INTRODUCTION TO CI/CD: CONCEPTS AND TERMINOLOGY

Aleksey Tsalolikhin

aleksey@verticalsysadmin.com

25 Oct 2018

TABLE OF CONTENTS

DEFINITION OF TERMS: CI/CD

CONTINUOUS INTEGRATION (CI)

Continuous
(adjective) Forming an unbroken whole; without interruption.
-- Oxford Dictionaries

Integration
(noun) The action or process of integrating.
-- Oxford Dictionaries

Integrate
(verb) to unite with something else; to incorporate into a larger unit
-- Merriam-Webster

CONTINUOUS INTEGRATION (CI)

> ***Continuous integration (CI)*** *is the practice of merging all developer working copies to a shared mainline several times a day. ... The main aim of CI is to prevent integration problems.*

-- Wikipedia entry "Continuous Integration"

CONTINUOUS INTEGRATION (CI)

> ***Continuous Integration*** *is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly."*

-- "Continuous Integration", Martin Fowler (Chief Scientist at ThoughtWorks, vendor of the first commercial Continuous Integration server)

CONTINUOUS DEPLOYMENT (CD)

> ***Continuous Deployment*** *is closely related to Continuous Integration and refers to the release into production of software that passes the automated tests.*

-- ThoughtWorks.com

CONTINUOUS DELIVERY (CD)

> ***Continuous Delivery*** *is sometimes confused with continuous deployment. Continuous deployment means that every change is automatically deployed to production. Continuous delivery means that the team ensures every change can be deployed to production but may choose not to do it, usually due to business reasons. In order to do continuous deployment one must be doing continuous delivery.*

-- Wikipedia

PIPELINE

Pipeline
(noun) A system through which something is conducted, especially as a means of supply: "Farther down the pipeline are three other approaches to vaccine development" (Boston Globe).
-- American Heritage Dictionary of the English Language

Conduct
(verb) To serve as a medium for conveying; transmit: "Some metals conduct heat."
-- American Heritage Dictionary

Convey
(verb) To transport or carry to a place: "Pipes were laid to convey water to the house"
-- Oxford Dictionaries

