Quick Lab: Create and deploy Enterprise-Scale Multi-Language Microservices

Chris Bailey, IBM Runtimes, baileyc@uk.ibm.com
Graham Charters, WebSphere Liberty Development, gcharters@uk.ibm.com

Create and Deploy Enterprise Multi-Language Microservices

Introduction

This lab walks you through the steps required to create, build and run a two microservices: an Node.js service built using Express.js, and a Java service built using OpenLiberty.

The microservices are built using the **IBM Cloud App Service** in a web browser. This offers a guided approach that enables you build cloud native apps in minutes. It eliminates the need to manually edit configuration files and allows you to easily add a select set of IBM Cloud Services to you project, provisioning those services as required. Moreover, you can easily attach a DevOps Toolchain to your project using the **IBM Cloud Continuous Delivery Service** that supports continuous integration and deployment to either a **Cloud Foundry** or **IBM Cloud Container Service** – a managed Kubernetes-based environment on the IBM Cloud. While the Cloud App Service is free, it simplifies the construction and deployment of cloud native apps by easily integrating with other paid IBM Cloud Services.

The microservices will then be enhanced to call each other, and you'll be able to visualise the request starting in the Node.js microservice and passing to the Java microservice using **OpenTracing and Zipkin**. OpenTracing is a community standard for how to track requests that travel across microservices, and Zipkin is an open source implementation of a monitor service that collects and visualizes the data.

This lab requires an account with IBM Cloud. You may use your existing account or create a new account (Section 2).

Setting Up

Before starting this lab, please do the following:

1. Go to http://ibm.biz/startmylab

2. Select **Create and deploy Enterprise-Scale Multi-Language Microservices** lab from the dropdown and click Ok
    a. You will be brought to the sign up page to register for an IBM Cloud Platform account. If you do not have an account, please register for one.
    b. This lab does require you to have an IBM Cloud Platform account in order to create the applications using the IBM Cloud App Service.

3. Clone the project for this lab:

    a. Click on the Launchpad (rocket logo) in the toolbar, search for "Terminal" and click on the icon.

    b. Clone the GitHub project containing the FoodTracker application we are going to extend:

    ```
    git clone http://github.com/seabaylea/NodeLibertyLab
    ```

    c. A copy of these instructions are available included in the project. You can open the instructions using:

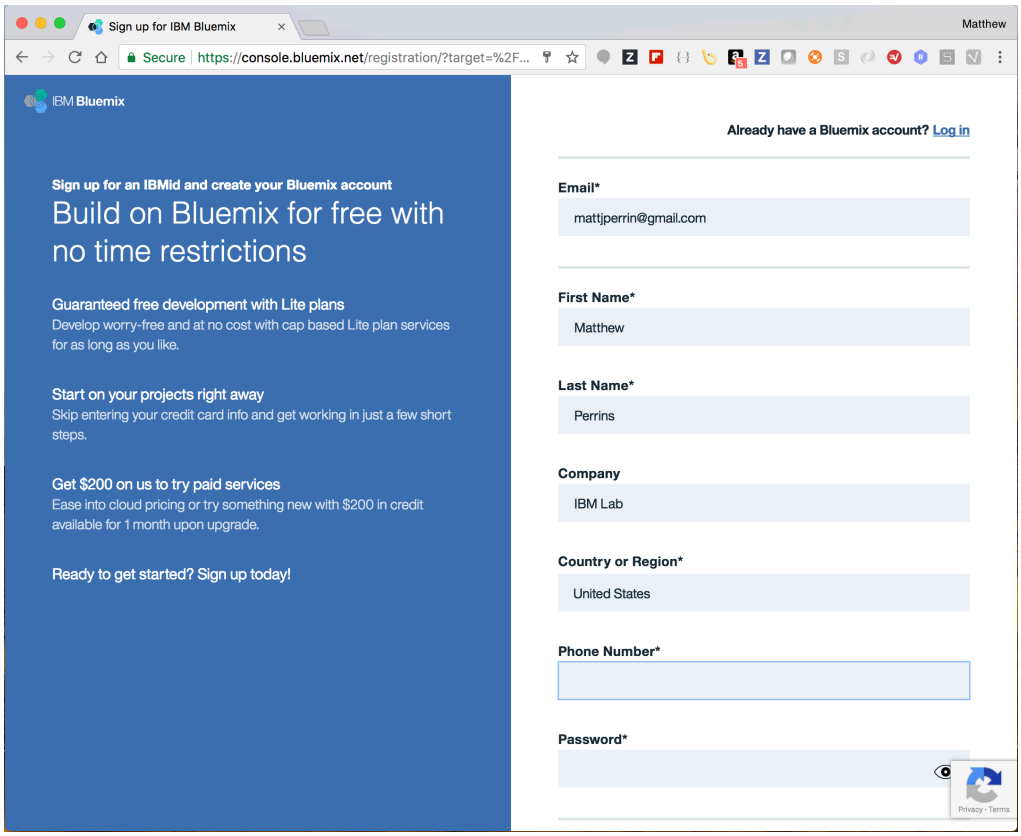
    ```
    open "NodeLibertyLab/ Create and Deploy Enterprise Multi-
       Language Microservices.pdf"
    ```

NEXT STEPS:
If you **do** have an existing IBM Cloud account, skip Section 1 (Create an IBM Cloud Account) and proceed to Section 2 (Create a Java Liberty microservice)

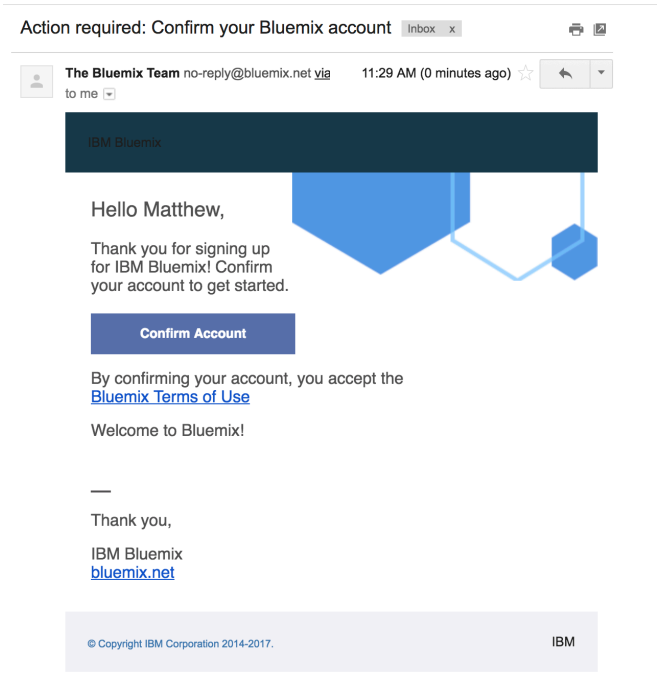
1. Create an IBM Cloud Account

If you **do** have an existing IBM Cloud account, skip to the next section.

1. To run these lab instructions, you will need to create a trial account on the IBM Cloud. You can do this free of charge and all the instructions contained in this lab do not incur any additional costs.
2. Create an IBM Cloud Trial account by navigating to this link. https://console.bluemix.net, and click on **Create a free Account.**
3. Complete the Registration Form with your details and a valid email address.

