Getting Started with IBM Bluemix

Hands-On Workshop

Module 4: DevOps Services

Table of Contents

Copyright IBM Corporation 2013-2015. All rights reserved.

Workshop overview

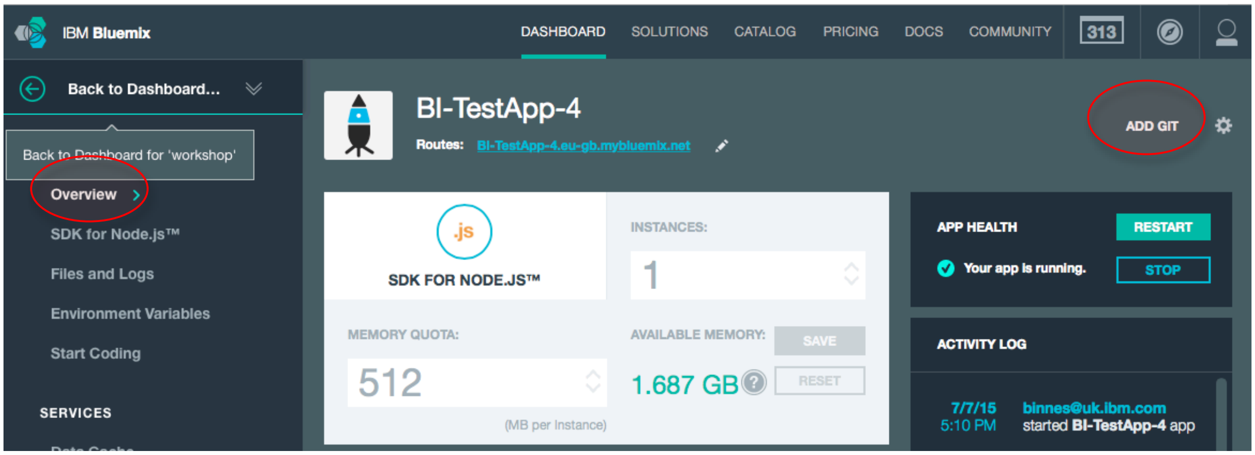
In this workshop, you will:

- Learn how Bluemix and DevOps Services work together
- Commit changes to a Git repository

Exercise 4a: Bluemix integration with DevOps services

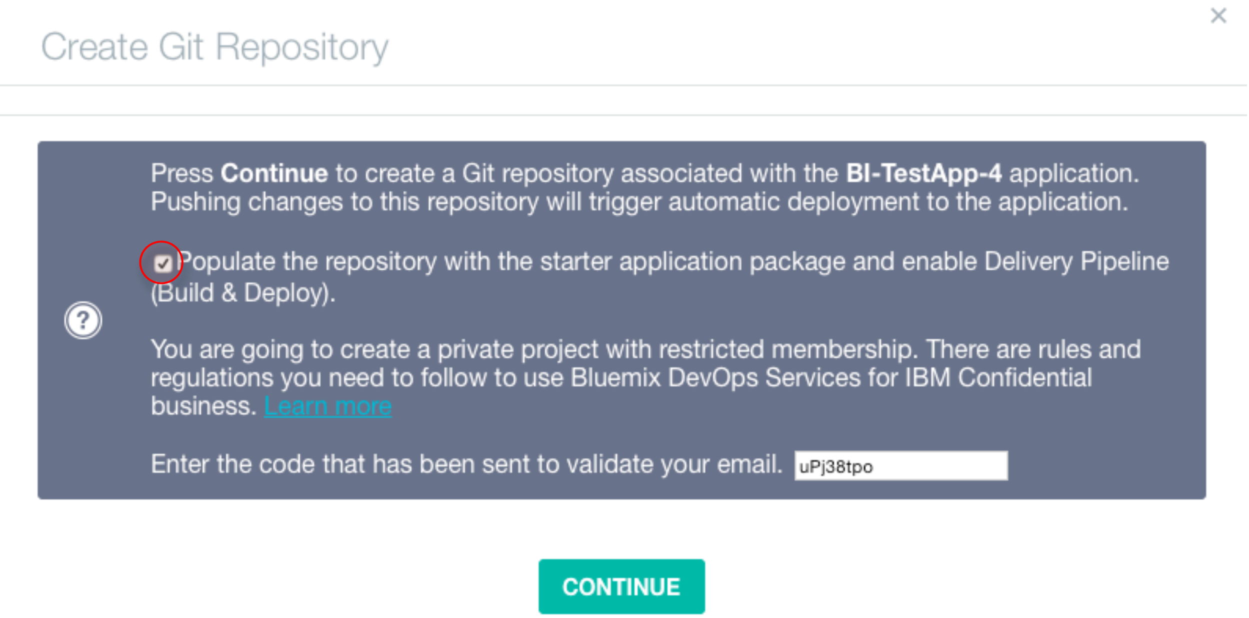
1. Log in to Bluemix http://bluemix.net or https://console.eu-gb.bluemix.net and deploy the Node.js Cache Web Starter boilerplate from the Catalog.

