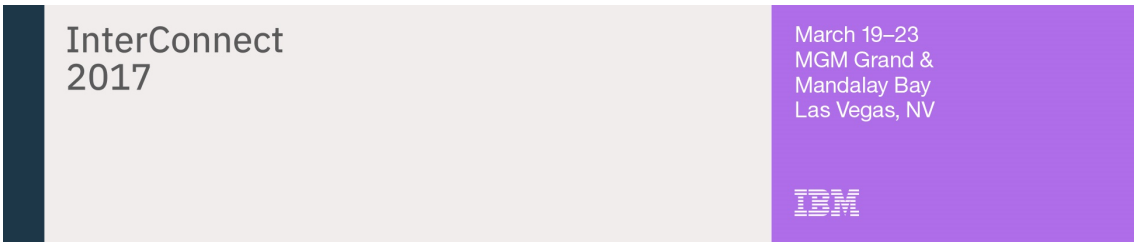


Bootcamp
Session 1414
DevOps and Cloud Service Management and Operations

Pam Geiger, Bluemix Enablement
Jim Palistrant, Bluemix Enablement

Creating Open Toolchains for IBM Bluemix

Objective

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

To create this toolchain, you use a sample application 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 is created.

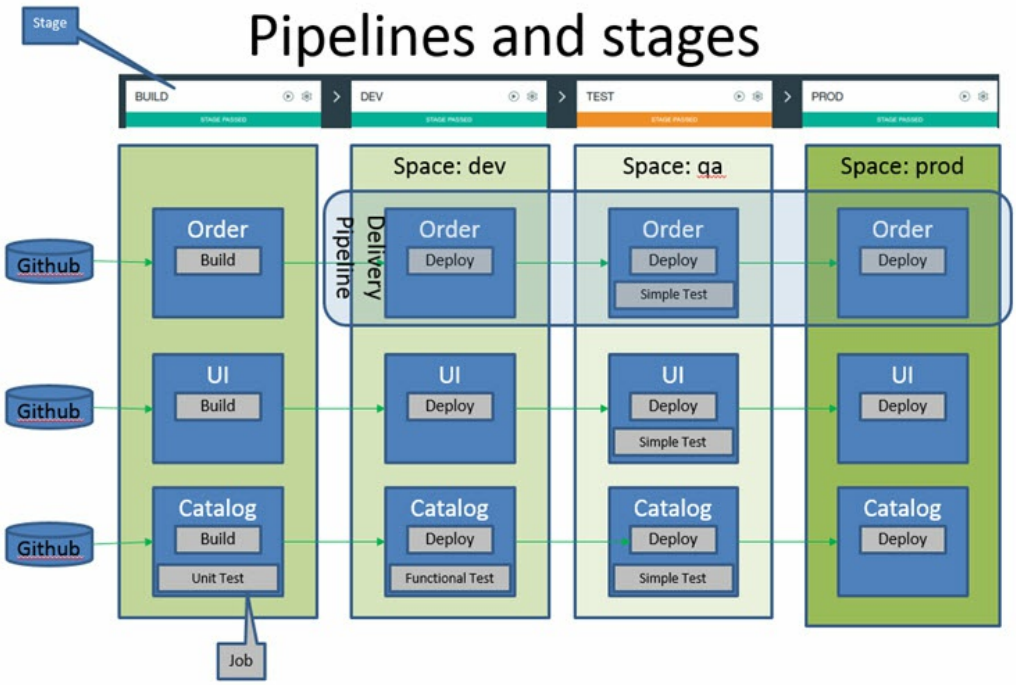
Pipelines, stages and deployment environments - oh my!

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

- The code for the online store is already in three different GitHub repositories, one per microservice. As part of creating the Continuous Delivery toolchain, you clone the repositories to your own GitHub account.
- Three delivery pipelines are also created, again one per microservice. Each pipeline can be run in parallel.
- Each delivery pipeline consists of a number of stages (Build, Dev, Test and Prod).
- Each stage will consist of one or more jobs that perform a task such as build the code, deploy the code, or test the code.
- As part of the respective delivery pipeline, each microservice is deployed to three environments: development, test, and production. You end up with nine deployed applications.
- While we could edit locally (using Eclipse for example) and push the code up to

Bluemix, we will instead use the Eclipse Orion Web IDE, which you can use to edit your code and deploy it with the pipeline from a web browser.

Conceptually, the process looks like:



Teaming

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

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

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

Prerequisites

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

Labs

Lab 0 Create Bluemix and GitHub accounts

Objective

This lab sets up the prerequisite accounts for the remaining labs.

Prerequisites

Prior to running these labs, you must have a Bluemix account, a GitHub account and access to a lab laptop. Follow the steps in this lab to create one or both of those accounts. Note you will need access to your eMail account to confirm the account setup activity.

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 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.

Lab 1 Create Toolchain for Sample Application

Objective

This lab takes you through the process of creating the Continuous Delivery toolchain for the sample online application.

Tasks:

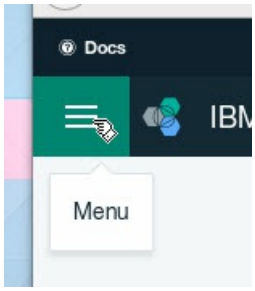
- Task 1: Log in to Bluemix
- Task 2: Display Microservices Toolchain panel
- Task 3: Understand Microservices Toolchain panel
- Task 4: Create Microservices Toolchain

Task 1: Log in to Bluemix

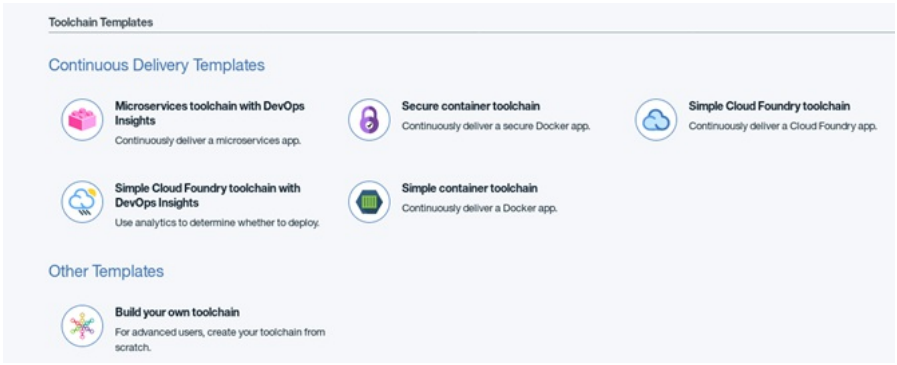
1. In a web browser, go to the Bluemix login page: https://console.ng.bluemix.net/
2. Login into Bluemix by entering your Bluemix account and password.

Task 2: Display Microservices Toolchain panel

1. We need to get to the DevOps Services. Click on the hamburger menu.



2. Click on **Services** then **DevOps**.
3. Click on **Toolchains**.
4. Click on **Create a Toolchain**.



Task 3: Understand Microservices Toolchain panel

1. There are a number of ways to create a Toolchain. You can use a provided template, create a new toolchain from scratch, or create a toolchain from an application. For this

exercise, you start with an existing template, the Microservices toolchain. Click **Microservices toolchain with DevOps Insights**.

2. The diagram that you see shows the completed toolchain. For an explanation of this toolchain, read the paragraphs on the left. This toolchain integrates various tools, some of which you will configure:

- GitHub
- Delivery pipeline
- Slack

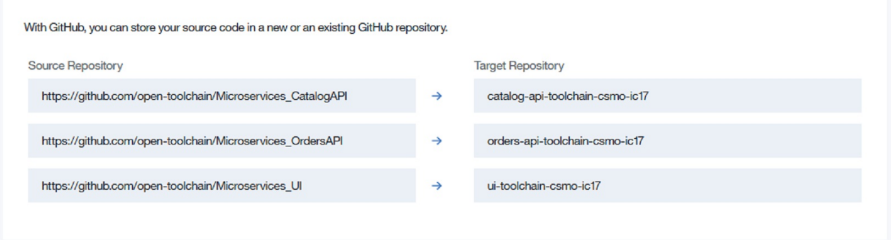
Some of which you will not configure:

- PagerDuty
- Sauce Labs

And some of which do not require configuration.

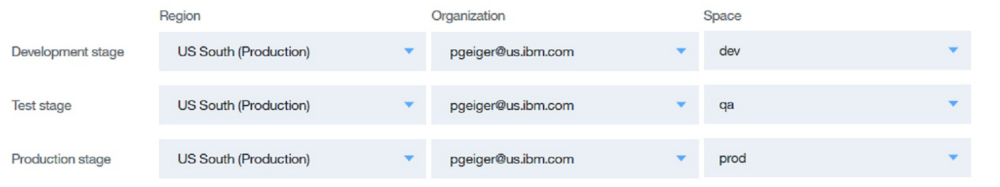
- DevOps Insights
- Eclipse Orion Web IDE

3. The **Organization** field is the name of the Bluemix Organization in which this toolchain will be created.
4. The **Toolchain Name** name is a generated name for this Toolchain. Rename it to something memorable (as you would in real life) or leave it at the default generated name. In the examples shown here, the toolchain name is **toolchain-csmo-ic17**.
5. The tools that make up this toolchain are shown under Tool Integrations. To configure an integration, click the tool to display the configuation information. Click **GitHub**. GitHub is where the source of the application is stored, one GitHub repo per application (so three GitHub repos). You set up your Toolchain to create a clone of each repo for use in this lab.
6. If you haven't authorized Bluemix to access GitHub, you need to:
    1. Click **Authorize** to go to the GitHub website.
    2. Enter your GitHub username and password.
    3. Click **Sign in**.
    4. Click **Authorize application**.
7. Once authorized, you see the three Source Repositories (one for each of Catalog, Orders and UI) where the code is stored and three corresponding Target Repositories, where the Source Repositories will be cloned. The Target Repository name is generated and just like Toolchain Name, you can leave the default generated name or make it something more memorable.



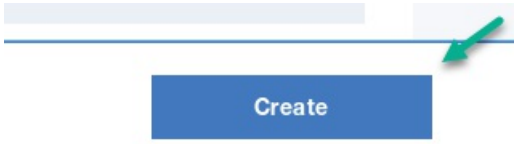
8. Click on **Delivery Pipeline**. You will be creating three delivery pipelines, one for each microservice. This is where the application name for each microservice will be specified, as well as the Bluemix Region, Organization and Space where the microservices will be deployed.
9. The application names for the three microservices must be unique in the Bluemix environment. If you used a generated name for the toolchaing, then you can leave them as generated. If you used the same name as in the examples you need to add something to make the names unique, such as your name or initials. In the example here, -lab is added to the names.

10. Pipelines can only be created in the US South region so to keep things simple you deploy to only the US South region and in the Organization we are logged into.
11. You have three spaces for the environment corresponding to the lifecycle.
    1. Development (**dev**) where code development takes place
    2. Testing or Quality Assurance (**qa**) where testing takes place
    3. Production (**prod**) where the application is available to end users (in the lab scenario, we do not restrict access to the dev or qa applications but in real life you would).

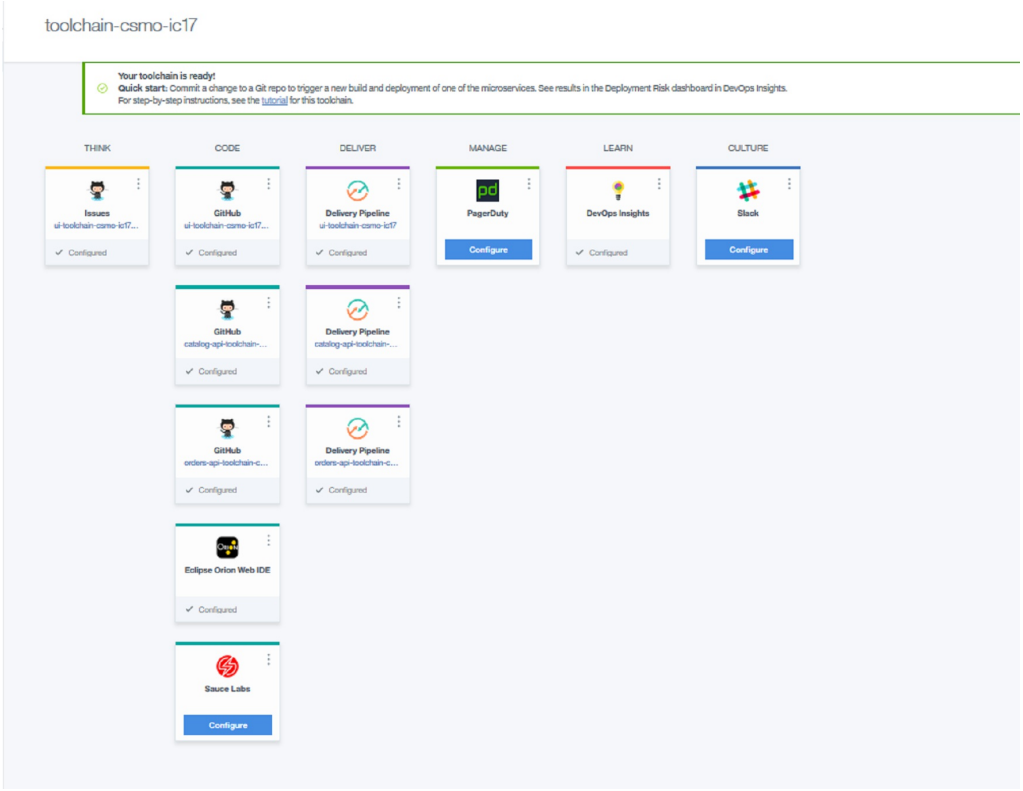


Task 4: Create Microservices Toolchain

1. Click **Create** to create the toolchain.



2. The Microservices toolchain is created. If it takes more then a few moments for the various tiles to display either Configured or Configure, refresh the browser.



Lab 2 Build and deploy to dev space

Objective

This lab will build all three microservices and deploy them to the Development space.