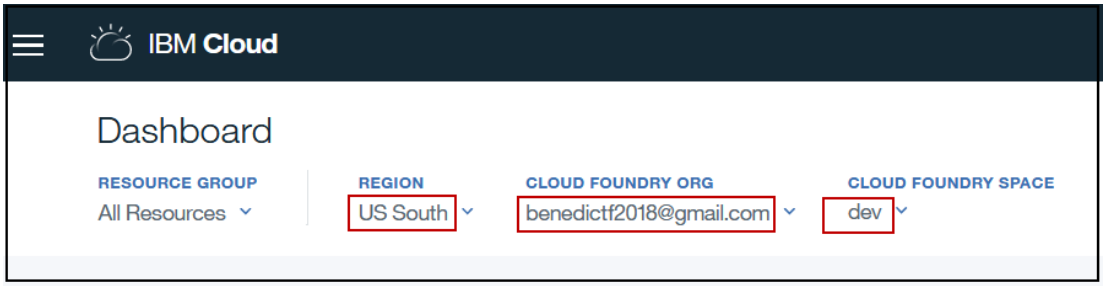


4. Click on **Create Account.**
5. Confirm your registration by accessing your email and clicking on **Confirm Account.**



6. You can now log into IBM Cloud.
7. Click on **Login** and enter your email and password
8. You will finally see the Dashboard View, which will be empty as you have not created any services or apps at this point. Please note the **Region**, **Org** and **Space** settings.

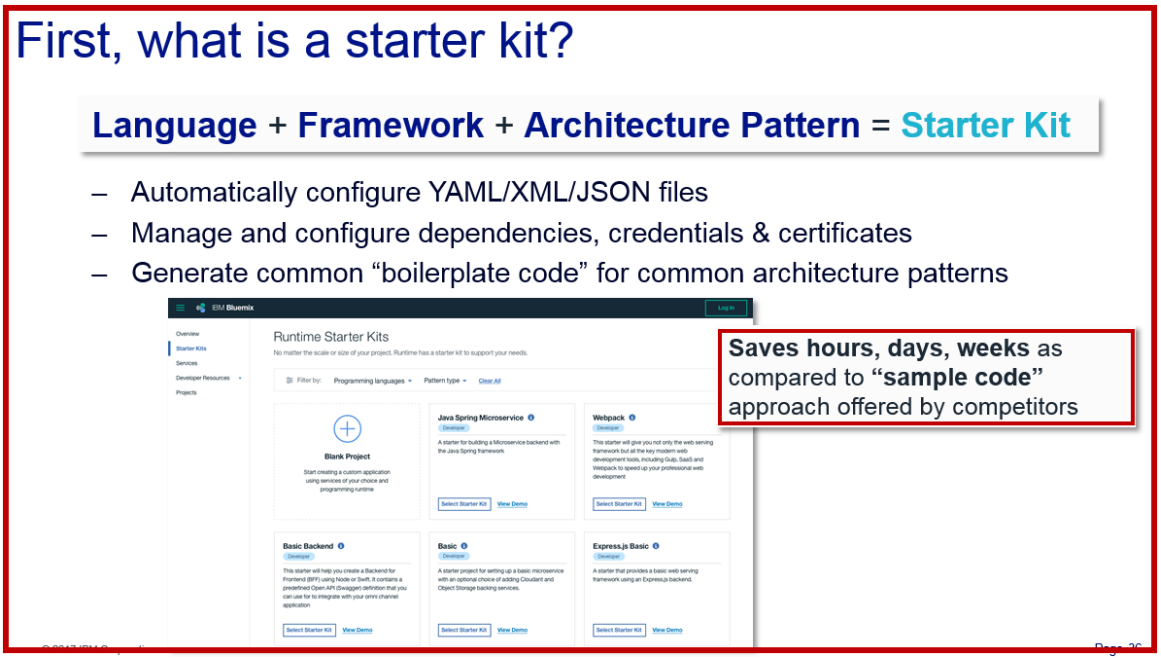
   Typically this will be **US South**, the Org setting will be same as your email address and by default you are placed in a Space called **dev**, as shown below.



2. Create a Java Liberty Microservice on IBM Cloud

In this section, you will use the IBM Cloud to create a Java Liberty microservice application using the IBM Cloud App Service from IBM.

- Access the IBM Cloud App Service
- Create from the Java Microprofile Microservice Starter Kit



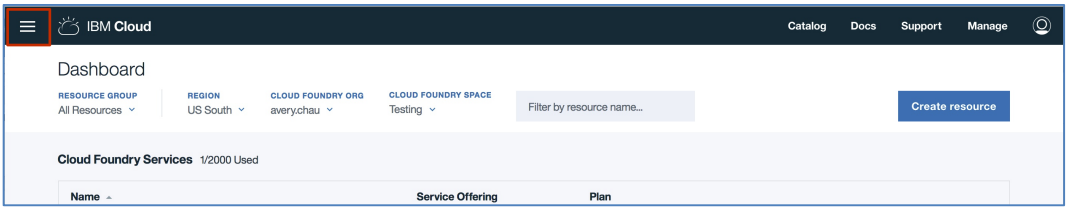
The IBM Cloud App Service jump-starts cloud native development by allowing a developer to create a project, pick an application starter kit and deploy a production-ready application within minutes to the IBM Cloud.

The platform's code generation technology creates a starter application in the developer's preferred language and framework, which can be tailored to their needs and use case. Any services that are required in support of the use case, are provisioned automatically.

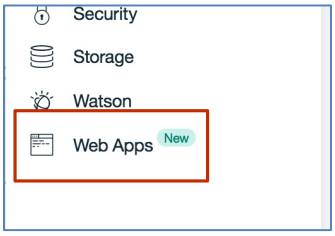
Developers can debug and test on their local workstation or in the cloud, and then use the DevOps Toolchain to collaborate with others to automate the delivery process.

1. Start by logging in to IBM Cloud.

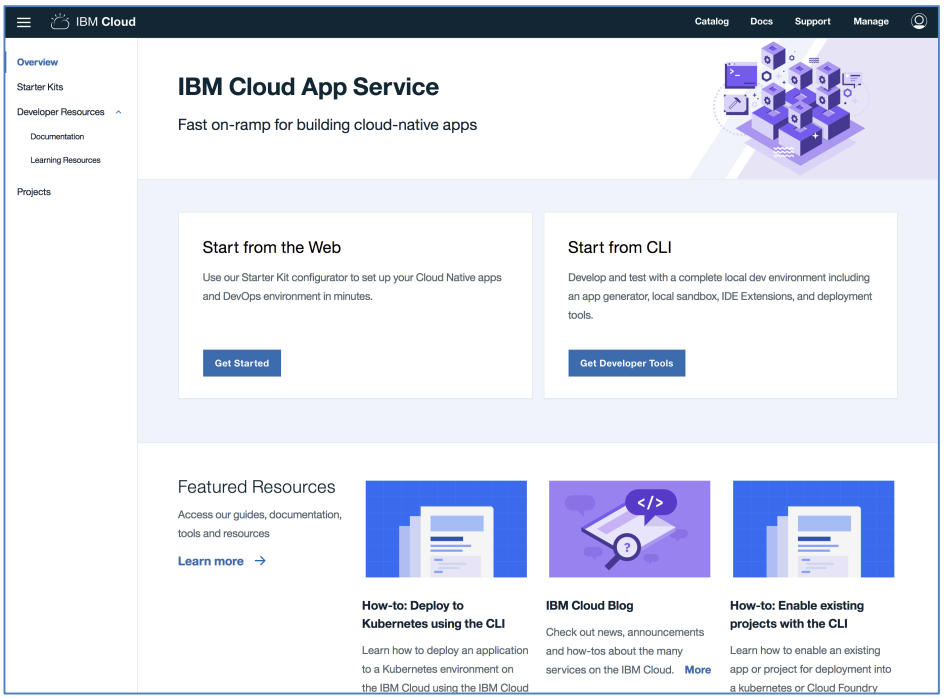
   While logged into the IBM Cloud, click on the **Navigation** menu in the header:



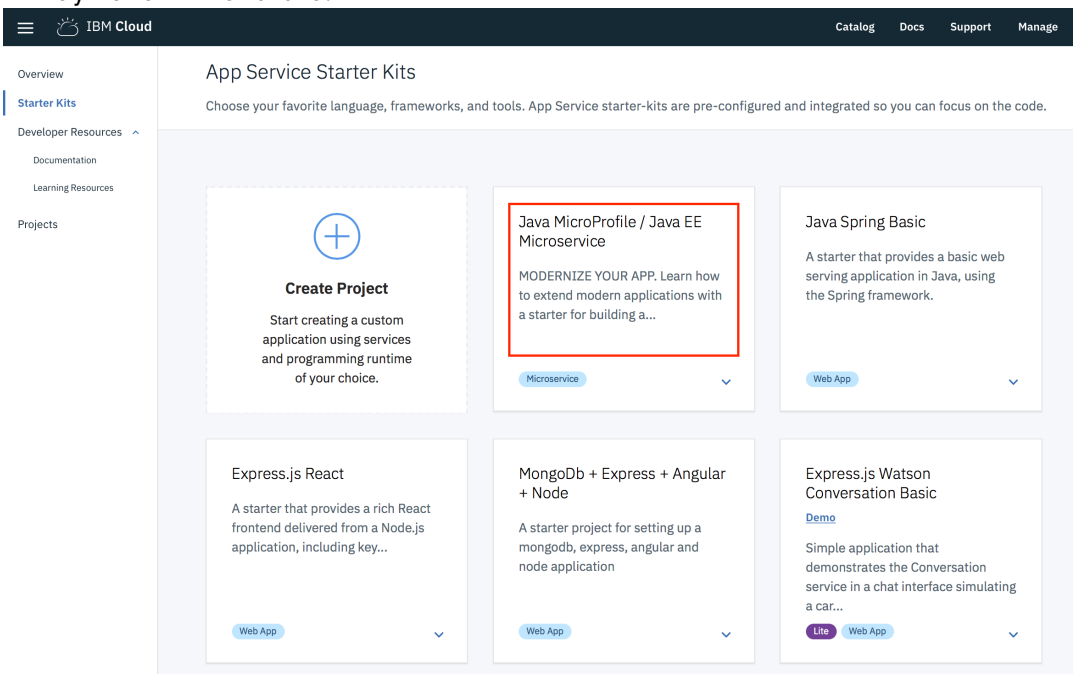
   Then click on **Web Apps** to access the **IBM Cloud App Service** cloud-native development experience on IBM Cloud.



2. Take a few moments to explore the main Overview screen for the IBM Cloud App Service. From here you may access and discover hand-picked content related to cloud-native development including the **Resources**, **Community**, and **Starter Kits** sections. As explained IBM Cloud provides two approaches to building cloud native apps – a web-based and a CLI-based approach. This Lab will use the IBM Cloud App Service, which provides a web console.



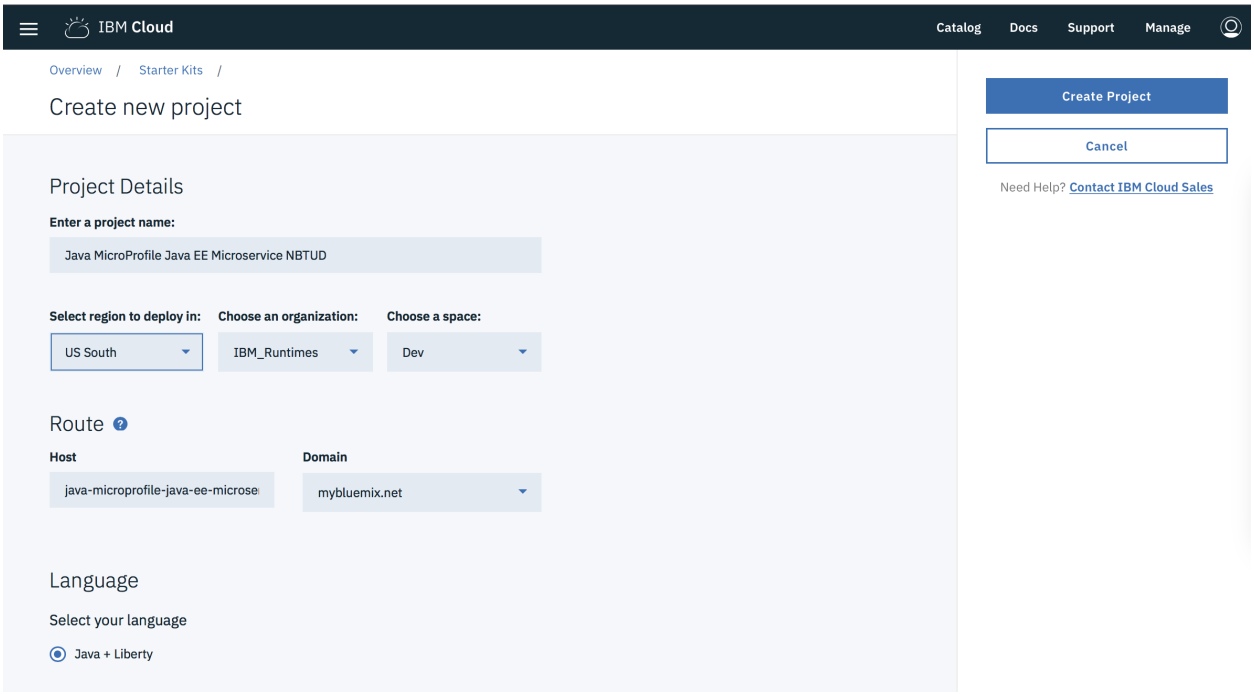
3. Navigate to the **Starter Kits** menu on the left-hand side. You will see a range of starter kits that are designed to provide production code starting point for a variety of common cloud-native development use cases. They include patterns and technology framework stacks for key programming languages that are commonly used by developers. Have a look through the list of starter kits for other types of projects you may have in the future.



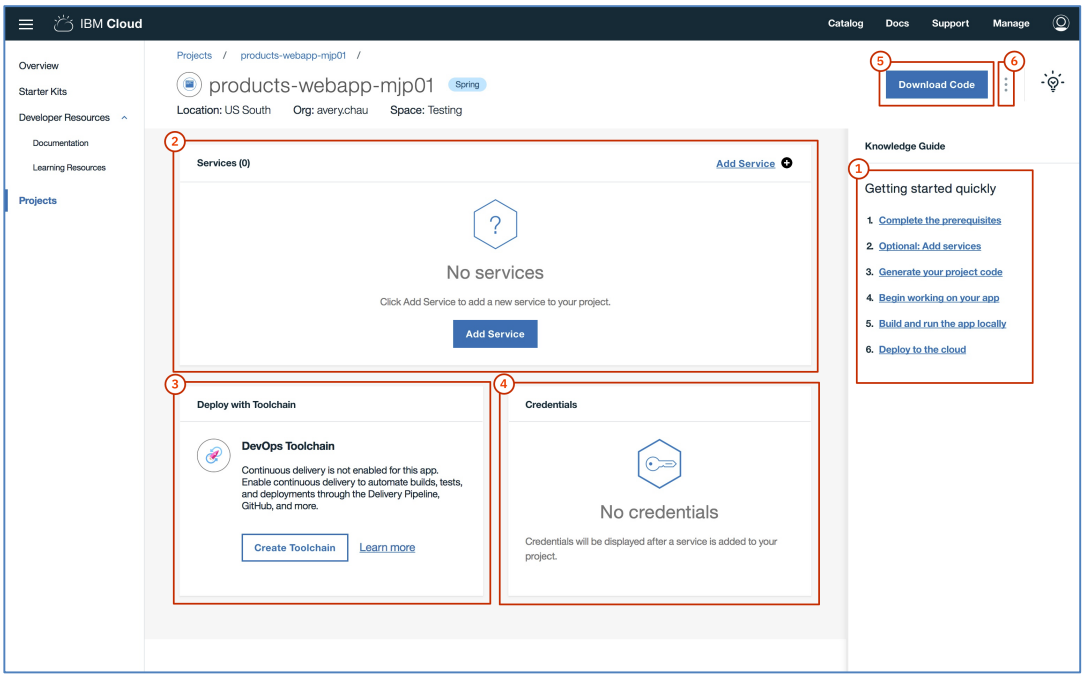
4. Scroll down the list and click on the tile labelled **Java Microprofile / Java EE Microservice** to select the Starter Kit.

