Lab 4: DevOps on IBM Cloud

Development Operations (DevOps) is a software methodology that integrates application development and information technology (IT) operations.

Application development includes writing code, testing the code, building fixes, integrating the fixes, building the application, and deploying the application.

IT operations include managing the environment on which the applications run, providing computer power to the applications, and making the software secure, scale, and run more efficiently.

The issues arise because development and operations were traditionally separate groups, “living” in their own isolated world. DevOps blurs the lines between the development tasks and operational tasks by integrating processes and tools.

DevOps provides real value to the business. For example, it enables continuous delivery, so as soon as new application features are complete, they can be automatically rolled into production. In turn, this action reduces time-to-market, provides competitive advantages, and reduces cost.

DevOps automates the deployment of fixes after they are tested and approved. DevOps enables developers to customize and change applications quickly, improving customer satisfaction.

DevOps enables a more stable environment and better application quality. The combination of a shared code base, continuous integration, test-driven techniques, and automated deployments expose problems in application code, infrastructure, or configuration earlier.

In this lab, you explore DevOps services in IBM Cloud. The IBM Cloud catalog provides multiple tools for DevOps, but this exercise is focused on IBM Cloud Continuous Delivery. Continuous Delivery enables you to build, test, and deliver applications by using DevOps practices and industry-leading tools.

Learning Objectives

After completing this exercise, you should be able to perform the following tasks:

- Enable your application to use IBM Cloud Continuous Delivery.

- Create a Git repository to manage your source code.
- View and edit code in the Eclipse Orion Web integrated development environment (IDE).
- Build and deploy code to IBM Cloud.
- Test the application in IBM Cloud.

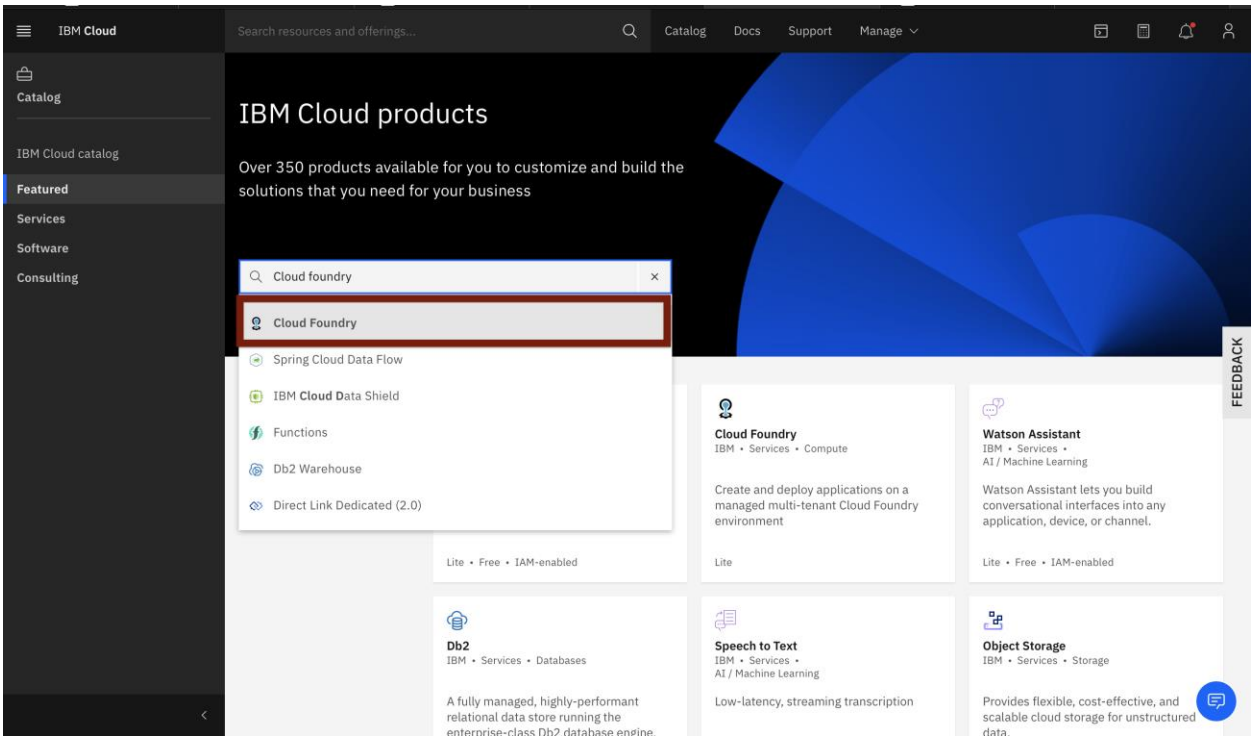
Prerequisites

You must have an IBM Cloud account.

Step 1: Re-creating your application

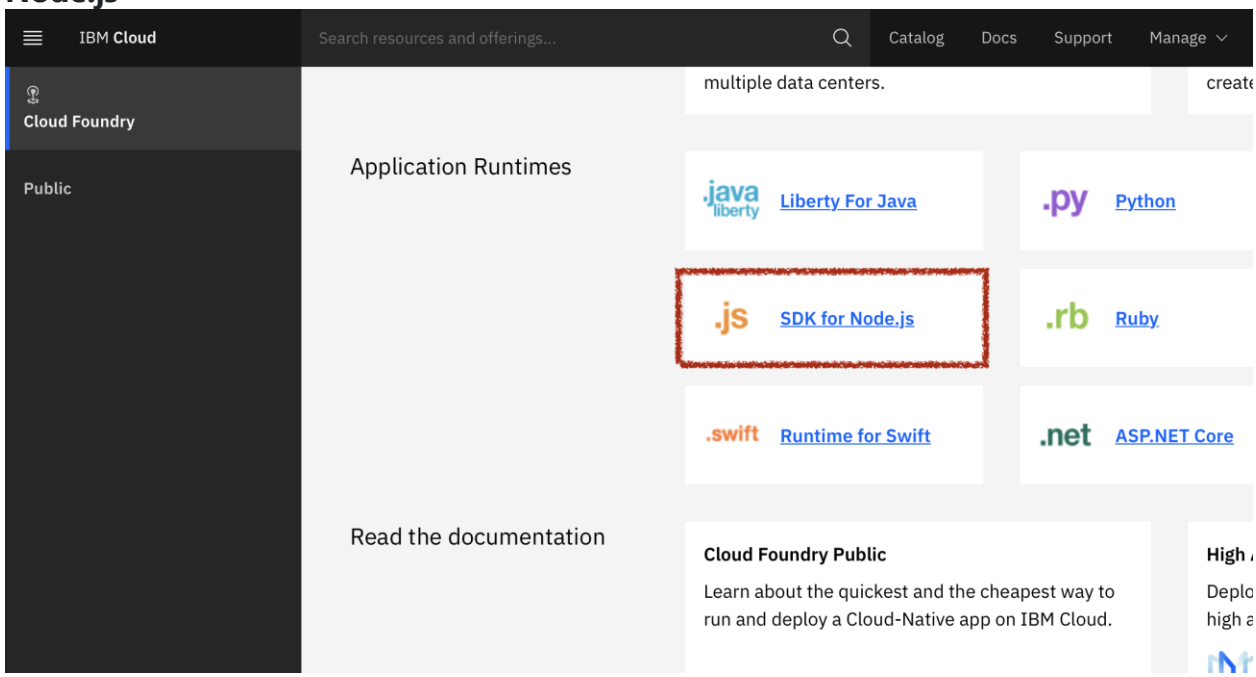
In this step, you re-create the sample Node.js application that you deleted at the end of previous lab, with Eclipse.

1. Open the IBM Cloud catalog in your web browser: https://cloud.ibm.com/catalog
2. If you are not already logged in, it takes you to the login page. Log in to your IBM Cloud account with your IBM Id and password.
3. It takes you to the catalog page. In the search field look for **Cloud Foundry** and select the option as seen in the below image.



Search Cloud Foundry

4. In the Cloud Foundry page, scroll down to **Application Runtimes** and choose **SDK for Node.js**



SDK for Node JS

5. Choose your region, choose your plan and enter a **unique name** for the application. You can use the same application name you used in the previous lab, which is `CD0201-xxx-nodesample`, replacing xxx with the first 3 characters of the uuid you generated in with `uuidgen`. The hostname is autogenerated based on the application name. In the image below, the application name is **CD201-38a-nodesample** (*38a* being the first 3 characters from the uuidgen). Accept the defaults for the other fields.

> The organization is set by default to the email you logged in with as shown in the next image. The space is set by default to dev as shown in the following image.

IBM Cloud | Search resources and offerings... | Catalog | Docs | Support | Manage

Catalog / Services /

.js Create a Cloud Foundry Sample App

Docs

Create | About

Select a location

Select a location

London (eu-gb)

Select a pricing plan

Displayed prices do not include tax. Monthly prices shown are for country or location: United States

Plan	Features	Pricing
Lite 64 MB 128 MB 256 MB	**Lite apps are free** You get up to 256 MB of memory while you work on your apps. **Lite apps sleep after 10 days of development inactivity.**	Free
Standard 256 MB+		$0.07 USD/GB-Hour

Configure your resource

Select a runtime

- .java Liberty for Java™
- .js SDK for Node.js™
- .net ASP.NET Core
- .go Go Community
- .php PHP Community
- .py Python Community
- .rb Ruby Community
- .swift Runtime for Swift
- tomcat Tomcat Community

App name

CD0201-38a-nodesample

Host name

CD0201-38a-nodesample

Domain

eu-gb.cf.appdomain.cloud

Choose an organization

ankisr4@gmail.com

Choose a space

dev

Tags

Examples: env:dev, version-1

Create Node JS Cloud Foundry application

6. Click **Create**. IBM Cloud proceeds to deploy your application. Your application stages and deploys in a few minutes.

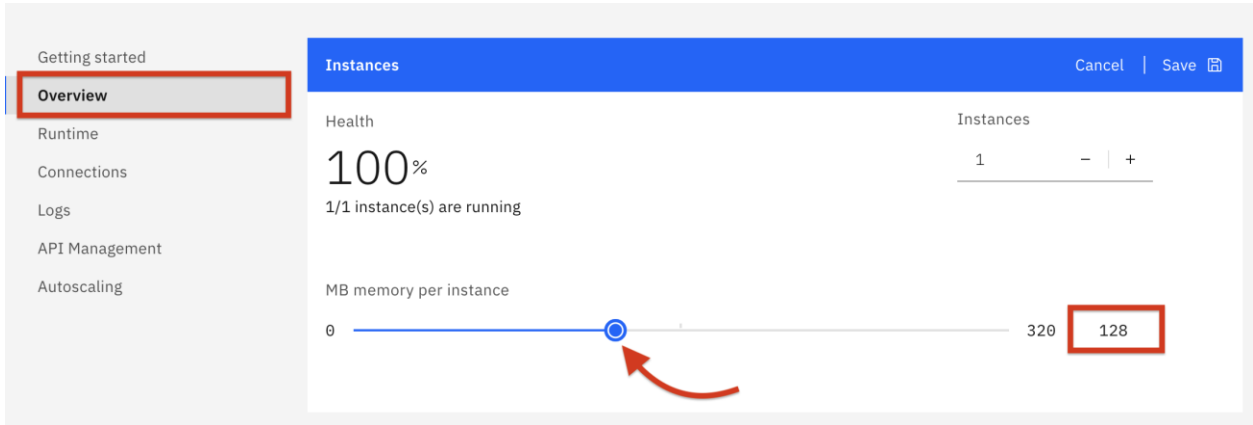
> **NOTE**: *Wait until the application finishes staging and is running in IBM Cloud before you proceed to the next step. Look for the indication that your app is running.*

Step 2: Examining the IBM Cloud application

In this step, you explore the application overview page in your IBM Cloud account. The overview page lists the status of your application and the resources that it uses. Set up a source code repository to manage the artifacts that make up your IBM Cloud application:

Open the application page for your sample application:

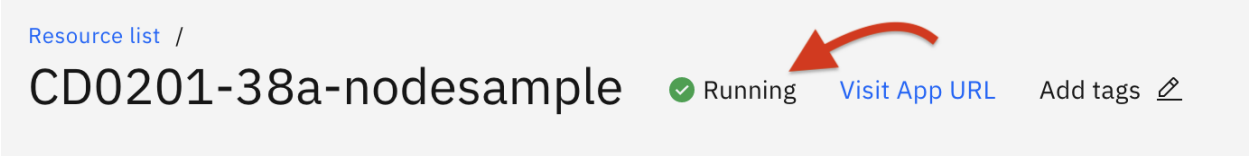
1. In the Application Details page, click **Overview** from the left navigation bar, as shown in the next image and set the memory quota to 128 MB, using the slider or by typing as shown in image below.