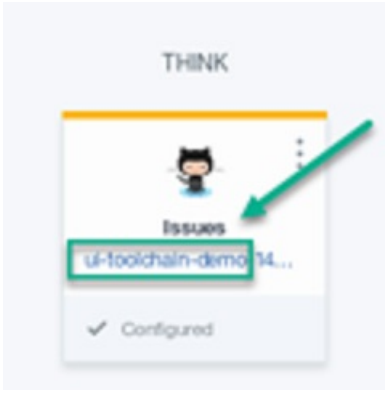
Tasks:

- Task 1: Explore the Microservices toolchain
- Task 2: Examine the delivery pipelines

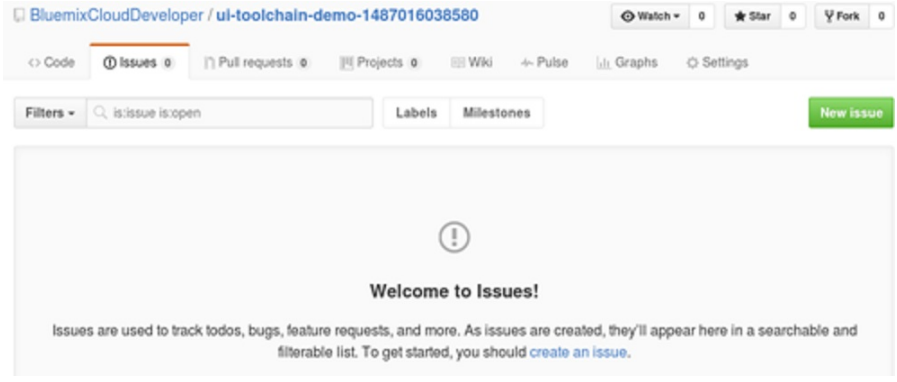
Task 1: Explore the Microservices toolchain

1. IBM Bluemix created the Continuous Delivery Toolchain based on the Microservices template. At the top you see the phases of the Bluemix Garage Method and where each tool fits in the method.

2. **Think** is where the GitHub Issues database is listed. Click on the **ui-toolchain-csmo-ic17** link (or right-mouse button click and select **Open Link in New Tab**, then select the new tab).



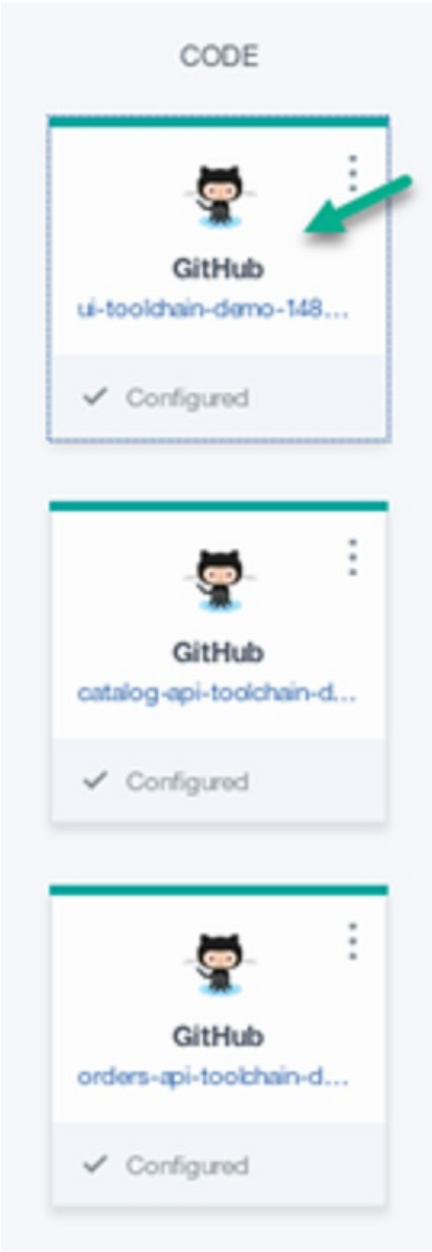
3. This displays the **GitHub Issues** page. Issues are used to track todos, bugs, feature requests, and more. Each GitHub repository (*repo* for short) can include issues. The Microservices template we used only included issues for the UI repo, but could be modified to add it to the others as well if desired.



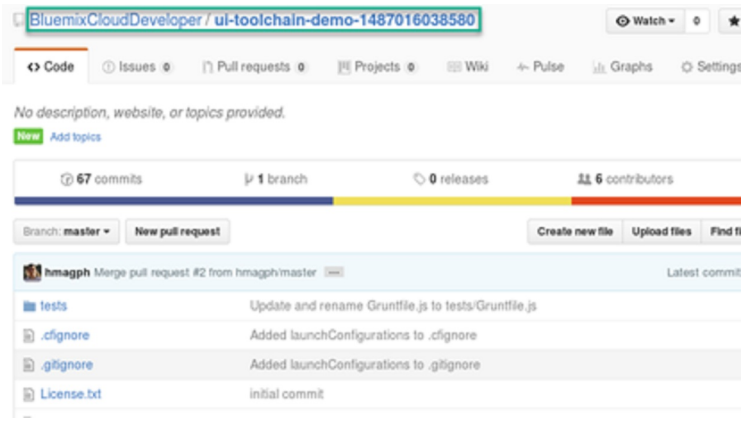
Return to the Microservices toolchain by either clicking on the **Go back one page**

arrow on the browser or, if you clicked the right-mouse button to open a new tab, close the GitHub Issues tab. (Note that the remainder of these lab instructions will 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, Sauce Labs and Eclipse Orion Web IDE are integrated.
    - Clicking on one of the three repos

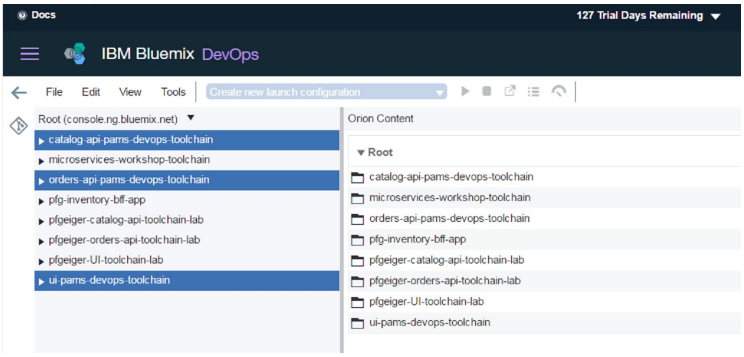


will display the respective (cloned) repo



The creation of the Toolchain did clone three repos in GitHub.

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

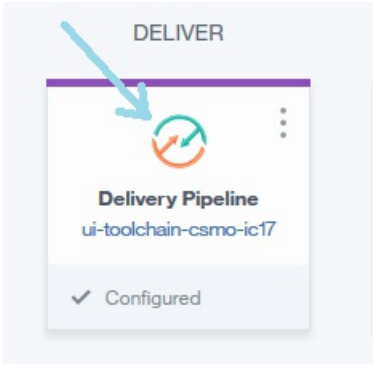


- We do not have a Sauce Labs account, so we really don't need the Sauce Labs integration. We will leave it alone for now.

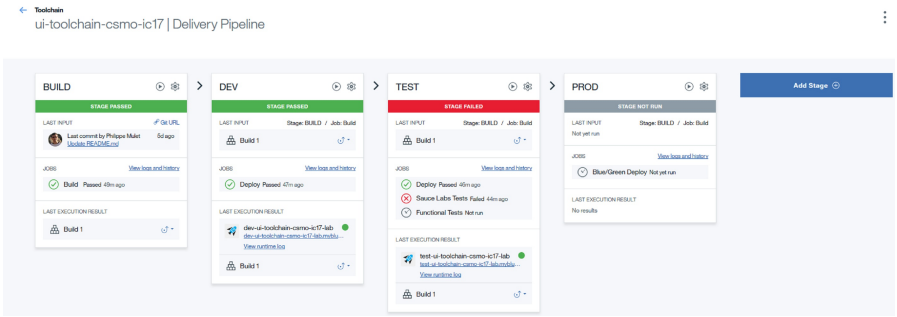
5. **Deliver** is where the code gets built, tested and deployed through the integrations of build pipelines, one per microservice. You explore build pipelines later.
6. **Manage** is where management tools, such as Pager Duty, get integrated.
7. **Learn** is where tools helping to drive application insight, such as DevOps Insights, get integrated.
8. **Culture** is where tools helping teams collaborate more effectively, such as Slack, get integrated.

Task 2: Examine the delivery pipelines

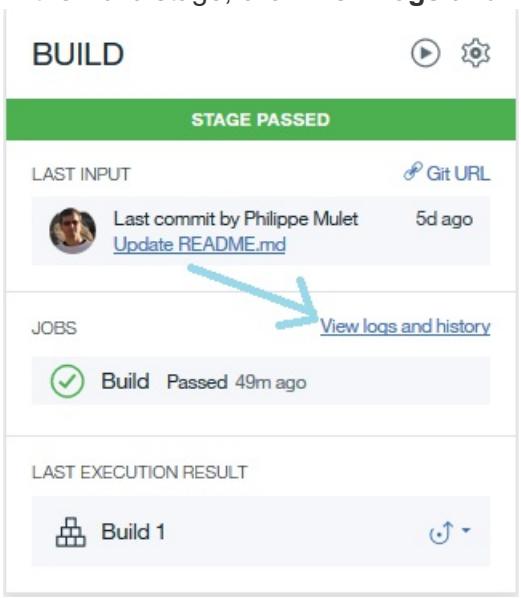
1. Click on the circle in the center of the *ui-toolchain-demo* Delivery Pipeline tile



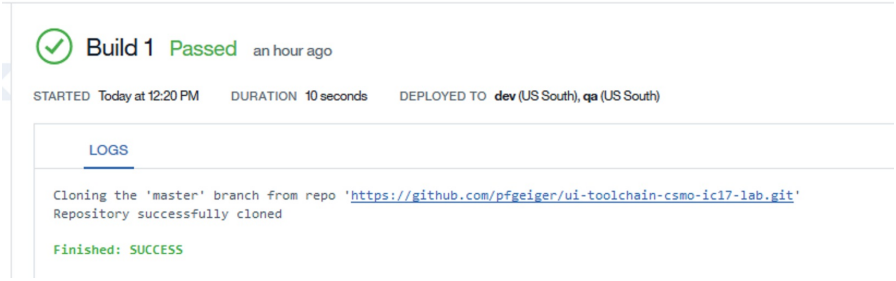
to display the UI delivery pipeline.



2. While you were busy exploring the toolchain, the various pipelines (remember, we have 3) started the build process. The UI delivery pipeline displays the status of each stage in the UI pipeline. The **Build** and **Dev** stage passed, while the **Test** stage failed. The **Prod** stage was not even attempted. If you look at the other 2 pipelines (catalog and orders) you would see similar results.
3. In the Build stage, click **View logs and history**



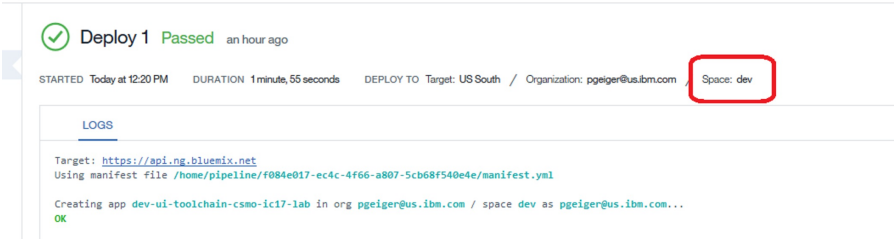
to display the commands and results of the Build stage, in this case simply cloning the repo.



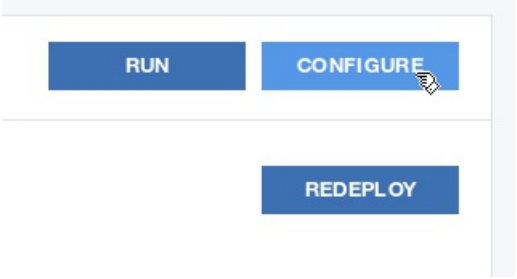
Click the arrow to the left of Pipeline

to return to the delivery pipeline.

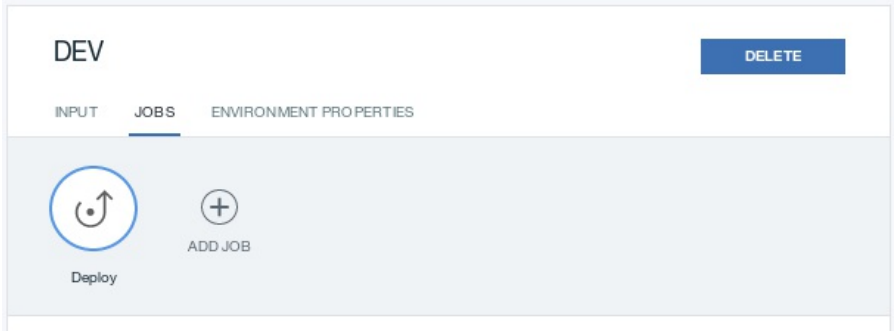
4. In the **Dev** stage, click **View logs and history** to display the commands and results of the Dev stage. This stage deployed the UI microservice to the dev space.



5. If you scroll through the log, you can see all the details of that job.
6. Click **Configure** in the upper right hand corner of the display



to display the Stage Configuration screen.

