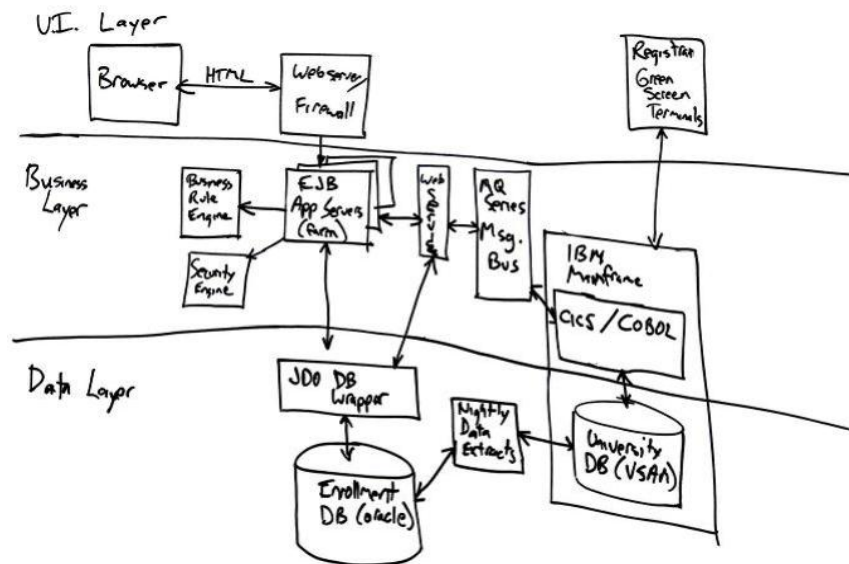


WHITEPAPER

Java Deployments in an Enterprise Environment



Vincent Partington
CTO, XebiaLabs

Table of contents

Summary	3
1 Why deployment matters	4
1.1 About this paper	4
1.2 Introduction	4
1.3 Development and IT operations impact	5
1.4 Business impact	5
2 Pitfalls for a reliable deployment process	7
3 Overcoming the pitfalls	10
3.1 Requirements for a Deployment Automation solution	10
4 Deployit – automated deployment for Java applications	12
4.1 Architecture of Deployit	12
4.2 Using Deployit	13
4.3 Benefits of using Deployit	14
5 Start automating your deployments today!	15
5.1 About XebiaLabs	15
Appendix - Relevant literature	16

Summary

The proper deployment of Java applications can be a cumbersome and tedious process. Errors during the deployment process not only impact the Development and IT operations side of an organization, but also the business side. Unpredictable deployment processes result in longer time-to-market. In addition, they negatively affect the (perceived) software quality.

In order to reach an optimal deployment process, a company usually has to overcome one or more of the eight deployment pitfalls listed in this whitepaper. Examples of such pitfalls are:

- Using lengthy deployment manuals that are incomplete and not up-to-date.
- Working with middleware environments that are not in-synch.
- Having an incomplete view of all existing deployments.

These pitfalls cause serious problems for a smooth and reliable deployment process.

Automating deployments using a Deployment Automation solution is a significant step forward in eliminating these deployment problems.

This whitepaper describes (also) eight requirements that can be used as a rule-of-thumb when considering a Deployment Automation solution. State-of-the-art Deployment Automation solutions provide:

- Comprehensive middleware support.
- Best-practice deployment scenarios.
- An extensible platform that protects current investments.
- Scalability up to several 1000s of servers and applications.
- Cross-platform support.
- Role-based security.
- An open API for creating plug-ins and other customization purposes.
- Reporting & analysis capabilities for insight & analysis.

Deployit, as delivered by XebiaLabs, is a proven Deployment Automation solution that helps organizations like airline company Air France KLM with their deployment challenges. Deployit provides Air France KLM and other customers with substantial savings in terms of time and money. In addition, SLAs can be met and deployment processes are much more predictable since repetitive tasks have been automated. Moreover, time-to-market for new applications is now shorter since it was possible to eliminate unnecessary waiting times and error-prone hand-over moments.

