Bluemix DevOps Toolchain Lab

Objective

This series of labs shows how to set up a productive 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. Toolchains are available in the US South region only.

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 pre-configured 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.

Each microservice is deployed to up to three environments: development, test, and production. You end up with multiple deployed microservices.

You will be creating a toolchain and adding the following integrations:

- Three GitHub repositories (repos), with GitHub Issues enabled. Each microservice has one GitHub repo.
- Three delivery pipelines, one for each GitHub repo. The pipelines are independent and can run in parallel.
- The Eclipse Orion Web IDE, which you can use to edit your code and deploy it with the pipeline from a web browser.
- Slack, which you can use to get real-time notifications about the status of builds and deployments. comment: # (- [IBM Alert Notification] is configured to alert the team when events happen) comment: # (- Bluemix Availability Monitoring is configured to monitor the application in production and alert the team when outages occur)

Labs

- Lab 0: Getting setup
- Lab 1: Read Template Paper
- Lab 2: Create Order Toolchain Using Pre-Built Template
- Lab 3: Create Catalog Toolchain by Hand
- Lab 4: Create UI Toolchain from deployed application
- Lab 5: Integrate Slack

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM

Lab 0: Getting Setup

Objective

This lab takes you through the various activities to prepare for the rest of the labs.

Prior to running these labs, you must have access to a lab laptop, an active Bluemix account, a GitHub account and Slack access. Your Bluemix account needs to have an appropriate organization and spaces. Follow the steps in Lab 0 to set these up. *Note:* If you need to login to the virtual machine running on the lab laptop, the ID is `localuser` and the password is `passw0rd`.

Tasks:

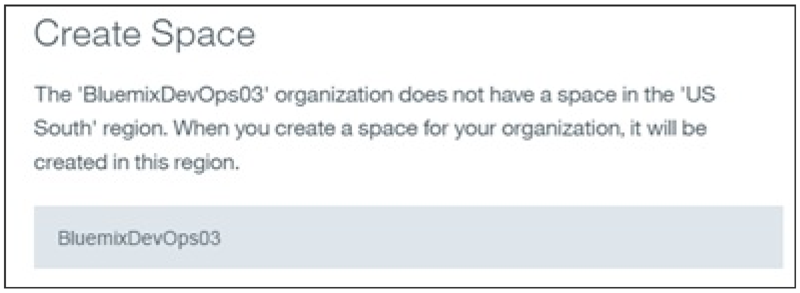
Task 1: Create Bluemix trial account

1. If you already have an active Bluemix account, you can skip this task.
2. Open a web browser and enter the following URL: https://console.ng.bluemix.net/
3. Click on the **Sign Up** button.
4. Follow the directions to fill out the form. Note you will need access to an eMail account to confirm the account setup activity. Make note of the password you specify.
5. Click **Create Account**. This will cause Bluemix to send an email to the eMail account you specified.
6. Login into the eMail account you specified. Open the eMail with the subject: *Action Required: Confirm your Bluemix account*.
7. Click on the **Confirm Account** button.
8. You now have an active Bluemix trial account.

Task 2: Create Bluemix organization and spaces

1. Log in to Bluemix at: https://console.ng.bluemix.net/

2. If this is your first time logging into this Bluemix account, You will be asked to enter an organization name. Ensure you are in the *US South* region, pick one of the suggested defaults, such as your Bluemix ID or eMail address as the Organization name and click **Create**.



Create Space

The 'BluemixDevOps03' organization does not have a space in the 'US South' region. When you create a space for your organization, it will be created in this region.

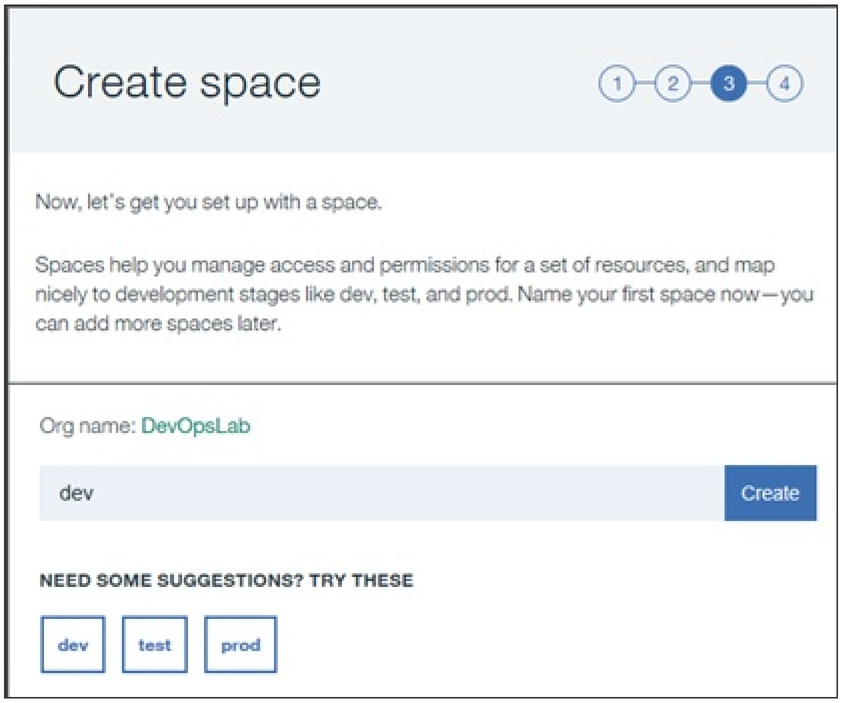
BluemixDevOps03

If this is not your first time to log in to this Bluemix account, and this is a trial account, use the Organization you created when you first logged into this Bluemix account.

3. We will create three spaces (*dev*, *qa* and *prod*) for use in this lab. If this is your first time logging into this Bluemix account, you will be prompted to create a space right after creating an Organization. Enter **dev** as the

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM

space name and click **Create**.


Create space

Now, let's get you set up with a space.

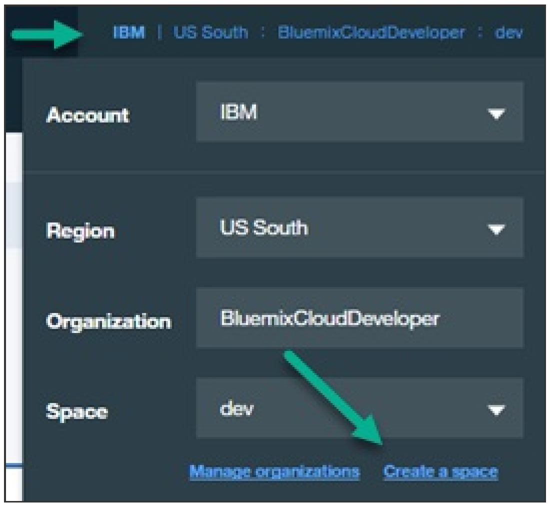
Spaces help you manage access and permissions for a set of resources, and map nicely to development stages like dev, test, and prod. Name your first space now—you can add more spaces later.

Org name: DevOpsLab

dev

Click **I'm Ready**. If you are not prompted to create a space, create the *dev* space when you create the *qa* and *prod* space below.

4. Click the upper-right hand corner environment settings and click on **Create a space**.


IBM | US South : BluemixCloudDeveloper : dev

Account: IBM

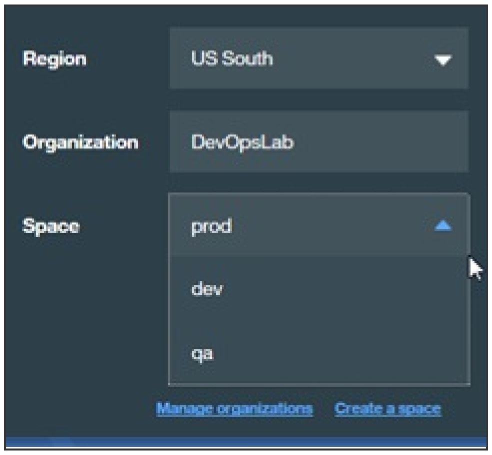
Region: US South

Organization: BluemixCloudDeveloper

Space: dev

5. Enter **qa** as the space name and click **Create**.
6. Repeat the steps to create a space called **prod**.
7. If you have created all three spaces correctly, when you click the upper-right hand corner environment settings and click on the down-arrow for *Space* you will see all three spaces.

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM


Region: US South
Organization: DevOpsLab
Space: prod

Task 3: Create GitHub account

1. If you already have a GitHub account, skip this task.
2. In a web browser, enter the following URL: https://github.com/.
3. Follow the directions to fill out the form. Note you will need access to an eMail account to confirm the account setup activity. Make note of the password you specify.
4. Click on the **Sign up for GitHub** button. This will cause GitHub to send an email to the eMail account you specified.
5. Login into the eMail account you specified. Open the eMail from GitHub with the subject: *Please verify your email address*.
6. Click on the **Verify email address** link.
7. You now have an active GitHub account.

Task 4: Set up Slack access

There are a number of ways to access Slack. One is through the web browser, another is through a desktop application and a third is through your mobile device. The method used in these labs in web browser. Feel free to install the desktop or mobile app if you prefer accessing Slack in that way.

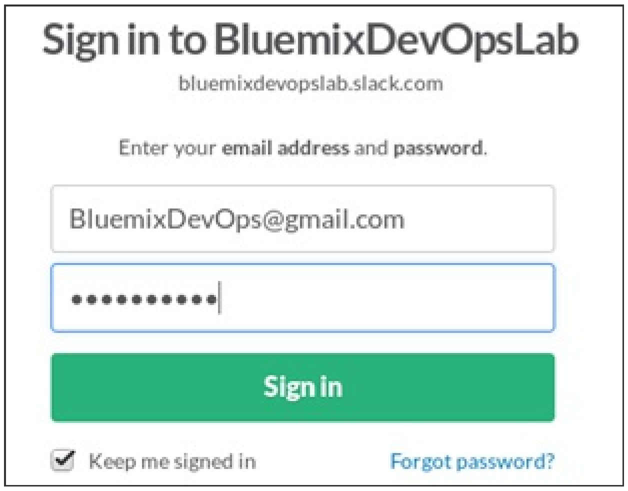
1. In a web browser, open a new tab and go to the following URL to go to the (already created) Slack team. You could create and administer your own Slack team if you would prefer, the instructions for doing so are not part of this exercise. Additionally, you could also download and install the Slack app to your SmartPhone and access Slack that way.

   https://bluemixdevopslab.slack.com

2. Enter the following information:

   1. Email address: **BluemixDevOps@gmail.com**
   2. Password: **bluemix4me**

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM


Sign in to BluemixDevOpsLab

bluemixdevopslab.slack.com

Enter your **email address** and **password**.

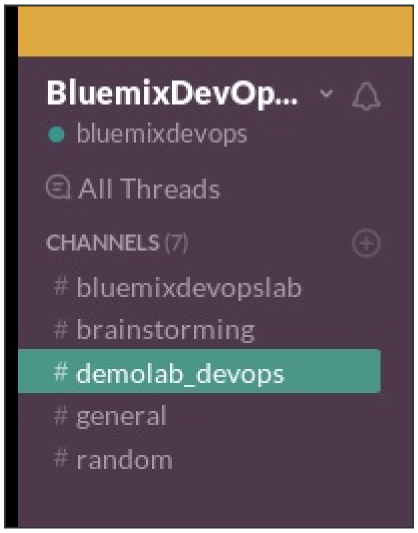
BluemixDevOps@gmail.com

••••••••••

and click **Sign in**.

Why don't we have you use your *own* email address? If we did, the Slack team administrator would have to give that email address permission to access the team. So if you want manage your own Slack team, you can do so, remembering to modify the Team name, channel, etc. for your Slack team.

3. Click on the **demolab_devops** channel to show the messages for that channel.



This channel will show all the messages the Toolchain sends to it.

© Copyright IBM Corp. 2017

Course materials may not be reproduced in whole or in part without the prior written permission of IBM

Lab 1: Read Template Paper

Objective

This lab is a reading exercise that will introduce you to some of the other ways to create and use Toolchains. There is no need to perform the activities described in this paper, this section is for information and learning only.

Tasks:

- Review paper on using templates to create tool chains located here: https://ibm.biz/Bds4cG **Tasks**:
- Task 1: Review paper on using templates to create tool chains

Task 1: Review paper on using templates to create tool chains

This is a reprint of the article located at: https://ibm.biz/Bds4cG

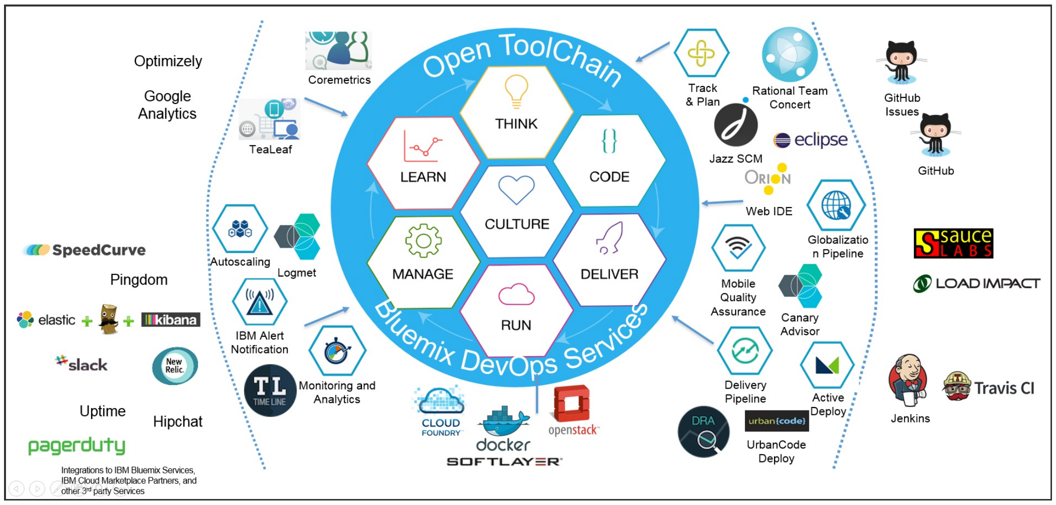
IBM® Bluemix® now automates the setup of DevOps toolchains that work well together in real-world projects. You no longer need to spend precious time assembling and maintaining tools from a rapidly evolving ecosystem; instead, you can maximize focus on your business objectives. You can create or clone toolchain templates and roll them out at scale to teams across your organization. You can then invite any number of users to your organization to grant them access to your toolchains.

You can try out a toolchain by clicking this link.

The toolchain is for an online store application that consists of three microservices. It's preconfigured for continuous delivery across multiple promotion environments, source control, blue-green deployment, functional testing, issue tracking, online editing, and messaging.

Alternatively, from the DevOps dashboard tab, click the **Toolchains** tab and click **Create a Toolchain**. The Microservices toolchain is one of the toolchain templates.

This evolution of IBM Bluemix DevOps Services is now an integral part of the Bluemix platform, and it is initially available on the public cloud (Bluemix Public). The toolchain technology is based on an open architecture, called *Open Toolchain*, that provides an API for integration of arbitrary tools, including IBM and third-party tools, either hosted in Bluemix or elsewhere, even in your enterprise network. Open Toolchain also provides the mechanics to define a toolchain from textual templates that can be automatically instantiated, and API support for analytics that can drive your DevOps intelligence.

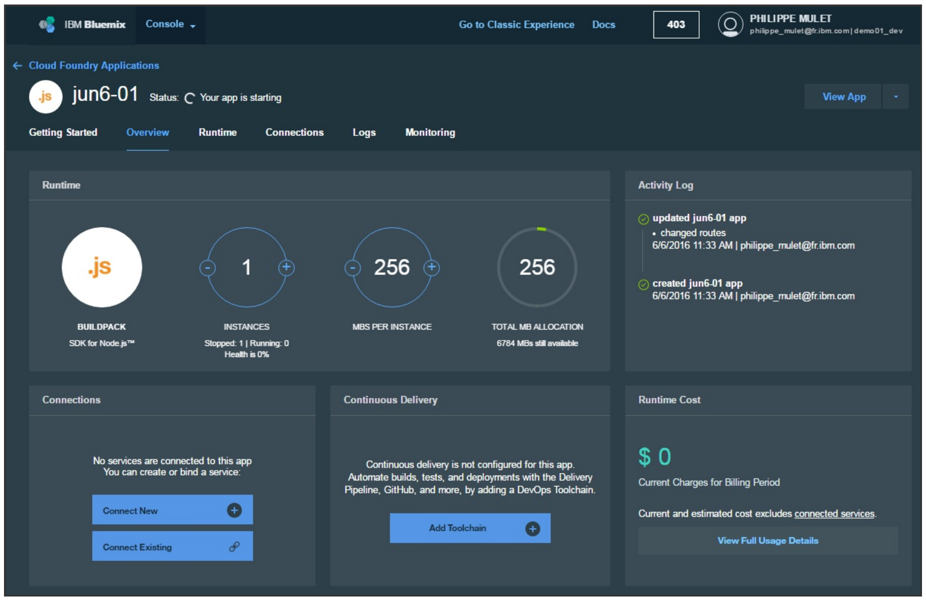


© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM

You can get started with toolchains in Bluemix in several ways:

- Directly from an application's Overview page, on the Continuous Delivery tile, click **Add Toolchain** to associate a continuous delivery toolchain for this application.