2. After the application is running, select the **Overview** page and then click **ADD GIT** on the Application Overview page.



7/7/15 5:10 PM binnes@uk.ibm.com started **BI-TestApp-4** app

3. Add your DevOps Service details and click **Sign In**. Ensure that you leave the option to populate the repository selected and then click **CONTINUE**.



Create Git Repository

Press **Continue** to create a Git repository associated with the **BI-TestApp-4** application. Pushing changes to this repository will trigger automatic deployment to the application.

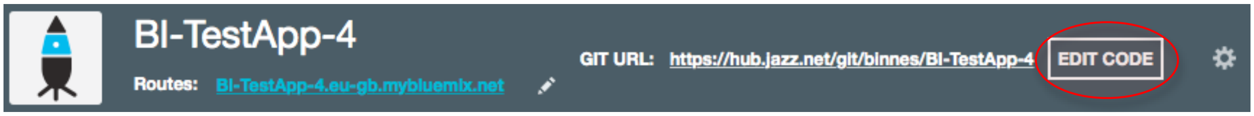
☑ Populate the repository with the starter application package and enable Delivery Pipeline (Build & Deploy).

You are going to create a private project with restricted membership. There are rules and regulations you need to follow to use Bluemix DevOps Services for IBM Confidential business. Learn more

Enter the code that has been sent to validate your email. uPj38tpo

4. On the Overview page, click **EDIT CODE**.

Copyright IBM Corporation 2013-2015. All rights reserved.


BI-TestApp-4
Routes: BI-TestApp-4.eu-gb.mybluemix.net
GIT URL: https://hub.jazz.net/git/binnes/BI-TestApp-4
EDIT CODE

You'll go to DevOps services, and the sample application code will be imported and ready to work with.

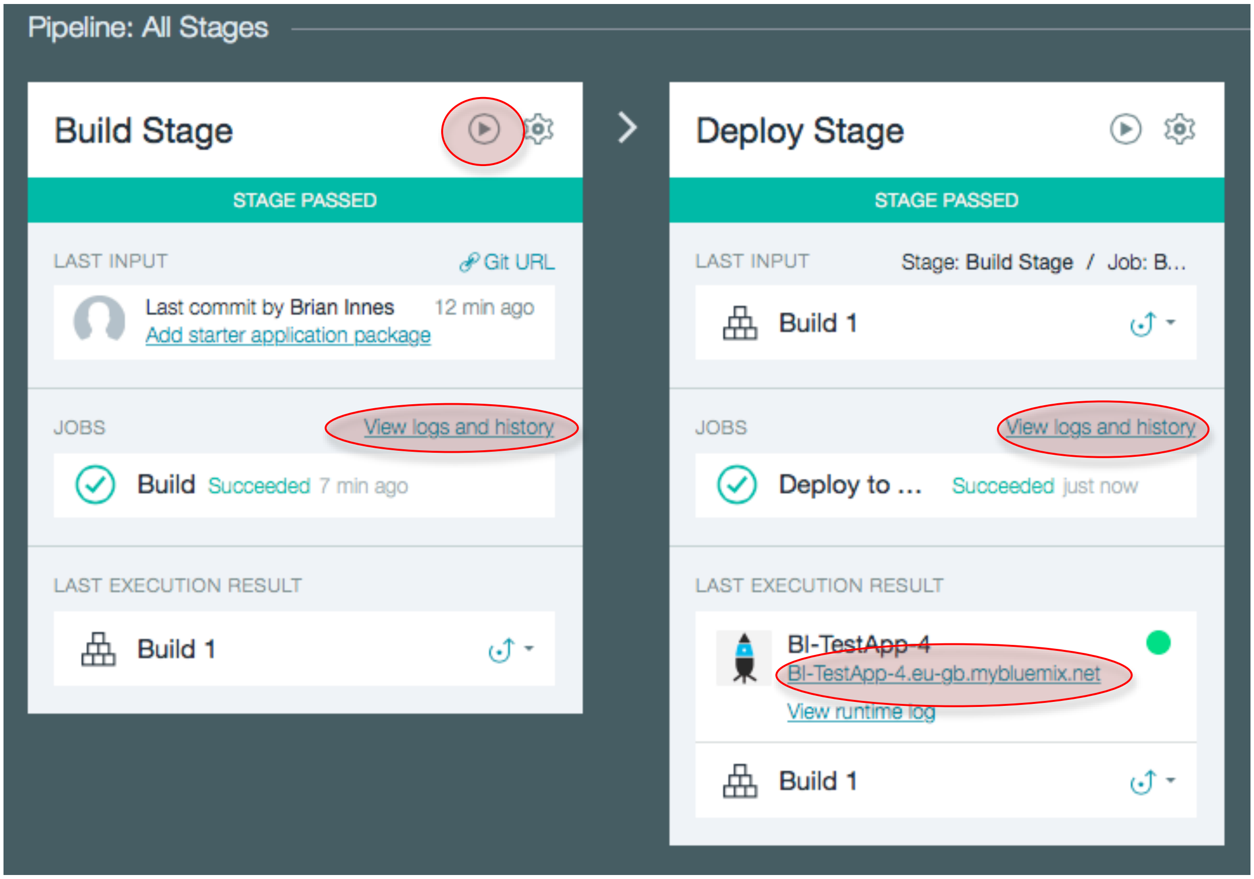
5. Click **BUILD & DEPLOY**. The project is configured to deploy code changes to Bluemix when they are pushed to the Git repository.

