



InterConnect 2016

The Premier Cloud & Mobile Conference

Session: Lab 4084

Create Your Own Continuous Build, Integration and Delivery Pipeline in the Cloud

Lab Instructions

Authors:
Eric Samuelsson, IBM
Mike Melick, IBM
Chris Brealey, IBM

February 21 – 25
MGM Grand & Mandalay Bay
Las Vegas, Nevada

February 2016 edition

NOTICES

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

TRADEMARKS

IBM, the IBM logo, and ibm.com are trademarks or registered 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.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

IT Infrastructure Library is a Registered Trade Mark of AXELOS Limited.

ITIL is a Registered Trade Mark of AXELOS Limited.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

© Copyright International Business Machines Corporation 2015.

This document may not be reproduced in whole or in part without the prior written permission of IBM.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Introduction	3
Application Overview.....	4
Application 1 – Catalog API.....	5
Application 2 – Orders API.....	5
Application 3 - UI.....	5
Getting Started.....	6
Sign up for Bluemix and DevOps Services.....	6
Starter Project	6
Deploy.....	7
First Deployment – Catalog (D2BM).....	7
Second Deployment – Orders (D2BM).....	9
Third Deployment – UI (Manual Deployment)	10
Edit Code.....	12
Deploy Code via Web IDE	12
Web IDE.....	12
Build And Deploy	14
Auto Deploy to Development.....	14
Add a Build Stage:	14
Add a Deploy Stage:.....	15
Commit a change:	15
Manually Deploy to Production	16
Create a production space:.....	16
Create a production deployment stage:.....	17
Manage Work	19
Tracking & Planning.....	19
Add track and plan to the Catalog API project:	19
Create a work item in your backlog:	19
Plan your work:.....	20
Track your work:	20
Test.....	21
Simple Unit Test.....	21
Add unit tests to the build:	21
Try the unit tests:	22
Fix the bug in the tests:.....	23
Complete the story:	23
UI Test.....	24
Add UI tests to the deploy stage:	24
Run the UI tests:.....	25
Update the UI tests:	26
Going Further (Optional Exercises).....	27