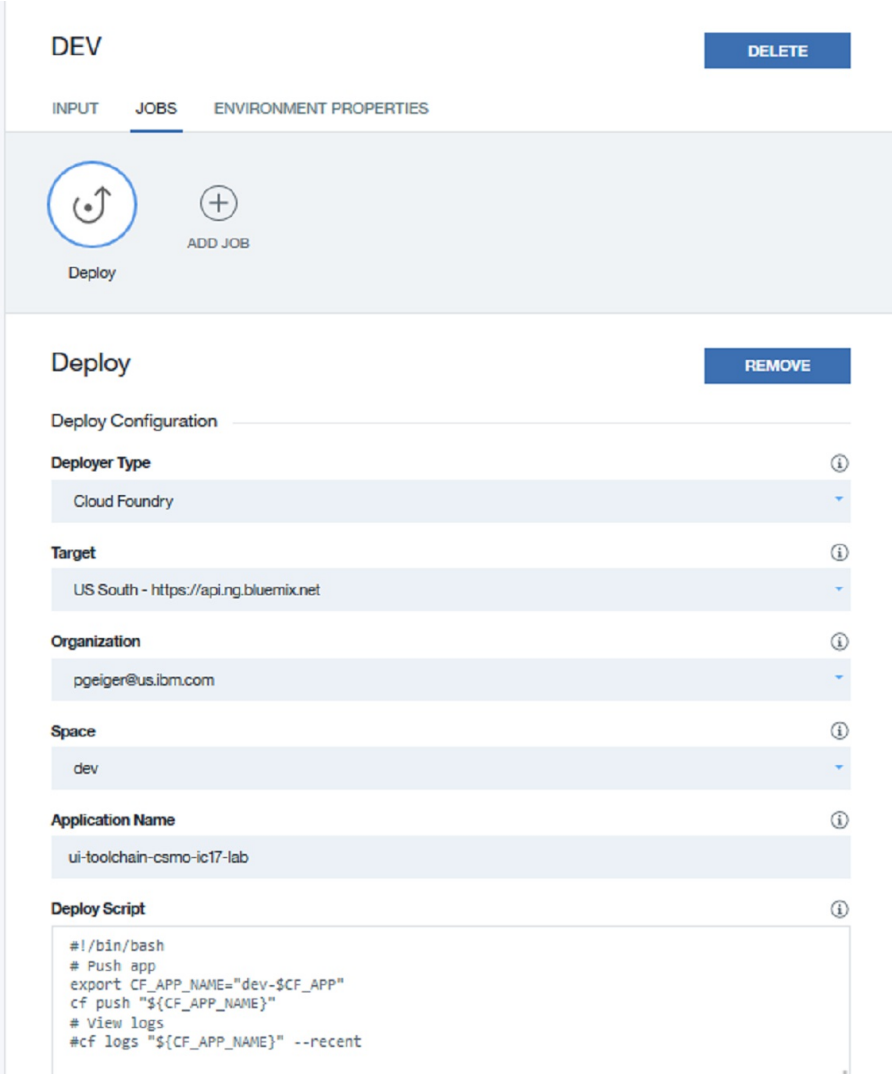


7. The Stage Configuration displays details about the stage. The *INPUT* tab displays the input to the stage, the *JOBS* tab displays the discrete jobs of the stage, and the

ENVIRONMENT PROPERTIES tab displays variables used by the jobs in the stage.

8. The UI Dev stage has just one job, the *Deploy* job. The job name can be changed by simply typing over the name. The job can be removed by clicking the **Remove** button and a new job can be added by clicking the **Add Job** button.

9. Details of each job are displayed when the job is selected.

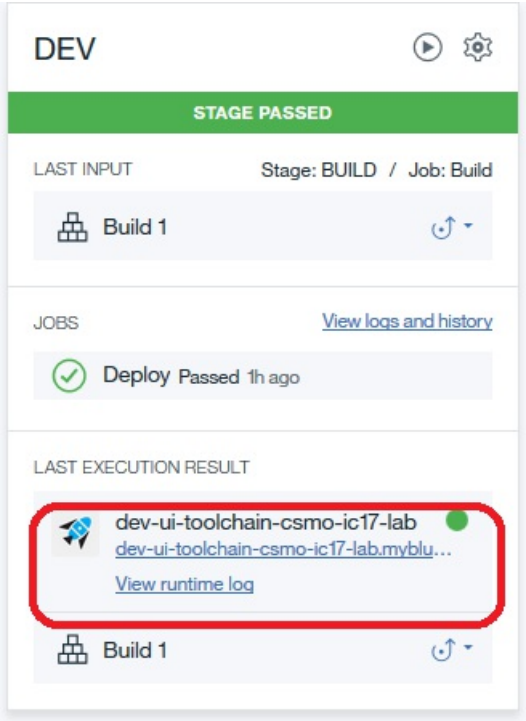


The lines in the Deploy Script:

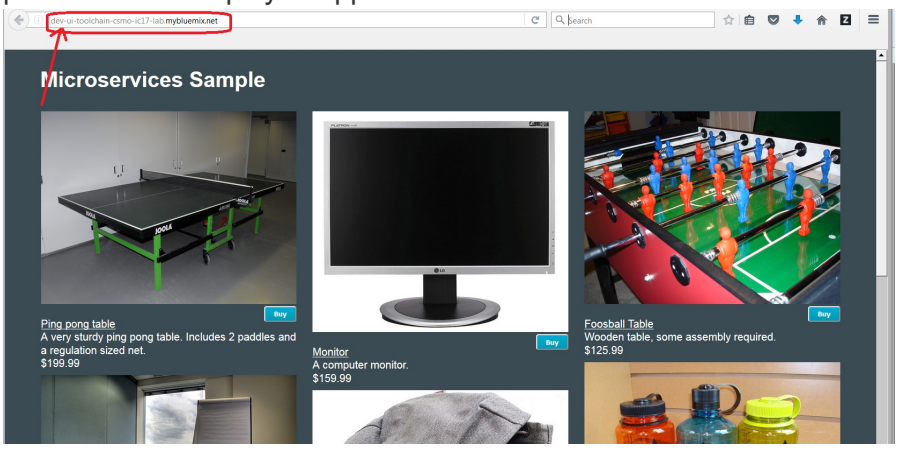
```
export CF_APP_NAME="dev-$CF_APP"
cf push "${CF_APP_NAME}"
```

set the name of the application to deploy as the application name prefaced with *dev* and issues the Cloud Foundry command to deploy it.

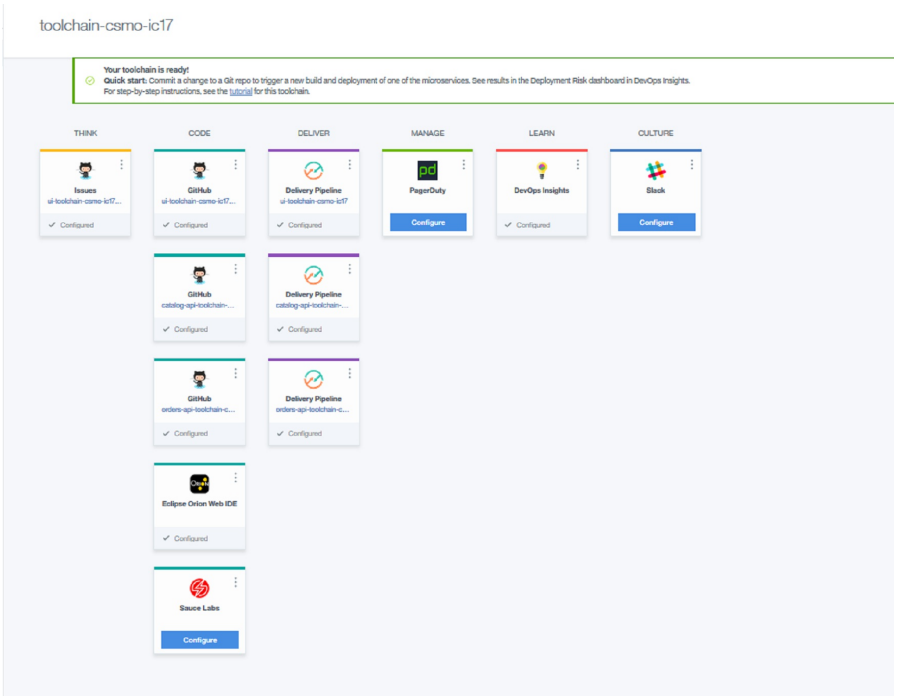
10. Return to the delivery pipeline. Click on the application link to display the application.



11. The application is displayed. We deployed all the microservices to the *dev* space and prefaced each deployed app with *dev*.



12. Close the application tab to return to the delivery pipeline, then return to the Pipeline and finally return to the Toolchain.


toolchain-csmo-ic17
Your toolchain is ready!
Quick start: Commit a change to a Git repo to trigger a new build and deployment of one of the microservices. See results in the Deployment Risk dashboard in DevOps Insights.
For step-by-step instructions, see the tutorial for this toolchain.
THINK
CODE
DELIVER
MANAGE
LEARN
CULTURE
Issues
GitHub
Delivery Pipeline
PagerDuty
DevOps Insights
Slack
Configure
Configured
Eclipse Orion Web IDE
Sauce Labs

Lab 3 Customize Toolchain to add Slack Integration

Objective

This lab will integrate Slack into the Continuous Delivery Toolchain. Slack is a cloud-based team collaboration tool. You integrate Slack into your Toolchain so team members get notified when development events, such as builds, occur.

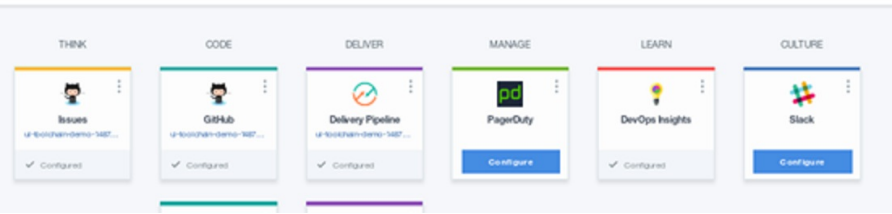
Tasks:

- Task 1: Integrate Slack
- Task 2: Work with Slack
- Task 3: Modify Toolchain for Sauce Labs test job

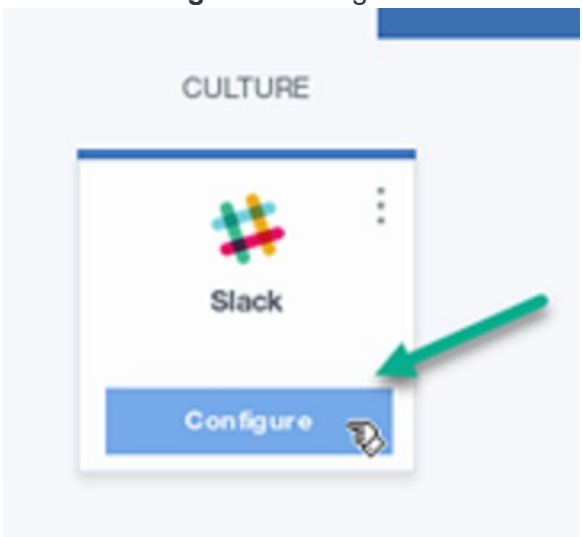
Task 1: Integrate Slack

1. If you needed to add Slack to a Toolchain, you would click **Add a Tool** on the Toolchain display and select **Slack** from the available integrations. You don't have to do this in this case, as the Microservices template already included Slack in the Toolchain. Since you didn't configure it when you created the toolchain, it needs to be configured now. A Slack user ID is provided for use with this exercise (*bluemixinterconnect*).
2. You should be displaying the Toolchain.

   toolchain-demo-1487016038580

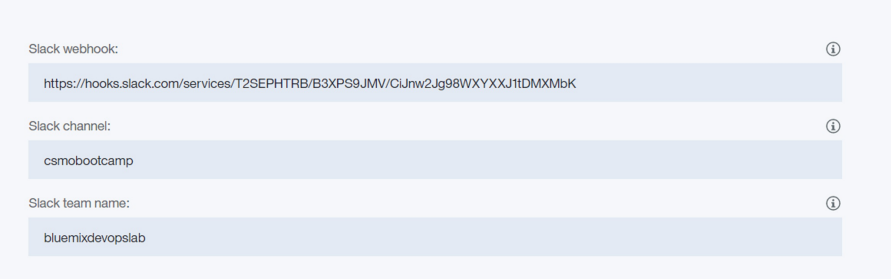
   If not, click on the hamburger menu, then click on **Services**. Click on **DevOps** and then click on **Toolchains**. And finally click on the toolchain you created.
3. Click on **Configure** to configure the connection between Bluemix and Slack.

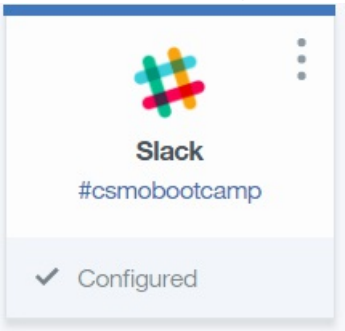
4. Enter the following for Slack webhook (all one string with no blanks or spaces):

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

5. Enter the following for Slack channel: **csmobootcamp**
6. Enter the following for Slack team name: **bluemixdevopslab**



5. Click **Save Integration** to save the information.
6. Slack is now Configured.

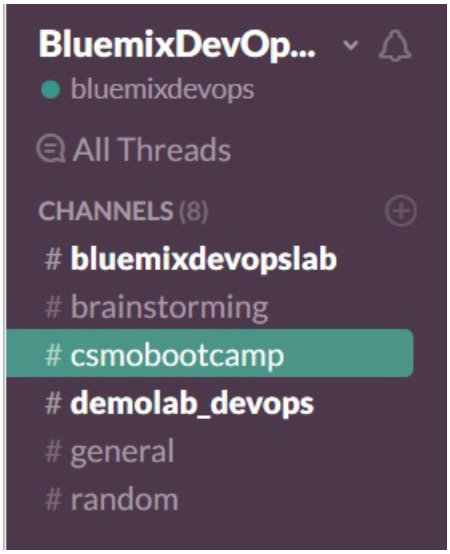


Task 2: Work with Slack

1. In the browser, open a new tab and go to the following URL to go to the (already created) Slack team. https://bluemixdevopslab.slack.com
2. Enter the following information:
    1. Email address: **BluemixDevOps@gmail.com**
    2. Password: **bluemix4me**

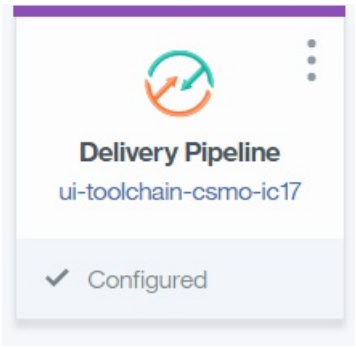
and click **Sign In**.

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

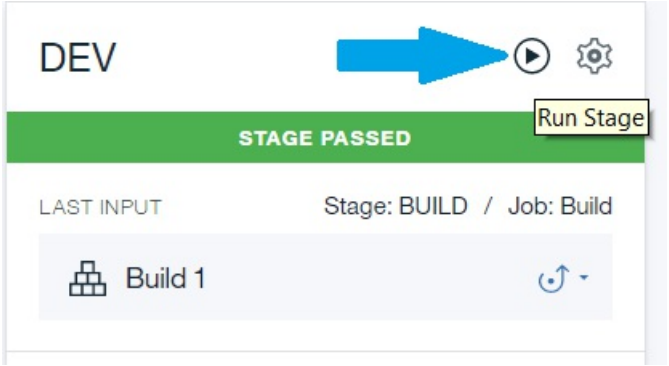


This channel will show all the messages the Toolchain sends to it.

