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. We will integrate Slack into our Toolchain so team members get notified when development events, such as builds, occur.

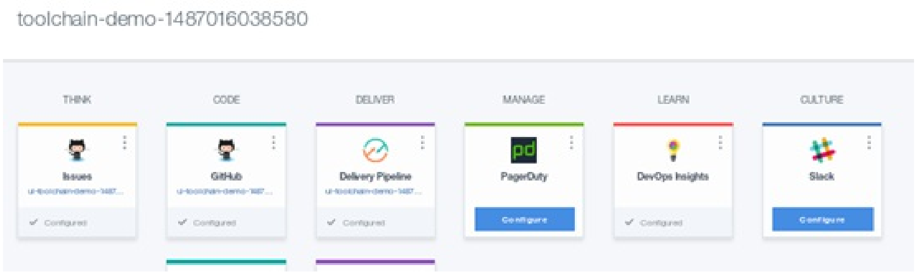
Tasks:

- Task 1: Integrate Slack
- Task 2: Examine Slack notifications

Task 1: Integrate Slack

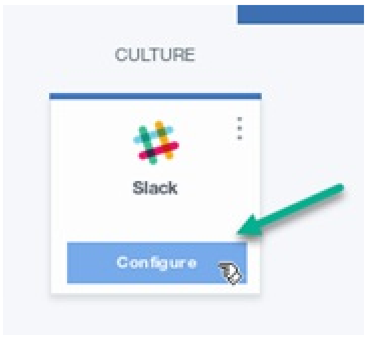
1. If we needed to add Slack to a Toolchain, we would click **Add a Tool** on the Toolchain display and select **Slack** from the available integrations. We don't have to do this as the Microservices template already included Slack in the Toolchain but we did not configure it. We also have a Slack user ID already created (*bluemix_interconnect*).

2. You should be displaying the Toolchain.



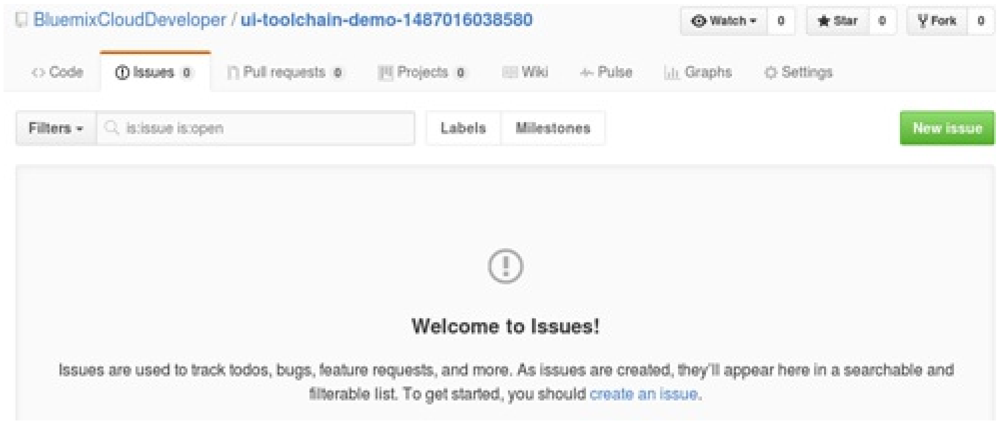
If not, click on the hamburger menu, then **Menu**. Then click on **Services** then **DevOps**. Then click on **Toolchains**. And finally click on the toolchain you created.

3. Click on **Configure** to configure the connection between Bluemix and Slack.



4. Enter the following all as one string as the Slack webhook.
   https://hooks.slack.com/services/

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.