You can now modify the code and deploy to Bluemix from DevOps services.

6. To test the build and deployment, click the **Run Stage** (▶) icon in the **Build Stage** section of the pipeline. You see that the build is queued, running, and finally that it succeeded.

After the build has completed, a deploy operation will be queued and run. You should see the deployment running and then succeeding.

7. To launch the application, click the route.

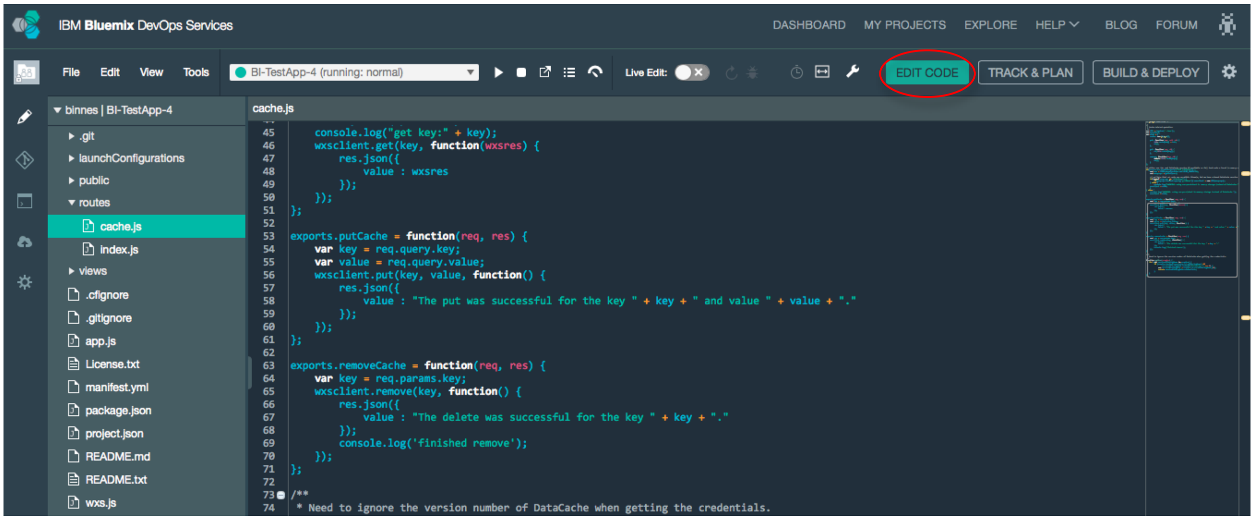


Tip: You can view logs and history for the build and deployment by clicking the links in the Build and Deploy stages.

