Kobee

Using Kobee for DevOps

Bridging Development and Operations

Table of contents

Bridging development and operations

Software application development, including the implementation and maintenance hereof, involves many complex processes combining a series of roles, tools and deliverables. Those processes need to be efficient, repeatable, auditable, predictable, secure, maintainable and cost-effective.

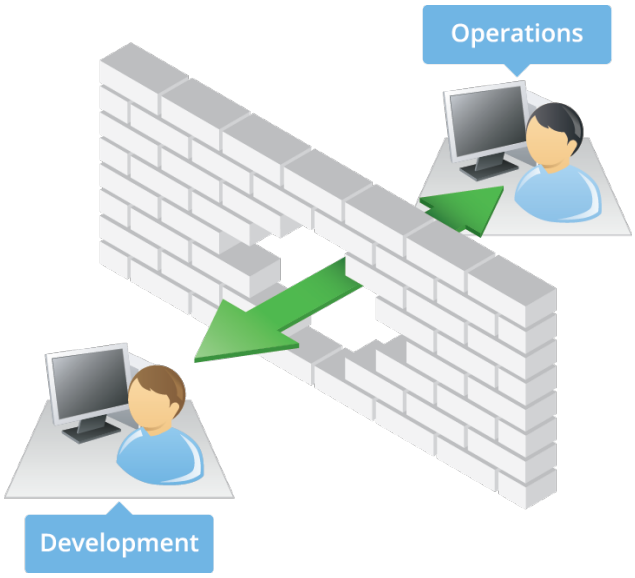
Many companies have already implemented various levels of automation to streamline those processes, such as Waterfall, Agile, Continuous Integration or Continuous Delivery.

Today, another methodology gains in importance, namely DevOps

DevOps focusses on the interdependence of users, software development and IT operations. It stresses the importance of communication, collaboration and integration between all relevant stakeholders be it the business, software developers or people responsible for operations and production.

To optimize the interdependence and communication, a controlled, reliable and automated overall solution achieving the highest level of automation (Continuous Deployment) is essential.

This DevOps way of working can be achieved by implementing an efficient **Application Lifecycle Management (ALM) solution.**



DevOps improves the communication between all stakeholders

A modern ALM solution is an **integrated set of tools** for:

- Requirements Management
- Versioning
- Automated Build & Deploy procedures (to test and production environments)
- Lifecycle and Approval Management

Furthermore, the ALM solution adheres to the relevant process standards and governance rules.

Modern ALM solutions have to be:

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.

The list below indicates the problems software development organizations are facing without the use of a modern ALM / DevOps solution.

Without DevOps

Development teams lose too much time with the build and deploy work, too many avoidable mistakes are being made and too much effort is necessary 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

Advantages of DevOps

Less time and resources needed for developing, testing and deploying (multi-environment) applications

Reduced risk of human errors

Improved efficiency and productivity

Improved application quality and release reliability

Faster time-to-market

Improved customer satisfaction

Using Kobee for DevOps

Kobee is a cross-platform web-based solution for Application Lifecycle Management. It combines DevOps initiatives (continuous build and continuous integration) and lifecycle management to support the complexity of service-oriented architectures and highly distributed systems.

Being a modern ALM solution, Kobee can be used in any development environment, be it a mainframe or distributed environment. It also integrates with less commonly used environments, such as ODI.

The ability of Kobee to communicate with a variety of Issue Tracking and Versioning tools, saves software development companies from adapting to new tools. The tools already in use can be integrated within the overall application life cycle.