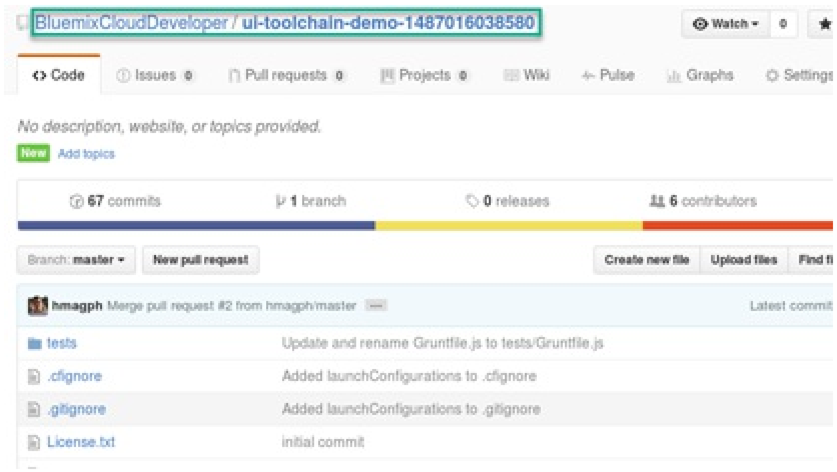


Welcome to Issues!

Issues are used to track todos, bugs, feature requests, and more. As issues are created, they'll appear here in a searchable and filterable list. To get started, you should create an issue.

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 page. (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



No description, website, or topics provided.

New Add topics

67 commits 1 branch 0 releases 6 contributors

Branch: master New pull request Create new file Upload files Find fi

hmagph Merge pull request #2 from hmagph/master Latest commit

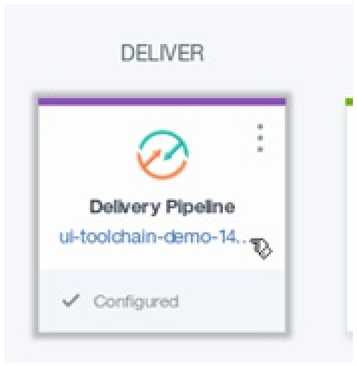
tests	Update and rename Gruntfile.js to tests/Gruntfile.js
.cfignore	Added launchConfigurations to .cfignore
.gitignore	Added launchConfigurations to .gitignore
License.txt	initial commit

while 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.

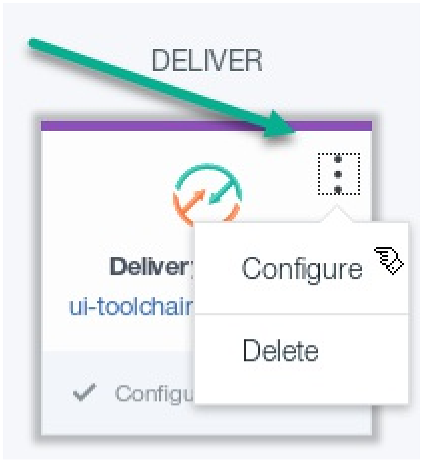
5. **Deliver** is where the code gets built, tested and deployed through the integrations of build pipelines, one per microservice. We explore build pipelines later.
6. **Manage** is where the intergrations to management tools, such as Pager Duty, get added.
7. **Learn** is where the integrations to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

Task 2: Examine the build pipelines

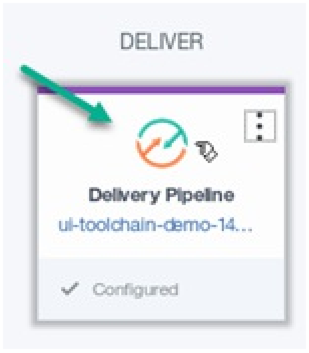
1. Clicking on one of the GitHub repo links or the **Configured** link will display the respective GitHub repo.