  Note: You can insert your own GitHub repository as part of this process.



updated jun6-01 app
- changed routes

6/6/2016 11:33 AM | philippe_mulet@fr.ibm.com

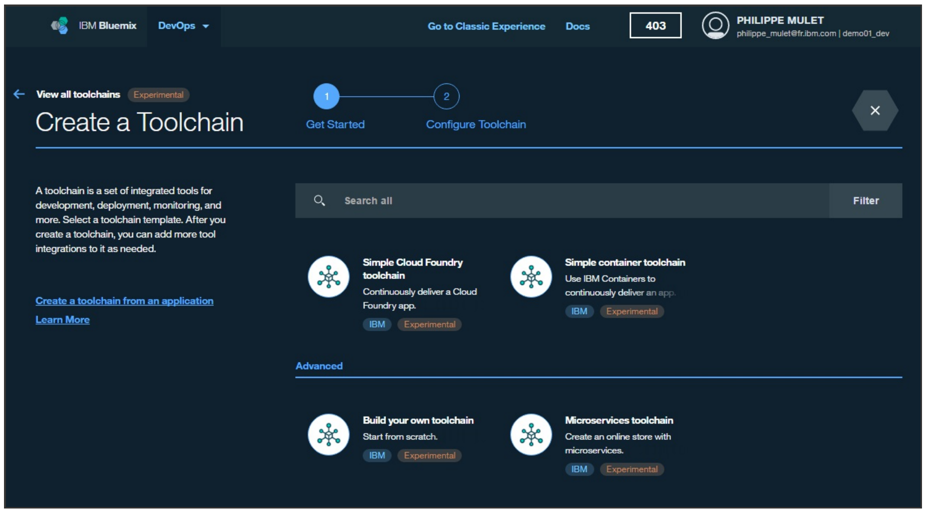
created jun6-01 app
6/6/2016 11:33 AM | philippe_mulet@fr.ibm.com

No services are connected to this app
You can create or bind a service:

Continuous delivery is not configured for this app. Automate builds, tests, and deployments with the Delivery Pipeline, GitHub, and more, by adding a DevOps Toolchain.

Current and estimated cost excludes connected services.

- From the Toolchains tab in Bluemix, you can find the toolchains for your organization and add more or use predefined templates by clicking the add button.



A toolchain is a set of integrated tools for development, deployment, monitoring, and more. Select a toolchain template. After you create a toolchain, you can add more tool integrations to it as needed.

Continuously deliver a Cloud Foundry app.

Use IBM Containers to continuously deliver an app.

Start from scratch.

Create an online store with microservices.

Templates are defined textually by using YAML syntax. The templates are stored in GitHub repositories, and they can be freely cloned and adjusted to fit your needs. The templates that are shown in the previous image are available in this GitHub organization: https://github.com/open-toolchain. You can clone or fork them as needed. Pull requests are welcome.

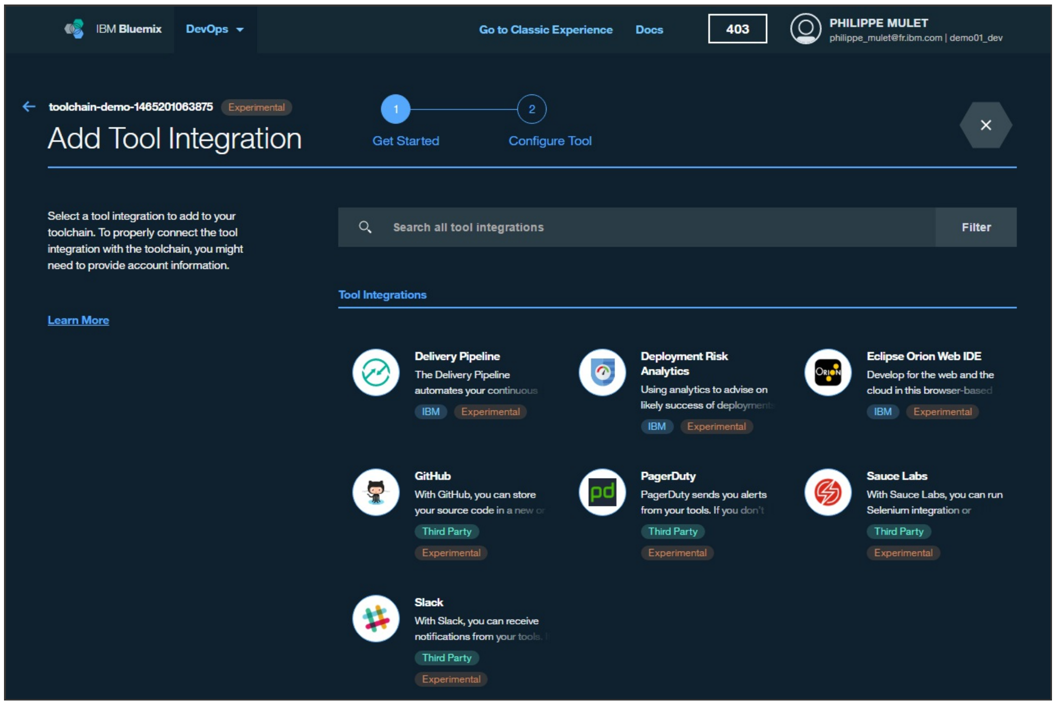
- From a **Create toolchain** button on a web page, readme file, blog post, or article that implements a toolchain when you click it. This functionality is like the Deploy to Bluemix button.

In addition to the button that is provided in this blog post, you can see how the Bluemix Garage Method is effectively

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM

turning microservices theory into practice with a toolchain template and a button. The Bluemix Garage Method combines practices for successful cloud projects with advice on tools and toolchains for delivering applications in various problem domains.

After you create a toolchain, you can add more tool integrations to it, reconfigure the tool integrations, or remove a tool integration. To find the Tools Integration page, click the add button, which looks like a blue hexagon. The list of available integrations is expected to grow quickly.



Select a tool integration to add to your toolchain. To properly connect the tool integration with the toolchain, you might need to provide account information.

Delivery Pipeline The Delivery Pipeline automates your continuous — IBM, Experimental

Deployment Risk Analytics Using analytics to advise on likely success of deployment — IBM, Experimental

Eclipse Orion Web IDE Develop for the web and the cloud in this browser-based — IBM, Experimental

GitHub With GitHub, you can store your source code in a new or — Third Party, Experimental

PagerDuty PagerDuty sends you alerts from your tools. If you don't — Third Party, Experimental

Sauce Labs With Sauce Labs, you can run Selenium integration or — Third Party, Experimental

Slack With Slack, you can receive notifications from your tools. — Third Party, Experimental

Stay tuned for more information about the Open Toolchain SDK and template format. In the meantime, you can learn more and explore in many ways:

- Watch this toolchains introduction video.
- Learn more from the toolchains documentation.
- Jump into the toolchain experience on Bluemix at https://console.ng.bluemix.net/devops. Let us know what you think by clicking the feedback link in the lower-right corner of the toolchain pages.

Enjoy toolchains on Bluemix!

Philippe Mulet Lead Architect, IBM Bluemix DevOps Services

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM

Lab 2: Create Order Toolchain Using Pre-Built Template

Objective

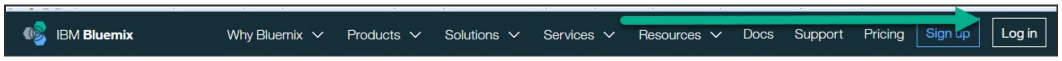
This lab creates a simple toolchain for the Orders API microservice and then examines it. It assumes that a Bluemix organization and *dev*, *qa* and *prod* Spaces are already created.

Tasks:

- Task 1: Log into IBM Bluemix
- Task 2: Create Toolchain
- Task 3: Explore Toolchain
- Task 4: Explore the Delivery Pipeline

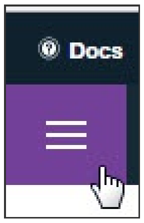
Task 1: Log into IBM Bluemix

1. If you are not already logged into IBM Bluemix, log into IBM Bluemix (https://www.ibm.com/cloud-computing/bluemix/).



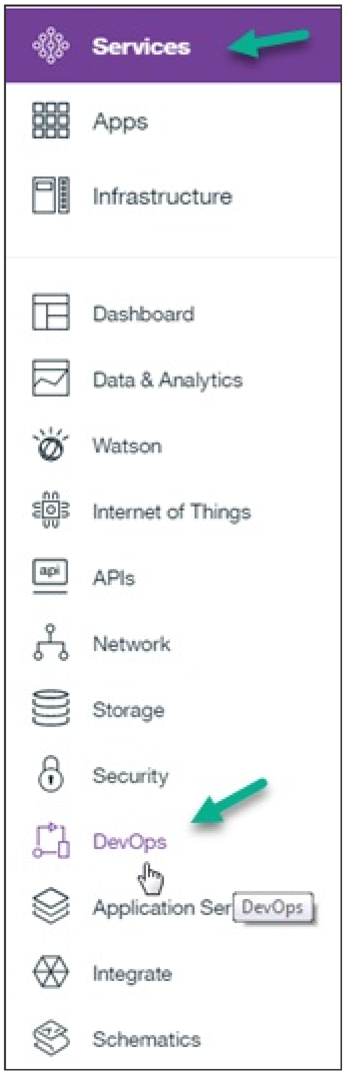
Task 2: Create Toolchain

1. We need to get to DevOps Services. Click on the **Bluemix menu bar** in the upper left corner.

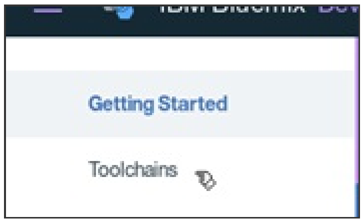


and click on **Services** then **DevOps**.

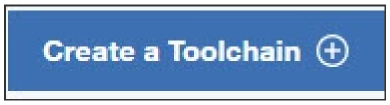
© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM



2. Click on **Toolchains**.



3. Click on **Create a Toolchain**.



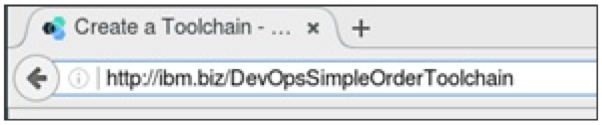
4. These are the Toolchain templates provided by Bluemix. We could use one of those to start creating a Toolchain and customize the necessary information such as the current GitHub repo, Toolchain name, etc.

   Instead, a custom Toolchain template was created (from the provided *Simple Cloud Foundry toolchain*) and we will use this to create our first Toolchain. (This means you too could create custom templates for your

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM

enterprise!)

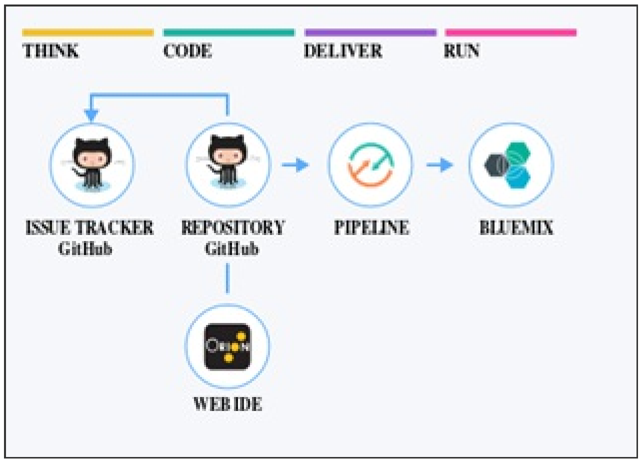
5. Enter the following into the web browser (this will get expanded): http://ibm.biz/DevOpsSimpleOrderToolchain



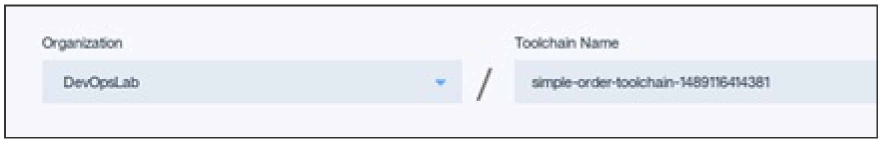
and press *Enter*.

6. If you haven't authorized Bluemix to access GitHub, you need to:
   1. Click **Authorize** to go to the GitHub website.
   2. If prompted for credentials, enter your GitHub username and password. and click **Sign in**.
   3. Click **Authorize application**.

7. The input panel for the Toolchain creation is displayed. A graphical representation of the Toolchain is at the top. This Toolchain uses the GitHub repository to store and version code. It includes the GitHub Issue tracker. It has one Delivery Pipeline for building and deploying code and the application gets deployed to Bluemix.



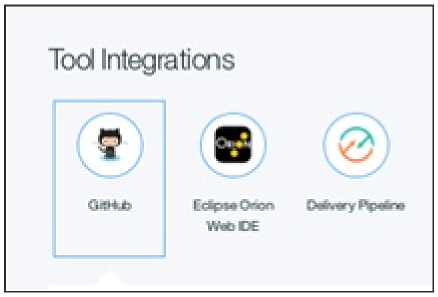
8. The *Organization* is pre-populated with the Organization we created earlier. The *Toolchain Name* is pre-populated with the name specified in the Toolchain template and a timestamp appended to keep the Toolchain Name and associated artifacts unique. An enterprise might rename the Toolchain and associated artifacts to something more meaningful, but we will leave it as is. These instructions will not use a specific timestamp as the timestamp will be unique. All the fields and field values are specified in the Toolchain template.



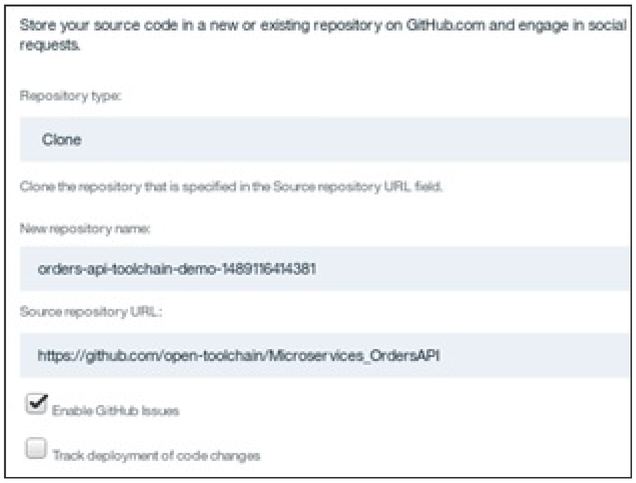
Organization: DevOpsLab / Toolchain Name: simple-order-toolchain-1489116414381

9. The next section displays the configuration for the Tool Integrations. The blue box surrounds and highlights the tool for which integration information is displayed, initially GitHub.

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM



10. The next section has details about the GitHub repo. It will be cloned to the *New repository name* from the *Source repository URL* so we can make changes without impacting anyone else. The *New repository name* will be created for the GitHub user we created.



Store your source code in a new or existing repository on GitHub.com and engage in social requests.

Repository type:

Clone

Clone the repository that is specified in the Source repository URL field.

New repository name:

orders-api-toolchain-demo-1489116414381

Source repository URL:

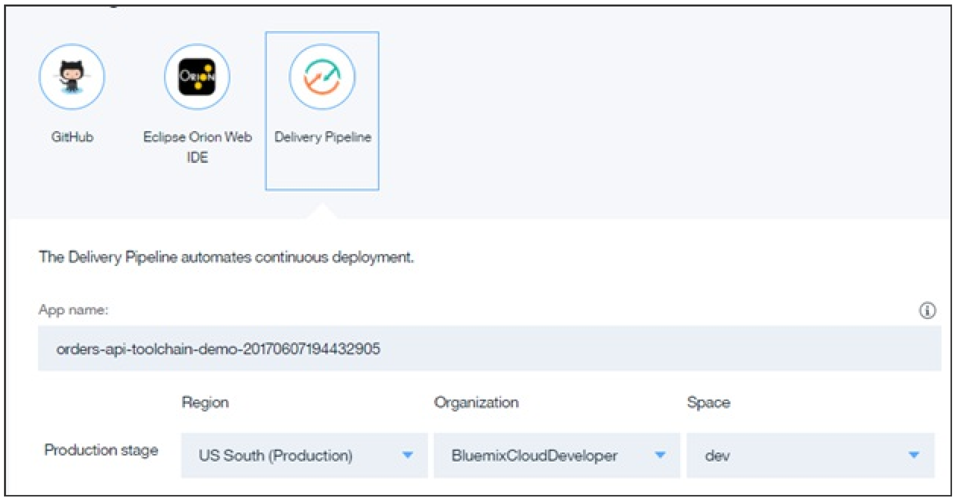
https://github.com/open-toolchain/Microservices_OrdersAPI

☑ Enable GitHub Issues

☐ Track deployment of code changes

11. Click on *Eclipse Orion Web IDE*. No information is needed to integrate this browser based Integrated Development Environment (IDE).

12. Click on *Delivery Pipeline*. This displays the configuration information for the Delivery Pipeline. The application name to be deployed is the *App name*. This pipeline only has one Stage or set of jobs. We will see what gets specified there later. This Stage will deploy the *Orders app name* to the specified Region (US South), Organization, and Space (dev).



The Delivery Pipeline automates continuous deployment.

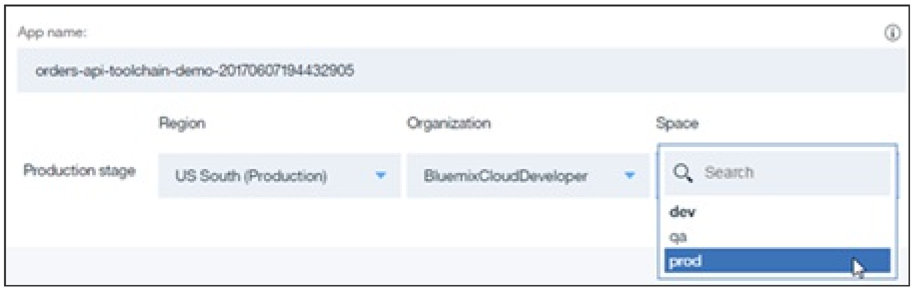
App name:

orders-api-toolchain-demo-20170607194432905

	Region	Organization	Space
Production stage	US South (Production)	BluemixCloudDeveloper	dev

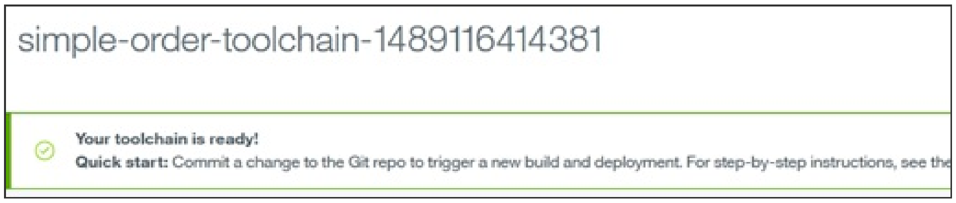
13. Use the drop down arrow for space to specify *prod*.

