Redbooks
ibm.com/redbooks

Hybrid Cloud Data and API Integration
Integrate Your Enterprise and Cloud with Bluemix Integration Services

Srinivas Cheemalapati
Yi-an Chang
Shahir Daya
Matthieu Debeaux
Odilon Magroski Goulart
Vasfi Gucer
Rahul Gupta
Shamim Hossain
David Kwock
Jordan T Moore
David N Nguyen
Bobby Woolf

Analytics

Cloud

IBM

Redbooks

International Technical Support Organization

Hybrid Cloud Data and API Integration

November 2015

SG24-8277-00

Note: Before using this information and the product it supports, read the information in “Notices” on page xvii.

First Edition (November 2015)

This edition applies to IBM Bluemix Version 1.0.

© Copyright International Business Machines Corporation 2015. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

© Copyright IBM Corp. 2015. All rights reserved.

Figures

© Copyright IBM Corp. 2015. All rights reserved.

Tables

© Copyright IBM Corp. 2015. All rights reserved.

Examples

© Copyright IBM Corp. 2015. All rights reserved.

Notices

This information was developed for products and services offered in the U.S.A.

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

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

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

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

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

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

© Copyright IBM Corp. 2015. All rights reserved.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®
Bluemix™
Cast Iron®
CICS®
Cloudant®
dashDB™
DataPower®
DB2®
developerWorks®
Global Technology Services®
IBM®
IBM UrbanCode™
IBM Watson™
IBM z™
IBM z Systems™
IMS™
Informix®
Language Environment®
POWER®
Power Systems™
POWER8®
PureApplication®
QRadar®
Redbooks®
Redpaper™
Redbooks (logo) ®
Tivoli®
WebSphere®
z Systems™
z/OS®

The following terms are trademarks of other companies:

Netezza, and N logo are trademarks or registered trademarks of IBM International Group B.V., an IBM Company.

SoftLayer, and SoftLayer device are trademarks or registered trademarks of SoftLayer, Inc., an IBM Company.

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

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

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

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

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

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

Other company, product, or service names may be trademarks or service marks of others.

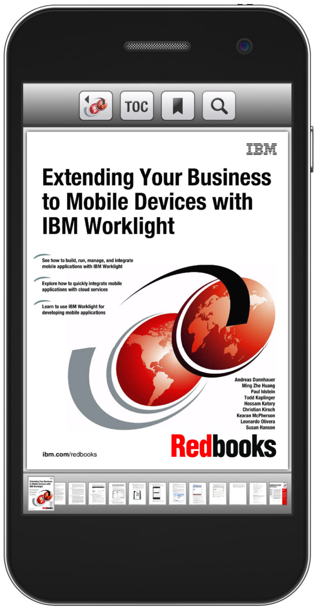
IBM REDBOOKS PROMOTIONS

Find and read thousands of IBM Redbooks publications

- ► Search, bookmark, save and organize favorites
- ► Get up-to-the-minute Redbooks news and announcements
- ► Link to the latest Redbooks blogs and videos

Get the latest version of the Redbooks Mobile App

iOS **Download Now** Android



IBM

Extending Your Business to Mobile Devices with IBM Worklight

See how to build, run, manage, and integrate mobile applications with IBM Worklight

Explore how to quickly integrate mobile applications with cloud services

Learn to use IBM Worklight for developing mobile applications

Andreas Dannhauer
Ming Zhe Huang
Paul Idstein
Todd Kaplinger
Hossam Katory
Christian Kirsch
Kearan McPherson
Leonardo Olivera
Susan Hanson

ibm.com/redbooks

Redbooks

Promote your business in an IBM Redbooks publication

Place a Sponsorship Promotion in an IBM® Redbooks® publication, featuring your business or solution with a link to your web site.

Qualified IBM Business Partners may place a full page promotion in the most popular Redbooks publications. Imagine the power of being seen by users who download millions of Redbooks publications each year!

ibm.com/Redbooks
About Redbooks → Business Partner Programs

THIS PAGE INTENTIONALLY LEFT BLANK

Preface

IBM® Hybrid Integration Services is a set of hybrid cloud capabilities in IBM Bluemix™ that allows businesses to innovate rapidly while, at the same time, providing IT control and visibility. It allows customers to quickly and easily build and operate systems that mix data and application programming interfaces (APIs) from a wide variety of sources, whether they reside on-premises or in the cloud.