1. Why deployment matters

1.1 About this paper

Without accurate, efficient and effective deployment there is no running software application. We cannot state it more simply than this. Software deployment is a crucial step in any software development project. Still, more often than not it is undervalued and underestimated. This whitepaper describes the importance of flawless deployment processes, the business benefits of proper deployments for both IT and non-IT departments, the root causes of less-than-flawless deployment processes and some recommendations for solving these in a practical manner.

1.2 Introduction

The following real-life case story about one of the world's largest airline companies, Air France KLM, demonstrates that a proper deployment process is very important for a company:

For Air France KLM, Internet technology plays a crucial role in internal processes as well as in communication with its customers. Passengers buy tickets and choose their seats online, and pilots use web applications to access weather forecasts and to file their flight plans. Air France KLM estimates it typically works on about 35 different projects simultaneously, with a total of around 200 Java EE application deployments per week.

The weak spot in the development process concerned the transition of developed code from one phase to another. About 70% of all software deployments faced these problems. Deployments required a lot of manpower, which was not only inefficient but also subject to errors. Air France KLM therefore decided that they needed a more powerful deployment management system to help reduce repetitive work, boost productivity and improve quality.

Read <http://www.xebialabs.com/sites/default/files/1/KLM%20customer%20story.pdf> for the full story.

Before we analyze the drawbacks to a flawed deployment process and dive into the root causes, we will first define what we mean by “deployment”:

Definition: Deployment is the “series of activities needed to make an application available to end users”. This includes the full scope of deployment activities: de-installation of the old software packages, installation of the new software packages, updating the data source(s), modifying firewall settings, reconfiguring the web server and restarting all services involved.

Or rephrased in more simple terms: deployment is everything needed to ensure that if a developer, tester or end user opens his browser and types in *www.my-application.com*, he will see the application up and running, fully functional with (live) data.

In the remainder of this whitepaper, we refer to this definition whenever we use the word *deployment*.

1.3 Development and IT operations impact

Portability, the ability of software to be transferred from one environment to another, is an integral part of the ISO-9126 software quality standard. This makes deployment, as a corner stone of portability, a crucial element in every software development project. The major pain point for developers when it comes to deployments is that their application can behave very unpredictably when it is run in conjunction with other applications for the first time. If a developer had an easy way to deploy the application on his own development environment before handing it over to the test department, that would be preferred, since it would definitely save a lot of time, discussions and arguments.

Poor deployment processes impact not only Development, but also IT operations:

- Less agility in releasing new features. As a consequence, you'll end up with a small number of big-bang deployments rather than a continuous stream of incremental deployments.
- Unpredictability. This results in higher contingency costs than required, since you need to have people stand-by in case something unforeseeable might go wrong.
- Unplannability. Being unpredictable also makes deployments hard to plan: a deployment of only a few hours can be scheduled during the evening, a deployment that runs the risk of taking a full day cannot.
- Loss of talented people, since tedious deployment tasks limit people in doing other (more interesting) work.
- Poor ability to learn from mistakes due to the lack of consistent error analysis.
- Risk of becoming too dependent on key resources (the ones who created the deployment scripts).

1.4 Business impact

Traditionally, when people talk about deployment, they relate it to the IT operations department. However, deployment impacts the business side of organizations as well.

Several studies show that a significant amount of time and money is spent on deployment of software. And modern web applications consisting of a set of loosely-coupled web services only have led to increased deployment complexity. This is supported by one of our surveys that indicate an average increase in maintenance efforts of 15% for companies that deploy web applications.

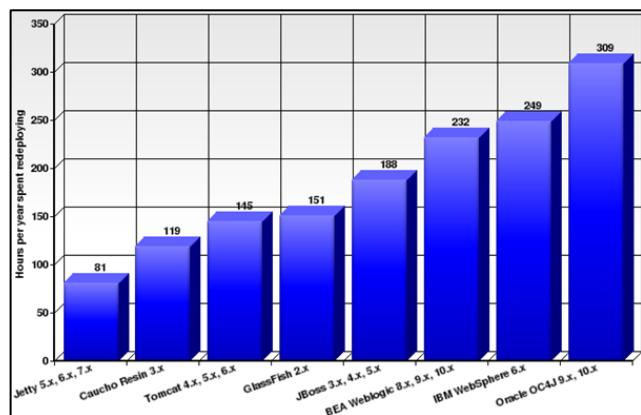


Fig. 1: Time spent on deployment per year per container (hours)

Another survey carried out by an independent analyst firm amongst 250 U.S. based companies found that companies spend, on average, \$852,000 per year to create, maintain and support deployment scripts.