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM


App name:
orders-api-toolchain-demo-20170607194432905

	Region	Organization	Space
Production stage	US South (Production)	BluemixCloudDeveloper	prod

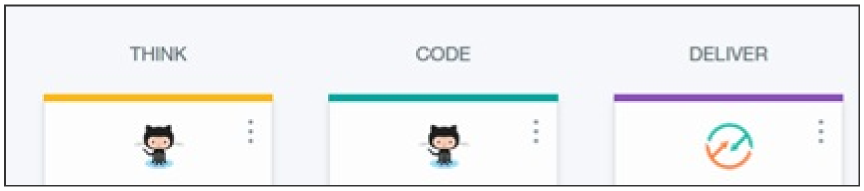
14. Click **Create**. Your Toolchain is created and you are redirected to the Toolchain panel.


simple-order-toolchain-1489116414381

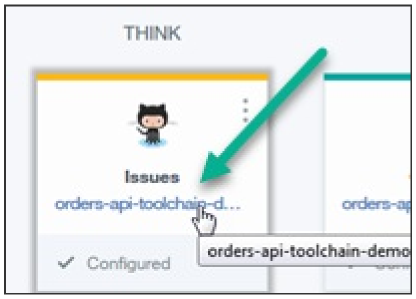
Your toolchain is ready!
Quick start: Commit a change to the Git repo to trigger a new build and deployment. For step-by-step instructions, see the

Task 3: Explore Toolchain

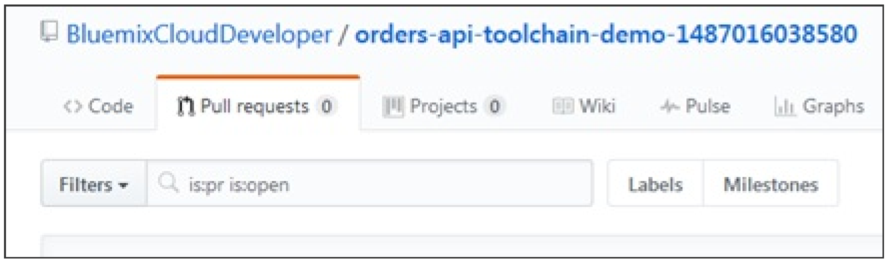
1. IBM Bluemix created the Continuous Delivery Toolchain based on the customized simple template. At the top you see the pieces of the Bluemix Garage Method and where each tool integration fits.



2. **Think** is where the GitHub Issues database is listed. Click on the **orders-api-toolchain-demo-*timestamp*** link (or right-mouse button click and select **Open Link in New Tab**, then select the new tab to save time later on).



3. This displays the **GitHub Issues** page. Issues are used to track todos, bugs, feature requests, and more. Each GitHub repository (*repo* for short) can include issues.


BluemixCloudDeveloper / orders-api-toolchain-demo-1487016038580

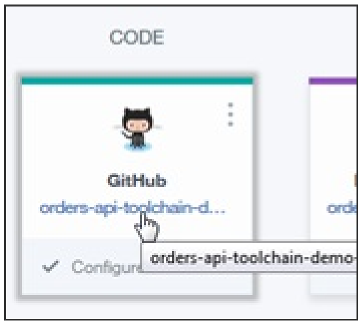
Return to the Toolchain by either clicking on the **Go back one page** arrow on the browser or, if you clicked the

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM

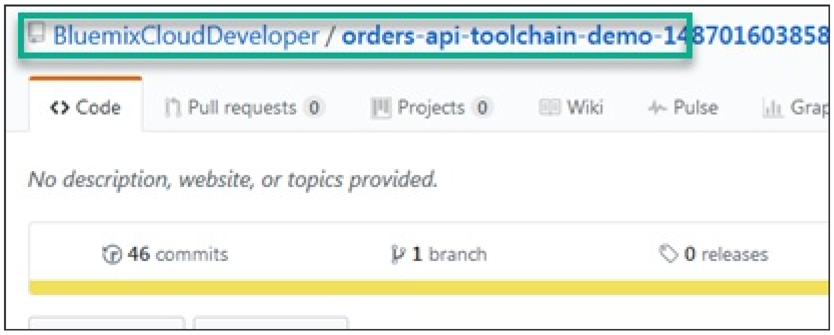
right-mouse button to open a new browser tab, close the GitHub Issues browser tab. (Note that the remainder of these lab instructions may not go into this level of detail on opening and closing pages and tabs - pick the method that is best for you.)

4. **Code** is where GitHub code repos and Eclipse Orion Web IDE are integrated.

    1. Clicking on the repo

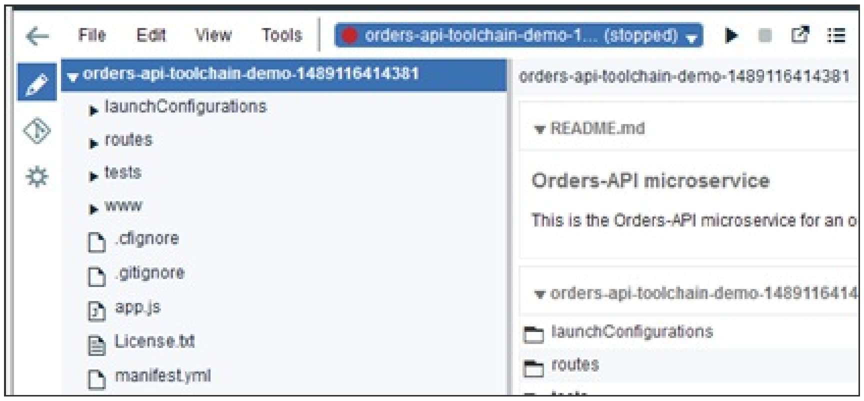
       will display the respective (cloned) repo

       No description, website, or topics provided.

       The creation of the Toolchain did clone the repo in GitHub. Return to the Toolchain page by either clicking on the **Go back one page** arrow on the browser or, if you clicked the right-mouse button to open a new browser tab, close the GitHub Issues browser tab.

    2. Clicking on the **Eclipse Orion Web IDE** will display the Web editor.

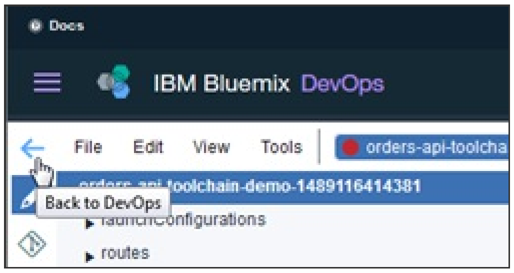
       
       File
Edit
View
Tools
orders-api-toolchain-demo-1... (stopped)
orders-api-toolchain-demo-1489116414381
launchConfigurations
routes
tests
www
.cfignore
.gitignore
app.js
License.txt
manifest.yml
orders-api-toolchain-demo-1489116414381
README.md
orders-api-toolchain-demo-1489116414
launchConfigurations
routes

       Orders-API microservice

       This is the Orders-API microservice for an o

    3. Return to the Toolchain by clicking on the Bluemix back arrow.

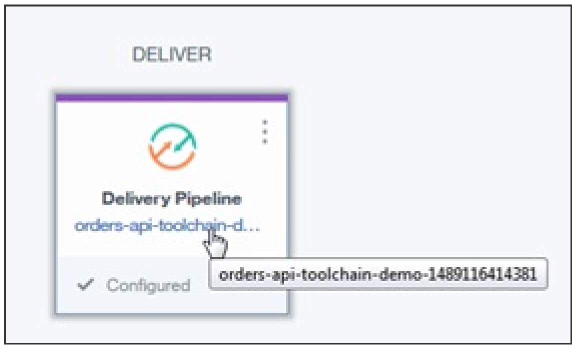
© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM



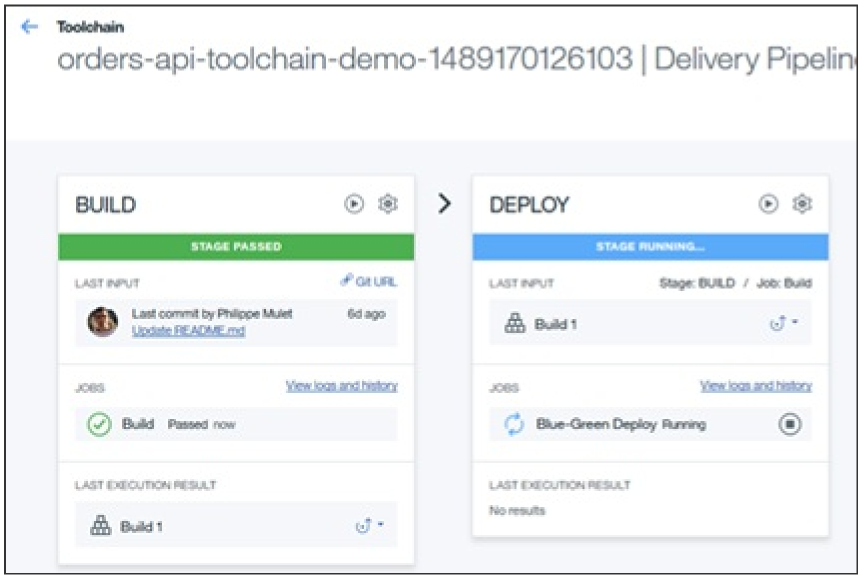
5. **Deliver** is where the code gets built, tested and deployed through the integrations of build pipelines. We look at those in the next task.

Task 4: Explore the Delivery Pipeline

1. Click on the circle in the center of the *orders-api-toolchain-demo-timestamp* Delivery Pipeline tile.


DELIVER
Delivery Pipeline
orders-api-toolchain-d...
Configured
orders-api-toolchain-demo-1489116414381

to display the Orders API Delivery Pipeline.



2. While we were busy exploring the toolchain, Bluemix started to execute the just created Delivery Pipeline. A *Delivery Pipeline* consists of one or more *Stages*. Stages organize input and jobs as your code is built, deployed, and tested. Stages accept input from either source control repositories (SCM repositories such as GitHub) or build jobs (build artifacts) in other stages. When you create your first stage, the default settings are set for you on the *INPUT* tab for the stage.

   By default in a stage, builds and deployments are run automatically every time changes are delivered to a

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM

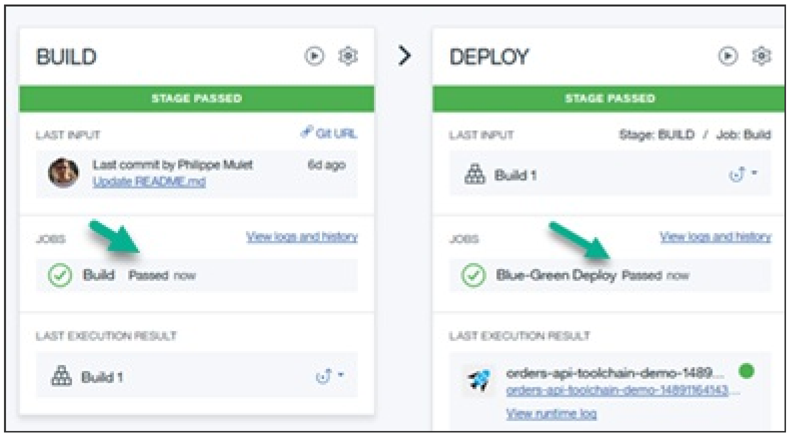
project's SCM repository. Stages and jobs run serially and enable flow control for your work. For example, you might place a test stage before a deployment stage. If the tests in the test stage fail, the deployment stage won't run.

Stages consist of one or more *Jobs*. A job is an execution unit within a stage. Jobs in a stage run sequentially. By default, if a job fails, subsequent jobs in the stage do not run.

Before a job is run, its working directory is populated with input defined at the stage level. For example, you might have a stage that contains a test job and a deploy job. If you install dependencies on one job, they are not available to the other job. However, if you make the dependencies available in the stage's input, they are available to both jobs.

Except for Simple-type build jobs, when you configure a job, you can include UNIX shell scripts that include build, test, or deployment commands. Because jobs are run in ad hoc containers, the actions of one job cannot affect the run environments of other jobs, even if those jobs are part of the same stage.

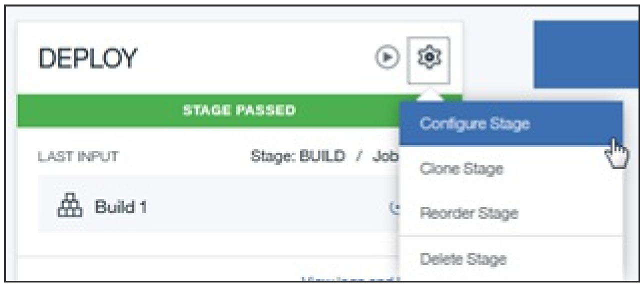
The delivery pipeline displays the status of each stage in the pipeline. Depending on how fast you went through the steps after the Toolchain was created, you may have to wait a moment or two for the *DEPLOY* stage to complete.



The **BUILD** and **DEPLOY** stage passed.

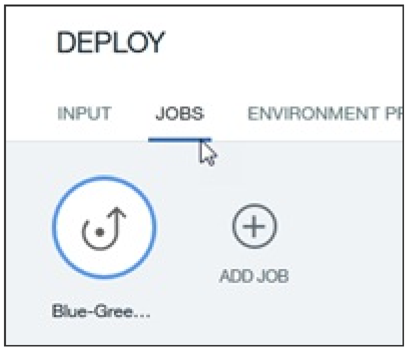
3. The *BUILD* stage in this case is simply cloning the repo.

4. In the *DEPLOY* stage, click the gear icon and then **Configure Stage** to configure the stage.



5. By default, the JOBS view is displayed. The *INPUT* tab displays the input to the stage, the *JOBS* tab displays the discrete jobs of the stage, and the *ENVIRONMENT PROPERTIES* tab displays variables used by the jobs in the stage.

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM



6. There is only one job in this stage, the job named *Blue-Green Deploy*. Scrolling down you see the *Deployer Type* (Cloud Foundry), *Target* region (US South), *Organization*, *Space* (prod) and the deployed *Application Name* (orders-api-toolchain-demo-*timestamp*).

7. Look at the *Deploy Script*. This script does a *Blue-green* deployment. A blue-green deployment is a release technique reducing downtime and risk by running two identical production environments called Blue and Green. At any time, only one of the environments is live, with the live environment serving all production traffic.

8. The first section put a comment into the log file and then issue the Cloud Foundry (*cf* command to create the service *cloudantNoSQLDB*. It the service already exists, the script simply continues.

```
echo "Attempting to create cloudantNoSQLDB Lite myMicroservicesCloudant for use by the
microservices. It is not a problem if it already exists, we simply continue."
cf create-service cloudantNoSQLDB Lite myMicroservicesCloudant
```

9. The next section of the script echos a comment to the log and then issues the cf (Cloud Foundry) *app* command for the application (*$CF_APP*) we want to deploy. If the application is not found (for example if this is the first time the application is being deployed), the *app* command fails, at which point the Cloud Foundry *push* command is used to deploy the *$CF_APP*.

```
# Push app
echo "If the $CF_APP does not exist, push the app."
if ! cf app $CF_APP; then
  cf push $CF_APP
```

10. The next section, which gets executed if the application is already deployed, defines how to undo what we are about to try in case of error.

```
else
  OLD_CF_APP=${CF_APP}-OLD-$(date +"%s")
  rollback() {
    set +e
    if cf app $OLD_CF_APP; then
      cf logs $CF_APP --recent
      cf delete $CF_APP -f
      cf rename $OLD_CF_APP $CF_APP
    fi
    exit 1
  }
```

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM

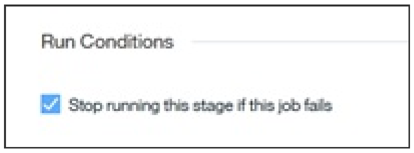
11. The next section renames the deployed application (*$CF_APP*) to a temporary name (*$OLD_CF_APP*), issues the cf (Cloud Foundry) *push* command to deploy the new version and if the cf *push* is successful, deletes the old version using the cf *delete* command.

```
set -e
trap rollback ERR
echo "If the $CF_APP does exist, rename it."
cf rename $CF_APP $OLD_CF_APP
echo "And push out the new version."
cf push $CF_APP
echo "If the push is successful, delete the old app."
cf delete $OLD_CF_APP -f
```

12. The final section exports the application name and URL so other jobs in the Pipeline can use it.

```
# Export app name and URL for use in later Pipeline jobs
export CF_APP_NAME="$CF_APP"
export APP_URL=http://$(cf app $CF_APP_NAME | grep urls: | awk '{print $2}')
# View logs
#cf logs "${CF_APP}" --recent
```

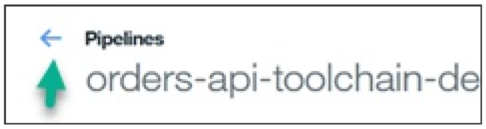
13. Under the Deploy Script window is the option to stop the stage if this job fails, selected for this job.


Run Conditions

☑ Stop running this stage if this job fails

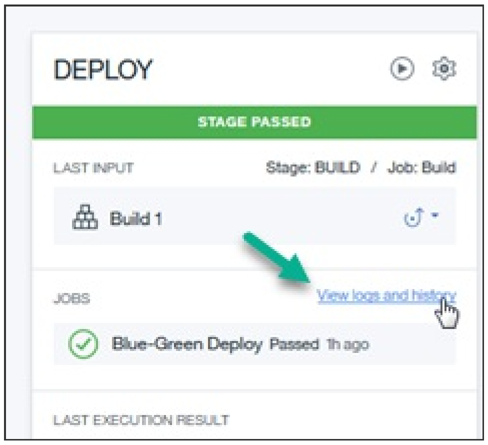
14. Click on ENVIRONMENT PROPERTIES to see the user-defined environment properties used to share information in the Pipeline. There are also a number of pre-defined Environment Properties. For example, the *CF_APP* property is pre-defined and for deployments is the name of the app to deploy. This property is required for deployment and can be specified in the script itself, the deploy job configuration interface, or the project's manifest.yml file.