Copyright IBM Corporation 2013-2015. All rights reserved.

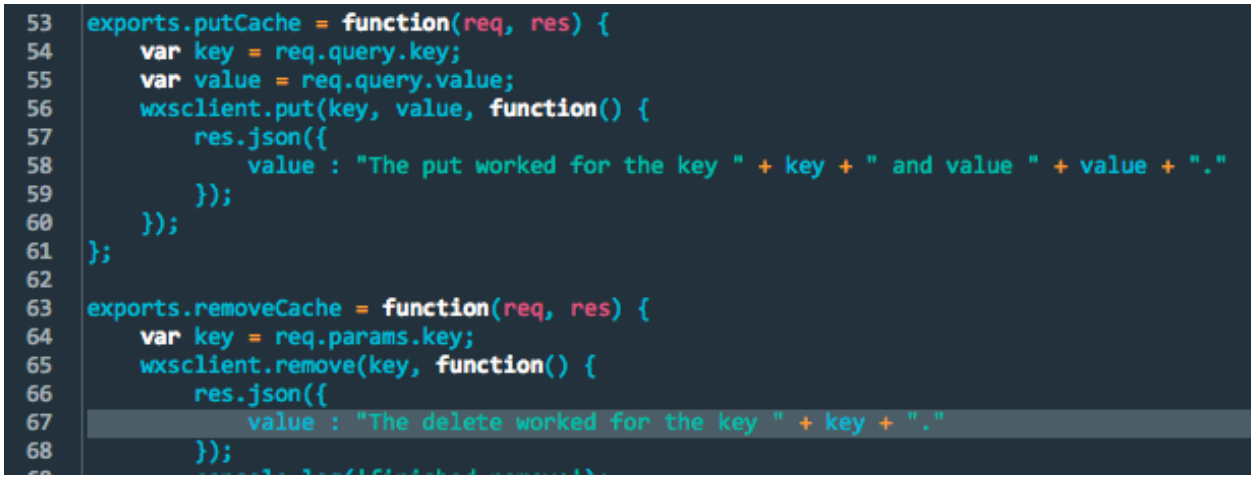
Exercise 4b: Working in DevOps services

1. In DevOps services, switch to the **EDIT CODE** section so that you can edit the code.



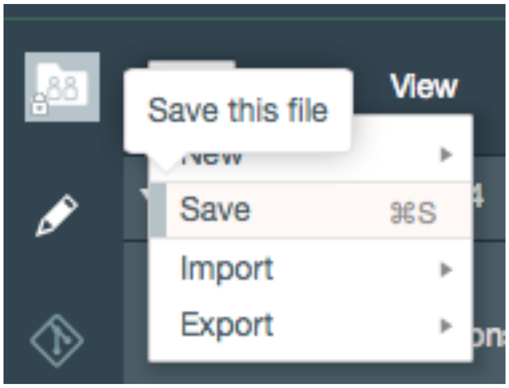
```
45      console.log("get key:" + key);
46      wxsclient.get(key, function(wxsres) {
47          res.json({
48              value : wxsres
49          });
50      });
51  };
52
53  exports.putCache = function(req, res) {
54      var key = req.query.key;
55      var value = req.query.value;
56      wxsclient.put(key, value, function() {
57          res.json({
58              value : "The put was successful for the key " + key + " and value " + value + "."
59          });
60      });
61  };
62
63  exports.removeCache = function(req, res) {
64      var key = req.params.key;
65      wxsclient.remove(key, function() {
66          res.json({
67              value : "The delete was successful for the key " + key + "."
68          });
69          console.log('finished remove');
70      });
71  };
72
73  /**
74   * Need to ignore the version number of DataCache when getting the credentials.
```

2. Open the `routes/cache.js` file and scroll to line 58 to find the two strings that are displayed on a successful put ("The put was successful …") or delete ("The delete was successful …") and change the strings to new messages:



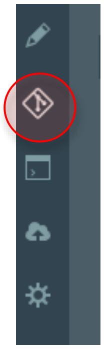
```
53  exports.putCache = function(req, res) {
54      var key = req.query.key;
55      var value = req.query.value;
56      wxsclient.put(key, value, function() {
57          res.json({
58              value : "The put worked for the key " + key + " and value " + value + "."
59          });
60      });
61  };
62
63  exports.removeCache = function(req, res) {
64      var key = req.params.key;
65      wxsclient.remove(key, function() {
66          res.json({
67              value : "The delete worked for the key " + key + "."
68          });
```