4. Leaving the Slack browser tab open, switch over to the Toolchain browser tab.
5. Click on the UI Delivery Pipeline tile to display the UI Delivery Pipeline.

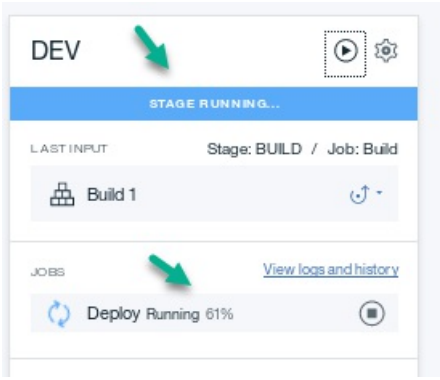


6. Click on the arrow in the DEV stage to run the jobs in the DEV stage.

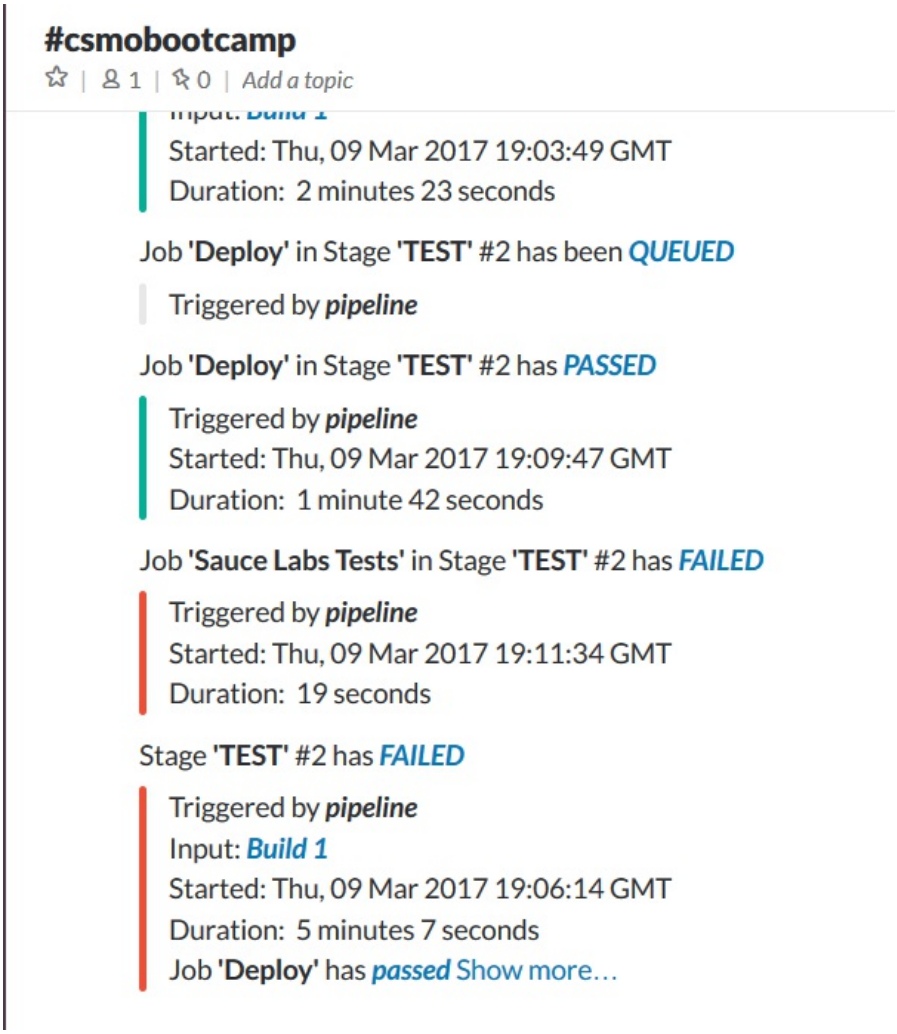


7. The DEV stage runs the jobs and you get a visual indication of the progress of the

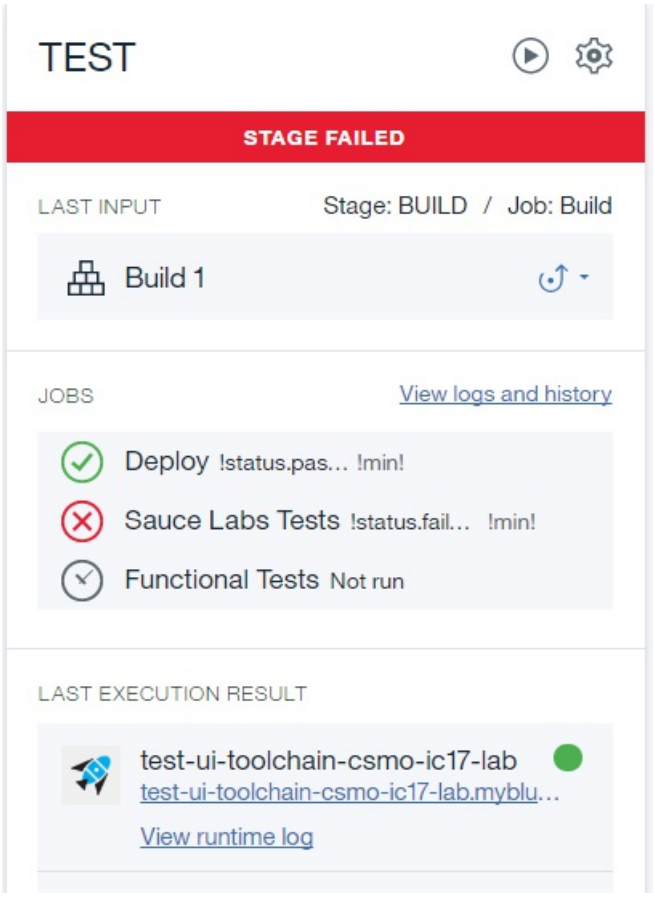
stage.



8. Switch to the Slack browser tab. Here you also get a visual indication of the progress of the DEV stage process. This is useful for notifying team members when events occur without them being logged into Bluemix. The DEV stage passed but the TEST stage failed. The *Deploy* job was successful, but the *Sauce Labs Test* job failed and the *Functional Tests* job was not attempted.

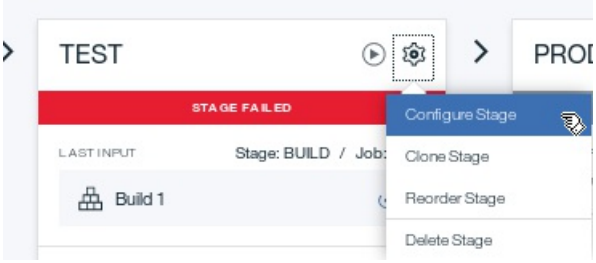


9. Switch back to the Bluemix browser tab. The Toolchain indicates the same results. Something is wrong with the Toolchain.

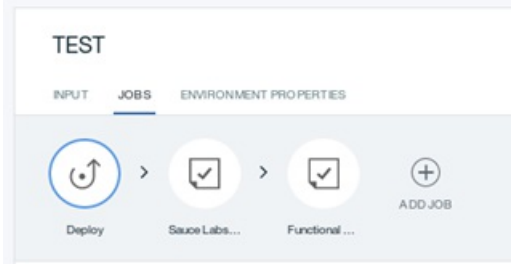


Task 3: Modify Toolchain for Sauce Labs test job

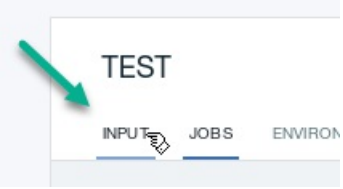
1. When you ran the DEV stage. Why did the TEST stage start? In the TEST stage tile, click on the gear and then click **Configure Stage**.



2. The TEST stage has three jobs, *Deploy*, *Sauce Labs Test*, and *Functional Tests*.



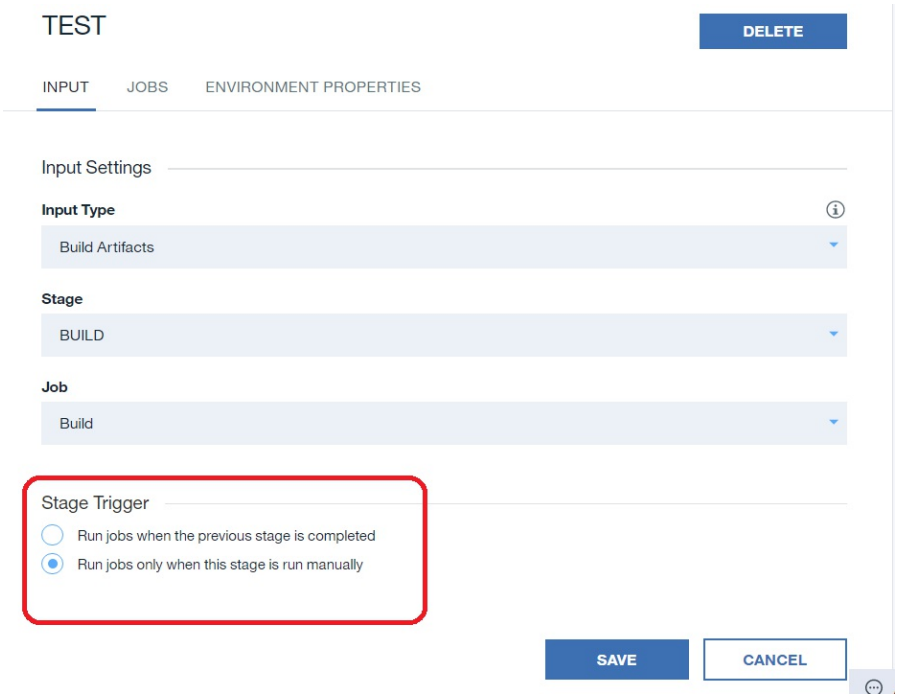
3. The *Deploy* job is highlighted (the blue circle around the icon), so details for the *Deploy* job are displayed.
4. Click **INPUT** to display the input settings for the TEST stage.



5. The Stage Trigger for the TEST stage indicates that this stage will run when the prior stage is complete. So in this case, when the DEV stage is complete, the Toolchain started the TEST stage.

6. Click on **JOBS** to display the jobs for the TEST stage.
7. Click on the **Sauce Labs Test** icon to display the details for the *Sauce Labs Test* job.
8. At the bottom of the details, under *Run Conditions*, the option to stop the stage if this job (the *Sauce Labs Test* job) fails. That explains why this stage stopped and why the *Functional Tests* job did not run.

9. Why did the *Sauce Labs Test* job fail? As you may recall, you didn't configure that tool integration when you created the Toolchain. Sauce Labs requires a valid userid and password. If you had one, you could configure the Sauce Labs integration with those details. So we have two choices, either remove the *Sauce Labs Test* job from the TEST stage or, assuming we may one day get a Sauce Labs account, allow the *Sauce Labs Test* job to fail but continue the stage. For the purpose of these lab exercises, you really don't need to deploy to multiple spaces, so instead change the stage configuration to run manually. If you weren't going to add Sauce labs in the future, it would be a good idea to remove the Job from the stage and remove the Sauce labs tool from the toolchain.
10. Click **Input** to return to the Input tab.
11. Under Stage Trigger, select **Run jobs only when this stage is run manually**.



12. Click **Save**. The Test stage will now only run when you click Run Stage to start it.
13. Return to your toolchain and repeat Steps 10-12 for the other catalog and orders microservices.

Lab 4 Customize Toolchain to add Bluemix Availability Monitoring

Objective

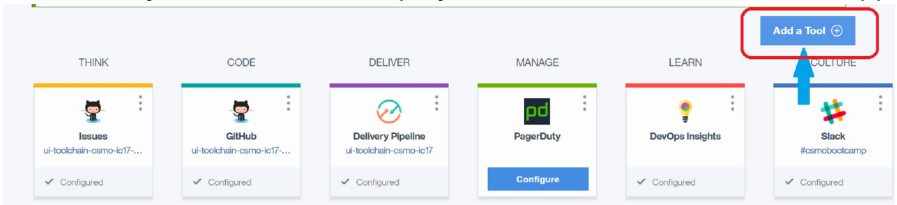
This lab integrates Bluemix Availability Monitoring into the Continuous Delivery Toolchain. BAM Bluemix Availability Monitoring helps DevOps teams ensure their applications are always available and meeting user expectations for response time as they roll out continuous updates. The service, which is tightly integrated into the DevOps toolchain, runs synthetic tests from locations around the world, around the clock to proactively detect and fix performance issues before they impact users.

Tasks:

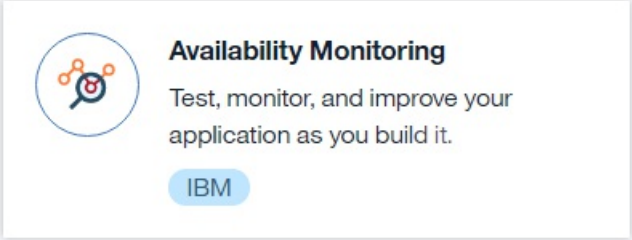
- Task 1: Integrate Bluemix Availability Monitoring
- Task 2: Work with Bluemix Availability Monitoring

Task 1 Integrate Bluemix Availability Monitoring

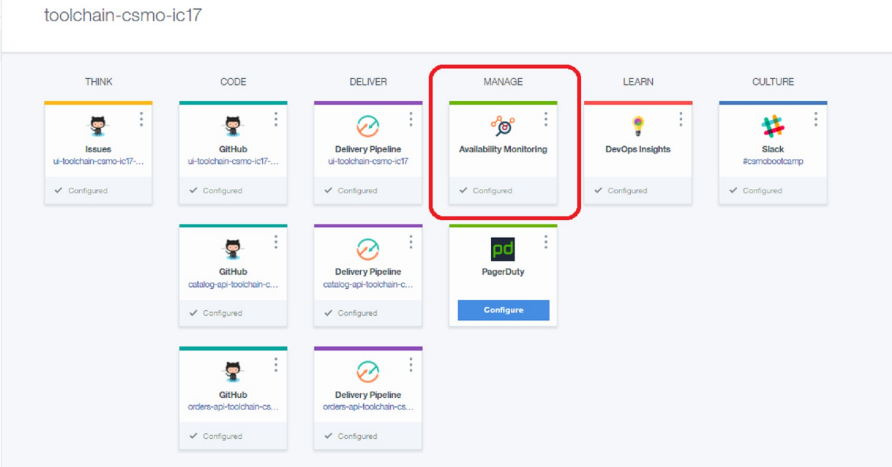
1. Make sure your toolchain is displayed and click **Add a Tool** in the upper right corner.



