Zend Blueprint for Continuous Delivery

The PHP Company

Fundamentals of Continuous Integration

with Jenkins  and zend Server

by Slavey Karadzhov

Introduction

Continuous Delivery is a methodology, a mindset change and a leadership practice that focuses on how to achieve rapid application delivery throughout the software application lifecycle. This relies on the adoption of automation to streamline manual processes and enforce consistency and repeatability in the software delivery pipeline, as well as enhanced collaboration and shared metrics and processes across Dev and Ops teams [1].

One of the fundamental requirements for establishing Continuous Delivery is the implementation of an automated Continuous Integration system that automatically and consistently builds the application from code components and resources to a deployable package.

This paper will demonstrate the steps and advantages of implementing a Continuous Integration system utilizing Jenkins and the Zend Server Continuous Delivery platform.

The Zend Blueprint for Continuous Delivery

Zend's Blueprint for Continuous Delivery codifies best practices for implementing each step of the software delivery cycle and provides patterns, integrated with Zend Server, (Zend's platform for Continuous Delivery of PHP applications), that help implement those best practices.

The Blueprint provides guidelines on the underlying infrastructure should be constructed to deliver optimal value. The goal is to implement practical approaches that will accelerate and strengthen each step in the process of bringing applications more rapidly from code to production.

This model for Continuous Delivery is based on practical implementation concepts gathered through the experience of the Zend Professional Services team in working with numerous large and small clients.

Learn more about the Blueprint

www.zend.com/continuousdelivery

The Continuous Delivery Blueprint

Version Control	Continuous Integration			Infrastructure Automation	Release Automation	Quality Control & Approvals	Application Management
	Build	Code Quality	Packaging				
Developers working in branches. Trigger CI when merging into main master branch/ trunk	Pull and organize application artifacts and dependencies	Run unit tests and static code analysis tools to ensure code quality	Assemble application artifacts including dependencies & configuration into deployable validated package	Automated provisioning of infrastructure resources & platforms	Automated application deployment across all runtime environments	Project specific functional testing & production sign-off	Provide instant feedback and diagnostics on deployed application to development and operations teams

Continuous Integration: The Fundamentals

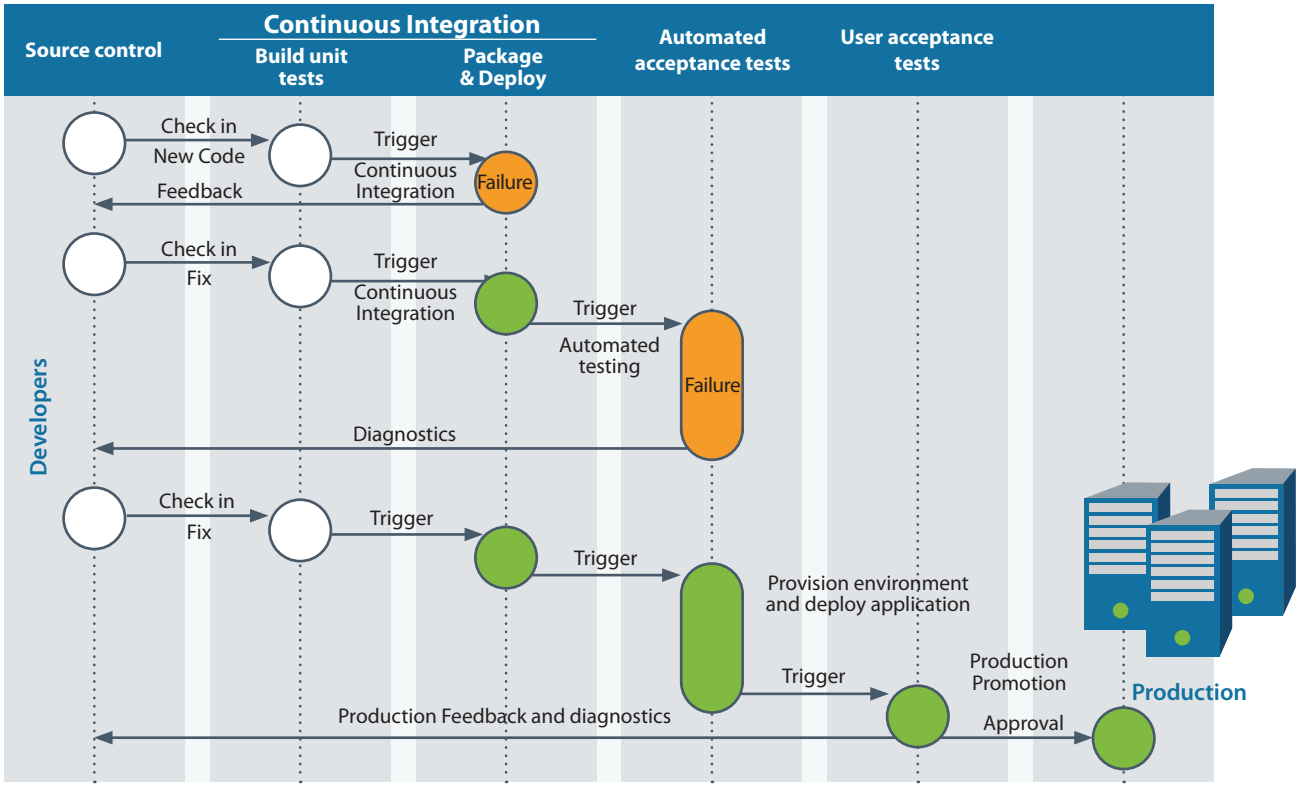
A key principle of a Continuous Delivery approach is that the application is always production ready. This means that all artifacts required to build the application are stored and managed in a version control system, and that code must be consistently and regularly checked in to the source mainline to ensure small increments between builds. Basically, the code in mainline is expected to be continuously production ready.

