Solving the Continuous Delivery Puzzle

A whitepaper review of how to leverage DeployHub, IT Automation (Ansible, Chef, Puppet) Jenkins and CloudBees for a stronger Continuous Delivery Foundation.

By Tracy Ragan, CEO, DeployHub

Contents

Summary

With so many tools serving the continuous delivery market, what job each tools does may be unclear. This whitepaper explores the roles of ARA, IT Automation, and Continuous Integration for building out a stronger and broader continuous delivery pipeline to support enterprise strength CD.

IT Automation engines such as Ansible, Chef and Puppet are designed to automate cloud provisioning, manage network devices, manage server configurations, and deliver RPMs for application deployments. In addition, Jenkins for Continuous Integration (CI) is often used for building and releasing software applications. So the question one should ask is “Why do I also need Application Release Automation if I already have both an IT engine and a CI server for handling application deployments?” This whitepaper answers that specific question and explores how a better Continuous Delivery solution can be gained by integrating the CloudBees Jenkins Platform with Ansible and DeployHub.

Primary difference between IT Engines and ARA

IT Automation engines are designed to automate the management of servers, containers, network devices, and cloud based data centers. They control configuration details and deploy RPMs (Redhat Package Management files). Developers who use IT Engines for application deployment most commonly do so by first creating an RPM for their application and then passing that RPM to the IT Engine. RPMs were designed for performing Linux installations; however over the years RPMs are now more universally accepted. The problem however is an RPM is OS dependent. An RPM written for Linux must be updated for other operating systems such as AIX. In addition, RPMs must be scripted for each application, hiding critical information about the Application Stack packaging and component relationships with configuration details and deployment logic. RPMs were designed as a delivery mechanism, not an end-all solution to application release automation.

Organizations looking at Application Release Automation (ARA) solutions are seeking a more flexible and transparent way to manage the packaging and deployment of the Application Stack without relying on the use of RPMs. ARA solutions provide an easy interface for package creation with the ability to track component versions to application versions along with the deploy logic at the component level. And unlike using an RPM, the packaging and associated logic can be installed to any Operating System across the continuous delivery pipeline. In essence ARA automates and controls what would otherwise be scripted and hidden in a RPM and then extends the automation and control to the final delivery step, tracking components to endpoints (physical, virtual, cloud, containers), supporting roll forward and rollback, database updates and historical reporting.

Both It Automation and ARA have their place in the delivery process and work better integrated as a complete solution. Using ARA for infrastructure management is doable, just as using IT Automation for application deployments is doable. You can always use a kitchen knife in place of a screwdriver when you are in a pinch. But when faced with building a house, most people invest in a screwdriver, or better yet a screw gun. The point being - choose the right tools for the job.

Benefits of ARA over RPMs

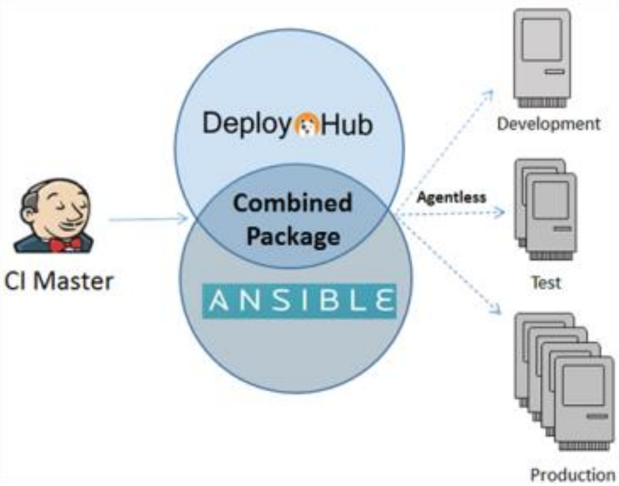
When it comes to packaged software (Oracle, WebSphere, Tomcat, etc.) most developers would prefer to use the RPM from the vendor and use an IT Engine to perform the deployment. The job of installing the new software is done without much effort. It follows the old KISS rule (Keep It Simple Stupid).