2. Click **Availability Monitoring** from the list of Tool integrations.

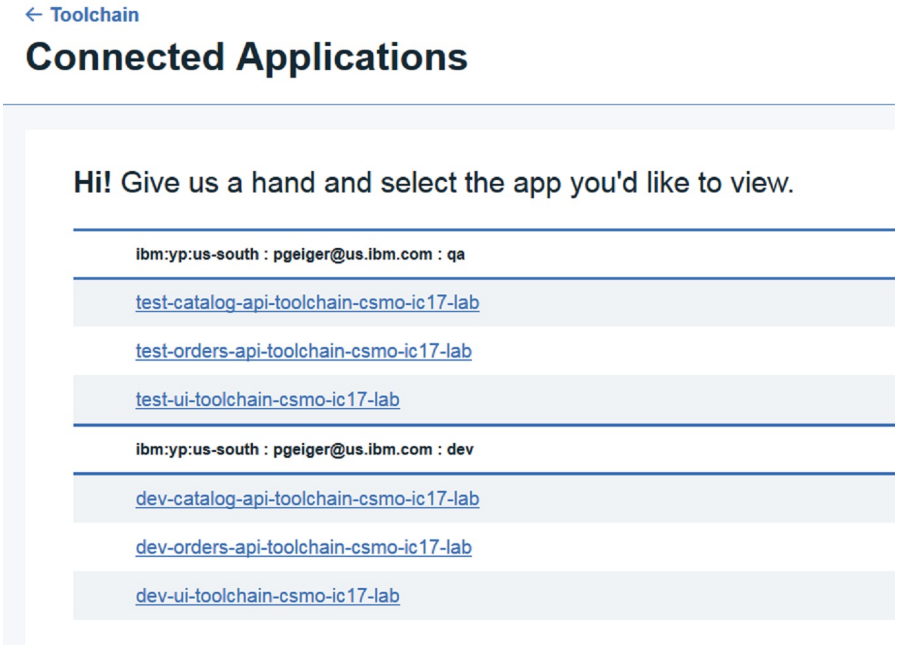


3. Notice that this tool does not require configuration, so click **Create Integration** to add it to your toolchain. It is added under the **Manage** phase, and indicates that it is configured. You also receive a notification in the Slack channel that the service has been bound to the toolchain.

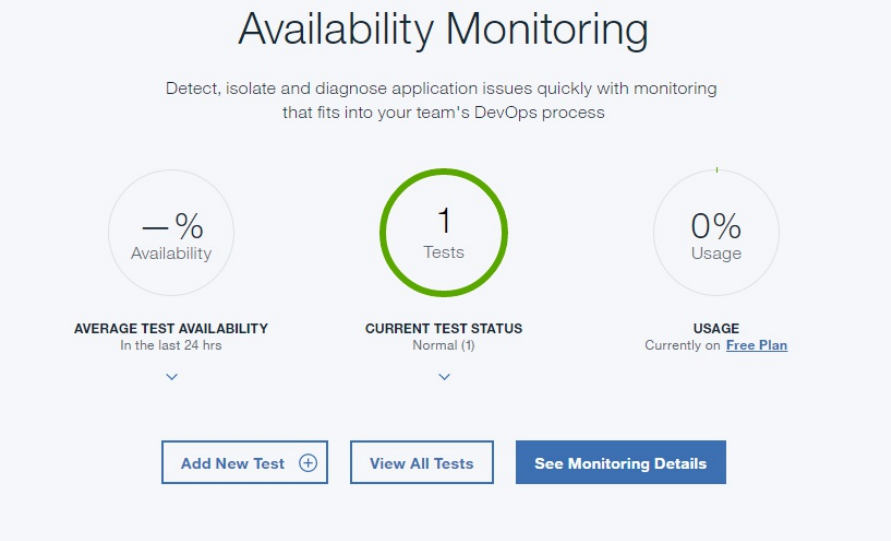


Task 2: Work with Bluemix Availability Monitoring

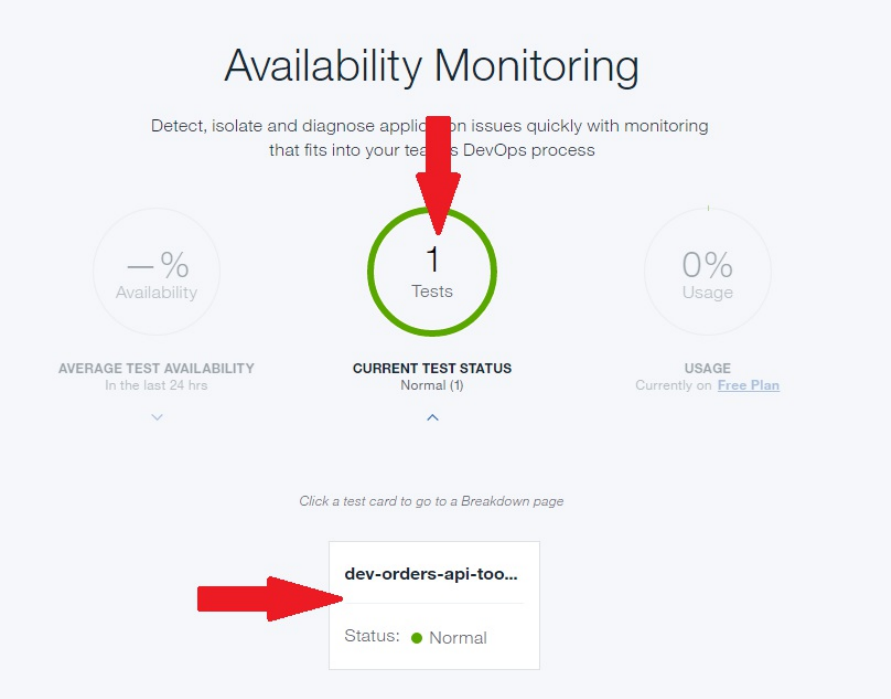
1. From the toolchain, click **Availability Monitoring**. A list of the apps that are bound to the service is displayed. Noticed that you have all 3 microservices running in both test and dev. This is because the Deploy Job passed succcessfully before the Test Job failed.



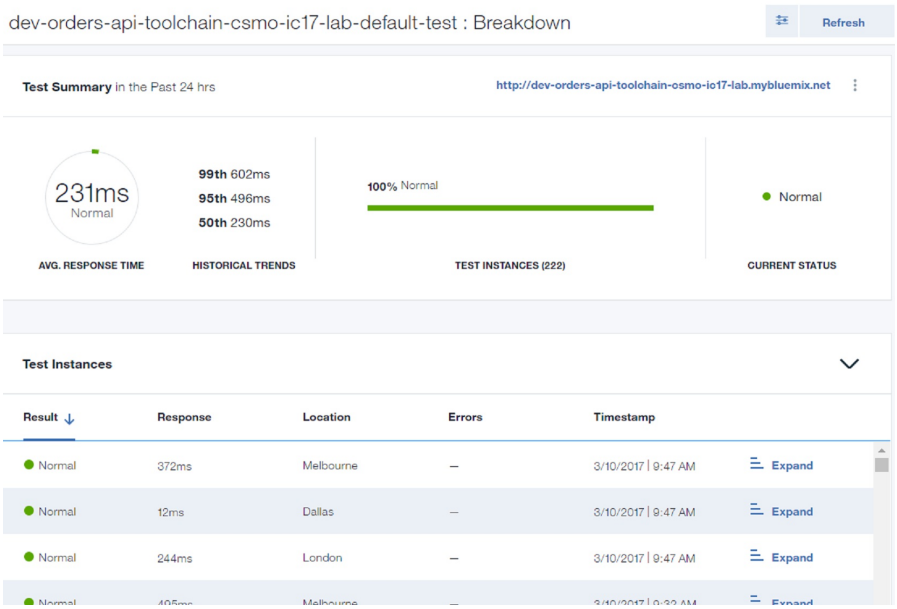
2. Click the name of one of the monitored Apps to display the Availability monitoring information for that app. Because it hasn't been running long, there is no information to see yet.



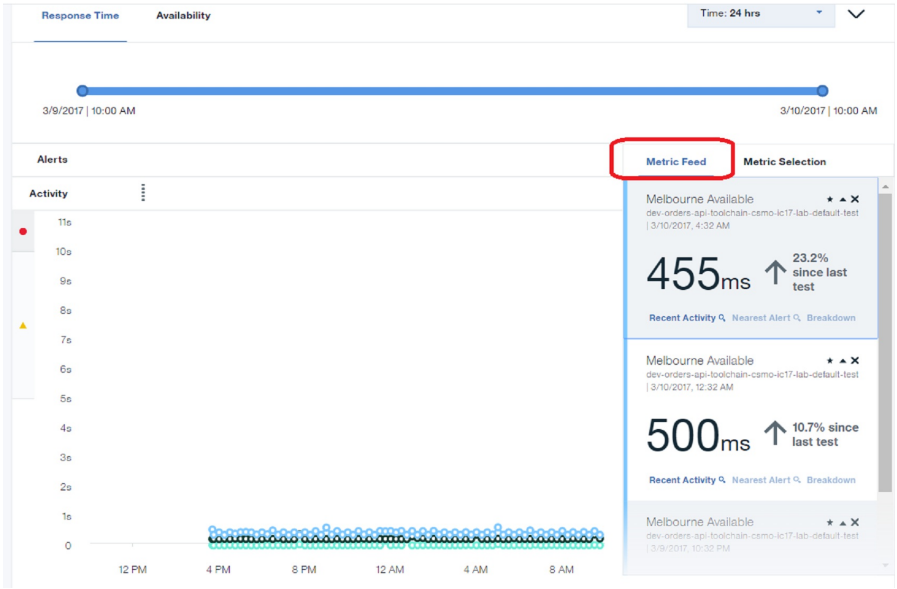
3. By default a single test is configured. The main application URL is monitored by default. Any other URLs and services that you monitor can be inside or outside of Bluemix and do not need to be related to the associated Cloud Foundry application. Click the circle that is labeled **1 Test**, then click the test card to display the breakdown.



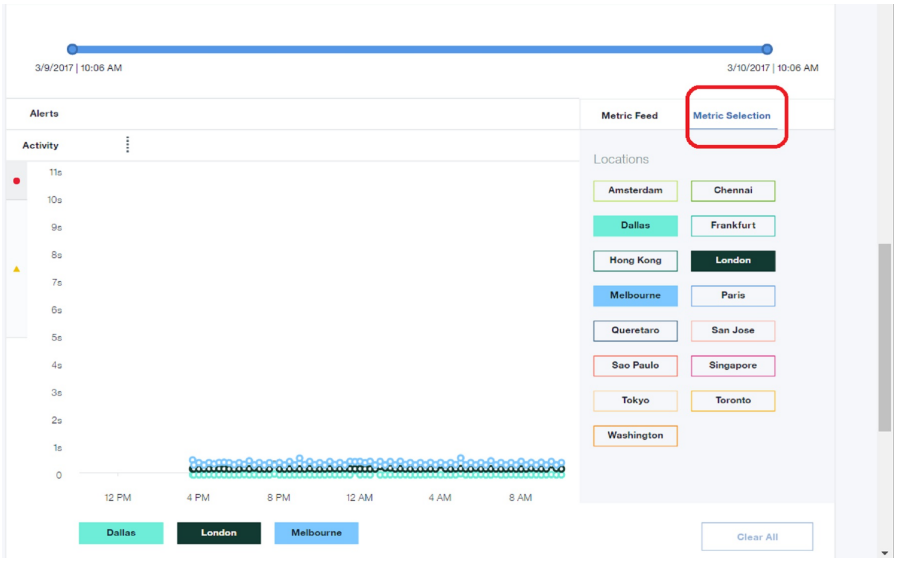
4. This page displays a summary of the tests that have been run in the last 24 hours. You see a test summary at the top, including average response time, historical trends, and current status. This page also shows the details for each of the tests that have been run from 3 default locations, as well as response time and availability information from each of the locations.



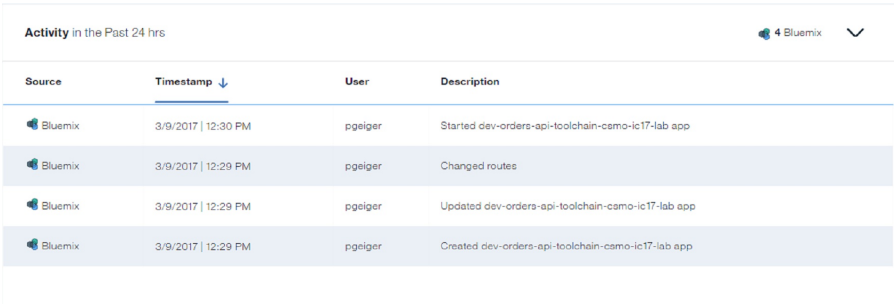
5. Scroll down to the graph that shows response time. Response times for each of the 3 locations is displayed in the graph. Click on one of the locations to display response time for that location only. Click single points on the graph to display the Metric Details.



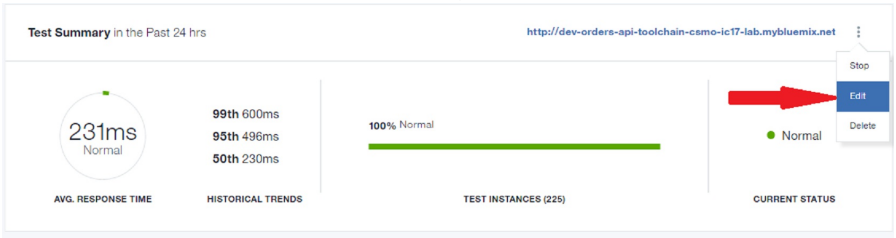
6. Click **Metric Selection**. Here you see the different locations that can be used for availability tests. Click on locations to add/remove them from the graph.



