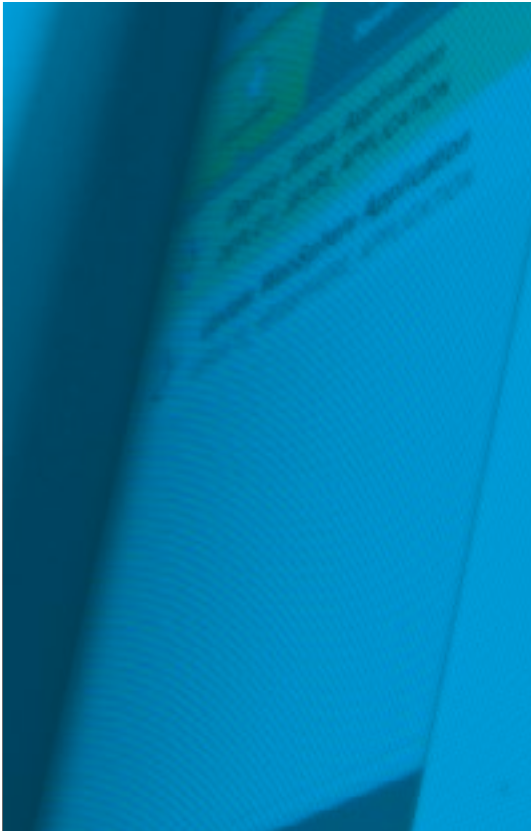




Practical DevOps Using ARA

Packaging, workflows and a generic deployment model.



Complexity Changes Everything

Applications and services are made up of more components and integrations today than they were just a few years ago. Coupled with a much more competitive business world, this complexity has changed everything for both development (Dev) and operations (Ops).

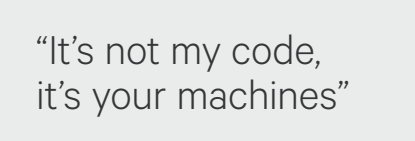
Dev is pressed to deliver more functionality, faster, focusing on small, incremental changes, using agile methodologies and tools such as continuous integration.

The Ops environment is much larger these days, with many more servers and services than ever before. The production environment, which used to be all physical, is now almost entirely virtualized, so that Ops can spin up servers in an instance. Ops are also managing many more resources today than ever before.

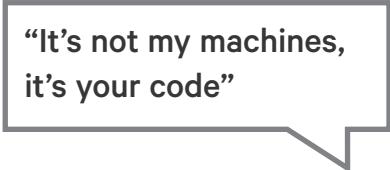
As development focuses on fast delivery by chopping up services, developing more in parallel, and making applications work as a whole, more configuration steps are required and more interdependencies are created.

Also, with Agile, developers are spending a lot less time on 'internal' projects that could have simplified some aspects of complexity. Ops hates to fix developer problems and developers don't want to spend time on configurations and deployment. Over time, this has become a major finger-pointing exercise between Dev and Ops. *"It's not my code, it's your machines"* vs. *"It's not my machines, it's your code"*.

A 'DevOps' movement has emerged that puts forth the idea that "Dev should think like Ops and Ops should think like Dev". This is necessary to address this problem and requires Ops to participate early in the development cycle. For Ops to use advanced configuration management solutions, such as Puppet and Chef, also helps greatly. But we are never going to be 'home free' until we tackle the last mile. This whitepaper focuses on automating application deployment across DevOps, the capabilities that are required, and how those capabilities apply to either Dev or Ops or both throughout the lifecycle.



"It's not my code,
it's your machines"



**"It's not my machines,
it's your code"**

Growing Application Matrix and Losing Control

The application lifecycle is an ongoing iterative process where applications constantly get developed, tested, patched, upgraded and re-developed across a number of ‘stages’, including: development, functional testing, integration testing, staging and production. While stage names and numbers vary between applications and enterprises, the process is roughly the same. Moving from stage to stage always involves deploying parts, or possibly the entire application, to a new set of servers, also known as an ‘environment’. Additional requirements include uninstalling previous versions in some cases or installing to a parallel location (for testing) in others.