These studies show that problematic deployment processes directly affect businesses' bottom line, since they cause:

- High business continuity risks when unplanned downtimes occur or when deployments take longer to complete than expected.
- Delayed go-to-market resulting in missed opportunities. For new initiatives or in agile markets, this also involves the risk of being too late in the market to claim a substantial market share.
- Poor end customer satisfaction. Promises made to customers cannot be met, since new releases take too long to become available. As a consequence, the underlying business drivers for the end customer (e.g. better order fulfillment, better invoice management or cheaper stock replenishments) cannot be met in time.
- Deteriorated market perception of your company in case you have to postpone a scheduled release due to deployment issues.
- Postponed ROI and extended pay-back time on your application development initiatives, having a negative impact on the company's cash flow position.

In summary, the deployment of software has a large impact on both the IT operations and the business operations side of a company, therefore improvements in this area are worth serious consideration.

2. Pitfalls for a reliable deployment process

Deployment processes are more complex than they look at first sight. How hard can it be to deploy a piece of software onto a server? Well, if you consider that the average web application deployment not only requires the installation of the EAR/WAR file itself, but also needs changes to the JDBC data source, the correct configuration of HTTP ports and virtual hosts and some changes in the SQL scripts, (re)configuration of the firewall, updates to the static content on the web server and stopping and restarting of the web server, deployment suddenly looks a lot more complex and error-prone than it originally seemed. Actually, the **root pitfall** for all successful deployments is: *underestimating the complexity of deployments*.

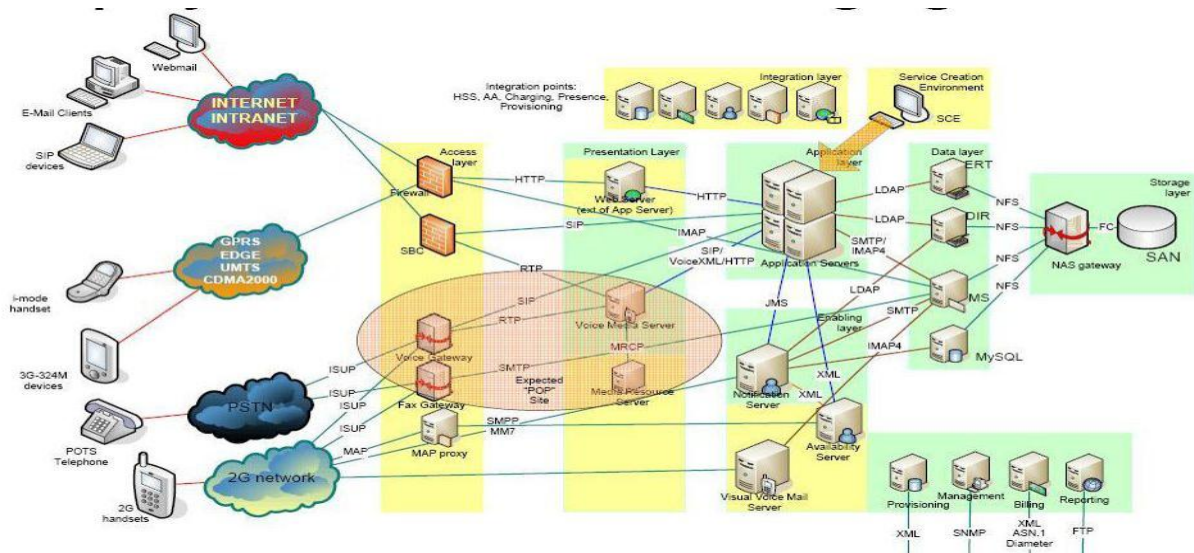


Fig. 2: Enterprise deployment is a complex task

Below, we will discuss eight pitfalls that cause organizations to have suboptimal deployment processes, resulting in the drawbacks outlined in the previous section.