5. Within the Create New Project view, enter a project name of "**JavaMicroservice**". Once you've completed the form, click on the **Create Project** button.

   NOTE: Please ensure that you select **US South** as the Region.



6. The App Configuration/Details view is now displayed. This view is used to manage various aspects of an app:
   - The right-hand side (1) offers a Getting Started Quickly guide.
   - The main body has an area where you can create new and associate existing services to the app (2)
   - The bottom of the screen offers an area where you may configure a DevOps toolchain (3)
   - The bottom of the screen also offers an area where you may access credentials to any services that you have added to this app (4)
   - At the top of the screen, it is possible to download the code (5) to the app without any DevOps configuration.
   - Using the menu in the top right (6), you can also rename of the project or delete it.



7. Click “Download Code” to download the project to your local machine.

8. Open a Terminal window to access the command line:
   Click on the Launchpad (rocket logo) in the toolbar, search for “Terminal” and click on the icon.

9. Go to the project directory

```
cd ~/Downloads/JavaMicroservice
```

You now have a cloud ready Java microservice, based on Microprofile and OpenLiberty. Because we are going to use this as part of a multi-microservice deployment, we now want to add OpenTracing in order to be able to track requests across services.

1. Edit the `pom.xml` to add a dependency on MicroProfile 1.3 and OpenTracing:
   Either copy the pom.xml from the Git project you cloned (recommended):

```
cp /Users/user/NodeLibertyLab/LibertyAssets/pom.xml .
```

   Or manually update the pom.xml file using the following steps:

   a. Open the file:

```
open -a TextEdit ./pom.xml
```

   b. Add the following dependencies into the dependencies section:

```
    <dependency>
       <groupId>io.opentracing</groupId>
```

```
        <artifactId>opentracing-api</artifactId>
        <version>0.30.0</version>
    </dependency>
    <dependency>
        <groupId>org.eclipse.microprofile.opentracing</groupId>
        <artifactId>microprofile-opentracing-api</artifactId>
        <version>1.0</version>
    </dependency>
```

c. Adding the following to the plugins section:

```
            <plugin>
                <groupId>com.googlecode.maven-download-plugin</groupId>
                <artifactId>download-maven-plugin</artifactId>
                <version>1.4.0</version>
                <executions>
                    <execution>
                        <id>install-tracer</id>
                        <phase>prepare-package</phase>
                        <goals>
                            <goal>wget</goal>
                        </goals>
                        <configuration>
                            <url>https://repo1.maven.org/maven2/net/wasd
ev/wlp/tracer/liberty-opentracing-zipkintracer/1.0/liberty-opentracing-
zipkintracer-1.0-sample.zip</url>
                            <unpack>true</unpack>
                            <outputDirectory>${project.build.directory}/
liberty/wlp/usr</outputDirectory>
                            <!-- <md5>/md5> -->
                        </configuration>
                    </execution>
                </executions>
            </plugin>
```

and replace the ‘net.wasdev.wlp.maven.plugins’ plugin section with the following:

```
            <plugin>
                <groupId>net.wasdev.wlp.maven.plugins</groupId>
                <artifactId>liberty-maven-plugin</artifactId>
                <version>2.0</version>
                <configuration>
                    <assemblyArtifact>
                        <groupId>io.openliberty</groupId>
                        <artifactId>openliberty-runtime</artifactId>
                        <version>18.0.0.1</version>
                        <type>zip</type>
                    </assemblyArtifact>
                    <configFile>src/main/liberty/config/server.xml</conf
igFile>
                    <packageFile>${package.file}</packageFile>
                    <include>${packaging.type}</include>
                    <bootstrapProperties>
                        <default.http.port>${testServerHttpPort}</defaul
t.http.port>
```

```
                        <default.https.port>${testServerHttpsPort}</defa
ult.https.port>
                    </bootstrapProperties>
                </configuration>
            </plugin>
```

2. Edit the `src/main/liberty/config/server.xml` file to load the additional features: Either copy the server.xml from the Git project you cloned (recommended):

```
cp /Users/user/NodeLibertyLab/LibertyAssets/src/main/liberty/config/server.xml
./src/main/config/
```

   Or manually update the pom.xml file using the following steps:

   a. Open the file:

```
open -a TextEdit ./src/main/liberty/config/server.xml
```

   b. Set the feature manager section to the following:

```
<feature>microprofile-1.3</feature>
<feature>jndi-1.0</feature>
<feature>mpOpenTracing-1.0</feature>
<feature>usr:opentracingZipkin-0.30</feature>
```

   c. And add the following anywhere in the server section:

```
<keyStore id="defaultKeyStore" password="yourpassword" />
```

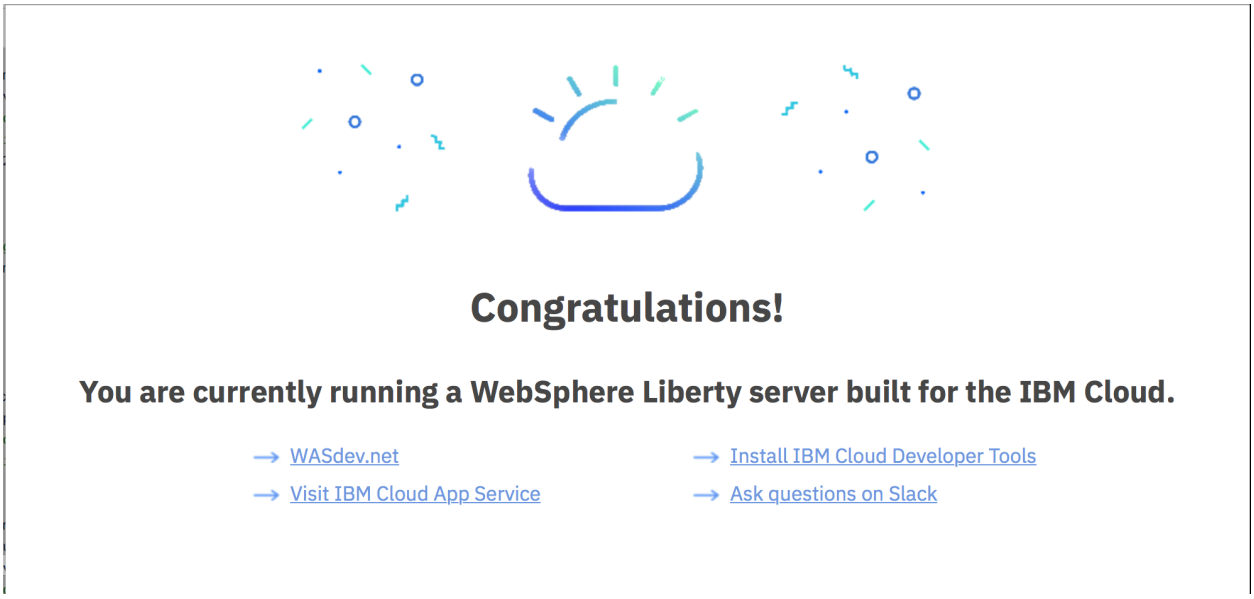
3. The Microprofile/Liberty microservice is now ready to install and run:

```
mvn install
mvn prepare-package
mvn liberty:run-server
```