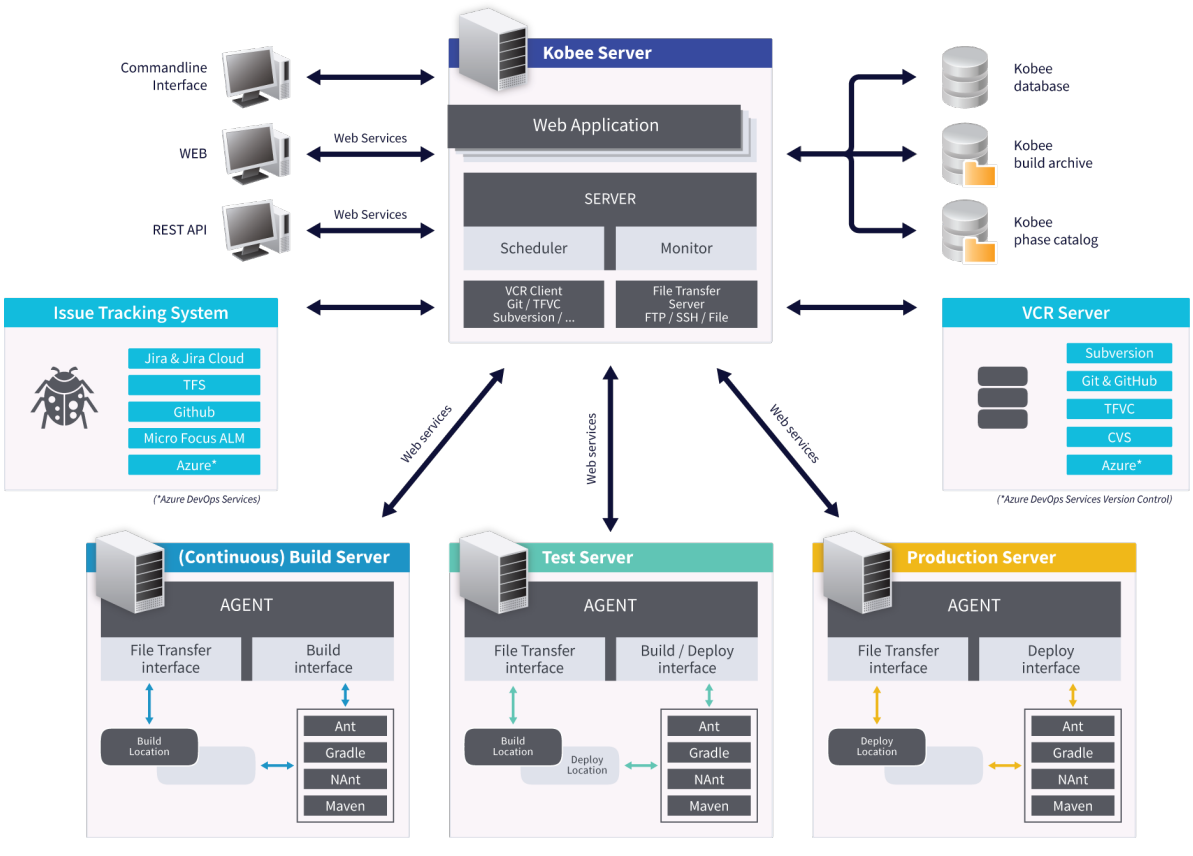
Kobee starts where the actual development ends. This simply means that when a developer commits his code to the version control repository, Kobee will notice this change and launch the related build and deploy processes.

> "Kobee can be used in any development environment, be it a mainframe or distributed environment. It also integrates with less commonly used environments, such as ODI."

Kobee architecture and functionality

Automated build and deploy procedures will solve most of the issues mentioned before 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.

Kobee is a platform and environment-independent Application Lifecycle Management solution aimed at ensuring a DevOps way of working throughout the whole lifecycle of an application. Kobee is a web-based application with a server-agent architecture.



Developers use 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. Kobee complements the developer's IDE and offers the following main services:

Life Cycle definition

Ability to define 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 Kobee phases needed to have Kobee 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.