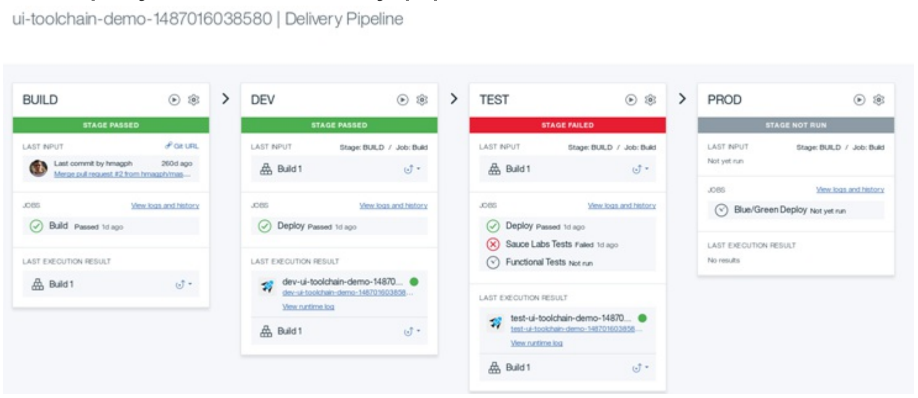
2. You can either click on the three vertical dots on the upper-right of the *ui-toolchain-demo* Delivery Pipeline tile to display **Action** and then click on **Configure**

   or click on the circle in the center of the *ui-toolchain-demo* Delivery Pipeline tile

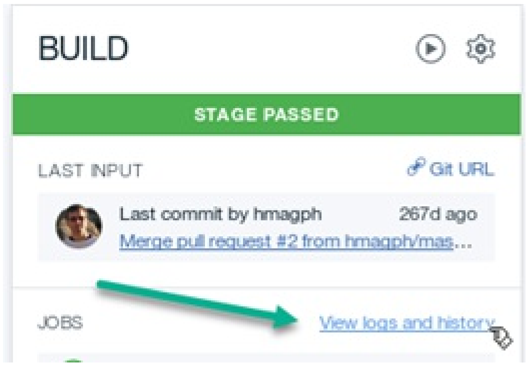
   to display the UI delivery pipeline.

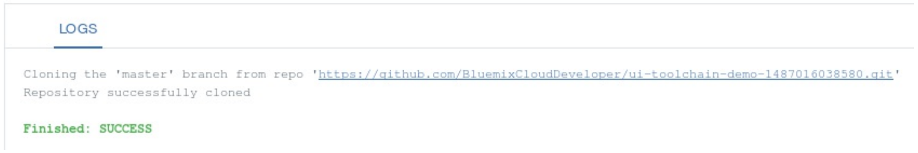
3. While we were busy exploring the toolchain, the various pipelines (remember, we have 3) started the build process. The UI delivery

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.

4. In the Build stage, click **View logs and history**

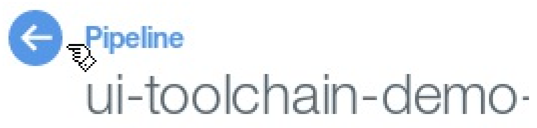


to display the commands and results of the Build stage, in this case simply cloning the repo.



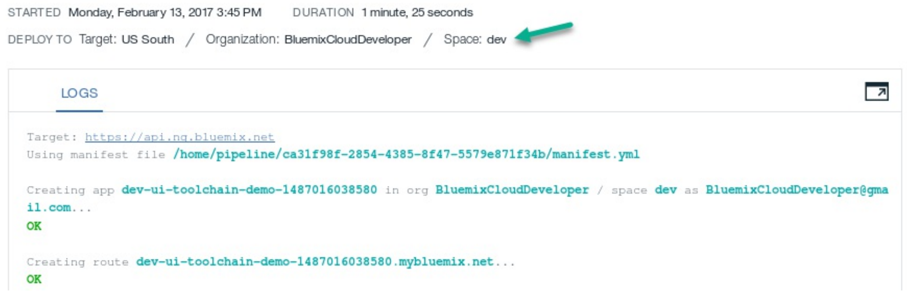
```
Cloning the 'master' branch from repo 'https://github.com/BluemixCloudDeveloper/ui-toolchain-demo-1487016038580.git'
Repository successfully cloned

Finished: SUCCESS
```

Click the arrow to the left of Pipeline



to return to the delivery pipeline.

5. 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.

