Application Release Automation (ARA) Vs. Continuous Delivery

A whitepaper review of the feature and process differences between Continuous Delivery and Application Release Automation (ARA)

By Tracy Ragan, COO, OpenMake Software

Contents

Summary

This whitepaper compares the difference between Continuous Delivery and Application Release Automation (ARA). It will explore critical feature for automating releases that are not provided by a Continuous Delivery solution including server configuration management, rollback and roll forward, database updates, component packaging, security access and agentless delivery.

In a Nutshell

Continuous Delivery (CD) is a process, not a solution. Continuous Delivery is an extension of Continuous Integration. When a software update is saved to the version repository, the Continuous Integration workflow is triggered to execute steps that may include the calling of a script to compile code into binaries (Continuous Build), followed by a script to deliver the binaries to a list of servers (Continuous Delivery). In some cases where the production environment is made up of only a small set of servers, the Continuous Integration process may support production deployments, but in most organizations CI is used mainly by development and testing teams. When someone states they are doing Continuous Delivery, they are saying that they use their CI process to execute a deployment script.

Application Release Automation (ARA) is designed to fully orchestrate the delivery of software including infrastructure and database updates, server configuration management, calendaring, roll-forward, rollback, security access and component packaging. A Continuous Delivery process may call an ARA solution to perform the orchestration of the deployment, replacing the one-off deployment scripts written by developers.

The confusion between these two solutions is in the appearance. You might have a really smart developer who can write a script to deliver code to particular endpoints. You execute your Continuous Integration process and bingo, the binaries are delivered. So why should you use an ARA solution when the scripts you are running work fine? The following critical features are not delivered by a Continuous Integration server and are required to fully automate, control and audit your application releases.

Server Configuration Management

It is true. A script called by a CI process can set server parameters. But the script is not the place to store or manage this type of information. ARA solutions centralize the management of server settings and configurations and can do so for thousands of machines. This centralization provides a quick, self-service portal for anyone, from development through production, to view, analyze and update configurations if needed, without executing a script.

" The challenge is not how the scripts were called (via a CI process), the challenge is that the scripts are called. Scripting is a poor solution for managing Server configuration. "

When this low level configuration data is managed inside a script, only the author of the script will know where in the script the configuration settings are located, and exactly how and if the script is actually setting them. Scripts can easily obfuscate a configuration setting. What you thought was being done by the script was actually commented out in the script, or the script failed to loop through all endpoints as the server list was not updated. Transparency is the biggest issue with the scripts making releases difficult to understand.

The challenge is not how the scripts *were called* (via a CI process), the challenge is that the scripts *are called*. Scripting is a poor solution for managing Server configuration.

" DeployHub tracks the versions of your release. If the testing team is on version 4 and they need to get to version 6, DeployHub knows to execute the updates of version 5 and 6 catching all incremental processing along the way, including the database updates."

Database Updates, Rollback and Version Jumps (Roll Forward)

Deployments are not always successful which requires an immediate rollback process. Continuous Delivery calling one-off scripts can be easily used to delete the entire directory where the deployment was copied, and then reinstall an older version. In addition, there are often cases where the deployment needs to jump versions, what is called Roll Forward in OpenMake DeployHub. For example, testing did not take the last few builds, but now wants the latest release. Again this can easily be done by a Continuous Delivery workflow calling a script by simply deleting all directory contents and installing the required release level. These examples are a simplified view of rollback and roll forward processing, and they are not very realistic.

In reality, deployments are associated to a database. Even a change as small as adding a word to a look-up table for a drop down list requires an update to the database. More involved database updates require dropping tables, adding tables and dropping or adding columns.

Managing table updates requires more than just deleting and coping some files around. For this reason, the deployment engine should always be integrated with updates to the database. ARA solutions such as OpenMake DeployHub handle the tables and the binaries together. You can easily rollback your deployment to a previous level of the database, or Roll Forward if jumping versions is required. DeployHub tracks the versions of your release. If the testing team is on version 4 and they need to get to version 6, DeployHub knows to execute the updates of version 5 and 6 catching all incremental processing along the way, including the database updates.