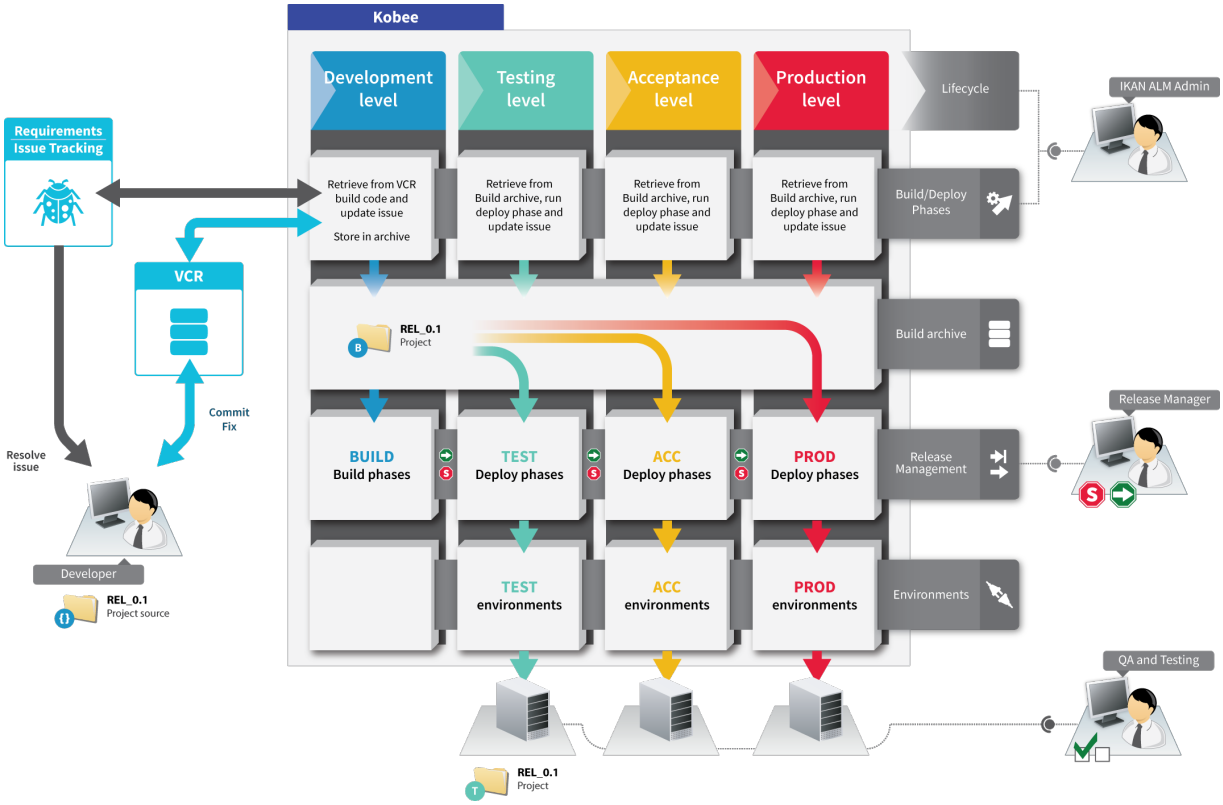
Deploy process

Consists of what we call a number of core phases, solution phases and custom phases. Core phases are Kobee phases needed to have Kobee 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, etc.

Approval process

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



DevOps solutions for diverse environments

In the following paragraphs, we will briefly describe three possible DevOps solutions: mainframe, distributed and non-standard environments. All three share automation of builds and deployments, improved traceability and enhanced communication between all stakeholders, which are the fundamentals of a DevOps way of working.

Kobee 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.

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 and IBM have refreshed the mainframe development IDEs and offer TOPAZ and RDZ 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, Kobee takes over.

Kobee 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 Kobee using non mainframe technology, but serving the mainframe.

Automated build and compile

A sample compile process using the Kobee 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 can be: ASSEMBLE, COBOL, PL/1, SDF2, DB2, CICS, … using IBM or Endevor compilers.
- 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 Kobee 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 (the load modules and associated listings, dbrms and plans) is transmitted to the z/OS mainframe machine and the jobs are submitted for updating Binds, Cics Phasein or Debugger, or for managing other specific objects.

- Once the jobs have been executed, the results are brought back to Kobee.

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.

Kobee and distributed development

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

As an example, we refer to ourselves: all Kobee development and release management is done using our own Kobee 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, Kobee 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 Kobee 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 DevOps 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 Kobee solution, this can be easily achieved by setting up the lifecycle and by defining the Build environment within that lifecycle using the Kobee Phases. Those phases will take care of compile 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.

Kobee and a non-standard environment

Kobee not only serves the standard development environments. It also complements less commonly used development environments, like for example Oracle Data Integrator (ODI), which is Oracle's data extraction, transformation and data load tool.

Specifically for ODI, we offer a versioning component and dedicated solution phases for the ODI Build and Deploy processes.

Development

In an ODI environment, development is usually done using ODI Studio. The following important steps are versioning the developed code and automating the build and deploy processes.

Versioning

A versioning component (VCR4ODI) has been developed in collaboration with our partner D&T who is specialized in ODI.

VCR4ODI is the link between ODI Studio and Subversion. When committing ODI objects to Subversion, VCR4ODI creates a file for each object and a second file with the dependencies for that object. The restore operation can just restore the object itself or the object with all its dependencies. As such, VCR4ODI manages the ODI objects dependencies.

Thanks to an advanced locking mechanism, VCR4ODI also provides support for concurrent development and will send out warnings if an object, or objects, are locked by another user.

In its latest release of ODI (release 12), Oracle has also added a versioning component. The Kobee ODI Solution Phases will not only work with our own versioning component, ODI customers may also use Oracle's new versioning component.