3. Save the changes by clicking **File > Save**.

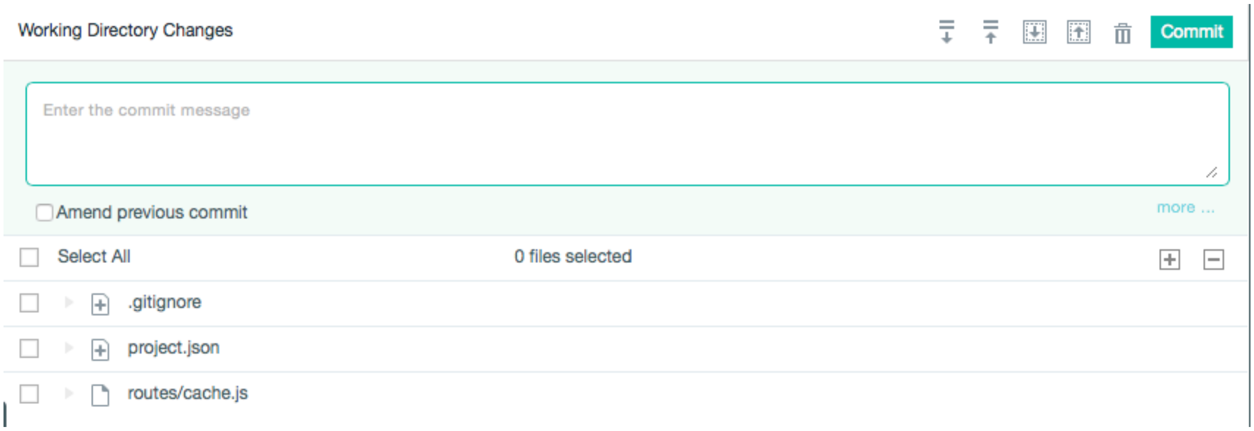


To get the changes pushed to Bluemix, commit the changes to the Git repository.

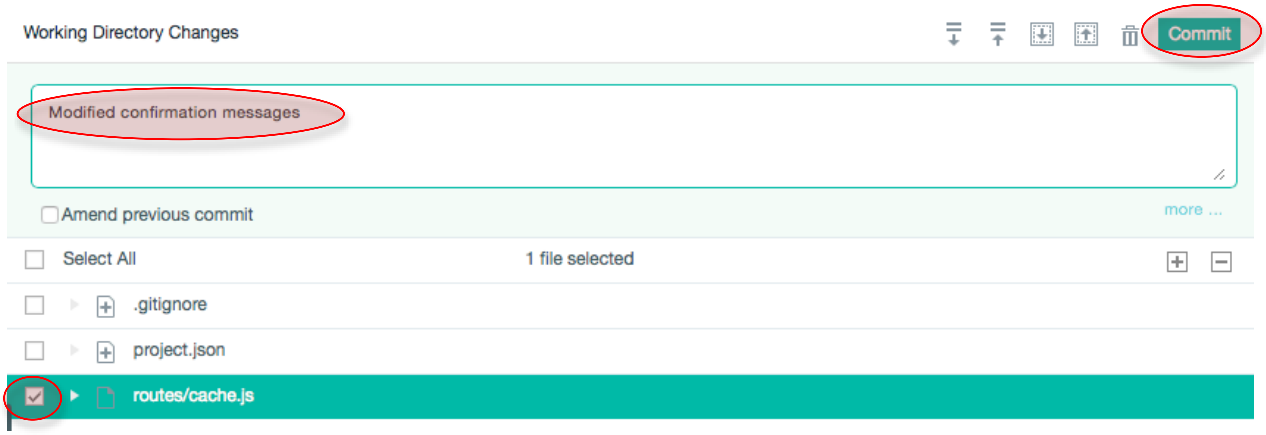
4. To work with the Git repository, switch into the Git section.



The files that have local changes are shown in the **Changed Files** section. When a project is first added to DevOps services, some configuration files are updated automatically, in this case, the first 3 files in the list.