Bluemix Garage Method 27
Tool Integrations (Beta)..... 27
Resource guide 28

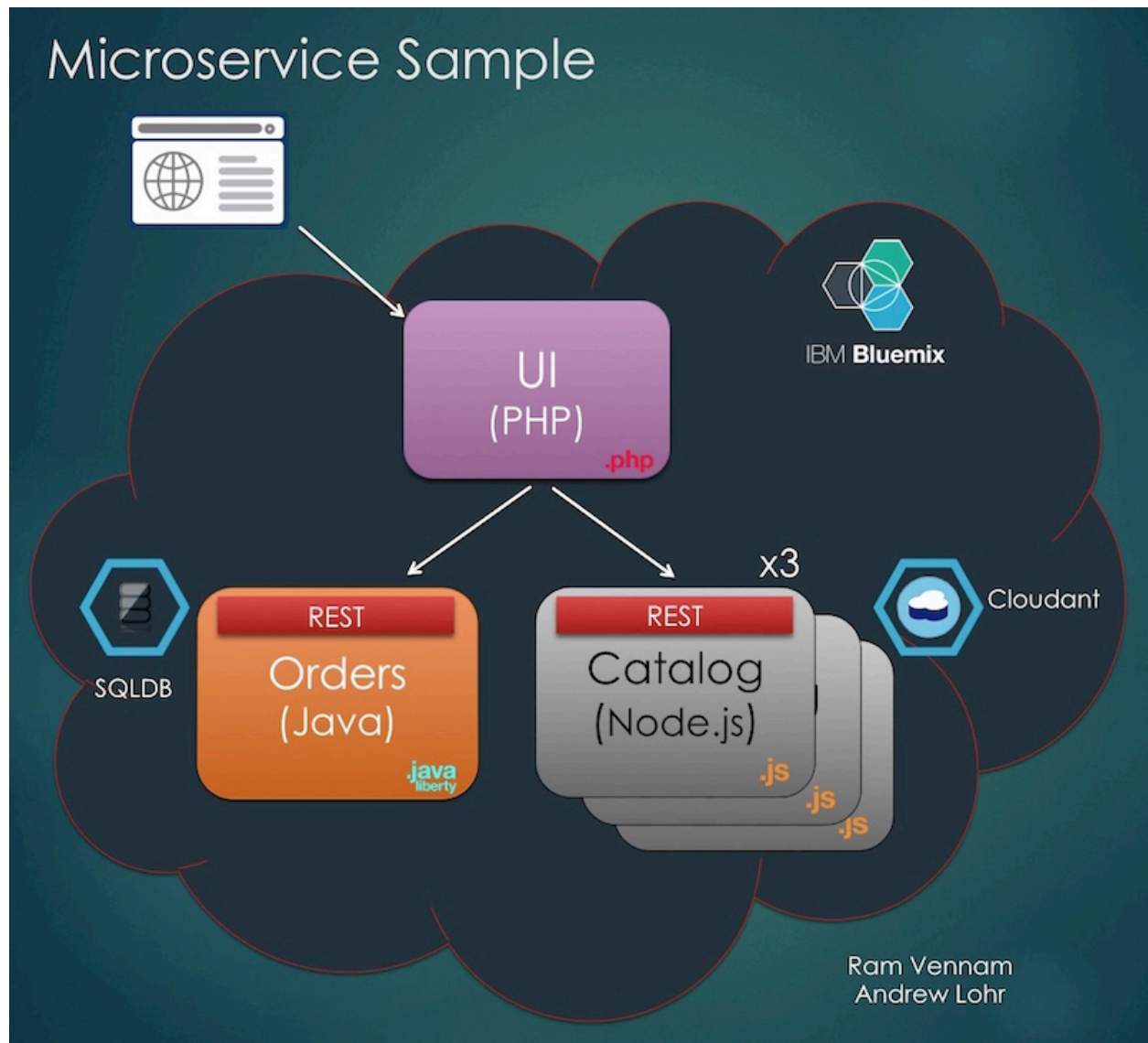
Introduction

Microservices have been gaining increasing traction as a way to deliver applications, especially in a cloud based environment. They offer the ability to break down a complex system into smaller independent pieces that can be developed and deployed on their own, typically without impacting the overall stability of the application and in a continuous fashion. They can be written in almost any language and can be developed/maintained by a single team with deep knowledge of the whole service.

To be successful with microservices you need the right people, culture, processes and tools. You need an environment where continuous integration, continuous deployment and DevOps are practiced.

This lab will use a simple microservice based application to highlight how tools in a DevOps toolchain can help you effectively develop, test and deploy your microservices in a quick, repeatable and automated fashion. Such a toolchain will provide you with the high degree of confidence necessary to allow frequent deployments of your changes to your production environments.

Application Overview



In this lab, you'll be working with a very simple e-commerce application composed of three microservices. One of the major strengths of microservices is the ability to choose which language/runtime you want to use for each application allowing you to pick the language/runtime best suited to the function of each microservice. Each of the microservices in this lab is written in a different language, JavaScript (Node.js), Java, and PHP, and the DevOps tools will be used to bring the three microservices together in Bluemix to run as one logical application.

Application 1 – Catalog API

The catalog API is implemented in Node.js and uses a Cloudant database. Node.js is a great fit for the Catalog API due to its' non-blocking, event driven I/O and it's ability to handle a high number of requests. Node.js also has an extensive set of modules available to help speed development and streamline the integration with NoSQL databases such as Cloudant.

Application 2 – Orders API

The orders API microservice is a more transactional system in nature. Java (JAX-RS) and a relational SQL database were chosen for the Orders API as they are well suited to this type of application.

Application 3 - UI

The UI for the application was written in PHP as it allows for the easy creation of User Interfaces and quick access to web resources.

Getting Started

Sign up for Bluemix and DevOps Services

Before you can get started, you'll require a valid ID for Bluemix and DevOps Services. Don't worry you can try them both for free! If you don't have an existing ID, sign up for one by following the steps below:

1. Navigate to <https://console.ng.bluemix.net/registration/> and sign up.
2. As part of the Bluemix registration process, you will receive an e-mail asking you to confirm your account. If you do not confirm, you are not registered. If you do not receive a confirmation e-mail, send a note to id@bluemix.net.
3. Navigate to <https://hub.jazz.net> and click **LOG IN** in the top right corner.
4. After being automatically signed into DevOps Services you will be asked for an Alias. This will be the username for your account.

Starter Project

To make things a little bit easier (especially for those using a paper copy of the lab instructions) there is a starter project on Github containing links to all the materials you'll need to use during this lab. There's even a copy of these lab instructions posted there if you prefer to use an electronic copy.

The starter project is located at: <https://github.com/melickm/Lab4084>

Deploy

First Deployment – Catalog (D2BM)

Now that you've got IDs for Bluemix and DevOps Services, you can start to deploy your microservices. For the Catalog API you'll be using the **Deploy to Bluemix** button. Clicking on the deploy button will create a DevOps project, clone a GitHub repository, create a delivery pipeline, deploy and run your application in Bluemix. All in a single click! That one button click sure does a lot of work for you.


1. Click on the **Deploy to Bluemix** button below, or use the first one in the starter project at: <https://github.com/melickm/Lab4084>



2. If you aren't already logged in, click on the **LOG IN** button.
3. All the fields on the page will be pre filled for you. Select **Deploy**

Deploy this application to Bluemix

Deploying this app will create a private DevOps Services project for you. [Learn more.](#)



MICROSERVICES-CATALOGAPI

GIT URL: https://github.com/melickm/Microservices_CatalogAPI
GIT BRANCH: master

APP NAME

Microservices-CatalogAPI-melickm-1810

REGION

ORGANIZATION

SPACE

IBM Bluemix US So

MikeOrg

Lab4084-dev

DEPLOY

You are logged in as [melickm@ca.ibm.com](#). [Log out.](#)
[Terms of Use](#)

- Once the application is done deploying to Bluemix, select **View your App**. Note: you will need the route for the UI application, so we suggest you keep the tab with your app running open.
- You can run a quick test to see if the app is working by following the instructions on the homepage. Note: the format will be in JSON which will be consumed by the UI application once it's running.
 - For example: <http://microservices-catalogapi-melickm-1810.mybluemix.net/items> returns a web page that starts with the JSON below

```
{
  "total_rows": 8,
  "offset": 0,
  "rows": [
    {
      "id": "010bf87eea2853a14d13c8aa2d52305b",
      "key": "010bf87eea2853a14d13c8aa2d52305b",
      "value": {
        "rev": "1-ee1268c3b44a8329c60af2d28b932903"
      }
    }
  ]
}
```

```
    },
    "doc": {
      "_id": "010bf87eea2853a14d13c8aa2d52305b",
      "_rev": "1-ee1268c3b44a8329c60af2d28b932903",
      "name": "Ping pong balls",
      "color": "white",
      "quantity": 97,
      "description": "3 star ping pong balls, regulation size.",
      "usaDollarPrice": 12,
      "imgsrc":
"http://upload.wikimedia.org/wikipedia/commons/f/fd/Tischtennisball-
weiss-004.jpg"
    }
  },
  ...
}
```

Second Deployment – Orders (D2BM)

Once again, you'll use the **Deploy to Bluemix** button to get your second microservice running. Don't worry, it'll be as easy as the first!


1. Click on the **Deploy to Bluemix** button below, or use the second one in the starter project at: <https://github.com/melickm/Lab4084>



2. All the fields on the page will be pre filled for you. Select **Deploy**

Deploy this application to Bluemix

Deploying this app will create a private DevOps Services project for you. [Learn more.](#)




MICROSERVICES-ORDERSAPI

GIT URL: https://github.com/melickm/Microservices_OrdersAPI
GIT BRANCH: master


APP NAME

Microservices-OrdersAPI-melickm-1826


REGION

IBM Bluemix US So 

ORGANIZATION

MikeOrg 

SPACE

Lab4084-dev 

DEPLOY

You are logged in as [melickm@ca.ibm.com](#). [Log out.](#)
[Terms of Use](#)

- Once the application is done deploying to Bluemix, select **View your App**.

Note: As the steps in the deploy to Bluemix UI complete, they become links to the project, repository, build & deploy pipeline, and deployed application running in Bluemix.

Third Deployment – UI (Manual Deployment)

Now that you've got the first two microservices running using the **Deploy to Bluemix** button, you'll do things a little bit differently for the third. This time you'll fork an existing project in DevOps Services to pull the code into a DevOps Services Git repository.

- Navigate to the UI project's overview page by clicking on the link below or by using the link in the starter project at: <https://github.com/melickm/Lab4084>

https://hub.jazz.net/project/melickm/Microservices_UI/overview

2. Click **Fork Project** on the right side of the page
3. Enter your project's name, **Microservices_UI**. Make sure that the boxes **Add features for Scrum development** and **Make this a Bluemix Project** are checked.
4. For the region confirm **IBM Bluemix US South** is selected.

Fork Project

Name your project: melickm | Microservices_UI-199 ✓

URL: https://hub.jazz.net/project/melickm/Microservices_UI-199

☐ Private project (Invited team members only)

☐ Restrict membership (IBM only)

You can restrict this project's membership because your email address ends with ibm.com. If this project is for **IBM confidential** business, you must select this option and agree to certain conditions. [Learn more](#)

☐ I accept the terms and conditions

☒ Add features for Scrum development ⓘ

☒ Make this a Bluemix Project ⓘ

Select a Bluemix space to bill your services to:

Region	IBM Bluemix US South
Organization	MikeOrg
Space	Lab4084-dev

These selections can be changed later in the options for your Project

CANCEL **CREATE**

5. Click **CREATE**

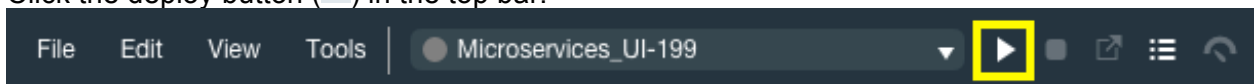
Now you have a copy of the Microservices UI code in your own project. You can edit it and run it as you see fit. And.... in the next sections, you'll do just that.

Edit Code

Deploy Code via Web IDE

To deploy an application means to bundle the project artifacts, create an app at Bluemix, transfer the bundled app to Bluemix, and start the app. Bluemix application names and URLs are created using properties, which are often defined in the `manifest.yml` file in a project. The "Microservices UI" project already has the manifest created for you.

1. Click **Edit Code** in the upper-right corner of your new project to open the Web IDE.
2. In the left navigation pane of the Web IDE, select **manifest.yml**. This file will allow you to control how your application gets pushed to Bluemix. For example you can change the name of the app that appears in Bluemix by editing your **manifest.yml**.
3. Click the deploy button (▶) in the top bar:





Note: When you deploy through the Web IDE, you are deploying the changes that are in your Web IDE's workspace. For example, you may have changes you are currently working on that you have not yet pushed to your project's repository. Those changes will be deployed.

4. Once the application is done deploying, you can click on the Open URL button (🔗) button to see it live. As both of the API services were previously deployed, the entire application should now be running, and you should see the simple catalog provided by the application.

Web IDE

While deploying from here was pretty easy, there is a lot more that the Web IDE has to offer. When you clicked on Edit Code, you were brought to a fully functional IDE (Integrated Development Environment) running within your browser. Content assist, code completion, syntax highlighting, error checking, built in source control and support of nearly any language. All of this available without leaving your browser!

For the purposes of the lab, let's make a simple change to the UI to customize the application.

1. In the left pane of the Web IDE, select **index.php**.
2. Locate the row (line 3) `$title = "Microservices Sample";`
3. Replace `Microservices Sample` with `your name Microservices Application`
4. Click File> Save.
5. Click the Deploy button ().
6. If prompted to stop and redeploy your application, click OK.
7. After your application is deployed, click on the Open URL button () in the top gray bar.

Notice the change of the title at the top of the application. It's good that you are finally getting some recognition for all that hard work!

Although this was a pretty simple change to your application, you were able to make the change and deploy a running instance of it with only a few button clicks in your browser.

Build And Deploy

So far you've got your two backend projects building and deploying to a development environment thanks to the **Deploy to Bluemix** buttons you used earlier, but the UI has only been deployed from the Web IDE. Although that's a great way to develop and test your code iteratively, you probably don't want to be responsible for manually pushing the code from your IDE every time you or your teammate makes a change. This section of the lab will help you set up a deployment pipelines to do this for you automatically.

Auto Deploy to Development

Add a Build Stage:

1. Click on the **Build & Deploy** button to navigate to the build and deploy page for the UI project.
2. Create a Build stage by clicking on the **+ ADD STAGE** button.
3. Change the name from **MyStage** to **Build**
4. Ensure the following defaults are selected:
 - a. **Input Type = SCM Repository**
 - b. **Git URL** points to your UI repository
 - c. **Branch = master**
 - d. As you'd like the build to be automatically triggered, ensure the **Run jobs whenever a change is pushed to Git** radio button is selected.
5. Next click on the **JOBS** tab to go to the jobs page where you'll add a build job
6. Click on **ADD JOB** and select **Build**.
7. Click on the **Builder Type** to see the different types of builds that can be run. For this demo, a **Simple** build will be enough.

8. Click **SAVE**.

Now every time a change is pushed to the Master branch in your Git repository, the build stage will be triggered to grab the change and make the build available to later stages in the pipeline.


