Achieving Continuous Deployment with DeployHub and Jenkins

A whitepaper review of how to achieve Continuous Deployments using Jenkins and DeployHub, an Open Source next generation software deployment Solution.

By Tracy Ragan, CEO, DeployHub, Inc.

*"DeployHub's **strong integrations with Jenkins and** RedHat's Ansible make it a compelling option to those looking to extend those tool investments…The product is well-suited for organizations and teams of all sizes in need of a **low-priced, yet capable, ARA product.**"*

Magic Quadrant and Critical Capabilities for ARA, 2017 Gartner Report

Table of Contents

Summary

Continuous Deployment is no longer a 'unicorn.' It is an absolute reality. And it can be done affordably with open source DeployHub. This whitepaper covers how to achieve Continuous Deployment using DeployHub *Team* to manage your continuous deployments by Jenkins & Blue Ocean.

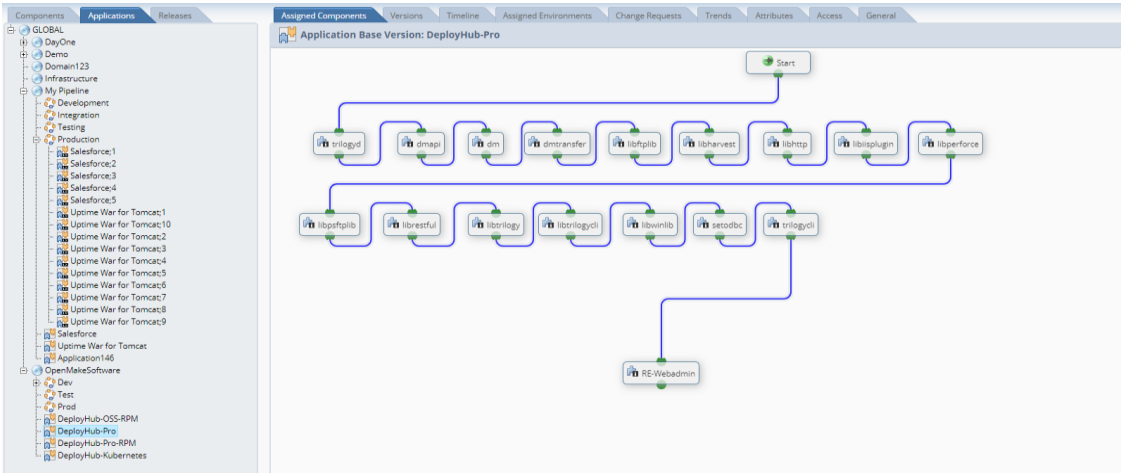
DeployHub *Team* is an open source continuous deployment solution that evolves deployments from an 'endpoint release' practice to an intelligent repository of deployment data. DeployHub *Pro* is a supported upgrade with additional enterprise security features. DeployHub supports iterative deployments across traditional and modern architectures. Its uniqueness is in the versioning engine. DeployHub versions every change in your deployment configuration serving as a single point of truth for every software release. DeployHub is agentless and supports server based and server-less environments such as Kubernetes. It empowers development teams with next generation software deployments for fast, frequent and safe iterative updates.

Continuous Deployment Across the Pipeline

Often time continuous deployment is defined as the process of updating Production 'live.' But what that often means is fast, small updates. In order to achieve faster and smaller updates, you must have the means to perform deployments in an iterative manner. DeployHub does this by versioning your deployment configurations using Application Versions, Component Versioning, Actions and Environments. To understand how this is done, we will first explore DeployHub so you can then understand how it integrates into Jenkins, allowing Jenkins to be a central point of control for both continuous delivery and continuous deployments all the way to production.

Declaring Your Application Package with Components

The ability to declare your application configuration is core to deployments. This is normally done with scripts. And while scripts have served this space admirably, DeployHub provides versioning, reuse, adaptability and feedback in a way that scripts alone cannot achieve. Unlike deploy scripts, DeployHub isolates the application declaration from the Environment to which it will be deployed, supports the sharing of common deployment methods across 'like' teams and provides reporting to link your Jenkins Build Job to Components and their final End Points. In DeployHub you declare your Application in a collection of Components, referred to as a Package. Components include files, binary objects, infrastructure Components, workflow logic for installation and Attributes that define environment variables and the Type. A blueprint designer is used to create the Application Package.