15. Click the left arrow to the left of *Pipeline* to return to the Delivery Pipeline.

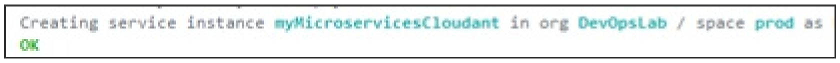


16. In the **DEPLOY** stage, click **View logs and history** to display the commands and results of the stage.

Course materials may not be reproduced in whole or in part without the prior written permission of IBM

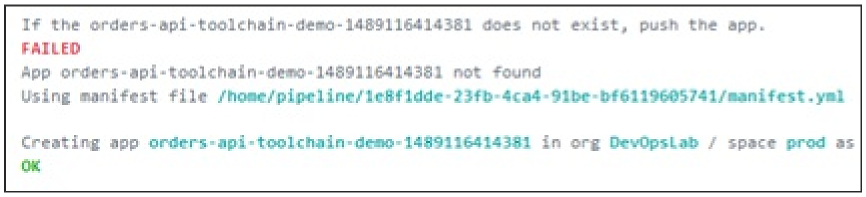


17. If you scroll through the log, you can see the details of the job execution. Note that the *_myMicroservicesCloudant* service was created.



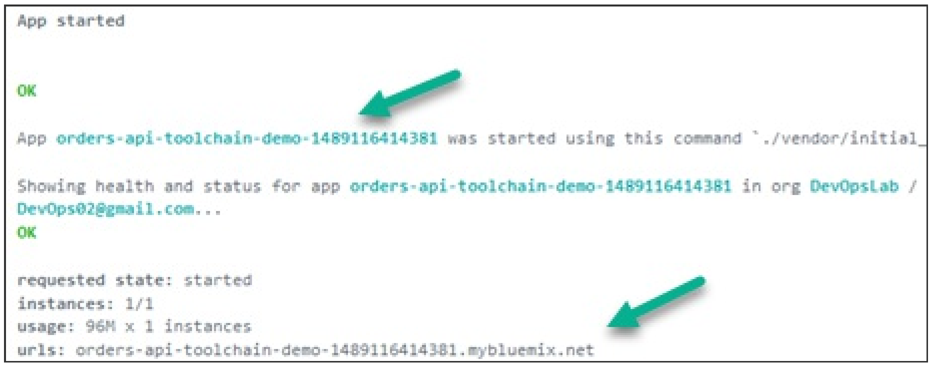
```
Creating service instance myMicroservicesCloudant in org DevOpsLab / space prod as
OK
```

18. Then the existence of the deployed app *orders-api-toolchain-demo-timestamp* is checked for. As it does not exist, there is no need to rename it. Just deploy it.



```
If the orders-api-toolchain-demo-1489116414381 does not exist, push the app.
FAILED
App orders-api-toolchain-demo-1489116414381 not found
Using manifest file /home/pipeline/1e8f1dde-23fb-4ca4-91be-bf6119605741/manifest.yml

Creating app orders-api-toolchain-demo-1489116414381 in org DevOpsLab / space prod as
OK
```

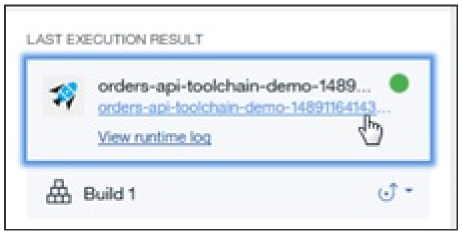
19. Scroll past the all the necessary runtime packages to get to the bottom of the log, showing the application starting and the URL to access the application.



```
App started


OK

App orders-api-toolchain-demo-1489116414381 was started using this command `./vendor/initial_

Showing health and status for app orders-api-toolchain-demo-1489116414381 in org DevOpsLab /
DevOps02@gmail.com...
OK

requested state: started
instances: 1/1
usage: 96M x 1 instances
urls: orders-api-toolchain-demo-1489116414381.mybluemix.net
```

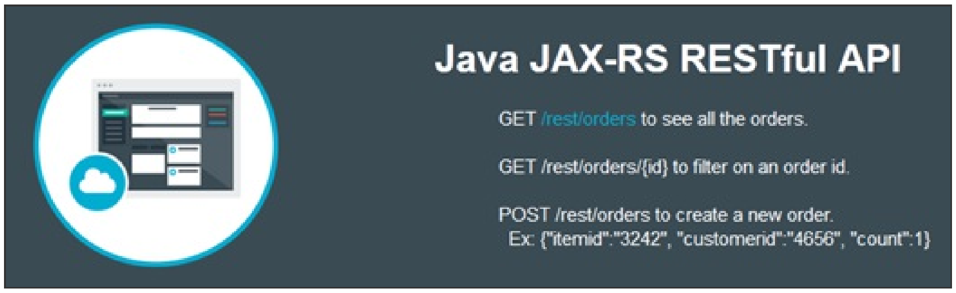
20. Click the left arrow to the left of *Pipeline* to return to the Delivery Pipeline.

21. Click the application URL to display the application.


LAST EXECUTION RESULT
orders-api-toolchain-demo-1489...
orders-api-toolchain-demo-14891164143...
View runtime log
Build 1

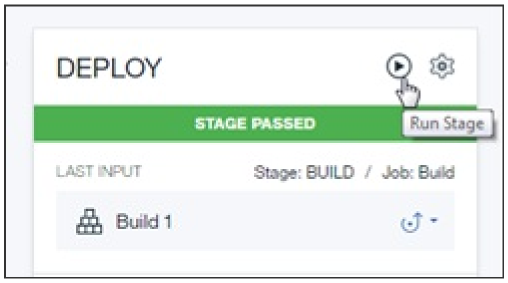
Course materials may not be reproduced in whole or in part without the prior written permission of IBM

22. The application is displayed in a new browser tab.


Java JAX-RS RESTful API
GET /rest/orders to see all the orders.
GET /rest/orders/{id} to filter on an order id.
POST /rest/orders to create a new order.
Ex: {"itemid":"3242", "customerid":"4656", "count":1}

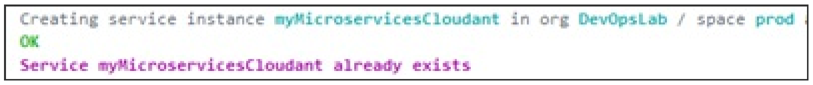
23. Close the browser tab displaying the application.

24. On the *DEPLOY* stage click the **Run Stage** arrow to run the stage again.



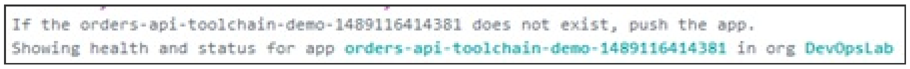
25. When the stage completes, view the log.

26. Note the *_myMicroservicesCloudant* service already existed so there is no need to create it.



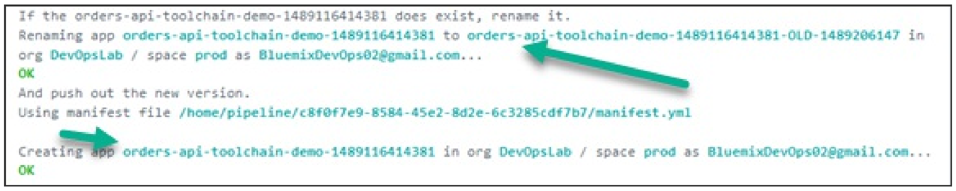
```
Creating service instance myMicroservicesCloudant in org DevOpsLab / space prod
OK
Service myMicroservicesCloudant already exists
```

27. Then the existence of the deployed app *orders-api-toolchain-demo-timestamp* is checked for. This time it does exist as Cloud Foundry returns information about it.



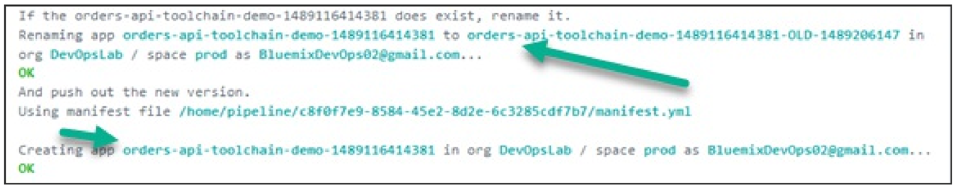
```
If the orders-api-toolchain-demo-1489116414381 does not exist, push the app.
Showing health and status for app orders-api-toolchain-demo-1489116414381 in org DevOpsLab
```

28. Since the application does exist, rename it and push out the new version of the application.



```
If the orders-api-toolchain-demo-1489116414381 does exist, rename it.
Renaming app orders-api-toolchain-demo-1489116414381 to orders-api-toolchain-demo-1489116414381-OLD-1489206147 in
org DevOpsLab / space prod as BluemixDevOps02@gmail.com...
OK
And push out the new version.
Using manifest file /home/pipeline/c8f0f7e9-8584-45e2-8d2e-6c3285cdf7b7/manifest.yml

Creating app orders-api-toolchain-demo-1489116414381 in org DevOpsLab / space prod as BluemixDevOps02@gmail.com...
OK
```

29. Scroll to the bottom where, after the new version of the application is deployed, the old (and renamed) version is deleted.



```
If the orders-api-toolchain-demo-1489116414381 does exist, rename it.
Renaming app orders-api-toolchain-demo-1489116414381 to orders-api-toolchain-demo-1489116414381-OLD-1489206147 in
org DevOpsLab / space prod as BluemixDevOps02@gmail.com...
OK
And push out the new version.
Using manifest file /home/pipeline/c8f0f7e9-8584-45e2-8d2e-6c3285cdf7b7/manifest.yml

Creating app orders-api-toolchain-demo-1489116414381 in org DevOpsLab / space prod as BluemixDevOps02@gmail.com...
OK
```

30. Click the left arrow to the left of *Pipelines* to return to the Delivery Pipeline.

31. Click **Toolchains** to return to the *Toolchains* page.

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM

Lab 3: Create Catalog Toolchain by Hand

Objective

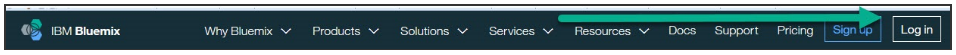
This lab manually creates a simple toolchain for the Catalog API microservice and then configures it. It assumes that a Bluemix Organization and *dev*, *qa* and *prod* Spaces are already created.

Tasks:

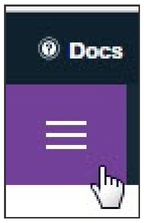
Task 1: Create Toolchain

Throughout the lab, the phrase *timestamp* is used to indicate the same timestamp string that was appended to *simple-order-toolchain*. While a timestamp string is not required, it does help make the name of the created objects unique.

1. If you are not already logged into IBM Bluemix, log into IBM Bluemix (https://www.ibm.com/cloud-computing/bluemix/).

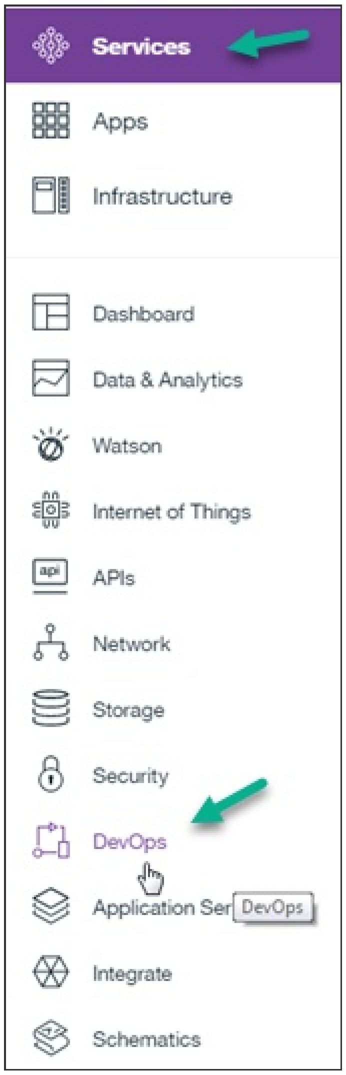


2. If you don't see a button called *Create a Toolchain*, you need to get to DevOps Services. Click on the **Bluemix menu bar** in the upper left corner.



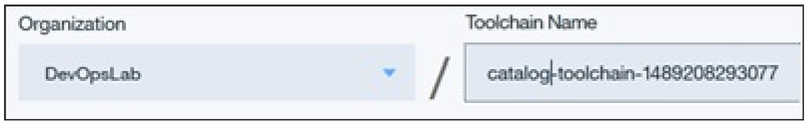
and click on **Services** then **DevOps**

Course materials may not be reproduced in whole or in part without the prior written permission of IBM



and click on **Toolchains**.

3. Click **Create a Toolchain**.
4. Click on **Build your own toolchain**.
5. Change the *Toolchain Name* from *empty-toolchain-timestamp* to **catalog-toolchain-*timestamp***.


Organization: DevOpsLab / Toolchain Name: catalog-toolchain-1489208293077

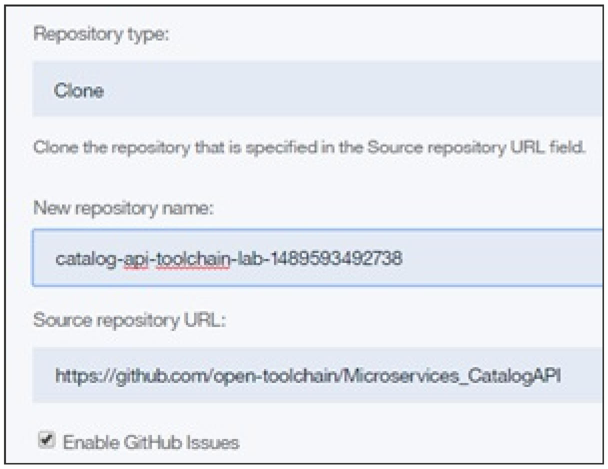
6. Click **Create** to create the Toolchain.
7. The (empty) *catalog-toolchain-timestamp* is displayed.

Task 2: Add and Configure GitHub Integration

The code for the Catalog microservice already exists in a GitHub repository (https://github.com/open-toolchain/Microservices_CatalogAPI). We will clone this repository and link to the clone.

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM

1. Click on **Add a Tool** on the right side of the screen to add a Tool Integration.
2. Click on **GitHub** to add integration with GitHub to the Toolchain.
3. Select *Clone* as the Repository type.
4. Enter **catalog-api-toolchain-lab-*timestamp*** for the *New Repository Name*.
5. Enter *https://github.com/open-toolchain/Microservices_CatalogAPI* for the Source repository URL.



Repository type:

Clone

Clone the repository that is specified in the Source repository URL field.

New repository name:

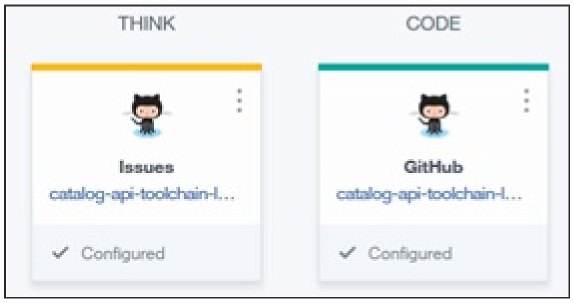
catalog-api-toolchain-lab-1489593492738

Source repository URL:

https://github.com/open-toolchain/Microservices_CatalogAPI

☑ Enable GitHub Issues

6. Click on **Create Integration**. The integration is created.



Task 3: Add Eclipse Orion Web IDE to Toolchain

If we want to modify the application, one convenient way is to use the Eclipse Orion Web IDE.

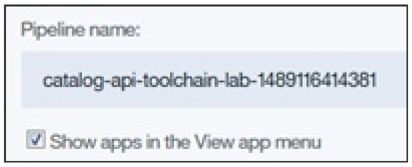
1. On the *catalog-toolchain-timestamp* toolchain's Tool Integrations page, click **Add a Tool**
2. Click **Eclipse Orion Web IDE**.
3. No configuration is needed, so click **Create Integration**.

Task 4: Add Catalog Delivery Pipeline

Now that you have a Git repository clone of the code, we will add a *Delivery Pipeline* to deploy and test the application.

1. Click on **Add a Tool** on the right side of the screen to add a Tool Integration.
2. Click on **Delivery Pipeline** to create a new Delivery Pipeline (we will add Stages and Jobs to this).
3. For 'Pipeline name:', enter "**catalog-api-toolchain-lab-*timestamp***" and ensure 'Show apps in the VIEW APP menu' checkbox is checked.

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM


Pipeline name:

catalog-api-toolchain-lab-1489116414381

☑ Show apps in the View app menu

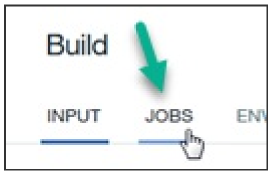
4. Click **Create Integration**.
5. The *catalog-api-toolchain-lab-timestamp* Delivery Pipeline is created.

Task 5: Add Build stage to Catalog Delivery Pipeline

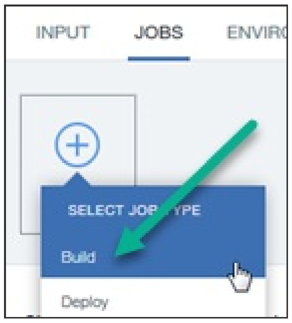
Now to configure the *catalog-api-toolchain-lab-timestamp* Delivery Pipeline. We will make this pipeline a little more complex. We will add four stages: Build, Dev, Test and Prod.

- The **Build** stage has two jobs, performing the initial build of the code from the GitHub Repository then some unit tests.
- The **Dev** stage has two jobs, taking the output from the Build stage and deploying on Bluemix into the *dev* space, then performing automated functional tests.
- The **Test** stage has two jobs, taking the output from the Dev stage and deploying on Bluemix into the *qa* space, then performing automated tests.
- The **Prod** stage has one job, taking the output from the Test stage and deploying on Bluemix into the *prod* space. This stage will perform a Blue-Green deployment, checking to see there is an earlier instance of this application running and if it is, keep it around in case the deploy of the new version of the app has problems. If the new version deploys successfully, the old version is deleted. If not, the new version is deleted and the old version continues to run.

