Continuous Integration & Delivery

Iterate software delivery in a continuous cycle

William Chan
Lead Platform Engineer, 605.tv, Capital One, FreeWheel (Comcast)

Objectives

- Learn what the different types of testing are and what they test
- Learn what tests are helpful when time is a constraint
- Learn how to unit test properly with mocks and stubs

What is Continuous Integration?

Software practice of integrating the software components from other teams often to detect integration errors earlier on.

Difference between Continuous Integration, Delivery & Deployment

Continuous delivery is continuous integration but goes a step further to enable on-demand deployments by pre-building artifacts

Continuous deployment is continuous integration & delivery with automatically deploying the artifacts if all build stages passes

Continuous Integration, Delivery & Deployment Workflows

1. Implement software features and add tests
2. Deploy software to integration environment
3. Run static code analysis on the software
4. Run tests on the software in the integration environment
5. Build the artifacts and put it in a centralized location
6. Deploy the artifacts to different environments from the centralized location

Why Continuous Integration, Delivery & Deployment?

It helps software teams to collaborate to develop cohesive software.

- Levels the amount of effort required for integration
- Shorten feedback loops to help build more cohesive software and improve flexibility on feature development
- Improve visibility and transparency to the build process
- Software features can be delivered more rapidly and often

Tangible Software Engineering

- Source code - code that is written by software engineers
- Artifacts - compiled binaries or artifacts produced from the source code that runs on operating systems

Operating Software

Software tend to run in different environments to help ensure the quality of the software before releasing to clients or users.

QA environments are for running the application in an environment that is similar to production usually integrating with other component dependencies so that product owners can verify the software functionality before releasing the software to production for clients.

Production environments are where the software is available for clients and users to use

Continuous Integration, Delivery & Deployment Implementation

Continuous Integration, Delivery & Deployment Implementation Overview

- Implement tests to guarantee quality, providing confidence that the code works
- Implement static code analysis to ensure that the memory doesn't leak memory, vulnerabilities in software, and code complexity is minimized
- Centralized repository to build software off of eg. Git master branch
- Different environments with a degree of consistency with one another to verify that the software runs as expected before releasing it to production
- Centralized artifact repository for which to store a single deployable to different environments

Strategies

- Branching strategy for getting it in a single location to deploy in the centralized repository
- Rollback strategy if the organization needs to deploy a previously working version quickly

Continuous Integration & Delivery Tools

Source Control - software to store and version software changes
- Git
- Subversion (SVN)
- Mercurial

Automation Software - software that automates your build pipelines
- Jenkins
- Bamboo
- CircleCI
- TravisCI
- Weave

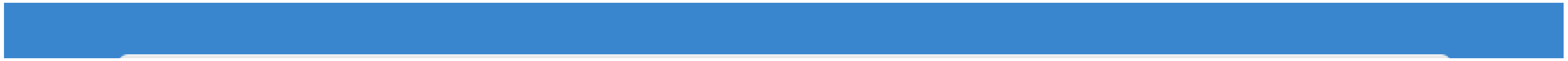
Artifact Repositories - software that stores built packages
- Artifactory
- Nexus
- Apache Archiva