There are three important and foundational elements that underpin any Continuous Integration System.

1. **Commitment to building tests**. The first is one of the most important elements; the commitment of the development team to produce a comprehensive test suite at the unit level and functional level together with their code. This is essential to achieve the "guaranteed" level of code quality that can be production ready at any moment.
2. **Never break the build**. The goal is that the application must be ready to be built, packaged and deployed on every committed change. This (of course), means that broken or untested code should never be committed.
3. **Version Control**. Another foundational element to Continuous Integration is version control. To be ready to focus on implementing Continuous Integration, the code has to be managed by strict version control policies.

When the code is always production ready, managed by strict version control policies, and the development team has adopted an agile development and unit testing practice, organizations can move to implement a Continuous Integration environment.

Each distinct stage in the Continuous Delivery blueprint can be then implemented to complete the Continuous Delivery workflow. Below is a sequence diagram explaining the suggested flow of steps in a Continuous Delivery workflow.



Continuous Integration

The Continuous Integration approach is designed to create an automation environment ensuring that every change to the application results in a releasable version, and that any version can be built automatically at the push of a button. At a broader level, Continuous Delivery aims to make the entire end to end release process of that application hands free (automated) where the built application (from Continuous Integration system), is delivered frequently to the testing and then production environments. The goal of the Continuous Delivery cycle is to ensure consistency and high quality by delivering fast user centric feedback [2]. Continuous Integration is an important subset of Continuous Delivery.

