An Example of a Continuous Integration Delivery Pipeline

by Seth - Monday, September 26, 2016

http://www.sethgagnon.com/an-example-of-a-continuous-integration-delivery-pipeline/

This article will provide an example delivery pipeline used through continuous integration, as well as, the DevOps tools involved in the automation of building, testing, and deploying code through your SDLC.

Background

In the IT world today, you can't go to any organization, regardless of the industry, without hearing the term "DevOps." In short, it originally stems from the effort to have better communication and synergies between your Development organization and your Operations organization, hence the term DevOps. The main focus was to improve the efficiency, quality, and speed to market within the software development world. Furthermore, you may hear and see a lot about continuous integration and continuous delivery/deployment. There are differences between the two for sure, but for the purposes of simplicity, I would like to really focus on breaking down the different stages of the process. This way you can get the basics and hit the ground running with a delivery pipeline in your organization. The delivery pipeline can be broken down into a few major buckets of work, or stages, as mentioned below.

In my opinion, and again trying to keep things simple for now, the stages can be broken down as follows:

1. Source Code Control (Management)
2. Build Automation
3. Unit Test Automation (could also include Integration Testing here as well)
4. Deployment Automation
5. Monitoring – not included in this discussion, and can be added at any time

I have outlined the progression of these stages in Figure 1 below. Let's review each of the stages in a little more detail.

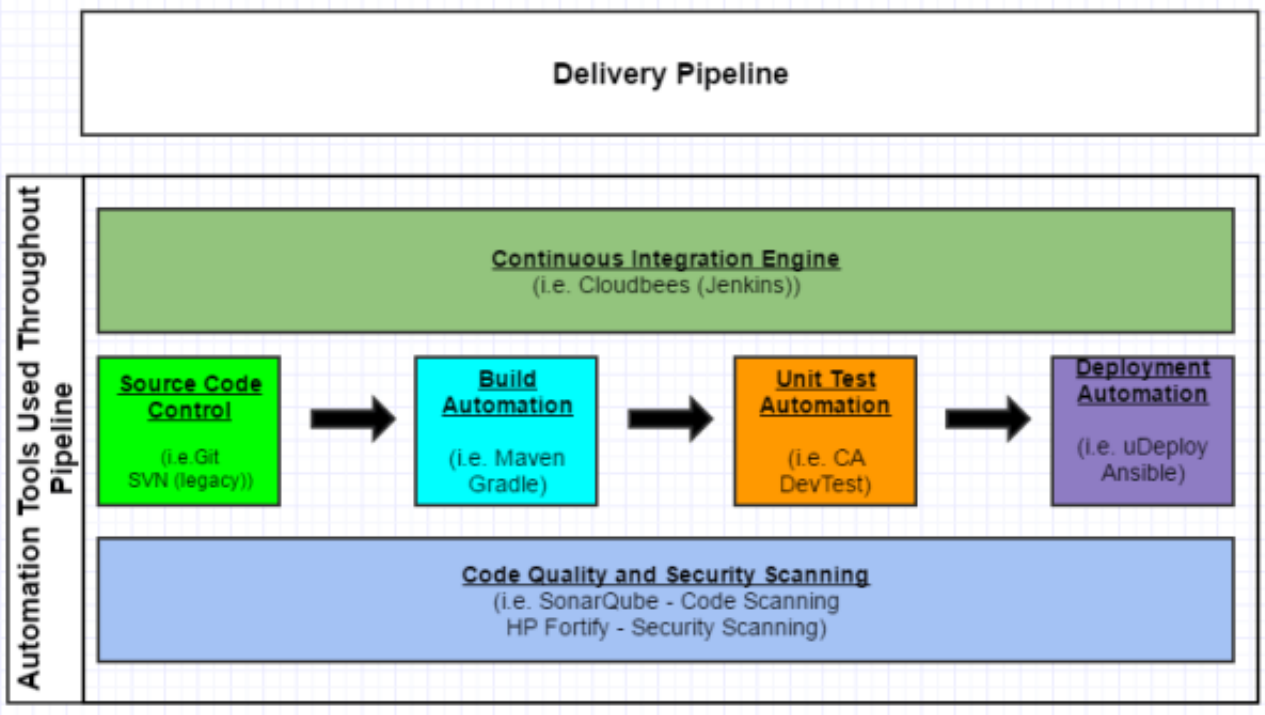

Figure 1 – Continuous Integration Delivery Pipeline Sample

Source Code Control (Management)

Background

Source code management, or source code control, is certainly not a new topic. This has been around for decades and has evolved over time. The basics here are that your organization stores its code in a source code control system or repository, so that it can be tracked, maintained, versioned, and audited. You do not want the developers storing the code on their laptops or virtual machines and trust that will suffice for managing the code.

Possible Tools

1. Git is probably the most widely used SCM system out there. It is an open source system.
2. Subversion (SVN) has been around for quite some time. It is also an open source system. It is still heavily used across many organizations out there, but there has been more of a push towards Git. If you are just starting out, I would highly recommend using Git.

Git and Subversion comparison can be found here.

Build Automation

Background

Once a source code management system is in place and actively being used by your development team, the team will need to be able to compile and build their code. This is probably the first step in the whole

chain of continuous integration events. This is what gets the ball rolling. The code needs to build cleanly before you can even think about deploying out to your environments for testing and production.

Possible Tools

1. Gradle is an open source build automation system. Pretty widely used by top companies like Netflix, Google, and LinkedIn.
2. Maven is another open source build automation system.