Application Overview

> **Note:** If the warning message *Changes May Be Overridden* is displayed, click **Close**.

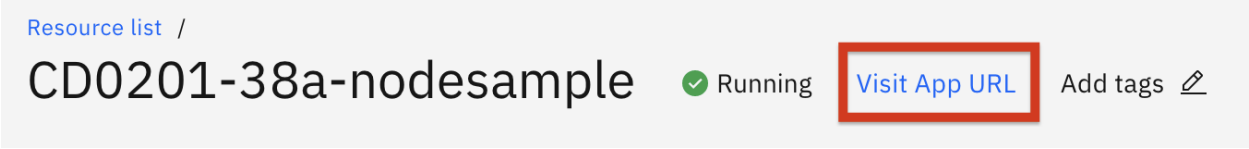
2. Your application restarts. Wait until your app is in the running state as shown in the image.



Application Running

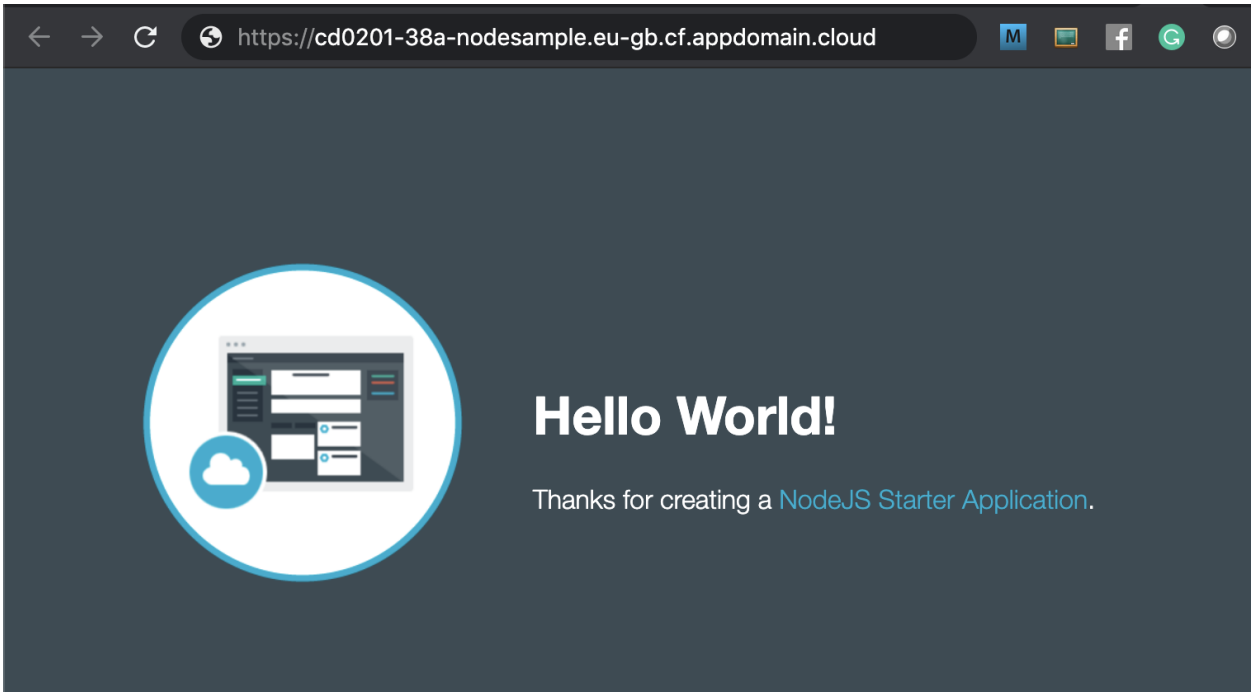
Test the sample application:

1. Click **Visit App URL** for your application as in the following image.



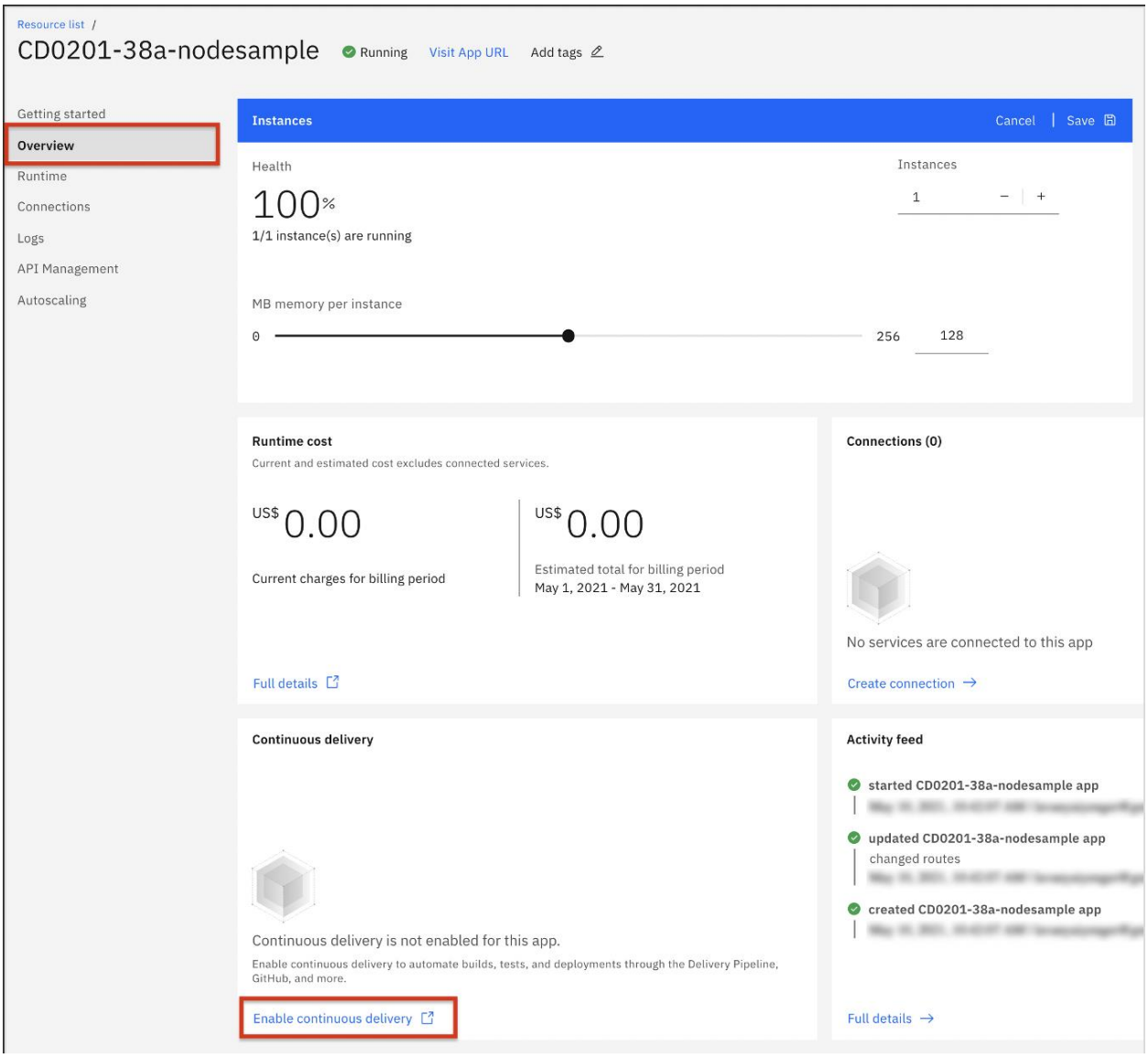
Visit App URL

2. A new tab opens in the browser that shows your app. Confirm that the sample application appears as shown in the following image.



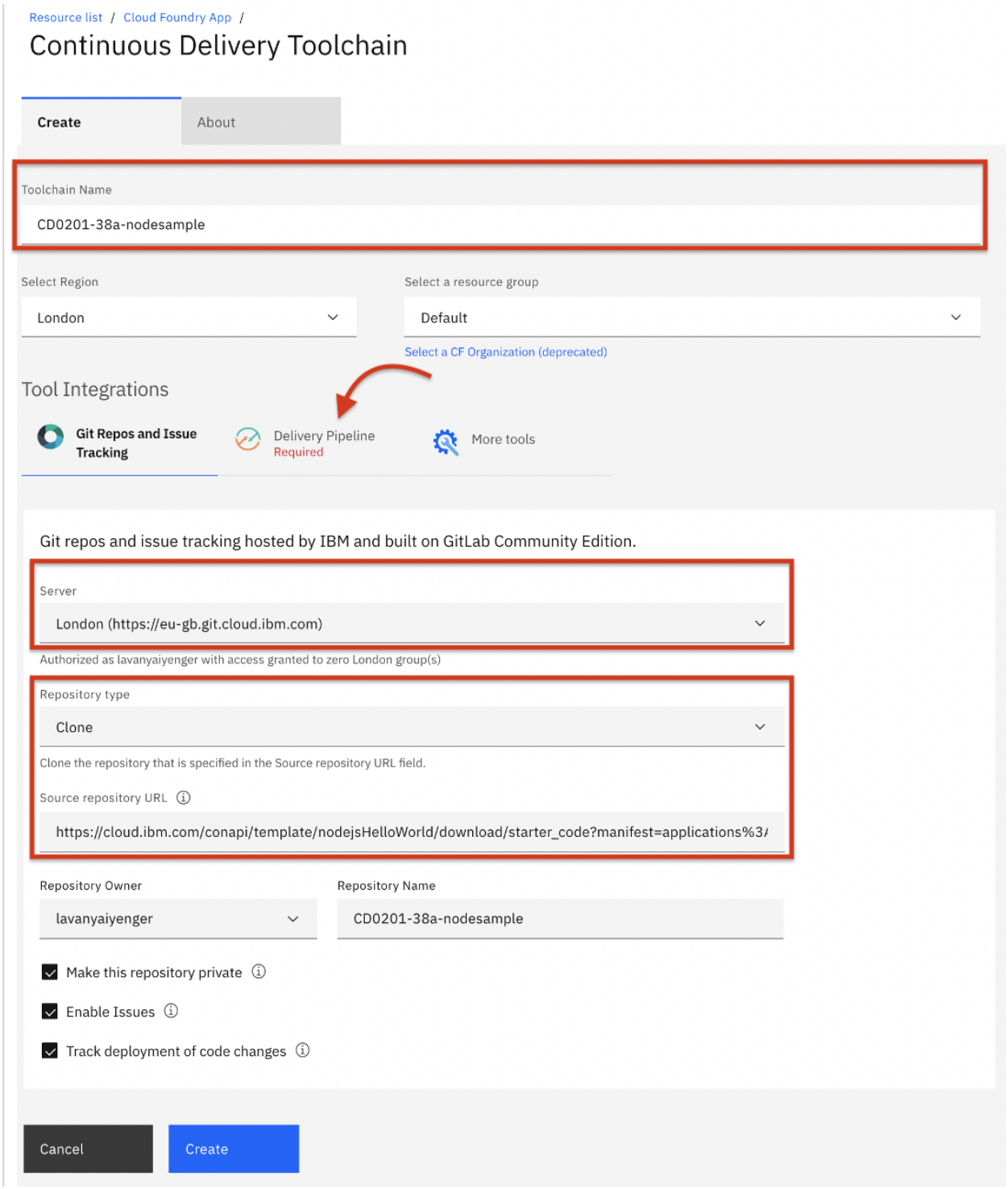
Hello World! window

3. Close the sample application web page.
4. In the overview page for the CD201-xxx-nodesample application, scroll down to Continuous delivery and click **Enable Continuous Delivery** as shown in the following image.