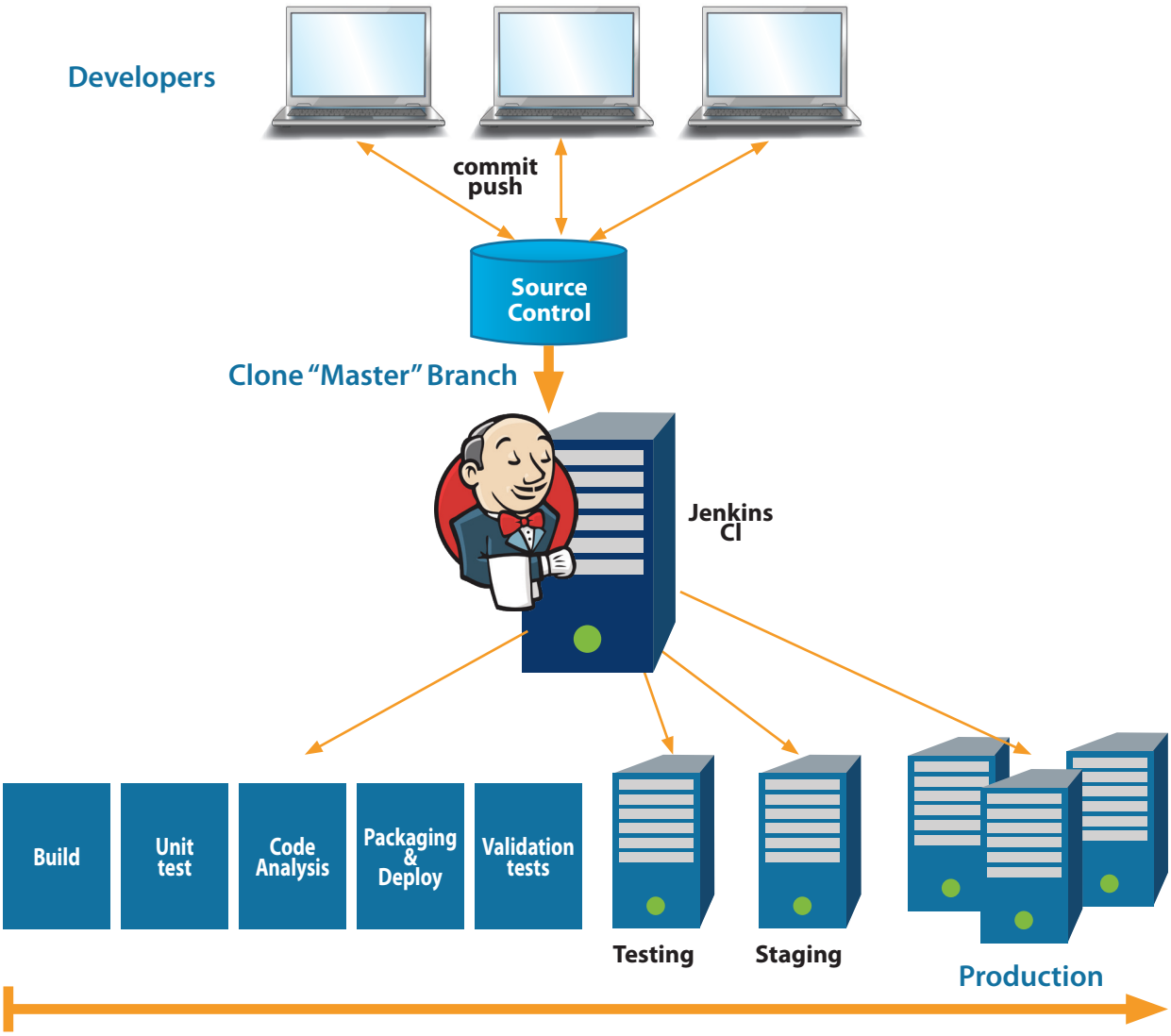
The goal of Continuous Integration is to automate traditionally manual stages in the application development process in which a feature complete version of the software is moving for the first time from development to the system integration and integration testing phase.

With Continuous Integration, applications are built from a very early stage in the development process at specific frequent intervals or on every change checked in by the developers. This effectively eliminates the need for integration testing and the cost associated with developers spending time on this phase. The enablement of frequent incremental builds and mandating a comprehensive automated testing process also allows developers to detect problems early and as a result, ensure higher application quality.

To illustrate a typical Continuous Integration workflow, consider the following scenario:

See illustration on next page

1. During the development of the application, the developers are working on their local machines and once they are ready to submit their new code changes they commit them to the central source code repository. In the diagram below the Source Control Management (or version control) system is a Git repository.
2. When the code is checked in to the mainline (or the release manager merges code changes from one of the developers), the new code addition to the main branch triggers a new release.
3. The Continuous Integration system (Jenkins in this case) monitors the version control system for changes and launches the build process.
4. The Continuous Integration server gets the source code from the repository.
5. The Continuous Integration server runs unit tests to validate the quality.
6. The Continuous Integration server packages the files into distributable units and deploys the package onto a test server and validates the package and basic functionality by running automated functional tests.
7. The Continuous Integration server deploys the same package on the testing environments for thorough user acceptance testing.
8. Once the acceptance tests are successful the same package is deployed on the production servers. Implementing infrastructure and release automation enables end to end automation and allows provisioning and deploying the application packages on all servers with a click of a button.



Benefits of Zend Server and Jenkins Integration

Zend Server, when integrated with Jenkins, enables a completely automated delivery process from the Continuous Integration system, into staging or production. Zend Server will automatically deploy the package onto a server, or a group of servers, whether in staging or production. If the target server is part of a Zend Server cluster, then all the other Zend Server nodes in the cluster will receive the same application and configuration automatically. This capability to package and deploy all of the code and configuration and libraries required for a PHP application onto a clustered environment in an automated manner also includes the ability to rollback instantaneously across a cluster to previous application versions if required.

Zend Server also has the benefit of making it simple to ensure that a common and consistent application stack and configurations are in place across dev, test and production. This includes library management capabilities that ensure framework versions and other dependencies are correctly managed across each environment. This makes it much easier to bring applications from Continuous Integration/ Development environment into staging and production without issues. Zend Server also alerts the team to prevent errors caused by configuration inconsistencies.

Zend Server: A Platform for Continuous Delivery of PHP Applications

The Zend blueprint for Continuous Delivery has the Zend Server application platform at its core. Zend Server is designed to help deliver your apps rapidly, iteratively and consistently.

Consistent. Automated. Collaborative. No problem.

