



## Quick Lab: Create Highly Scalable Web Application Microservices with Node.js

**Chris Bailey**, IBM Runtimes, [baileyc@uk.ibm.com](mailto:baileyc@uk.ibm.com)

**Michael Dawson**, IBM Runtimes, [Michael\\_Dawson@ca.ibm.com](mailto:Michael_Dawson@ca.ibm.com)



# Create Highly Scalable Web Application Microservices with Node.js

---

<b>1. INTRODUCTION</b>	<b>3</b>
<b>2. IBM CLOUD APP SERVICE: CREATE AN ACCOUNT</b>	<b>6</b>
<b>3. IBM CLOUD APP SERVICE: CREATE A NODE.JS MICROSERVICE ON IBM CLOUD</b>	<b>8</b>
<b>4. NODESERVER CLI: CREATE A NODE.JS MICROSERVICE</b>	<b>13</b>
<b>5. RUN THE EXPRESS.JS MICROSERVICE</b>	<b>15</b>
<b>6. CONCLUSION</b>	<b>28</b>
<b>7. RESOURCES</b>	<b>29</b>
<b>ACKNOWLEDGEMENTS AND DISCLAIMERS (V8)</b>	<b>31</b>



# Introduction

---

This lab walks you through the steps required to create, build and run a cloud native Express.js microservice in less than 5 minutes.

The microservice can either be built using either:

1. A web browser using the **IBM Cloud App Service**

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 your 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.

2. The command line using the **NodeServer CLI**

This provides a local command line utility to create the Node.js cloud native application. This creates applications that are “Cloud Ready”, including all the necessary configuration to create a DevOps pipeline to your project using Jenkins or the IBM Cloud Continuous Delivery Service, and for deployment to Cloud Foundry, Docker and Kubernetes technologies, such as those provided by IBM Cloud.

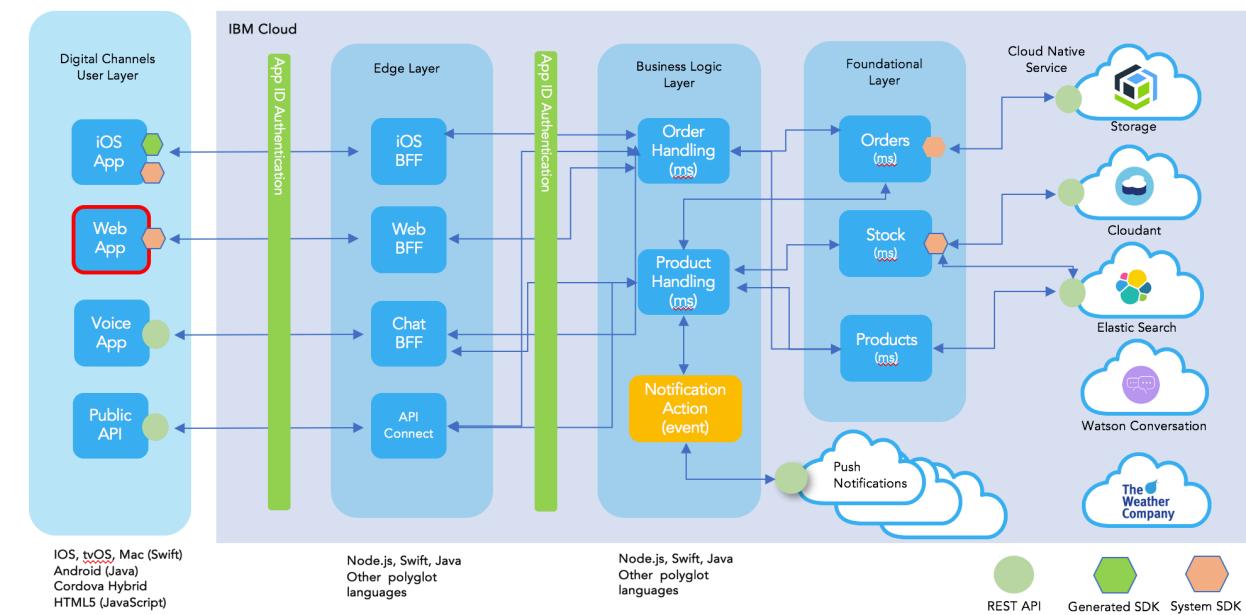
This lab allows you to use either approach. Note that using the **IBM Cloud App Service** requires you to have an account with IBM Cloud. You may use your existing account or create a new account (Section 2).

For this lab, we are going to focus on an example cloud native web application architecture. We are going to build one part of this architecture: a Foundational Layer Microservice that provides a REST API. For more information on a Reference Implementation for building a cloud native OmniChannel Application using a Microservices architecture refer to IBM Garage Method: <https://github.com/ibm-cloud-architecture/refarch-cloudnative>



## Example: Order Management Use Case

Example Omni Channel application service different use end points and requirements , using Cloud Native Micro Services architecture



At the conclusion of this lab, you will have completed the following steps:

1. Create a Express.js Microservice Application
2. Run the Express.js microservice, with full support for monitoring, request tracking, and liveness checking, and access it via the web browser.



# Setting Up

---

Before starting this lab, please do the following:

1. Go to <http://ibm.biz/startmylab>
2. Select **Create Highly Scalable Web Application Microservices with Node.js** lab from the dropdown and click Ok
  1. 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.
  2. This lab does not require you to have an IBM Cloud Platform account but it allows you to do some optional steps.
3. Clone the project for this lab:
  1. Click on the Launchpad (rocket logo) in the toolbar, search for “Terminal” and click on the icon.
  2. Clone the GitHub project containing the FoodTracker application we are going to extend:

```
git clone  
http://github.com/seabaylea/NodeMicroservices
```
  3. A copy of these instructions are available included in the project. You can open the instructions using:

```
open "NodeMicroservices/ NodeMicroservices.pdf"
```

## NEXT STEPS:

If you want to use the **IBM Cloud App Service**, proceed to Section 2 (Create an Account) or Section 3 (Create a Node.js Microservice on IBM Cloud)

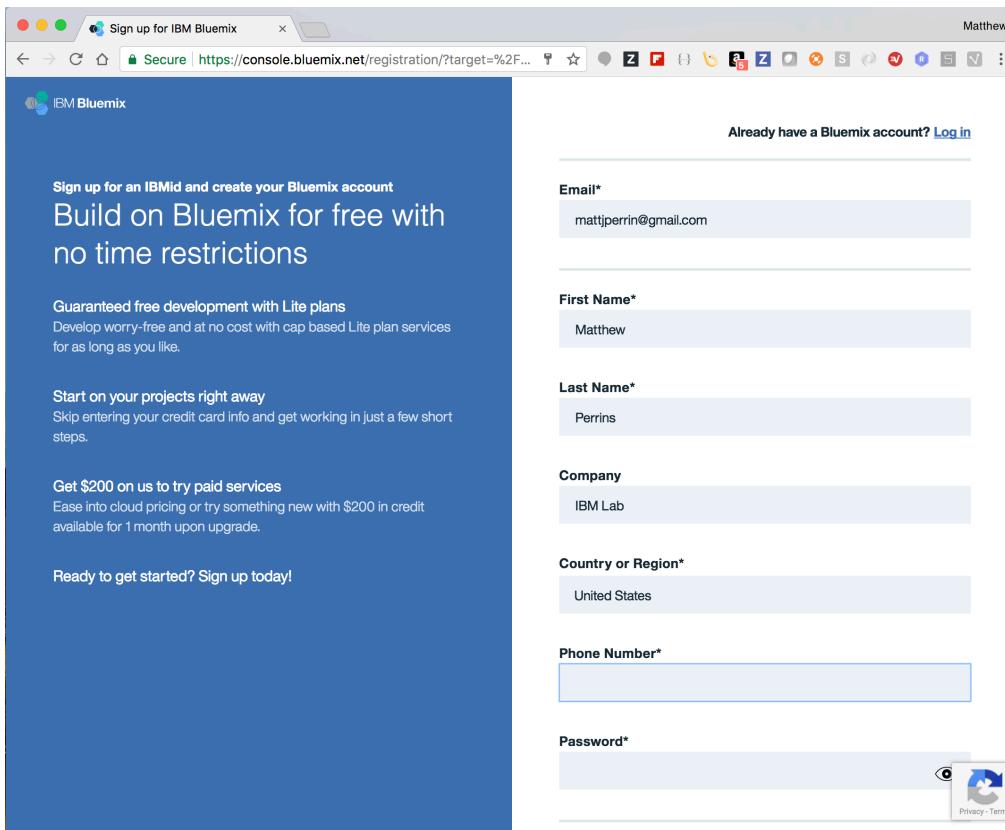
If you want to use the **NodeServer CLI**, proceed to section 4 (Create a Node.js Microservice)



## 2. IBM Cloud App Service: Create an Account

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

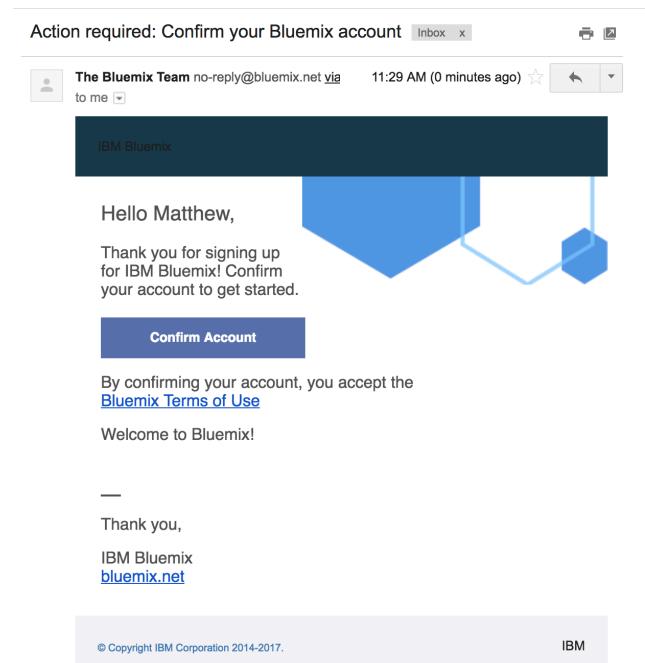
- To run these lab instructions, you will need to create a trial account on the IBM Cloud. You can do this free of charge and all the instructions contained in this lab do not incur any additional costs.
- Create an IBM Cloud Trial account by navigating to this link.  
<https://console.bluemix.net>,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



The screenshot shows a web browser window titled "Sign up for IBM Bluemix". The URL in the address bar is <https://console.bluemix.net/registration/?target=%2F...>. The page has a blue header with the IBM Bluemix logo. The main content area has a dark blue background with white text. It says "Sign up for an IBMid and create your Bluemix account" and "Build on Bluemix for free with no time restrictions". There are three sections of text: "Guaranteed free development with Lite plans", "Start on your projects right away", and "Get \$200 on us to try paid services". Below these is a button "Ready to get started? Sign up today!". On the right side, there is a registration form with fields for Email\*, First Name\*, Last Name\*, Company, Country or Region\*, Phone Number\*, and Password\*. There is also a "Log in" link for existing users. At the bottom right of the form area is a "Privacy - Terms" link.

