Automic ARA & Jenkins

How to use Automic Release Automation (ARA) to complement current Jenkins installations

Automic™ White Paper

Summary:

Jenkins/Hudson has a substantial footprint in the build & continuous integration (CI) space. This paper will answer the two most common questions from customers who are familiar with Jenkins: “Why not use Jenkins for a DevOps scenario?”, and “How is Automic ARA different?”. It will also address how to use Automic ARA to complement existing Jenkins installations and create an end-to-end deployment process.

What to look for when researching an ARA / DevOps solution

DevOps is neither a market nor a methodology – it is a philosophy. DevOps espouses the idea that IT organizations should model themselves so that agility becomes a way of life. Development and operations teams work in collaboration with the single goal of agile delivery to production. DevOps originated with a focus on continuous integration and delivery. Its success depends on the use of tools to automate and streamline configuration, release and monitoring processes relating to deployment. ARA solutions aim to address the needs of organizations seeking to adopt the DevOps philosophy.

What makes something an ARA or a non-ARA solution? The May 2013 Gartner report, ‘Know the Application Release Automation Vendor Landscape to Shortlist the Best Vendors for your Organization’, identifies four main capabilities that need to be present in an ARA solution:

1. Automation
2. Environment Modeling
3. Workflow Management
4. Integration Adapters

In the same paper, Gartner also defines four key functions an ARA solution should include:

- Deployment of actual data, application code or artifacts
- Deployment of the specific configuration settings for each individual environment (such as development, test, quality assurance, staging and production)
- Process workflow design for automation tasks, people steps or both
- Environment modeling and/or provisioning binaries (such as middleware, DBs and application servers)

Automic defines five competencies that need to be addressed in any ARA scenario:

1. **Packaging** – creating a collection of applications and configurations that must be deployed at the same time
2. **Dependency Analysis –** modeling full application dependencies between application and components

3. **Promotion Model –** defining a path to deliver tested packages in to more critical environments
4. **Deployment Model –** using steps to install package contents and configurations in to operating environments
5. **Compliance and Visibility –** adherence to processes, validating deployed applications and configurations

These five ARA competencies are implemented as three capabilities – the What, How and Where of a deployment. The three Automic ARA capabilities perfectly map Gartner's key functions of an ARA solution.

- **Packages** – What version of the application and related configurations will be deployed?
  - ***Deployment of actual data, application code or artifacts***
- **Workflows** – How will the steps of approval, deployment and validation for this application be defined?
  - ***Process workflow design for automation tasks, people steps or both***
- **Models** – Where will the application be deployed and which environment-specific configurations need to be addressed?
  - ***Environment modeling and/or provisioning binaries***
  - ***Deployment of the specific configuration settings for each individual environment***

Why not use Jenkins in a DevOps scenario?

Jenkins is a CI tool with automation capabilities that have historically been used for builds and basic deployments to development sandboxes. Because Jenkins can chain together any type of script and select a single target environment, some administrators have tried to extend its use beyond the development team to meet their growing and more complex enterprise-wide ARA needs. Often, scripting extensions into Jenkins, which might already be available in-house, can be an initial step towards a full ARA process. Unfortunately Jenkins is not architected to be an ARA solution, and thus misses most of the core ARA capabilities and competencies DevOps teams need. When compared with the functions defined by Gartner for ARA, Jenkins is lacking in the areas of workflow management, environment modeling, process workflow design and the deployment of the specific configuration settings for each individual environment.

Jenkins is capable of executing scripts, which allows it to excel at building software and then automatically push the resulting build to a development sandbox. ARA offers more than automatic pushing of a build to development: ARA solutions use constructs to manage and scale deployments of complex applications across multiple environments – environments that often have different properties and configurations to manage.

For Jenkins to get close to what an ARA solution offers, a lot of custom scripting and maintenance of those scripts is required. It is not unusual to have one or more talented 'automation superheroes' on a team doing nothing but maintaining the Jenkins script-based deployment system. The problem a DIY deployment system faces is twofold:

1. The Jenkins scripters are typically developers who are not collaboratively working with operations to craft a holistic solution.
2. Regulations surrounding separation of duties concerns are violated. Heavily modified scriptextended systems also tend to be weak in implementing standardizations that enable auditability and visibility to what is occurring during deployments.