Even the smartest developer would struggle with writing a script that would handle any combination of a Rollback or Roll Forward requirement. The ability to jump to any version or rollback to any version with the database changes is one of the core purposes of an ARA solution. Scripting alone cannot provide this type of orchestration. The alternative is to have the DBAs involved in the release which then makes your Continuous Integration process rely on manual intervention. ARA prevents any bottlenecks in a Continuous Integration workflow by handling the database updates automatically.

Component Packaging

Your software release is comprised of many different binaries files, flat files, database updates and infrastructure components. In the process of Continuous Delivery, a script is called that references just the application binaries (.war, .jar, .dll, .exe) leaving out other critical components such as what version is required for the java runtime, the database or even the operating system. Some ARA solutions, such as Puppet and Chef, were designed to support 'infrastructure as code.' These solutions, developed by Linux System Admins, were intended to help simplify the management of the server infrastructure. A common practice with these types of solutions is to use a 'manifest' file that represents the target server end state. Remote agents on the endpoints compare the master manifest residing on the master server and if different an update is deployed.

"DeployHub supports a highly reusable domain structure that clearly displays the available shared components for all servers. This gives the developers the information they need and stops the guess work."

These types of solutions have been used to include the update of application code along with the infrastructure code. To do this, a developer codes the master manifest to include the application code. In some cases, manifest solutions are only used for application code even though they were intended for infrastructure management. Using a manifest can be better than a script as there is at least a standard for writing the manifest. If someone must, they can learn the manifest syntax, open the text file and see what is required for the release.

The challenge with manifest ARA solutions is in the standardization and sharing of components. In the manifest, you specify the location of where the objects come from. This means that two development teams using the same infrastructure could have manifest that contradict one another. This lack of communication drives teams to build everything under their own VM so they do not cause issues for another team. This solution is similar to killing a house fly using dynamite when a fly swatter will do the job. However, without the use of a complete ARA solution it is the only way to solve the problem. With scripts and manifest files there is no way to see who is using what versions of the shared components and to what servers those shared components are installed.

DeployHub supports a highly reusable domain structure that clearly displays the available shared components for all servers and binds the components to the servers. DeployHub is the only solution that provides component packaging and supports the sharing and binding of components to servers across the lifecycle.

"The calendar can be used to block a release, avoiding the mistake of delivering a release during peak business hours or can be used to request a release."

Calendars vs Delivery Pipelines

In the Continuous Delivery process, calendars are not used. If CD is being used this means that a workflow that includes a deploy is executed based on a trigger. CD uses what is called a *Delivery Pipeline* to help manage multiple executions of the CI workflow. A Delivery Pipeline sounds like it will tell you what is about to be delivered to your production environment, but it does not. A CI Delivery Pipeline is used to manage concurrent check-in, build and deploy workflows. If a team is compiling and delivering code on a very frequent basis they may end up with multiple workflows running at the same time. The Delivery Pipeline manages the concurrency level for the build, test and deploy steps of the workflow. These steps do not map to lifecycle states. They simply map to what the workflow is executing -1)execute compile script 2)execute static code analysis, 3) execute deploy script and coordinates the concurrent processing of these steps.

CD and the delivery pipeline may work very well for development teams who want changes moved out to their development servers quickly. However it is not designed to manage or communicate the releases that are headed for testing or production environments. This is the purpose of a Release Calendar.

ARA solutions include a calendar feature that allows for a release to be scheduled. ARA can support the Continuous Delivery philosophy with the goal of achieving more frequent, incremental releases. An ARA deploy can be included in the developers CD workflow to satisfy the developers requirements. As code moves up the lifecycle, the releases become less frequent. For example, testing may agree to take a new release every day where production control will only take the releases that pass testing.

DeployHub manages a calendar for each 'environment.' An Environment is defined for each stage of the lifecycle. The calendar can be used to block a release, avoiding the mistake of delivering a release during peak business hours or can be used to request a release, allowing developers to notify testing or production that a release is ready. A release can also be scheduled to run at certain times of the day allowing testing and production control to define the release frequency to a manageable number. The Calendar gives you a clear view of all software releases for all stages in your lifecycle.