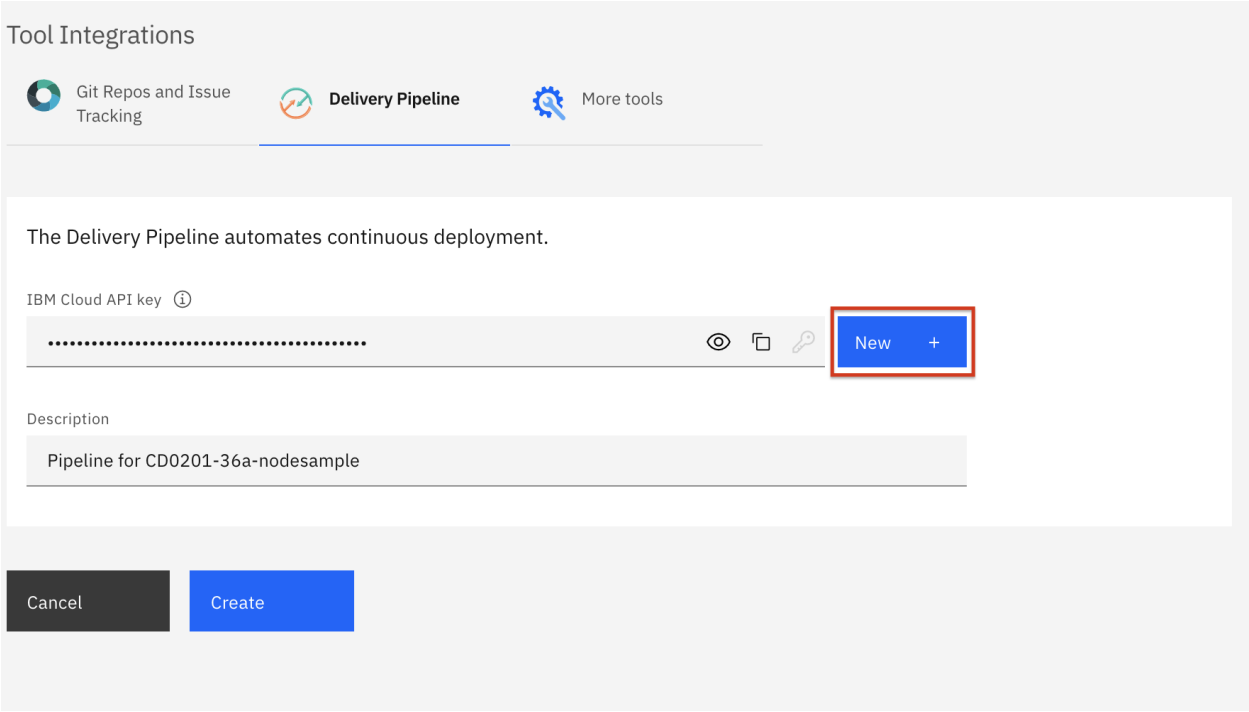
Continuous delivery

2. **Toolchain name** is auto-generated as the same name as the application. The region is the same as that of the application. The starter code repository is cloned by default. Click **Delivery Pipeline** in the Continuous Delivery Toolchain setup page.



Continuous Delivery Toolchain

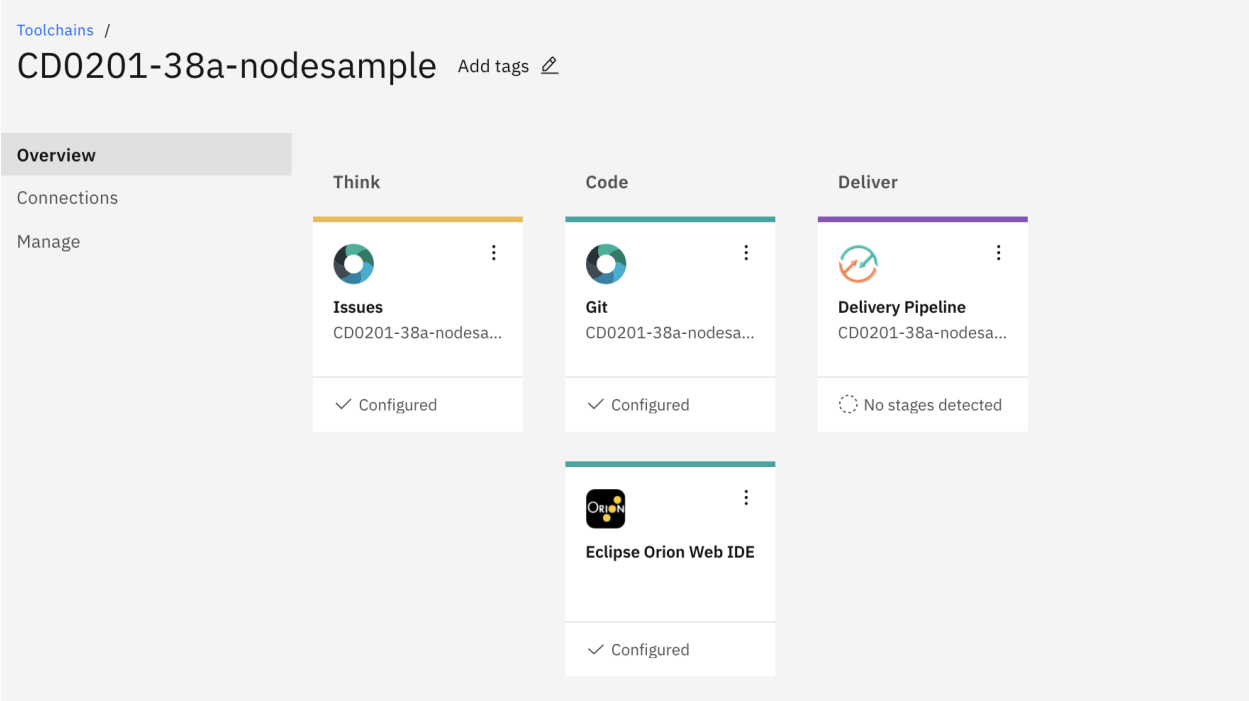
3. In the **Delivery Pipeline** tab, click on **New** to generate a new key and follow the instructions. Once the key is generated, click on **Create** to create the toolchain.



Continuous Delivery Toolchain

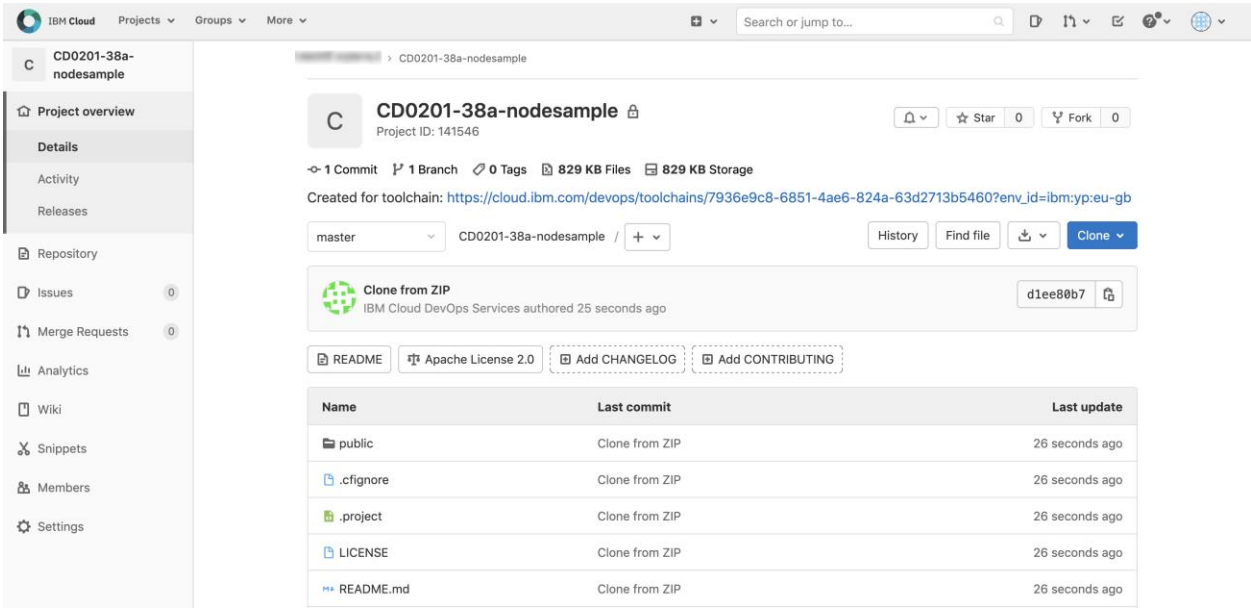
> **Note**: Wait until the wizard creates the Git repository for your application. By enabling Continuous Delivery Toolchains, you perform a Git clone for the IBM Cloud starter code by default.

4. You are redirected to the Toolchains dashboard as in the following image.



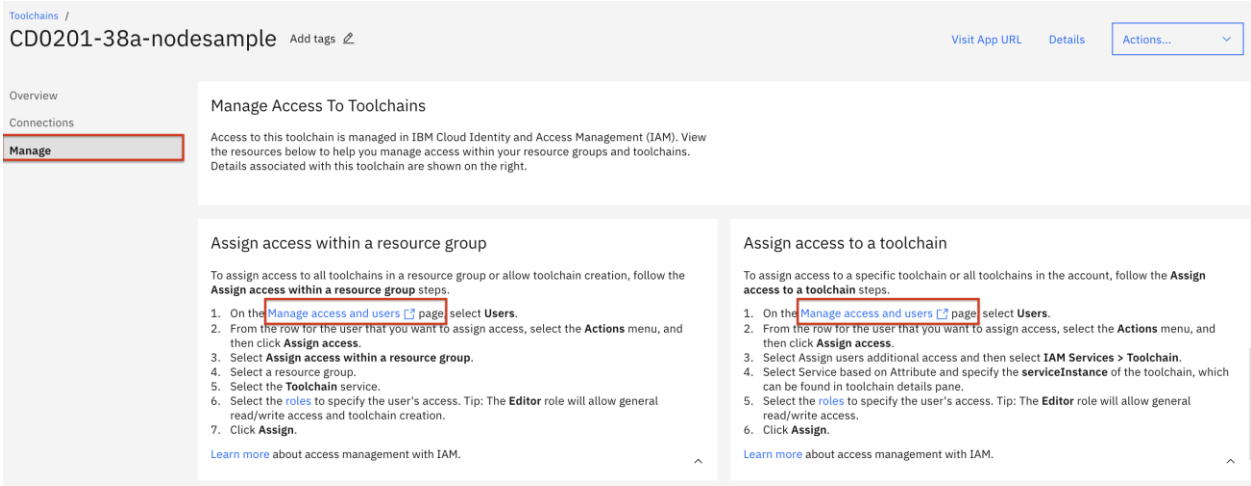
Toolchains dashboard

5. Confirm that the Git repository was created by right-clicking **Git** and then selecting **Open link in new tab**. The Git repository manager is displayed, as shown in the following image.



Web-based Git repository manager

6. Return to the **Toolchain** browser tab, click **Manage**. You can use this to grant users belonging to this organization access to the toolchain.



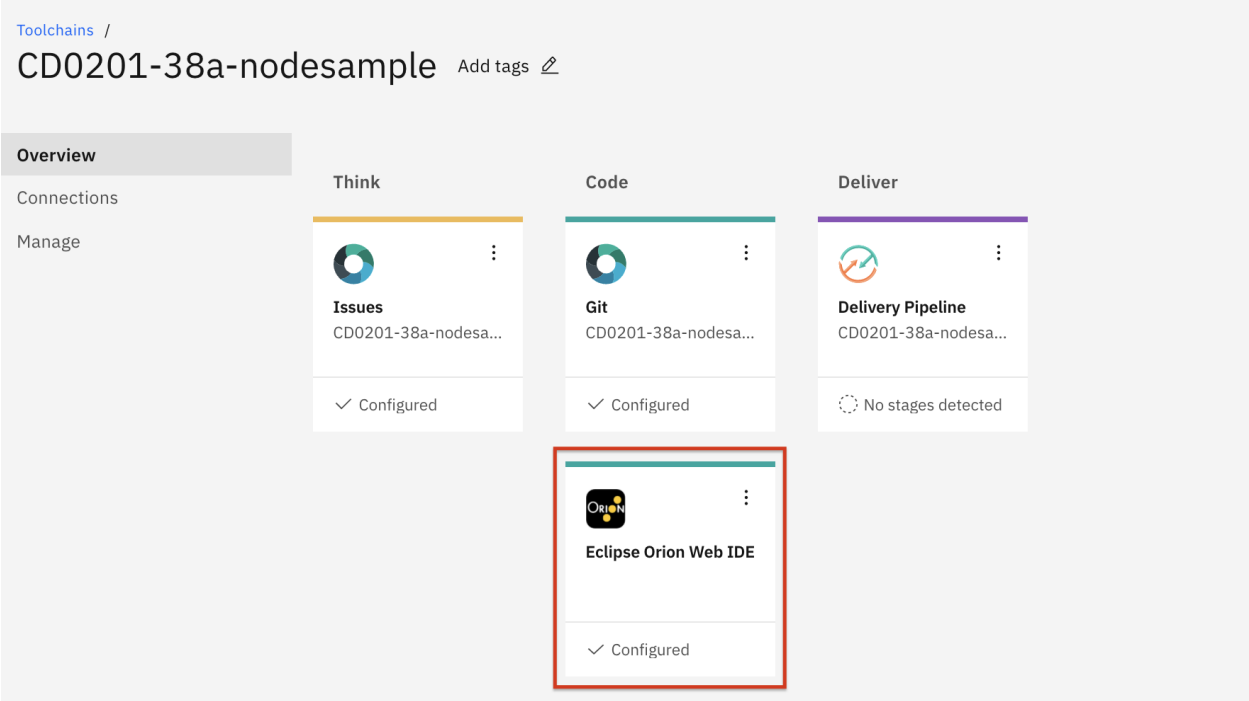
Managing access for the toolchain

> **Information**: IBM Cloud Continuous Delivery creates a Git repository as a change management system. You can use any Git client to work with the artifacts that are stored in the repository.

Step 3: Reviewing Eclipse Orion for your application

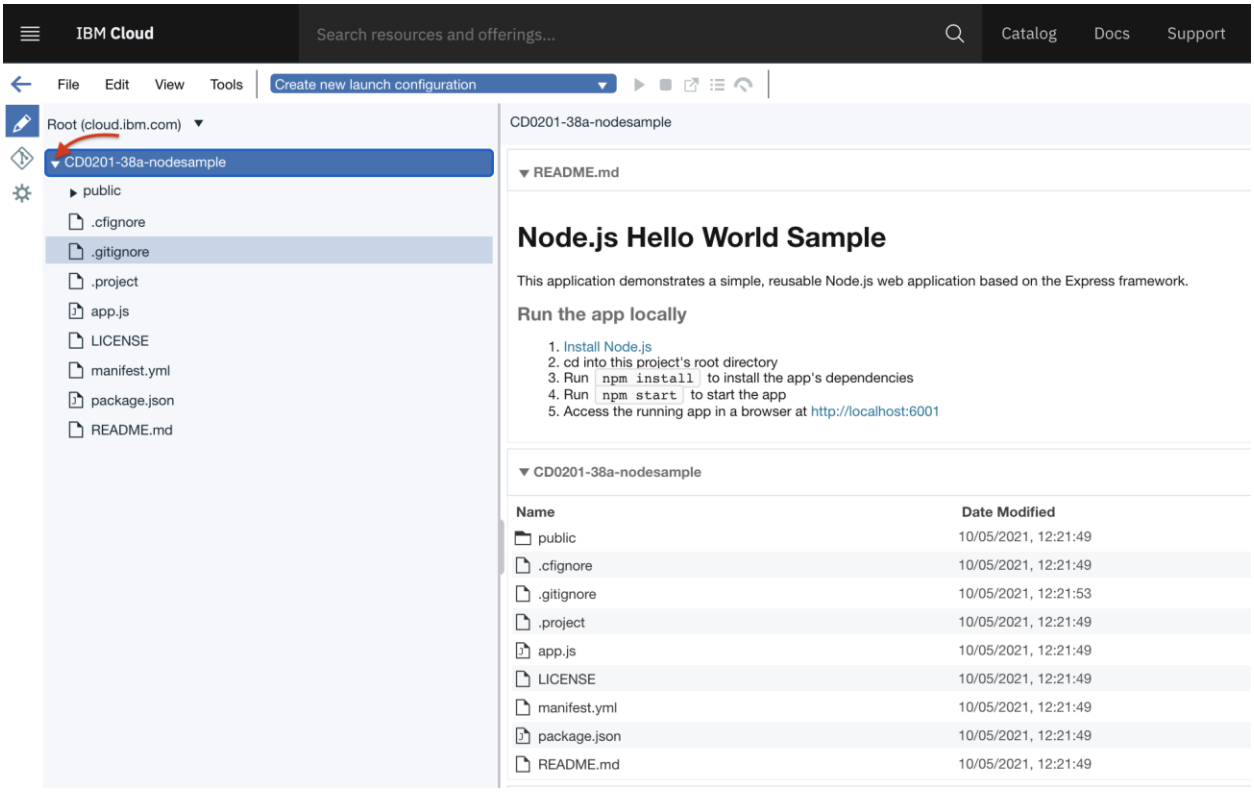
Examine the Web IDE features in IBM Cloud Continuous Delivery:

1. Right Click **Eclipse Orion Web IDE** and open in new tab. This will open the Web-IDE.



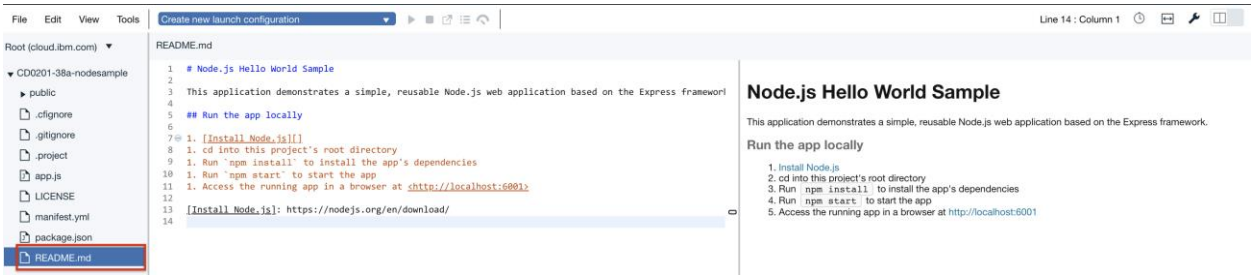
Project directory

2. Open the project directory in the Web-IDE and explore it.



Eclipse Web-IDE

3. Open the README.md document by clicking it as shown in the following image.

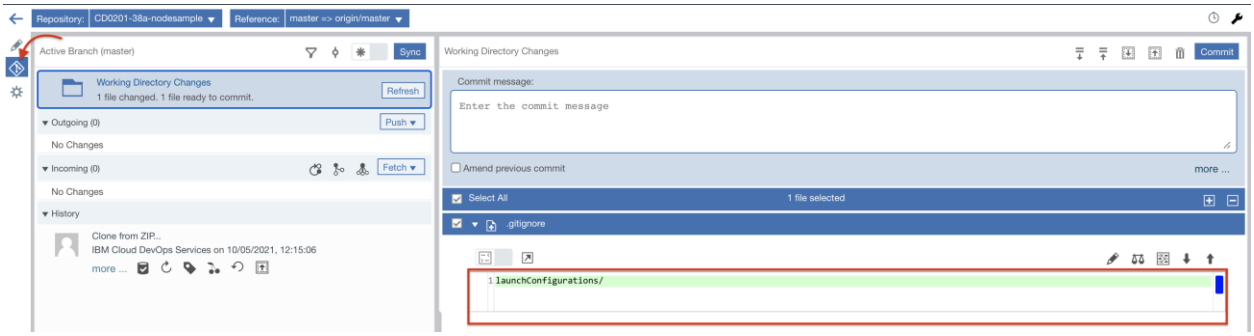


Opening the README.md document

The readme file provides a quick summary of the repository that manages your application artifacts and source code. It is a preferred practice to provide an up-to-date description of your project in this document, especially for projects that are publicly shared for all users.

> **Information**: The readme file is written with the Markdown syntax, a lightweight markup language for annotating plain text documents to be displayed as Hypertext Markup Language (HTML) documents. For more information about Markdown, see the following website: http://daringfireball.net/projects/markdown/

4. Click on the **Git** icon to see all the changes in the repository.

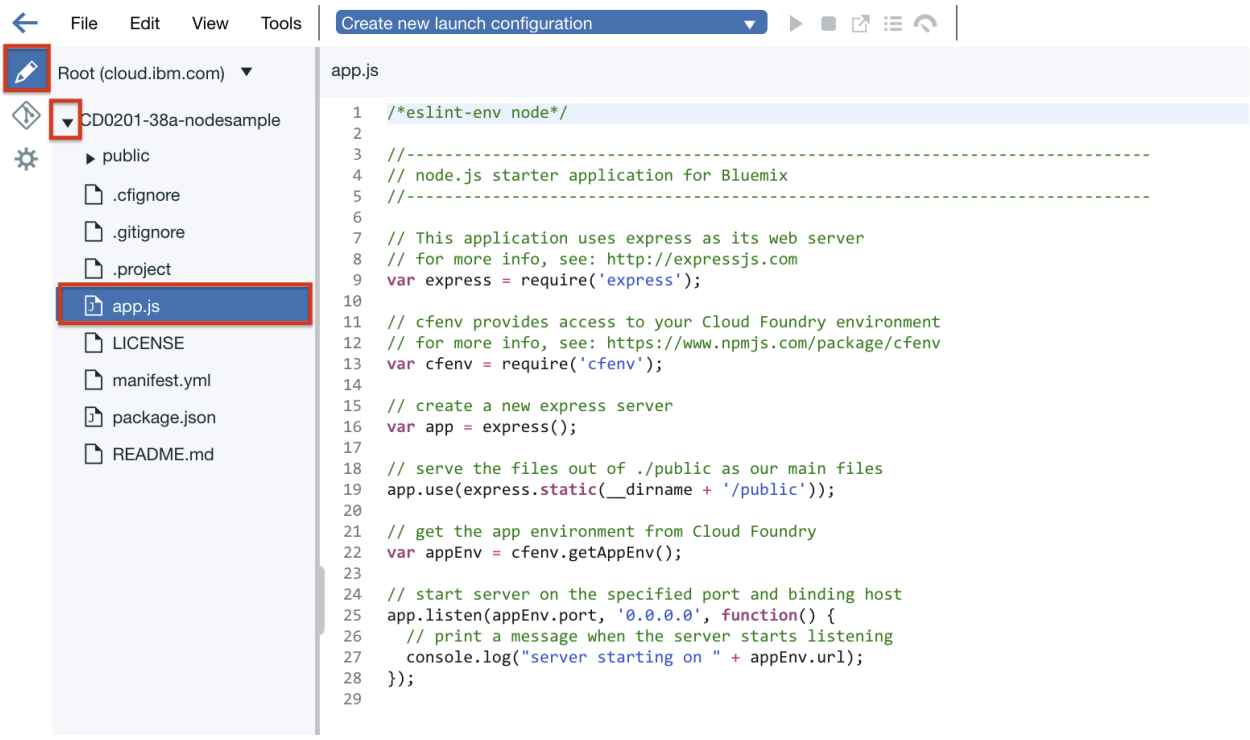


Commits menu

> Note *As you have not done any changes yet, you will just see the default updation to `.gitignore` as the only change.*

Step 4: Editing the sample application

1. Go back to **Edit** mode on Eclipse Orion Web IDE. Click app.js from the project directory and examine the contents of the app.js source code in the editor as shown in the following image.

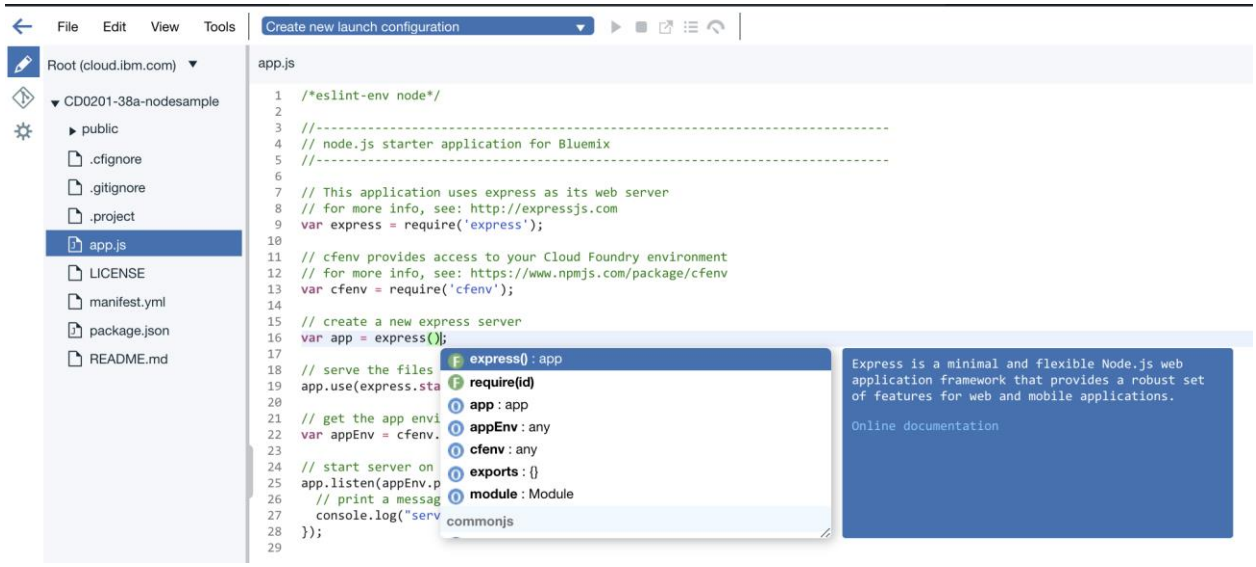


Clicking app.js

The editor in the web application provides the same features as the desktop Eclipse application:

- Syntax highlighting
- Static code analysis of the JavaScript code
- A preview of the document

2. Place your cursor on line 16, next to the term express() in var app=express().
3. Press **Ctrl + Spacebar** on your keyboard. You should see the window shown in the following image.

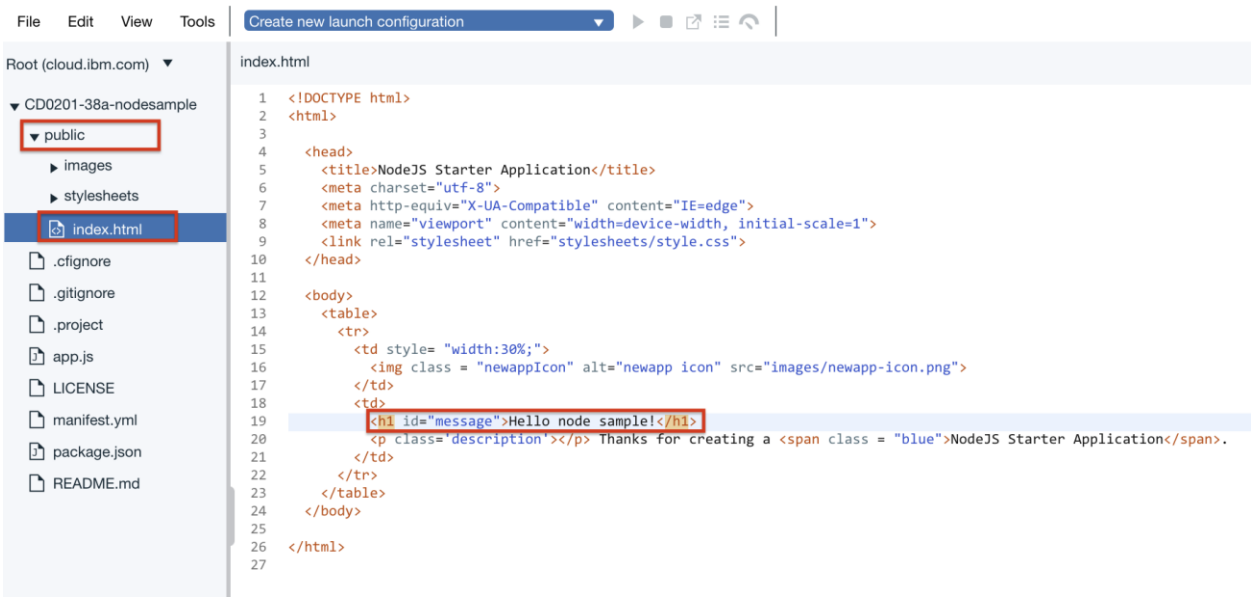


Code completing feature

The code completion feature in the JavaScript editor supports Node JavaScript modules and the standard JavaScript functions.

4. In the **public** folder, go to **index.html** and click on it to open the file. Select the sentence in line 19. Change the phrase inside the \

tags to **Hello node sample!**, as shown in the next image.



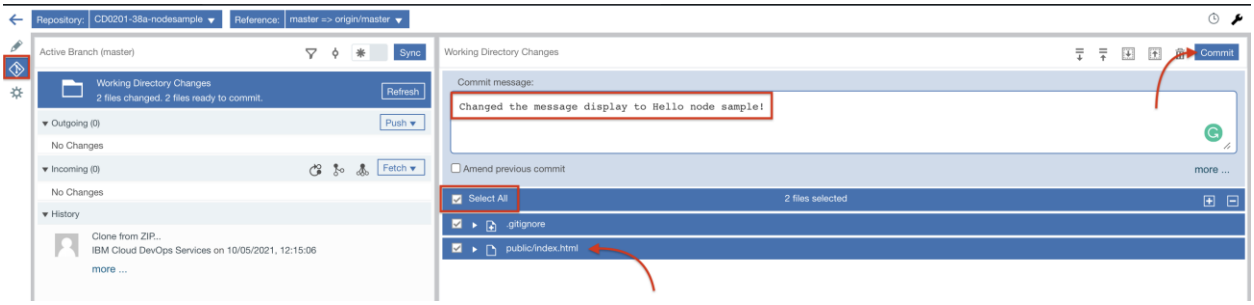
The index.html file

5. All the code changes are automatically saved. To force a save, press **Ctrl + S** to save your changes.

Step 5: Committing your changes to the Git repository

In this step, you see the effects of the source code changes by committing your changes to the Git repository, then pushing the changes to your IBM Cloud application.

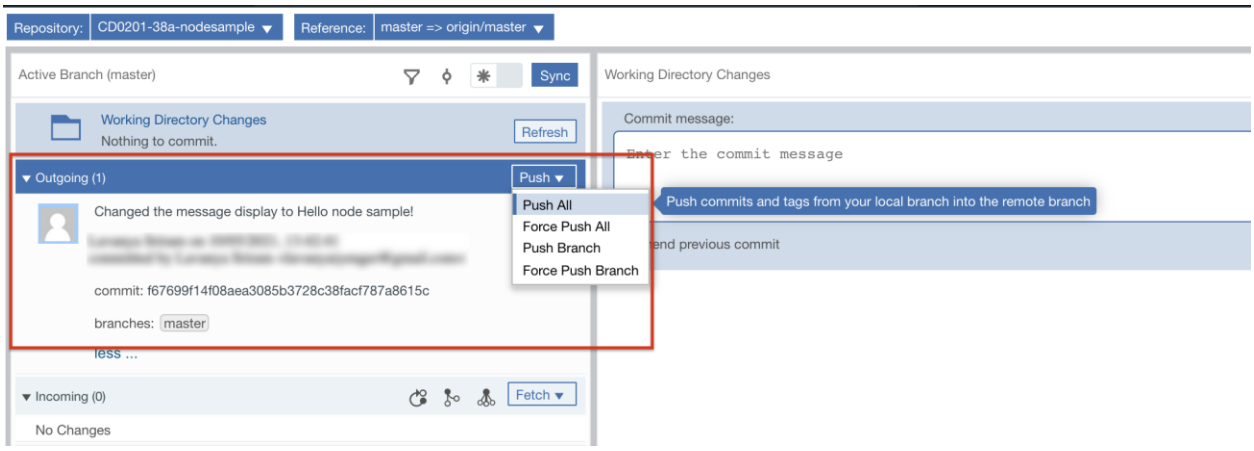
1. Click the **Git** icon on the left navigation bar, to go the git respository.
2. It shows that the **index.html** file has changed. Add commit message. Make sure that the **Select All** check box is selected. Click **Commit** to save the files into the Git repository.



Committing the changes

> **Information**: If you are prompted to provide author and committer name as mandatory fields, enter your name and email.

3. Examine the Working Directory Change view. Confirm that there are no more updated files to commit from the working directory and that the changeset is in the Outgoing section as shown in the following image. Click **Push** and choose **Push All** to permenantly write these changes to the repository. Once you **push**, the application is



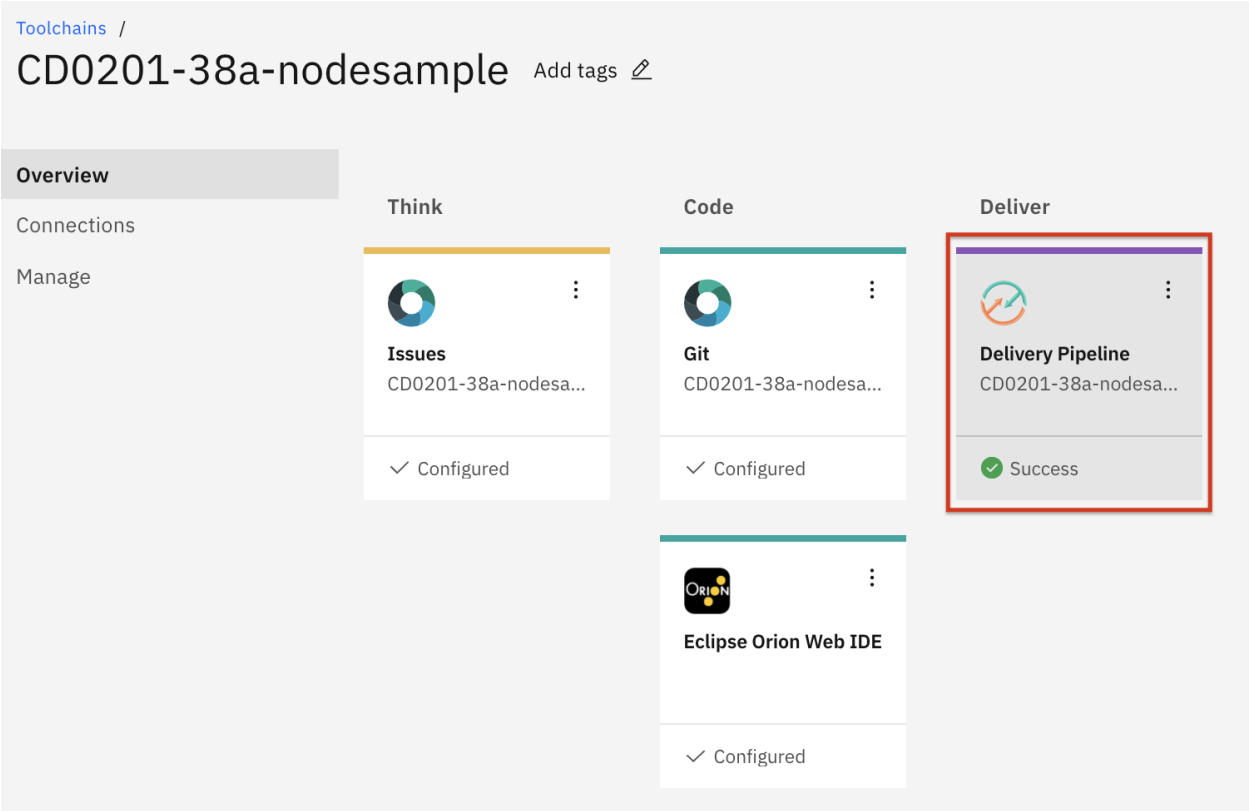
Pushing Changes: Outgoing

> **Note**: If more than one developer is editing the application, consider checking in your changes and pushing the most recent revision from the Git repository.