With consistent and clean application updates becoming more critical to the business, one bad deployment from a DIY system that development owns and few can access or understand not only opens up IT organizations to risk, but also public embarrassment, a drop in consumer confidence and, of course, financial loss.

When comparing Jenkins capabilities to the Gartner ARA standard, Jenkins is lacking in the areas of workflow management, environment modeling, process workflow design, and the deployment of the specific configuration settings for each individual environment.

How does Automic ARA meet the four main Gartner capabilities?

Automation

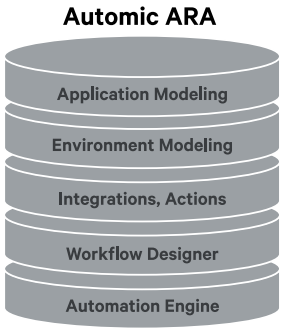
The first component to consider in an ARA solution is a workflow engine, and the foundational component of Automic ARA is just that: its platform. Automic, formerly UC4, has a 20+ year history in the IT automation space. With an Automation Engine (AE) capable of scaling to millions of tasks per day, conditional logic, calendaring, queuing and environment reservation capabilities, which extend into ARA. The platform is extremely configurable for any automation scenario and fast to deploy with one agent per server enabling any of its automation capabilities.

Workflow Management and Integration Adapters

The core Automic automation platform extends into two other major capabilities of an ARA solution:

Workflow Management and Integration

The design of Automic ARA allows workflow definition and integration capabilities with all enterprise systems to be controlled in one place. One AE controls all – but with multi-tenancy that allows for up to 9,999 different clients across the enterprise.



All Automic automation processes are implemented as workflows. Automic ARA Workflows are composed using a simple drag-and-drop interface. Each Workflow comprises of a series of Actions (which can also be referred to as automation steps or integration adapters). Each Action in an Automic Workflow is designed to execute a discrete function, interfacing with any OS or an application. Figure 1 is an example of a simple Workflow composed of four Actions.

Automic ARA includes over 500 out-of-the-box Actions, each with a defined integration point to do everything from moving a file to rebooting a server, updating an incident in a ticketing system and deploying a schema update. In addition to predefined Actions, custom Actions can also be built, which in turn can integrate with **any system using a Web Service, API or Command-line Interface**. Actions only need be created once – they are then available for re-use in any workflow in any client.

Figure 1: The Automic ARA Workflow Editor

Automic ARA is a **master orchestrator** – an Automic ARA implementation does not look to rip and replace, it looks to organize processes across disparate systems and teams into one fullservice automation hub touching any number of operations systems and applications.

Environment Modeling

A critical ARA consideration is the Modeling. The implementation of modeling in Automic ARA controls the most challenging part of any implementation: the configurations.

Configurations, or incorrect configurations, can make deployments difficult to manage and hard to debug. The complexity of applications is often reflected by all of the configurations that need to be managed. Web servers can often have tens, if not hundreds, of specific configurations that need to be managed every time an application is deployed. An ARA solution should alleviate the worry of how to manage configurations and free administrators from hours in front of an administrative management console and/or manually updating and juggling properties files.

Automic modeling is divided into two logical areas: application modeling and environment modeling.

Application models define what the application looks like, including the technology stack, component breakdown and the corresponding application general configurations. Environment Models define the server stack (physical, VM, cloud), deployment targets and environment- specific configurations.

Breaking up the model into two aspects gives achieves granular configuration control. Nobody wants to have to apply the same configuration on 100 objects if all of those configurations are exactly the same. Thus in Automic ARA if a configuration is the same across all instances of a deployment in Automic ARA, it would characterize that as an Application level configuration in the application model. Environmentsensitive configurations, such as database names, web server definition strings and ports, are stored separately in the Automic ARA Environment Model.

At runtime, an Automic workflow will take the configurations on the application model with the configurations from the environment model and dynamically deploy the application. If a new configuration, or new server is brought into the mix then only a model update is required; the Workflow and Deployment package remain untouched.

Which additional Automic ARA capabilities will lead to a repeatable, reliable deployment?

Packaging and Promotion

Automic ARA does not deploy builds, it deploys packages. A package is the payload which needs to be deployed and it may or may not represent a build. For example, it could be a stand-alone patch from a vendor.