Gradle and Maven comparison can be found here and here.

Unit Test Automation

Background

Developers unit test their code to ensure that the functionality they are building works as expected. In an ideal world, the development team should be saving these unit tests, so that they can be reused and also put into a regression test bed.

Possible Tools

There are multiple tools out there for helping developers unit test their code. Many of these tools are open source and can be used freely.

1. JUnit is an open source unit test framework. This is pretty widely used in the industry.
2. CA DevTest allows for the automation of unit testing, as well as a few other bells and whistles, like service virtualization.

Deployment Automation

Background

For the last stage in the process, delivery teams need to deploy their code/applications out to various test environments and, of course, production. To reduce errors and overhead in the deployment process, while increasing speed to market, this step can be automated through a variety of tools and methods.

Possible Tools

1. IBM Urbancode uDeploy allows you to model a process and orchestrate your deployment. This process can then be repeated across all your environments, and of course tweaked for each environment as needed.
2. Ansible is an open source IT automation tool. It can be used for everything from configuration management to product installation to application deployments.This tool is rapidly gaining acceptance and momentum in the DevOps community.

Conclusion

In conclusion, this is a quick overview to be able to get you started down the path of continuous integration and the DevOps world. Below in Figure 2 is a sample of what the whole flow looks like from committing your code to your repo to deploying the code to an environment. I thought this would help put all the stages mentioned into perspective. You can see how the tools interact with each other (some of these tools were not mentioned in this article), as you move on your journey to production.

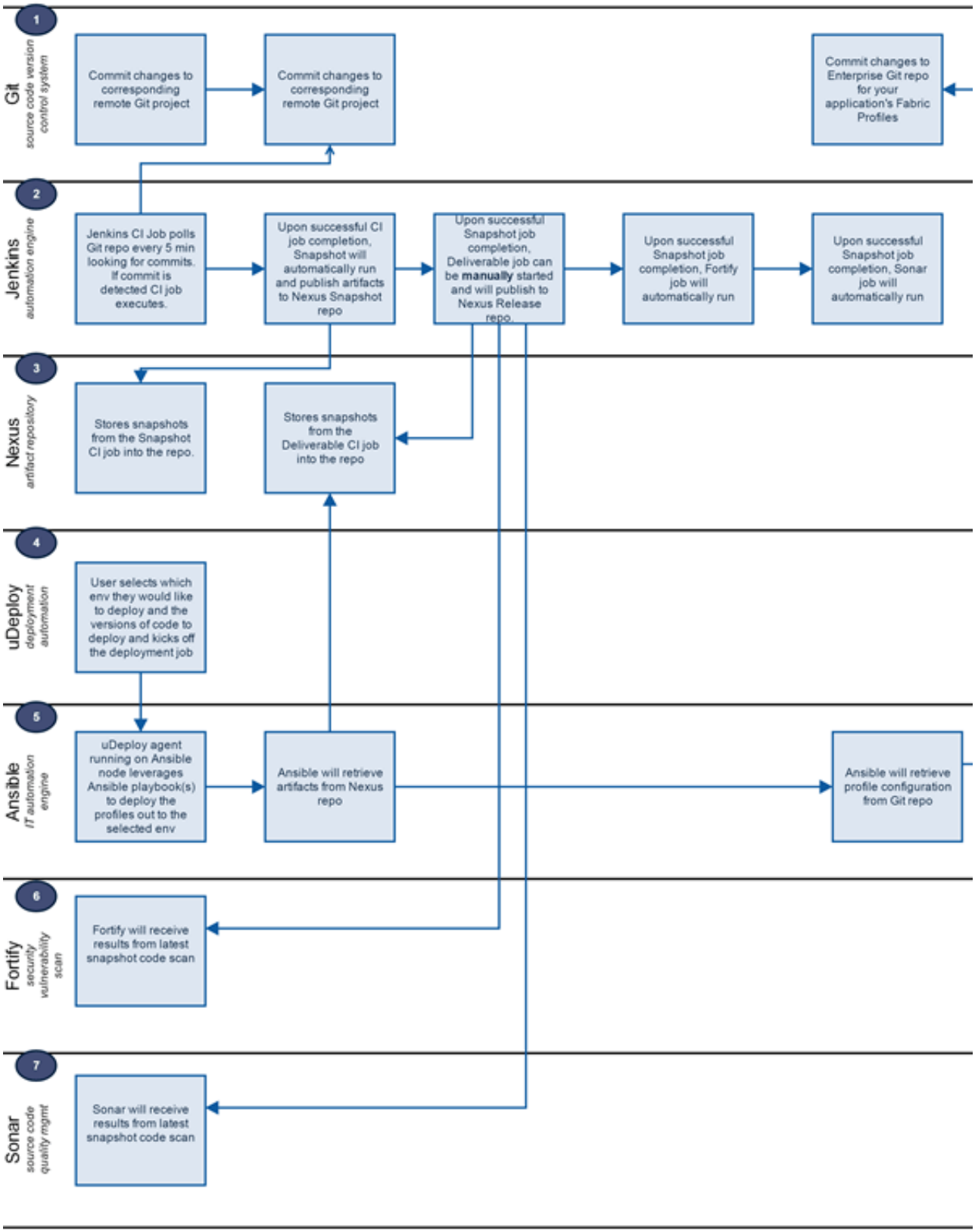

Figure 2 – Continuous Integration Process Flow

PDF generated by Kalin's PDF Creation Station