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





- You can now log into IBM Cloud.
- Click on **Login** and enter your email and password
- 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.

A screenshot of the IBM Cloud Dashboard. The top navigation bar shows the IBM Cloud logo. Below it, the word "Dashboard" is displayed. On the left, there is a "RESOURCE GROUP" section with a dropdown menu set to "All Resources". To the right of this are three dropdown menus: "REGION" set to "US South", "CLOUD FOUNDRY ORG" set to "benedictf2018@gmail.com", and "CLOUD FOUNDRY SPACE" set to "dev". All three dropdown menus are highlighted with red boxes.



## 3. IBM Cloud App Service: Create a Node.js 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.

1. Access the IBM Cloud App Service
2. Create from the Blank Project Starter Kit

### First, what is a starter kit?

**Language + Framework + Architecture Pattern = Starter Kit**

- Automatically configure YAML/XML/JSON files
- Manage and configure dependencies, credentials & certificates
- Generate common “boilerplate code” for common architecture patterns

Saves hours, days, weeks as compared to “sample code” approach offered by competitors

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.

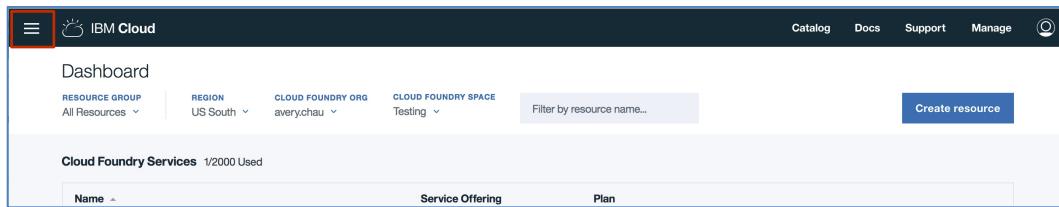
The platform’s code generation technology creates a starter application in the developer’s preferred language and framework, which can be tailored to their needs and use case. Any services that are required in support of the use case, are provisioned automatically.

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.

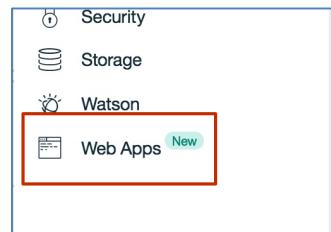


3. Start by logging in to IBM Cloud: <http://console.bluemix.net>

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.



4. 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.



The screenshot shows the IBM Cloud App Service landing page. At the top, there's a navigation bar with 'IBM Cloud' and links for Catalog, Docs, Support, Manage, and a profile icon. On the left, a sidebar has 'Overview', 'Starter Kits' (which is expanded), 'Developer Resources' (with Documentation and Learning Resources), and 'Projects'. The main content area has a title 'IBM Cloud App Service' with the subtitle 'Fast on-ramp for building cloud-native apps'. It features two main sections: 'Start from the Web' (using a Starter Kit configurator) and 'Start from CLI' (developing and testing with a local dev environment). Below these are 'Featured Resources' including 'How-to: Deploy to Kubernetes using the CLI', 'IBM Cloud Blog', and 'How-to: Enable existing projects with the CLI'.

5. 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.

6. Scroll down the list and click on the tile labelled **Create Project** to create a custom project.



7. Within the Create New Project view, enter a project name of NodeMicroservice. Once you've completed the form, click on the **Create Project** button.
8. Select a Language of **Node.js**, and click the “**Add file**” button to add a OpenAPI (aka Swagger) document. Select the “petstore.yaml” file from the NodeMicroservices directory.

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

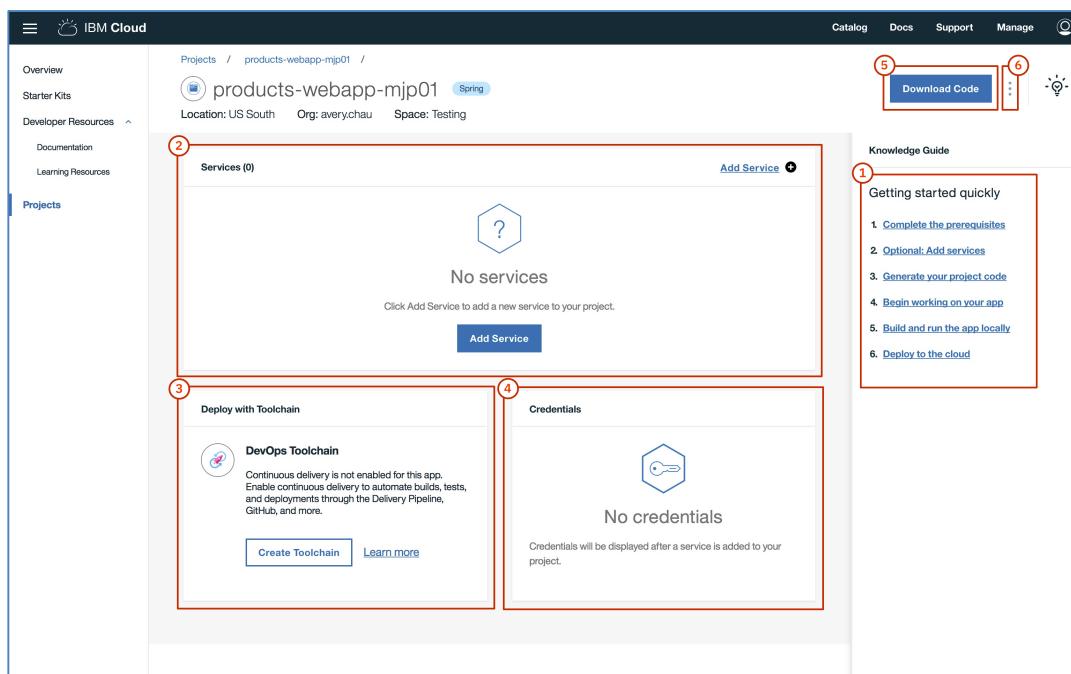


The screenshot shows the 'Create new project' page in the IBM Cloud interface. At the top, there's a navigation bar with 'IBM Cloud' and a sidebar icon. Below it, the path 'Overview / Starter Kits /' and the title 'Create new project'. The main area is titled 'Project Details' and contains fields for 'Enter a project name:' (with 'Expressjs Microservice UJRZS' entered), 'Select region to deploy in:' (set to 'US South'), 'Choose an organization:' (set to 'IBM\_Runtimes'), and 'Choose a space:' (set to 'Dev'). A 'Route' section follows, with 'Host' set to 'expressjs-microservice-ujrzs' and 'Domain' set to 'mybluemix.net'. Finally, a 'Language' section where 'Node.js' is selected.

9. Click “**Create Project**” to create the project.
10. The App Configuration/Details view is now displayed. This view is used to manage various aspects of an app:
  1. The right-hand side (1) offers a Getting Started Quickly guide.
  2. The main body has an area where you can create new and associate existing services to the app (2)



3. The bottom of the screen offers an area where you may configure a DevOps toolchain (3)
4. 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)
5. At the top of the screen, it is possible to download the code (5) to the app without any DevOps configuration.
6. Using the menu in the top right (6), you can also rename or delete it.



7. 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  
cd NodeMicroservice
```

Note: You may need to unzip the download. If you need to, use the following:

```
unzip -d NodeMicroservice NodeMicroservice.zip
```

To continue the lab, move on to Section 5 (Run the Node.js Microservice)

## 4. NodeServer CLI: Create a Node.js Microservice

---

In this section, you will use the NodeServer CLI to create an Express.js microservice application that is “Cloud Ready” and can be run locally or used to any cloud that supports any of CloudFoundry, Docker and Kubernetes, including IBM Cloud.

This includes installing the NodeServer CLI and using it to create a Node.js Microservice

1. 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.

2. Install the Yeoman utility by running the following command line:

```
sudo npm install -g yo
```

If prompted, enter a password of ‘password’

1. Install the NodeServer utility by running the following command line:

```
sudo npm install -g generator-nodeserver
```

To continue the lab, move on if prompted

This has now installed the NodeServer CLI utility which allows you to quickly create a cloud enabled Node.js application.

13. Create a directory with the name of your project

```
mkdir NodeMicroservice  
cd NodeMicroservice
```

2. Create your Node.js microservice

Run the following command on the command line:

```
yo nodeserver
```

This will prompt you to answer a number of questions. Provide the following answers:

```
[2018-03-14T16:33:08.932] [INFO] generator-nodeserver - Package info ::  
generator-nodeserver 1.0.11  
? Project name NodeMicroservice
```



```
? OpenAPI Document (None) /Users/user/NodeMicroservices/petstore.yaml  
? Add IBM Cloud Service Enablement? No
```

This creates a fully working skeleton Node.js project that provides monitoring, metrics and request tracking, which can then be extended with your application logic.

As we provided an OpenAPI Document (aka a Swagger definition), additional template code has been added to implement those API calls. The definition we provided is the “Petstore” sample, which provides the REST APIs for a simple petstore.

If you explore the generated assets you will see that there a number of configuration files generated for you, including:

1. Jenkinsfile A pipeline for Jenkins and IBM Cloud Private
2. .bluemix/\* A pipeline for IBM Cloud Continuous Delivery Service
  
3. Dockerfile A dockerfile to build a Docker image for the application
4. chart/\* A helm chart to enable deployment to Kubernetes
5. manifest.yml A configuration file for use with Cloud Foundry



## 5. Run the Node.js Microservice

---

In this section, you will enable the optional request tracking capability, and run the Express.js microservice locally, including viewing its monitoring and metrics capabilities.

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

- a. Open the file:

```
open -aTextEdit ./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. 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 an open source monitoring solution that is designed to integrate into Docker and Kubernetes deployments, and monitors requests across individual microservices.

```
docker run -d -p 9090:9090 -v  
/Users/user/nodemicroservices/prometheus.yml:/etc/prometheus/prometheus.yml  
prom/prometheus
```

This will start running the Prometheus monitoring service in a Docker container on port 9090. Prometheus is an open source monitoring solution that is designed to integrate into Docker and Kubernetes deployments to monitor applications.

3. Install the dependencies, and run the application:

```
npm install  
npm start
```



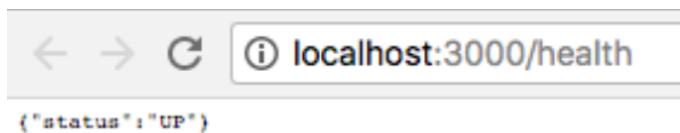
4. View the liveness endpoint:

To check that the application is running successfully, view the liveness endpoint:

Liveness endpoint: <http://localhost:3000/health>

The **liveness endpoint** provides a response that states whether the application is successfully running. This can be used by Cloud Foundry and Kubernetes to automatically restart the application if it fails or becomes unresponsive.

It should report the following:



A screenshot of a web browser window. The address bar shows the URL `localhost:3000/health`. Below the address bar, the browser's navigation controls (back, forward, refresh) are visible. The main content area of the browser displays the JSON response: `{"status": "UP"}`.



## 5. View the Swagger Explorer

---

As the Node.js microservice was built using a Open API (aka. Swagger) specification, skeleton code to implement the defined REST APIs have been added to the application, along with Swagger UI, which provides an Explorer.

The Explorer acts as both documentation for the provided REST API, and as a way of testing out the API in the browser.

### 1. View the Swagger Explorer

The Swagger Explorer us hosted by the Node.js microservice on /explorer and reachable on the following URL:

Swagger Explorer: <http://localhost:3000/explorer>

Once you open the explore, you will be presented with the Petstore API:

The screenshot shows the Swagger Petstore API Explorer. At the top, there's a green header bar with the 'swagger' logo, the URL 'http://localhost:3000/swagger/api', and a 'Explore' button. Below the header, the title 'Swagger Petstore' is displayed, followed by a brief description: 'A sample API that uses a petstore as an example to demonstrate features in the swagger-2.0 specification'. It credits 'Created by Swagger API Team' and links to 'MIT'. The main content area shows a 'default' API group with four operations listed: 'GET /pets' (blue), 'POST /pets' (green), 'DELETE /pets/{id}' (red), and 'GET /pets/{id}' (blue). To the right of these operations are three buttons: 'Show/Hide', 'List Operations', and 'Expand Operations'. At the bottom left, it says '[ BASE URL: /api , API VERSION: 1.0.0 ]'.

### 2. Test an API in the Explorer

As well as showing you the APIs that have been defined, it is possible to use those APIs directly through the Explorer.

#### a. Select the GET / pets API

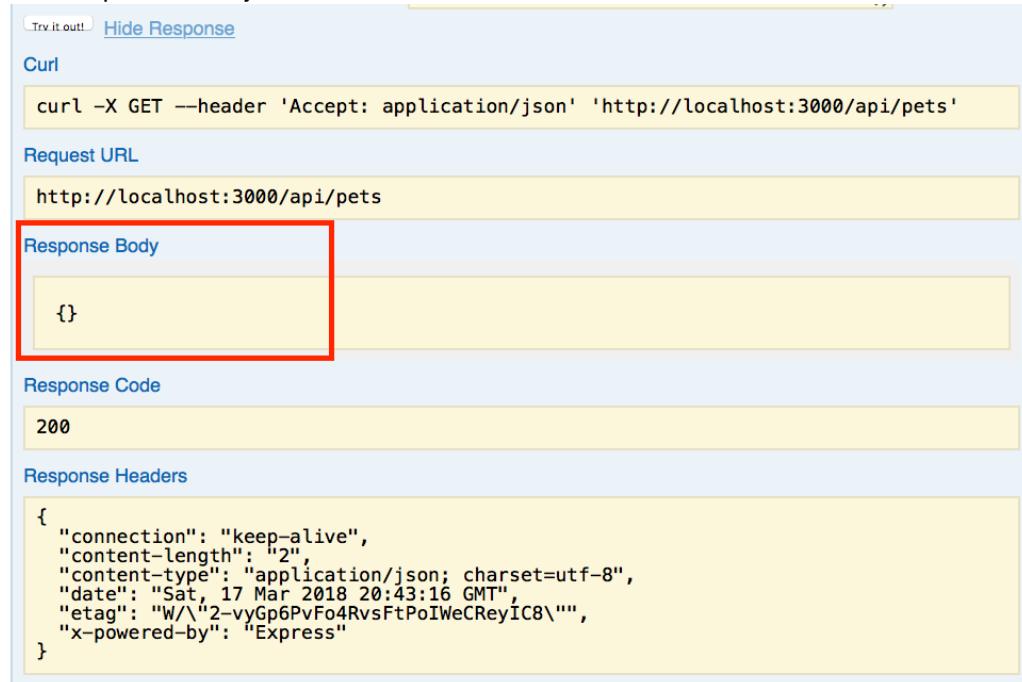
Selecting one of the APIs expands it to show more information, including any additional information on the API and the types of data it accepts or responds with.

#### b. Click the “Try it out!” button

The explorer also provides boxes to allow you to set up sample data and then a “Try it out!” button to make a request with that data and see the response. As



only the skeleton code is currently implemented, an empty result is returned for the Response Body:



The screenshot shows the Swagger Explorer interface with the following details:

- Curl:** curl -X GET --header 'Accept: application/json' 'http://localhost:3000/api/pets'
- Request URL:** http://localhost:3000/api/pets
- Response Body:** {} (This section is highlighted with a red box.)
- Response Code:** 200
- Response Headers:**

```
{  
  "connection": "keep-alive",  
  "content-length": "2",  
  "content-type": "application/json; charset=utf-8",  
  "date": "Sat, 17 Mar 2018 20:43:16 GMT",  
  "etag": "W/\"2-vyGp6PvFo4RvsFtPoIWeCReyIC8\"",  
  "x-powered-by": "Express"  
}
```

The Swagger Explorer provides an invaluable tool during development, making it easy to incrementally develop and manually test the APIs. It also acts as documentation and testing for developers who need to build their own microservices that use your microservice.



## 6. View the Node.js Monitoring Dashboard

As part of building the Node.js microservice with either the IBM Cloud App Console or the Nodeserver CLI, an in-built monitoring dashboard has been added to the skeleton project.

This provides resource usage metrics for CPU and memory, as well as performance metrics for a large number of capabilities, including inbound and outbound HTTP and WebSocket requests, and outbound requests made of databases and messaging systems.

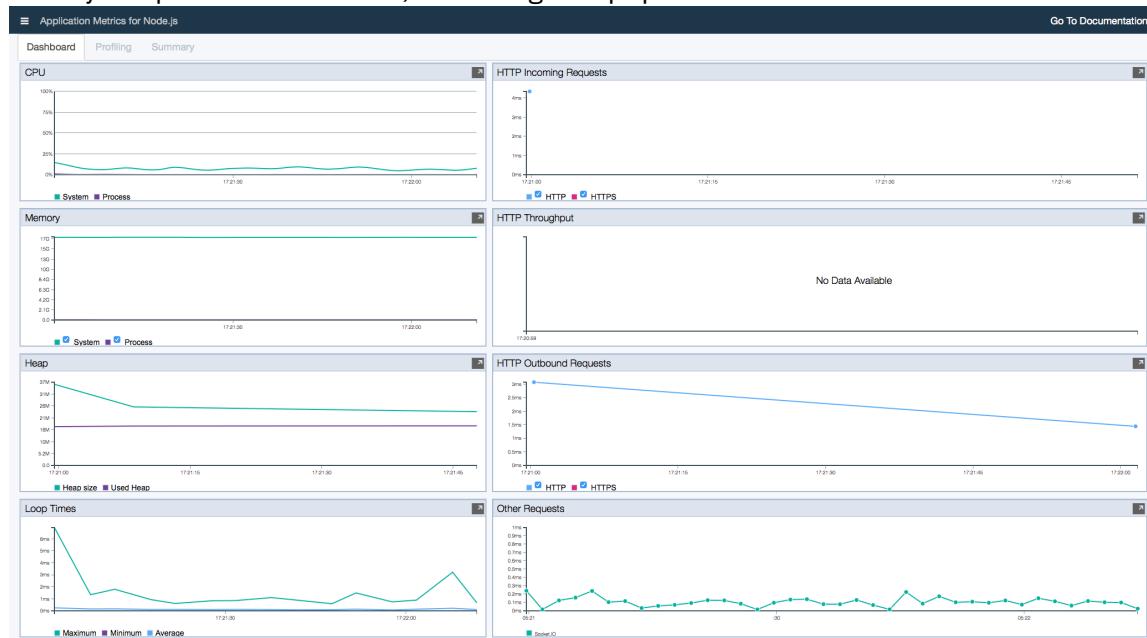
The dashboard also provide the ability to request additional data, in the form of function level performance profiling, heap memory usage dumps, and a “node report” that provides a summary of the state of the running application.

### 1. View the Monitoring Dashboard

The monitoring dashboard is hosted by the Node.js microservice on /appmetrics-dash and reachable on the following URL

Monitoring Dashboard: <http://localhost:3000/appmetrics-dash>

Once you open the dashboard, it will begin to populate with data.



### 2. Make some HTTP requests of the Node.js microservice

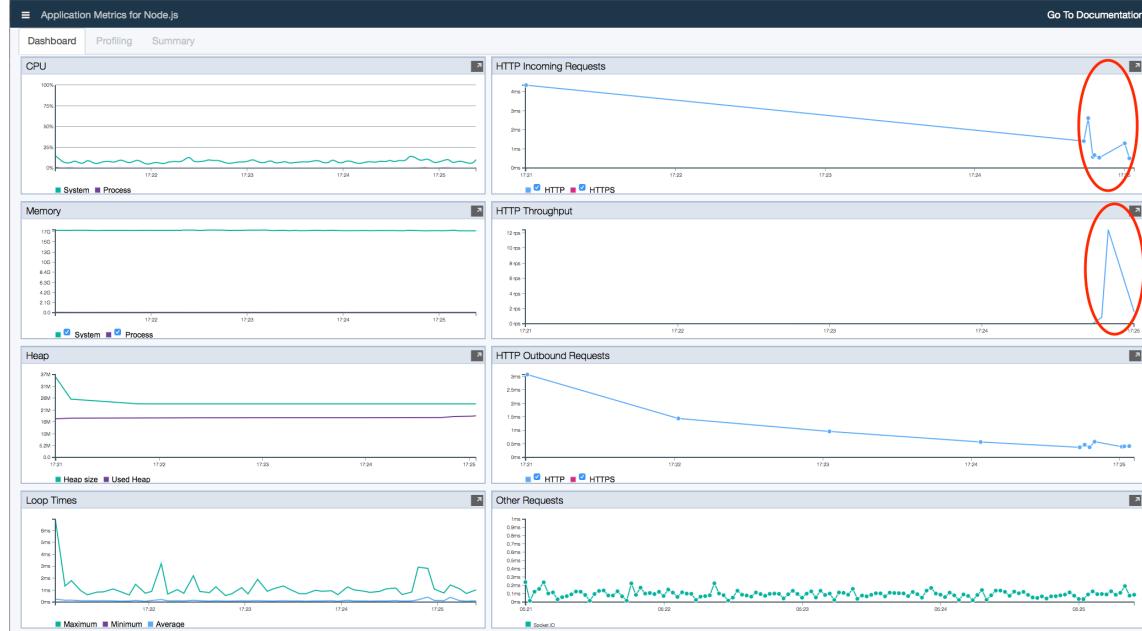
Open the the liveness endpoint in a new browser window:

Liveness endpoint: <http://localhost:3000/health>



And drive some load by reloading the page using the ⌘+R key shortcut.