Applications and Component Blueprints

Workflows, Actions and Attributes

Applications and Components are declared with Pre and Post Actions that define how the Component or Application is to be installed. Templated Pre and Post Actions, including Ansible Galaxy Roles, are provided out of the box. Actions can be customized, and most importantly shared across teams. Environment variables and Types are defined to a Component Attribute and resolved at the Environment level during the deployment. Environments also have Attributes for variables and Types that map to the Component, i.e., a database Component maps to a Database End Point.

"Any change to an Application or Component, including their Actions and Attributes, is treated like code and stored in the deployment version control database."

Deployment Version Control

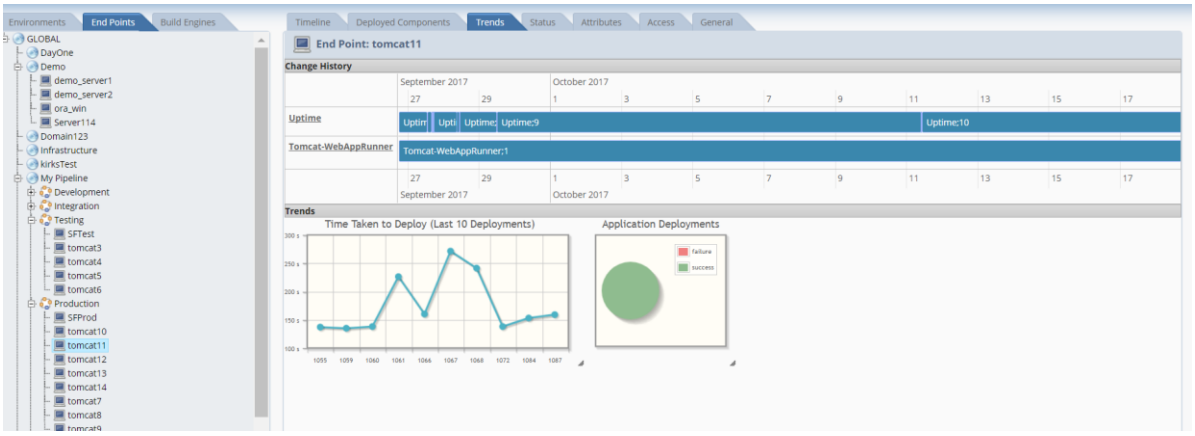
When Applications and Components are initially created, they have a 'base' version. Any change to an Application or Component, including their Actions and Attributes, is treated like code and stored in the deployment version control database. Rolling back, rolling forward or jumping a version becomes easy. The Application version is retrieved from the database and deployed incrementally based on deltas. Monolithic releases are not required reducing the risk of deployments by minimizing changes and accelerating the overall deployment process.

Infrastructure and Database Components

The Infrastructure stack and Database definitions can be added to an Application Package as Components. Ansible, Chef or Puppet can be used to define your Infrastructure Components, including Kubernetes configurations. These configurations are versioned along with your Application Package, creating a 'super' package that deploys the application stack, database and infrastructure requirements. If any change is made in the infrastructure or database, the information is versioned with the Application package so a rollback, roll-forward or version jump incudes these critical pieces.

Environments

DeployHub Environments are managed separate from the Application Package. In scripting, this is often done by referencing an external source that list all of the required Endpoints for a specific location (dev, test, prod). DeployHub expands on this concept through the use of variable and 'Type' Attributes. Environments and End Points have variable and Type Attributes that map back to the Component. For example, a War file is defined with the Type of "application" so it is installed to all Endpoints that are defined with the Type of "application." Variables Attributes are resolved by the Environment, or when needed by the Endpoint. The variable value is passed to the Component based on Environment at the time of the deployment. DeployHub supports a variety of Endpoints including physical, virtual, containers (Docker and Kubernetes) and cloud platforms.