1. Click on the **Delivery Pipeline** tile for the catalog-api-toolchain-lab delivery pipeline.
2. Click **Add Stage**.
3. This is the *INPUT* portion of the stage. Note that the *Input Type* is set to Git Repository_ and the *Git Repository*, *Git URL* and (Git) *Branch* are pre-filled. Also, *Stage Trigger* is set to "Run jobs whenever a change is pushed to Git", resulting in this stage running when Git is updated.
4. Rename the stage from *MyStage* to **Build**.
5. Click the *JOBS* tab so we can add some jobs.



6. Click the **+** and select **Build** for the JOB TYPE.



On the Build configuration panel, note that:

- The job name is *Build* (just like the stage name.)

© Copyright IBM Corp. 2017

Course materials may not be reproduced in whole or in part without the prior written permission of IBM

- *Builder Type* is set to "Simple" (other options are available on the pull-down).
- *Run Conditions* is set to "Stop running this stage if this job fails" to prevent any other jobs in this stage from running and to mark the stage failed if this Job fails.

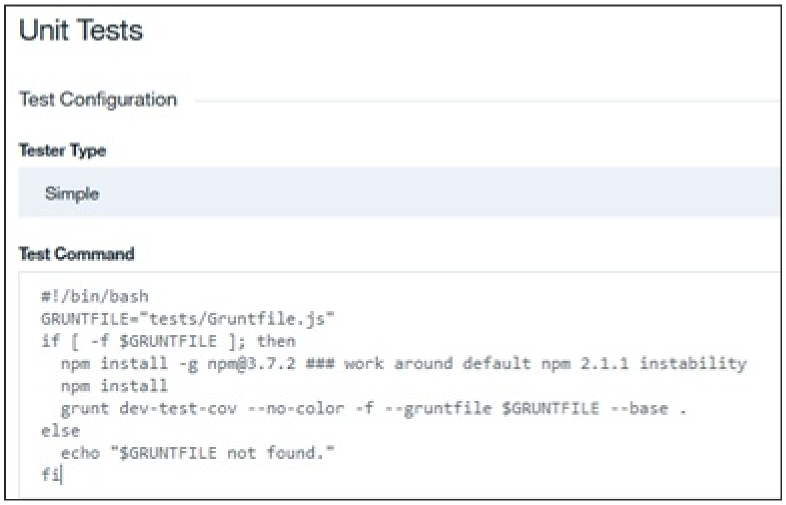
7. Click **ADD JOB**, this time selecting **Test** for the JOB TYPE.

8. Rename the job from *Test* to *Unit Tests*.

9. Enter the following for the *Test Command*. *Note:* You can enter the following URL into another browser tab to display the code for easy copy and pasting: http://ibm.biz/CatalogAPIDevUnitTest

```
#!/bin/bash
GRUNTFILE="tests/Gruntfile.js"
if [ -f $GRUNTFILE ]; then
  npm install -g npm@3.7.2 ### work around default npm 2.1.1 instability
  npm install
  grunt dev-test-cov --no-color -f --gruntfile $GRUNTFILE --base .
else
  echo "$GRUNTFILE not found."
fi
```

This script checks to see if the file *tests/Gruntfile.js* exists. If it does, we install a version of npm then run an automated Grunt tests. If the file *tests/Gruntfile.js* does not exist, simple echo a line into the log file.


Unit Tests

Test Configuration

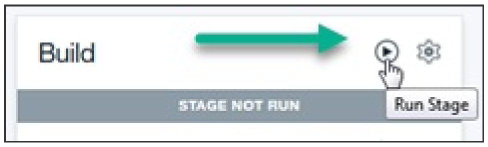
Tester Type: Simple

Test Command

```
#!/bin/bash
GRUNTFILE="tests/Gruntfile.js"
if [ -f $GRUNTFILE ]; then
  npm install -g npm@3.7.2 ### work around default npm 2.1.1 instability
  npm install
  grunt dev-test-cov --no-color -f --gruntfile $GRUNTFILE --base .
else
  echo "$GRUNTFILE not found."
fi
```

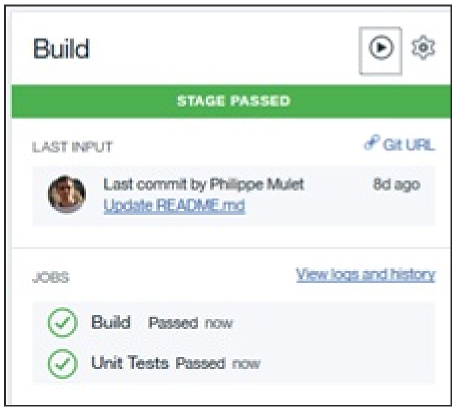
10. Click **Save** to save the *Build* stage.

11. The *Delivery Pipeline* displays the **Build** stage. This stage has not been run. Click on the **Run Stage** icon to run the build.



12. The JOBS section shows the Build (job) progress. After a few moments, the **Build** stage has been successfully executed.

Course materials may not be reproduced in whole or in part without the prior written permission of IBM



13. Click *View logs and history* for the jobs to examine the logs for each job. When done, return to the Delivery Pipeline.

Task 6: Add Dev stage to Catalog Delivery Pipeline

Now we add the *Dev* stage and jobs. The *Dev* stage has two jobs. The first job deploys the just built Catalog API microservice and deploys it into the *dev* space on Bluemix and the second job performs some automated tests on the deployed microservice.

1. Click on **ADD STAGE**.
2. Name the stage **Dev**. Note that:

- *Input Type* is set to Build Artifacts (from the **Build** stage).
- *Stage* and *Job* are both *Build*.
- *Stage Trigger* is set to "Run jobs when the previous stage is completed", resulting in the Dev stage running when the **Build** stage successfully completes.

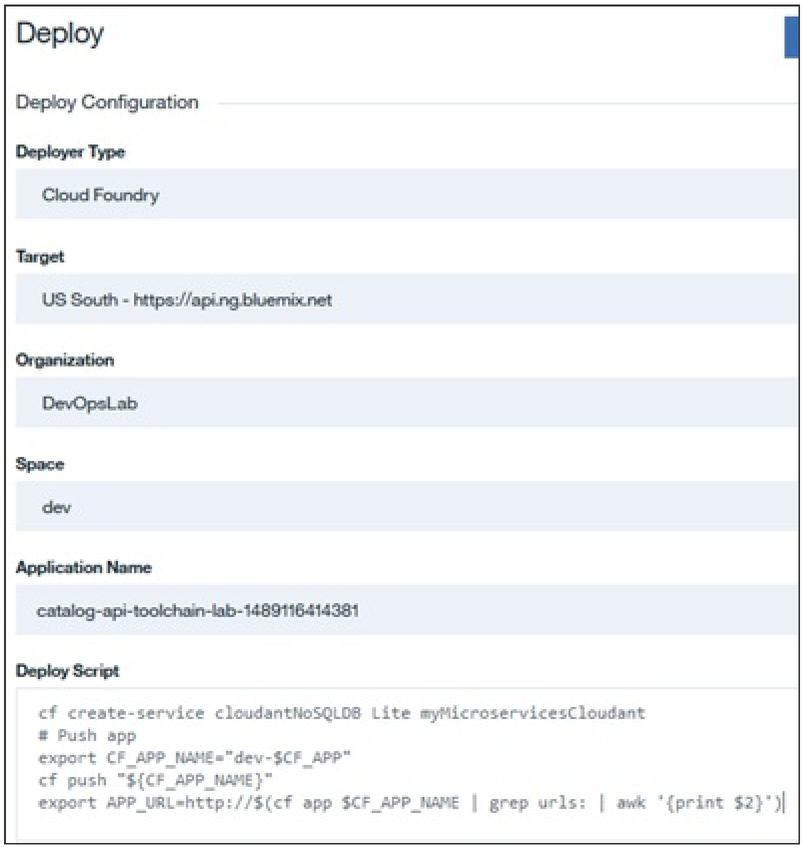
3. Click the **JOBS** tab and add a new job of type **Deploy**. Note that:

- *Deployer Type* is set to "Cloud Foundry" (other options are available on the pull-down).
- *Target* is set to "US South - https://api.ng/bluemix.net" as this is where the code will be deployed.
- *Space* is set to "dev".
- *Application Name* is "catalog-api-toolchain-lab-*timestamp*".

3. Type the following into the *Deploy Script* section. This script first creates the cloudantNoSQLDB (remember, if it already exists, the script simply continues). Then the variable *CF_APP_NAME* is set to the application name (*catalog-api-toolchain-lab-timestamp*) has the space name *dev* added to the front of the name. This keeps the name unique as we will deploy this application to the *qa* and *prod* space later. *Note:* You can enter the following URL into another browser tab to display the code for easy copy and pasting: http://ibm.biz/CatalogAPIDevDeploy

```
cf create-service cloudantNoSQLDB Lite myMicroservicesCloudant
# Push app
export CF_APP_NAME="dev-$CF_APP"
cf push "${CF_APP_NAME}"
export APP_URL=http://$(cf app $CF_APP_NAME | grep urls: | awk '{print $2}')
```

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM


Deploy

Deploy Configuration

Deployer Type: Cloud Foundry

Target: US South - https://api.ng.bluemix.net

Organization: DevOpsLab

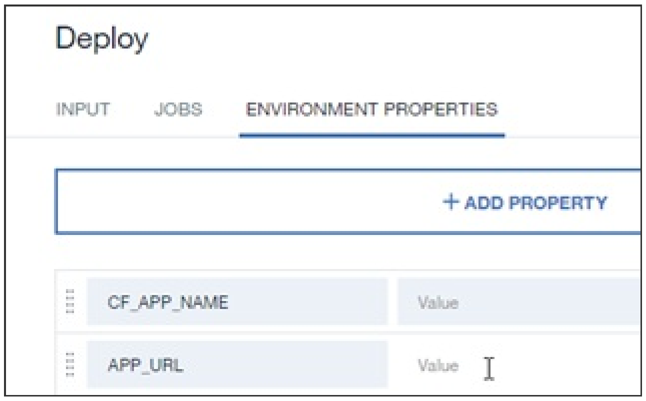
Space: dev

Application Name: catalog-api-toolchain-lab-1489116414381

Deploy Script

```
cf create-service cloudantNoSQLDB Lite myMicroservicesCloudant
# Push app
export CF_APP_NAME="dev-$CF_APP"
cf push "${CF_APP_NAME}"
export APP_URL=http://$(cf app $CF_APP_NAME | grep urls: | awk '{print $2}')
```

4. The bash script just entered into the Deploy Script references both the *CF_APP_NAME* and *APP_URL* environment variables (remember, *CF_APP* is provided by default). These two environment variables are used to pass information between jobs in this stage and need to be added to the environment variables as Text.
5. Click the **ENVIRONMENT PROPERTIES** tab.
6. Click **ADD PROPERTY** and select **Text Property**.
7. Enter **CF_APP_NAME** as the 'Name'. Do not enter anything for the 'Value'.
8. Click **ADD PROPERTY** and select **Text Property**.
9. Enter **APP_URL** as the 'Name'. Do not enter anything for the 'Value'.



CF_APP_NAME	Value
APP_URL	Value

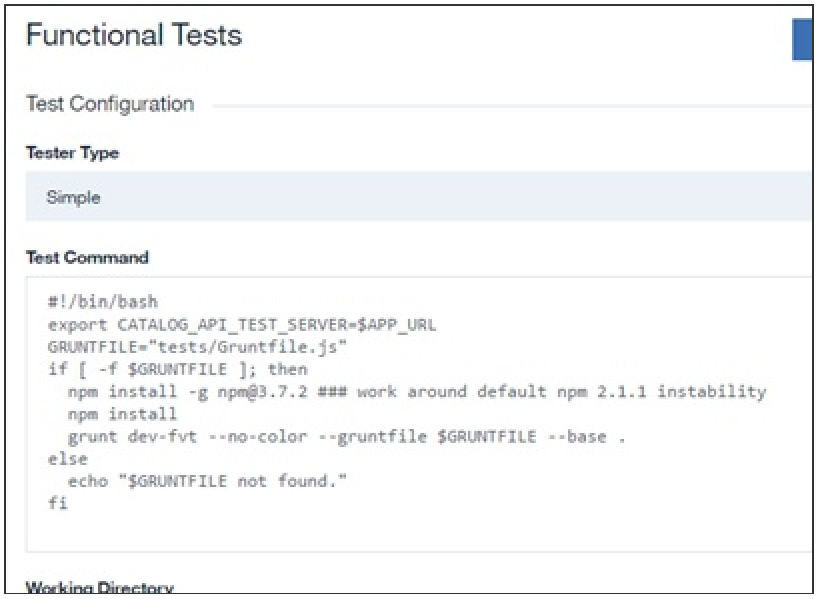
10. Click the **JOBS** tab and add a new job of type **Test**.
11. Change the jobs name from *Tests* to **Functional Tests**.
12. Note that the *Tester Type* is *Simple*.
13. Enter the following code to the *Test Command* section. *Note:* You can enter the following URL into another browser tab to

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM

display the code for easy copy and pasting: http://ibm.biz/CatalogAPIDevFVT

```
#!/bin/bash
export CATALOG_API_TEST_SERVER=$APP_URL
GRUNTFILE="tests/Gruntfile.js"
if [ -f $GRUNTFILE ]; then
  npm install -g npm@3.7.2 ### work around default npm 2.1.1 instability
  npm install
  grunt dev-fvt --no-color --gruntfile $GRUNTFILE --base .
else
  echo "$GRUNTFILE not found."
fi
```

This bash shell runs the same functional test scripts on the catalog service but this time against the deployed application in the *dev* space.


Functional Tests

Test Configuration

Tester Type: Simple

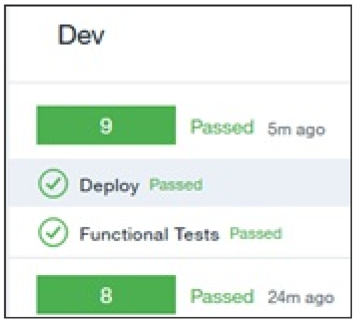
Test Command

```
#!/bin/bash
export CATALOG_API_TEST_SERVER=$APP_URL
GRUNTFILE="tests/Gruntfile.js"
if [ -f $GRUNTFILE ]; then
  npm install -g npm@3.7.2 ### work around default npm 2.1.1 instability
  npm install
  grunt dev-fvt --no-color --gruntfile $GRUNTFILE --base .
else
  echo "$GRUNTFILE not found."
fi
```

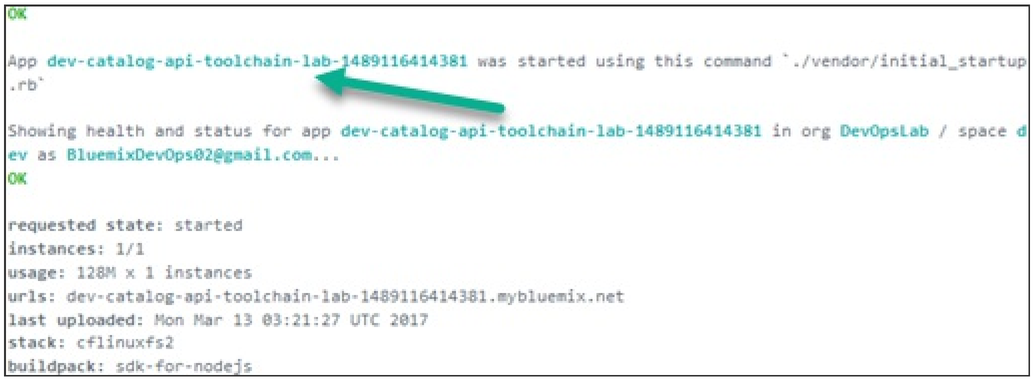
Working Directory

14. Click **Save** to save the *Dev* stage.
15. The *Delivery Pipeline* displays the *Build* and *Dev* stages. The *Dev* stage has not been run. Click on the **Run Stage** icon (the right arrow in the *Dev* stage) to run the *Dev* stage, deploying the Catalog application to the *dev* space and executing the functional tests.
16. The *JOBS* section shows the Stage was successful. Click on "View logs and history" to the Stage history.
17. Stage History displays the execution history of the stage in reverse chronological order (so the most recent on top and the oldest at the bottom). Within the history of a stage execution, the job history is displayed in the order in which the job was attempted. For example, the following screen shot shows that this stage was executed twice. Within the most recent execution (*9*), the *Deploy* job was attempted (and passed) followed by the *Functional Tests* job (which also passed). Your screen will probably just have **1** attempt.

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM

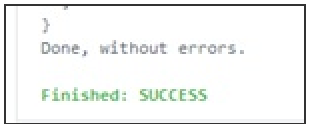


18. This display shows that the *Dev* stage ran both jobs and they both passed. Initially, the log for the *Deploy* job is displayed. Scrolling to the bottom and you see the application was deployed as *catalog-api-toolchain-lab-timestamp* into the *dev* space.



```
OK

App dev-catalog-api-toolchain-lab-1489116414381 was started using this command `./vendor/initial_startup
.rb`

Showing health and status for app dev-catalog-api-toolchain-lab-1489116414381 in org DevOpsLab / space d
ev as BluemixDevOps02@gmail.com...
OK

requested state: started
instances: 1/1
usage: 128M x 1 instances
urls: dev-catalog-api-toolchain-lab-1489116414381.mybluemix.net
last uploaded: Mon Mar 13 03:21:27 UTC 2017
stack: cflinuxfs2
buildpack: sdk-for-nodejs
```

19. Scroll back to the top and click the **Test** job to display the log for it. Scroll to the bottom.



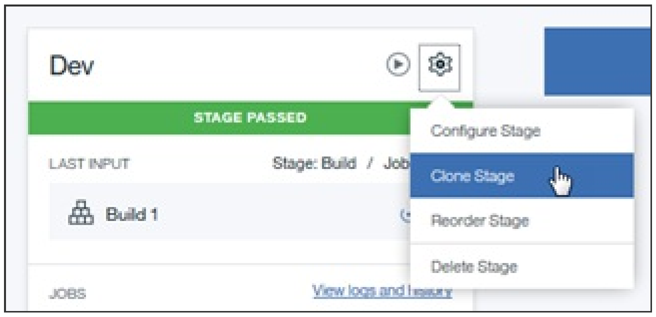
```
}
Done, without errors.

Finished: SUCCESS
```

20. Return to the Delivery Pipeline.

Task 7: Add Test stage to Catalog Delivery Pipeline

Now we add the *Test* stage and associated jobs. The *Test* stage has two jobs. The first job deploys the just built Catalog API microservice and deploys it into the *qa* space on Bluemix and the second job performs some automated tests on the deployed microservice. This time, we will clone the *Dev* stage and make some modifications.