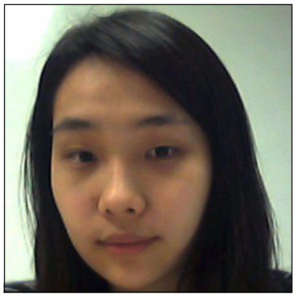
In many cases, you want to expose your IT assets from your private cloud as APIs and at the same time have best overall manageability and control of who uses your assets and how.

Bluemix provides a set of services such as Secure Gateway, API Management, Connect and Compose, DataWorks, and API Catalog, which enable Hybrid Cloud Integration capabilities. This IBM Redbooks® publication provides preferred practices around developing cloud solutions using these Hybrid Integration Services that help you maintain data consistency, manageability, and security for critical transactions.

Authors

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.

Srinivas Cheemalapati is a Cloud Advisor and Certified IT Architect in the IBM Cloud Unit. He has 27 years of experience that spans cloud computing, developing first-of-a-kind and complex IT architectures and solutions, Intel x-86 server development, wireless communications, and embedded technologies. He holds multiple patents and has a Master's degree in Computer Science from North Carolina State University. He is currently working on Serious Gaming, IoT, Bluemix PaaS, Software Defined Networking, Cloud Orchestration, and Network Function Virtualization areas.



Yi-an Chang is an IBM Software Engineer in the IBM Systems Middleware division. She has over 4 years of experience in application development and is currently focused on development and integration of IBM strategic cloud services, including several Hybrid Integration Services exposed via the IBM Bluemix platform described in this publication. Joy is interested in all aspects of Web Application Development, and is working on building full stack experience in the area.

© Copyright IBM Corp. 2015. All rights reserved.

Shahir Daya is an IBM Executive Architect in the GBS Global Cloud Center of Competence. He is IBM Senior Certified Architect and an Open Group Distinguished Chief/Lead IT Architect. Shahir has over 20 years at IBM with the last 15 focused on application architecture assignments. He has experience with complex high volume transactional web applications and systems integration. Shahir has led teams of practitioners to help IBM and its customers with application architecture for several years. His industry experience includes retail, banking, financial services, public sector, and telecommunications. Shahir is currently focused on Cloud application development services and in particular platform-as-a-service (PaaS) such as IBM Bluemix, containerization technology such as Docker, design, and development of Systems of Engagement (SOE), and Cloud-based DevOps including IBM Bluemix DevOps Services.

Matthieu Debeaux is an IBM Certified IT Specialist with IBM Cloud in France. He has 5 years of experience working in cloud computing, virtualization, and IT process management both on sales and delivery projects. Matthieu is currently focused on Cloud technologies such as OpenStack and Docker, and DevOps solutions, in particular IBM UrbanCode™ Deploy and Release. He holds a Master's degree in Computer Science from Grenoble Institute of Technology.

Odilon Magroski Goulart is a Master IT Architect (TSA) for Infrastructure Services at IBM Strategic Outsourcing (GTS) in Brazil and has been with IBM since 2002. Before he joined the Infrastructure Services, Odilon served in IBM as an IT Specialist on SAP Systems when he became certified by SAP, Microsoft, IBM DB2® and other software and hardware technologies. Odilon has also expertise in Cloud, several operating systems, databases systems, networking, and development. He authored an IBM Redbooks publication in 2006, 2013, and 2014 and became an IBM Redbooks Thought Leader in 2013. Odilon Goulart has a Bachelor's degree in IT from State University of Campinas.

Vasfi Gucer is an IBM Redbooks Project Leader with the IBM International Technical Support Organization. He has more than 18 years of experience in the areas of systems management, networking hardware, and software. He writes extensively and teaches IBM classes worldwide about IBM products. His focus has been on cloud computing for the last 3 years. Vasfi is also an IBM Certified Senior IT Specialist, Project Management Professional (PMP), IT Infrastructure Library (ITIL) V2 Manager, and ITIL V3 Expert.

Rahul Gupta is an Advisory IT Architect with IBM Global Technology Services® (GTS) in the US. He is a Certified SOA Architect with 9 years of professional experience in IBM messaging technologies. At his current assignment, he works as a middleware consultant for various clients in North America. His core experiences are in lab testing, performance tuning, and Level 3 development for IBM Integration Bus and WebSphere® MQ. Rahul has been a technical speaker for messaging-related topics at various WebSphere conferences. He is a recognized inventor by the IBM innovation community.