Pitfall #1: Infrastructure design ignores deployment

The first pitfall consists of neglecting deployment aspects when defining and designing a new IT infrastructure. It is not uncommon to set up a fully redundant application server infrastructure for load balancing and fail-over purposes. Typically, it will be split up in different administrative domains, each requiring a separate deployment activity. But contrary to elements like performance, security, up-time and license costs, deployment is not taken into account in the design phase. Still, the ease of deployment very much depends on your infrastructure design.

Pitfall #2: Deployment is overlooked by developers

According to the ISO-9126 standard, one of the aspects of software quality is the ease of making changes to deployments. However, more often than not this is not included in the application designs. In some way this is understandable as well, since deploying an application in an isolated development environment is relatively simple to do. But deploying

it safely, reliably and in a scalable manner in a production environment is quite something else.

The software architecture chosen can be a further impediment to smooth deployments. Componentized, modular designs are usually welcomed for ease of (code) maintenance, but can become a deployment nightmare if interdependencies are not architected and documented well, especially when components are run on different machines.

Pitfall #3: Development and deployment environments differ substantially

Another pitfall is developing your application on a different (middleware) environment than the one on which it will eventually be deployed. Usually developers work on a stand-alone desktop environment. They develop and test against the JDK version as installed on their machine, using their preferred servlet engine, their own local database with a limited set of test data and their preferred browser. Production environments, however, typically consist of a constellation of (virtual) servers and databases, often partly hidden from each other through a DMZ.

Pitfall #4: Unused deployment manuals

Most system administrators and other IT experts recognize this pitfall all too well. Extraordinary elaborate deployment manuals, usually up to 50 pages (or even more), contain endless sequences of deployment steps that need to be followed one-by-one. This is a rather tedious, time-consuming and *human-unfriendly* way of working. And as a result, people often put these manuals aside and don't read them at all. Besides, it is virtually impossible to keep these oversized manuals up-to-date each time a change occurs in the physical infrastructure, middleware or the application itself.

Pitfall #5: Middleware environments are out-of-synch

Another common pitfall is working with out-of-synch middleware environments on your DTAP (development, test, acceptance and production) servers. For instance, differences in how you set up load balancing and clustering usually lead to different outcomes when deploying the same software application. Unsynchronized middleware environments cause deployments to become very unpredictable. And although clouds and virtualization can sometimes overcome this pitfall, it is not always a panacea.

Pitfall #6: Incomplete view of all existing configurations

The lack of a complete overview of all existing software and hardware configurations is another pitfall. Most organizations do not have a clear picture of which activities have to be done on what systems, and in what order. What firewalls need to be reconfigured, what web server needs to be updated first, does the database driver also have to be upgraded? The availability of information on all systems in the IT landscape, including their configurations and (network) properties in use, is a challenge for a lot of organizations.

Pitfall #7: Deployment itself is not tested

Most development teams test their own software application well, removing show-stoppers and other serious bugs before release. The 7th pitfall is not about the testing of the application itself, but the absence of proper testing of the deployment of the software. Ease-of-deployment is an important requirement for enterprise software. That's why it is also important to (re)test the deployment on multiple environments. Testing the deployment includes more than updating the EAR file by hand and then only checking if all services can be started. It also includes actually opening a few end user screens, entering some test data and then trying to save these into the database.

Pitfall #8: "Fully automated deployment is simply not possible"

Last but not least, this pitfall is perhaps the hardest to overcome, because it is part of a set of deeper beliefs about what can be done and what is impossible. Many middleware experts claim it is not possible to fully automate the deployment process. This is an important reason why error-prone and unpredictable deployment processes continue to prevail. Manual (or partially manual) deployment processes remain dependent on key experts with specific application knowledge, long after those experts have been assigned to new projects (or have even left the company). Crucial deployment knowledge is in the heads of people, not stored and automated in systems. And the generally accepted idea that deployments cannot be fully automated ensures that this situation perseveres.