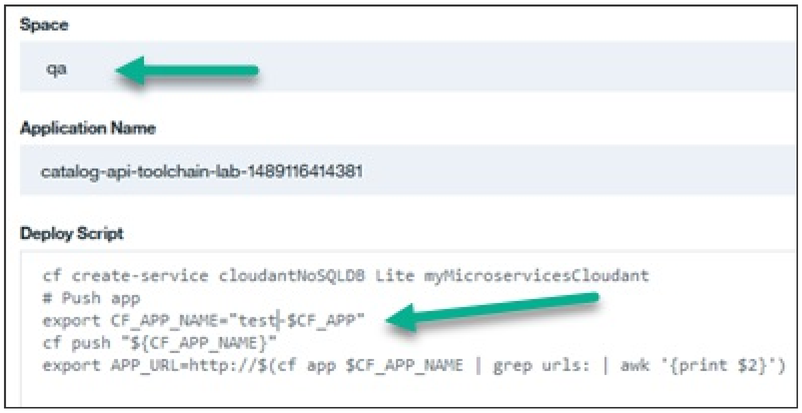
1. On the *Dev* stage, click the *Configure Stage* gear icon and select **Clone Stage**.



2. Rename the cloned stage to **Test** from *Dev [copy]*.
3. On the *Jobs* tab, for the *Deploy* job, change the space to **qa** from *dev*..

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM

4. Change the *Deploy* deploy script so CF_APP_NAME gets set to **"test-$CF_APP"** from *"dev-$CF_APP"*.



```
cf create-service cloudantNoSQLDB Lite myMicroservicesCloudant
# Push app
export CF_APP_NAME="test-$CF_APP"
cf push "${CF_APP_NAME}"
export APP_URL=http://$(cf app $CF_APP_NAME | grep urls: | awk '{print $2}')
```

5. Switch to the *Functional Test* job.
6. Change the *Test Command* to:

```
#!/bin/bash
# invoke tests here
echo "Testing of App Name ${CF_APP_NAME} was successful"
```

   This 'test' script just echos the app name to the console log. In a real environment, we would execute automated test tools and scripts to validate the deployed service still worked.
7. Click **Save** to save the *Test* stage.
8. The Delivery Pipeline displays the *Build*, *Dev* and *Test* stages. The *Test* stage has not been run. Click on the **Run Stage** icon to run the *Test* stage and deploy the order API to the *test* space.
9. As before for the *Dev* stage, the **JOBS** section of the *Test* stage shows the *Deploy* and *Functional Tests* jobs were successful. Click **Functional Tests** to display the log for the *Functional Tests* job. Notice the "Testing of App Name" message was echoed to the log.
10. Return to the Delivery Pipeline. Click on the application URL (http://test-catalog-api-toolchain-lab-*timestamp*.mybluemix.net/) to access the running application. Note that *test-* was added to the start of the application name.
11. Close the application browser window.
12. On the Delivery Pipeline display, click on the **View runtime log** link to examine the log for this runtime. Return to the Delivery Pipeline.
13. The **Test** stage has been successfully added and executed.

Task 8: Add Prod stage to Catalog Delivery Pipeline

Now we will add the final stage to the Delivery Pipeline, the *Prod* stage. This stage has one job, which performs a *Blue-green* deployment to the *prod* space. As you may remember from the simple Order lab, a blue-green deployment is a release technique reducing downtime and risk by running two identical production environments called Blue and Green. At any time, only one of the environments is live, with the live environment serving all production traffic.

So during deployment, this stage will check to see there is an earlier instance of this application running and if it is, keep it around in case the deploy of the new version of the app has problems. If the new version deploys successfully, the old version is deleted. If not, the new version is deleted and the old version continues to run. To do this, we will clone the *Dev* stage and make some modifications.

1. Ensure the catalog-api-toolchain-lab *catalog-api-toolchain-lab-timestamp* Delivery Pipeline is displayed.
2. Clone the *Dev* stage.

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM

3. Rename the cloned stage to **Prod** (from *Dev [copy]*).
4. On the *Jobs* tab, change the Deploy Job name to **Blue/Green Deploy** and change the space from *dev* to **prod**
5. Change the deploy script to the following *HINT*: It is very similar to the script we used for the Order Pipeline lab, perhaps you configure that Job to copy and paste the deploy script *or* you can enter the following URL into another browser tab to display the code for easy copy and pasting: http://ibm.biz/CatalogAPIProdBlueGreenDeploy

```
#!/bin/bash
echo "Attempting to create cloudantNoSQLDB Lite myMicroservicesCloudant for use by the
microservices. It is not a problem if it already exists, we simply continue."
cf create-service cloudantNoSQLDB Lite myMicroservicesCloudant
export CF_APP_NAME="prod-$CF_APP"
# Push app
echo "If the $CF_APP_NAME does not exist, push the app."
if ! cf app $CF_APP_NAME; then
  cf push $CF_APP_NAME
else
  OLD_CF_APP_NAME=${CF_APP_NAME}-OLD-$(date +"%s")
  rollback() {
    set +e
    if cf app $OLD_CF_APP_NAME; then
      cf logs $CF_APP_NAME --recent
      cf delete $CF_APP_NAME -f
      cf rename $OLD_CF_APP_NAME $CF_APP_NAME
    fi
    exit 1
  }
  set -e
  trap rollback ERR
  echo "If the $CF_APP_NAME does exist, rename it."
  cf rename $CF_APP_NAME $OLD_CF_APP_NAME
  echo "And push out the new version."
  cf push $CF_APP_NAME
  echo "If the push is successful, delete the old app."
  cf delete $OLD_CF_APP_NAME -f
fi
# Export app name and URL for use in later Pipeline jobs
# export CF_APP_NAME="$CF_APP"
export APP_URL=http://$(cf app $CF_APP_NAME | grep urls: | awk '{print $2}')
# View logs
#cf logs "${CF_APP}" --recent
```

6. Click **Save** to save the *Prod* stage.
7. Click on **Run Stage** to run the *Prod* stage and deploy the Catalog API app to *prod* space.
8. The JOBS section of the *Blue/Green Deploy* shows the Deploy was successful. Inspect the *Blue/Green Deploy* Job log to see where the app was deployed and the *Functional Tests* Job log to ensure the tests were successful.
9. Return to the Delivery Pipeline.
10. Click the application URL in the *Prod* stage to access the running application. Note that *prod-* was added to the start of the application name. Where was that changed?
11. Close the application browser window. The **Prod** stage has been successfully added and executed, deploying the application to the *prod* space.
12. Click on the left arrow to the left of *Toolchain* to return to the _catalog-toolchain-*timestamp* page.
13. Click on the left arrow to the left of *Toolchains* to return to the _Toolchains page.

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM

Lab 4: Create UI Toolchain from Deployed App

Objective

This lab manually deploys the UI microservice, creates a simple Toolchain from the deployed application and then configures the Toolchain. It assumes that the *DevOpsLabs* Organization and *dev*, *qa* and *prod* Spaces are already created.

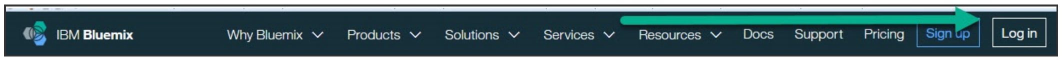
Tasks:

- Task 1: Deploy UI Application
- Task 2: Create Toolchain
- Task 3: Build Application

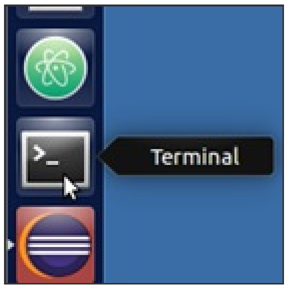
Throughout the lab, the phrase *timestamp* is used to indicate the same timestamp string that was appended to *simple-order-toolchain*. While a timestamp string is not required, it does help make the name of the created objects unique.

Task 1: Deploy UI Application

1. If you are not already logged into IBM Bluemix, log into IBM Bluemix (https://www.ibm.com/cloud-computing/bluemix/).



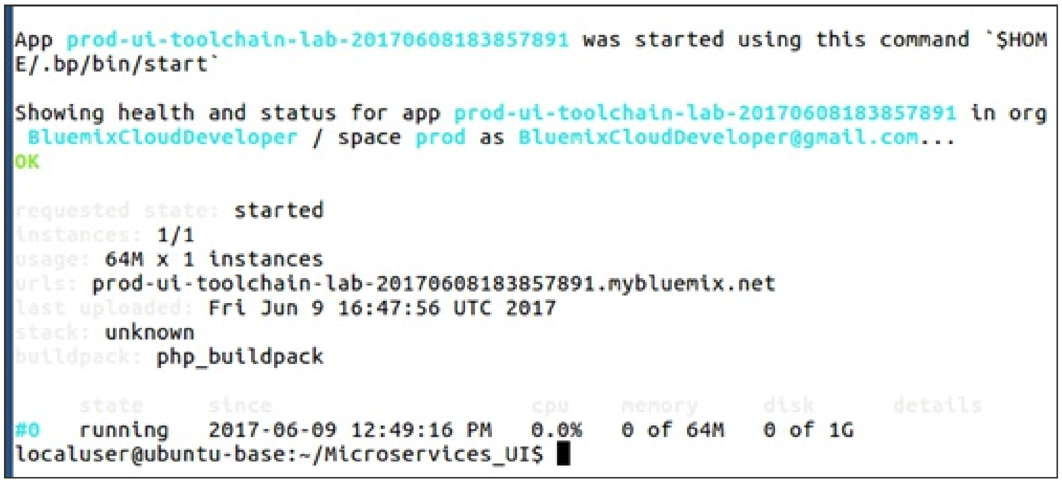
2. If you are using the Ubuntu VMware image, open a terminal Window by selecting **Terminal** from the Launcher.



If you are using your own machine, open up a command prompt.

3. The sample code that you will be using is in a github repository. We will clone (make a copy) onto our local machine, In a terminal window, enter the following command: `git clone https://github.com/open-toolchain/Microservices_UI`
4. Change into the just created directory. `cd Microservices_UI`
5. If you are curious, you can enter the ls command to see the files.
6. Login to Bluemix from the command line by entering the following command: `bx login -a https://api.ng.bluemix.net -u userid@domain.com -o _org_name_ -s prod`
7. Push the application to Bluemix with the following command: `bx app push prod-ui-toolchain-lab-<i>timestamp</i>`

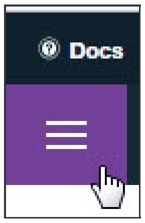
© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM



```
App prod-ui-toolchain-lab-20170608183857891 was started using this command `$HOM
E/.bp/bin/start`

Showing health and status for app prod-ui-toolchain-lab-20170608183857891 in org
 BluemixCloudDeveloper / space prod as BluemixCloudDeveloper@gmail.com...
OK

requested state: started
instances: 1/1
usage: 64M x 1 instances
urls: prod-ui-toolchain-lab-20170608183857891.mybluemix.net
last uploaded: Fri Jun 9 16:47:56 UTC 2017
stack: unknown
buildpack: php_buildpack

     state     since                    cpu    memory    disk      details
#0   running   2017-06-09 12:49:16 PM   0.0%   0 of 64M   0 of 1G
localuser@ubuntu-base:~/Microservices_UI$
```

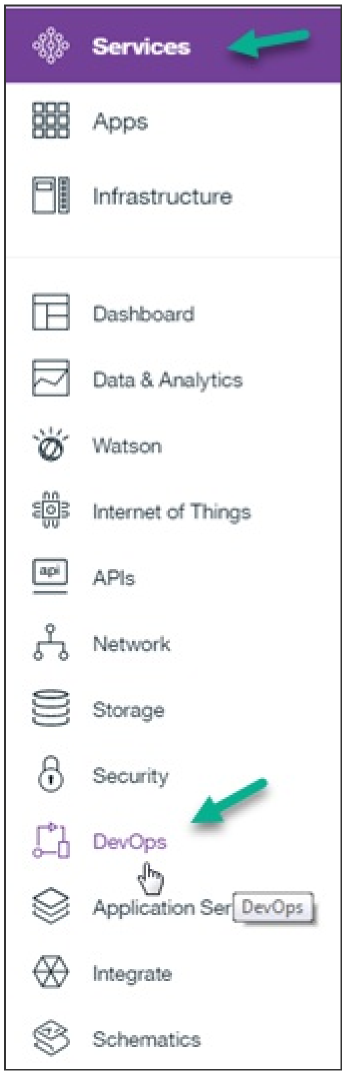
Task 2: Create Toolchain

1. Return to the Bluemix console.
2. If you are not on the Toolchains page (if you don't see a button called *Create a Toolchain*), click on the **Bluemix menu bar** in the upper left corner.



   and click on **Services** then **DevOps**

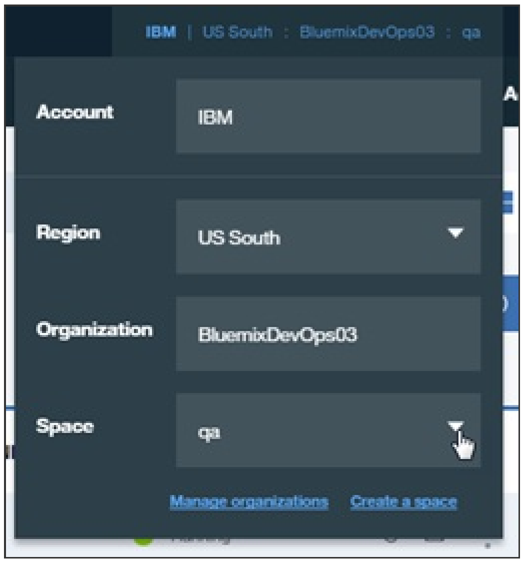
© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM



and click on **Toolchains**.

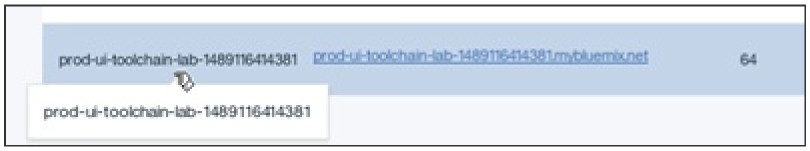
3. Click **Create a Toolchain**.
4. Click on the link **Create a toolchain from an application**. It is on the left of the screen.
5. If the *prod* space is not displayed in the upper right hand corner, click the upper-right hand corner account settings and select **prod** as the *Space*.

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM


IBM | US South : BluemixDevOps03 : qa

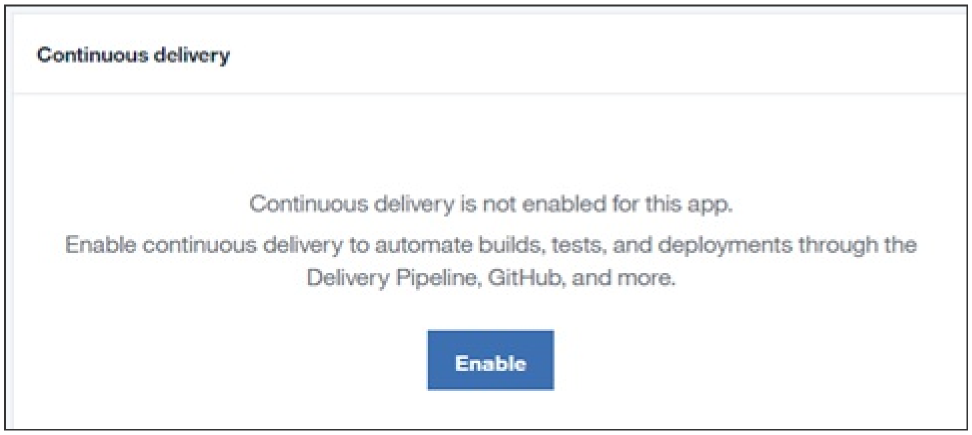
Account	IBM
Region	US South
Organization	BluemixDevOps03
Space	qa

6. Click on the application to display the application dashboard.
7. Click on your (just created) application to display the application overview page.


prod-ui-toolchain-lab-1489116414381 prod-ui-toolchain-lab-1489116414381.mybluemix.net 64

prod-ui-toolchain-lab-1489116414381

8. Click on the **Enable** button for *Continuous Delivery*.

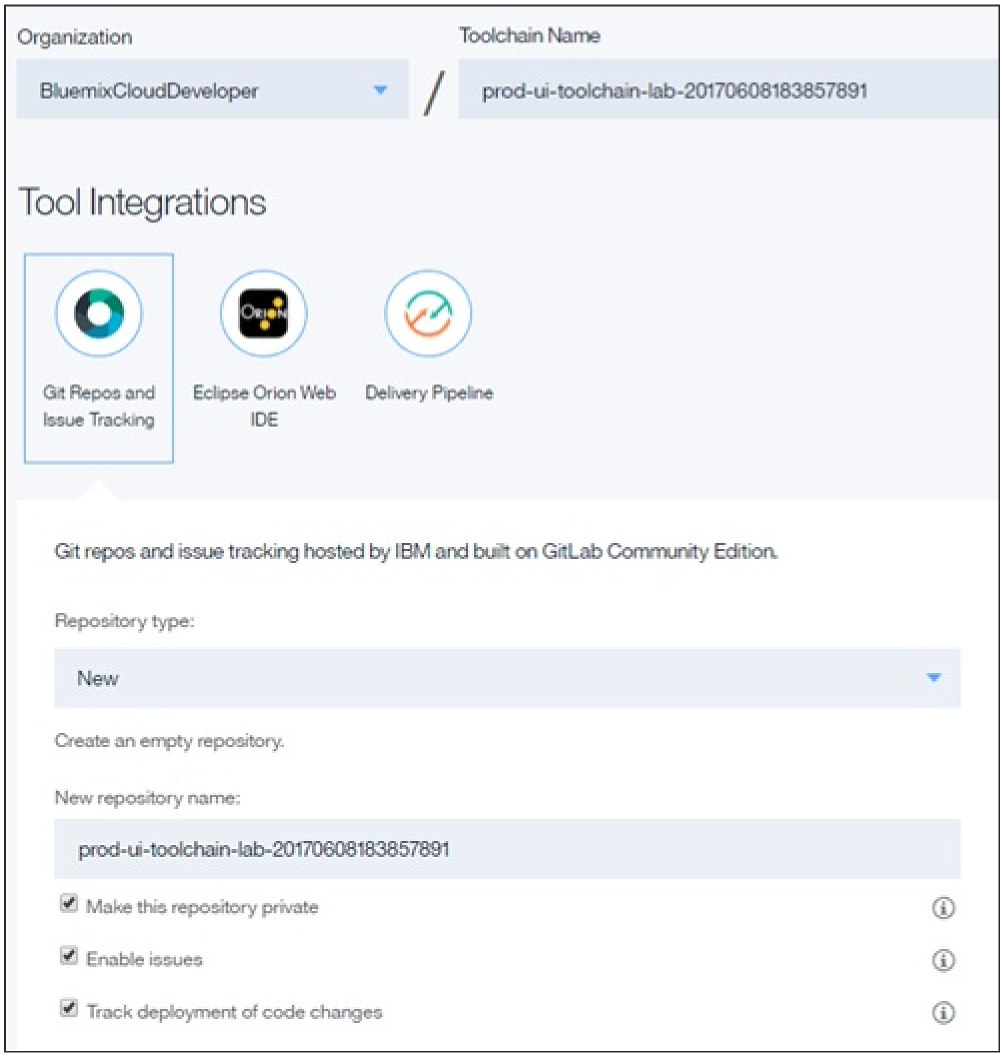

Continuous delivery

Continuous delivery is not enabled for this app.

