml" />
</copy>
<copy todir="${target}/hsqldb-database" overwrite="yes">
  <fileset dir="${source}/hsqldb-database"/>
</copy>
<copy todir="${target}/tomcat-webserver" overwrite="yes">
  <fileset dir="${source}/tomcat-webserver"/>
</copy>
```

Added Docker Elements to the Build Script

**Success** Docker_Customers / H_1-0 / CONTBUILD / Build# 2
17: Initiated by schedule 1
Created by Schedule on: 12/22/16 5:32:00 PM

Summary | Phase Logs | Results | Approvals | Issues | Sources | Modifications | Dependencies

◄ Back | Refresh | Build History

### Results

**Build: 5**

| | |
|---|---|
| **Build File Name** | Docker_Customers_H_1-0_b2_CONTBUILD_win.tar.gz |
| **File Size** | 20.8 MB |
| **Archive Status** | Present |

✔ Download Build Result
- 📁 Docker_Customers_H_1-0_b2_CONTBUILD_win.tar.gz
  - 📁 hsqldb-database
  - 📁 sql
  - 📁 tomcat-webserver
  - 📄 ALMDemo_customers.war, 21791696B, 5:32 PM
  - 📄 deployProdLevel.xml, 2699B, 5:32 PM
  - 📄 deployTestLevel.xml, 3459B, 5:32 PM
  - 📄 docker-compose.yml, 902B, 5:32 PM

Build Result in IKAN ALM

Next, the Build Result can be deployed: the IKAN ALM DockerCompose Phase will stop and remove the existing containers with a Docker-compose down (this will not fail if there are no containers running yet, but will just log the errors).

It will build and start the containers as specified in the Docker-compose.yml file, first the database and then the webserver.

```
dockerComposeVersion:
    [echo] Lauching 'docker-compose version' to get the version ... Is unix: true
    [exec] docker-compose version 1.9.0, build 2585387
    [exec] docker-py version: 1.10.6
    [exec] CPython version: 2.7.9
    [exec] OpenSSL version: OpenSSL 1.0.1t  3 May 2016

dockerComposeDown:
    [echo] Lauching 'docker-compose down --rmi all' to stop and remove existing containers and images
    [exec] Stopping target_webserver_1 ...
    [exec] Stopping target_database_1 ...
    [exec] [2A[2K
    [exec] Stopping target_webserver_1 ... done[2B[1A[2KStopping target_database_1 ... done[1BRemoving target_webserver_
    [exec] [2A[2K
    [exec] Removing target_webserver_1 ... done[2B[1A[2KRemoving target_database_1 ... done[1BRemoving network target_da
    [exec] Removing image tomcat-webserver

dockerComposeUp:
    [echo] Lauching 'docker-compose up -d --build --force-recreate' to build, recreate and start the containers from the yml file
    [exec] Creating network "target_default" with the default driver
    [exec] Building database
    [exec] Step 1 : FROM blacklabelops/hsqldb
    [exec]  ---> cc4dd7484b04
    [exec] Step 2 : MAINTAINER IKAN\wob
    [exec]  ---> Using cache
    [exec]  ---> 8aecbafdc620
    [exec] Step 3 : COPY hsqldb.script /opt/database/hsqldb.script
    [exec]  ---> d6af583c50ad
    [exec] Removing intermediate container bff5efa4d5ad
    [exec] Step 4 : ENV HSQLDB_USER ALMTEST
    [exec]  ---> Running in f87151d5e4c2
    [exec]  ---> 7aad1c6aa3a7
    [exec] Removing intermediate container f87151d5e4c2
    [exec] Step 5 : ENV HSQLDB_PASSWORD almtest
    [exec]  ---> Running in af56546574aa
    [exec]  ---> 458b78a705b2
    [exec] Removing intermediate container af56546574aa
    [exec] Building webserver
    [exec] Successfully built 458b78a705b2
    [exec] Step 1 : FROM tomcat:7-jre8-alpine
    [exec]  ---> 1ede6d40df32
    [exec] Step 2 : MAINTAINER IKAN\wob
    [exec]  ---> Using cache
    [exec]  ---> fb3a702c5272
    [exec] Step 3 : COPY customers_test.war $CATALINA_HOME/webapps/
    [exec] Creating target_database_1
    [exec]  ---> 11542385e5ce
    [exec] Removing intermediate container cb39c280bdf5
    [exec] Successfully built 11542385e5ce
    [exec] Creating target_webserver_1