PIPELINE

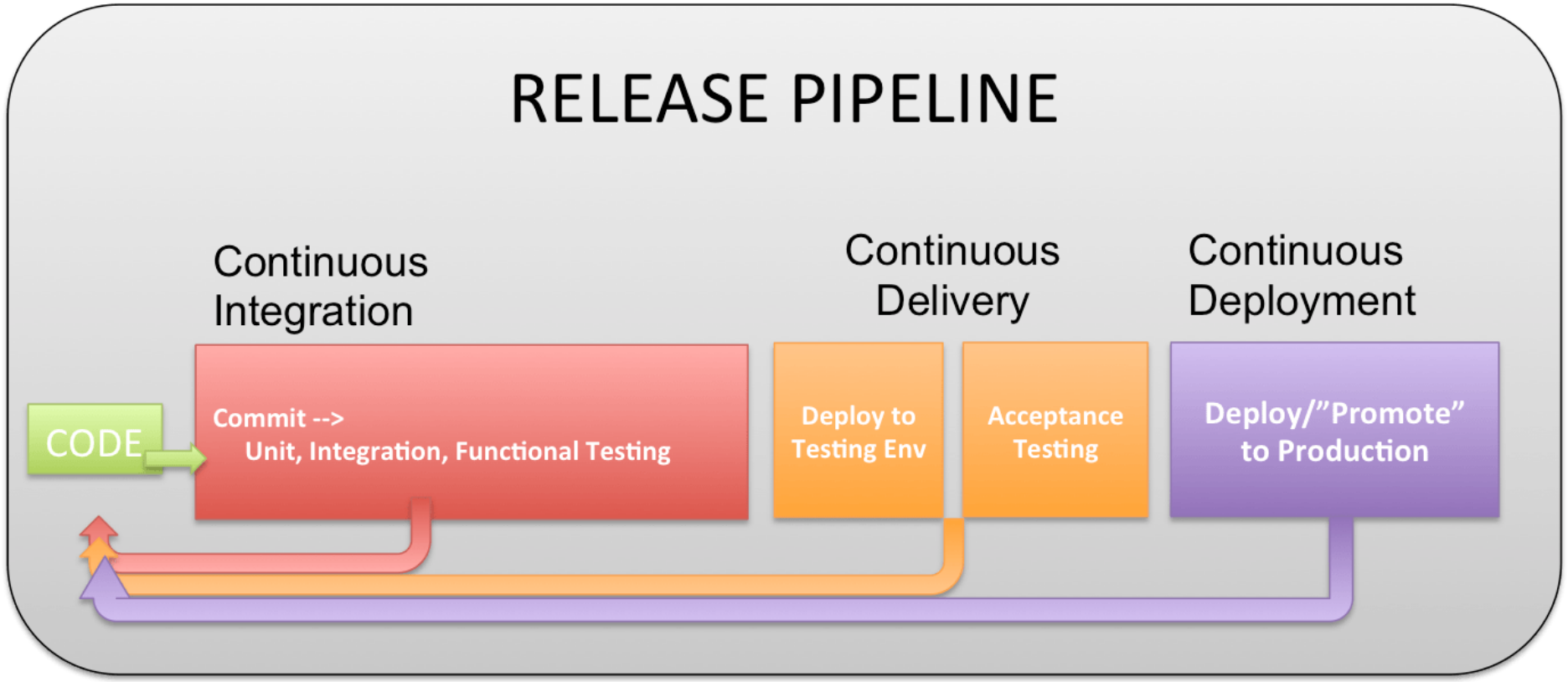


Image credit: "I want to do Continuous Deployment" article on devops.com

TESTING

This section will define terms pertaining to testing, specifically different types of testing.

UNIT TESTING

> ***Unit testing*** *The type of testing where a developer (usually the one who wrote the code) proves that a code module (the "unit") meets its requirements.*

-- Free On-Line Dictionary of Computing

UNIT TESTING

> *The primary goal of **unit testing** is to take the smallest piece of testable software in the application, isolate it from the remainder of the code, and determine whether it behaves exactly as you expect. ... Unit testing has proven its value in that a large percentage of defects are identified during its use.*

-- Microsoft Developer Network

UNIT TESTING

> *In computer programming, **unit testing** is a software testing method by which individual units of source code ... are tested to determine whether they are fit for use.*

-- Wikipedia

INTEGRATION TESTING

Integration Testing

A type of testing in which software and/or hardware components are combined and tested to confirm that they interact according to their requirements. Integration testing can continue progressively until the entire system has been integrated.

-- Free On-Line Dictionary of Computing

INTEGRATION TESTING

> ***Integration testing*** *... is the phase in software testing in which individual software modules are combined and tested as a group. ... Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.*

-- Wikipedia

FUNCTIONAL TESTING

> ***Functional testing*** *... bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered. ... Functional testing usually describes what the system does.*
>
> *Functional testing does not imply that you are testing a function (method) of your module or class. Functional testing tests a slice of functionality of the whole system.*

-- Wikipedia entry, "Functional testing"

ACCEPTANCE TESTING

> ***Acceptance testing*** *Acceptance testing is a test conducted to determine if the requirements of a specification or contract are met. ... [It is defined as] formal testing with respect to user needs, requirements, and business processes conducted to determine whether a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system. Acceptance testing is also known as user acceptance testing (UAT), end-user testing, operational acceptance testing (OAT) or field (acceptance) testing.*

-- Wikipedia entry for "Acceptance testing"

BENEFITS OF CONTINUOUS INTEGRATION

> *Releasing software frequently to users is usually a time-consuming and painful process. Continuous Integration and Continuous Delivery (CI/CD) can help organizations ... by automating and streamlining the steps involved in going from an idea ... to the delivered product to the customer.*

-- OpenShift blog, "CI/CD with OpenShift"

BENEFITS OF CONTINUOUS INTEGRATION

INTEGRATE AT LEAST DAILY

> *Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.*
>
> *By integrating regularly, you can detect errors quickly, and locate them more easily.*

-- ThoughtWorks.com

BENEFITS OF CONTINUOUS INTEGRATION

SOLVE PROBLEMS QUICKLY

Because you're integrating so frequently, there is significantly less back-tracking to discover where things went wrong, so you can spend more time building features.

Continuous Integration is cheap. Not integrating continuously is expensive. If you don't follow a continuous approach, you'll have longer periods between integrations. This makes it exponentially more difficult to find and fix problems. Such integration problems can easily knock a project off-schedule, or cause it to fail altogether.