Environment Endpoints and Trends

"DeployHub Pro expands the use of Calendars for 'no touch' approvals."

Provisioning and Containers

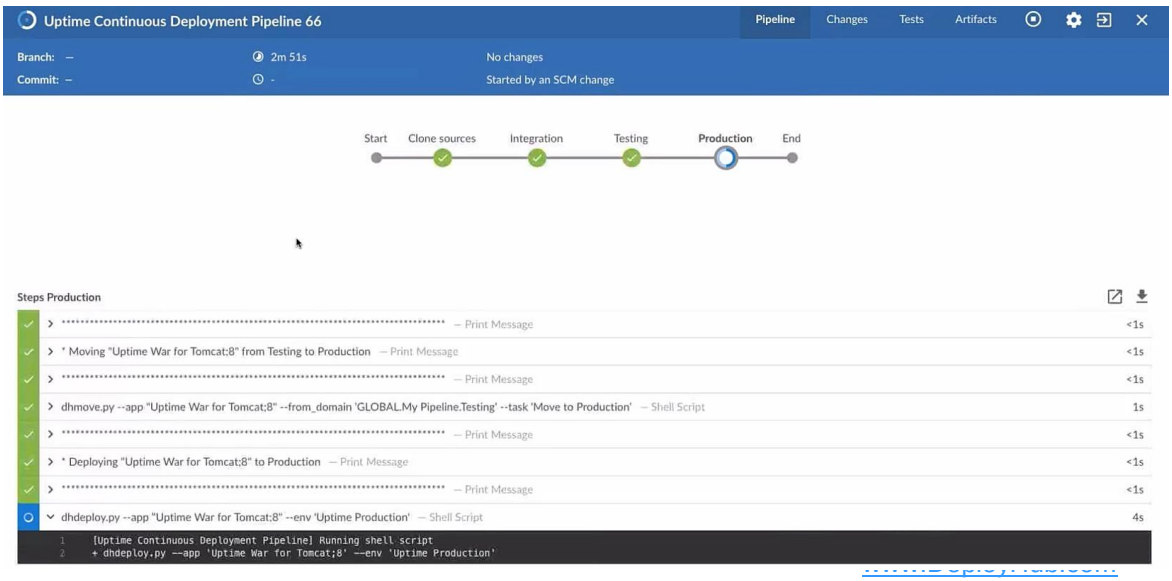
DeployHub supports the provisioning of Environments by using a Pre step to your Application Package. When you define your Application, you add the Pre Step logic to perform the provisioning. Differences in provisioning configurations are handled by the Environment Attributes.

Environment Calendars in DeployHub Pro

Unique to DeployHub is the concept of managing Calendars per Environment. Development, pre-prod and production all have their own Calendars that display what has been deployed. Calendars show on-demand deployments that have occurred, and scheduled auto-deploys. DeployHub *Pro*, an upgrade to the open source option, expands the use of Calendars for 'no touch' approvals. DeployHub *Pro* adds the concept of an Environment 'owner.' Owners can block out time frames or define when deployments will be accepted. This is an important part of the Jenkins integration.

Reasons for Driving Jenkins Continuous Deployments with DeployHub

The core reason to use DeployHub with Jenkins has mostly to do with the complexity around enterprise production environments. DeployHub eliminates complex deployment scripting, easily scales to large environments and supports rollbacks and roll-forwards. DeployHub extends the traceability of your Jenkins Build Job to End Points and includes a Jenkins Plug-in that shows Applications, Components, Environments and deployment Logs. Once your Application Package and Environments are defined, developers, testers and production teams can stay within Jenkins to execute and track deployments. DeployHub *Pro* Calendars add value by creating a 'no-touch' approval process across the Blue Ocean pipeline. DeployHub's Version Control engine versions Application Packages and Components automatically from within the DeployHub's Jenkins Plugin for easy rollback, roll-forward or even version jumping.



"DeployHub's agentless architecture makes implementation fast and easy, and without the concerns from production about installing yet another tool's agent on to their production machines or images."