Enable continuous delivery to automate builds, tests, and deployments through the Delivery Pipeline, GitHub, and more.

9. The Continuous Delivery Toolchain creation page is displayed and pre-filled with information about the application.

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM



Organization

BluemixCloudDeveloper / Toolchain Name: prod-ui-toolchain-lab-20170608183857891

Tool Integrations

Git repos and issue tracking hosted by IBM and built on GitLab Community Edition.

Repository type: New

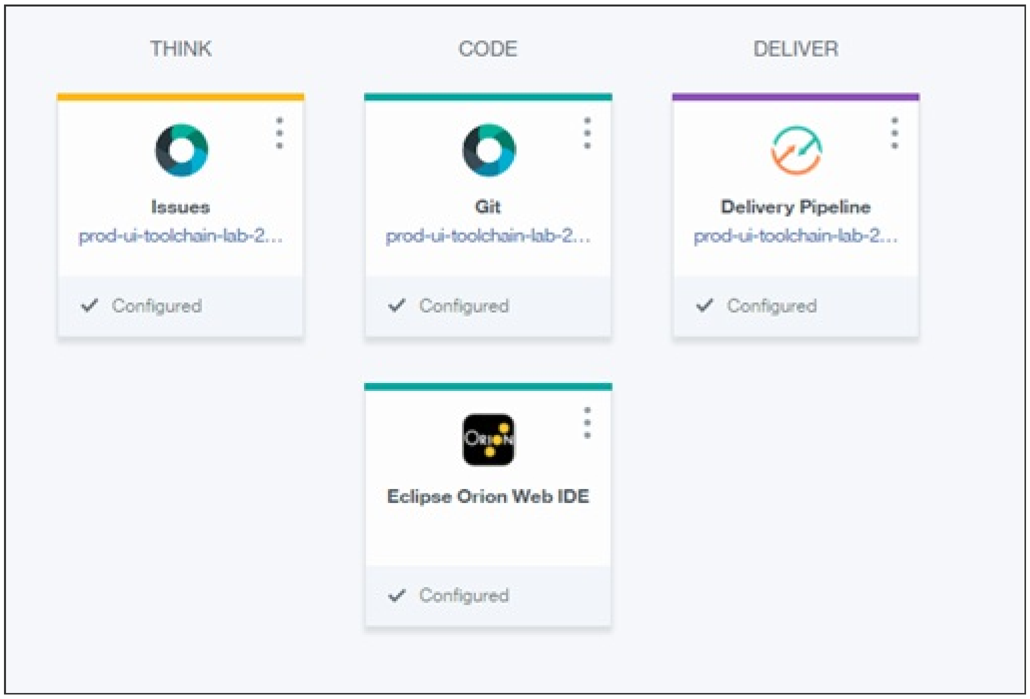
Create an empty repository.

New repository name: prod-ui-toolchain-lab-20170608183857891

- [x] Make this repository private
- [x] Enable issues
- [x] Track deployment of code changes

10. We need to clone the Git repo (remember, earlier from the command line we created a local cloned copy on the local machine). The pre-built integration is to *Git Repos and Issue Tracking*, the IBM hosted repos and issue tracking based on GitLab.
11. Change *Repository type:* to **Clone**.
12. Enter **https://github.com/open-toolchain/Microservices_UI** as the *Source repository URL*.
13. Click **Create**. The Toolchain is created.

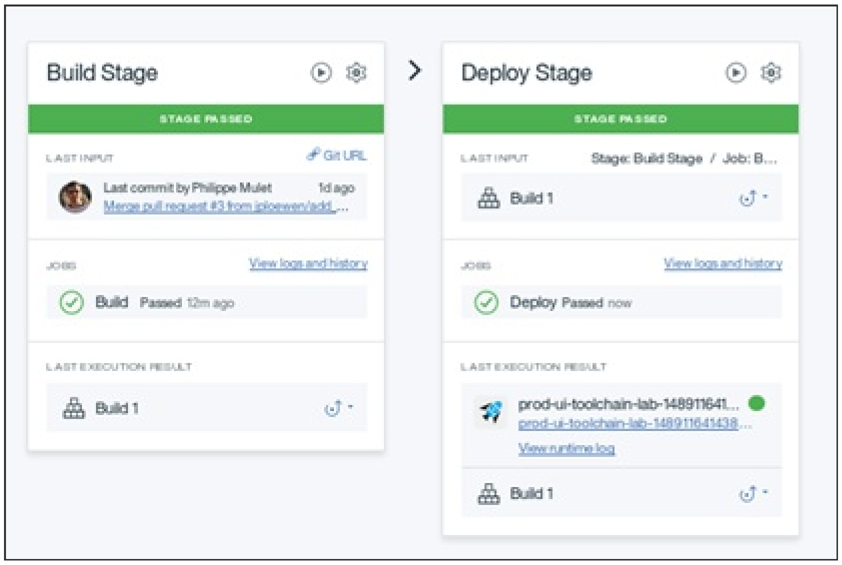
© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM



Note the *Issues* and *Git* icons are different then before as *Git Repos and Issue Tracking* is being used.

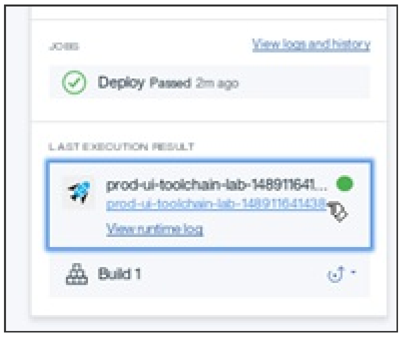
Task 3: Build Application

1. On the Bluemix *Create a Toolchain* page, click the blue arrow to the left of *Toolchains* to return to the Toolchains.
2. Click the just created Toolchain *(prod-ui-toolchain-lab-timestamp)*.
3. Click on the *Delivery Pipeline* tile.
4. Run the *Build Stage*.
5. The *Build* and *Deploy* stages complete (the *Deploy Stage* was started as a result of the *Build Stage* completing successfully).

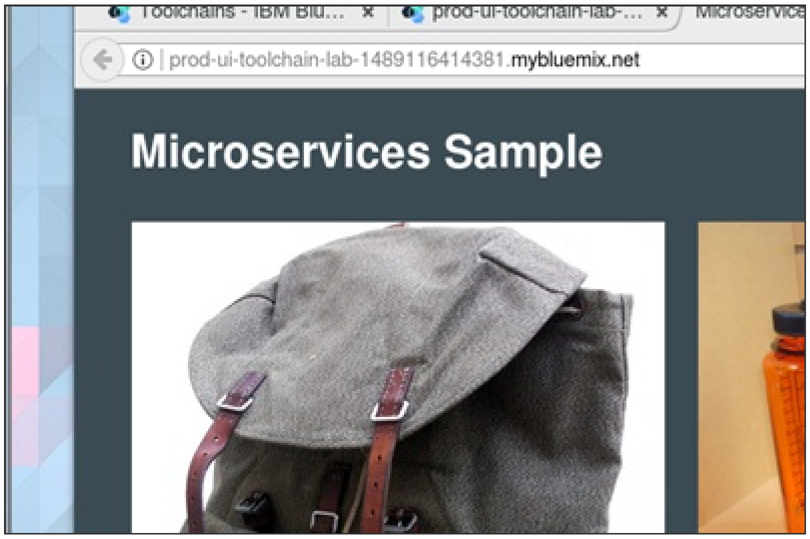


6. Click on the application URL.

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM



7. Assuming the microservices names match up, the application works. If not, don't worry, somewhere along the way the ***timestamp*** may have been mistyped.


Toolchains - IBM Blu...
prod-ui-toolchain-lab-...
Microservice
prod-ui-toolchain-lab-1489116414381.mybluemix.net
Microservices Sample

8. Close the application tab.
9. If you wish, you can add more jobs or stages.
10. On the *(prod-ui-toolchain-lab-timestamp)* page, click the blue arrow to the left of *Toolchains* to return to the Toolchains.

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM

Integrate Slack

Objective

Bluemix DevOps services include toolchains working together to support your development and deployment tasks. The integrations across a toolchain help ease the workflows between the tools so they work in synergy. This lab demonstrates how to set-up and use Bluemix toolchains with Slack to make your move to DevOps easier. **Note:** Toolchains are currently available in the US South region only and the instructions in this lab are written for the US South region.

This lab uses the simple toolchain for the Orders API microservice.

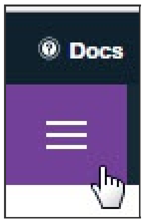
Teaming

Software development is a team activity. The lab scenario shows how Bluemix Continuous Delivery tool integrations can be used to alerts teams when activities, such as builds or deployments, occur.

Tasks

Task 1: Integrate Slack into Toolchain

The visual display of the Delivery Pipeline is great if you want to watch it. But if you don't want to watch or if you did not even start the build, how do you get notified of events that you are interested in? One way is Slack. Slack provides real-time messaging for team communications. You can integrate Slack with your Bluemix DevOps Services project so that notifications about build results from your Delivery Pipeline are posted on a Slack channel.

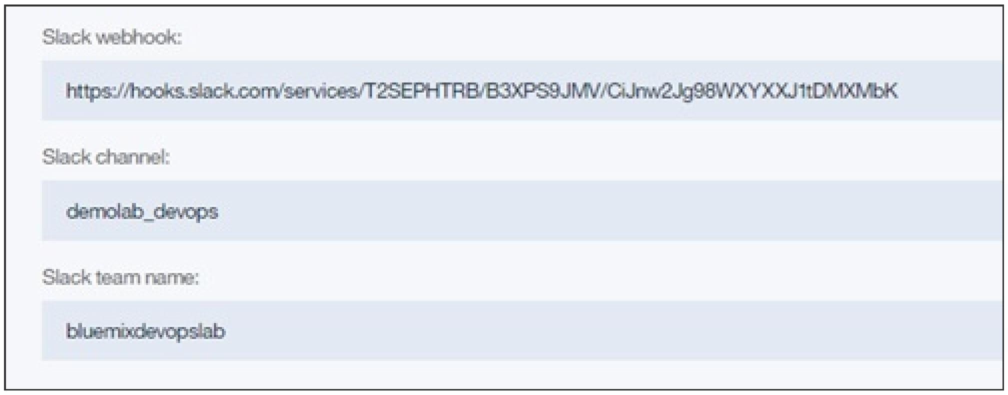
1. If you are not on the Toolchains page (if you don't see a button called *Create a Toolchain*), cick on the **Bluemix menu bar** in the upper left corner.
   

   and click on **Services** then **DevOps**.
2. Click **simple-order-toolchain-*timestamp***.
3. On the **simple-order-toolchain-*timestamp*** Toolchain page, click **Add a Tool**.
4. Select **Slack**
5. On the Slack Configuration page:
   - Enter the following all as one string as the Slack webhook. This is also available for copying and pasting at: http://ibm.biz/SlackWebHook.

```
https://hooks.slack.com/services/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
```

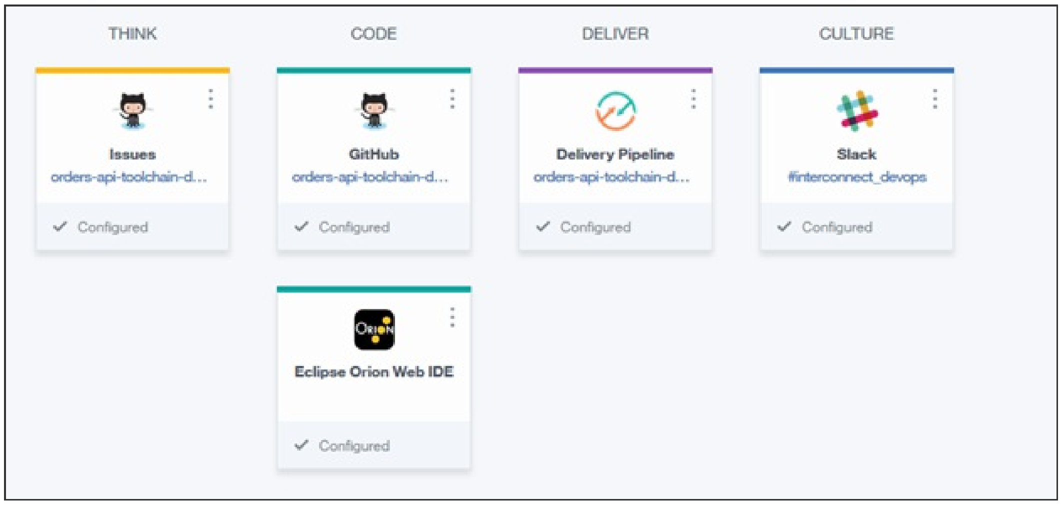
© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM

- Enter **#demolab_devops** as the Slack channel.
- Enter **BluemixDevOpsLab** as the Slack team name.



Slack webhook:
https://hooks.slack.com/services/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
Slack channel:
demolab_devops
Slack team name:
bluemixdevopslab

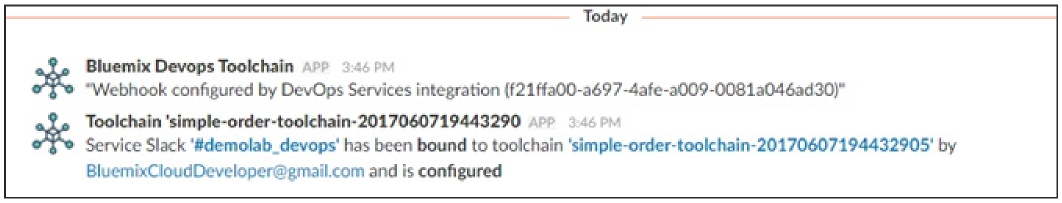
6. Click **Create Integration**
7. The integration with Slack is added.



8. Switch to the browser tab that has Slack open. If the tab is closed:

- Open a new browser tab
- Go to the following URL to go to the Slack team https://bluemixdevopslab.slack.com
- If asked, enter the following information:
    - Email address: **BluemixDevOps@gmail.com**
    - Password: **bluemix4me** and click **Sign in**.

9. Slack notifies us that the integration was added.



Today

Bluemix Devops Toolchain APP 3:46 PM
"Webhook configured by DevOps Services integration (f21ffa00-a697-4afe-a009-0081a046ad30)"

Toolchain 'simple-order-toolchain-20170607194432900' APP 3:46 PM
Service Slack **'#demolab_devops'** has been **bound** to toolchain **'simple-order-toolchain-20170607194432905'** by BluemixCloudDeveloper@gmail.com and is **configured**

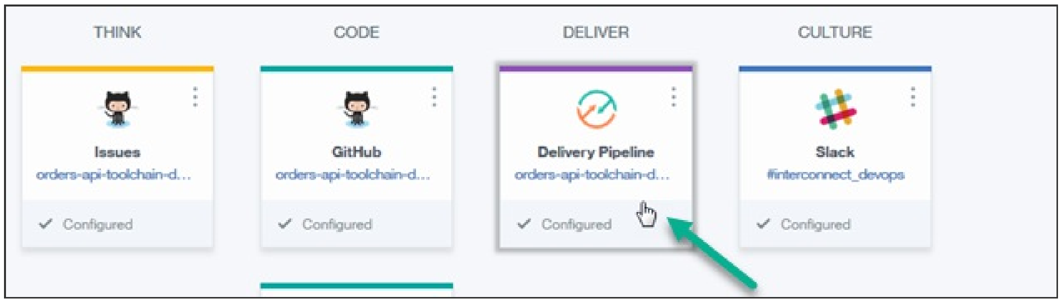
Task 2: Test Slack Integration

Looking at the *demolab_devops* Slack channel, we might see that other team members have been busy. Let's see what

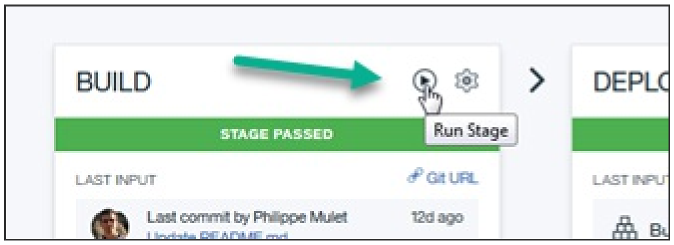
© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM

happens when we initiate a build.

