Chapter 14
Design and Implementation of Service Management in DevOps Enabled Cloud Computing Models

Shelbee Eigenbrode
IBM, USA

Suheil Nassar
IBM Cloud, USA

ABSTRACT

This chapter examines the importance of including value-add service management practices early in the Continuous Integration/Continuous Delivery (CI/CD) pipeline. The authors will also address the importance of establishing a balance between the development and delivery of features with the development and delivery of practices that support overall infrastructure and service management capabilities. Without fully encompassing all of these practices, the DevOps benefits of reducing time-to-market for a set of features can be negated by a potential increase in security exposures as well as overall quality issues. Within this chapter, several key service management practices are identified as well as the importance of fully incorporating those practices into a DevOps adoption.

INTRODUCTION

According to a recent Gartner study, 25% of Global 2000 companies are expected to adopt DevOps by 2016 (Gartner Inc., 2015). Cloud computing has proven to be an ideal platform for enabling DevOps practices due to the inherent speed and agility cloud computing provides. Companies and enterprises of all sizes are rapidly embracing DevOps practices due to the recognized benefits including:

- Increased agility,
- Decreased time-to-market,

DOI: 10.4018/978-1-5225-0759-8.ch014
Improved quality, and
Ability to respond quickly to the customer.

The benefits of delivering new features and functions utilizing DevOps practices is resulting in all of the benefits above for many companies; however, isolating the use of DevOps practices to only features and functions can drastically limit the overall benefits that can be gained by adopting DevOps practices to the underlying infrastructure, platform and service management activities.

BACKGROUND

This chapter is not intended to provide a detailed description of all DevOps practices. There are many other resources available that speak at length about the various key practices included in the DevOps methodology. However, this chapter will focus on the practices of Continuous Integration and Continuous Delivery so those two practices will be defined for clarity within this chapter. Continuous Integration is the practice of frequently integrating code into a common automated build process that automatically validates the build then executes layers of automated testing and validation. An essential outcome of Continuous Integration is that source code changes from multiple team members are continuously being merged into a common code stream, or branch. Continuous Delivery is the practice of ensuring that the merged, tested, and validated code base can be automatically and reliably delivered at any time. Continuous Integration is a prerequisite for Continuous Delivery.

The adoption of Continuous Integration through Agile and DevOps practices allows Developers to frequently commit code and integrate it into a common automated build process including varying degrees of validation. The label of Developer in this case includes anyone that contributes code, which could potentially be someone from traditional functional roles such as operations or security. The label of code in this case should also not be limited to code providing a specific feature or function. Code can, and should, include infrastructure, service management and operational components.

Excluding service management and operational considerations early in the DevOps delivery pipeline can quickly negate all of the benefits gained through adopting DevOps practices. Service management and operational considerations includes all practices related to delivering and operating a given solution. This includes a broad variety of practices including but not limited to: patch management, configuration management, security, monitoring, change and incident management, and capacity management. This also includes the automation of day-to-day operational activities within these practices such as self-healing capabilities, automatic scaling, and automatic security remediation. The DevOps delivery pipeline essentially defines how code will move from development to production utilizing continuous integration and continuous delivery practices. A standard delivery pipeline will identify the activities and tools that enable Continuous Integration (CI) and Continuous Delivery (CD) practices. The detailed pipeline will also typically include identified practices such as: automated provisioning, configuration management, orchestration, quality gates and feedback loops. Within the pipeline, it’s key to understand that all of these activities are being integrated in small batches to optimize the ability to deliver frequent updates quickly. The common DevOps practices included in a CI/CD delivery pipeline provide the ideal mechanism for including and automating operational aspects quickly and early.