Eliminate Complex Scripting

When Jenkins calls a scripted deployment process the scripts are written for each Environment. The development script is often used as a starting point to create more complex scripts for pre-prod and production. The complexity of the deployment script is tied both to the number of endpoints and the installation processing performed on the individual Endpoints. The production script becomes exponentially more complex than pre-prod or development scripts. This scripting process can quickly grow too large to manage and contain critical details that are known only to the author. Scripts cannot perform historical reporting such as Endpoint inventory or deployment comparison reports.

DeployHub solves deployment complexity by using reusable installation processing actions. The reusable actions perform the pre and post installation steps across all Endpoints of the Environment. Environment, Endpoint, Component and Application attributes are kept separate from the reusable actions. This separation enables DeployHub to adapt your Application Package across dev, pre-prod and production Environments. It uses a relational database to store data for historical reporting or the creation of custom reports.

Easily Scales to Support the Enterprise

DeployHub easily auto scales to hundreds of Endpoints without the need for additional work. Because it requires no Endpoint agents, pre-prod and production environments do not need to be pre-configured to perform deployments. DeployHub's agentless architecture makes implementation fast and easy, and without the concerns from production about installing yet another tool's agent on to their production machines or images. Moving to cloud environments or containers such as Docker or Kubertetes is easily supported. DeployHub can manage deployments to server-less datacenters with the ability to provision on-demand. Provisioning is controlled at the Environment level by the reusable actions. The development deployment could be to a Docker Container, pre-prod to Amazon AWS and production to physical and virtual Endpoints. No changes are required. The Application Package would be defined once and then deployed in many different ways by DeployHub called via the Jenkins Pipeline.

Deployment Version Control for Fast Rollback, Roll-forward Deployments

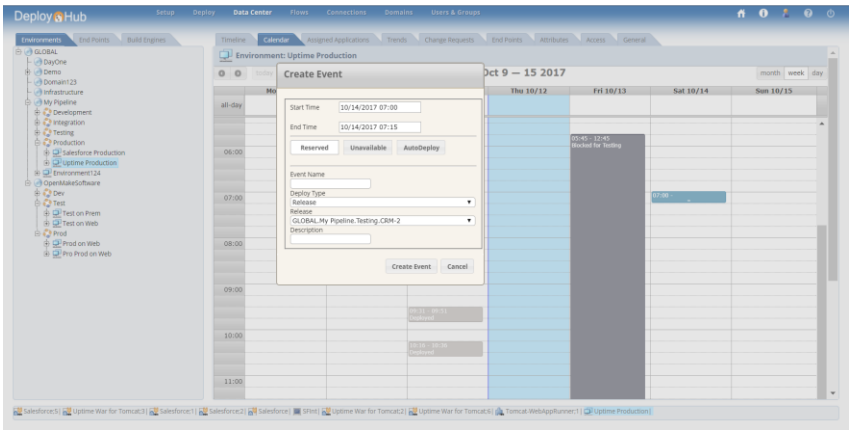
Application Package and Components version are created automatically using the DeployHub's Jenkins Plugin. When an Application Package is 'approved' to move forward down the pipeline, a new version of the Application Package will be created. This new version includes the new code updates from the developers. Versioning allows for quick rollback, roll-forward or version jumping.

"DeployHub's Calendars create a 'no-touch' approval process giving Environment Owners the ability to manage each Environment as needed, and without stopping the Jenkins CD process"

DeployHub *Pro* Calendar and the Jenkins Pipeline

The ultimate goal of continuous deployments is to allow code updates to be installed across all environments, based on success or fail status from continuous testing and automated checks. In some cases, Environment owners may want the ability to block the deployments coming down the pipeline to their Environments. Whatever the reason (maintenance windows, production freezes. etc.), DeployHub *Pro* allows the Environment owner to manage what is moving into their Environments using the Calendar.