3. Overcoming the pitfalls

What we learn from these eight pitfalls is that deployments are much harder to carry out well than might seem at first sight. It's not uncommon to succumb to one (or more) of these pitfalls. At the same time, as this whitepaper will try to demonstrate in the remainder of this section, *you can get deployments under control*. A suitable deployment automation product can address the pitfalls and overcome the deployment problems they cause. But how do you recognize a suitable deployment automation solution?

3.1 Requirements for a Deployment Automation solution

Below we outline the eight most important requirements a Deployment Automation solution should meet.

1. *Comprehensive middleware support*

Deployment automation tooling should not impose restrictions on the middleware that you use. It should be flexible in adopting the middleware of your choice, be it e.g. Oracle WebLogic, IBM WebSphere Application Server or JBoss Application Server.

2. *Adoption of best-practice deployment scenarios*

When looking for a Deployment Automation solution, it is recommended to investigate if the solution also supports deployments without downtimes to minimize application downtime, next to a full "clean deployment". Another capability to look for is the support of complex cluster and web server configurations. Needless to say, the automation deployment tool should also support a wide range of artifacts, data sources, libraries, JMS resources and static content.

3. *Extensible*

As companies evolve over time, so do IT environments. In fact, they are subject to constant change. This implies that you're looking for a Deployment Automation solution that can cope with these future changes. As an example, one could decide to build bespoke additions to existing COTS applications. Or perhaps develop a completely new application in-house. The deployment solution should be able to work with these future software additions and ideally be shipped with its own development platform. This platform allows you to make custom extensions to your deployment configurations, and should have an open and intuitive API. This way, you ensure you have investment protection in your deployment solution and it can grow with your company's demands.

4. *Scalable*

Enterprise organizations literally have 1000s of servers and applications to look after. The deployment automation solution should not limit your organization's ability to manage deployments across these kind of server and application landscapes. It should also be scalable and flexible enough in that it supports new middleware versions as they become available, so that you can freely upgrade your IT landscape whenever needed.

5. *Cross-platform support*

It needs no further explanation that a deployment tool for Java EE applications should work across all platforms supported by the JVM. Preferably, it should run in a non-intrusive mode, leaving no footprint on your (production) servers. Being able to start and monitor Java EE

deployments via a browser is another point for consideration, since it enables your organization to remotely set up and control your application deployments.

6. Customizable

Even though deployment best-practices are both relevant and important to use, there is usually no 100% fit with each and every (company-specific) IT landscape. This implies that the Deployment Automation solution should allow fine-tuning of your deployment workflows as well as support the creation of application-specific plug-ins, e.g. through a well-documented plug-in API. This API should also cater for in-house developed applications and custom extensions.

7. Secure

Applying role-based permissions for managed objects and environments should be supported by the deployment solution. It saves time and money and prevents mistakes if the deployment automation tool can integrate with your security infrastructure using standard protocols like LDAP. On the other hand, security permissions should be flexible enough to allow for so-called “self service deployments” by developers (e.g. to a test environment).

8. Transparent

Large-scale deployment environments often lack insight into what happened and who did what, when. It would be beneficial if the deployment automation tool can provide an instant overview of the state of your operational landscape. In addition, auditing and traceability support is required to keep track of your change history. Being able to generate progress and monitoring reports on all deployment activities will help your organization to further streamline and optimize your IT operations.

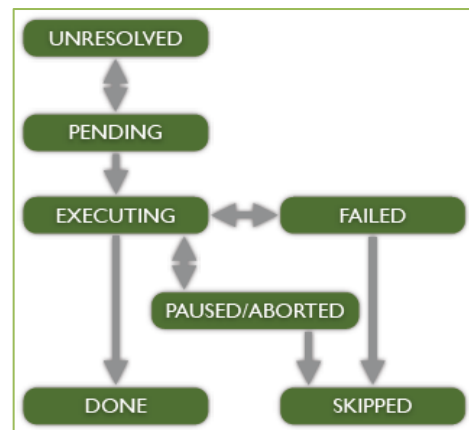


Fig. 3: Transparency in deployment processes is required

These eight requirements can be used as a rule-of-thumb for quickly assessing if a Deployment Automation solution is worth further consideration. Additional requirements may and usually will apply, depending on the specific demands that you have for your deployment environment, security regulations and preferred way-of-working.