In a CI integration scenario, a package in Automic ARA is often correlated directly to a build. The promotion path of the build/deployment, as well as which individuals or groups have the right to trigger a promotion across the path, is defined in the application model. Promotion paths and user permissions can be defined and configured per application.

Figure 2 shows how Jenkins continues to manage the CI builds. When Jenkins completes a build, an associated Automic ARA Deployment Package is created. The Deployment Package is then moved onto a target environment promotion path created in the application model. In a CI example, the build triggered by a file check-in from a developer would be automatically deployed to a development sandbox. Promotion of the Deployment Package to all other environments would be determined by a configurable set of role and group permissions. Automic integrated approvals, as well as approvals from external ticketing systems, can also be evaluated before a deployment is allowed.

Visibility and Compliance

Automic ARA makes it easy to explain to auditors and management teams how application deployment processes are implemented, as well as how deployments adhere to compliance, with clear graphical monitoring of executions. In addition, these graphical flows must meet administrators' and developers' expectations, providing all of the technical deep logging and reporting needed to log all Action executions, or assist in debugging what failed.

Figure 3: Example of Automic ARA workflow execution monitoring

Figure 3 shows an example of Automic ARA Live Monitoring. A graphical monitor of the workflow execution, including text-based output reports, is generated and saved for each deployment execution. A package deployment history per environment is kept to track all migrated changes.

Environment snapshots, as highlighted in Figure 4, keep track of every configuration and component deployed. Easily understanding the history and contents of a deployment package makes managing configuration drift much simpler. Want to understand if a package has ever been deployed in to an environment? Look at its deployment history. Want to understand why an environment that was fine before lunch is suddenly broken at 2pm? Run a snapshot compare to see if any unauthorized/manual updates to configurations or files have been made.

Figure 4: Example of detailed Snapshot configuration

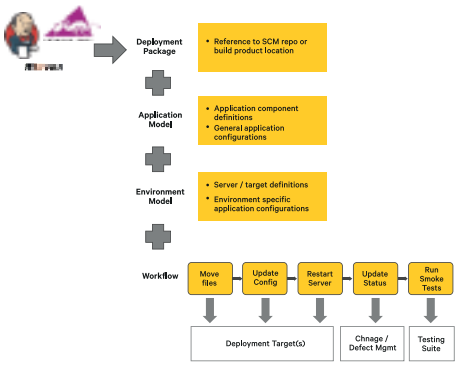


Figure 5: Example of how Jenkins fits into the Automic ARA structure

How can I use Jenkins and the tools I have today with Automic ARA?

With Automic ARA, there is no reason to rip and replace any pre-existing build engine. As illustrated in Figure 5, Automic can seamlessly integrate with Jenkins to manage the post-build steps of an integrated deployment. In fact, any software currently used in-house, from change and service ticketing systems, to testing suites or even infrastructure configuration can be executed as a part of an Automic ARA workflow, as shown in Figure 6.

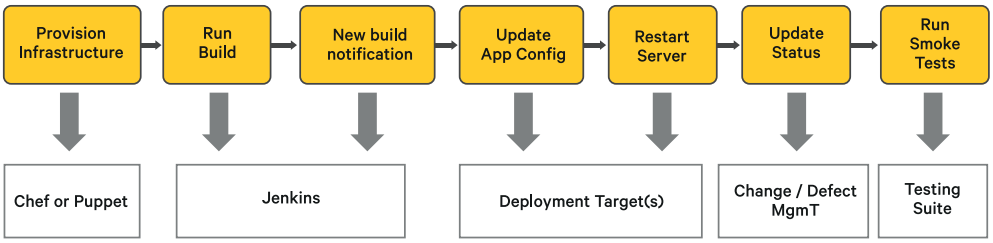


Figure 6: Conceptual End-to-End ARA Workflow

Conclusion

Automic works well with Jenkins' continuous integration capabilities. Since Automic is a tool-agnostic platform, it also works well with TFS, Maven, Ant or whichever build system is currently in use. Automic ARA shines when orchestrating and implementing complex processes that are currently being handled by DIY scripting systems or manual configurations. Customers who already use and like Jenkins can leverage its core CI capabilities into an integrated, cross-enterprise Automic ARA process where Modeling, Integrations and Workflow are defined and managed centrally for a repeatable, reliable and auditable result.