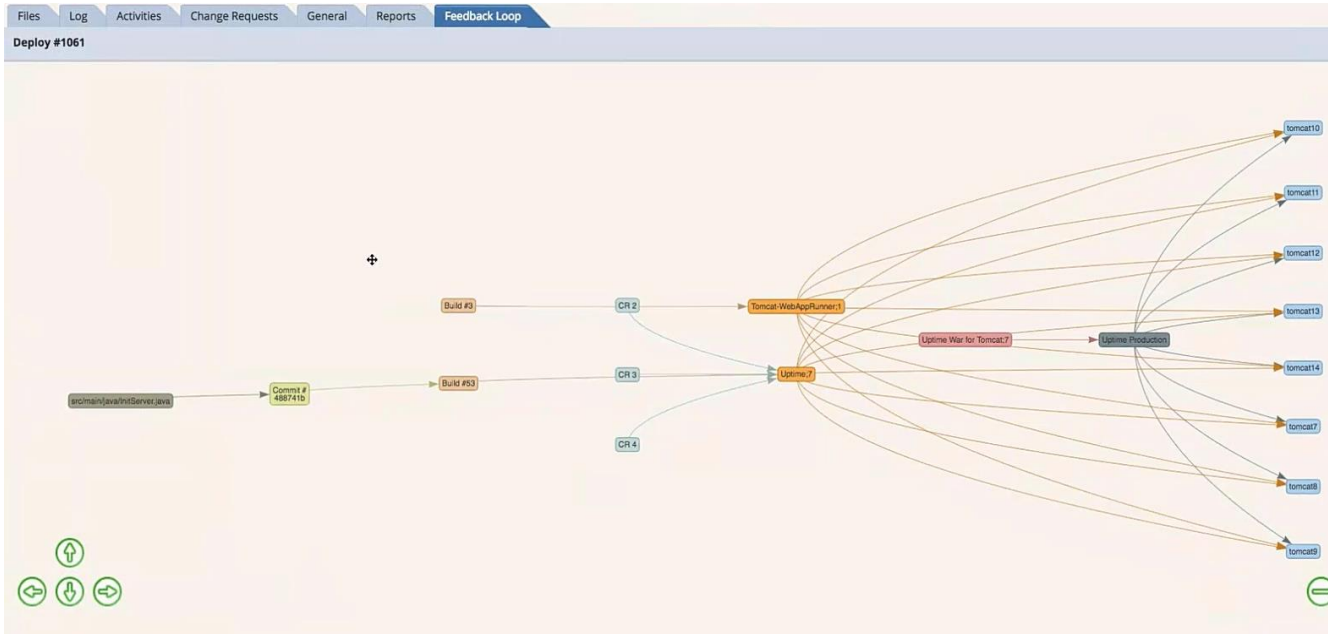
Pre-prod and Prod Environment Owners can 'close' or 'open' Calendars as needed. When Deployments are pushed across the Jenkins Pipeline, DeployHub *Pro* checks the Environment Calendar to determine if the Environment is 'opened' or 'closed.' When 'closed' a warning is returned to Jenkins and the Deployment is stopped. Once the Environment is re-opened, DeployHub *Pro's* version jumping will bring the Application Version to the latest level as the Pipeline process continues. Calendars create a 'no-touch' approval process giving Environment Owners the ability to manage each Environment as needed, and without stopping the CD process.



Environment Calendar with Blocked Days and Times

Jenkins and DeployHub Integration Points

The DeployHub Plug-in supports a process that allows most users to stay within the Jenkins environment to get their work done. Applications, Components, Environments and Deployments can be managed from within Jenkins. DeployHub tracks the Jenkins Build Job as part of the Application Package linking the Build Job to the locations to which the resulting Application Version was installed. This is shown in a Continuous Feedback Loop. With DeployHub *Pro*, the Continuous Feedback Loop links your issues coming from tools such as Jira, Bugzilla, or GitHub to the source code commit, to the Jenkins Build Job and forward, creating a Continuous Feedback loop that traces a source code commit to the locations to which the updates were installed.