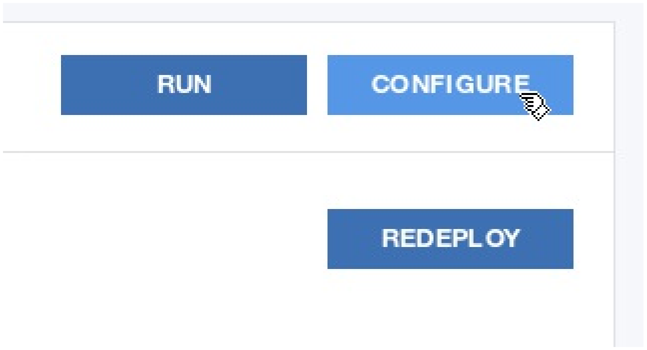


STARTED Monday, February 13, 2017 3:45 PM   DURATION 1 minute, 25 seconds

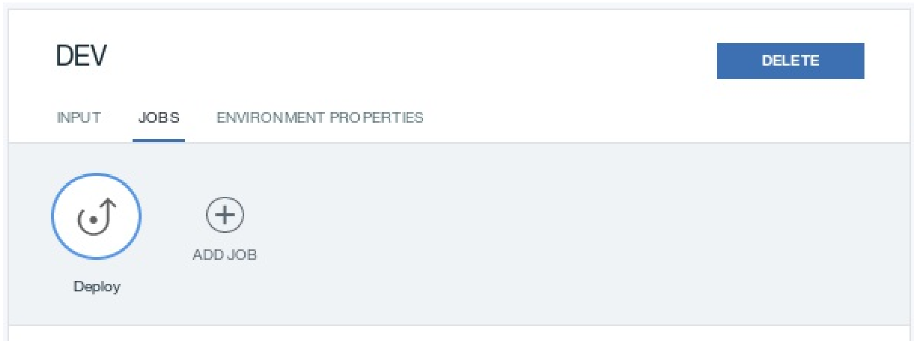
DEPLOY TO Target: US South / Organization: BluemixCloudDeveloper / Space: dev

```
Target: https://api.ng.bluemix.net
Using manifest file /home/pipeline/ca31f98f-2854-4385-8f47-5579e871f34b/manifest.yml

Creating app dev-ui-toolchain-demo-1487016038580 in org BluemixCloudDeveloper / space dev as BluemixCloudDeveloper@gma
il.com...
OK

Creating route dev-ui-toolchain-demo-1487016038580.mybluemix.net...
OK
```

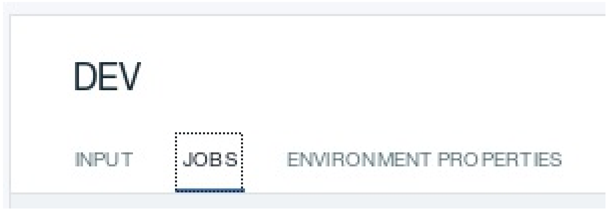
6. If you scroll through the log, you can see all the details of that job.
7. Click **Configure** in the upper right hand corner of the display



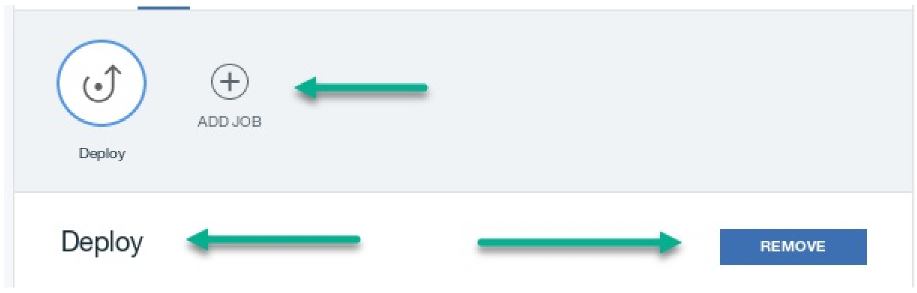
to display the Stage Configuration screen.