4. Deployit – automated deployment for Java applications

In today’s world the process of bringing Java EE applications from development into production needs to be automated to run businesses in a more predictable, reliable and cost-effective way. This is no longer a utopia. The mental pitfall of assuming that it is not possible to fully automate the deployment process has been proven wrong by XebiaLabs.

XebiaLabs believes that everyone with sufficient rights should be able to deploy applications within a few mouse clicks. It has materialized this belief in its flagship product called Deployit.

Deployit offers end to end Java EE deployment automation to IBM WebSphere, Oracle WebLogic, JBoss and TomCat environments. Deployit not only deploys Java EE applications, it also configures other middleware components such as web servers, databases, security proxies (like Tivoli Webseal and DataPower), firewalls and load balancers. Deployit integrates with build frameworks like Maven, continuous integration tools like Hudson and Bamboo and with IBM Tivoli Change and Configuration Management Database and HP Universal CMDB for updating hardware and software configuration information.

4.1 Architecture of Deployit

Deployit is based on a **Unified Deployment Model**. Deployments across different types of middleware involving various deployment artifacts are carried out in a consistent, uniform way. This helps organizations to efficiently manage their Java application deployments in a heterogeneous middleware environment.

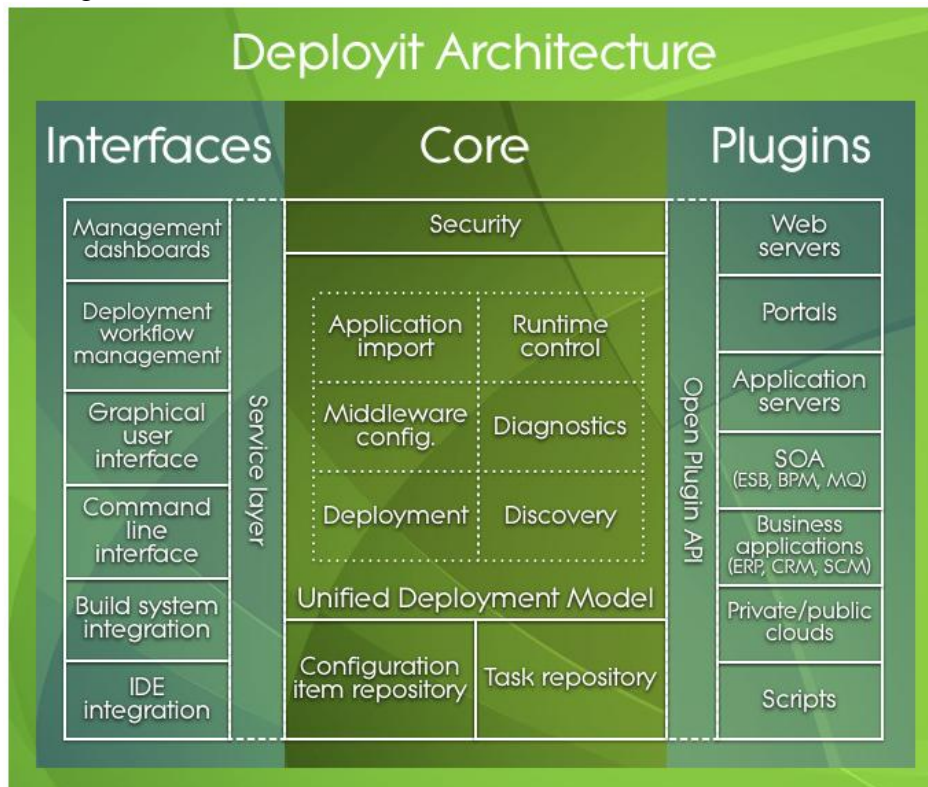


Fig. 4: Deployit’s architecture

From the **task repository**, the user can select a fixed set of tasks. These tasks describe e.g. how to configure the middleware. Furthermore, they define how runtime control is handled for starting/stopping middleware, how to diagnose problems during deployment of applications and of course, how to actually deploy applications to the middleware environments. The task repository also keeps track of all executed task instances, so that these can be used afterwards for audit and analysis purposes.