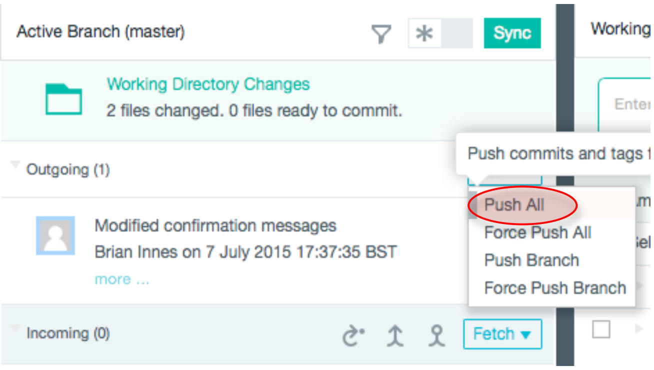
5. Select the JavaScript file, add a commit message, and then click **COMMIT**.



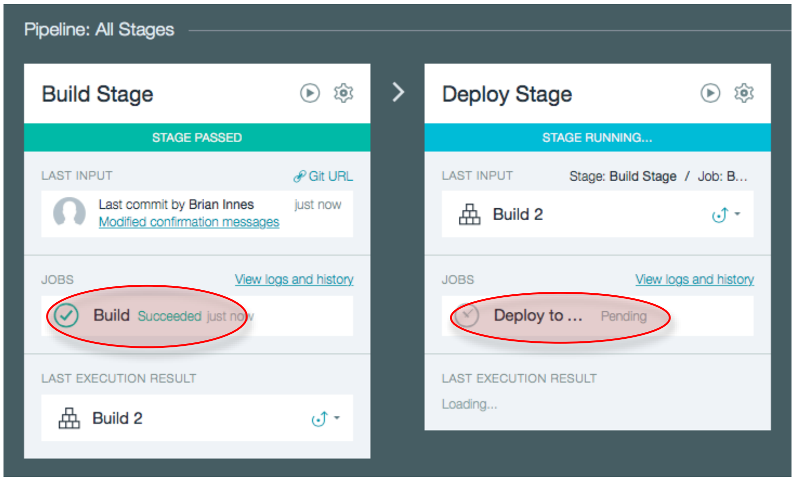
The commit was made to the local branch. The builder works from the remote branch, so you must push the changes back to the remote branch.

6. In the left column, expand the **PUSH** menu and click **Push All**.

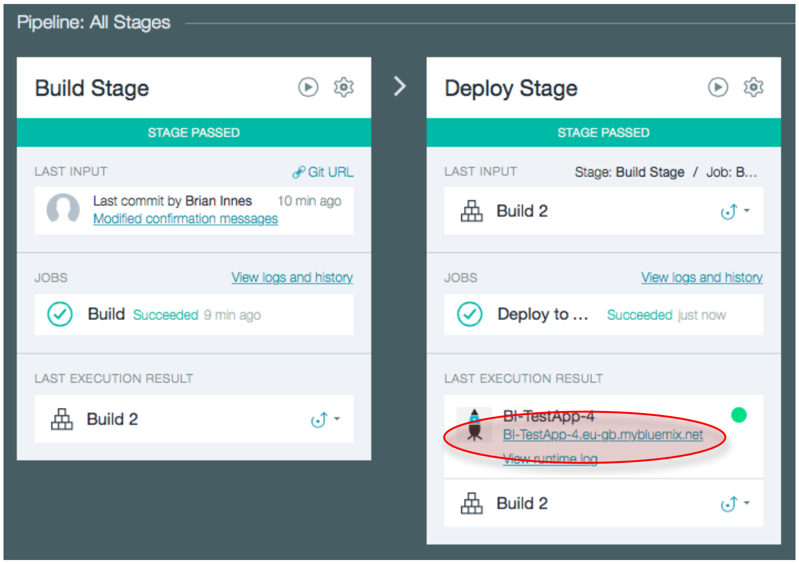
Copyright IBM Corporation 2013-2015. All rights reserved.



If you quickly jump to the BUILD & DEPLOY window, you can see that a build has automatically started, which will then be automatically deployed to Bluemix if the build is successful.



7. After the deploy is completed, test the application to verify that the code changes are running. Launch the application by clicking the link in the deploy stage.



Tip: If a build fails, click the build in the Builder where you can see a detailed build history. You can also access the logs, files, and details of changes included in the builds. The build history and logs can help you determine why a build failed.

Copyright IBM Corporation 2013-2015. All rights reserved.