Add a Deploy Stage:

1. Add a Deploy stage by clicking on the **+ ADD STAGE** button.
2. Change the name from **MyStage** to **Deploy**.
3. The input settings should be defaulted to pick up the build from the **Build** stage you set-up previously. Confirm:
 - a. **Input Type** is **Build Artifacts**
 - b. **Stage** is **Build**
 - c. **Job** is **Build**.
4. As this build should be auto deployed to the dev environment, leave the **State Trigger** to **Run jobs when the previous stage is complete**.
5. Next, click on the **JOBS** tab to go to the jobs page where you'll add a Deploy job.
6. Click on **ADD JOB** and select **Deploy**.
7. By default the job will be configured as a Cloud Foundry deployment targeting Bluemix's US South pod, under your current organization and the dev space. You shouldn't need to change any of the values from the default provided.
8. Click **SAVE** on this stage.

You've now got a simple pipeline that will automatically take a change committed to your Git repository and deploy it as a running app in Bluemix. No more manual pushes necessary!

Commit a change:

Now that you've got your pipeline all setup, let's go back and commit the change to customize your application.

1. Click the **EDIT CODE** button to go back to the Web IDE where you previously changed the title of the application.
2. On the left hand navigation click on the Git repository button ()
3. In the box in the **Working Directory Changes** section, type a comment similar to the one below indicating what you've changed: `Customized the title`
4. Select each of the changes in the Changed Files section and click the **Commit** button.
5. In the **Outgoing** section, click **Push** to push your changes to the master branch.
6. Click on the **BUILD & DEPLOY** button to get back to your pipeline and watch your committed change be deployed to you dev space.

Your changes have now been committed to your repository and are available to others. Additionally the pipeline automatically picked up these changes and deployed them to Bluemix.

Click on the link in the **LAST EXECUTION RESULT** section of the Deploy stage to see the application running with your customizations

Manually Deploy to Production

As shown in the previous section, you can configure a multi-stage build and deployment pipeline to support your DevOps approach to software development. In this section, you will extend the pipeline for the Catalog API application to include a deployment to your production (prod) space.


Note: You'll only deploy the Catalog API application during this lab. In order for the application to be completely deployed to a production environment, you would have to perform a similar addition of a production deployment step in the other two micro services.

Create a production space:

1. Navigate to the Bluemix dashboard if you are not there already by clicking **DASHBOARD** in the top menu.
2. Expand the **ORG:** dropdown menu if it isn't already.
3. Click **Create a Space** in the left menu.
4. In the Create a Space dialog, input `prod` as your space name and click **CREATE**.

You've now got a distinct space for your application to deploy into. Using a separate space for your production (prod) instance will allow you to create two distinct copies of your application so that you can work with them independently.

Create a production deployment stage:

1. Navigate back to the Catalog API pipeline. Hint: If you don't have a link handy, you can go to <https://hub.jazz.net>, find the project in the list of **My Projects** and click on the pipeline icon ().
2. You could create the production stage from scratch as you did for the development stage, but since they are similar let's clone the **Dev Deploy** stage instead. Click on the configuration icon for the **Dev Deploy** stage and select **Clone Stage**.
3. Change the name of the stage to `Prod Deploy Stage`.
4. Production uses the same builds as the development environment, but they are selectively pushed there after the quality has been confirmed.
 - a. Leave the **Input Type**, **Stage** and **Job** to reflect the builds coming from the Build stage.
 - b. Change the **Stage Trigger** to **Run jobs only when this stage is run manually** to allow for selective deployment to production.
5. Switch to the **JOBS** tab to configure the deployment job.
6. As the deploy job was copied from the dev deployment, it needs to be updated for the production deployment.
7. Change the space to `prod`.
8. Ensure the application name is unique within your Bluemix prod space
9. By default, the deployer will use the instructions in the manifest. Rather than deploying to a url with a random string of characters, we want this to be a consistent place where your end users will be able to access your app. In the Script section, after

```
cf push "${CF_APP}"
```

```
add -n whateveryouwantyoururltobe.
```

Note: the hostname needs to be unique across all Bluemix apps, so be creative.

For example: `cf push "${CF_APP_NAME}" -n whateveryouwantyoururltobe`

10. Click **SAVE**.

11. Assuming you're happy with the build you deployed to development, click on the play button (▶) on the **Prod Deploy Stage** to deploy the build to production. Note: you can also drag and drop a build from a previous stage to deploy it.