The cross-platform, web-based user interface provides a clean and smooth user experience, whilst the Command Line Interface makes it **easy to integrate** Deployit into your existing build and integration. Deployit also offers several management dashboards for monitoring progress.

Furthermore, Deployit offers out-of-the-box integrations with leading change management solutions like HP Universal CMDB and IBM Tivoli CCMDB to update the configuration information if needed. Companies that do not use a commercially available CMDB can rely on Deployit's deployment lifecycle workflows which offer support for light-weight change management processes. For developers, Deployit provides integrations with Maven, Bamboo, Hudson and Eclipse.

Deployit ships with a number of standard middleware plugins. Deployit currently comes with plugins for e.g. IBM WebSphere AS, IBM WebSphere Portal, Oracle WebLogic, Oracle Service Bus, JBoss Application Server and Tomcat. When needed, you can develop your own custom plugins against the **open plugin API** using Deployit's SDK.

4.2 Using Deployit

Deployit is a role-based solution that can be used in parallel by various users or departments, and fully supports a DTAP philosophy. As the picture below illustrates, a company typically works with multiple versions of the same software application at the same time.

Developers work on upcoming releases, and use Deployit for deployments to a stand-alone team development server. Meanwhile, test departments use Deployit for deploying not-yet-released applications on a test server to be able to run their system and integration tests. They also deploy a test version in an acceptance environment, which typically consists of a shadow image of the production environment. Finally, the end user only works with the approved version that was deployed onto a production server by the system administrator.

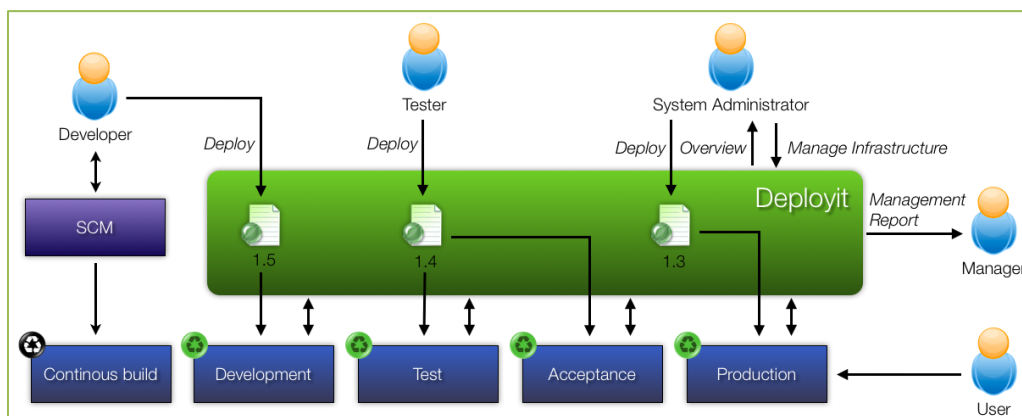


Fig. 5: Deployit manages deployments of multiple versions in use by different people

All deployment activities and progress made can be followed by IT management, who use Deployit to create all kind of deployment statistics and management reports. Based on these reports, future development and test cycles can be further optimized. This results in more predictable processes, saving time and money by eliminating deployment and configuration issues.

4.3 Benefits of using Deployit

Deployit is a robust, comprehensive and easy-to-use deployment automation tool. Unlike custom scripts and generic process engines, Deployit's intelligent deployment engine guarantees a safe and reliable deployment of your Java applications in an ever-changing environment.

This leads to actual savings in your IT operations, as can be proven by ROI studies that we carried out at two of our customers. One customer saves approximately € 900,000 annually by using Deployit for 60 different applications, maintained by an IT operations team of 20 FTEs and a development staff of 50 FTEs. Another customer saves € 390,000 annually through the use of Deployit, based on the deployment of 28 Java applications. This customer has outsourced the development of its applications to India, but all deployments are done by an in-house IT operations team consisting of 6 full-time employees.