BUILD SUCCESSFUL
```

IKAN ALM Deploy Log File

### Enhanced communication between all stakeholders

To enhance the communication between the developers, the test and QA departments and the management team, notifications can be sent out automatically, and pre- or post-approvals can be specified on Test and Production levels, enabling a verification moment before or after the deployment to Test or Production. For Bimodal Mode 2, as it is for innovation, you can limit your lifecycle to Build and Test environments.

# Benefits of the implementation of an ALM solution

## Which organizations benefit?

- Any organization can benefit from ALM.
- ALM does not only enforce your processes, it also guarantees that the process is repeatable, reliable and documented, this both for Bimodal Mode 1 and Mode 2.
- It will free your key people of repetitive, less interesting tasks.
- The size of your team does not matter, both small and large teams can benefit. Of course, the larger the team, the greater the benefits.
- IKAN ALM will allow you to renovate and open your legacy environments and, at the same time, it will help you to experiment and innovate.
- With IKAN ALM, Bimodal Mode 2 is supported and allows you to easily move to Mode 1 once the experiments are over.

## Benefits for the organization as a whole

**Gartner**® The positive outcome of an ALM solution on an organization is rather impressive. Gartner mentions three principal values that can be expected from adopting ALM:

1. Enhanced management transparency & visibility
2. Effective execution of challenging processes
3. Better results for the business.

The benefits for an **organization as a whole** fall down in two categories.

### Tangible (direct financial impact)

- Improved cost and budget management ability

- Reduced time spent to build and deploy applications in production

- Improved time-to-market

### Intangible (more difficult to make tangible)

- Improved release consistency and quality
- Enhanced service to customers
- Timeline improvement of application releases
- Less human mistakes
- Ability to control the development process at any moment
- Maximization of stakeholder satisfaction
- Optimized collaboration between teams
- Creation of an Agile IT environment

If we look at the **benefits for each of the stakeholders**, we can distinguish four important elements: requirements, versioning, build and deploy. The requirement phase can mainly be linked to the business analysts, the versioning benefits are significant for developers, IT managers and the communication and collaboration through the team, the build benefits are mainly for the release managers and the deploy benefits are mainly applicable for the deployment engineers.

## Requirements (Business analysts)

- Better interpretation of requirements
- Visibility and traceability between requirements, tasks, related code and deployments
- Associated client and business collateral with each requirement, including emails, documents, graphics and other related information
- Real-time project status tracking
- Smoother communication and coordination
- Software solution accommodates real business need

## Versioning (Developers)

- Imposes a standard way of working for the entire team
- Allows traceability, promotes accountability and makes it easier to find the right person to solve a problem
- Quickly generate of a list of the changes made, making it easier to provide the information on what has changed, and why, from version to version
- No time wasted on old code
- Enhanced team collaboration and communication
- Allows a rollback to earlier versions
- Allows to deliver revisions, updates and cross-platform versions

## Automated Build (Release managers)

- Significantly improves the quality of your product
- Minimizes the number of 'bad builds'
- Accelerates the compile and link processing
- Eliminates redundant tasks
- Eliminates dependencies on key personnel
- Saves history of builds and releases in order to investigate issues
- Minimizes integration risk
- Less time-consuming

## Automated Deploy (Deployment engineers)

- Earlier and more frequent feedback from testers & end-users
- Less human errors
- Significantly increased user confidence
- Creation of an Agile development environment
- Accelerated time-to-market
- Deployment phase is not dependent on one person anymore, more persons can deploy (developers, testers, users etc.)
- Transparency and accountability by centralized deploy repository
- Smoother communication and coordination
- Increased flexibility in resource needs and planning
- Less downtimes
- Less time-consuming

# Summary

IKAN ALM can be used as an enterprise-wide solution for Bimodal, covering any environment or platform. With one and the same platform, mainframes, Windows or Unix (Linux) environments can be managed, be it for Bimodal Mode 1 and Mode 2.

IKAN ALM comes with standard build and deploy phases for many environments. Implementing those phases is just a question of collecting and configuring the required parameters, such as paths, user IDs and passwords, to obtain fully automated processes.

For Bimodal Mode 2, it is fairly simple to integrate your required build and deploy scripts into IKAN ALM.

**The key questions to answer are:**

- Is the code versioned or can the code be versioned?
- What are the company's building standards?
- What are the company's deployment standards?
- What life cycle do you use?

# Conclusion

A uniform, standards-based ALM or Bimodal solution like IKAN ALM brings the following value to your organization:

- Proper storage and versioning of all software objects
- Cost reduction in both development, testing and deployment
- Automated and repeatable tasks
- Reduced risk thanks to automated, repeatable build and deploy processes
- Improved application quality
- Faster time-to-market
- Less resources and time needed for build and deploy to test and production
- Better communication between stakeholders
- Better visibility over the whole process
- Well-defined and automated development and deploy process
- Support for Bimodal Mode 1 and Mode 2
- One common lifecycle management platform

# For more information

For more information about IKAN ALM and how we can help you to achieve DevOps within your organization, refer to:

**IKAN ALM architecture:**

http://www.ikanalm.com/whitepapers/IKAN_ALM_Architecture.pdf

**IKAN ALM Phases concept:**

http://www.ikanalm.com/documents/HowToALM%205.6_UseAndDevelopPhases.pdf

**EasiRun Europa GmbH (Partner)**
Stockheimer Weg 20
61250 Usingen - Germany
phone +49 6081 9160-30
vertrieb@easirun.com
www.easirun.com

**Hogwart (Partner)**
Viale dei Mille, 7
20129 Milano - Italy
phone: +39 02 8417 9551
www.hogwart.it

**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.ikanalm.com