Once the code has been developed and correctly versioned, Kobee will take care of the Build and Deploy processes. Those processes can be fully automated using the dedicated ODI Solution Phases.

Automated Build and Deploy phases
The ODI Build phase will collect the ODI objects and their dependencies from Subversion, will create a release and will put that release in an archive.

When deploying to a Test or Production environment, the customer has the choice to deploy development objects and/or so- called ODI scenarios.

Other Oracle environments supported
Next to ODI a customer can easily add other Oracle development environments for building and deploying Oracle applications, such as OWB or Fusion Middleware.

Enhanced communication between all stakeholders
Just as for the mainframe and distributed environments, Kobee will also improve the communication between all stakeholders using approvals and notifications.

The benefits of implementing DevOps

An integrated ALM/DevOps solution brings a bunch of advantages for the company as a whole. Within that company, every stakeholder will also experience significant time and money savings. It goes without saying that productivity gains for one stakeholder have an impact on the other stakeholders and finally on the entire company.

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

Kobee can be used as an enterprise-wide solution for DevOps, covering any environment or platform. With one and the same platform, mainframes, Windows or Unix (Linux) environments can be managed.

It 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.

The only key questions to answer are:

- Can the code be versioned? or is it already versioned?
- What are the company's building and deployment standards?
- What life cycle do you use?

Conclusion

The positive outcome of an ALM solution on an organization is rather impressive. Not only the organization as such benefits, each stakeholder will experience his own benefits.

A uniform, standards-based ALM / DevOps solution like Kobee doesn't only enforce your processes, it also makes sure that the process is repeatable, reliable and documented.

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.

Gartner mentions three principal values that can be expected from adopting ALM:

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

ALM and DevOps are hot topics, there is already a lot of literature about these topics out there. In our appendix you will find a list of industry expert quotes.

For more information

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

Kobee architecture:
https://www.kobee.io/whitepapers/kobee_architecture.pdf

Kobee Phases concept:
https://docs.kobee.io/how-to-use-and-develop-phases-en/5.7/UseDevelopPhases.html

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

Modern mainframe development:
https://www.kobee.io/whitepapers/modern-mainframe-development-and-ALM.pdf

Kobee and the mainframe:
https://www.kobee.io/whitepapers/integrating-kobee-and-mainframe.pdf

Appendix: Expert opinions

ALM increases team collaboration and reduces cycle time by 70%
(Case study: AIG insurance group leverages ALM to attain IT performance architecture advantage, Zdnet.com, D. Gardner interview)

With all these collaboration and reporting improvements, daily tasks completion rates have risen by 20%
(Case study Lac Viet Computing Corp)

The completed software projects on time double
(Case study Lac Viet Computing Corp)

ALM increases the productivity of developers by 50%
(Case study Société GRICS)

ALM reduces deployment resources by 50%
(ALM summit 2013, M. Uppal and E. Sher for a large healthcare organization BUPA)

Entire team productivity increases with 30%
(Forrester report, J.W. Lipsitz and M. Evangelista, 2013)

ALM increases the number of weekly deployments by 300%
(ALM summit 2013, M. Uppal and E. Sher for a large healthcare organization BUPA)

ALM increases the number of deployed environments by >200%
(ALM summit 2013, M. Uppal and E. Sher for a large healthcare organization BUPA)

ALM reduces the deployment resources by 50%
(ALM summit 2013, M. Uppal and E. Sher for a large healthcare organization BUPA)

ALM doubles the functionality of the delivered software
(ALM summit 2013, D. Fletcher and C. Remillard)

ALM increases test coverage by 30%
(Case study thetrainline.com, thoughtworks)

ALM improves time-to-market with 50%
(Case study thetrainline.com, Thoughtworks)

ALM triples the values for the customers
(ALM summit 2013, D. Fletcher and C. Remillard)

ALM and the increased visibility ensures an increase of quality by 25% and decrease of development time by 10% to 20%
(Case study Comet Solutions Inc.)

37% faster delivering their software to market
(QSM Associates, The agile impact report, 2008)

According to Forrester an ALM investment repays itself after 3 to 6 months

(Forrester report , J.W. Lipsitz and M. Evangelista, 2013)

On average the deployment time shrinks from 2 hours to 10 minutes and can be done by 1 person instead of a team of release managers

(ALM summit 2013, M. Brittain)

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

info@kobee.io
www.kobee.io