The table below sums up other benefits of working with Deployit:

Role	Benefits offered by Deployit
Developer	<ul style="list-style-type: none"> - Faster feedback cycle because self service deployments are possible. - Waiting time for system administrators to deploy applications is reduced to zero.
System administrator	<ul style="list-style-type: none"> - More parallel deployments possible per unit of time. - Less repetitive work, since all necessary steps have been automated. - No hand-over moment with developers anymore: developers have already self-serviced their application deployment beforehand in a test environment.
IT management	<ul style="list-style-type: none"> - SLAs are met. Unnecessary delays (caused by unsuccessful deployments) at the end of the development cycle are eliminated. - Predictable and plannable deployments. - Improvement in software product quality, since deployments can be tested and monitored. - Motivated team since tedious, repetitive tasks have been fully automated.
Business line management	<ul style="list-style-type: none"> - Faster time-to-market for new or enhanced software applications due to less waiting time and fewer hand-over moments. - Guaranteed business continuity due to support for hot deployments. - Lower contingency costs and less stand-by personnel needed when deployments to production environments are scheduled. - Faster ROI and shortened pay-back time.

Table 1: Benefits of Deployit

5. Start automating your deployments today!

In this whitepaper, we have discussed the business and IT pain points associated with traditional deployment processes. We have also seen the pitfalls and ways to avoid these by choosing a state-of-the-art deployment automation tool like Deployit.

If you are interested in learning more about Deployit and what it could mean for your specific situation, please do not hesitate to contact us. We will be pleased to spend time with you over the web, on the phone or in person to discuss your needs and demands.

You could also download the free Deployit Personal Edition. This edition ships with a single user license. The Personal Edition can be downloaded from <http://www.xebialabs.com/deployit-personal-edition-request>.

And in case you are still not convinced that deployments can really be automated, let us know! We are always interested in better understanding the views and opinions of our peer deployment experts.

5.1 About XebiaLabs

XebiaLabs is the leading provider of Deployment Automation solutions. With its flagship product Deployit, XebiaLabs automates Java EE deployments in an enterprise environment. Deployit allows organizations to manage their middleware environments by predictably executing and automating their Java application deployments.

XebiaLabs was founded in 2008, and serves various customers around the world. Headquartered in the Netherlands, XebiaLabs is an international company with development and sales offices in the USA, France, Germany and India. In other countries, XebiaLabs works through a network of local representatives.

For more information, please visit www.xebialabs.com or contact one of our offices.

Appendix – Relevant literature

Survey results on Java EE Containers: Heaven or Hell?, Zero Turn Around,
<http://www.zereturnaround.com/blog/java-ee-container-heaven-hell-survey-results/>

ISO 9126 – Software Quality, International Organization for Standardization,
http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=22749

The Extended ISO Model of Software Quality,
<http://www.serc.nl/quint-book/index.htm>

Forrester Blog for Vendor Strategy Professionals, JP Garbani, Forrester,
http://blogs.forrester.com/vendor_strategy/2009/05/automating-the-deployment-of-applications.html

Wikipedia on Software Deployment,
http://en.wikipedia.org/wiki/Software_deployment

Automating the JEE deployment process, Indic Threads.com Conference on Java Technology, Vikas Hazrati, December 2009,
<http://www.slideshare.net/xebiaindia/automating-the-jee-deployment-process>

Top 8 Java EE deployment pitfalls, Java Forum Stuttgart, Andrew Phillips, July 2009,
<http://www.xebialabs.com/webinar-8-pitfalls-when-deploying-java-applications>

So what is a deployment really? Robert van Loghem,
<http://blog.xebia.com/2009/07/08/so-what-is-a-deployment-really/>

Deployment vaak het ondergeschoven kind, Vincent Partington,
http://www.computable.nl/artikel/ict_topics/development/3060842/1277180/deployment-vaak-het-ondergeschoven-kind.html

KLM/AIR FRANCE speeds up Java application deployment with XebiaLabs' Deployit,
<http://www.xebialabs.com/sites/default/files/1/KLM%20customer%20story.pdf>



For more information, please contact us or visit: www.xebialabs.com

Xebia Labs | Utrechtseweg 49 | 1213 TL Hilversum | The Netherlands | Phone: +31 (0)35 6729067 | Email: info@xebialabs.com