Step 6: Deploying the application from the Git repository to IBM Cloud

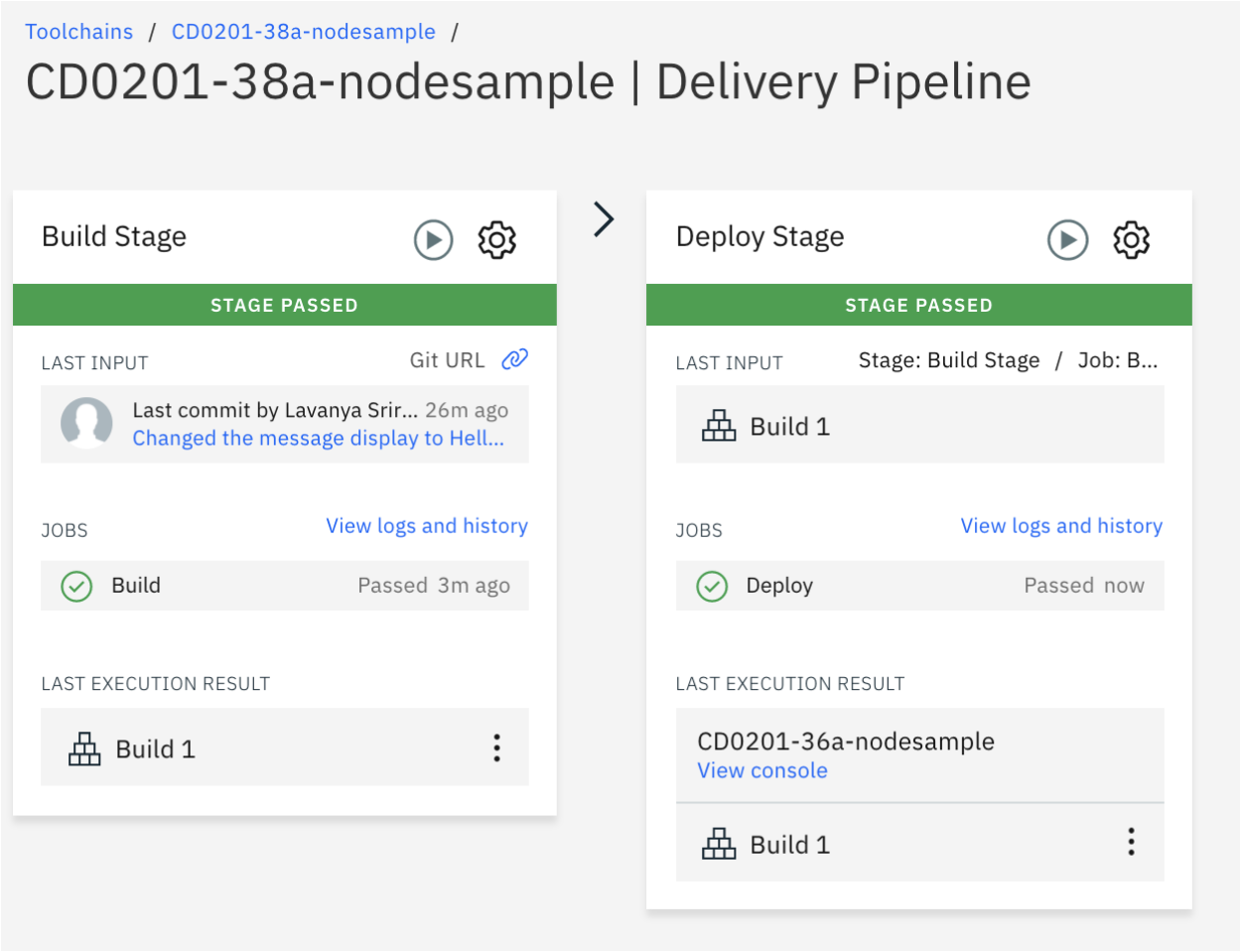
You can also deploy your application to IBM Cloud by directly from the Web IDE.

1. Go to the **Toolchain** dashboard and click on **Delivery Pipeline**. *If you don't have the Toolchain dashboard open, go to the application page and choose **View ToolChain** in the **Continuous Delivery** tab.*



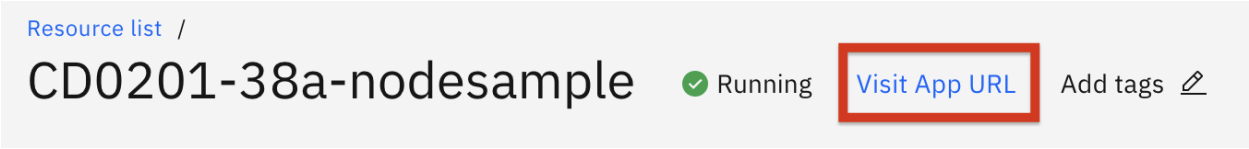
Delivery Pipeline

2. Examine the delivery pipeline. Wait until the build and deploy tasks complete as shown in the following image. You can also explicitly start it by clicking on the **play** button.



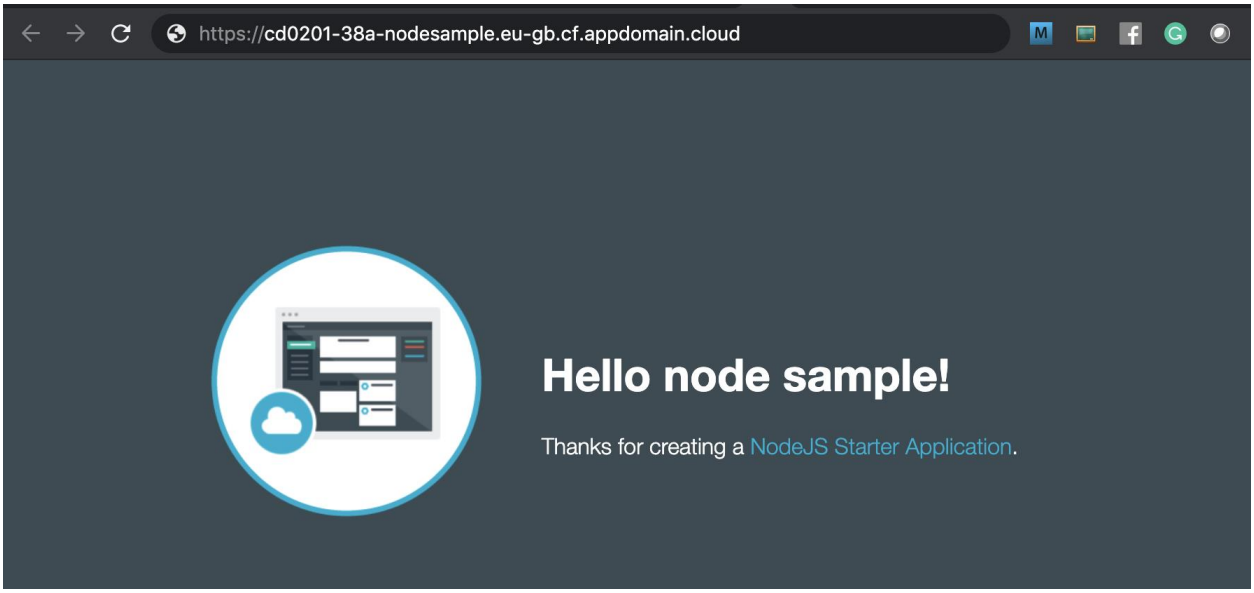
Examining the Delivery Pipeline

3. Click **Visit App URL** for your application as in the following image.



Visit App URL

4. Confirm that the application web page changed to “Hello node sample!” as shown in the following image.

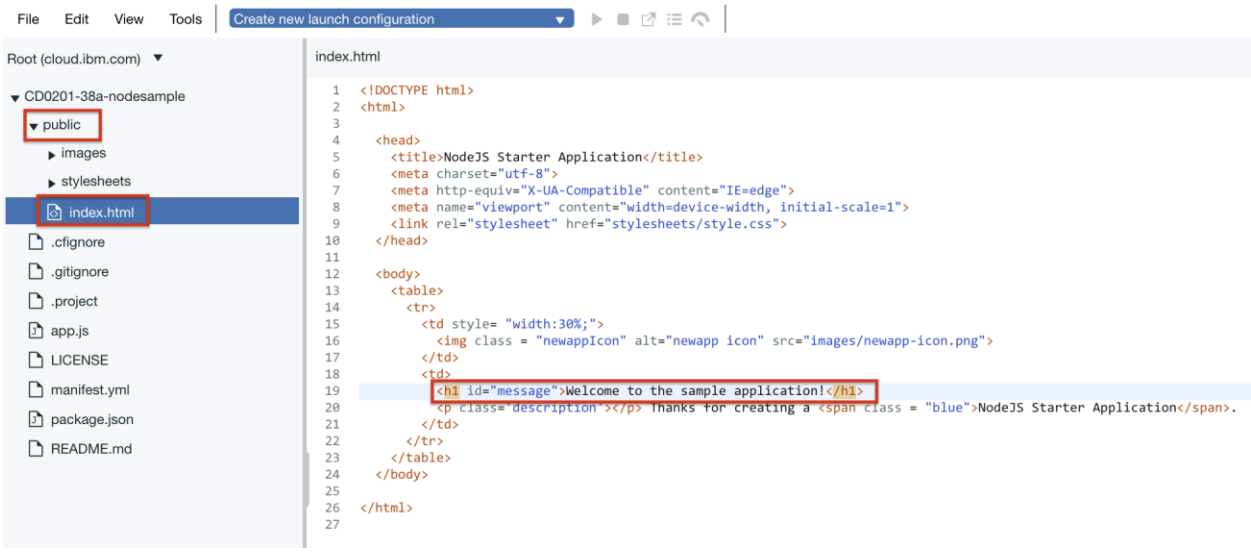


Hello node sample

Step 7: Deploying the application directly from the Web IDE

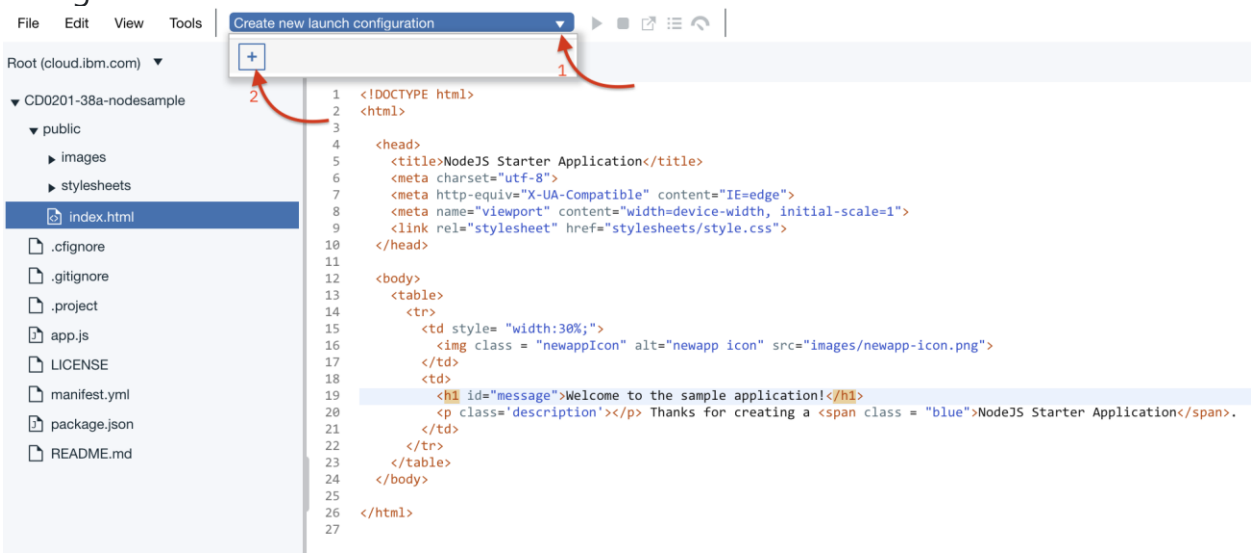
You can also publish changes directly from the Web IDE. With this technique, you can quickly test changes to your code on an actual IBM Cloud account:

1. In the Eclipse Orion Web IDE, open **index.html** in the **public** folder as shown in the following image. In line 19, change the heading to “Welcome to the sample application!”. To force a save, press **Ctrl + S**.