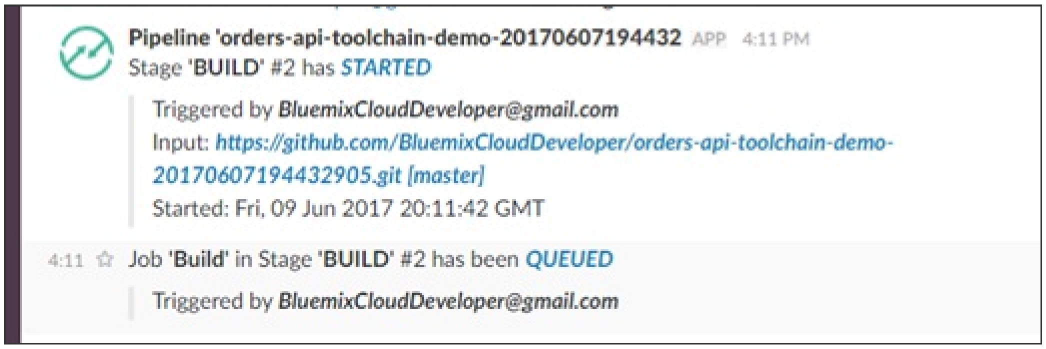
1. Switch to the browser tab displaying the Bluemix console. It should be displaying the **simple-order-toolchain-*timestamp*** Toolchain page. If not, click on the Bluemix menu bar in the upper left corner, click on Services then DevOps. Click on Toolchains. Then click on **simple-order-toolchain-*timestamp***.
2. Click on the **Delivery Pipeline** tile for the orders-api-toolchain-demo Delivery Pipeline.



3. Start an application build by clicking on the *BUILD* stage **Run Stage** icon.

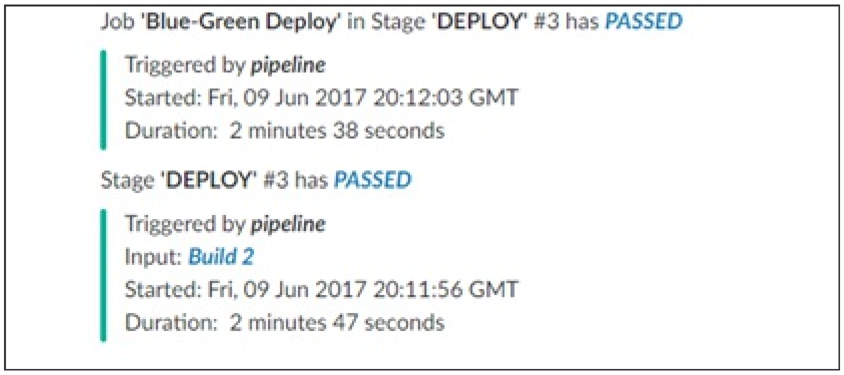


4. Switching to the Slack browser tab, the Slack bluemixdevopslab channel displays the progress of this activity.



Pipeline 'orders-api-toolchain-demo-20170607194432 APP 4:11 PM
Stage 'BUILD' #2 has *STARTED*
Triggered by *BluemixCloudDeveloper@gmail.com*
Input: *https://github.com/BluemixCloudDeveloper/orders-api-toolchain-demo-20170607194432905.git [master]*
Started: Fri, 09 Jun 2017 20:11:42 GMT
4:11 Job 'Build' in Stage 'BUILD' #2 has been *QUEUED*
Triggered by *BluemixCloudDeveloper@gmail.com*

5. And the completion.



Job 'Blue-Green Deploy' in Stage 'DEPLOY' #3 has *PASSED*
Triggered by *pipeline*
Started: Fri, 09 Jun 2017 20:12:03 GMT
Duration: 2 minutes 38 seconds
Stage 'DEPLOY' #3 has *PASSED*
Triggered by *pipeline*
Input: *Build 2*
Started: Fri, 09 Jun 2017 20:11:56 GMT
Duration: 2 minutes 47 seconds

6. Through the Bluemix Continuous Delivery integration with Slack, teams get notified of events in the Delivery Pipeline.
7. Switch to the browser tab displaying the Bluemix console. Click the blue arrow in the upper left hand corner to return to the

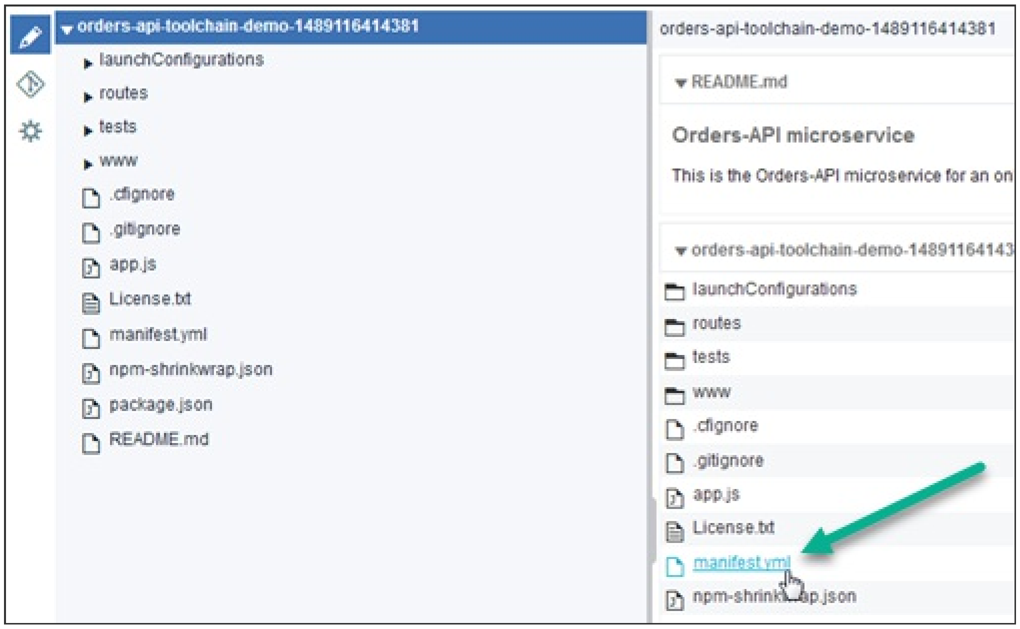
© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM

Toolchain.

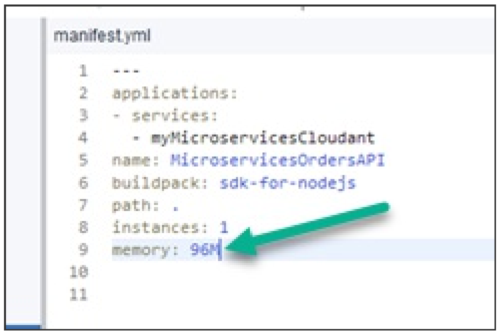
Task 3: Break the application build

Let's see what happens when the application build is broken.

1. On the **simple-order-toolchain-*timestamp*** Toolchain page, click the **Eclipse Orion Web IDE** tile.
2. In the file directory, click manifest.yml to open the file.


orders-api-toolchain-demo-1489116414381
launchConfigurations
routes
tests
www
.cfignore
.gitignore
app.js
License.txt
manifest.yml
npm-shrinkwrap.json
package.json
README.md
orders-api-toolchain-demo-1489116414381
README.md
Orders-API microservice
This is the Orders-API microservice for an on
orders-api-toolchain-demo-14891164143

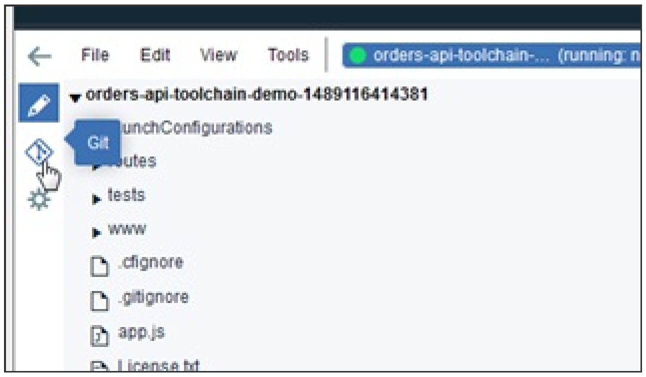
3. Update the value for memory to 96G. This setting intentionally increases your memory to exceed the quota for your org. Your changes are automatically saved.


manifest.yml

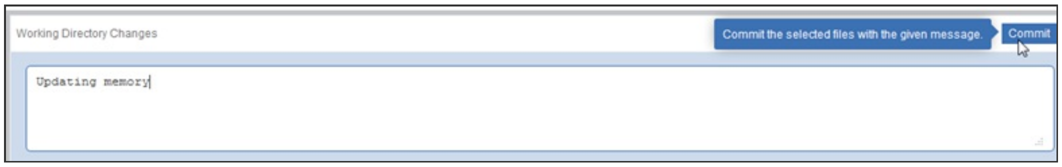
```
---
applications:
- services:
  - myMicroservicesCloudant
name: MicroservicesOrdersAPI
buildpack: sdk-for-nodejs
path: .
instances: 1
memory: 96M
```

4. From the Eclipse Orion Web IDE menu, click the **Git** icon.

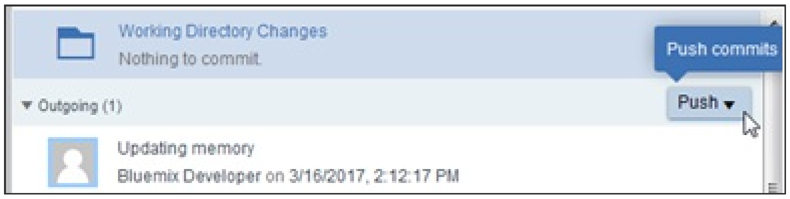
© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM



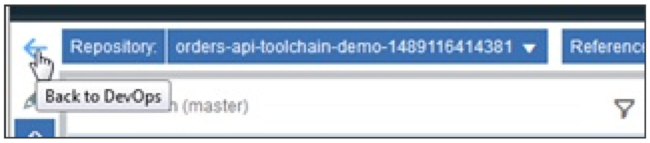
5. Enter a relevant comment and click **Commit** to push the changes in the local master branch.


Working Directory Changes
Commit the selected files with the given message.
Commit
Updating memory

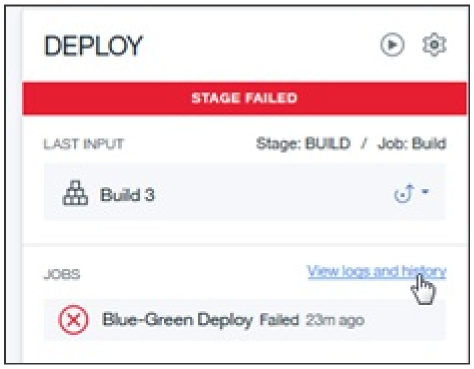
6. Click **Push** to put the changes into GitHub.


Working Directory Changes
Nothing to commit.
Push commits
Outgoing (1)
Push
Updating memory
Bluemix Developer on 3/16/2017, 2:12:17 PM

7. Your changes are automatically built and deployed in the pipeline.
8. Click the blue arrow in the upper left hand corner to return to the Toolchain.

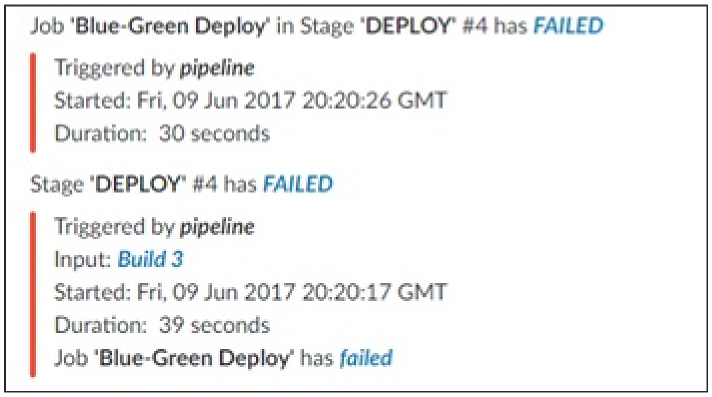


9. Click the tile for the *order-api-toolchain-labtimestamp* Delivery Pipeline. Our change caused the deployment to fail.



10. Switch to the Slack browser tab, Slack also notifies the team about the Deployment failure.

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM



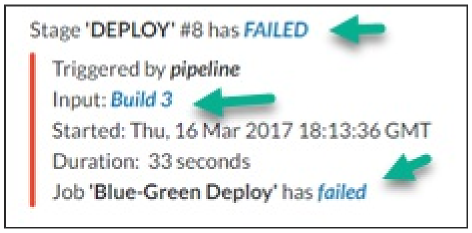
Job 'Blue-Green Deploy' in Stage 'DEPLOY' #4 has *FAILED*

Triggered by *pipeline*
Started: Fri, 09 Jun 2017 20:20:26 GMT
Duration: 30 seconds

Stage 'DEPLOY' #4 has *FAILED*

Triggered by *pipeline*
Input: *Build 3*
Started: Fri, 09 Jun 2017 20:20:17 GMT
Duration: 39 seconds
Job 'Blue-Green Deploy' has *failed*

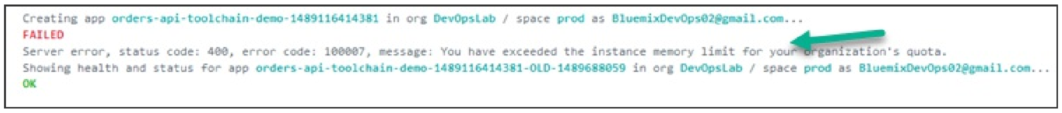
11. In Slack, there are a few links to go to the Stage, Job Input or Job. Click the **Failed** link for the Blue/Green Deploy job.



Stage 'DEPLOY' #8 has *FAILED*

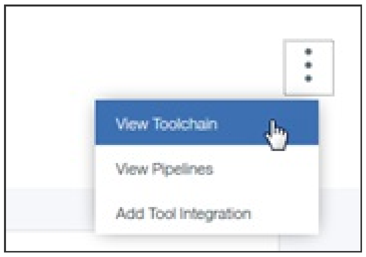
Triggered by *pipeline*
Input: *Build 3*
Started: Thu, 16 Mar 2017 18:13:36 GMT
Duration: 33 seconds
Job 'Blue-Green Deploy' has *failed*

12. Looking at the *Blue/Green Deploy* Job log (and scrolling down), we can see why it failed.



```
Creating app orders-api-toolchain-demo-1489116414381 in org DevOpsLab / space prod as BluemixDevOps02@gmail.com...
FAILED
Server error, status code: 400, error code: 100007, message: You have exceeded the instance memory limit for your organization's quota.
Showing health and status for app orders-api-toolchain-demo-1489116414381-OLD-1489688059 in org DevOpsLab / space prod as BluemixDevOps02@gmail.com...
OK
```

Return to the Toolchain by clicking on the vertical ellipsis on the upper right of the Bluemix console page and selecting **View Toolchain**.

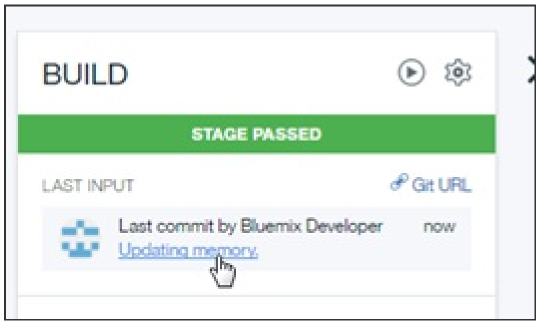


Task 4: Fix application

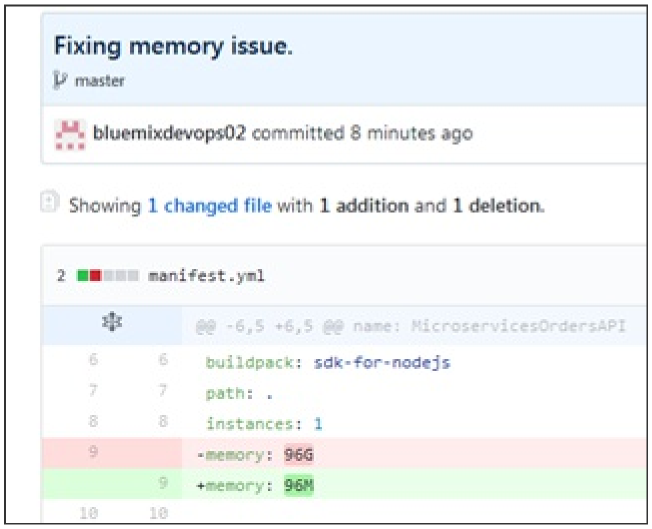
Now to fix the application.

1. On the **simple-order-toolchain-*timestamp*** Toolchain, click the **Eclipse Orion Web IDE** tile.
2. In the file directory, click manifest.yml to open the file.
3. Update the value for memory to 96M.
4. Now to Commit and Push the changes. From the Eclipse Orion Web IDE menu, click the **Git** icon.
5. Click **Commit** to put the changes in the local master branch (enter a relevant Commit comment).
6. Click **Push**. Your changes are automatically built and deployed in the pipeline.
7. Return to Toolchain page and click the Delivery Pipeline tile to watch the stages run in response to your code changes.
8. You could also display the Slack browser tab to display the Delivery Pipeline progress.
9. The deploy is successful and all the downstream stages run afterwards.
10. The Delivery Pipeline shows the Git URL and Git commit link.

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM



11. Click the Git commit link to show the changes that were committed that caused the Delivery Pipeline to execute.



Fixing memory issue.
master

bluemixdevops02 committed 8 minutes ago

Showing 1 changed file with 1 addition and 1 deletion.

```
2 manifest.yml
@@ -6,5 +6,5 @@ name: MicroservicesOrdersAPI
   buildpack: sdk-for-nodejs
   path: .
   instances: 1
-  memory: 96G
+  memory: 96M
```

12. Return to the Toolchains page.

© Copyright IBM Corp. 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM