IBM

InterConnect 2016
The Premier Cloud & Mobile Conference

Session: 5355

**Turbocharge your development with
IBM DevOps Services Toolchains**

Lab Instructions

Authors:

Ritchie Schacher
Sachin Patel
Evan Lobeto

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

February 2016 edition

NOTICES

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

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

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

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

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

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

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

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

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

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

TRADEMARKS

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

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

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

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

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

ITIL is a Registered Trade Mark of AXELOS Limited.

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

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

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

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

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

Resource guide

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

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

- Arrow

- Avnet

- Global Knowledge

- Ingram Micro

- LearnQuest

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

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

- IBM Training website

  – Bookmark the IBM Training website for easy access to the full listing of IBM training curricula. The website also features training paths to help you select your next course and available certifications.

  – For more information, see http://www.ibm.com/training

- IBM Certification

  – Demonstrate your mastery of IBM products to your employer or clients through IBM Professional Certification. Certifications are available for developers, administrators, and business analysts.

  – For more information, see http://www.ibm.com/certify

InterConnect 2016
The Premier Cloud & Mobile Conference

IBM

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

Lab Overview

DevOps practices require automated processes for the entire application lifecycle, from design through the development process to production support, involving heterogeneous tools assembled into a toolchain. This is a hands-on lab, as a companion to the sessions on toolchains and the IBM Bluemix Garage Method (https://www.ibm.com/devops/method). In this lab, you will explore IBM Bluemix DevOps Services and create a Build and Delivery Pipeline, that integrates with several third-party tools to automate the processes of build, test, deploy, notification, and monitoring of a sample Cloud Native application. The instructions will show you how to easily copy an existing project with a pre-defined pipeline, and walk you through the customized integrations and show you how to configure these integrations.

Lab Procedure
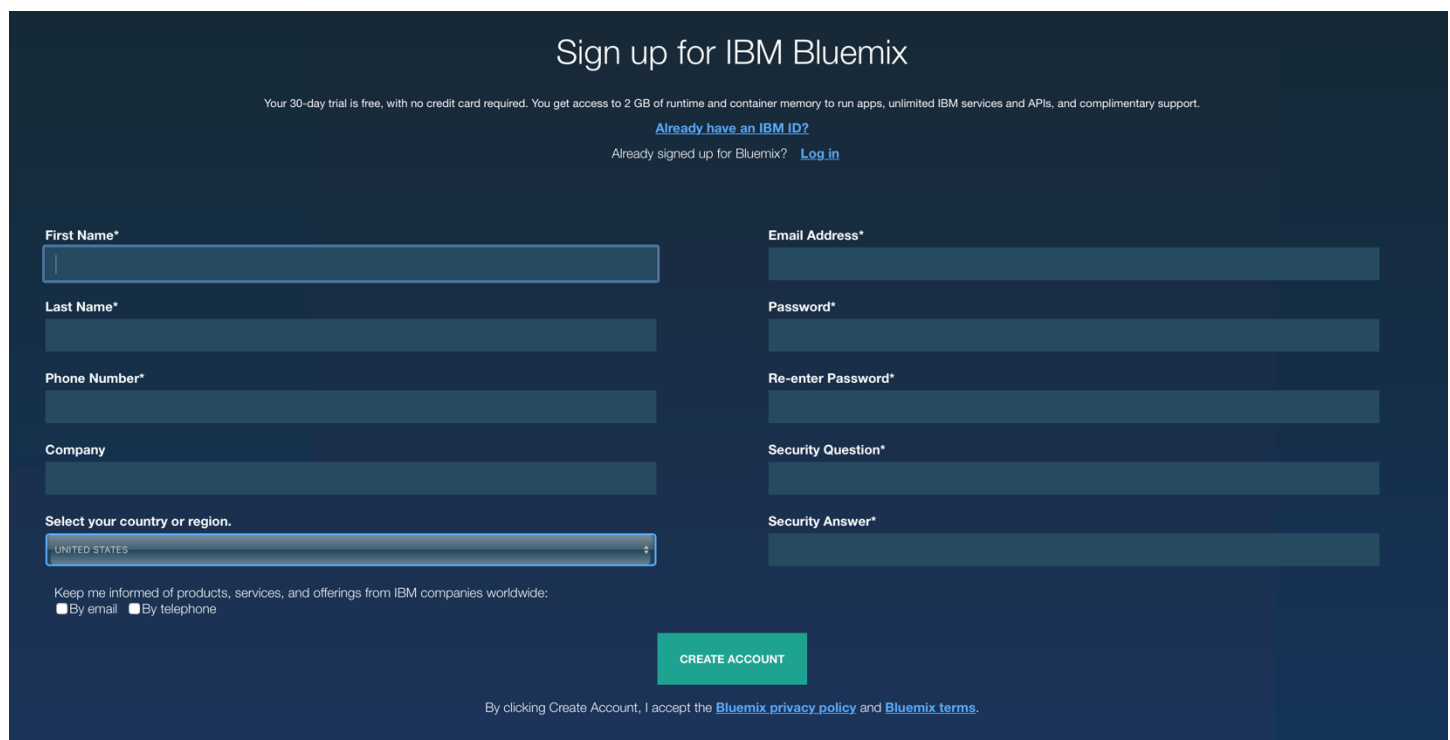
The website for the lab with useful links, PDF copy of the lab instructions, and contact info can be accessed at http://interconnect5355.github.io.

Step 1 - Configure Prerequisites

The following accounts need to be setup:

- A Bluemix account. Bluemix is a cloud-based software development platform. It requires an IBM id. If you don't have one, you will create it when you clone the sample app.

    https://console.ng.bluemix.net/registration/

During this process you will be asked to confirm your account creation via a link sent to the email address provided, and re-log into Bluemix.

For the accounts below you can create a new account, use an existing account or use the provided common accounts (below) to be shared by all lab attendees. This is recommended in the interest of time.

The following shared accounts are available for use during the lab session. Set this page aside to easily reference the account info and credentials.

Note: The shared accounts will be closed 3 days after the Interconnect conference ends.

Account	ID	Password	API-Key
Sauce Labs	interconnect5355	devops2016	9489b7ea-a5cb-49e5-a774-4821641809e5
SpeedCurve	interconnect5355 @gmail.com	devops2016	k7gdzfzxhyxxq0385jucz0oz8y5a24
New Relic	interconnect5355 @gmail.com	devops2016	2181f2ac6c0c021cdd84a4bacd45dc930 82034f9
Load Impact	interconnect5355 @gmail.com	devops2016	71322ae0c91908a9494f5738ee07c6170f a3a5c77d05ec21cfac8dbc29c75607
Slack	Team: interconnect5355 Email: Interconnect5355 @gmail.com User: interconnect5355_ user1	devops2016	xoxp-20972150389-20969690548-21571532949-7ad49d2cc7

Information and links to creating new accounts for the integrations used in this lab are:

- A **Sauce Labs** account and access key. Sauce Labs is a cloud-based functional testing platform. Use it to ensure that your apps are working as intended.

  https://saucelabs.com/signup/trial

- A **SpeedCurve** account and API key. SpeedCurve provides performance of your website, such as page size and how long it takes to load a page, and can show you the impact new features have on the performance of your website. It also allows you to set a performance budget so you can see and track when the performance of the website begins to lag.

  https://speedcurve.com/plan/trial/

- A **New Relic** account and license key. New Relic alerts you to app downtime before your users encounter problems so that you can discover and resolve problems in a timely manner.

  https://newrelic.com/signup

- A **Load Impact** account and API key. Load Impact tests your apps for scale and performance in varying geographies.

  https://app.loadimpact.com/account/register

- A **Slack** team that you have administrative rights to. Slack is a collaboration tool that can provide visibility into everything that is happening in your project, from code check-ins to deployment status, to errors on your website. Slack provides integrates with many third-party tools that are commonly used for application development.

  https://slack.com/create

Step 2

Clone the sample app by visiting the URL below:

https://bluemix.net/deploy/index.html?repository=https://github.com/oneibmcloud/devops-tutorial-1.git

If you haven't logged in yet, or if your session is expired, click **Log In**.
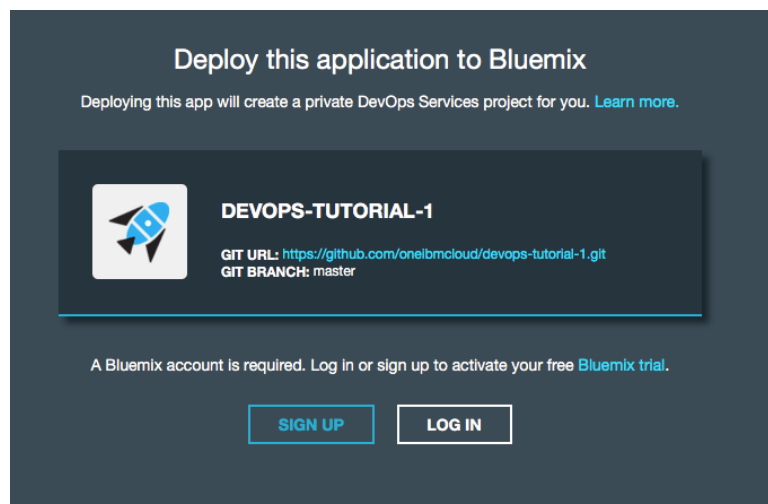
After you sign in to Bluemix you may be prompted to pick an alias if this is your first time using Devops Services.

Afterwards type your app's name; select your region, organization, and space; and click **Deploy**.

After a few seconds, the system will begin to create your own DevOps services project and your pipeline. Additionally, it copies the files from the sample project into a Git repository hosted by DevOps Services. The build and delivery pipeline will ultimately deploy your cloud-native application to Bluemix.

Note, however, that DevOps Services also supports integration with GitHub for projects created from scratch.

To enable the additional capability of the integrated tools in your pipeline, you will need to change a few things. You'll do this later in Step 5.

Step 3

As soon as the **Configured pipeline successfully** link becomes active, click it. You do not need to wait for the **Deploying to Bluemix** step to complete.
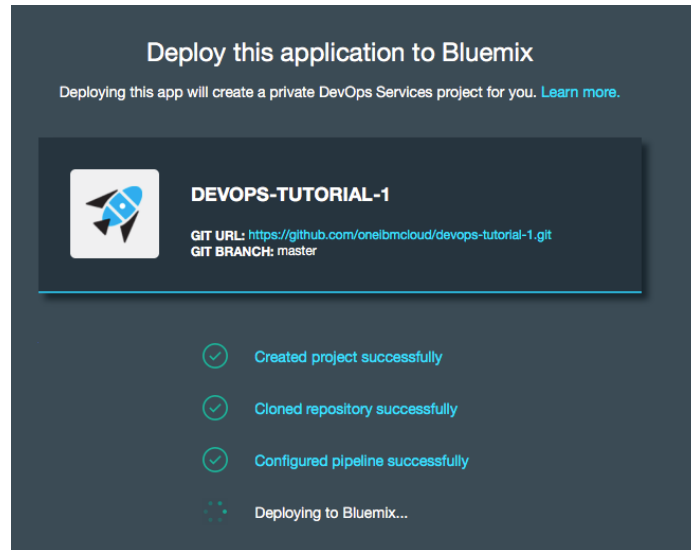
You will see all five stages below. If you don't see them, wait a little while and refresh your browser. Depending on the size of your window you may need to click on the scroll action to see any hidden stages.

1. Build
2. Run Unit Tests
3. Deploy to Test
4. Run Tests - Performance & Scale
5. Deploy to Prod
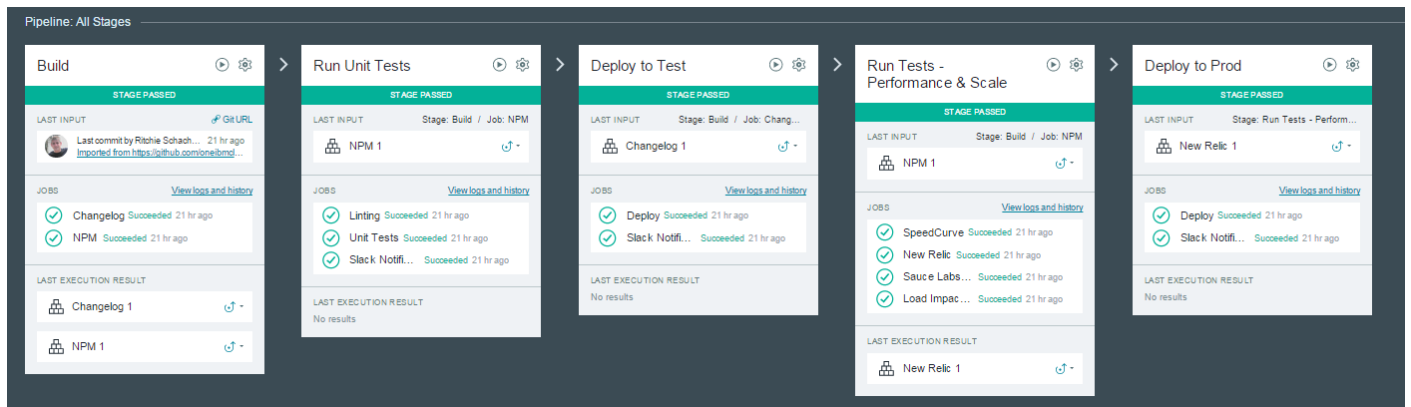
Step 4

To learn a bit more about the pipeline, watch the two minute introduction video displayed at the top of the pipeline when displayed for the first time. If you've closed it, it's accessible by the help icon or by going directly to:

https://www.youtube.com/watch?v=3OHMrr8iz5E&feature=youtu.be

Explore your project. Develop in the Web IDE, which you can open at any time by clicking the **EDIT CODE** button at the top of the page. The Web IDE will create your own personal virtual workspace in the cloud, based on a copy of the contents in your Git repository. You can use the Web IDE to code in your browser and make quick test deployments to Bluemix. At any time during development, you can also commit your changes back to the master branch and trigger automated builds and deployments. You will do this later in the exercise.

Documentation on the web based IDE can be found at: https://hub.jazz.net/docs/edit/
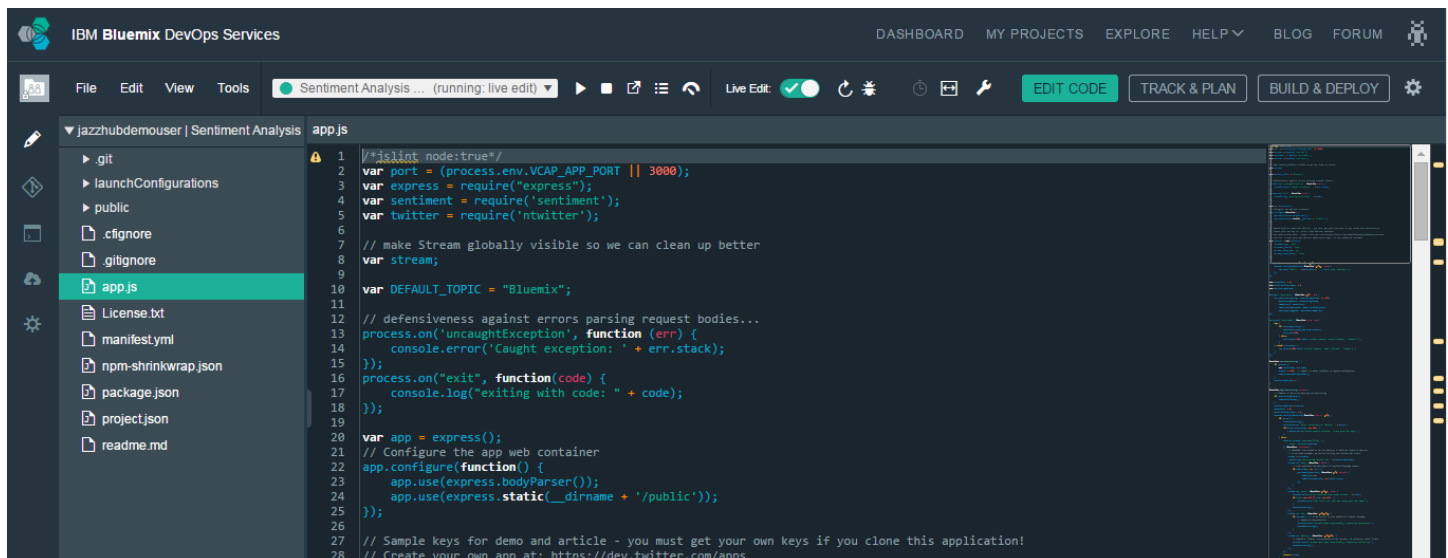
Step 5

The sample project provided all of the information that Bluemix needed to generate a nearly functional pipeline for your app. However, before all of the jobs can run successfully, though, you need to edit a few things. To see the pipeline, click **BUILD & DEPLOY.**

Notice the jobs that incorporate tools like Sauce Labs and SpeedCurve to test your app, and Slack to provide notifications of successful deployments. We'll modify the configuration for these stages to get these integrations in working order.

Note: If you do not want to provide API or license keys for these tools, the pipeline for your sample project can still run successfully. The sample project's pipeline is configured so that all integrations are optional. If you don't configure the integrations, the stages won't fail. However, in a real-world project, you may find the integrations are helpful and improve the rich functionality of your pipeline(s).

Step 6 - Build Stage

In this step we will start by looking at the configuration for first stage of the pipeline, **Build**.

1. Click the **Stage Configuration** icon, which looks like a gear.
2. Click **Configure Stage**.

You don't need to do anything in this particular stage—it's already configured for you—but take a look around.
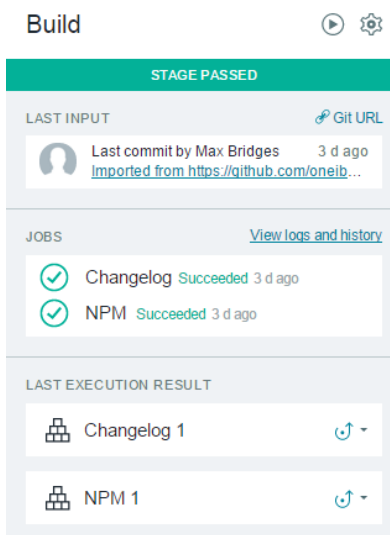
On the **INPUT** tab, notice that the master branch of your project's Git repository is providing the input for this stage.

3. Click on the **JOBS** tab.

The first job, Changelog, takes the stage input, adds a log of the changes since the previous build, and archives the results. This archive is used later in the pipeline for deployment to a production environment.

The second job, NPM, takes the stage input, installs the dependencies that are defined in the project's Node package file, and archives the results. These dependency files are used later by tests that are part of the pipeline.

Each job script is customizable. You can explore the various scripts throughout the stages to see what the individual jobs may be doing.

When you're finished looking around, at the bottom of the page, click **CANCEL**.

Step 7 - Unit Tests
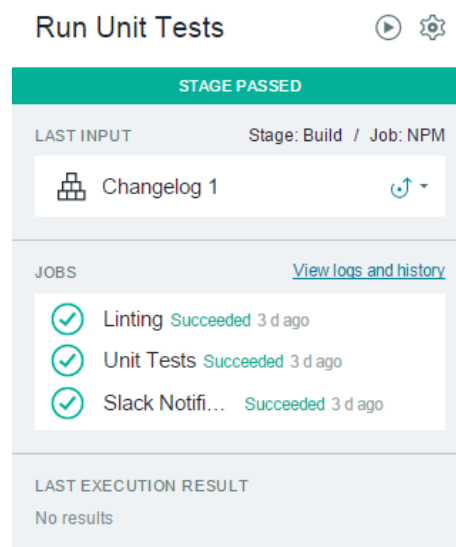
The second stage in the toolchain contains the following jobs and supplies them with the project's source control repository as input:

-   Linting, which runs open-source linters (JavaScript and CSS) on the project's code.
-   Unit Tests, which runs Karma and Mocha unit tests against the project's front-end, server, and API.
-   Slack Notification, which sends a notification to a Slack channel after all of the other jobs are completed. You'll define the Slack channel to send notifications to later.

The simplest possible pipeline consists of a stage with a build job in it and a stage with a deploy job in it. The code is compiled or archived, and then it is deployed to Bluemix. However, by implementing a pipeline like the one in this lab, you bring automated testing, quality control, and team notifications to a project.

You can inspect and modify the configured stages and jobs by clicking the **Stage Configuration** icon on the stage tile, and then clicking the **JOBS** tab. The linting and unit tests jobs use the Gulp build system to run predefined tests on the input from the Git repo. The sample project includes all of the dependency files that the tests need to run. To add more tests, see README.md in the project's test directory.

Step 8 - Slack Notifications

Because jobs run sequentially, the Slack Notification job posts a message to a Slack channel when all of the other jobs in the stage are completed. This job requires a valid incoming Slack webhook, which is not provided in the sample, and a message, which is provided. Both of these are set as part of the stage's environment properties.

1. On the Run Unit Tests stage tile, click the **Stage Configuration** icon and click **Configure Stage**.
2. Click **ENVIRONMENT PROPERTIES**.
3. Set the value of SLACK_WEBHOOK_PATH to a valid incoming Slack webhook. The Slack Notification job accesses the environment property by using this syntax: `${SLACK_WEBHOOK_PATH}`. If you haven't already generated a webhook, you can generate one from: https://my.slack.com/services/new/incoming-webhook/. Create a new channel in the format **ic5355_<initials>** where initials are your initials, and select that channel when creating a new webhook. For more information, visit https://api.slack.com/incoming-webhooks.

4. Click **SAVE.**

Run Unit Tests		DELETE
INPUT JOBS ENVIRONMENT PROPERTIES		

| + ADD PROPERTY	
APP_NAME	devops-tutorial-1-mbridges-1610
SLACK_WEBHOOK_PATH	https://hooks.slack.com/services/IBM/GARAGE/METHOD
SLACK_MESSAGE	Build done & all tests passed.

SAVE   CANCEL

Throughout the remainder of this lab, log into slack and open the channel you created (using the quick switcher at the bottom left of the window) to keep an eye out for incoming notifications.
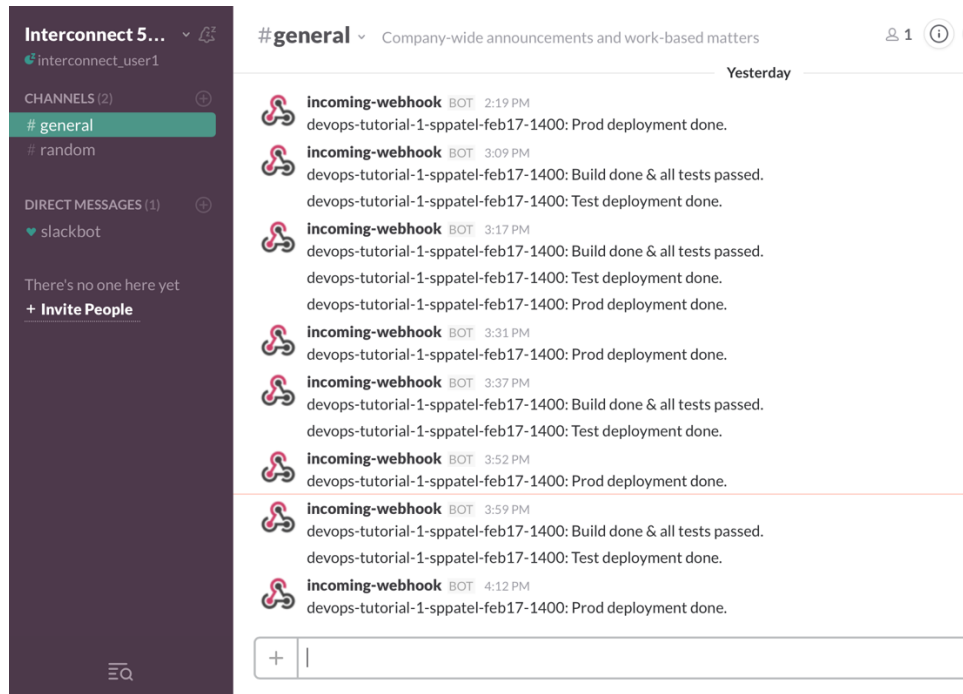
Step 9 - Deploy to Test
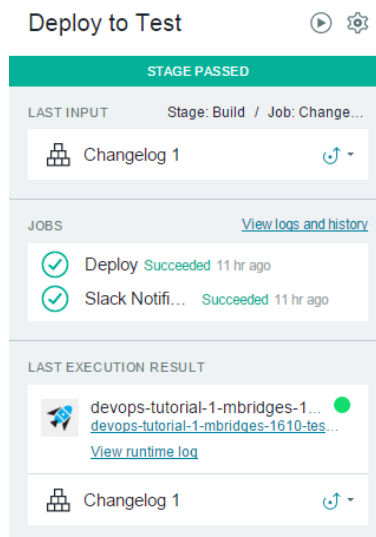
When you configure a pipeline, it is good practice to create a stage that you use to deploy to

an environment for tests or "staging." A staging environment mirrors the production environment. In the staging environment, you can run tests to find and fix errors before you deploy changes to the production server.

The "deploy to test" stage takes the build that the Changelog build job supplies in the first stage, and deploys a test instance by using a "blue-green" deployment technique. To see how this is done, view the configuration of the stage, and view the Deploy Script section of the Deploy job. After the blue-green deployment, a job then sends a deployment notification through Slack. You only need to add an environment property value for that notification job to be completed.

1. On the "Deploy to Test" stage tile, click the **Stage Configuration** icon and click **Configure Stage**.
2. Click **ENVIRONMENT PROPERTIES**.
3. Set the value of SLACK_WEBHOOK_PATH to a valid incoming Slack webhook. The Slack Notification job accesses the environment property by using this syntax: ${SLACK_WEBHOOK_PATH}. Feel free to reuse the webhook from the previous stage, or create a new channel and matching webhook.
4. Click **SAVE**.

After the stage completes, the stage will display a "Last Execution Result". This section of the stage card provides the status details of your running application along with which version of the build input the pipeline used to deploy the application. Within the "Last Execution Result" you can navigate directly to the Bluemix dashboard of the application (shown below) as well as the application itself.

Step 10 - Run performance and scale tests

Run Tests -
Performance & Scale

STAGE PASSED

LAST INPUT   Stage: Build / Job: NPM

Changelog 1

JOBS   View logs and history

✓ SpeedCurve Succeeded 11 hr ago
✓ New Relic Succeeded 11 hr ago
✓ Sauce Labs... Succeeded 11 hr ago
✓ Load Impac... Succeeded 11 hr ago

LAST EXECUTION RESULT

New Relic 5
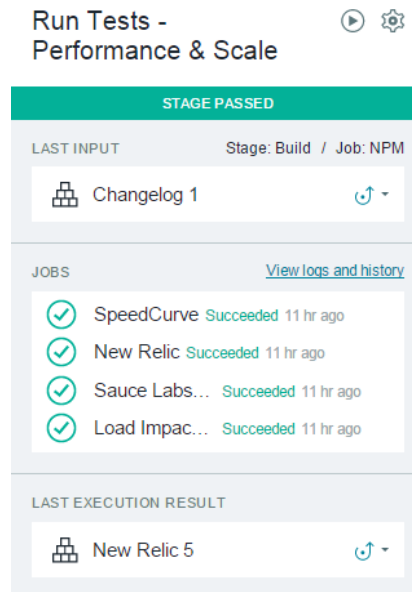
A complete toolchain allows your project to grow effortlessly in popularity and scale. This next stage helps to ensure that the changes that were made in your most recent builds don't harm the website's performance or reliability. This stage contains jobs that run functional tests by using Sauce Labs; trigger SpeedCurve and Load Impact, which are performance monitoring tools; and install an agent for New Relic, which is a website-monitoring tool. All of these jobs have already been pre-configured with scripts. You only need to edit the values of the stage's environment properties.

To configure the stage:

1. On the Run Tests - Performance & Scale stage, click the **Stage Configuration** icon.
2. Click **Configure Stage**.
3. Click **ENVIRONMENT PROPERTIES**.
4. For APP_NAME, leave the default value as is.
5. Provide a value for NEW_RELIC_LICENSE_KEY
6. Add the website that you deployed in the previous stage to your SpeedCurve account. You can find the URL of that site in the Last Execution Result section of the previous stage.
7. Provide a value for SAUCE_USERNAME.
8. For SAUCE_ACCESS_KEY, enter your Sauce Labs access key.
9. Provide a value for LOAD_IMPACT_API_TOKEN.
10. For the value of LOAD_IMPACT_TEST_URL:
    a. Create and run a test in Load Impact.
    b. Run the test against the site that was deployed from the Deploy to Test

stage. At the bottom of that stage, you can find the site URL under the "Last Execution Result" heading. Note, you may have to refresh the page to see this.

c. Go to the Load Impact website, and under Tests, click the test that you just created.

d. Copy the URL for that test; for example:

https://app.loadimpact.com/tests/3202434

e. Take the numbers at the end of that URL and substitute them for id in this URL:

https://api.loadimpact.com/v2/test-configs/{id}/start

For example, the test at https://app.loadimpact.com/tests/3202434 would translate to:

https://api.loadimpact.com/v2/test-configs/3202434/start

11. Click **SAVE**

Run Tests - Performance & Scale		DELETE
INPUT JOBS ENVIRONMENT PROPERTIES		

+ ADD PROPERTY

APP_NAME	devops-tutorial-1-mbridges-1610
NEW_RELIC_LICENSE_KEY	in102n59cnpaopsm01029590
SPEED_CURVE_API_KEY	1802mmQz1209
SAUCE_USERNAME	max
SAUCE_ACCESS_KEY	e833d9c8-e527-44d6-ab3e-45175461fb85
LOAD_IMPACT_API_TOKEN	1023901239aaeafaplx
LOAD_IMPACT_TEST_URL	https://app.loadimpact.com/tests/320u2434a
APP_URL	http://devops-tutorial-1-mbridges-1610-test.mybluemix.net/

+ ADD PROPERTY

SAVE   CANCEL

Step 11 - Push to production
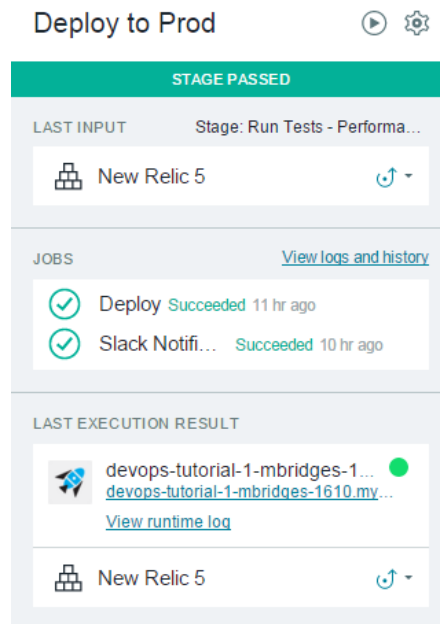
The final stage takes the output from the New Relic job in the previous stage, deploys it to a Bluemix production instance, and sends a deployment notification to a Slack channel. The New Relic website monitoring agent that was installed in the previous stage becomes active after the app is deployed in this stage.

The stage is mostly configured for you in the sample: you need an incoming webhook URL for Slack.

1. On the Push to Prod stage, click the **Stage Configuration** icon.
2. Click **Configure Stage**.
3. Click **ENVIRONMENT PROPERTIES**.
4. For the value of the SLACK_WEBHOOK_PATH property, type an incoming webhook for Slack.
    a. This webhook can be the same as the webhook that you used in the first stage, or you can use a different webhook. For example, you might have different Slack channels for build status and deployment status.
5. You can leave SLACK_MESSAGE as-is, or edit it to your liking.
6. Click **SAVE**.

At this point you should now have a fully functional sample toolchain!

Step 12 - Validate the pipeline

Near the top of the Build stage tile, click the **Run Stage** icon. That stage and anything after it are manually triggered. As the pipeline runs, its jobs will succeed and its stages will pass. This process will take between ten and twenty minutes. In the meantime, feel free to explore the pipeline. You can click on any job as it runs and see its logs in real time.

Additional documentation on the pipeline can be found at: https://hub.jazz.net/docs/deploy/

Now that your pipeline is running, consider the power of the toolchain. Browse to Sauce Labs to see the results of functional tests that were run against the app, or look at the app's performance on your SpeedCurve dashboard. If you open your Slack team's notification channels, you'll see that your builds and deployments were announced immediately after they finished. You can even configure New Relic to send you alerts based on your app's status.

On **Sauce Labs**, you can login and navigate to the test results by clicking the *example with chrome* link from the Dashboard page. This page will show more details about what is going on in the test, such as the HTTP requests and how the app responded to them. You can even see a video of the test's execution.

In **SpeedCurve**, after logging in you will see a dashboard page showing the median page load times for various parts of the app. If you click on the Deploy, Responsive, or Assets tabs in the upper right you can find even more information about how your app performs. This can help you improve and monitor changes when making code edits that deal with CSS loading, JavaScript rendering and more.
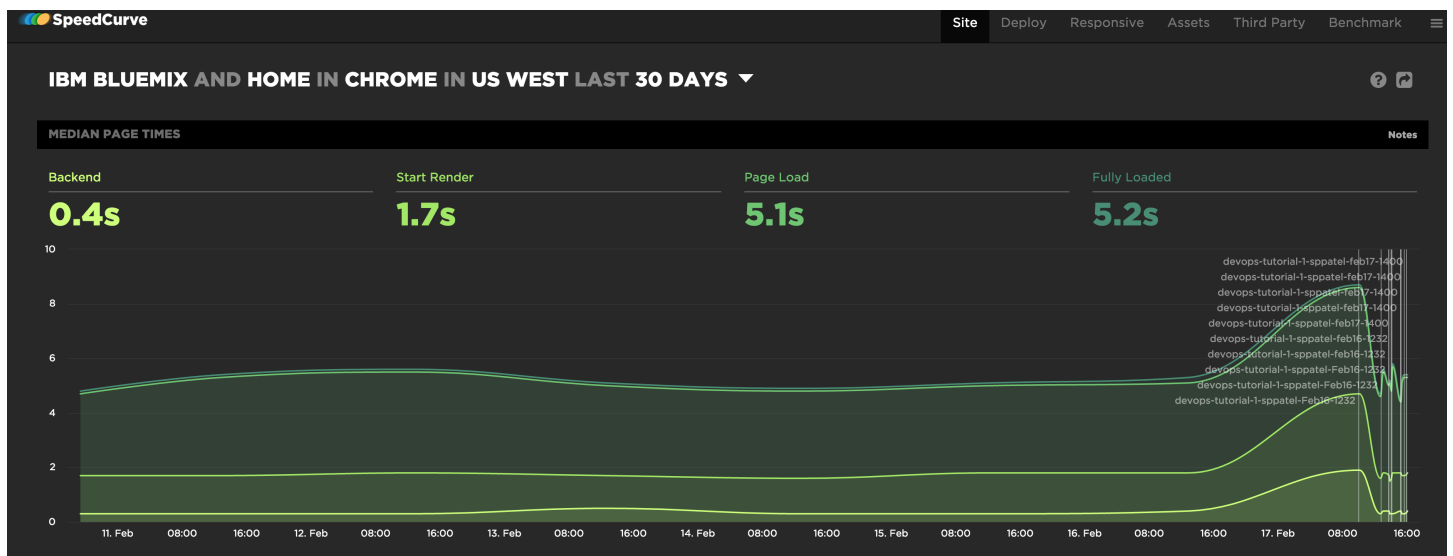
Load Impact is similar to SpeedCurve in that it tests app performance, but it tests how fast your app responds depending on the number of users. Here, after logging in, click on Tests on the left hand side and select your test. If you click on results from this page, you will see a

graph displaying the average page load time vs. the number of connections to the app. This is useful in determining how well your app scales across increasing numbers of users.

Metrics (2)	URLs (3)	Pages (1)	Logs

+ Show details ⚙

● VU load time   ● VUs active
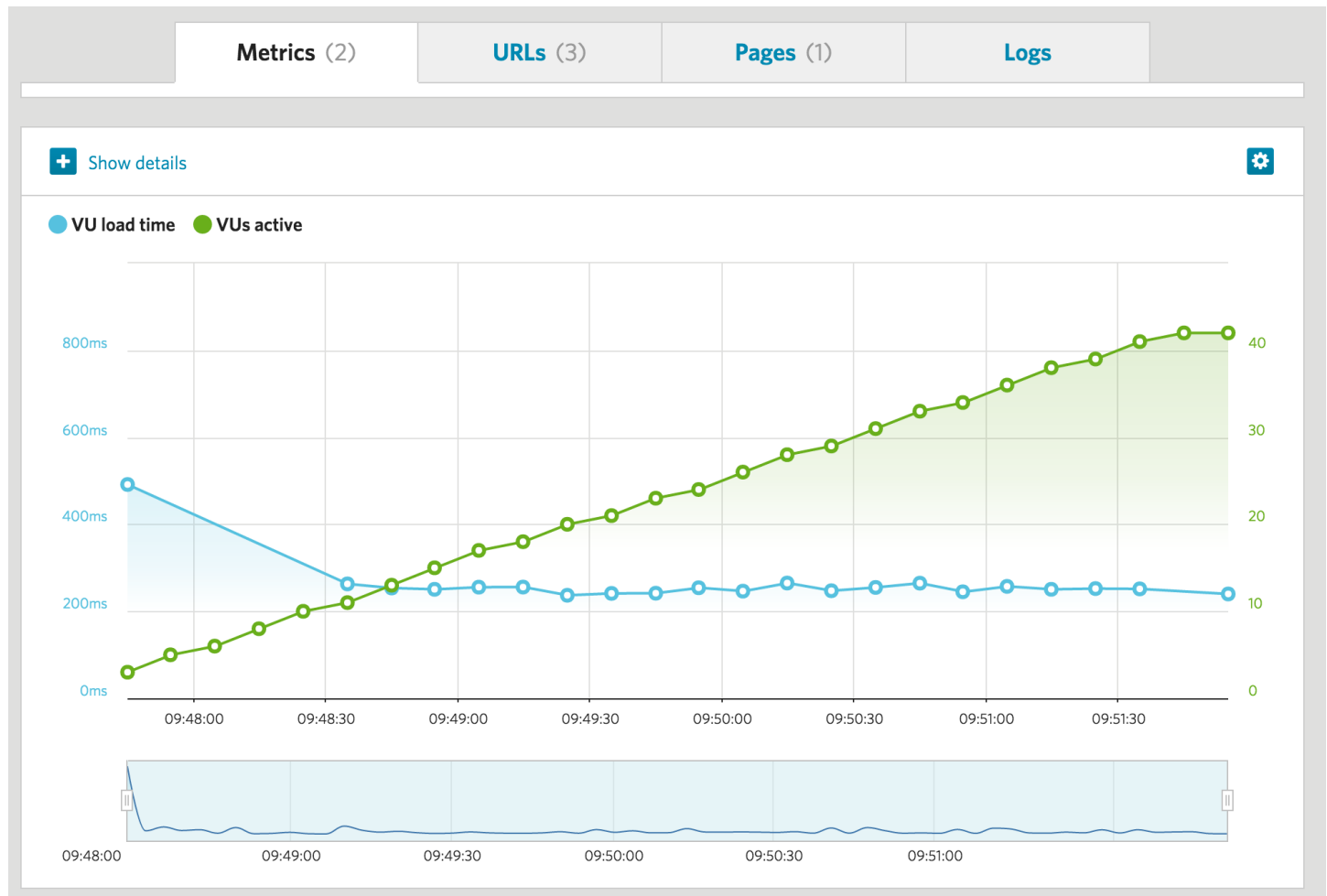
Step 13 - Update the project

Now that you have a functional sample toolchain, you're going to break it and learn about the DevOps Services Web IDE in the process.

By using the Web IDE, you can code in your browser and make quick test deployments to Bluemix. You can open it from anywhere in DevOps Services by clicking **EDIT CODE**.

To begin, in the upper-right corner, click **EDIT CODE**. Open the front-end unit tests file and edit it:

1. In the directory, expand **api** > **models**, and open the application.js file.
2. On line 28, edit the name 'Open Toolchain' to be something else. This change will cause the Mocha unit test to fail.
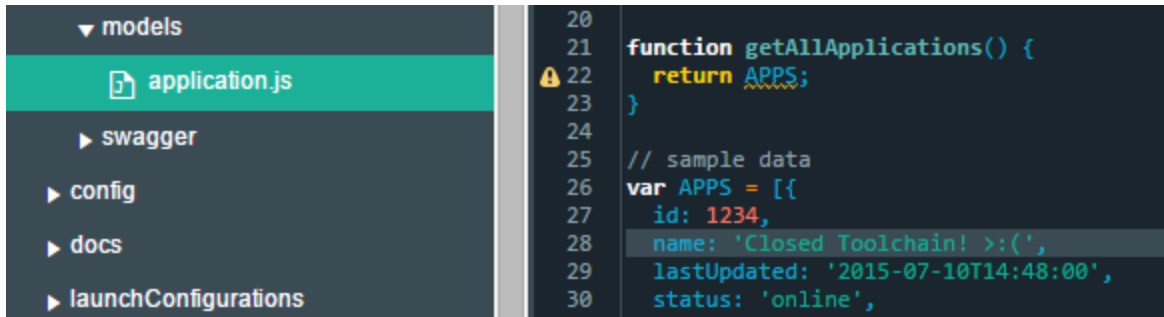
You don't need to save your change; the Web IDE automatically saves whatever you do.
Commit the change to your project's Git repo:

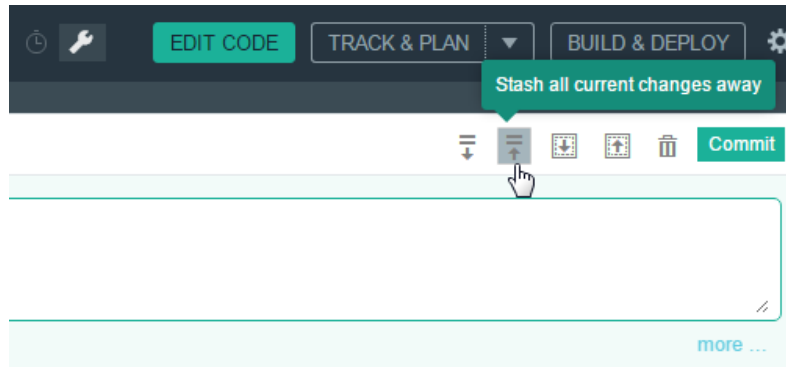1. From the menu on the left, click the **Git Repository** icon.
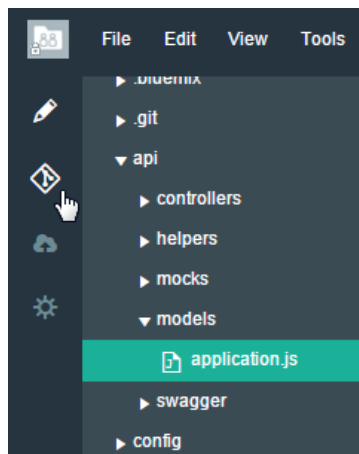
2. In the Working Directory Changes section, you'll see the application.js file. Select the check box next to it.
3. In the commit message field, type Making the unit tests fail.
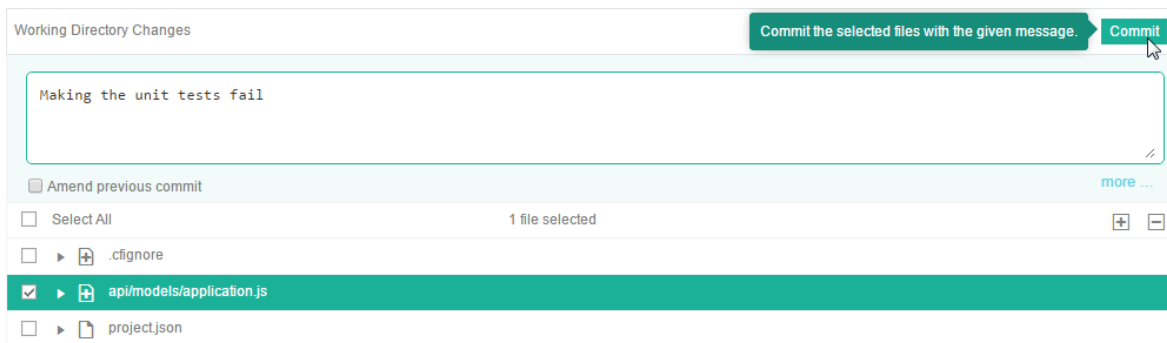
4. Click **Commit**.
5. If other files are in the Working Directory Changes, click the **Stash all current changes away** icon.

6. In the Active Branch section, in the Outgoing section, you'll see application.js. Click **Push** to push the change to your project's remote repository.

You just made a change to your project. Next, you'll see how the toolchain reacts to bad input.

Step 14 - Verify that the tests failed

To see failing tests in action, click **BUILD & DEPLOY**.

Your unit tests eventually fail because of the change you made. This failure causes the stage to fail, which, by default, means that the rest of the pipeline won't run. This example shows flow control in the pipeline: because the Run Unit Tests stage failed, there is no need to attempt a test deployment.
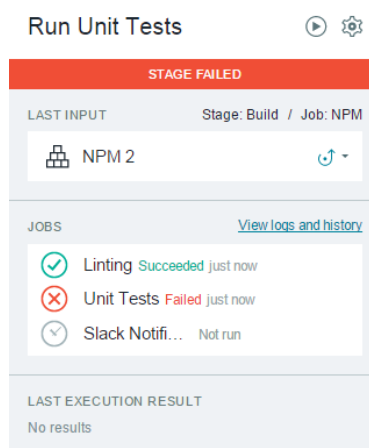
Step 15 - Revert the change

In the final step, you'll revert the change that caused the pipeline to fail.

1. Click **EDIT CODE**.
2. From the menu on the left, click the **Git Repository** icon.
3. In the Active Branch section, in the History section, select your most recent commit.
4. In the main window, click the **Revert** icon.
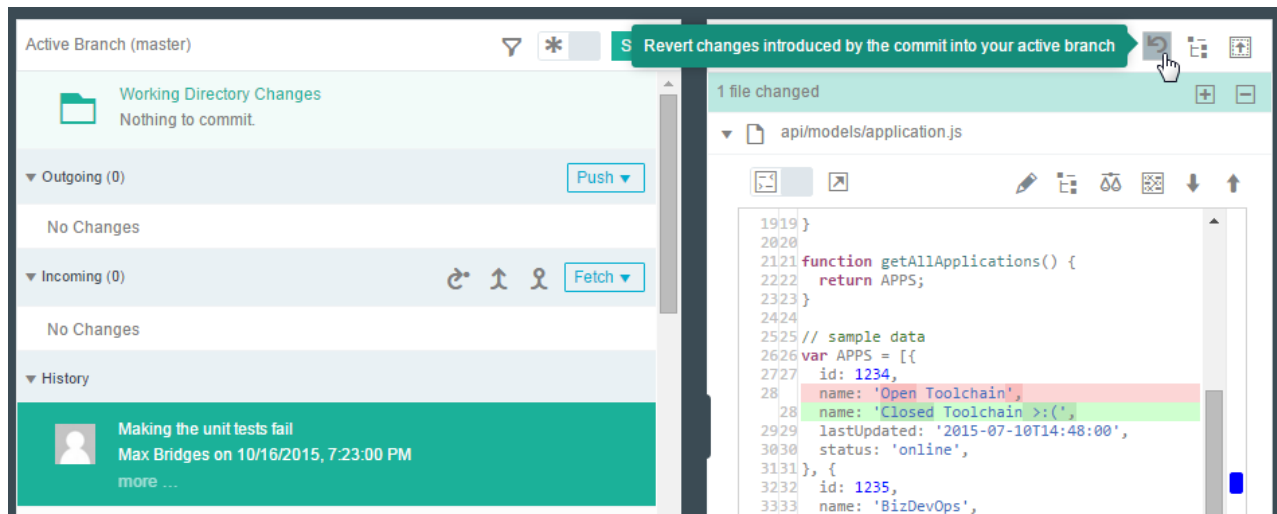
5. In the Active Branch section, in the Outgoing section, click **Push**.

Congratulations! You fixed your project! If you click **BUILD & DEPLOY**, you'll see that your Run Unit Tests stage is completely successfully passing now.

You just created your first build and delivery pipeline using IBM Blumix Devops Services, and configured integrations with many of the popular best of breed tools.

To learn more, visit our other tutorials at https://hub.jazz.net/tutorials/.