At the end of the startup messages, you will see a message similar to the following:

```
[INFO] [AUDIT   ] CWWKT0016I: Web application available (default_host): http://172.16.30.1:9080/JavaMi-
croProfileJavaEEMicroserviceNBTUD/
```

Copy that URL and open it in your browser. You should see a splash screen similar to the following:

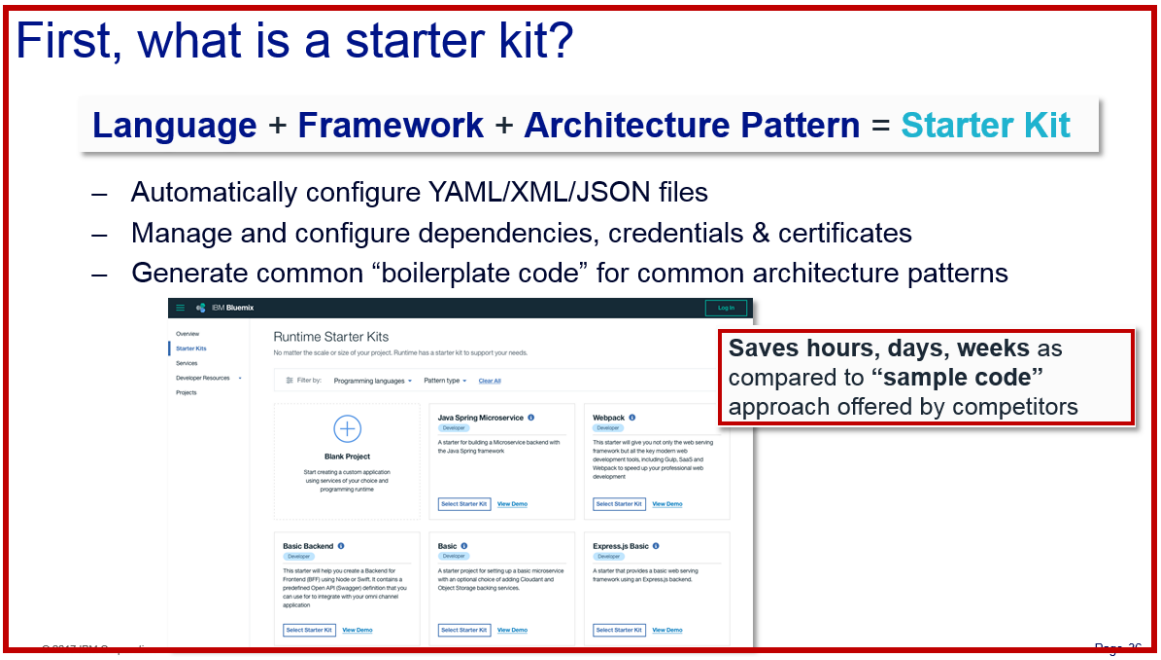


You're now ready to create a Node.js microservice that calls this URL.

3. Create a Node.js Express Microservice on IBM Cloud

In this section, you will use the IBM Cloud to create an Express.js microservice application using the IBM Cloud App Service from IBM.

- Access the IBM Cloud App Service
- Create from the Express.js microservice Starter Kit



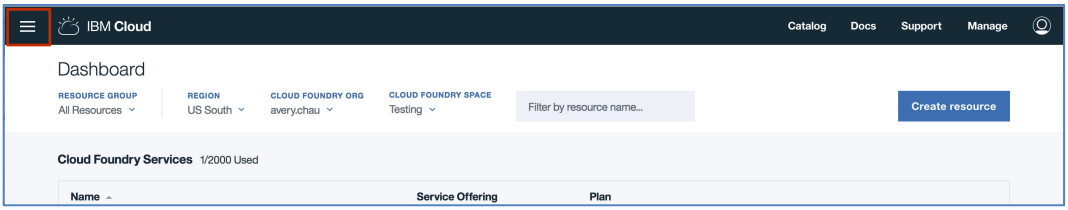
The IBM Cloud App Service jump-starts cloud native development by allowing a developer to create a project, pick an application starter kit and deploy a production-ready application within minutes to the IBM Cloud.

The platform's code generation technology creates a starter application in the developer's preferred language and framework, which can be tailored to their needs and use case. Any services that are required in support of the use case, are provisioned automatically.

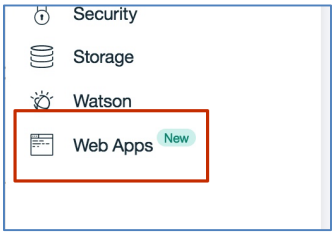
Developers can debug and test on their local workstation or in the cloud, and then use the DevOps Toolchain to collaborate with others to automate the delivery process.

4. Start by logging in to IBM Cloud.

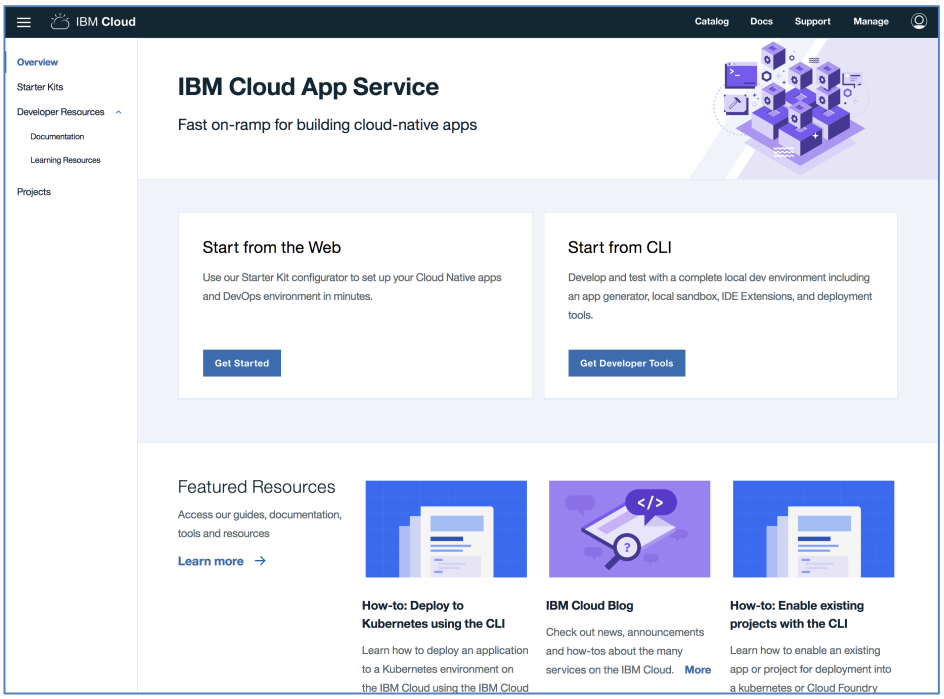
   While logged into the IBM Cloud, click on the **Navigation** menu in the header:



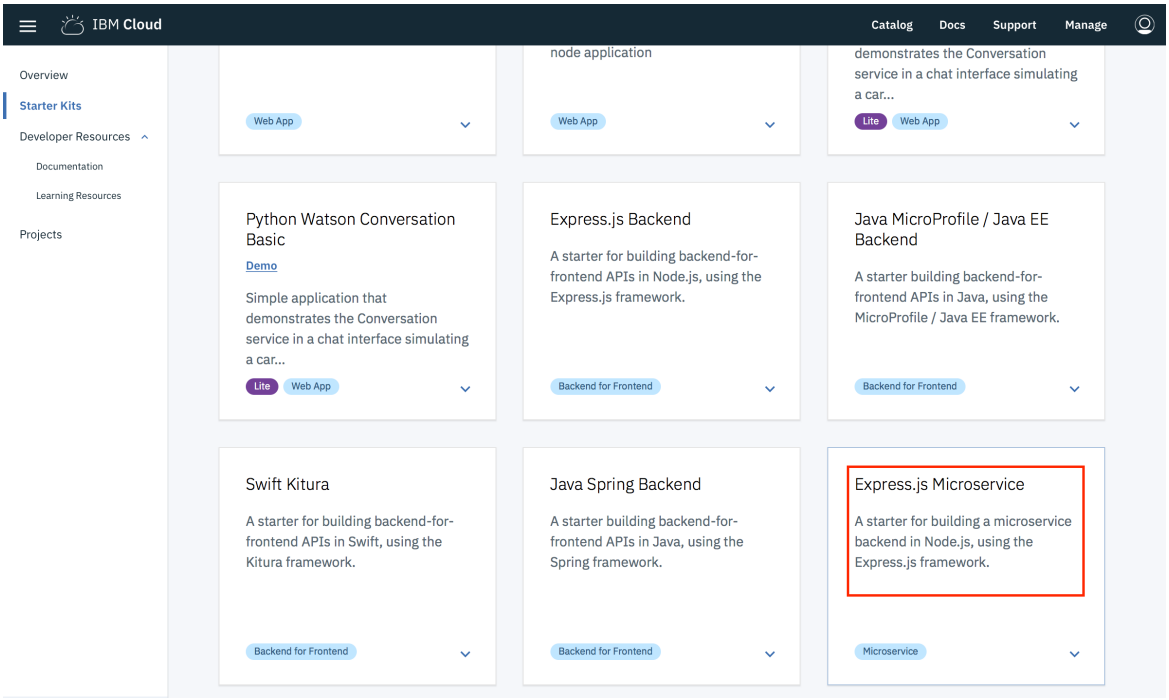
   Then click on **Web Apps** to access the **IBM Cloud App Service** cloud-native development experience on IBM Cloud.



5. Take a few moments to explore the main Overview screen for the IBM Cloud App Service. From here you may access and discover hand-picked content related to cloud-native development including the **Resources**, **Community**, and **Starter Kits** sections. As explained IBM Cloud provides two approaches to building cloud native apps – a web-based and a CLI-based approach. This Lab will use the IBM Cloud App Service, which provides a web console.



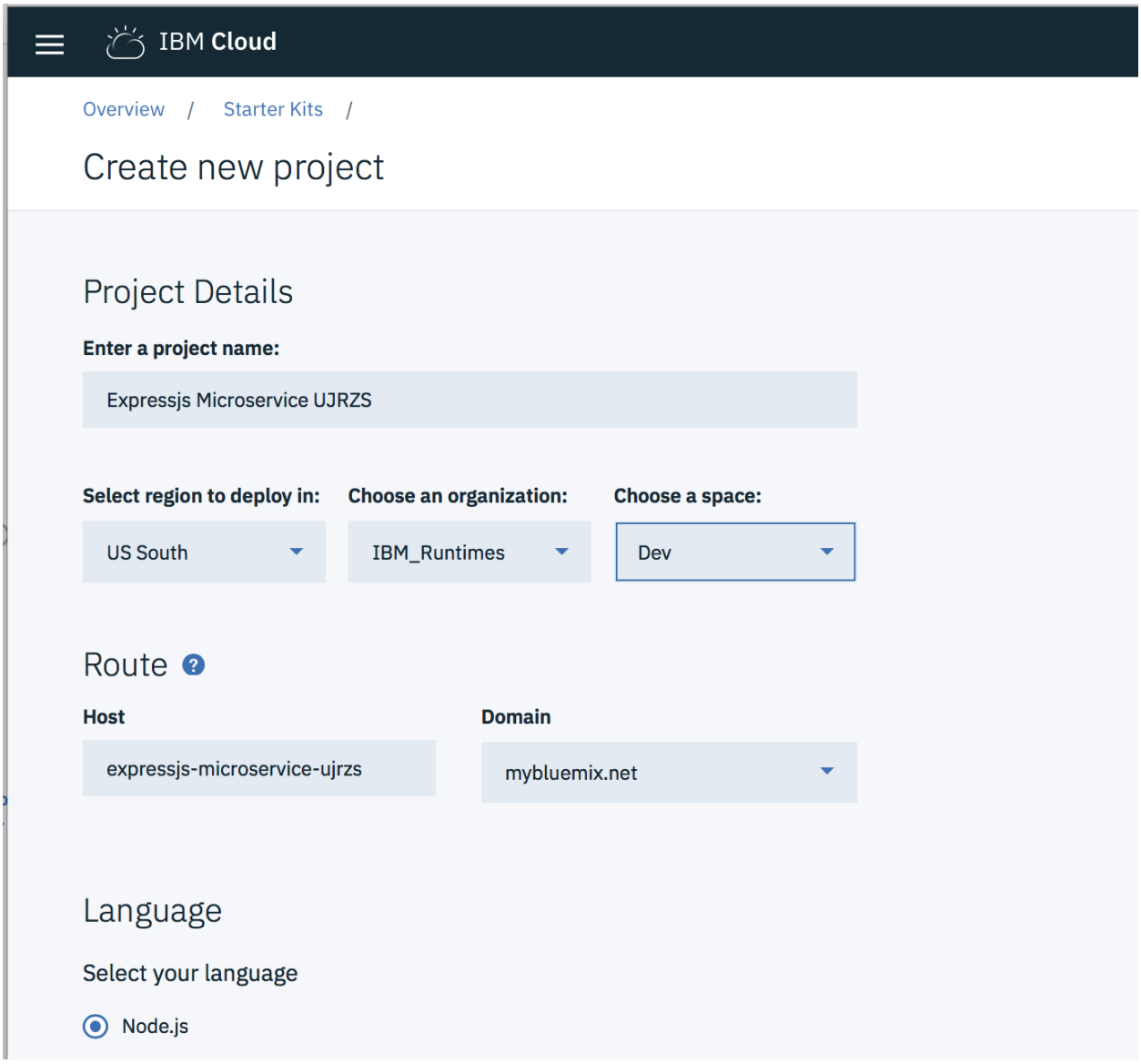
6. Navigate to the **Starter Kits** menu on the left-hand side. You will see a range of starter kits that are designed to provide production code starting point for a variety of common cloud-native development use cases. They include patterns and technology framework stacks for key programming languages that are commonly used by developers. Have a look through the list of starter kits for other types of projects you may have in the future.