The DevOps challenge increases every time a new component is added. With every variation in a configuration or target environment, there is an ever-growing matrix of components, environments and deployment requirements. When you consider the number of different groups, across different geographies that develop different applications and that ultimately have to directly or indirectly integrate with each other, many organizations are simply losing control. The past few years have seen a number of publicly visible failures to mission critical systems across all verticals. Banks, communication service providers and some of the Internet juggernauts have been brought down by uncontrolled or unexpected changes. “Faulty Deployments” and “Botched Upgrade Processes” are some of the headlines that explained how these failures occurred.

Application Release and Development Using ARA – Speed and Control

The DevOps movement is focused on solving the issues mentioned above. DevOps started from an understanding that the intersection between the development and operations world has become the biggest reason for IT to have seemingly lost control. DevOps teams initially focused on fixing the ‘culture bridge’ and the disconnection between development and operations, motivating both organizations to fuse as much as possible by collaborating throughout the application lifecycle. This is continuing today and will be an indispensable piece of the DevOps world in the future.

However, it is evident that scale is also a big issue, specifically in relation to the presence of an order-of-magnitude more servers in the operations environment. Evidently, tools to manage configurations and baselines on a large scale are needed, having stimulated development of tools such as Puppet Labs’ Puppet, OpsCode’s Chef, and others. A critical issue is being tackled with these tools, assuring basic infrastructure specs in even the largest scale environments. And while these solutions are excellent for standardizing large server environments, they are less suitable for application processes that cross environmental barriers or are cross-tier by nature. You could

say that configuration management tools use an ‘I need’ scheme, while ARA uses a ‘user-manual’ scheme. One is more environment-bound (i.e. ‘What exactly do I need on each server in my environment?’), and the other is more lifecycle-bound (i.e. ‘What are the correct steps necessary to move this application version/patch/release from this environment to the next?’). In the grand scheme of things, ARA provides control and effectiveness to the last layer of the IT stack, and for the most business-critical of them all – the applications. ARA uses key technology capabilities that are specifically designed to fit into Application Lifecycle and Management technologies; predominately packaging, workflows, and the deployment model.

Packaging, Workflows and the Generic Deployment Model

Enabling an automation solution to truly extend across the application lifecycle and address organizational processes and task transformation requires a high level of engineering and some key capabilities to ensure the solution is usable by both development and operations. It must provide a simple way to achieve its goal – to fully automate all application release and deployment tasks from development to production. Packaging mechanisms allow for the separation of content from logic. Workflows replace manual operations and scripts and the deployment model ensures that runtime settings and variables are both handled effectively and do not create unnecessary complexity to the solution.

Packaging

Packaging is a fundamental requirement of any Application Release Automation solution. Every release, from the smallest patch to the largest system upgrade, is made up of artifacts that originate from multiple sources (e.g. repositories, build servers, file shares and FTP servers) and each requires exact versions to match the current deployment. Binary files, configuration files, installation scripts, package installers of various kinds, as well as database scripts, can all be part of every new deployment. In fact, the probability of human error or script error during deployment always grows as the matrix of the number of artifacts and target servers in the environment grows. Humans aren’t great at doing the same task over and over again and scripts are subject to uncontrolled change and faulty executions by tired administrators. An ARA packaging mechanism should connect to multiple sources and let users define package content from each source. The version and content of each file is always validated and new versions of the entire package are created automatically with every revision of contained artifacts (e.g., when a new build is created). Packages represent a version container of an application or service. Packages are treated as living entities and can be assigned a status, that in turn can trigger workflows. Promotion paths assure

that a package can only be moved in a gradual and controlled way. Finally, packages are the foundation for providing a reliable and constantly updated inventory for the organization that has the full details of what is deployed and where, at any moment in time.

Workflows

Workflows are the ‘workhorses’ of ARA. They replace manual tasks that are required to install, upgrade, patch or remove an application, and can be reused, time and again, across any environment that the application lives in. Workflows are created visually on a canvas, much like a ‘Visio’-type block diagram. Actions are assembled from a huge library of existing actions for most of the common application containers and servers.

Examples below illustrate the breath of infrastructures and tools for which out-of-the-box actions can be assembled:

- Application servers such as: Weblogic, WebSphere, and JBoss/IIS
- Database servers, like Oracle and SQL-Server
- Integration servers such as BizTalk
- OS commands, package managers and source control systems.