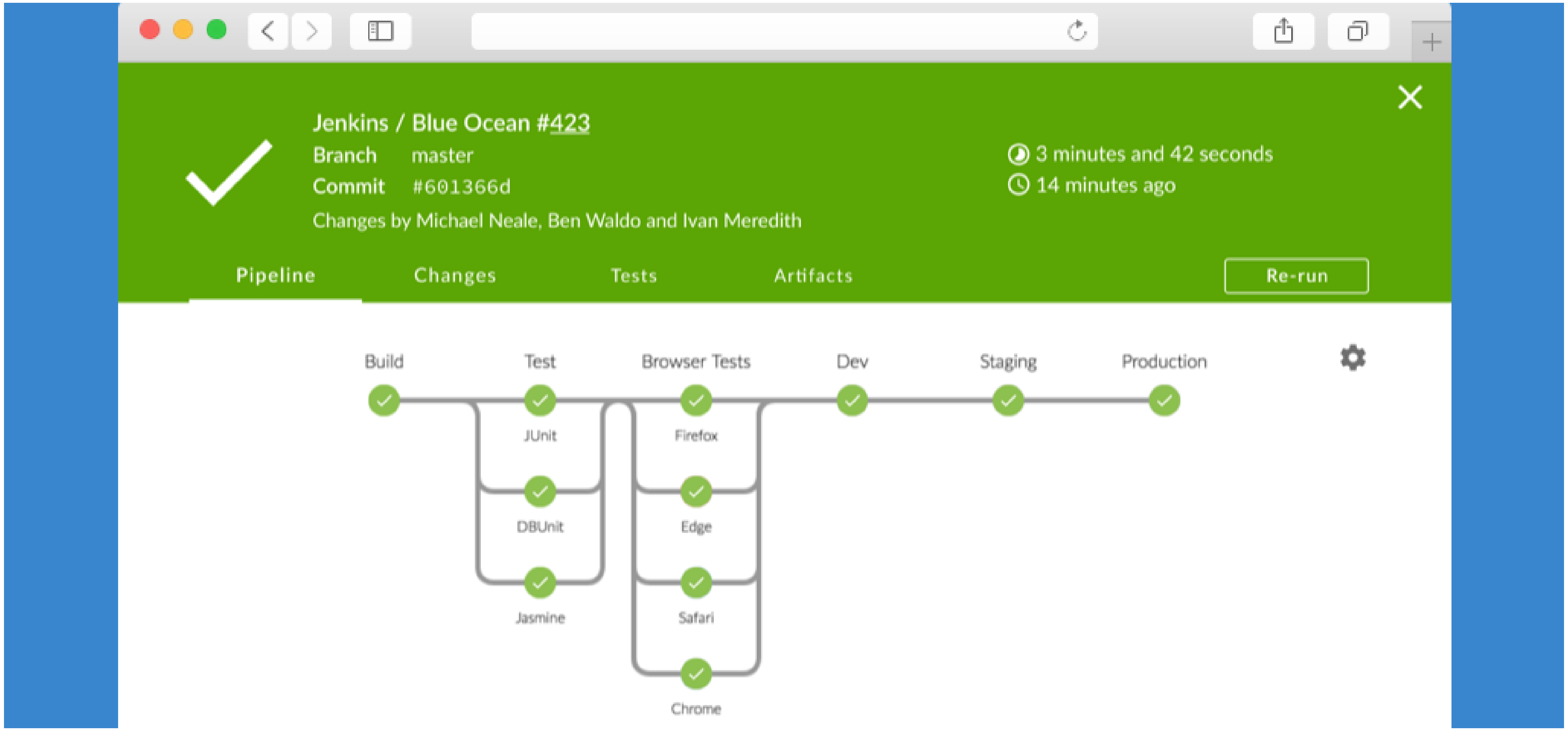
Other Helpful Tools

- Vagrant - configuration management for a local virtual machine

- Vagrant - configuration management for a local virtual machine
- Docker - containerizes your application so that it works agnostic to the environment it runs in
- Chef, Puppet, SaltStack, Ansible - server configuration management to ensure that all servers are consistent to avoid troubleshooting problems that arise from different versions of software
- CloudFormation, Terraform - configurations for building cloud infrastructure stacks

Sample Jenkins Pipeline





Source: https://jenkins.io/blog/2016/05/26/introducing-blue-ocean/ 12

TravisCI Sample Tutorial

`.travis.yml` file inside of Github

```
language: node_js
```

```
sudo: false
node_js:
- 10
- 9
- 8
- 7
- 6
install:
- cd istanbul-mocha && npm install && cd ..
script:
- cd istanbul-mocha && npm test
```

Adding to Github: https://github.com/marketplace/travis-ci
Source: https://raw.githubusercontent.com/codecov/example-node/master/.travis.yml

Summary

- Continuous integration, delivery and deployment lends a hand to the agile ecosystem to allow continuous feedback loops to be closed quickly
- Continuous integration, delivery and deployment removes barriers of effort to

coordinate efforts across teams

- Continuous integration, delivery and deployment requires tangible software items to be store in a centralized location, automation to create consistent environments across infrastructure and finally automation to deploy the tangible items into such environments

Thank you

William Chan
Lead Platform Engineer, 605.tv, Capital One, FreeWheel (Comcast)
http://linkedin.com/in/wchan2 (http://linkedin.com/in/wchan2)