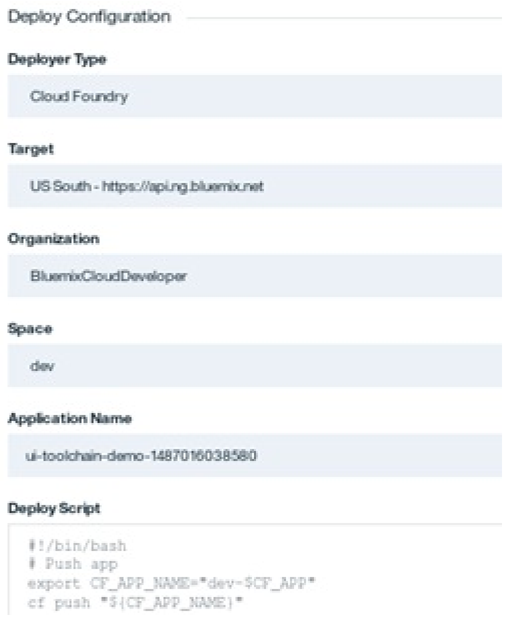
8. The Stage Configuration displays details about the stage. The *INPUT* tab displays the input to the stage, the *JOBS* tabs displays the discrete pieces of the stage, and the *ENVIRONMENT PROPERTIES* tab displays variables used by the jobs in the stage.



9. 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.



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



Deploy Configuration

Deployer Type	Cloud Foundry
Target	US South - https://api.ng.bluemix.net
Organization	BluemixCloudDeveloper
Space	dev
Application Name	ui-toolchain-demo-1487016038580

Deploy Script

```
#!/bin/bash
# Push app
export CF_APP_NAME="dev-$CF_APP"
cf push "${CF_APP_NAME}"
```

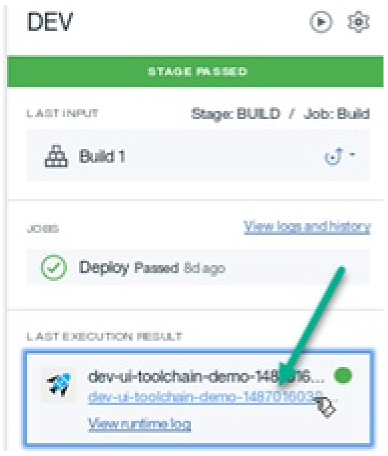
The lines in the Deploy Script:

`export CF_APP_NAME="dev-$CF_APP"`

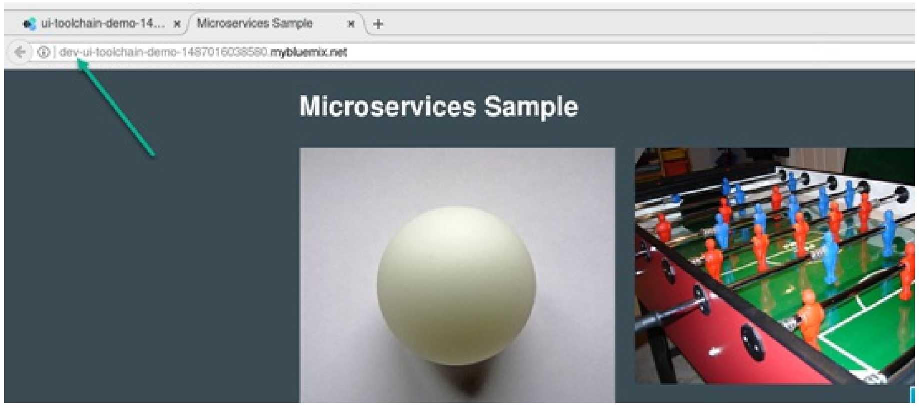
`cf push "${CF_APP_NAME}"`

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

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



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


ui-toolchain-demo-14...
Microservices Sample
dev-ui-toolchain-demo-1487016038580.mybluemix.net
Microservices Sample