When the stage finishes, you'll have a second instance of your Catalog API microservice running within the newly created **prod** space. This will allow you to do frequent changes to your application and have them automatically deployed to your development environment. When you're comfortable with the quality of those changes you can deploy them out to your production environment for your users.


Manage Work

Tracking & Planning

As teams grow, and projects become larger and more complex, the need for transparency about the work being done, and planning to ensure the important work is being done first become increasingly important. No longer can you keep the entire plan on a scrap of paper on your desk!

In this section you'll use the Track and plan functionality to create a very simple backlog of improvements and show how changes in the source can be linked to the work items they support.

Add track and plan to the Catalog API project:

1. Open the **Catalog API project** you created previously by navigating to DevOps Services, signing in if you're not already authenticated and click on your project on the **My Projects** page.
2. Click the gear button () in the upper right corner to open the Project Settings page.
3. Click **OPTIONS**.
4. On the page that loads, select the **Enable Track & Plan** checkbox (if it is not already enabled) and click **SAVE**.
5. Click **TRACK & PLAN** in the upper right corner. The Track & Plan section is your place to organize your ideas.


Create a work item in your backlog:

1. On the left pane, click **Backlog**.
2. On the right pane, type the following in the **Create a work item...** field: `As a developer, I want to add automated unit tests in my pipeline to prevent the deployment of poor quality code.`
3. In the icon bar below, click the **Owned By** icon and select your username.


4. The tool will automatically detect the work item type is story and set the type for you.
5. Click **Create**.
6. Repeat steps 2-5 to create a second work item with the following: As a developer, I want to add automated UI tests in my pipeline to prevent the deployment of poor quality code.

Plan your work:

Even though there are only two work items, let's setup a rank-ordered backlog to ensure the most important work items are getting done first. The rank-ordered backlog is a natural fit for a continuous integration and deployment configuration like the one being used in this lab. It provides developers with a clear picture of the next most important work item, and allows leads and managers to easily establish this priority across all work items. As items are completed, they can be dropped into the pipeline and deployed, tested along the way, and even deployed directly to production.

1. To begin with the work items will be unranked. Drag the "...automated UI tests..." work item to the top of the list to give it a ranking. Alternatively you can click on the **RANK** icon () and assign the number 1.
2. Next, drag the "...automated unit tests" item above the "...automated UI tests..." work item to make it the #1 ranked work item. The unit tests for the Catalog API have been deemed to be a higher priority than the UI tests.

Track your work:

1. Still in the backlog view, click on the **Status: New** icon () for the "...automated unit tests..." story.
2. In the Status dialog, click **Start Working**. This will change the status to **In Progress** and let your teammates know you've started working on the item.

Now that you've accepted the work you'll add the unit tests in the next section.

Test

Simple Unit Test

Unit tests are a first line of defense against bugs. A good set of unit tests will let you refactor fearlessly, squash bugs without introducing new ones, and produce better code. Adding these unit tests to your build pipeline and automatically running them against every commit can provide you the confidence necessary to allow your build to be deployed to a development, test or even a production environment.

In this exercise, you'll add a set of existing mocha tests into the build pipeline and trigger them to run on each commit.

Add unit tests to the build:

1. Click on the **Build & Deploy** button to get to the Catalog API's build & deploy pipeline.
2. The unit tests will run right after the build, and will be used to prevent a bad build from being deployed to the Dev stage. Click on the configure icon (⚙️) for the Build Stage and select **Configure Stage**.
3. On the **JOBS** tab click **ADD JOB** and select **Test**.
4. Select **Simple** for the **Tester Type**, as the mocha tests will be run using a script.
5. In the **Test Command** box insert the snippet below:

```
#!/bin/bash
npm install
XUNIT_FILE=tests/TEST-items.xml node_modules/mocha/bin/mocha -R spec-
xunit-file tests/server/items.spec.js
```
6. The above script will run the mocha tests in the items.spec.js and create an xunit formatted result file named TEST-items.xml.
7. Ensure **Enable Test Report** is checked and the **Test Result File Pattern** contains a pattern that will match the result file from the tests. For example, the default pattern of:
tests/TEST-*.xml.

8. The unit tests are a gating factor to deploying the build. If the unit tests don't run green, the Build Stage should be stopped so the build will not be deployed. Ensure **Stop running this stage if this job fails** is checked.
9. Click **SAVE**.