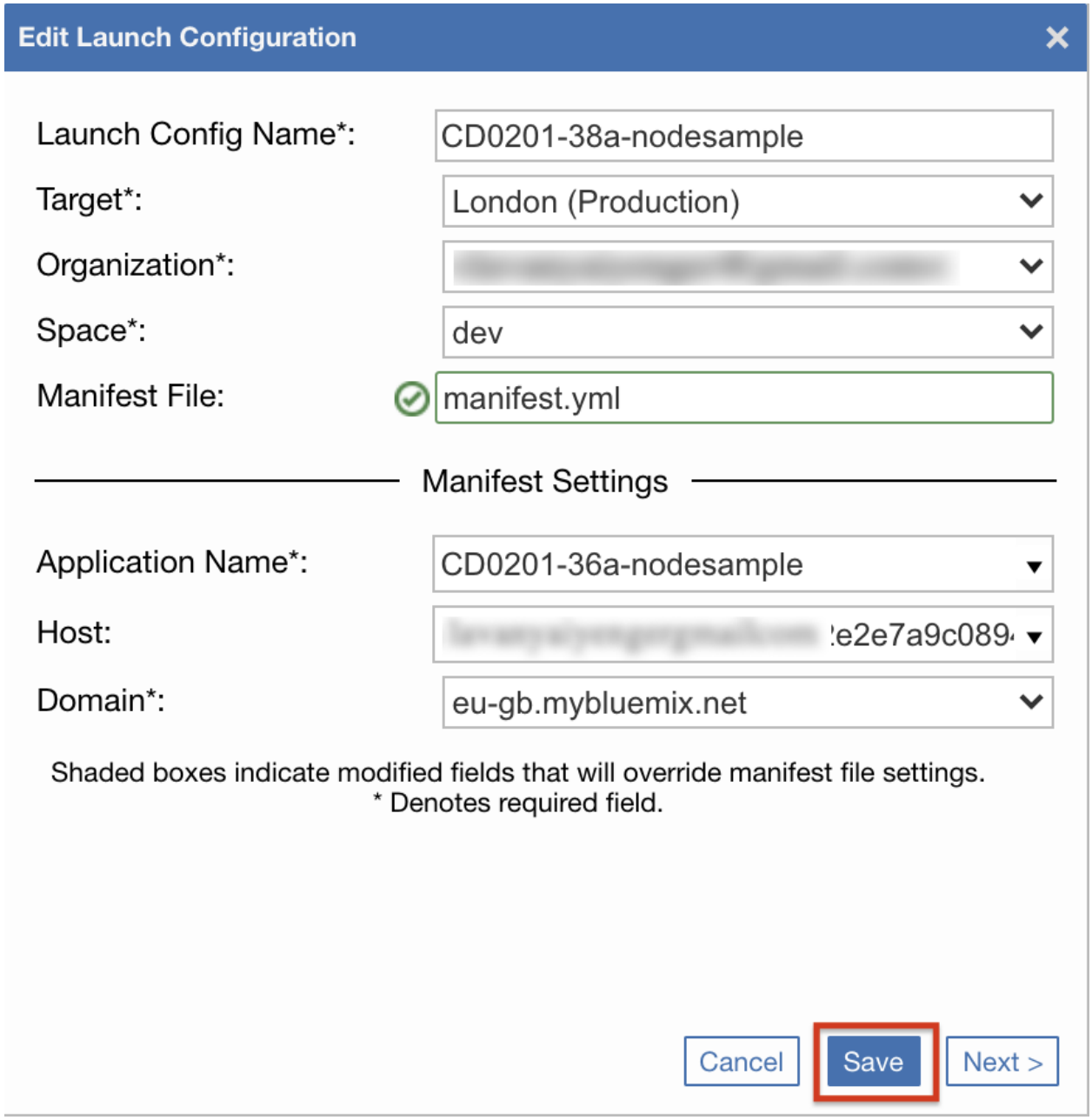
index.html

2. If the application already has a launch configuration, skip step 2 and step 3. From the top menu bar, select **Create new launch configuration** and click on the **+**, to add the configuration.



Add launch configuration

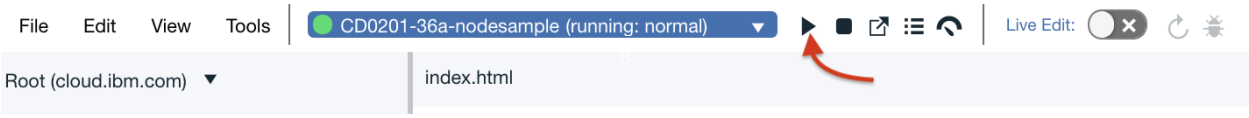
3. The launch configuration page comes up. Ensure the application name, region and other details are populated. Once you have verified all details, click on **Save**.



Add launch configuration

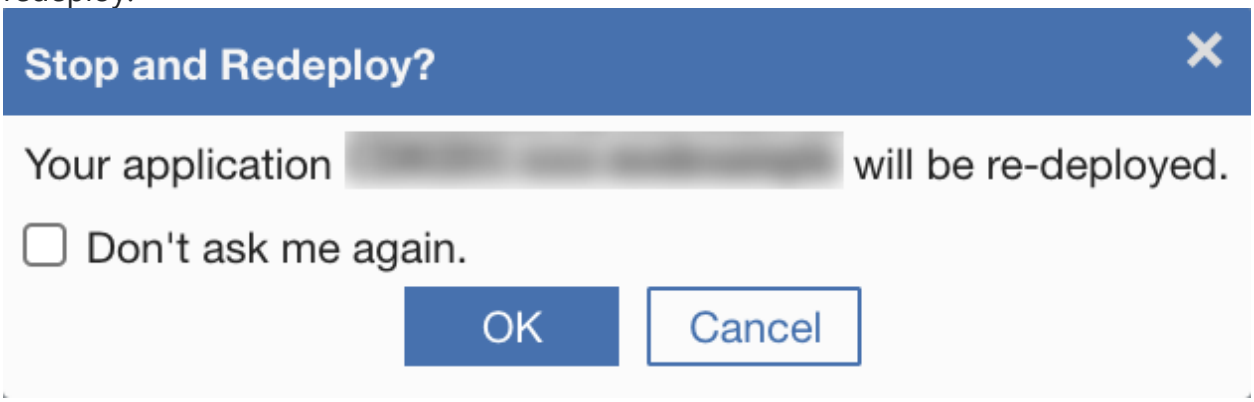
This initiates the deployment and starts the server. *Skip step 4 and 5 if this is the first time you are deploying. The next two steps pertain to deploying the application.*

4. From the menu on top, click on the **play** button to deploy the application.



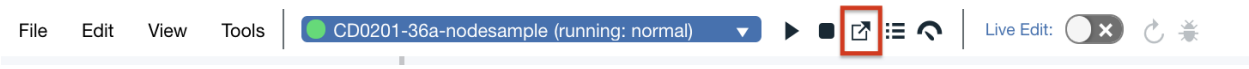
Deploy Application

5. A warning pop-up comes up asking if you want to stop the running application and redeploy. Click **ok** to redeploy.



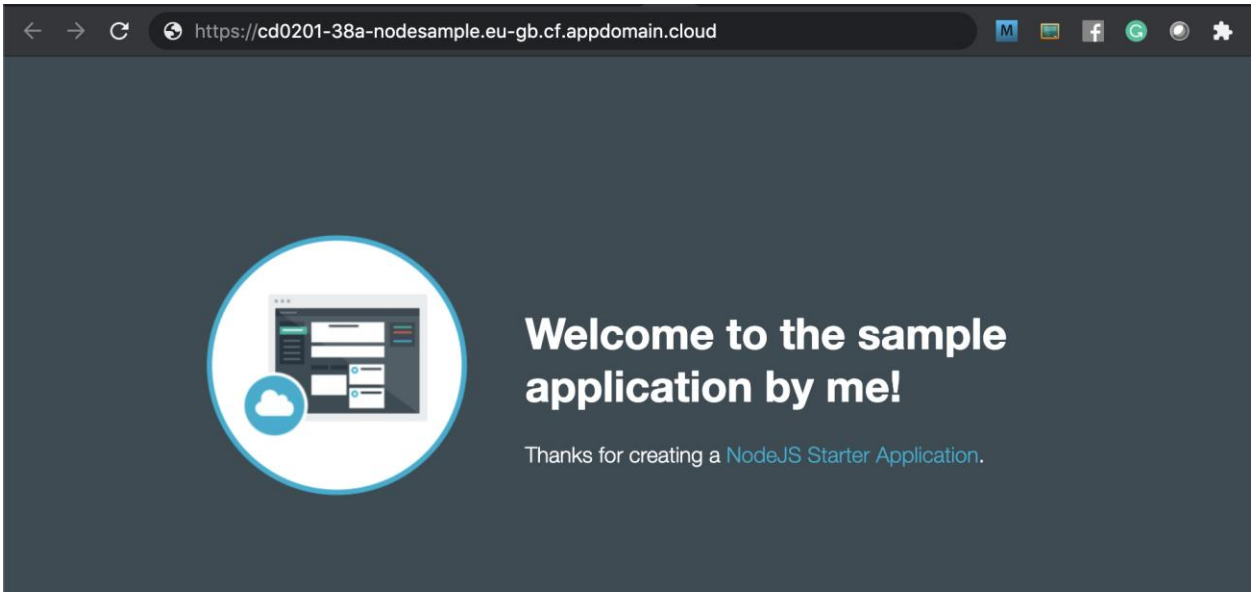
Redeploy Application

6. Confirm that the changes appear in the application by clicking on **Open the Deployed App** icon to view the changes to your application as shown in the next image.



Open the Deployed App

7. Verify that the updated heading appears in the sample application web page, as shown in the following image.



Verifying the updated heading

Step 8: Automatically push changes to IBM Cloud (optional)

Note: This part cannot be performed with an IBM Cloud Lite account. You need a Pay-As-You-Go account, or Subscription account.

With the stop and redeploy option, you must first manually deploy the changes and trigger an application restart through the server toolbar. Although this option is more convenient than having to commit your changes to the Git repository, it can be disruptive to your software development workflow for minor changes.

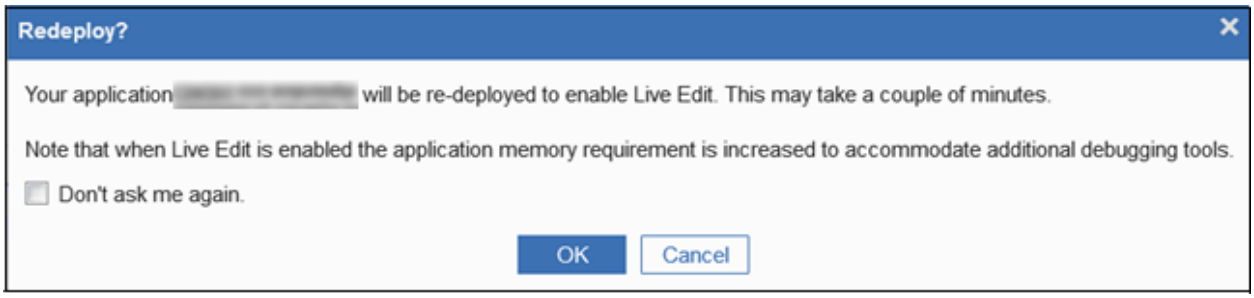
The IBM Cloud Live Sync feature automatically pushes any saved changes in the project workspace to a linked IBM Cloud application. For IBM SDK for Node.js applications, you can update static files and view the updates on IBM Cloud without restarting your application. Alternatively, for the non-static Node.js, such as JavaScript files, you need to restart only your application without having to deploy the changes.

To automatically push changes to IBM Cloud, complete the following steps.

1. Return to the Eclipse Orion Web IDE web page and enable the **Live Edit** feature

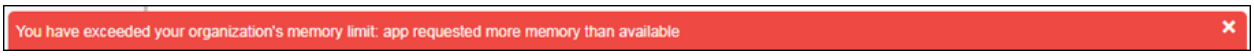
Live Edit

2. Click **OK** to redeploy the application and enable Live Edit mode, as shown in the next image. Enabling Live Edit mode allocates more memory to the application to enable debugging.