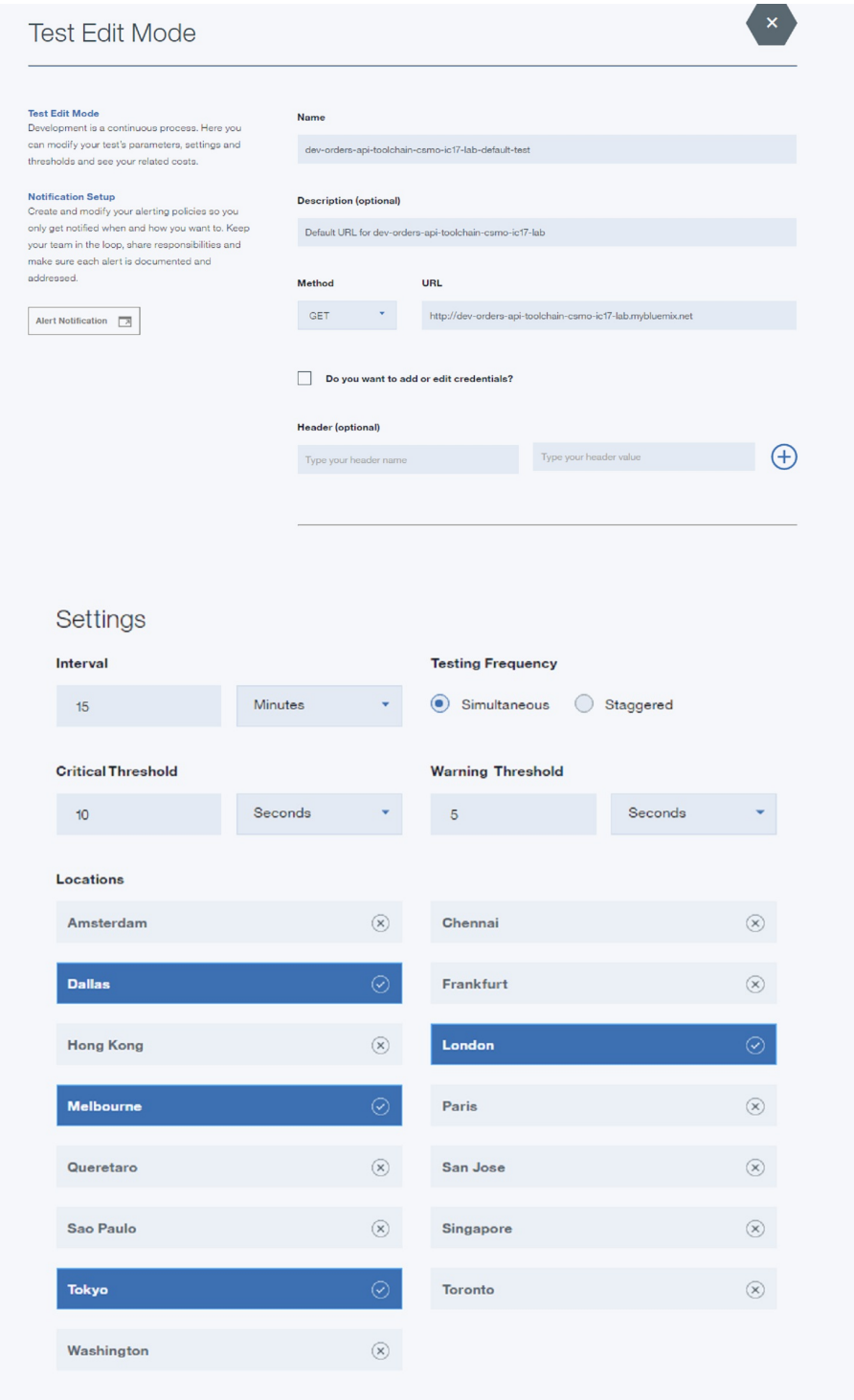
7. At the bottom of this page, you see the activity related to the monitored application in the last 24 hours.



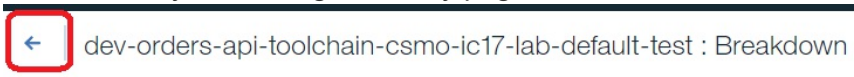
8. Scroll back to the top of the page and click Edit from the pull down menu beside the appliction name in the Test Summary Section.



9. Here you can make changes to the API that is being monitored, the test name, frequency, locations , etc. Click on a location name to add it to the test, or click on a selected location to remove it. Notice the Notification Setup on the left side of the window. If you were using IBM Alert Notification Service, you could configure your notifications for these tests from here. You will add that to the toolchain in the next lab.



10. Click the 'X' at the top of this window to close it. Click the arrow at the top to return to the Availability Monitoring Summary page.



11. Click **Add a New Test**. Notice the different types of tests that you can add:

- Response time and availability trends for this as well as dependent web pages and APIs

- Availability of the app in different locations
- Identify patterns by correlating performance with detailed alerts and development activity
- Imitate real end-user behavior by monitoring synthetic scripts.

Click on each of the options to see how easily new tests can be added

Due to time constraints you won't add additional tests now. If you would like to explore Bluemix Availability Monitoring in more detail, visit the open labs area and check out Lab Number 9902. When you are finished viewing the monitoring information, return to your toolchain.

Lab 5 Customize Toolchain to add IBM Alert Notification Service

Objective

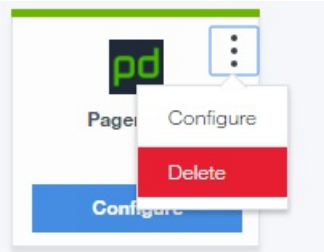
This lab integrates IBM Alert Notification Service into the Continuous Delivery Toolchain. ANS Doclink Use the Alert Notification service to create filter conditions for processing alerts, so that you receive only the alerts that you are interested in. You can create notification policies that contain filter conditions that, when met, trigger notifications. Notification policies define which alerts you are interested in by filtering for the alerts that, for example, are of critical severity, and whether to send a notification. The recipients can be individual users or groups.

Tasks:

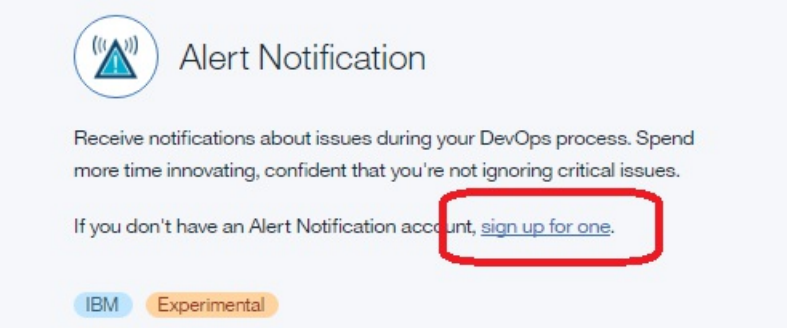
- Task 1: Integrate IBM Alert Notification Service to your Toolchain
- Task 2: Enable Bluemix Availability Monitoring for the Alert Notification Service
- Task 3: Configure Alert Notification Slack Channel
- Task 4: Create Notification Policies
- Task 5: Test the Alert Notification

Task 1 Integrate IBM Alert Notification Service to your Toolchain

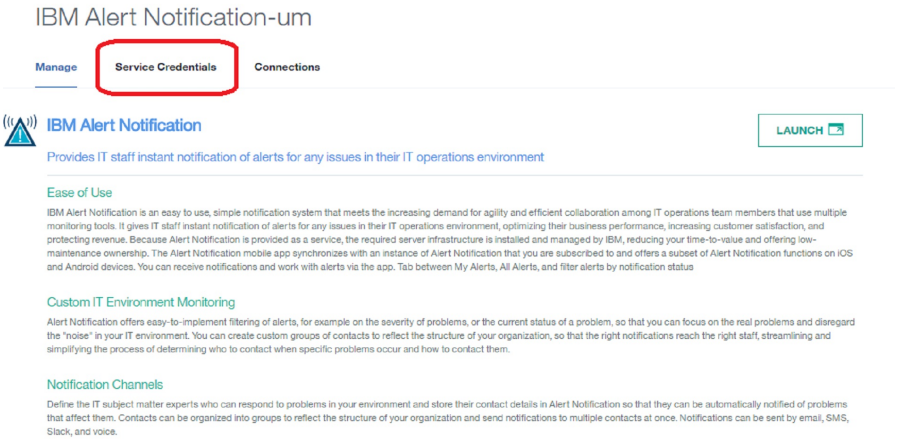
1. Because you are going to use IBM Alert Notification Service as your alerting service in this exercise, you can remove PagerDuty from the toolchain. From the Toolchain, click the selection menu on the PagerDuty card and select **Delete** and click Delete to confirm.



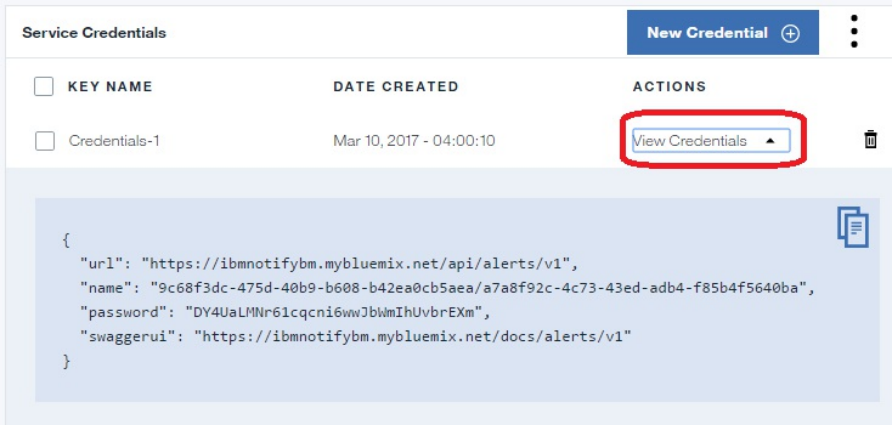
2. Click **Add a Tool**. In the Tool Catalog, click Alert Notification. Notice that you need an Alert Notification API URL, an API key name, and an API key password. To obtain that information, click the link that says **sign up for one**.



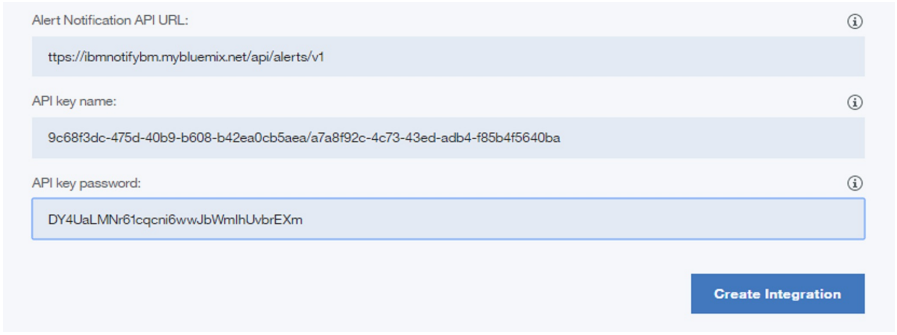
3. A new browser tab opens to allow you to add the Alert Notification Service to your Bluemix space. Keep the defaults for the Service name and credentials name and click **Create**. When the service is created, the Manage page for the new service is displayed.
4. Click **Service Credentials** to view the credential information. This is the information that is needed for the toolchain configuration.



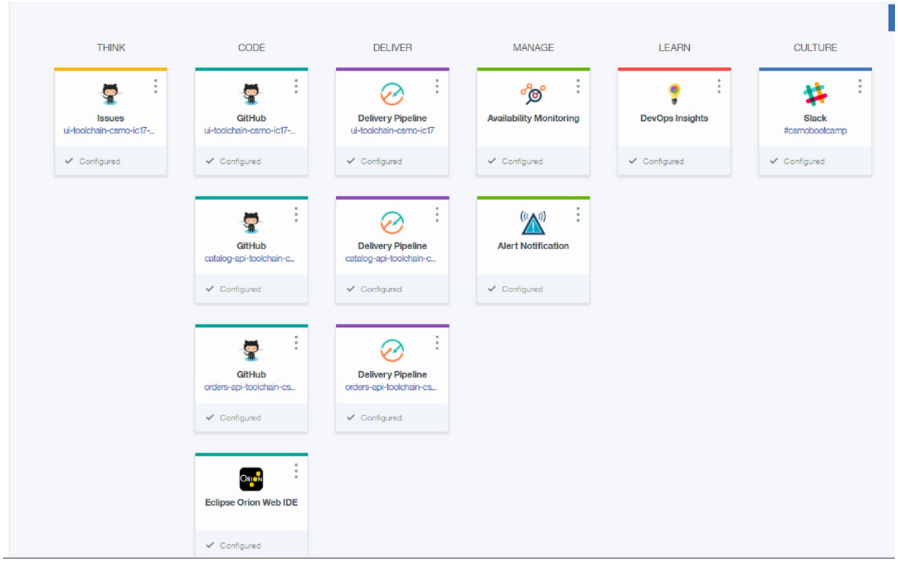
5. There is one Key listed. Click **View Credentials**.



6. Copy the **url**, **name**, and **password** to the corresponding fields in the Tool Integration configuration for your toolchain and click **Create Integration**. Your toolchain should still be open in a separate browser tab, so you can copy and paste between them.

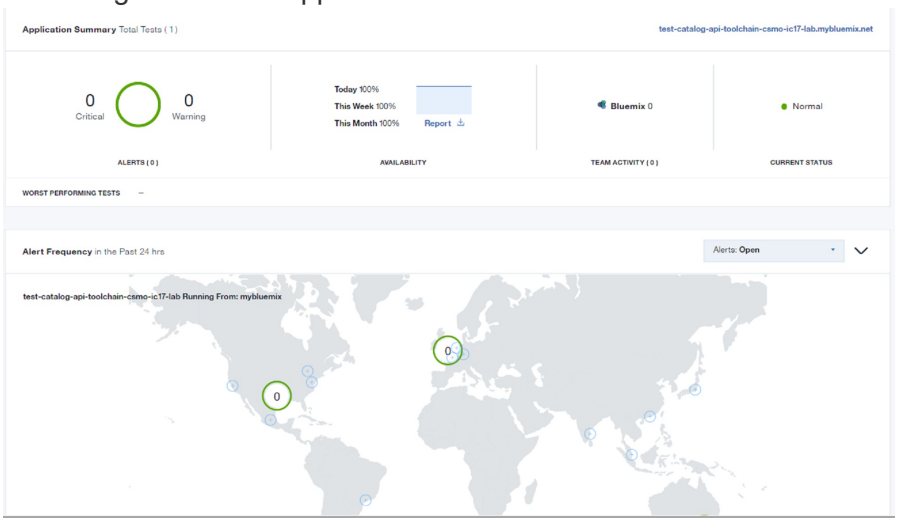


7. Alert Nofication is added to your toolchain and indicates that it is configured.



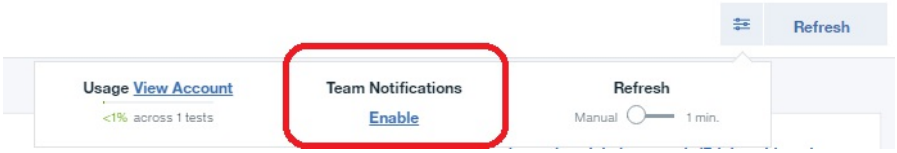
Task 2 Enable Bluemix Availability Monitoring for the Alert Notification Service

1. Click the **Availability Monitoring** card in your toolchain.
2. Click one of the applications and click **View All Tests**. This view displays all tests that are configured for the application.