"DeployHub allows you to define access to multiple objects from the definition of components to the execution of a release. And because one-off scripts are replaced, all changes to the process are tracked with historical timelines showing who made what change, on what date. "

Security Access

ARA solutions create a combination of security levels to maximize the security access to software deployments. While CD solutions can secure who can access and execute workflows, they do little to customize access across different environments. In addition, CD cannot control access to the underlying scripts, which is where the critical logic of a release is managed.

ARA solutions such as OpenMake DeployHub control the security access at multiple levels allowing a secure but flexible security schema. For example, developers may need access to define a software release in terms of application components, release flow logic and the ability to perform development releases integrated into the Continuous Integration server. However, the developers cannot have access to testing or production environments and can only 'request' a release to be performed at the higher states.

DeployHub allows you to define access to multiple objects from the definition of components to the execution of a release. And because one-off scripts are replaced, all changes to the process are tracked with historical timelines showing who made what change and on what date. CD solutions cannot deliver this level of audit tracking and instead rely on a version control tool to manage the scripts and manifest files.

Agentless Delivery

All Continuous Delivery and most ARA solutions require each end target (server) to use a remote agent to perform the deployment. The logic of the deployment is executed on each server by the remote agent. Remote agents are a weak link in software delivery and create a maintenance headache for complex enterprises with hundreds of servers. OpenMake DeployHub uses a unique agentless technology that speeds up pre-processing of the release logic and eliminates the need to install and maintain hundreds of remote agents.

Conclusion

Continuous Integration was designed to help developers manage build and deploy scripts using a workflow process that supports their specific development methodology. Continuous Integration calls deployment scripts as part of the continuous integration workflow, and this is referred to as Continuous Delivery. A Continuous Integration server does not perform builds or releases. Continuous Integration simply executes the scripts that perform these functions, centralizes the output of these scripts and provides some very high level reporting. Continuous Delivery does not automate the application release process, and scripts are not automation.

Application Release Automation (ARA) drives deep into the software deployment process to automate all aspects of a deployment creating a dynamic, model driven framework for orchestrating releases. ARA replaces the backend scripts, improves the management of server configurations, supports a release calendar, defines access controls, performs component packaging and tracks application versioning to support rollback and roll forward processing. ARA can be called by a Continuous Integration process to manage and orchestrate the Continuous Delivery. They are not mutually exclusive and perform very different functions. ARA should be carefully considered as part of an overall DevOps approach to software deployments.

About OpenMake Software

OpenMake® Software delivers highly reusable DevOps Solutions that allows our customers to go the 'last mile' in agile. DeployHub, our Application Release Automation solution, eliminates the friction in the continuous delivery pipeline as code moves between environments. Meister accelerates and streamlines the software compile process for highly efficient continuous builds. As a 100% self-funded organization, we have the freedom to focus on our customer's needs, delivering innovation in software builds and release. An investment in our DevOps solutions or Professional Services forms a 'technical partnership' that provides you expertise and support to solve your toughest build and release problems for today and tomorrow.

DeployHub is an Open Source project at www.DeployHub.org.

DeployHub Pro has advanced security and auditing features. A free version is available to small teams and can be downloaded at no cost from https://www.openmakesoftware.com/deployhub-free-downloads/

Tracy Ragan – COO and Co-Founder, OpenMake Software

Ms. Ragan has had extensive experience in the development and implementation of business applications. It was during her consulting experiences that Ms. Ragan recognized the lack of build and release management procedures for the distributed platform that had long been considered standard on the mainframe and UNIX. In the four years leading to the creation of OpenMake Software she worked with development teams in implementing a team-centric standardized build to release process. She can be reached at Tracy.Ragan@OpenMakesSoftware.com.

ASERVO Software GmbH
Konrad-Zuse-Platz 8
81829 München / Germany
T: +49 89 7167182-40
F: +49 89 7167182-55
Email: kontakt@aservo.com
www.ASERVO.com