These requests will appear in the HTTP Incoming Requests and HTTP Throughput views:

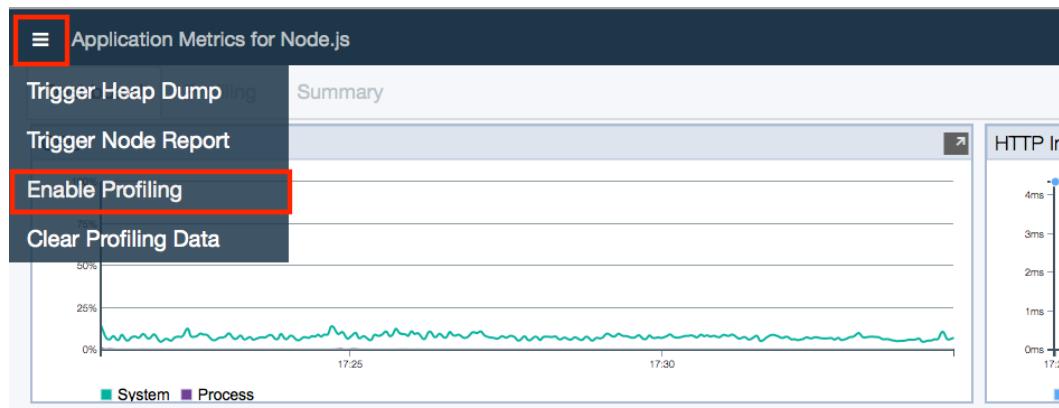


### 3. Enable and view Performance Profiling

The monitoring dashboard provides the ability for you to dynamically enable performance profiling, and to view the data using flame graphs.

#### a. Enable performance profiling

Use the pulldown menu in the top left hand corner to select Enable Profiling:



This enables the profiling, and enables the Profiling tab on the dashboard to allow you to view the collected data.

#### b. View the profiling data

Click on the Profiling tab to see a Flame Graph of the functions using CPU time in



the application:



The higher the “flame” the deeper the call stack of functions is. You can select a flame to see its corresponding stack trace in the right hand window.

The wider the cell in the flame, the more time is spent in that function.

The monitoring dashboard also allows you to request heapdumps and node reports, which are written to the local filesystem.



## 7. View Cloud Metrics in Prometheus

---

As well as building in the monitoring dashboard, which provides monitoring of the specific instance of the Node.js microservice, hosted by the microservice itself, support for Prometheus monitoring has been built in.

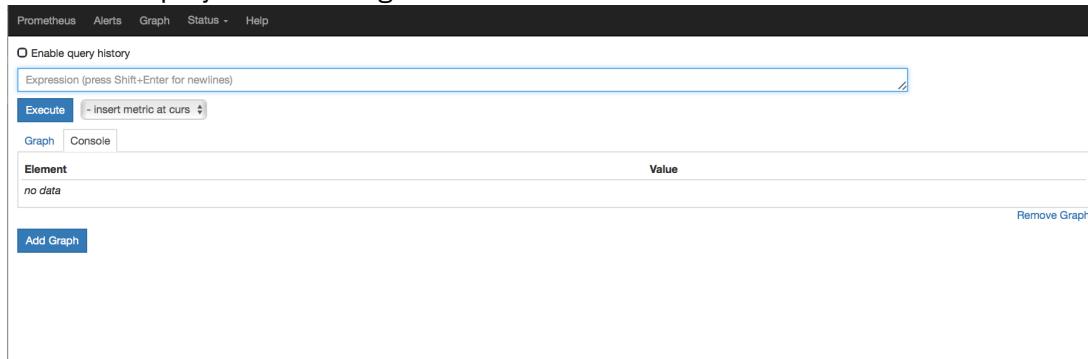
Prometheus is an open source cloud based monitoring solution that collects metrics from running applications and stores them for use in visualization and alerting. The data can be visualized using open source dashboarding technologies such as Graphana, but it also provides its own (simplistic) built-in option.

1. View the Prometheus Dashboard

The Prometheus dashboard is reachable on the following URL

Prometheus Dashboard: <http://localhost:9090/>

This will display the following:



2. See the status of running services

Prometheus will report the status of all of the services that it is monitoring, reporting whether they are currently UP or DOWN.



- Select the Status pull down menu and choose Targets

The screenshot shows the Prometheus UI with the 'Status' dropdown menu open. The 'Targets' option is highlighted with a red box. Other options visible in the menu include 'Runtime & Build Information', 'Command-Line Flags', 'Configuration', 'Rules', 'Service Discovery', and 'Graph' (which is currently selected).

This reports a very limited set because we are only collecting metrics for the Node.js microservice, and the Prometheus service itself.

The screenshot shows the 'Targets' page in the Prometheus UI. It lists two service instances:

- Node.js Microservice (1/1 up)**: Endpoint: http://docker.for.mac.localhost:3000/metrics, State: UP, Labels: instance="docker.for.mac.localhost:3000", Last Scrape: 5.693s ago, Error: None.
- prometheus (1/1 up)**: Endpoint: http://localhost:9090/metrics, State: UP, Labels: instance="localhost:9090", Last Scrape: 495ms ago, Error: None.

You can also select the “Endpoint” link to see the latest raw data reported from that service

- View the Node.js microservices metrics

Prometheus provides the ability to run queries and build graphs using the collected data. We can use this to look at information such as the CPU usage over time, or the responsiveness of HTTP URLs

- Select the Prometheus tab

The screenshot shows the Prometheus UI with the 'Prometheus' tab selected, indicated by a red box around the tab label.



- b. From the “insert metric” pull down, select “process\_cpu\_used\_ratio”:

The screenshot shows the Prometheus web interface. At the top, there are tabs for Prometheus, Alerts, Graph, Status, and Help. Below the tabs, there is a checkbox labeled "Enable query history". A text input field is present with placeholder text "Expression (press Shift+Enter for newlines)". Below the input field are two buttons: "Execute" (highlighted with a red box) and "- insert metric at curs". At the bottom of the interface are two tabs: "Graph" and "Console".