-- ThoughtWorks.com

BENEFITS OF CONTINUOUS INTEGRATION

Continuous Integration brings multiple benefits to your organization:

- *Say goodbye to long and tense integrations*
- *Increase visibility enabling greater communication*
- *Catch issues early and nip them in the bud*
- *Spend less time debugging and more time adding features*
- *Build a solid foundation*
- *Stop waiting to find out if your code's going to work*
- *Reduce integration problems allowing you to deliver software more rapidly*

-- ThoughtWorks.com

BENEFITS OF CONTINUOUS INTEGRATION

> *Continuous Integration doesn't get rid of bugs, but it does make them dramatically easier to find and remove.*

— Martin Fowler, Chief Scientist, ThoughtWorks

ORIGIN OF CONTINUOUS INTEGRATION

In the late 1990's, Don Wells introduced continuous integration at Chrysler:

> *Don proposed to the team that they set up an extra computer on an empty desk where all integration would take place. They would integrate and release new code to the repository when ever they wanted without prior permission so long as it ran all the unit tests. Management hated the idea. The team was mixed about it. Management played their trump card by not allowing Don to have an extra computer. So Don simply moved his own computer to the empty desk and told everyone it was the integration station. He wanted to do more pair programming anyway.*

(continued on next slide)

ORIGIN OF CONTINUOUS INTEGRATION

> *The real prize in that change was what came to be known as collective ownership. The entire team owns the entire code base. The entire team is responsible for developing and extending the system design. The team worked together cooperatively at a much faster pace than anyone expected. Don has some rough estimates and believes the team was going six and a half times faster.*

-- www.agile-process.org web page "Your host: Don Wells"

ORIGIN OF CONTINUOUS INTEGRATION

Martin Fowler was involved in the work at Chrysler.

> *This was my first chance to see Continuous Integration in action with a meaningful amount of unit tests. It showed me what was possible and gave me an inspiration that led me for many years.*

-- Martin Fowler's article "Continuous Integration"

ORIGIN OF CONTINUOUS INTEGRATION

ThoughtWorks created the first known Continuous Integration server, Cruise, in 2001. This Java-based tool was later open-sourced and renamed CruiseControl.

Around 2007, after finding CruiseControl limiting, [Jez] Humble worked alongside a ThoughtWorks team in Beijing to create the [ThoughtWorks] tool that later became Go (now styled GoCD).

In 2010, Humble and ex-ThoughtWorker Dave Farley published the first book on continuous delivery.

-- Wikipedia entry, "ThoughtWorks"

DEFINITION OF TERMS: DEVOPS

> *DevOps is... an umbrella concept that refers to anything that smoothes out the interaction between development and operations.*

-- "What is Devops?", Damon Edwards

DEVOPS

> *The term “DevOps” typically refers to the emerging professional movement that advocates a collaborative working relationship between Development and IT Operations, resulting in the fast flow of planned work (i.e., high deploy rates), while simultaneously increasing the reliability, stability, resilience and security of the production environment.*

-- "Top 11 Things You Need to Know about DevOps", Gene Kim's ITRevolution.com

DEVOPS

> *Today, DevOps is an understood set of practices and cultural values that has been proven to help organizations of all sizes improve their software release cycles, software quality, security, and ability to get rapid feedback on product development.*

-- 2017 State of DevOps Report

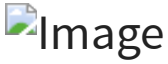
DEVOPS

> *Currently, **DevOps** is more like a philosophical movement ... At this early stage we're in, DevOps is more like a vibrant community of practitioners who are interesting in replicating the performance outcomes and culture as exemplified in the seminal John Allspaw/Tim Hammond 2009 Velocity presentation about doing "ten deploys a day" at Flickr.*

-- "DevOps Cookbook" article by Gene Kim

DEVOPS

John Willis, co-author of "The DevOps Handbook", with the Los Angeles chapter of LOPSA



HOW CI/CD RELATES TO DEVOPS

> *Continuous delivery and DevOps have common goals and are often used in conjunction, but there are subtle differences.*
>
> *While continuous delivery is focused on automating the processes in software delivery, DevOps also focuses on the organization change to support great collaboration between the many functions involved.*

-- Wikipedia entry for "DevOps"

HOW CI/CD RELATES TO DEVOPS

AUTOMATING DEPLOYMENTS

*The **deployment pipeline**, first defined by Jez Humble and David Farley in their book "Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation", ensures that all code checked in to version control is automatically built and tested in a production-like environment. By doing this, we find any build, test or integration errors as soon as a change is introduced, enabling us to fix them immediately. Done correctly, this allows us to always be assured that we are in a deployable and shippable state.*

To achieve this, we must create automated build and test processes that run in dedicated environments.