If moving the software from point A to B is all that is needed, our problem would be solved. However, there are other functions such as reporting and tracking the update, database management, incremental rollback, jumping versions, calendaring and inventory reporting that is required by larger, more complex organizations. These functions are critical to automating Continuous Delivery. So while using an IT Automation engine to deploy an RPM might work for development, it lacks the sophistication for a fully automated and traceable Continuous Delivery pipeline. This is where ARA steps in.

"OpenMake Software's DeployHub and Ansible are integrated to create a combined package."

Using the Best of Breed

Using the right tools for the job requires integrating ARA with IT Automation, Continuous Integration and Continuous Delivery. This allows an overall DevOps approach that uses RPMs for vendor updates, an IT engine for managing the infrastructure configuration, and ARA to coordinate the Application Stack, all managed as a combined package across the Continuous Delivery Pipeline driven by Continuous Integration.



Combining Ansible and DeployHub Packages

To simplify this coordination, DeployHub and Ansible are integrated to create a *combined* package. The process can be initiated by Continuous Integration calling DeployHub to create the combined package, working with Ansible for a completely integrated solution from development through production release.

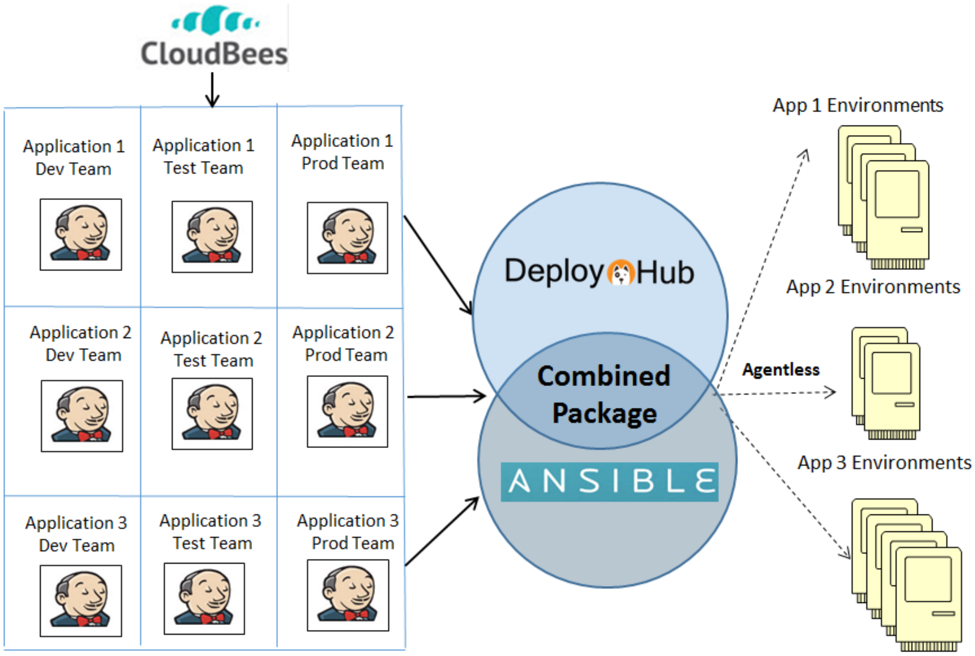
Ansible is leveraged to manage RPMs, server configuration, environment updates, cloud provisioning and other IT Automation tasks. DeployHub pulls together the Application stack, database updates and the infrastructure layer as a complete combined package, delivers the combined package, tracks the Component Versions, audits the process, tracks endpoints to Component Versions, supports rolling forward/rollback, version jumping, calendaring, security and delivers historical reports. The process can be initiated by a Continuous Integration trigger and pushed up the Continuous Delivery lifecycle for non-stop software deployments.

Continuous Integration Workflows, ARA and IT Automation

So the next question often is “We use Jenkins, so why should we use DeployHub or Ansible?” The real question is “Who drives the process and who does what job?” Jenkins does not do Application Release Automation or IT Infrastructure Management. Jenkins is an orchestration solution that sits on top of many processes, providing centralized initiation and reporting. It does not have specific features to support application packaging or IT Automation.