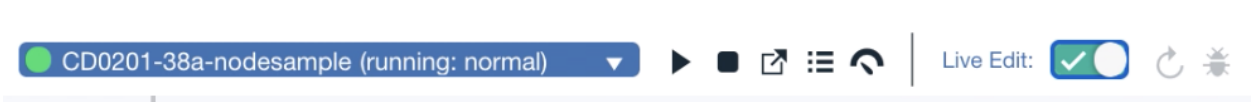
Redeploying application and enabling Live Edit mode

> **Information**: The quota for the IBM Cloud Lite account is 256 MB, and applications that have Live Edit enabled require 800+ MB on average; this is the reason why this part of the exercise cannot be performed with an IBM Cloud Lite account. The following image shows the error message that you receive if you are using and IBM Cloud Lite account to run this part of the exercise.



Error message

3. Confirm that Live Edit mode is enabled, as shown in the next image.



Live Edit mode enabled

> **Note**: If you have a synchronization error, restart the application.

4. Wait until the application is running in Live Edit mode, as shown in the next image. It might take a few minutes.

Waiting for the application to run in Live Edit mode

5. On the index.html web page, edit the heading to read **Welcome to the live edit sample application!**, as shown in the following image.

```
<tr>
  <td style= "width:30%;">
    <img class = "newappIcon" src="images/newapp-icon.png">
  </td>
  <td>
    <h1 id="message">Welcome to the live edit sample application!</h1>
    <p class='description'></p> Thanks for creating a <span class = "blue">NodeJS Starter Application</span>.
  </td>
```

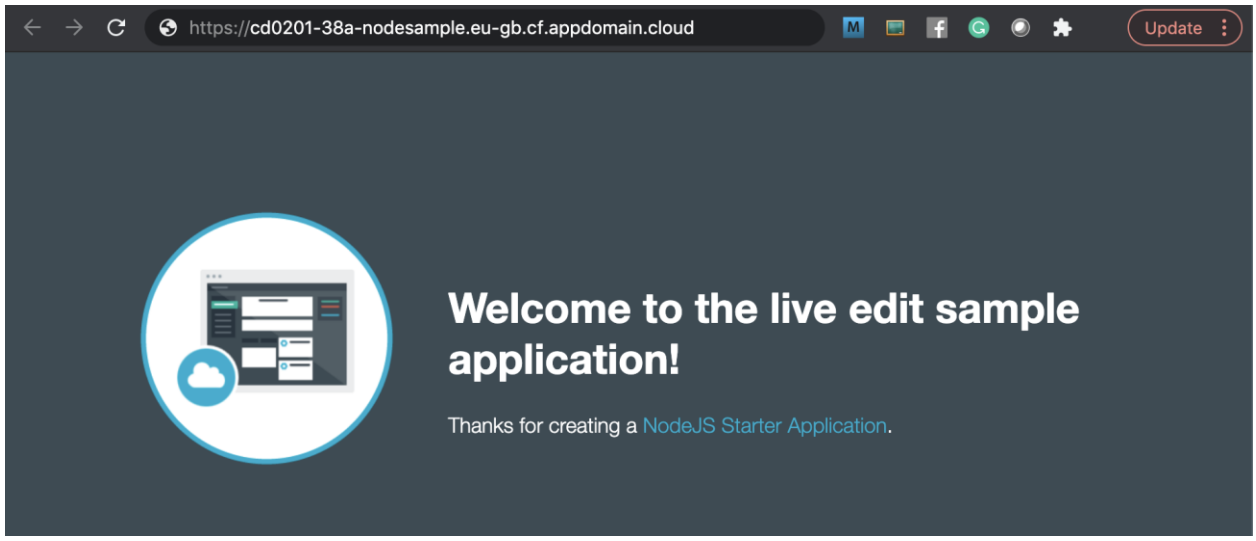
Editing the heading

> **Note**: With the Live Edit feature, you do not need to save or deploy your static files. Instead, the changes are automatically deployed to IBM Cloud. Examples of static files include HTML web pages and cascading stylesheets (CSS). You might still need to commit your files in Git if you want other people to see them.

6. Confirm that the changes appear in the application by clicking on **Open the Deployed App** icon to view the changes to your application as shown in the next image.

Open the Deployed App icon

7. Verify that the updated heading appears in the sample application web page, as shown in the next image.

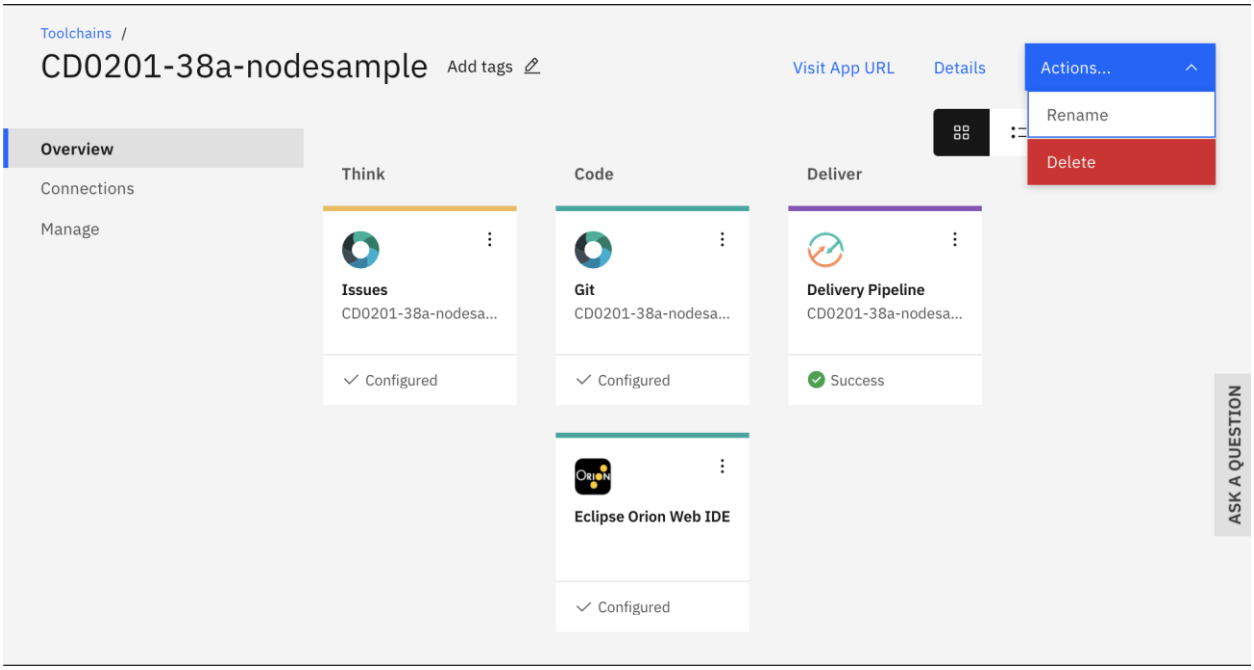


Verifying the updated heading

Step 9: Cleaning up the application from IBM Cloud and Continuous Delivery

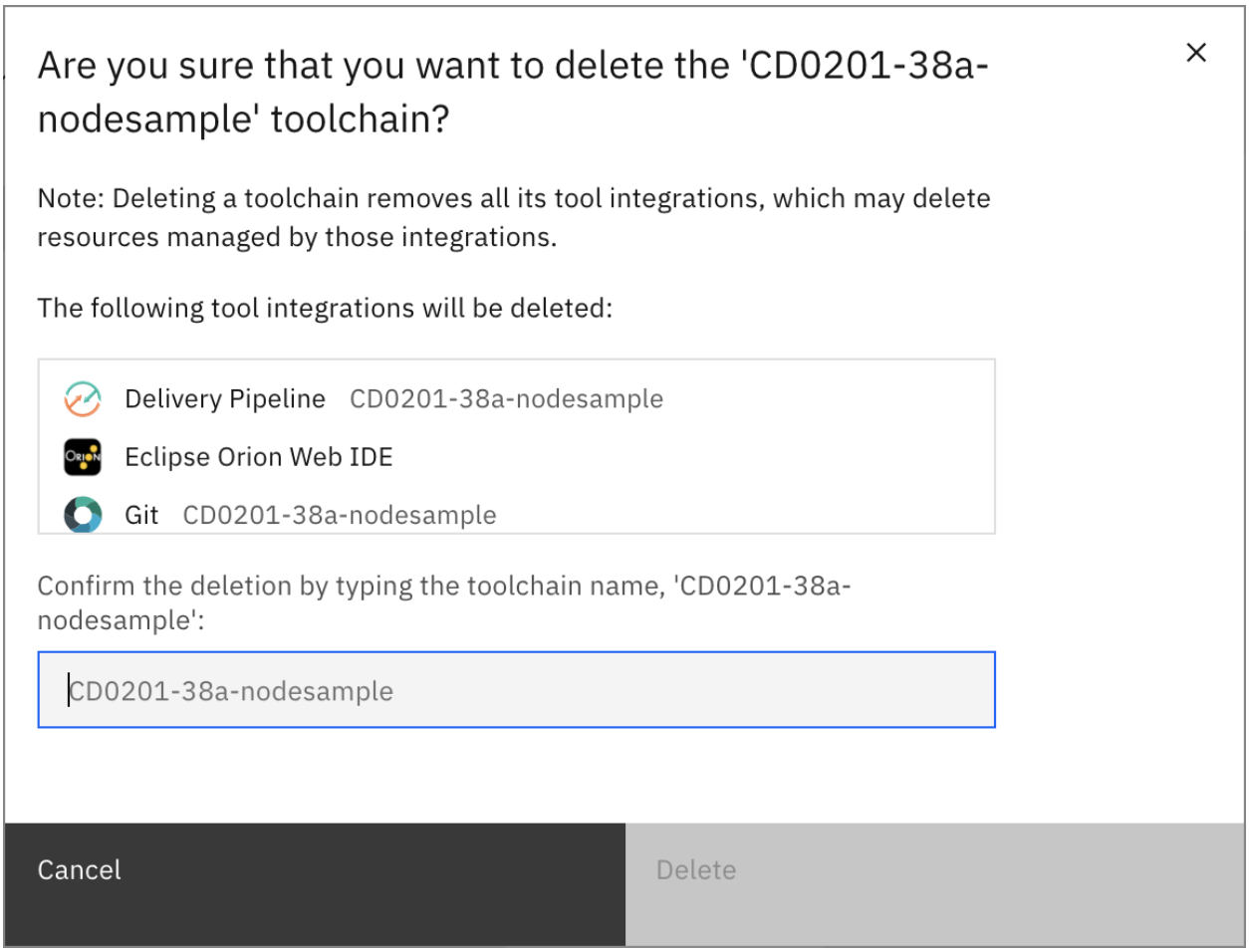
In this step, you clean up the application that you created by removing it form Continuous Delivery Toolchains dashboard:

1. Open the Continuous Delivery Toolchains dashboard web page. Click the **menu** icon and click **Delete**, as shown in the next image.



Deleting the application

2. Type your application name in the box and click **Delete** to permanently delete the project, as shown in the following image.



Permanently deleting the project

3. This will take you to the toolchains page, where you will see that the toolchain **CD0201-38a-nodesample** page doesn't exist anymore.
4. Log out of IBM Cloud.
5. Close your web browser.

Summary

In this lab, you used IBM Cloud Continuous Delivery to manage your IBM Cloud application that is written for the IBM SDK for Node.js server runtime.

Then, you saved your changes into the Git repository. Through the Delivery Pipeline, you pushed our committed source code changes to your IBM Cloud application.

In the last part of the lab, you deployed your changes directly from the project workspace. You also used the IBM Cloud Live Sync feature to push changes to static files without redeploying the application and without needing to restart it.

Next Steps

In this lab you, enabled your application to use IBM Cloud Continuous Delivery. You created a Git repostiory, viewed and edited code in the Eclipse Orion Web IDE, and then built, deployed and tested your application in IBM Cloud. If you are interested in continuing to learn about DevOps and Git, explore IBM DevOps.