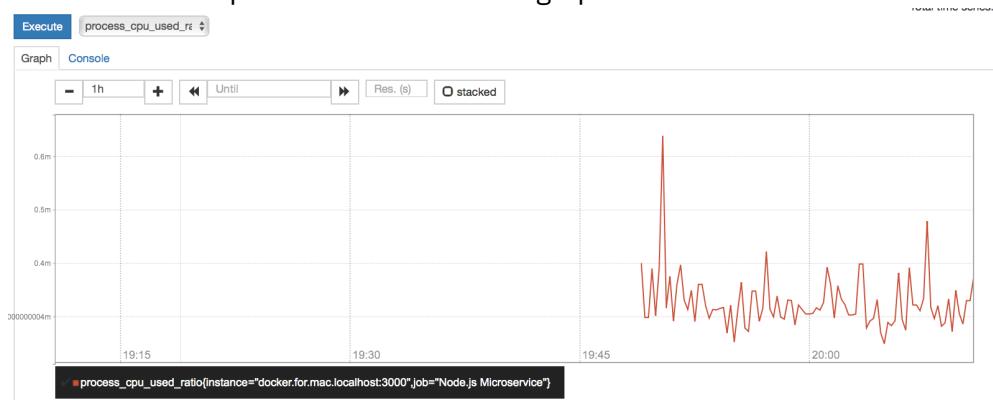
The “process\_cpu\_used\_ratio” represents the percentage of the machines CPU that is being used by each of the processes.

- c. Click “Execute” to see the data

This will display the data in table form:

Element	Value
process_cpu_used_ratio[instance="docker.for.mac.localhost:3000",job="Node.js Microservice"]	0.000272496

Click on the “Graph” tab to see this as a graph over time:



Prometheus has a significantly larger set of capabilities, including setting up alerts. Below is an example of the monitoring that can be built using Prometheus combined with a Grafana custom dashboard;:





## 7. View Cloud 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.

Whilst the real value of OpenTracing and Zipkin is seen when using multiple microservices, it is important to introduce it into microservices from the beginning where possible. For Node.js it can be added with one line of code, but for other languages and frameworks it needs to be developed into the application.

### 1. View the Zipkin request tracking data

Zipkin is automatically collecting request tracking “spans” from the Node.js microservice, and the data can be viewed at any time.

#### a. Visit the following URL:

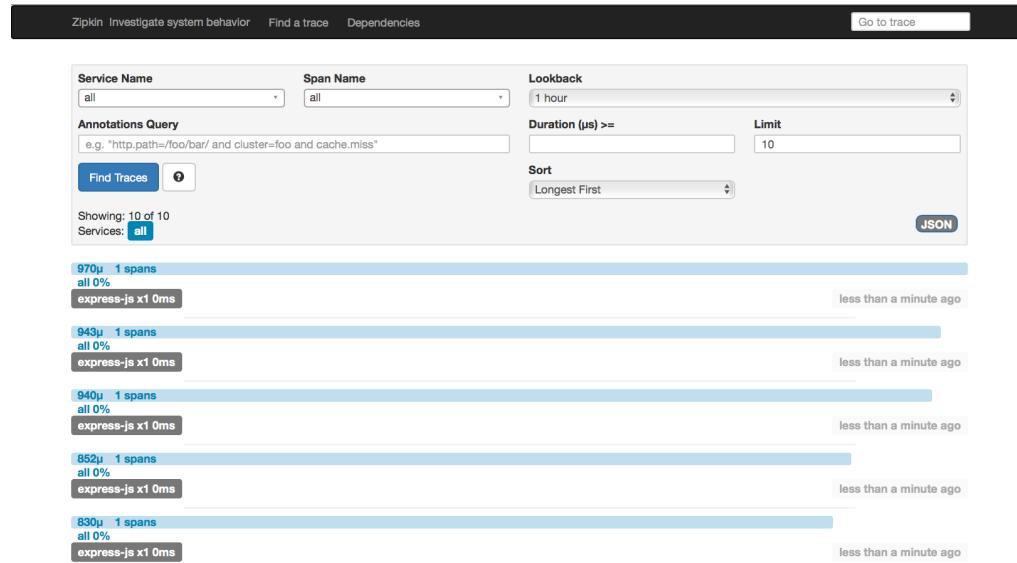
Zipkin Monitoring: <http://localhost:9411>

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

The screenshot shows the Zipkin web interface. At the top, there is a dark header with the text "Zipkin" and "Investigate system behavior". Below the header are three buttons: "Find a trace", "Dependencies", and "Go to trace". The main area contains several input fields and dropdown menus. On the left, there are "Service Name" and "Span Name" dropdowns, both set to "all". To the right of these are "Lookback" (set to "1 hour") and "Duration (μs) >= [empty field]" fields. Below these are "Annotations Query" (containing "e.g. \"http.path=/foo/bar/ and cluster=foo and cache.miss\"") and "Limit" (set to "10"). On the far right is a "Sort" dropdown set to "Longest First". At the bottom of this section is a large blue button labeled "Find Traces" with a red box drawn around it. To the right of the "Find Traces" button is a small question mark icon. A light blue footer bar at the bottom of the form area contains the text "Please select the criteria for your trace lookup."

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





This shows the duration of each request

If there were other microservices involved, you would see the time spent in each microservice that contributed to the overall elapsed time.

You will be able to see Zipkin being used across multiple microservices in [DevZone Quick Lab: Create and Deploy Enterprise-Scale, Multi-Language Microservices](#)



## 8. Conclusion

---

**Congratulations!** You have completed the lab.

You have successfully built and run a Express.js Microservice that is cloud-ready and can be deployed to any cloud supporting Cloud Foundry, Docker or Kubernetes , including IBM Cloud and IBM Cloud Private!



## 9. Resources

---

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