A Jenkins Workflow calls out to an external process that in turn performs the different functions from build, package, deploy and test. It is often the case that a single Jenkins server supports a single state in the lifecycle for a single team. CloudBees Jenkins Platform sits on top of multiple Jenkins Servers to standardize and manage larger Jenkins installations. Jenkins calls out to DeployHub which then creates the combined package with Ansible Components, versions the deployment, performs rollback/roll forward, tracks the component to the endpoint, and reports the results back to Jenkins.

“Jenkins calls out to DeployHub which then creates the combined package with Ansible Components, versions the deployment, performs rollback/roll forward, tracks the component to the endpoint and reports the results back to Jenkins.”



Agentless Delivery

DeployHub and Ansible are agentless solutions that dramatically simplify implementation of both IT Automation and ARA. These agentless solutions also greatly reduce long-term maintenance costs, bottlenecks and overhead. By allowing DeployHub and Ansible to take over the deployment of the combined deployment package without the need for agents, you eliminate hundreds of endpoint deployment slaves.

Eliminating deployment slaves does not take away the benefits of using Jenkins or CloudBees. It instead supports a simpler and faster way of moving a Jenkins Workflow across the Continuous Delivery pipeline without needing to install the Jenkins Executors to all endpoints for performing the delivery. The Jenkins Master orchestrates the Continuous Integration for each application team at each state in the Life Cycle. Where needed, a Jenkins Executor may be installed for testing functions, but the problem of a Jenkins Executor on every production and testing runtime endpoint is eliminated. CloudBees is then used to monitor, standardize and control the Jenkins Masters.

> *"Eliminating the Jenkins Executors does not take away the benefits of using Jenkins or CloudBees. It instead supports a simpler and faster way of moving a Jenkins Workflow across the Continuous Delivery pipeline without needing to install the Jenkins Executors to all endpoints for performing the delivery."*

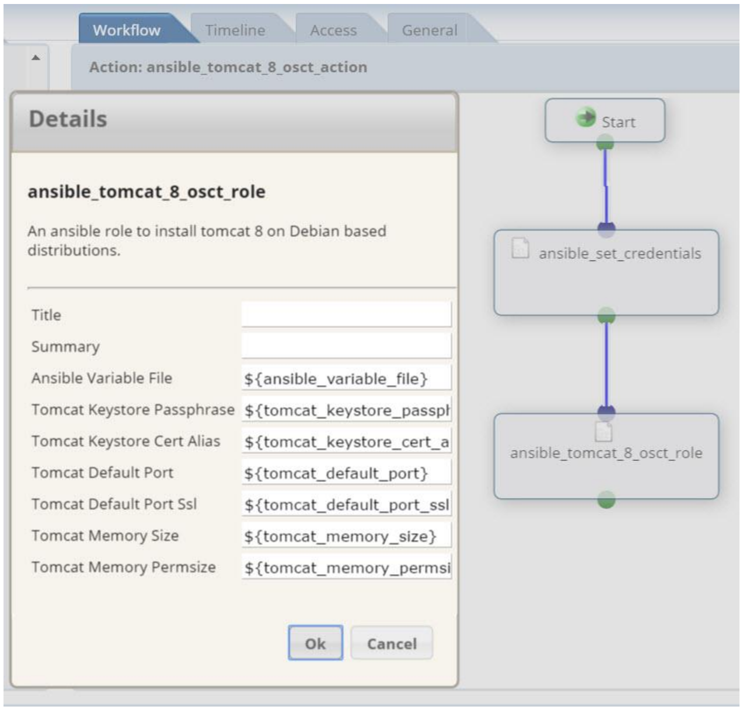
Agentless solutions represent the 'next generation' in ARA and IT Automation for many reasons. First, an agentless solution eliminates the costs associated with managing agent licenses and the time required to maintain the agents themselves. Second, in the data center of today, spinning up virtual machines, and managing containers with microservices represents a more elastic topology versus the static topologies of only a few years ago. With today's more elastic topologies, installing unused agents on each end-point isn't efficient in terms of management time and licensing cost.