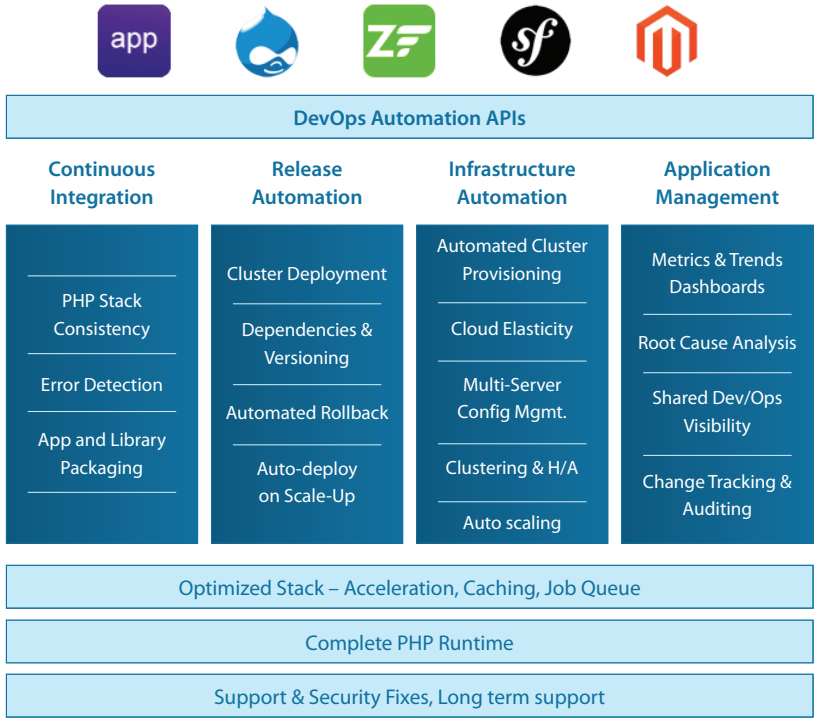
Standardize packaging and deployment testing.

Instantly provision standardized application infrastructure.

Automate app deployment, versioning, and rollback.

Understand & fix production issues and iterate rapidly.

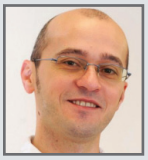
How does Zend Server enable Continuous Delivery?



References and resources:

[1] The Zend Blueprint for Continuous Delivery, Zend Technologies, www.zend.com/continuousdelivery

[2] http://www.thoughtworks.com/de/continuous-delivery



About the Author

Slavey Karadzhov is a Senior Consultant at Zend Technologies. He is a PHP 5/5.3, Zend Framework and MySQL Certified Engineer. Proud owner of two master degrees, in Computer Science from Sofia University, Bulgaria and in Software Technologies from Stuttgart University of Applied Science, Germany. Slavey is a strong open source supporter and recognized software innovator as well as the author of a Zend Framework 2 book called "Learn ZF2: Learning By Example" (http://learnzf2.com).

Zend Professional Services lead implementations of agile methodologies, application design, optimization and continuous delivery solutions at global enterprises running mission-critical PHP applications. The consultants focus on mentoring customers' development and operations teams to deliver faster releases of high-quality PHP applications with higher performance and availability.

About Zend

Zend partners with businesses to rapidly deliver modern apps across web, mobile and cloud. We helped establish the PHP language, which today powers over 200 million applications and web sites. Our flagship offering, Zend Server, is the leading Application Platform for developing, deploying and managing business-critical applications in PHP. More than 40,000 companies rely on Zend solutions, including NYSE Euronext, BNP Paribas, Bell Helicopter, France Telecom and other leading brands worldwide.

Corporate Headquarters: Zend Technologies, Inc. 19200 Stevens Creek Blvd. Cupertino, CA 95014, USA · **Tel** 1-408-253-8800 · **Fax** 1-408-253-8801
Central Europe: (Germany, Austria, Switzerland) Zend Technologies GmbH, St.-Martin-Str. 53, 81669 Munich, Germany · **Tel** +49-89-516199-0 · **Fax** +49-89-516199-20
International: Zend Technologies Ltd. 12 Abba Hillel Street, Ramat Gan, Israel 52506 · **Tel** 972-3-753-9500 · **Fax** 972-3-613-9671
France: Zend Technologies SARL, 105 rue Anatole France, 92300 Levallois-Perret, France · **Tel** +33-1-4855-0200 · **Fax** +33-1-4812-3132
Italy: Zend Technologies, Largo Richini 6, 20122 Milano, Italy · **Tel** +39-02-5821-5832 · **Fax** +39-02-5821-5400
Ireland: Zend Technologies, The Digital Court, Rainsford Street, Dublin 8, Ireland · **Tel** +353-1-6908019