DeployHub Continuous Feedback Loop with Jenkins Build Job

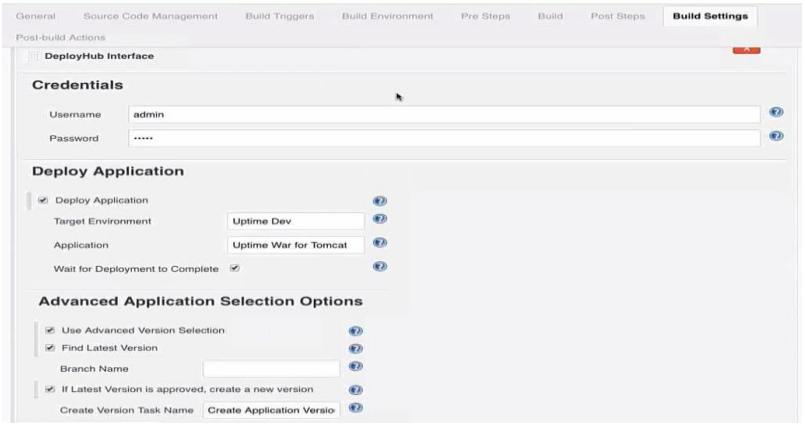
Installing the Jenkins Plug-in

DeployHub includes a Jenkins Wizard to assist you in setting up the Jenkins Plug-in. The Wizard will take you through defining your Application, Component and Environment linked to your Jenkins Master.

Using the DeployHub Jenkins Plug-In

Once installed, the Plug-in displays read access to all the DeployHub elements needed for managing your Applications and Components. 90% of the work needed to be done can be managed directly from within Jenkins.

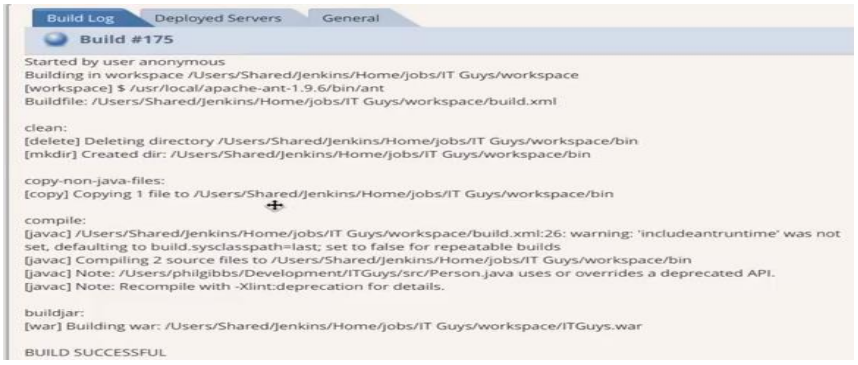
"90% of the work needed to be done can be managed directly from within Jenkins."



Sample DeployHub Jenkins Plug-in Interface

Deployment Logs

All logs associated to the Deployment are reported back to the Jenkins Build Job. Success/Fail is reported to Jenkins.