Like Ansible, DeployHub uses secure protocols such as FTP(S), SFTP/SSH and Win/SMB to support its agentless delivery. DeployHub can deploy too many different server platforms including: Windows, UNIX, Linux, iSeries, OpenVMS, Tandem, Stratus, IBM4690, Tru64.

DeployHub and Ansible Galaxy Roles

DeployHub includes integration to the Ansible Galaxy Roles. On start-up, DeployHub loads all available Ansible Galaxy Roles into the DeployHub database from the Ansible community site. These Galaxy Roles are defined to DeployHub as *Components* with *Actions*. *Components* are assigned to an *Application* and are versioned. With the Galaxy Roles immediately available, it is very easy to add the installation or update of Tomcat, WebSphere or Oracle using the Ansible Galaxy Role. No extra work required. It is all done for you. The new Ansible Component is defined to your Application with all of the required.

> *"On start-up, DeployHub loads all available Ansible Galaxy Roles into the DeplyHub database from the Ansible community site."*



Action: ansible_tomcat_8_osct_action

Details

ansible_tomcat_8_osct_role

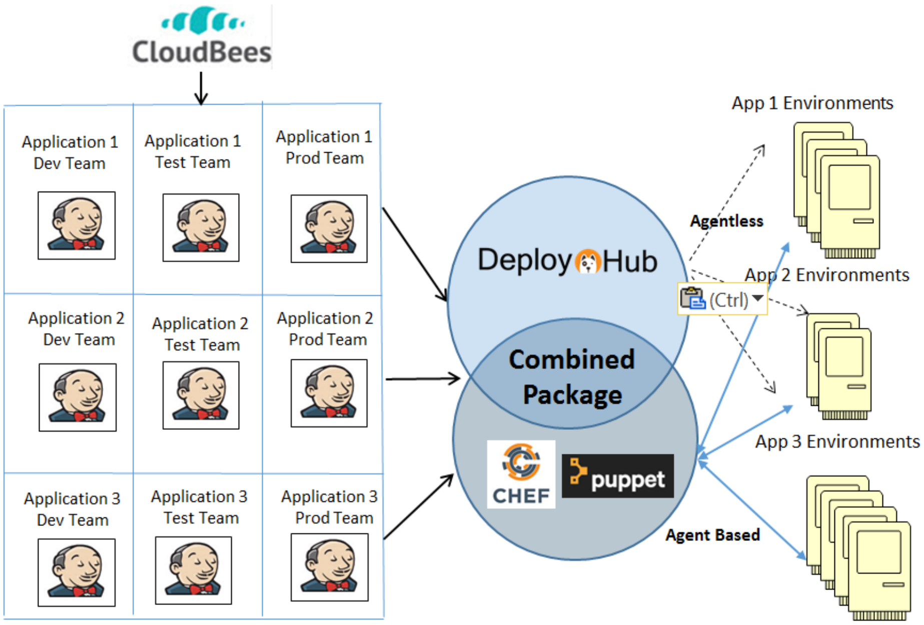
An ansible role to install tomcat 8 on Debian based distributions.

Title	
Summary	
Ansible Variable File	${ansible_variable_file}
Tomcat Keystore Passphrase	${tomcat_keystore_passpl
Tomcat Keystore Cert Alias	${tomcat_keystore_cert_a
Tomcat Default Port	${tomcat_default_port}
Tomcat Default Port Ssl	${tomcat_default_port_ssl
Tomcat Memory Size	${tomcat_memory_size}
Tomcat Memory Permsize	${tomcat_memory_permsi

DeployHub and Puppet or Chef

DeployHub also supports both Chef and Puppet to assist with creating the 'Combined Package.' Chef and Puppet can easily be called as part of the *Component Item* Workflow to perform these infrastructure layer steps. Once a Chef or Puppet Component is defined to DeployHub, DeployHub passes control to Chef or Puppet to perform the work. This means that you could define your Application to include a Chef Component to install or update Tomcat prior to the deployment of an EAR file. DeployHub will Version the Chef Component, call on Chef to perform the delivery of the Component, track the Chef Component to the Server and send the logs back to the Jenkins CI server that initiated the work.