-- "The DevOps Handbook", Chapter 10 "Enable Fast and Reliable Automated Testing"

ADOPTION

Automation is a key differentiator for organizations. ... Automation is a huge boon to organizations. High performers automate significantly more of their configuration management, testing, deployments and change approval processes than other teams. The result is more time for innovation and a faster feedback cycle. ... DevOps teams increased from 16% in 2014 to 19% in 2015 to 22% in 2016 to 27% in 2017.

-- 2017 State of DevOps Report

> *We found that the following all positively affect continuous delivery: comprehensive use of version control; continuous integration and trunk-based development; integrating security into software delivery work; and the use of test and deployment automation. Of these, test automation is the biggest contributor.*

-- 2017 State of DevOps Report

The biggest contributor to continuous delivery — bigger even than test and deployment automation — is whether a team can do all of the following:

- *Make large-scale changes to the design of its system without permission from someone outside the team.*
- *Make large-scale changes to the design of its system without depending on other teams to make changes in their own systems, or creating significant work for other teams.*
- *Complete its work without needing fine-grained communication and coordination with people outside the team. For example, not having to check 12 Google calendars to get feedback.*
- *Deploy and release its product or service on demand, independently of other services the product or service depends upon.*
- *Do most of its testing on demand, without requiring an integrated test environment.*
- *Perform deployments during normal business hours with negligible downtime.*

BASIC TASKS: BUILD, TEST, DEPLOY

FROM IDEA TO PRODUCTION

Idea --> Code --> Build --> Test --> Deploy

IDEA

Example:

"Let's write a 'hello world' program!"

CODE

Example:

```
$ cat hello.c
# include <stdio.h>
main()
{
    printf("Hello World");
}
$
```

BUILD

Build
(verb) *Computing* Compile (a program, database, index, etc.).
(noun) *Computing* A compiled version of a program; the process of compiling a program.

-- Oxford Dictionaries

Compile
Computing (of a computer) convert (a program) into a machine-code or lower-level form in which the program can be executed.

-- Oxford Dictionaries

Example:

```
$ gcc -o hello hello.c
$
```

TEST

Test
(noun) a process designed to find out whether something such as a machine or weapon works correctly or whether a product is satisfactory

-- Macmillan Dictionary

Example 1 - testing with Make - a successful test:

```
$ cat Makefile
test:
        ./hello > output.txt
        diff -q correct.txt output.txt

$ cat correct.txt
Hello World
$ make test
./hello > output.txt
diff -q correct.txt output.txt

$
```

Example 1b - induce failure:

```
$ date > correct.txt

$ make test
./hello > output.txt
diff -q correct.txt output.txt
Files correct.txt and output.txt differ
make: *** [Makefile:3: test] Error 1

$
```

Example 2 - Testing with Bash Automated Testing System (bats)

```
$ cat hello_test.bats
#!/usr/bin/env bats

@test "'hello' outputs 'hello world'" {
  result="$(./hello)"
  [ "$result" == "Hello World" ]
}
$ bats hello_test.bats
 ✓ 'hello' outputs 'hello world'

1 test, 0 failures
$
```

DEPLOY

Deploy
(verb) To place (people or other resources) into a position so as to be ready to for action or use.

-- Webster's 1913 Dictionary

Deploy
(verb) Bring into effective action; utilize.
"They are not always able to deploy this skill."

-- Oxford Dictionaries

DEPLOY

Example: deploying on the local host:

```
$ sudo cp hello /usr/local/bin/hello
$
```

Example: deploying to a remote environment:

```
$ scp hello root@production:/usr/local/bin/hello
$
```

BIBLIOGRAPHY

- Continuous Integration, article by Martin Fowler
- Continuous Deployment at IMVU: Doing the impossible fifty times a day, article by Timothy Fitz
- Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation book by Jez Humble and David Farley
- Continuous Integration: Improving Software Quality and Reducing Risk book by Paul M. Duvall, et al.
- 10+ Deploys Per Day: Dev and Ops Cooperation at Flickr video, 2009 Velocity presentation
- The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win by Gene Kim, et al.
- The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations by Gene Kim, et al.

P.S. WHAT ABOUT AGILE?

A common question I get is, how does CI/CD relate to Agile?

Agile software development values early delivery, short feedback loops and continuous improvement; the principles and technology of CI/CD enable that.