Out-of-the-box actions provide a standard and quality-assured way to consistently deploy new changes to various servers. Out-of-the-box actions also benefit from the ability to be executed using different credentials (impersonation) that are needed during the process without compromising security, and finally, are automatically audited 100% of the time.

Other important workflow characteristics are:

Generic – Workflows should be completely decoupled from any environment setting, credential or permission data, as well the content that it executes. This allows the same workflow to be constantly tested and reused with each new completely different package version. It also allows the same workflow to be executed across any environment, from test to production.

Rollback functionality – Rollback is essential to assure that an application, specifically in production, is never left in an inconsistent or inoperative state should anything fail to deploy correctly. Building rollbacks for every workflow is both difficult and ineffective. For example, assuring that you can

activate the rollback workflow after each workflow step will infinitely complicate your workflows. Instead, as is the case with Automic ARA, each out-of-the-box action already comes with built-in (generic) rollbacks. An effective rollback sequence can be activated at any moment without affecting the design of any single workflow.

Version-controlled – Just as application components and services are developed by multiple teams in parallel and result in different versions, so are deployment workflows. In fact, it can be the same teams that develop code that also develop workflows. The only way to work in parallel across multiple teams is through a version control system. It is therefore a critical ability for your ARA platform to enable multiple teams to collaborate and work effectively with multiple deployment workflows and multiple versions.

Secure – Deployment workflows touch the core of the business, especially when executed in production environments. As with most other processes in Enterprise IT, a segregation of duties is required, sometimes by law. For example, a person who designs the workflow should not have permission to execute that very workflow in certain environments, such as production, and vice versa. A person who has authority to execute or approve an execution of the workflow should not be able to even accidentally modify it and create a potential business outage.

Built-In (Generic) Deployment Model

A great deal of errors are caused by differences in the environment settings of a particular deployment, and not by workflow bugs. The problem is that software and applications are very susceptible to even the smallest variations in environment and settings. A simple example can be the differences in root directories between origin and destination servers. This can be a Windows root drive difference, such as c:\ vs. e:\, or a Unix mount point, such as /opt vs /var. A more subtle difference can be in port configuration of a service, execution credentials, and even the configuration of the environment, such as cluster sizes. Accounting for these important variables during a deployment execution is a tedious and error-prone process. Users have to verify and adjust for all of them on every target machine they touch, and the system as a whole. When you consider tens or even hundreds of servers that need to be updated, the matrix quickly becomes unmanageable manually and scripting requires hard programming that is many times done on a 'one-off' basis.

Automic's ARA technology has a built-in model that lets operators and developers control run-time settings in a single place. The system will propagate the correct settings to the correct executions at the correct moment, automatically. The model takes care of the variations that exist between

environments (e.g., number and types of servers), server settings (e.g., directories and version differences) and configuration data (e.g., variable values). The model includes deployment targets, profiles, logins and more. The model not only assures that the correct settings are propagated at the right time to the correct execution, but also play a crucial role in keeping workflows simple and manageable. In fact, not having a built-in model will result in very complex workflows and having to store the runtime settings in external sources, such as XML or databases. This can be both insecure and complicated from a workflow perspective, as the user needs to read, parse and branch execution flows as part of the workflow.

Conclusion

Application release automation is the only way to gain true control of a complex matrix of applications, releases and environment variations. The inability to fully control and audit change, deployment and release processes results in IT failures that can cost the business millions and put DevOps in a rough spot. ARA is the last hurdle in the DevOps process.

To overcome this last hurdle, it is imperative that an ARA solution provides these three critical capabilities:

- Packages and promotion paths, so you can be confident that what is deployed is correct.
- Workflows, so you have a standard and quality-assured way to consistently deploy new changes.
- Deployment models, to ensure the correct setting and configurations are applied to each environment.

Whilst there are other factors to consider, only by ensuring that your solution encompasses the above capabilities can you be certain that you will be able to meet the ever-increasing demands of the business to go faster, for less cost and stay in control.

For more information or a product demonstration please visit www.automic.com