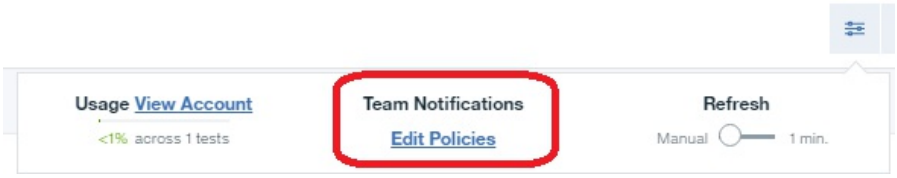
3. Click **Configure** in the upper right corner.

4. Under **Team Notifications**, click **Enable**. This configures the integation between Bluemix Availability Monitoring and the Alert Notification Service. Wait a few minutes for the configuration to complete. If you don't see the message change from 'Pending' to 'Edit Policies' after a few minutes, you may need to refresh your browser.

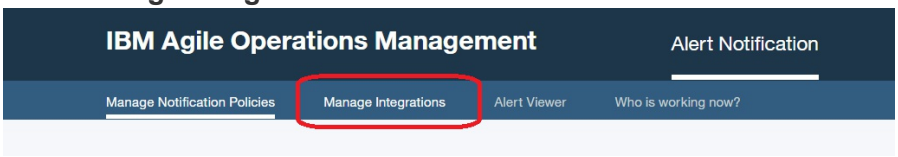


Task 3 Configure Alert Notification Slack Channel

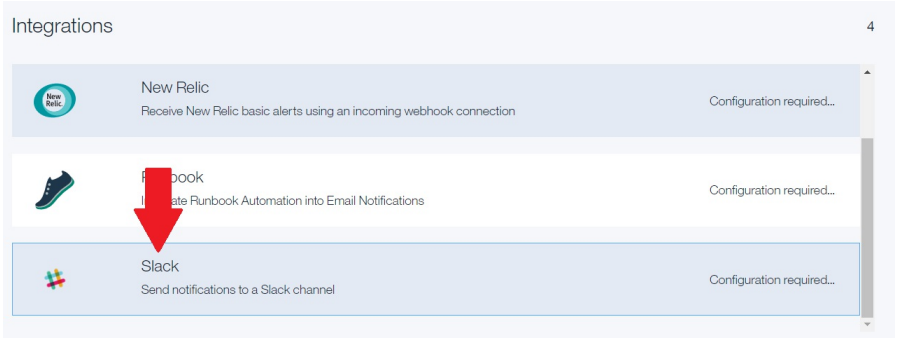
1. Click **Edit Policies** to configure your notification policies.



2. The Agile Operations Management console opens. This is where you manage your notification polices, set up integrations with other applications, and view the alerts that have been received. For this exercise, you will send all critical alerts to the same Slack channel that you used previously, so first you need to create the integration with Slack. Click **Manage Integrations**.

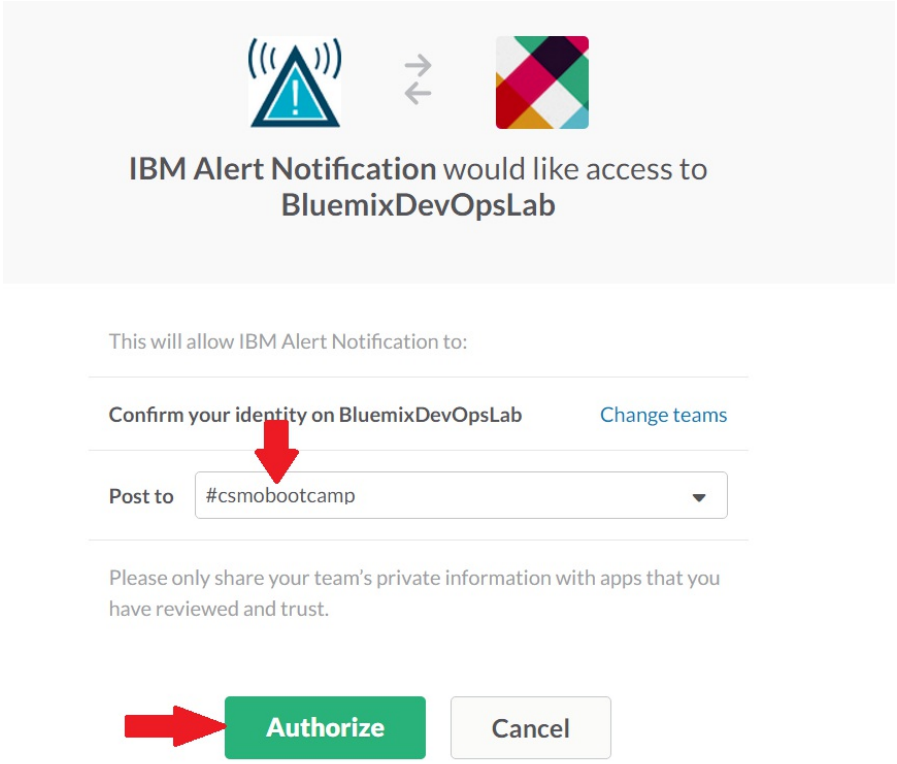


3. Click **Slack**.

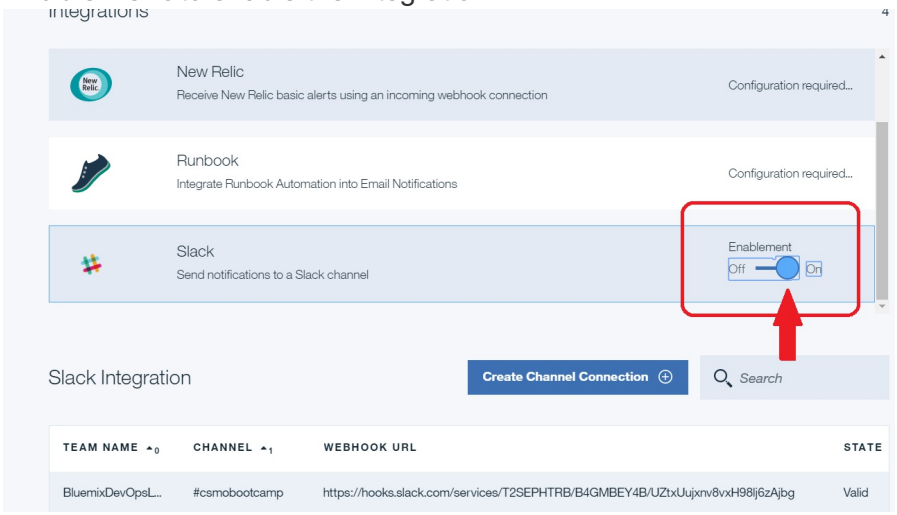


4. Click **Create Channel Connection** and **Accept**.
5. Enter **Bluemixdevopslab** for the Slack URL when prompted to sign into your team and click **Continue**.
6. Enter **BluemixDevOps@gmail.com** for the email address and **bluemix4me** for the password and click **Sign in**.
7. You are prompted to authorize IBM Alert Notification to post to the Slack Channel. Fo

the channel to post to, select #cmsobootcamp from the list of channels and click **Authorize**.



8. The Integration is created, but is disabled by default. Click the circle under **Enablement** to enable the integration.

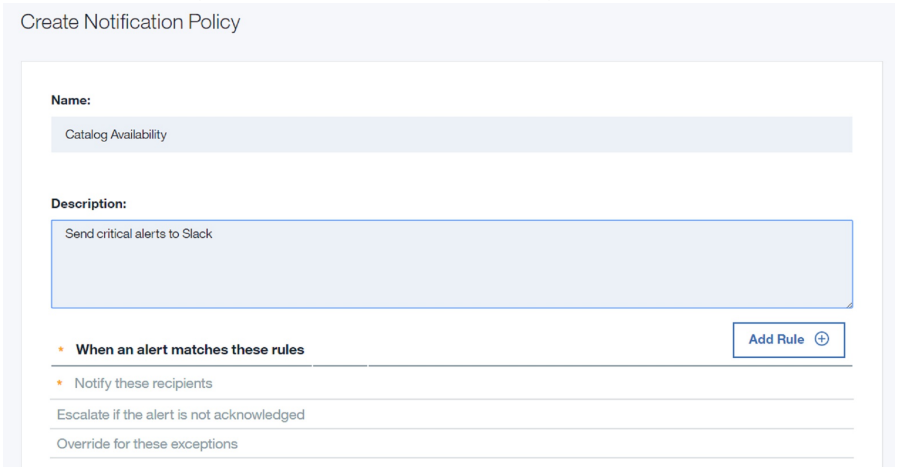


Task 4 Create Notification Policies

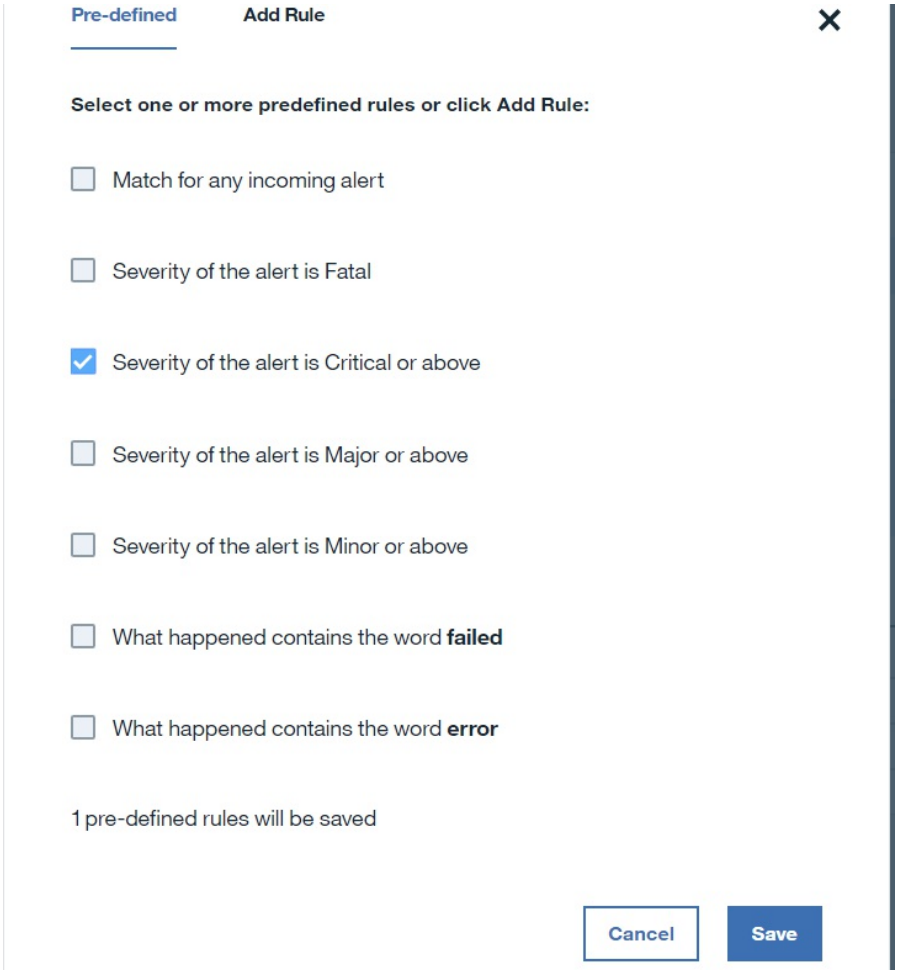
1. Next you need to create your notification policies. In this case, you want all events that are critical or above to be sent to the Slack channel. Click **Manage Notification Policies**. There is one sample notification policy that is disabled by default.
2. Click **Create a Notification Policy**.

3. Configure the Notifiction Policy as follows:

- Name: Any name of your choice It must be unique within your Alert Notification Service instance.
- Description: Any descriptive information of your choice.

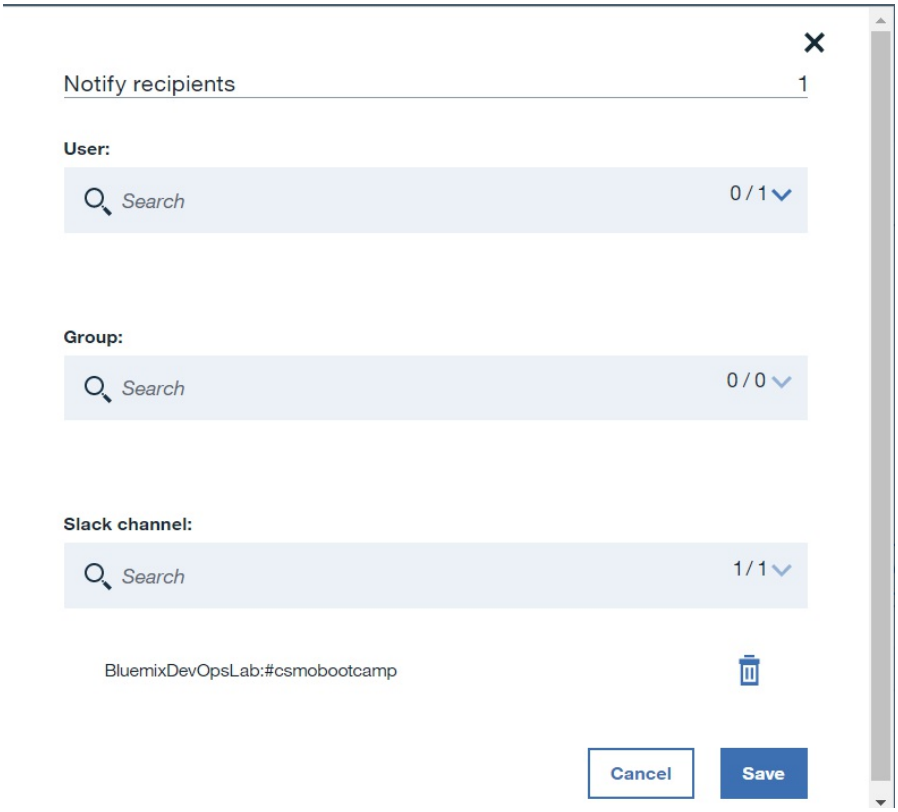


- Click **Add Rule**
- Select **Severity of the alert is Critical or above** and click **Save**.



