Delivering Value with Confidence
"Quality Delivery Pipeline"

Joseph W. Yoder, The Refactory, Inc. – USA
Hironori Washizaki, Waseda University – Japan
Ademar Aguiar, Faculdade de Engenharia, Universidade do Porto – Portugal

Many software development processes such as Agile and Lean focus on the delivery of working software that meets the needs of the end users. Many of these development processes help teams respond to unpredictability through incremental, iterative work cadences and through empirical feedback. There is a commitment to quickly deliver reliable working software that has the highest value to the those using or benefiting from the software. This can be done reliably if there is time taken to build confidence into the process and architecture. Continuous integration and delivery has been incorporated into many successful Agile processes. DevOps as a software engineering practice unifies software development (Dev) and software operation (Ops). To assist with quality delivery in these practices you need to provide a quality "Delivery Pipeline" to help assure the delivery meets the requirements and proper validation and checks are done before integration or release. This paper will focus on the "Delivery Pipeline" as a practice that can help sustain Delivering Value with Confidence.

Categories and Subject Descriptors
• Software and its engineering ~ Agile software development • Social and professional topics
• Software and its engineering ~

General Terms
Agile, Sustainable Delivery, Patterns, Confidence,

Additional Keywords and Phrases
Agile Software Development, Continuous Integration, Continuous Delivery, Innovation, Reliability,

ACM Reference Format:
Yoder, J.W., Washizaki, H., Aguiar A., 2018. Delivering Fast with Confidence "DevOps". HILLSIDE Proc. of Conf. on Pattern Lang. of Prog. 7 (October), 7 pages.

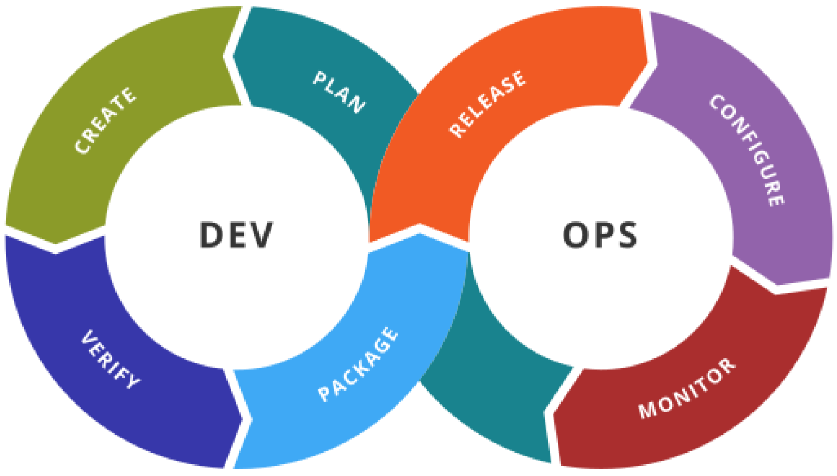
Author's email address: joe@refactory.com, washizaki@waseda.jp, ademar.aguiar@fe.up.pt

Introduction

There are many proven practices that are used during development to help sustain delivery at a good pace while maintaining confidence in the system. Some of these are related to Agile or Lean practices such as short delivery cycles, good testing, clean code, and continuous integration. Small delivery size with regular feedback through incremental releases has proven itself over the years and has become the de-facto standard for most agile practices. Delivery smaller pieces getting regular feedbacks so that adjustments can be made of problems noted earlier and fixed. Also keeping your code clean by including refactoring and evolving your architecture has become accepted as a key principle of most agile practices.

However blindly following Agile practices isn't sufficient to help sustain delivering quality software at a good pace with confidence. Quite often system qualities, such as reliability, scalability, or performance, are overlooked or simplified until late in the development process, thus causing time delays due to extensive refactoring and rework of the software design required to correct quality flaws. There are other practices that can help with this, such as reflection time, finding way to improve, and building quality into the process and product from the start [YWA, YW, YWW].

Continuous Integration and Delivery has become the defacto standard for most Agile and Lean processes. Many of these development processes help teams respond to unpredictability through incremental, iterative work cadences and through empirical feedback. Regular delivery of reliable working software that has the highest value to the those using or benefiting from the software is becoming more important for the success of many companies. Note that the first principle behind the Agile manifesto is: *"our highest priority is to satisfy the customer through early and* ***continuous delivery*** *of valuable software."* That has lead to the acceptance of DevOps which advocates automation and monitoring at all steps of software construction, from integration, testing, releasing to deployment.



https://en.wikipedia.org/wiki/DevOps#/media/File:Devops-toolchain.svg (CC BY-SA 4.0)

We previously outlined "Slack Time" as a core pattern for Delivering Value with Confidence. However there are many other practices to help with confidence. If we deploy software more frequently, it is important to ensure stability and reliability in our systems. The goal is to be able to safely and quickly release our software in a sustainable way. Continuous Delivery and DevOps have focused on both automation and team practices that can help with this. Automation and a good pipeline that gives confidence has proven invaluable to help sustain delivery with confidence. This paper will focus on the "Delivery Pipeline" pattern as a key practice to help sustain Fast Delivery with Confidence.

Quality Delivery Pipeline

"I come back to the same thing: We've got the greatest pipeline in the company's history in the next 12 months, and we've had the most amazing financial results possible over the last five years, and we're predicting being back at double-digit revenue growth..." — Steve Ballmer

https://pxhere.com/en/photo/776653 (CCO Public Domain)

Continuous Integration (CI) has become a key factor for most software development teams as a means to safely evolve the code ensuring that we have not broken anything that we have to interact with. A step to be successful with this is to setup servers and scripts for continuous integration. Continuous Delivery is an extension of this by taking the automatic or semi-automatic promotion of builds and process them into something that can be delivered.

Continuous Delivery is explicitly mentioned in the first of the 12 principles of Agile; "Our highest priority is to satisfy the customer through early and ***continuous delivery*** of valuable software." The goal is to take changes from version control to production by making deployment to different environments as automated and as quick as possible (hopefully within minutes).

How can we create a process that helps us deploy quickly and safely getting feedback during the process?

❖ ❖ ❖

Businesses can get very busy trying to meet the requirements where there is barely time, resources or people to do what is needed to keep a project afloat.

Building and releasing into production environments can be tedious and error prone.

Automating the testing and deployment process is challenging and requires a lot of expertise.

Lack of validation or testing of your release can be dangerous and costly.

❖ ❖ ❖

Therefore, develop and automate a quality delivery pipeline. This pipeline needs to be composed of steps to get and build the current version, run various validations on the build, and when successfully validated push the build into a staging or production environment.

Figure 1 outlines the core of what a minimum pipeline should be. In a sense it is analogous to pipes and filters where each step in the pipeline is a quality filter such as code quality, technical debt, security, etc. It is important that the pipeline be small and fast, instead of long and slow.

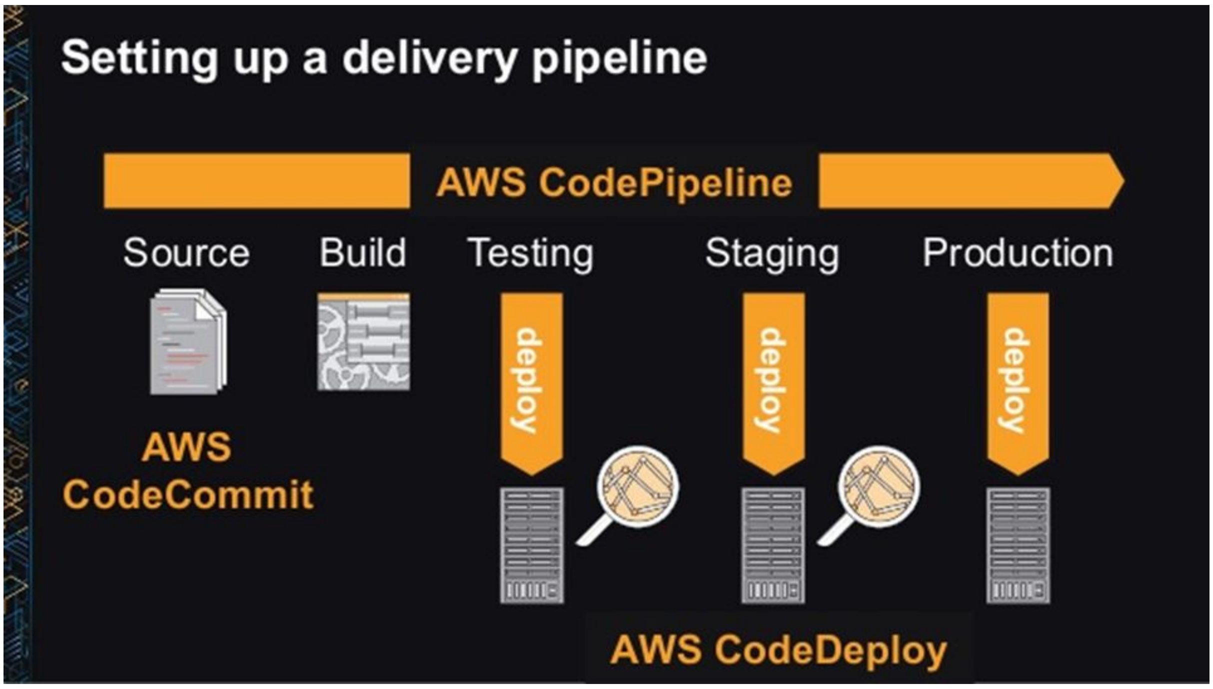


Figure 1: Minimum Delivery Pipeline[1]

Note that the minimum pipeline must get the code and build it, usually some form of testing is done such as unit testing, then if successful the build can be pushed into either a staging environment for QA testing and into the production environment. Although this minimum pipeline works it is not "ideal" and does not give confidence as there can be many issues with the release. It is important if we want to deliver with confidence that we validate important qualities before releasing into production. Also, Automating As you Go [ref] makes it so the process is quicker, can be done more easily, and is less error prone.

"Ideal" Quality Pipeline

A pipeline that offers more confidence would add involved testing steps and also validate important qualities. Some of these qualities might be code quality while others might be security, performance, or other architectural qualities. The following is an example of a more complete quality pipeline:

Build (Clone/Clean/Build) -> Test (unit/integration/contract) -> Code Quality (Sonar/CodeMetrics/TestCoverage/Fortify) -> Deploy QA (DB/App) -> Tests (Smoke/Acceptance/Full Integration/Manual) -> Sandbox -> Pre-healthTest (Trap Monitor) -> Deploy Blue/Green -> Health Smoke Tests -> Staged Release (limited impact) -> Full Deploy -> Notifications

*** Item in "red" are optional steps and are useful for confidence**

Note that validating the dependencies are healthy through a pre-health test before release minimized potential issues when going into a production environment. Building Monitors as

[1] By Julien Simon - https://www.slideshare.net/JulienSIMON5/aws-codecommit-codedeploy-codepipeline

part of the release and include as part of the pipeline is key (trap monitor can be used as a good default monitor).

Related patterns/practices to gain confidence and assist with a quality pipeline:

- Small Incremental Releases (keep releasing small changes into production)
- Automate as you go (automate what you can as soon as you can)
- Continuous Inspection (getting regular feedback)
- Blue/Green (Separate Deployment from Release)
- Staged Releases (first in a safe environment and spread out)
- Health Checks (Smoke Tests to make sure things are ok)

Summary

DevOps has become popular as a methodology that combines software development with deployment and operations with an overall goal of delivering value more frequently and with confidence. Having a *Quality Delivery Pipeline* is key to being able to reliably deliver quickly and with confidence. This paper outlined the beginning principles of that are part of a good delivery pipeline.

A good pipeline breaks down the software delivery process into various stages that focus on verifying the quality of the system before release. This can be from validating functionality to verifying code quality and ensuring certain system qualities such as security and performance are being met. The pipeline provides visibility to the team and everyone involved in delivering and maintaining the system. A minimum pipeline should include: build, test, quality checks, and deployment verification.

Acknowledgements

We'd like to thank our shepherd Foutse Khomh for his valuable comments and feedback during the SugarLoaf PLoP 2018 shepherding process. Finally we'd like to thank our 2018 SugarLoaf PLoP Writers Workshop Group, XXX, YYY, and ZZZ for their valuable comments and suggestions.

References ***NOTE TO SHEPHERD THESE WILL BE UPDATED***

[DeMarco] DeMarco T., *Slack: Getting Past Burnout, Busywork, and the Myth of Total Efficiency.* New York: Broadway Books a division of Random House, 2002.

[Kahneman] Kahneman, D., *Thinking, Fast and Slow.* New York: Farrar, Straus and Giroux, 2013.

[YWA] Yoder J., Wirfs-Brock R., and Aguilar A., "QA to AQ: Patterns about transitioning from Quality Assurance to Agile Quality," 3rd Asian Conference on Patterns of Programming Languages (AsianPLoP), Tokyo, Japan, 2014.

[YW] Yoder J. and Wirfs-Brock R., "QA to AQ Part Two: Shifting from Quality Assurance to Agile Quality," 21st Conference on Patterns of Programming Language (PLoP 2014), Monticello, Illinois, USA, 2014.

[YWW] Yoder J., Wirfs-Brock R. and Washizaki H., "QA to AQ Part Three: Shifting from Quality Assurance to Agile Quality: Tearing Down the Walls," 10th Latin American Conference on Patterns of Programming Language (SugarLoafPLoP 2014), Ilha Bela, São Paulo, Brazil, 2014.