Try the unit tests:

1. Now that unit tests have been added to the Catalog API, let's run them.
2. On the **Build Stage** click the play button (▶) to start the stage.
3. Click on the **Test** job to see the unit tests run.
4. If you get there quick enough, you'll see the execution of the tests as they progress. If not, the results will be available at the bottom of the Logs.
5. Once the job has finished, the **Test** job will be shown as failed. As the job was setup to stop the stage upon failure, the build was not passed onto the Deploy Stage. The failed tests have prevented the potentially bad build from reaching your development environment and more importantly your customers.
6. To figure out why the tests failed, click on the **Test** job in the **Build** stage if you aren't there already.
7. In the **LOGS** tab, go to the bottom of the job output. Note: you can click the scroll to the bottom icon (⏮) to quickly get to the bottom of the log.
8. Looking at the bottom of the log, you'll see a suspicious test, which is failing.
9. An easier way to determine which test(s) failed is to click on the **TESTS** tab above the log output.
10. On the screen you can see the pass fail rate, and drill down into the results of the tests by clicking on the **Mocha Tests** link. If you click on the details icon (🔍) for the failed test, you'll see the cause of the problem.


At this point you could open a new work item for the defect, but to keep things simple, let's continue to use the "...automated unit tests..." story to fix the bug.

Fix the bug in the tests:

1. Click on the **EDIT CODE** button to bring up the Web IDE.
2. Open the mocha test file `tests/server/items.spec.js` from the File Browser on the left.

3. Remove the faulty test which is located near the bottom of the file.

```
describe('superfluous testing', function() {  
  it('This test looks suspicious', function() {  
    assert.equal(1, -1);  
  });  
});
```

4. **Save** the changes.
5. On the left hand navigation click on the Git repository button ()
6. In the box in the **Working Directory Changes** section, type a comment similar to the following, being sure to replace `xxx` with the work item ID associated with the “...automated unit tests...” story you created: `Story xxx: removed unnecessary and broken test.`
7. Select each change in the **Changed Files** section and click the **COMMIT** button.
8. In the **OUTGOING** section, click **PUSH**.
9. Click on the **BUILD & DEPLOY** button to watch the change go through the pipeline. Now that all the unit tests are passing, the build will be automatically deployed to the development environment.

Complete the story:

Now that you’ve added automated unit tests to the pipeline, you can mark the story as complete so that everyone is aware of this great addition.

1. Click on **TRACK & PLAN** in the upper right corner.
2. In search field at the top left of the UI, type `automated unit tests` to search for your work item.
3. Select your work item from the list.


4. Click on the **LINKS** tab in the work item.
5. Notice there is a change set linked to the work item. This was the change you checked in to fix the broken unit tests.
6. (Optionally) Click on the link to view your changes in Git.
7. Click on the **OVERVIEW** tab to get back to the initial view of the story.
8. In the top right corner, change the status of the Story to **Complete**.
9. **SAVE** the change.

UI Test

Equally as important to testing your backend code is making sure your User Interface is functioning properly and is well behaved in different browsers and/or mobile devices. For testing the UI you'll be using integration with Sauce Labs which will allow you to easily automate functional testing on multiple operating systems and browsers.

For convenience, feel free to use the Sauce Labs account that we've created for you with **UserName** `Lab4084` and the **Access Key** `c61e8e3c-85bb-4dc0-b262-d98768f04af4`. If you'd prefer you can go to <https://saucelabs.com> and sign up for a trial account, or use your own account if you have one.

Add UI tests to the deploy stage:

1. (Optional) Go to the **Track & Plan** page and set the status of the "...automated UI tests..." work item to **In Progress**.
2. Click on the **MY PROJECTS** button.
3. Find your UI project, and click on the Build Pipeline icon () to get to the build & deploy page for the UI application.
4. Click on the gear button () in the upper right corner of the "Dev Deploy" stage.
5. Click on **ADD JOB** and select the **Test** job type
6. Name the job "UI Tests"

7. Select **Sauce Labs** for the **Tester Type**
8. In the **Username** field enter your Sauce Labs user name
9. In the **Access Key** field enter your Sauce Labs access key
10. Select **Custom...** for the **Test Execution Command**
11. In the **Custom Test Execution Command** box insert the snippet below:


```
#!/bin/bash
npm install
grunt test
```

12. Ensure **Enable Test Report** is checked and the **Test Result File Pattern** contains the following pattern:
*.xml

Now that we have added a test job to the stage we need to modify the stage to pass information from the deploy job to the test job:

1. Click the **ENVIRONMENT PROPERTIES** tab in the upper part of the stage configuration dialog
2. Click **ADD PROPERTY** and select **Text Property** for the property type
3. Name the property `CF_APP_NAME` and leave the value empty
4. Click the **JOBS** tab and select the “Deploy” job
5. In the **Deploy Script** box add the following line:
`export CF_APP_NAME="$CF_APP"`
6. The property will now be set during the execution of the deploy job and the value can be used by the test job. The “`CF_APP_NAME`” property is known to the Sauce Labs job type so there is no need to update the test job.
7. Click **SAVE** to save all change made to the stage

Run the UI tests:

1. Click the play button () on the “Dev Deploy” stage
2. Click on the “UI Tests” job and wait for the test to complete


3. Notice how the tests failed and examine the log to find the failing test. The log contains a link to the test execution result in Sauce Labs.
4. Another way to determine which test(s) failed is to click on the **TESTS** tab above the log output and expand the failing test result.
5. Test data, selenium logs and video recordings of the test run is archived and can be downloaded by clicking the **ARTIFACTS** tab.

The reason the tests failed is because you change the title of the UI application home page to include your name. Let's update the UI tests to reflect that change.

Update the UI tests:

1. Click on the **EDIT CODE** button to bring up the Web IDE.
2. Open the test file `tests/sauce/test-cases.js` from the File Browser on the left.
3. Adapt the failing test to use the new title;

```
it("Have correct title", function(done) {  
    browser  
        .get(url)  
        .title()  
        .should.become("your name Microservices Application")  
        .nodeify(done);  
});
```

4. **Save** the changes.
5. On the left hand navigation click on the Git repository button ()
6. Enter a comment in the box in the **Working Directory Changes** section similar to the one below where `xxx` is the number of your work item. `Story xxx: updated the UI tests to reflect customization of the title.`
7. Select the test script file in the **Changed Files** section and click the **COMMIT** button.
8. In the **OUTGOING** section, click **PUSH**.
9. Click on the **BUILD & DEPLOY** button to watch the change go through the pipeline. Verify that the UI tests are passing.

Going Further (Optional Exercises)

Bluemix Garage Method

This lab only touches on few of the ideas, processes and tools that can be used to transform your business; there are many more. The Bluemix Garage Method website shares IBM's own experiences as we undergo a transformation to adopt DevOps practices and create a culture of innovation and agility.

Spend a few minutes at the IBM Bluemix Garage Method website:

<https://www.ibm.com/devops/method> and see what's available. Read an article on micro-services, learn about toolchains and how they can be used with Bluemix DevOps Services, find something that piques your interest or just "bookmark" the website for later.

Tool Integrations (Beta)

If you are interested in seeing how additional tools can be integrated into Bluemix DevOps services, I suggest going through the tutorial below.

In the tutorial, you'll use the Deploy to Bluemix button to create a project with a simple pipeline with integrations to Sauce Labs and Slack preconfigured for you. As the build progresses, you'll receive notifications to a slack channel and test jobs will be run using your Sauce Labs account.

Automated integrations with DevOps Services (Beta) tutorial:

https://www.ibm.com/devops/method/tutorials/tutorial_automated

Resource guide

IBM Technical Client Training has enhanced its training capabilities, and extended reach into new cities and countries, by partnering with five highly qualified IBM Business Partners who provide high quality, authorized training for IBM Clients, IBM Business Partners, and IBM employees. IBM continues to provide authorized training curriculum and content, and also maintains overall ownership of the IBM Training ecosystem.

The delivery of public, private and customized training to IBM Clients and IBM Business Partners is now done by the IBM Global Training Providers (GTPs):

- Arrow
- Avnet
- Global Knowledge
- Ingram Micro
- LearnQuest

See ibm.com/training for information on the classes that the GTPs offer.

Completing this InterConnect lab is a great first step in building your IBM skills. IBM offers several resources to keep your skills on the cutting edge. Resources available to you range from product documentation to support websites and social media websites, including the following examples:

- IBM Training website
 - Bookmark the IBM Training website for easy access to the full listing of IBM training curricula. The website also features training paths to help you select your next course and available certifications.
 - For more information, see <http://www.ibm.com/training>
- IBM Certification
 - Demonstrate your mastery of IBM products to your employer or clients through IBM Professional Certification. Certifications are available for developers, administrators, and business analysts.
 - For more information, see <http://www.ibm.com/certify>



InterConnect 2016

The Premier Cloud & Mobile Conference

February 21 – 25
MGM Grand & Mandalay Bay
Las Vegas, Nevada