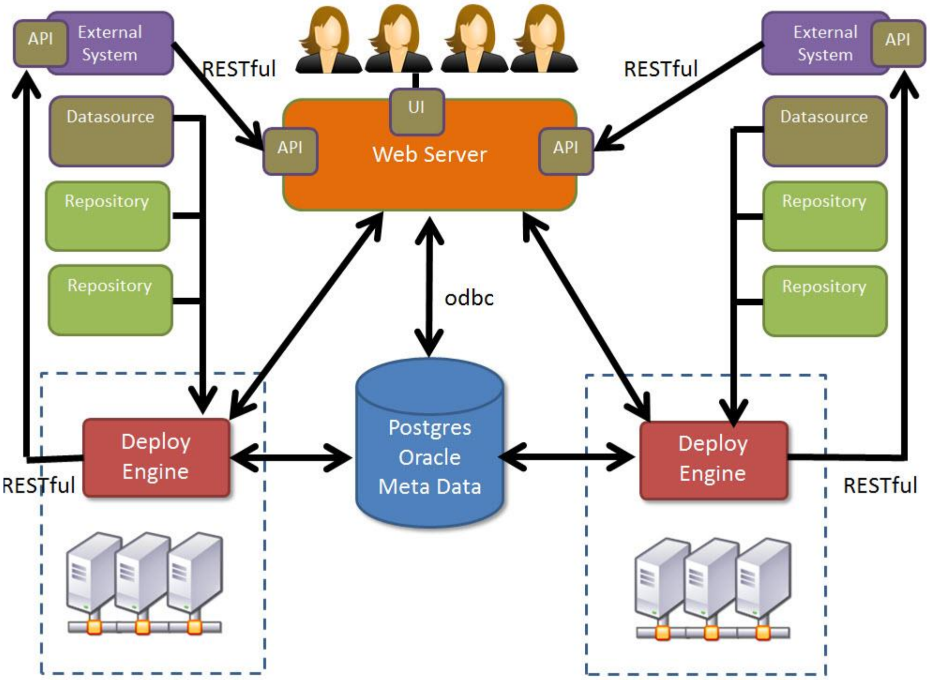
> *"Once a Chef or Puppet Component is defined to DeployHub, DeployHub passes control to Chef or Puppet to perform the work."*



