

InterConnect  
2017

March 19–23  
MGM Grand &  
Mandalay Bay  
Las Vegas, NV



# Hands-on Lab Session 3478

Creating Open Toolchains for IBM Bluemix  
Jim Palistrant, Bluemix Skills and Learning  
Development  
John Liu, DevOps Solution Architect

© Copyright IBM Corporation 2017

IBM, the IBM logo and ibm.com are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

This document is current as of the initial date of publication and may be changed by IBM at any time.

The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. This information is based on current IBM product plans and strategy, which are subject to change by IBM without notice. Product release dates and/or capabilities referenced in these materials may change at any time at IBM’s sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.



# Bluemix DevOps Toolchain Lab

## Objective

This series of labs shows how to set up a productive Continuous Delivery toolchain with a sample that consists of three microservices. After you finish this part of the series, you will be familiar with a toolchain that demonstrates practices from the IBM® Bluemix® Garage Method. **Note:** Toolchains are currently available in the US South region only and the instructions in this lab are written for the US South region.

To create this toolchain, we will use a sample to create an online store that consists of three microservices: a Catalog API, an Orders API, and a UI that calls both of the APIs. The toolchain is preconfigured for continuous delivery, source control, blue-green deployment, functional testing, issue tracking, online editing, and alert notification. We will explore the various integrations.

## Online Store sample

The online store consists of three microservices:

1. Catalog API: A back-end RESTful API that tracks all of the items in the store.
2. Orders API: A back-end RESTful API that tracks all store orders.
3. UI: A simple UI that displays all of the items in the store catalog, and that can create orders. This PHP UI calls both of the REST APIs.

The Catalog and Orders API are backed by a Cloudant store. As part of deploying this application a no cost Cloudant service instance will be created.

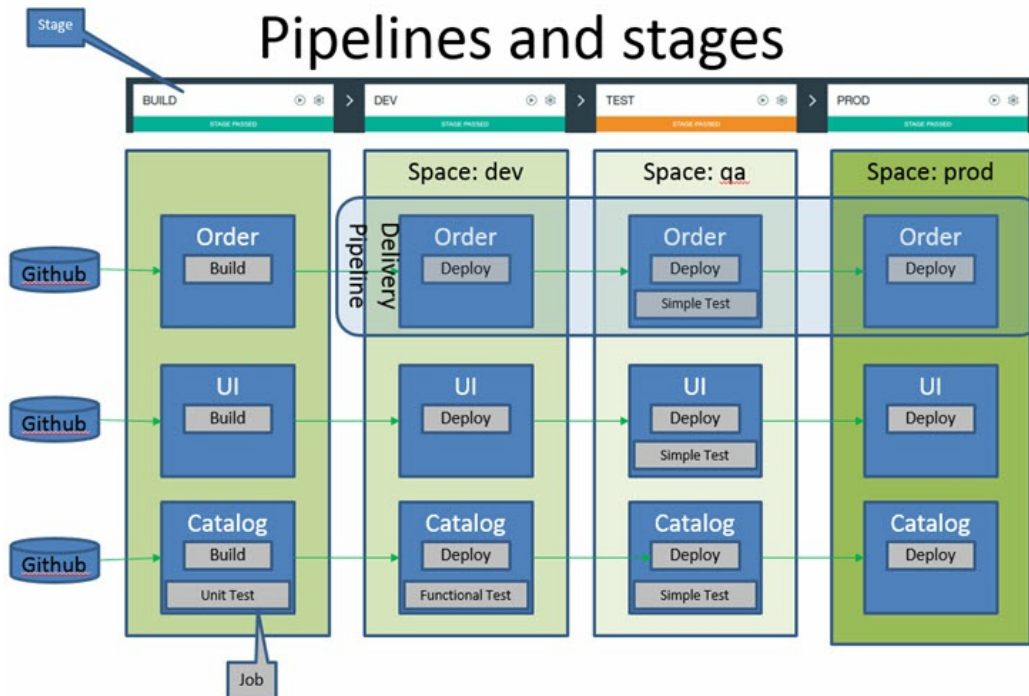
## Pipelines, stages and deployment environments - oh my!

In the real world, many enterprises have a process for developing, testing and deploy code to production. The lab scenario we will go through shows how Bluemix Continuous Delivery toolchains can be used to to automate that process.



- The code for the online store is already in three different GitHub repositories, one per microservice. As part of creating the Continuous Delivery toolchain, you will clone the repositories to your own GitHub account.
- Three delivery pipelines will also be created, again one per microservice. Each pipeline can be run in parallel.
- Each delivery pipeline will consists of a number of stages (Build, Dev, Test and Prod).
- Each stage will consist of one or more jobs that perform a task such a build the code, deploy the code, or testing the codes.
- As part of the respoective delivery pipeline, each microservice is deployed to three environments: development, test, and production. You end up with nine deployed apps.
- While we could edit locally (using Eclipse for example) and push the code up to Bluemix, we will instead use the Eclipse Orion Web IDE, which you can use to edit your code and deploy it with the pipeline from a web browser.
- We want our application to scale as needed so we will be deploying one of the microservices into Bluemix containers

Conceptually, the process looks like:



## Teaming

Software development is a team activity. The lab scenario also shows how Bluemix Continuous Delivery tool integrations can be used to alert teams when activities occur (such as builds or deployments) as well as when events happen (such as a build failing or an application outage).

- Slack is configured to alert the team when activities occur
- [PagerDuty | IBM Alert Notification] is configured to alert the team when events happen
- Bluemix Availability Monitoring is configured to monitor the application in production and alert the team when outages occur

It sounds like a lot ... and it is! Thankfully, it is all handled by a Bluemix Continuous Delivery toolchain. And even better, we will use an existing template to give a great starting point.



## Prerequisites

Prior to running these labs, you must have a Bluemix account, a GitHub account and access to a lab laptop. Follow the steps in Lab 0 to create one or both of those accounts.

## Labs

- [Lab 0: Create Bluemix and GitHub accounts](#)
- [Lab 1: Create Toolchain for Sample Application](#)
- [Lab 2: Build and deploy to dev space](#)
- [Lab 3: Customize Toolchain to add Slack Integration](#)
- [Lab 4: Customize Toolchain to allow full deployment](#)
- [Lab 5: Customize Toolchain to add Bluemix Availability Monitoring](#)
- [Lab 6: Add Bluemix IBM Alert Notification](#)
- [Lab 7: Modify Pipeline for Catalog to deploy Catalog to Containers](#)
- [Lab 8: Add auto-scaling support to Catalog](#)