Notice that you have the option to delay notications until a certain number of alerts have been received within a certain time period after the rule has been added.

- Click **Add Recipient**.
- Under **Slack Channel**, select the Slack Channel that you added from the pull down list. Notice that you can also send emails, texts, or pages to individual users or groups of users from here. Click **Save**



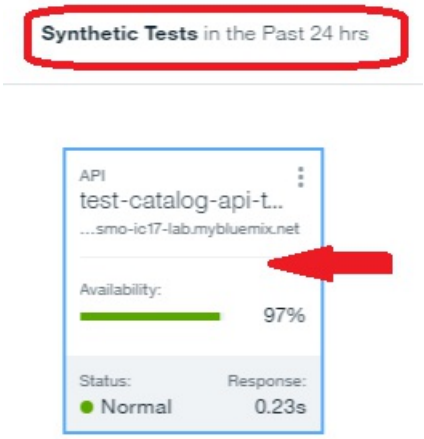
After the recipients have been added, you have the option to have them notified every time the rule is matched, or only the first time it is matched within an 8 hour period. Accept the default value of Every time the rule is matched.

- You also have the option to add escalation and override rules here, but in this case, you will not configure those options. Click **Save** to save the policy.

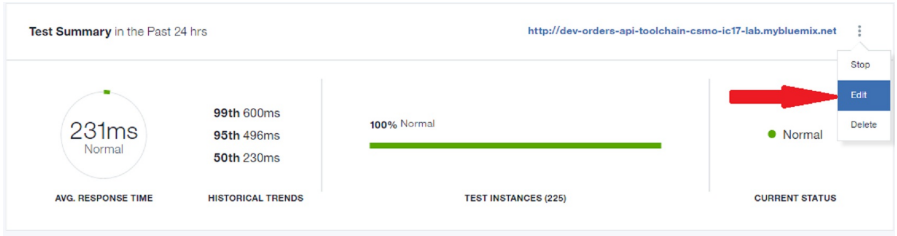
4. The policy is created and is disabled by default. **Enable** the policy.

Task 5 Test the Alert Notification

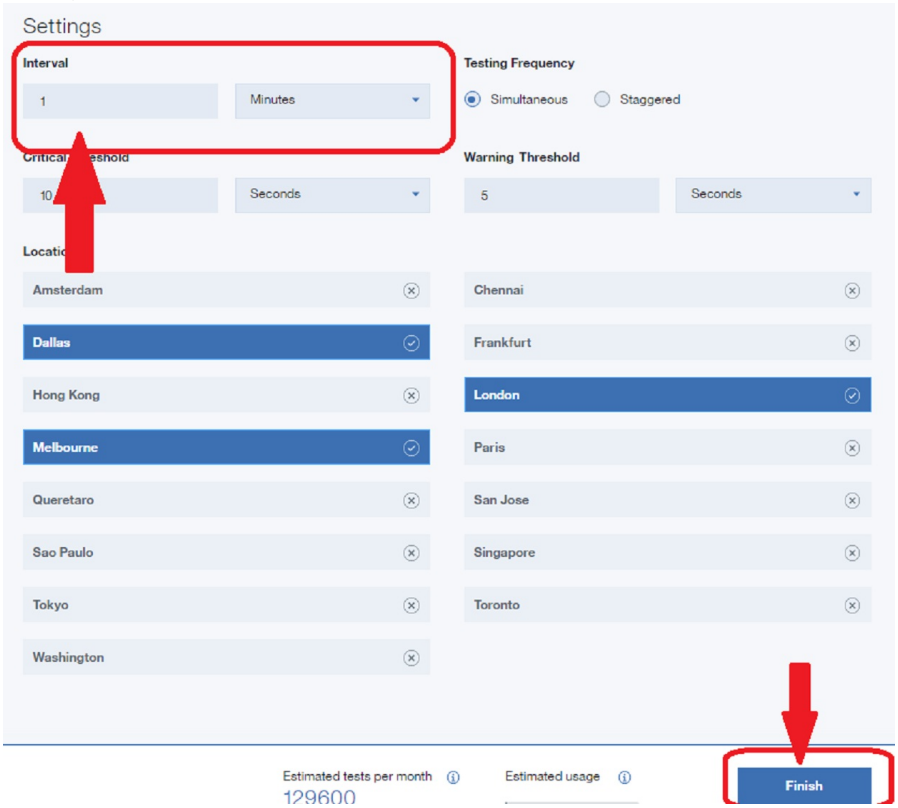
1. Now let's see the new Alert notification in action with Bluemix Availability Monitoring. Click the **Monitoring** tab in your browser to return to the Application Monitoring summary. You should be back to the All Tests view for the application that you selected. Locate the **Synthetic Tests** section and click the card for your applications test.



2. Click **Edit** to edit the test configuration.

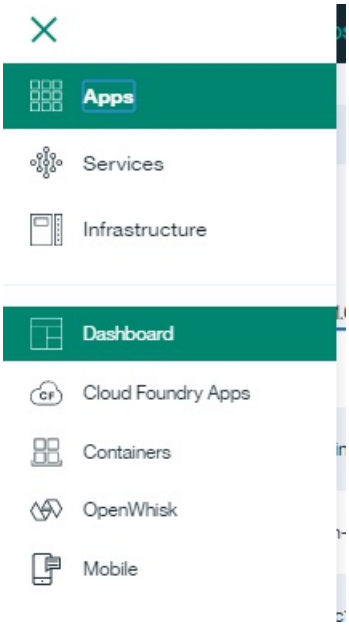


3. Change the frequency of the test in order to see the alert faster. Under **Settings**, change the **Interval** from 15 minutes to 1 minute and click **Finish**.

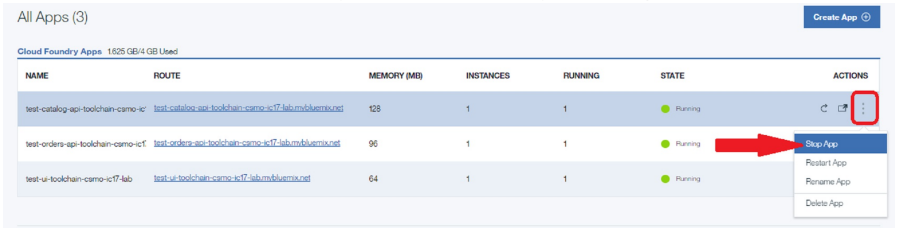


4. This is a test for application availability, so now you can go stop the application to test the notification policy. If you don't have a browser tab open to the Bluemix console, open a new tab to bluemix.net, and log in using your Bluemix credentials, if necessary.

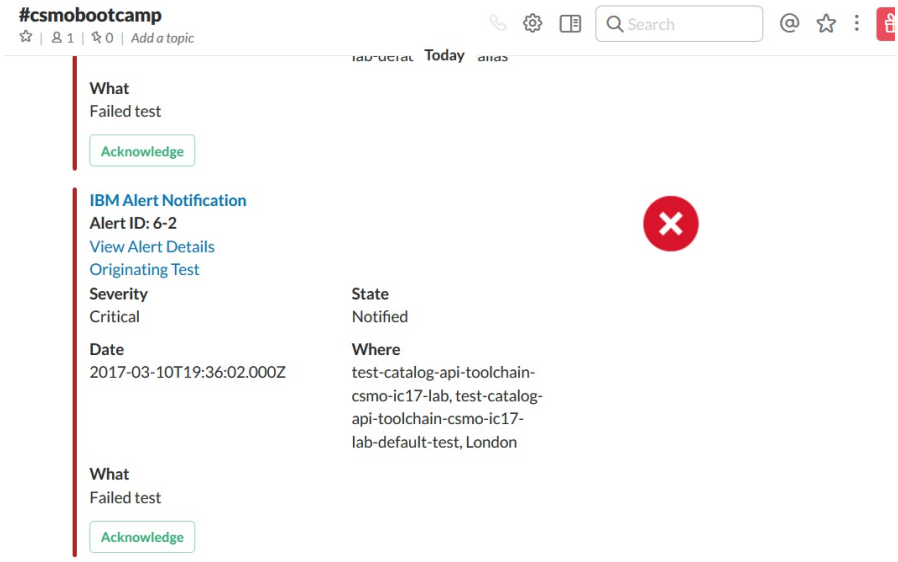
5. Click **Apps** and **Dashboard** from the hamburger menu.



6. Locate the App that you configured the monitoring and notification for. Ensure that you are in the right space for the app that you configured. In this example, it is the 'test-catalog-api-toolchain-csmo-ic17-lab' app in the qa space. From the actions menu, select **Stop App**. Confirm your selection by clicking **Stop**.

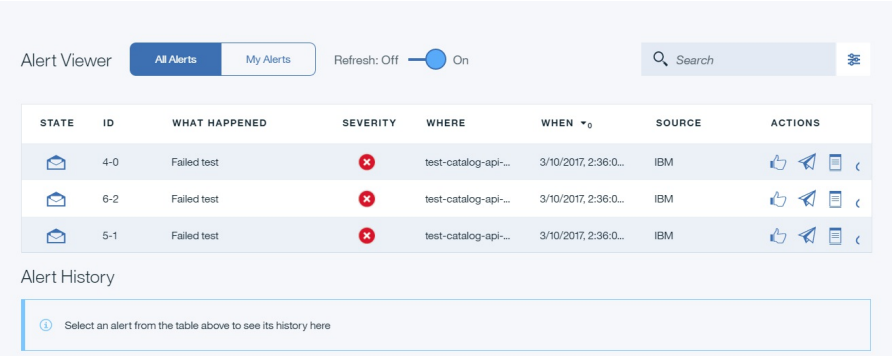


7. Click the tab for your Slack Channel. Because the testing interval is set to 1 minute, you should quickly start seeing notifications in Slack that the appication is down. Notice that you have the option of Acknowledging the Alert directly from Slack.

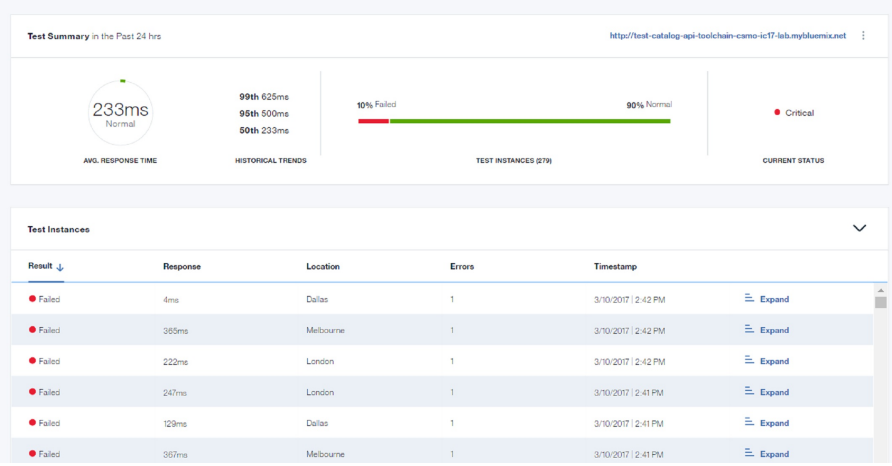


8. Click the **Manage Notifications** tab in your browser to return to the Alert Notification

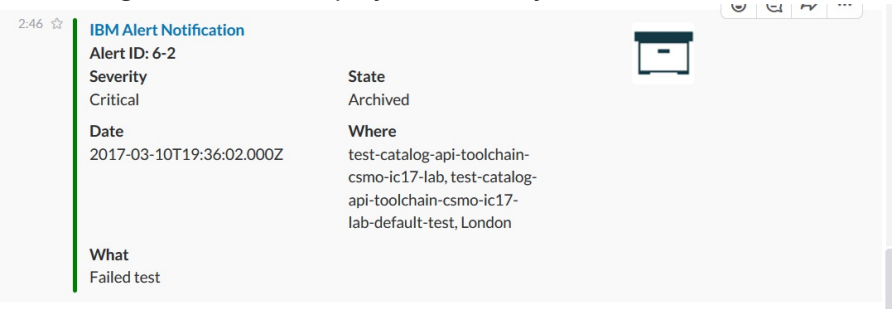
information and click **Alert Viewer**. You see the most recent events displayed there as well.

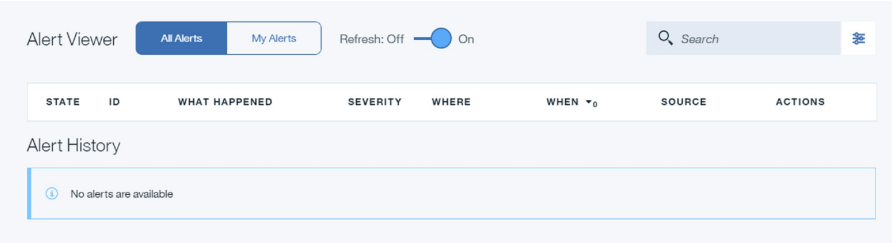


9. Click the browser tab for your Availability Test. The tab is titled 'API Breakdown'. You see the indication of the tests failing there as well.



10. Now restart the app. Return to the Bluemix Console and select **Start App** under Actions. After a few minutes, you will receive a notifications in Slack that the critical events have been archived because the test is no longer failing. You will see one archive event for each critical event that was received. Return to the Alert Viewer in Alert Notication and see that the Alerts have now been cleared there as well. If you are still seeing some alerts displayed, refresh your browser.





11. At this point, it would be a good idea to change the test interval back to a more realistic value.

There is a lot more that you could and probably would do for a complete Cloud Service Management and Operations story, but hopefully these exercises have given you a good starting point for how you can set up incident and service management in a Bluemix environment. The tools that were used in the exercises are examples of some of the functions that would typically be included, but you are free to use the tools of your choice for the various activities.