7. Scroll down the list and click on the tile labelled **Express.js Microservice** to select the Starter Kit.

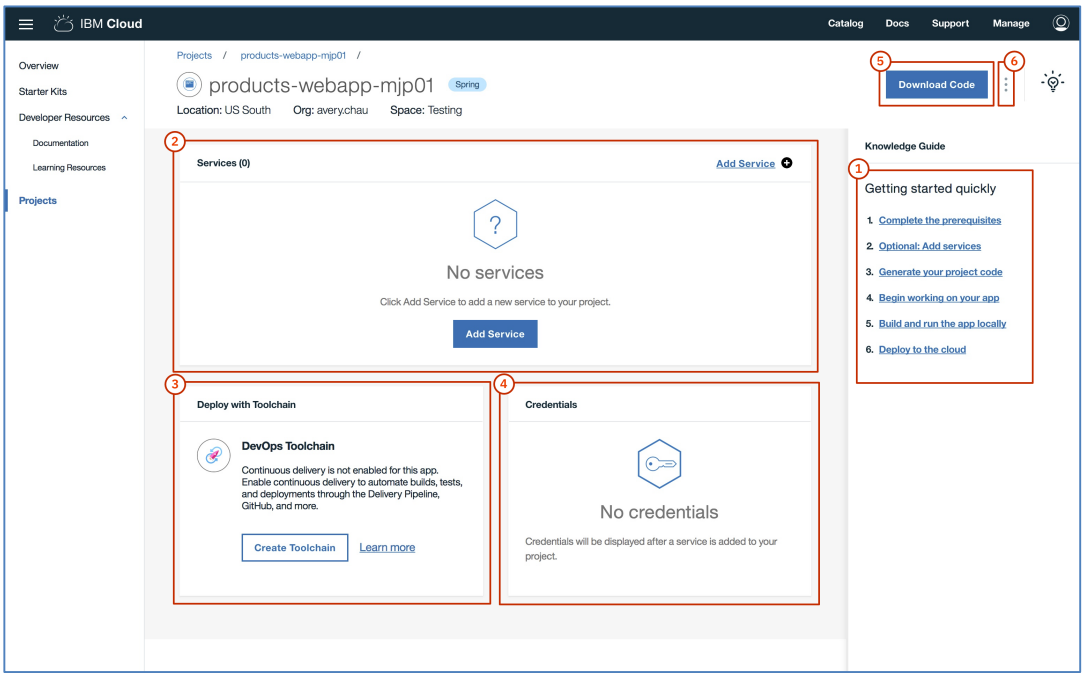
8. Within the Create New Project view, name the project "**NodeMicroservice**". Once you've completed the form, click on the **Create Project** button.

   NOTE: Please ensure that you select **US South** as the Region.



9. The App Configuration/Details view is now displayed. This view is used to manage various aspects of an app:
   - The right-hand side (1) offers a Getting Started Quickly guide.

- The main body has an area where you can create new and associate existing services to the app (2)
- The bottom of the screen offers an area where you may configure a DevOps toolchain (3)
- The bottom of the screen also offers an area where you may access credentials to any services that you have added to this app (4)
- At the top of the screen, it is possible to download the code (5) to the app without any DevOps configuration.
- Using the menu in the top right (6), you can also rename of the project or delete it.



10. Click “Download Code” to download the project to your local machine.

11. Open a Terminal window to access the command line:
    Click on the Launchpad (rocket logo) in the toolbar, search for “Terminal” and click on the icon.

12. Go to the project directory

```
cd ~/Downloads/NodeMicroservice
```

You now have a cloud ready Node.js microservice. Because we are going to use this as part of a multi-microservice deployment, we now want to add OpenTracing in order to be able to track requests across services.

1. Open the `server/server.js` file in your project and enable “`appmetrics-zipkin`”

Either copy the server/server.js from the Git project you cloned:

```
cp /Users/user/NodeLibertyLab/NodeAssets/server/server.js ./server/
```

Or edit the `server/server.js` file as follows.

a. Open the file:

```
open –a TextEdit ./server/server.js
```

b. Uncomment the code at the top of the file to enable Zipkin request tracking. And optionally change the `serviceName`:
The top of your file should be:

```
var appzip = require('appmetrics-zipkin')({
     host: 'localhost',
     port: 9411,
     serviceName:'NodeMicroservice'
 });
```

Once you've made the changes, save the file.

This has enabled OpenTracing for any request received by the Node.js microservice, and for any outbound requests it makes to other services.

2. Update the Node.js microservice to make calls against the Java microservice: edit the `server/server.js` file as follows.

a. Open the file:

```
open –a TextEdit ./server/server.js
```

b. Add the following code at the bottom of the file:

```
var options = {
  host: 'localhost',
  port: 9080,
  path: '/JavaMicroservice/',
  headers: {
    'Accept': 'text/html'
  }
};

const http = require('http');

app.get('/', (req, res) => {
    http.get(options, resp => {
        resp.setEncoding("utf8");
        let body = "";
        resp.on("data", data => {
            body += data;
        });
        resp.on("end", () => {
            res.send(body)
        });
    });
});
```

Once you've made the changes, save the file.

This has updated the Node.js microservice to make a request for the splash screen from the Java service every time it is called on '/', effectively acting as a proxy.

3. Install the dependencies, and run the application:

```
npm install
npm start
```

4. Use your browser to try out your new Node.js microservice by loading its built-in monitoring dashboard:
   Monitoring Dashboard: http://localhost:3000/appmetrics-dash

This should show you the same splash screen as you saw for the Java microservice, because the code you added to the Node.js microservice just proxies the result retrieved from the Java microservice.

4. View Request Tracking in Zipkin

OpenTracing is a standard for how to add request tracking to microservices, with Zipkin as a commonly found implementation of that standard that has been implemented for multiple languages and frameworks.

As well as the configuration of the Node.js and Java microservices to provide Zipkin data, you also need to install and run a Zipkin server that collects the data from the microservices and provides visualization and analysis. The Zipkin server is easy to install and run either locally or in the cloud using Docker.

1. Launch and run the Zipkin server request tracking monitoring service:

   ```
   docker run -d -p 9411:9411 openzipkin/zipkin
   ```

   This will start running the Zipkin monitoring service in a Docker container on port 9411. Zipkin is and open source monitoring solution that is designed to integrated into Docker and Kubernetes deployments, and monitors requests across individual microservices.

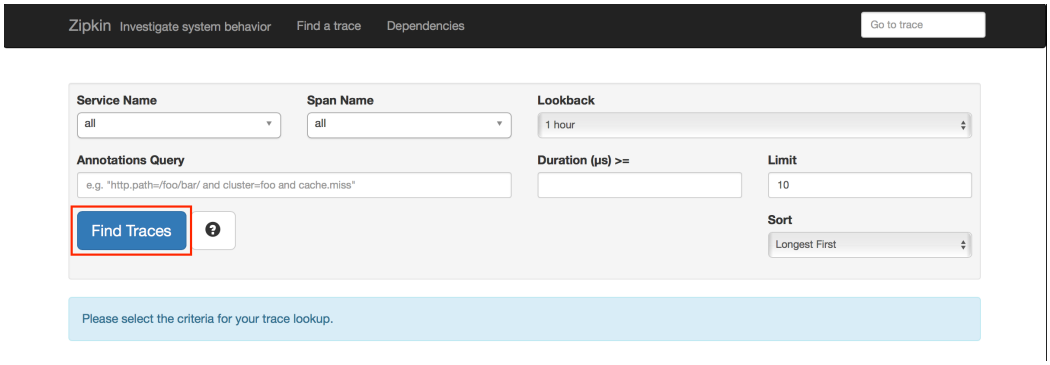
2. Make a request of the Node.js microservice, which in turn makes a request of the Java microsevice:
   Node.js microservice http://localhost:3000/

   Reload the page several times. This can be done using the ⌘+R key shortcut.