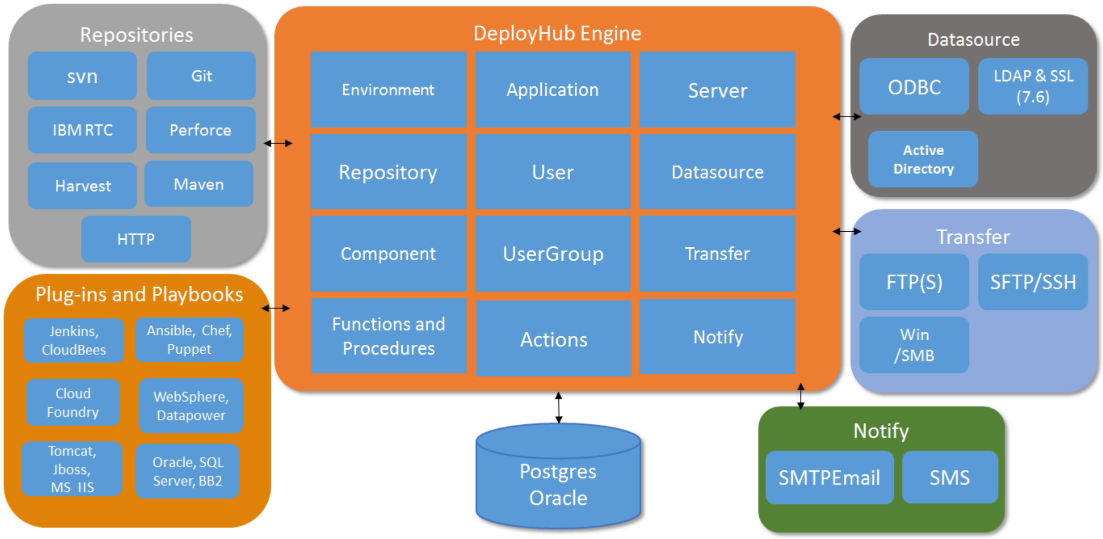
DeployHub Architecture

DeployHub's architecture is based on a web based UI with a Linux Engine that includes a relational databases backend (PostgreSQL or Oracle).   A Windows Engine is also available. You can configure DeployHub with multiple Deploy Engines for segregating *Environments* if needed. This is useful for large organizations with thousands of endpoints. DeployHub can connect to multiple repository types for retrieving binaries, such as Git, SVN, Maven, file system, FTP, etc. In addition you can connect to any type of data source to pull out parameters or other details stored externally.   There is a full set of APIs and a Jenkins Plug-in which allows Jenkins to initiate a DeployHub deployment as part of a Continuous Integration Workflow.

In addition to the above, DeployHub provides History, Inventory and Trend reports, and tracks all version updates to Components, Workflow Logic and Parameters. DeployHub can meet the toughest of Audit requirements.   It provides a complete audit trail of how the application was packaged and deployed and can link back to Meister to show what source code was used in a particular binary that went to production.



Architecture Diagram #1



Architecture Diagram #2

DeployHub Benefits

DeployHub was designed for agile teams with complex deployments to:

- Support the ability to model the organizations structure based on Domain and Sub-domains for self-service administration and sharing of objects.
- Release Planning with the ability to monitor ‘bug’ burn down rates.
- Continuous Feedback Loop tracking a Jira, Bugzilla or GitHub issue and Jenkins build log to the final endpoint to which the updated code was deployed.
- Support agentless deployments to multiple platforms delivering a single solution from Java to .Net, Unix to Tandem, and everything in between.
- Provide a high level of sharing and collaboration between project teams and operational teams.
- Bridge datacenter needs with the integration of Ansible Galaxy Roles;
- Support the ability to jump back or forward to any software release at any state in the lifecycle, including the necessary database updates.
- Centralize the management of deployment pipelines using an integrated calendar and approval gates.
- Define release packages using a graphical drag and drop interface that is easy for everyone to understand.
- Integrate with open source tools to leverage community knowledge and plug-ins.
- Off-load overworked CI Servers and improve the scalability of CI.
- Support releases to servers, clouds and containers.
- Achieve higher levels of deployment maturity leading to continuous process improvement and less risky software releases.
- Audit the server inventory with component, application and release versions.

DeployHub creates a release automation process that is easy to understand, allowing both the development teams and centralized release teams to share feedback and improve the process, quickly leading to less risky software releases and faster innovation cycles.

Conclusion

Building a stronger and faster Continuous Delivery pipeline needs the agility and intelligence of automation tooling. Because there are many choices in which type of tooling to adopt, it is important to understand what tool performs what job. ARA is used for managing the application stack. IT automation is used for managing data center configuration and the deployment of RPMs. Continuous Integration such as Jenkins serves as a high level workflow orchestration tool for each development team and stage. CloudBees helps standardize and manage multiple Jenkins Servers across large organization for better Continuous Delivery management. No one tool can support all of these features independently. A combined solution that leverages best of breed tools to automate the entire Continuous Delivery pipeline will ultimately deliver the speed, control and transparency required for supporting fast and flexible software turnovers.

DeployHub has close integration with Ansible Galaxy Roles, supports Chef and Puppet as Components, builds a ‘combined’ package of both application and infrastructure components, tracks Jira, Bugzilla and Github issues, and reports to Jenkins through a Plug-in. Its agentless design allows for organizations to take advantage of the benefits of Jenkins while also keeping tighter control over test and production environments.

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

DeployHub Pro has advanced security and auditing features. Learn more at https://www.DeployHub.com.

About DeployHub

Tracy Ragan – CEO and Co-Founder, DeployHub

We are software engineers who build scalable Agile DevOps solutions that solve continuous delivery problems. Nothing drives us more than helping customers dramatically accelerate software release cycles from continuous build through continuous deploy. Learn more at deployhub.com or deployhub.org for the open source project.