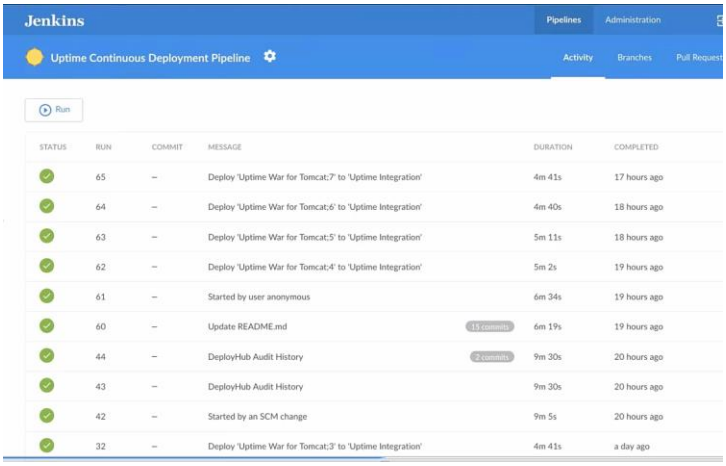


Log Reporting Back to Jenkins

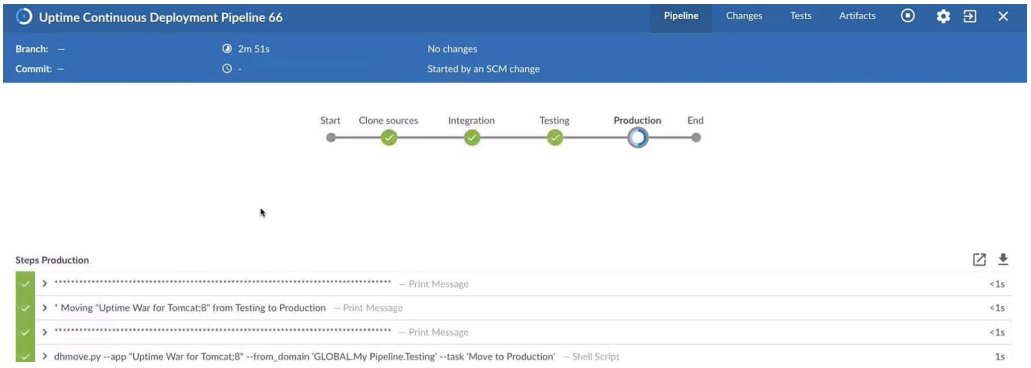
"DeployHub handles variable processing as part of the Package, resolved by each Environment uniquely."

Jenkins Blue Ocean Pipeline

As you execute your Pipeline, the Jenkins Job is executed for the different Environments. No manual updates to scripts are required to push your deployment across the pre-prod and prod Environments, even when these Environments are mixed, use different variables and are substantially larger in terms of End Points. There is no need to manipulate the *jenkinsfile* for Variable substitution; DeployHub handles variable processing as part of the Package, resolved by each Environment uniquely. All deployment versions are reported back to Blue Ocean along with their unique steps.



Blue Ocean Activity Log Showing DeployHub Deployments



Conclusion

Companies have invested in the use of Jenkins to manage Build Jobs, and the Continuous Delivery process. DeployHub extends that investment by 'hardening' the software deployment process across the dev, pre-prod and production Environments integrated into Jenkins and the Blue Ocean pipeline. DeployHub eliminates the complexity around deployment scripts creating a highly visible continuous deployment process that can be defined by development teams and consumed by pre-prod and production teams. DeployHub deployments are fully versioned, supporting fast rollback, roll forward or version jumping. Using Jenkins and DeployHub together allows you to master Agile's last mile – moving code updates to production environment at the pace of agile, in a safe and exposed process that meets the needs for all teams.

Pricing DeployHub Pro vs. DeployHub OSS

Feature	OSS	Pro
Agentless Deployments	☑	☑
Version Jumping	☑	☑
Application Packaging	☑	☑
Ansible Integration	☑	☑
Continuous Feedback Loop	☑	☑
Jenkins Integration	☑	☑
Smart Calendar		☑
Release Train Coordination		☑
Security Groups and Domains		☑
Change Request Tracking		☑
Change Request Burn Rates		☑

DeployHub Pro Subscription Pricing (unlimited Users & End Points)	**Annual Cost**
DeployHub Team	**$0**
DeployHub Pro	**$2,500 per year.**
DeployHub Enterprise with unlimited Projects	**$75k per year.**

Getting Started and More Resources

DeployHub is an open source project and includes a Hosted *Team* version that is free.

DeployHub Team Sign-up
The hosted team version can be used to deploy to unlimited endpoints by unlimited users.

DeployHub Team Download (on premise)
Uses a Docker Container certified by RedHat.

Join the Open Source Community
Help us create the best, open source continuous deployment platform available.

Learn more about DeployHub Pro
Need more security and support? DeployHub Pro starts at $208 per month per project team.

BlueOcean Video

About DeployHub, Inc.

We empower development teams with hosted next generation software deployments, specifically designed for the new Agile and microservices environment.

Tracy Ragan – CEO and Co-Founder, DeployHub, Inc.

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. She is an industry leader who served on the Eclipse board as a founding member for 5 years. She can be reached on Twitter @TracyRagan