Continuous Delivery: Automating the Deployment Pipeline
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Value of Continuous Delivery</td>
<td>1</td>
</tr>
<tr>
<td>The Disconnect Between Build and Deploy</td>
<td>3</td>
</tr>
<tr>
<td>Automating and Integrating the Deployment Pipeline</td>
<td>4</td>
</tr>
<tr>
<td>Integrate the Entire Toolchain</td>
<td>5</td>
</tr>
<tr>
<td>Model, Manage, and Visualize Your Deployment Pipeline</td>
<td>6</td>
</tr>
<tr>
<td>Application Centric Deployment Automation</td>
<td>7</td>
</tr>
<tr>
<td>Summary</td>
<td>7</td>
</tr>
</tbody>
</table>
The Value of Continuous Delivery

Continuous Delivery is a software delivery practice that requires you to build and deploy software that can be released into production at any time. In order to satisfy this requirement, you need to continuously integrate the software changes from development, build, test and deploy the executables, and be ready to release into production, as the business requires it. This is a major change to how most software is delivered in today’s enterprise IT.

Why is Continuous Delivery important to Enterprise IT? Because it enables the business to rapidly respond to the expectations of their customers while improving the quality of their products at a lower cost.

Fig. 1

The need for Continuous Delivery
The goal of Continuous Delivery is to find ways to deliver high-quality, valuable software in an efficient, fast, and reliable manner. Continuous Delivery is about market speed and moving quickly from whiteboard to rollout, faster than your competition. This market speed means a shorter feedback loop and a faster time to value. With a shorter feedback loop, you fail faster, fix faster, adjust faster, and succeed faster. This agility provides a distinct business competitive advantage which is why companies like Amazon and Netflix are dominating their respective competitive landscape, transitioning into adjacent markets, and taking market leadership. These companies can adapt faster to the changing marketplace, and deliver new features and services faster than their competitors.

The central pattern of continuous delivery is the deployment pipeline which is an automated implementation of your application’s build, deploy, test, and release process. The deployment pipeline is instantiated whenever a change is made to an application.

![Deployment Pipeline Diagram]

**Fig. 2**

High-level overview of a deployment pipeline
While the deployment pipeline is an effective pattern for getting software from development to release, mapping an automated deployment pipeline across an entire enterprise can be challenging. Organizations tend to start the journey by taking a more incremental approach, starting with development and build processes that terminate with Continuous Integration, which is the foundation of the deployment pipeline.

The Disconnect Between Build and Deploy

The deployment pipeline has its foundation in the process of continuous integration. There can be no continuous delivery without continuous integration (CI). Continuous Integration enables development teams to automate the integration, build and testing of code within the development environment. Using a CI server such as Jenkins, the development and build environments become integrated and automated. Builds can be triggered by various means, such as a commit in a version control system, scheduling via a cron-like mechanism, execution of a new build when other builds have completed or by requesting a specific build URL.

Fig. 3

Process disconnects across organizations inhibit Continuous Delivery
The problem is that in many software projects, once the build is done there is a big disconnect between the build and deploy processes. While development teams own the development and build processes, the deployment process generally transitions to another organization, which has a different set of tools, scripts and processes to deploy and test the builds across the deployment pipeline. Frequently the toolchain integration with the CI server is poor or non-existent. A literal handoff occurs and the deployment team begins a manually intensive process to deploy release candidates to test and production environments. This disconnect becomes a bottleneck as the deployments queue up, leading to poor quality deployments and longer cycle times at a higher cost to the business.

Automating and Integrating the Deployment Pipeline

Micro Focus’s Deployment Automation product solves this disconnect by working seamlessly with continuous integration tools such as Jenkins, and automating the entire deployment pipeline. Deployment Automation bridges this development and operations chasm by integrating across the entire toolchain, managing and visualizing the deployment of the release candidates across the entire pipeline, all from an application centric perspective.
Deployment Automation ships with plug-ins for many common deployment processes, and others are readily available for a wide variety of tools, such as testing tools, middleware tools, databases, servers, and other deployment targets.

Fig. 4
Automating the Deployment Pipeline with Deployment Automation

Integrate the Entire Toolchain

Deployment Automation supports over 80 toolchain integrations by providing a flexible, robust, and extensible plug-in architecture. Plug-ins provide basic processing functions as well as integration with third-party tools. Deployment Automation ships with plug-ins for many common deployment processes, and others are readily available for a wide variety of tools, such as testing tools, middleware tools, databases, servers, and other deployment targets.
To integrate with Jenkins, simply download, install and configure the Deployment Automation Jenkins plugin. Deployment Automation integrates with automated test tools and can initiate automated testing within the deployment process. You can also automate the provisioning of the entire application stack by using plugins for Chef, AWS or VMware. For a complete list of supported plugins visit our documentation center.

Model, Manage, and Visualize Your Deployment Pipeline

The deployment pipeline models your process of getting your software from version control to your customers. Every time a change is committed, a new deployment pipeline is instantiated. Depending on the application and the change, the deployment pipeline may branch to different required environments and paths. Deployment to these environments could be in sequence, in parallel or offered as optional stages that are manually selected. The number of deployment pipelines to manage may increase depending on the number of applications, teams, requirements and environments.

Deployment Automation helps you manage all your deployment pipelines easily and comprehensively. You can easily create, import, export and visualize all the deployment pipelines and associated environments. You can optionally request a deployment process only on the first environment in the pipeline. Thereafter, you can only promote on subsequent environments. This ensures that the same configuration is used for each environment in the pipeline.

Auto-promotion, if selected, will automatically promote the release candidate if it was successfully deployed into the first environment. Upon successful deployment into that environment, the application is automatically promoted to the next environment in the pipeline sequence. If each promotion is successful, the auto-promotion continues from environment to environment until promotion into the last environment in the pipeline.

Deployment pipeline views provide a holistic view of how each environment is used in each pipeline and you can quickly understand what is deployed and where it’s deployed, all within a single product.
It’s important to ensure that application components that are tested together are deployed together. Deployment Automation solves this problem by supporting application processes and snapshots.

Application Centric Deployment Automation

Today’s applications contain components that may be drawn from several different sources and deployed to different application tiers, platforms and resources. Coordination and orchestration are essential in the successful deployment of these applications. It’s important to ensure that application components that are tested together are deployed together. Deployment Automation solves this problem by supporting application processes and snapshots.

Application processes can run manually, automatically on some trigger condition, or on a user-defined schedule. When a component has several processes defined for it, the application determines which ones are executed and in what order. For example, an n-tiered application might have a Web tier and a database tier. Once approved, both the Web tier and database tier are deployed concurrently. The application process calls the associated component processes that contain all of the deployment steps for deploying the components to specific resources.

A snapshot represents the current state of an application in the environment. Snapshots specify what combination of component versions you deploy together. They are models you create before deploying the application. A snapshot specifies the exact version for each component in the application. When a snapshot is created, Deployment Automation gathers together information about the application, including the component versions, for a given environment.

For most users, the snapshot is pulled through the pipeline. A snapshot is created based on the application’s state within the environment, thus capturing the different versions of the components at that time.

Summary

Continuous Delivery can dramatically improve the way you deliver value to your customer but can be challenging to implement. Many organizations have implemented continuous integration, which is the foundation of continuous delivery, but fail to seamlessly transition and automate the deploy and release process. Deployment Automation can integrate, manage and automate your entire deployment pipeline all from an application centric perspective.