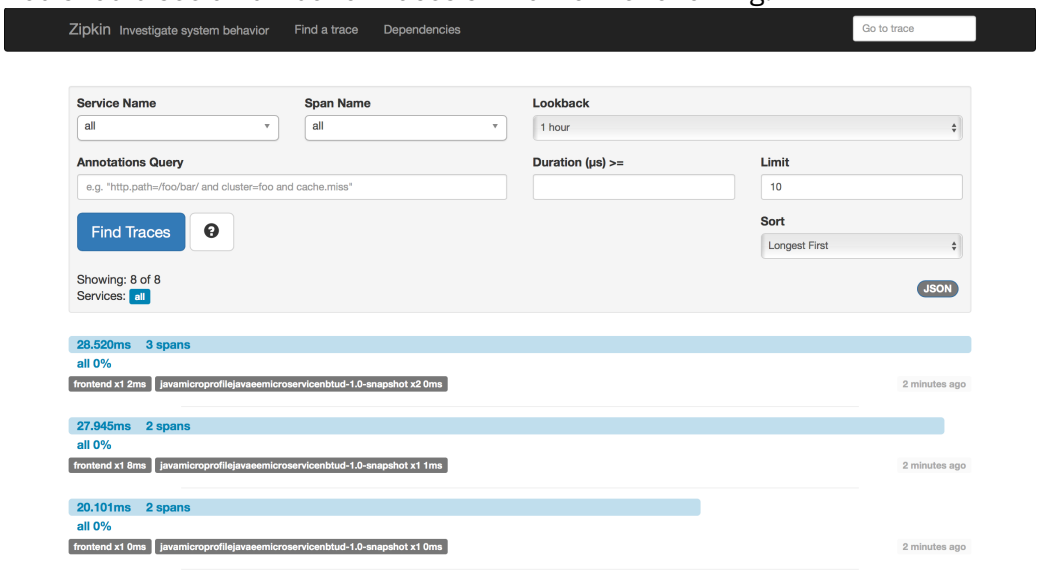
3. View the Zipkin request tracking data
   Now that the application is running, Zipkin is automatically collecting request tracking "spans" from the Node.js and Java microservices.

   a. Visit the following URL:
   Zipkin Monitoring: http://localhost:9441

   b. Select "Find Traces" to view all of the available Request Tracking spans that Zipkin is aware of.

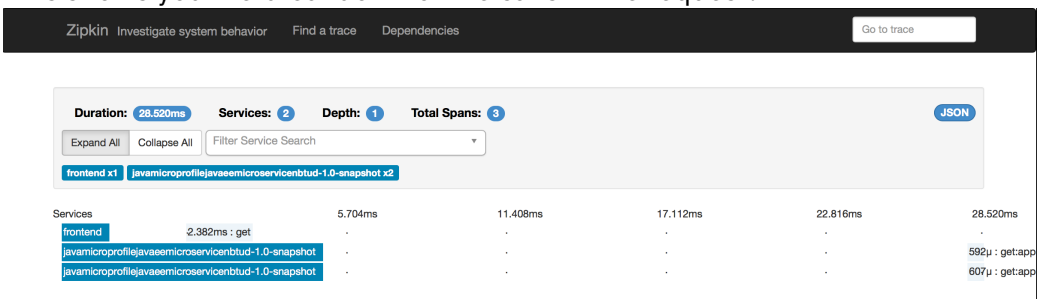


You should see a number of traces similar to the following:

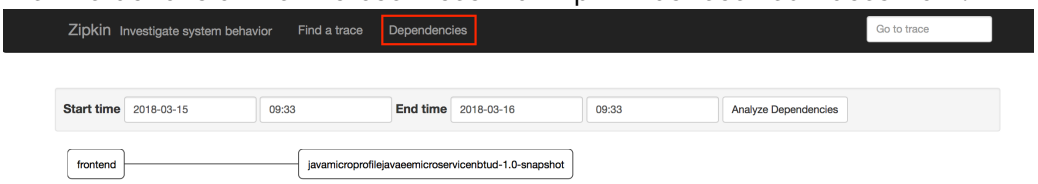


This shows that there are a number of traces that span multiple services. Here the top longest request took 28.520ms and spanned both “frontend” and “javamicroprofilejavaeemicroservicebtud-1.0-snapshot”.

c. Click on a trace to get more detail
This shows you the breakdown of the calls in the request:



4. View the discovered dependencies
Additionally you can click on the “Dependencies” menu item to see a visualization of the interactions of the microservices that Zipkin has received traces from:



This shows that Zipkin has discovered the relationship between our Node.js and Java

microservices. As the number of inter-related microservices increases, this becomes more and more valuable.

5. Conclusion

Congratulations! You have completed the lab.

You have successfully built both Node.js and Java Microservices that are cloud-ready, and seen how these two languages can work together with consistent monitoring and request tracking.

6. Resources

- IBM Cloud App Service: Build a Node.js app to deploy on Containers to IBM Cloud - https://www.youtube.com/watch?v=1qAjtduM2TY
- IBM Cloud App Service: http://bluemix.net/developer/appservice
- IBM Cloud Container Service: https://www.ibm.com/cloud/container-service
- IBM Cloud Garage Method: https://www.ibm.com/cloud/garage/
- IBM Cloud Continuous Delivery Service: https://www.ibm.com/cloud/continuous-delivery
- IBM Cloud DevOps: https://bluemix.net/devops
- IBM Cloud Developer Tools: https://www.youtube.com/watch?v=z-ByHuI41dU&t=76s
- IBM DevOps Overview: https://www.youtube.com/watch?v=psiMSUfn9oA
- Develop a Kubernetes App Toolchain: https://www.ibm.com/cloud/garage/tutorials/use-develop-kubernetes-app-toolchain?task=2
- IBM Cloud App Service: https://www.youtube.com/watch?v=VBZTwvLkGIk
- IBM Cloud Developer Tools (IDT): https://console.bluemix.net/docs/cloudnative/idt/index.html
- Using the IDT: https://www.youtube.com/watch?v=z-ByHuI41dU&t=76s
- Jump Start Your Microservice Development: https://www.youtube.com/watch?v=1HdtILoL6O4
- Spring@IBM Developer Center: http://developer.ibm.com/java/spring
- Node@IBM Developer Center: http://developer.ibm.com/cloud/node

Acknowledgements and Disclaimers (v8)

Copyright © 2017 by International Business Machines Corporation (IBM). No part of this document may be reproduced or transmitted in any form without written permission from IBM.
U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.
Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted according to the terms and conditions of the agreements under which they are provided.
IBM products are manufactured from new parts or new and used parts.
In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."
Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.
Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.
References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.
Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.
It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer is in compliance with any law.
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 in connection with this publication 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. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a particular, purpose.**
The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.
IBM, the IBM logo, ibm.com, Aspera®, Bluemix, Blueworks Live, CICS, Clearcase, Cognos®, DOORS®, Emptoris®, Enterprise Document Management System™, FASP®, FileNet®, Global Business Services®, Global Technology Services®, IBM ExperienceOne™, IBM SmartCloud®, IBM Social Business®, Information on Demand, ILOG, Maximo®, MQIntegrator®, MQSeries®, Netcool®, OMEGAMON, OpenPower, PureAnalytics™, PureApplication®, pureCluster™, PureCoverage®, PureData®, PureExperience®, PureFlex®, pureQuery®, pureScale®, PureSystems®, QRadar®, Rational®, Rhapsody®, Smarter Commerce®, SoDA, SPSS, Sterling Commerce®, StoredIQ, Tealeaf®, Tivoli® Trusteer®, Unica®, urban{code}®, Watson, WebSphere®, Worklight®, X-Force® and System z® Z/OS, are trademarks of International Business Machines Corporation, 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.