Shamim Hossain is an IBM Certified Cloud Solution Advisor and Cloud Solution Architect. He leads a cloud consultancy laboratory in IBM Australia to develop born-on-the-cloud applications using agile methodologies and design thinking. He holds a Master of Telecommunications Engineering from the University of Melbourne and a Bachelor of Computer System Engineering (first class honors) from Monash University. His expertise and interests include different areas of cloud computing, mobile computing, optical fibre communications, broadband, Internet engineering and the Internet of Things (IoT). He is a published author. He co-authored a book entitled "*Cloud Computing Service and Deployment Models: Layers and Management*" by IGI Global.

David Kwock is a Service Management Solution Architect in the US. He has 12 years of experience in the IT Service Management field, and has worked at IBM for 5 years. His areas of expertise include Service Management, service-oriented architecture, Cloud Computing, and IBM Dynamic Infrastructure. David has written extensively about IT Service Management, service-oriented architecture, and Cloud Computing, and has contributed to an IBM Redpaper™ publication about IBM Tivoli® Provisioning Manager for OS deployment in a retail environment and to an IBM Redbooks publication about integrating Tivoli solutions. He has also presented on these topics at several events, including IBM Executive Summits and Gartner conferences.

Jordan T Moore is a Software Engineer with IBM Systems Middleware as a member of the Client Job Rotation Program. He and his team are responsible for working to help clients leverage the value of IBM solutions as the clients work across our technical client-facing roles within enablement, SWAT, technical support, and services job roles while focusing on key areas of the IBM portfolio.

David N Nguyen is a Software Engineer in IBM Systems in the US. He has 15 years of experience in systems management, virtualization, and cloud software development. David is currently focused on bringing hybrid cloud solutions to life for enterprise customers building applications connecting public clouds to systems of record on IBM Power Systems™ and z Systems™. He has presented on these topics at events including IBM Edge and IBM Systems Technical Universities. David previously worked on PowerVC and IBM PureApplication® Server for Power Systems. He holds a Master’s of Science degree from the University of Minnesota.

Bobby Woolf is a Bluemix Technical Enablement Specialist for IBM Cloud Lab Services in Research Triangle Park, North Carolina, US. He specializes in developing best practices for Enterprise Bluemix and the integration of Bluemix applications with an enterprise’s existing IT assets. Before that, he helped enterprises successfully adopt IBM PureApplication System and flagship IBM WebSphere products such as WebSphere Application Server, WebSphere Process Server, WebSphere Enterprise Service Bus, and WebSphere Operational Decision Management. He is an authority on architectures like Java Enterprise Edition, service-oriented architecture (SOA), enterprise service bus (ESB), event-driven architecture (EDA), and complex event processing (CEP), and cloud. He published numerous articles on IBM developerWorks® and is the author or co-author of the books *Enterprise Integration Patterns*, *Exploring IBM SOA Technology and Practice*, *The Design Patterns Smalltalk Companion*.

Thanks to the following people for their contributions to this project:

Erick Griffin, Matthew Hamann, Charlie Lawrence, Jean-Claude Mamou, Sajan Sankaran, Donald R Woods
IBM US

Rob Phippen, Naomi Scott
IBM UK

Cyril Armand, Sebastien Chabrolles
IBM France

Now you can become a published author, too!

Here’s an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks
- Send your comments in an email to:

  redbooks@us.ibm.com
- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks
- Follow us on Twitter:

  https://twitter.com/ibmredbooks
- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806
- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm
- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

Part 1

Introduction to hybrid cloud concepts and products

In this part, we introduce the hybrid cloud concepts within the context of several use cases and also describe some of the IBM products and services that you can use to implement a hybrid cloud environment.

© Copyright IBM Corp. 2015. All rights reserved.

1

Introduction to hybrid clouds

In this chapter, we introduce the business challenges that drive enterprises to adopt hybrid cloud solutions. In addition, we define hybrid cloud and the different aspects of hybrid cloud solutions. Finally, in this chapter we look at business real-world use cases to showcase the business value of hybrid cloud solutions. In addition, we set the stage for how hybrid cloud solutions help drive business value to enterprises.

This chapter includes the following key sections:

- 1.1, “Business challenges and motivations for hybrid clouds” on page 4
- 1.2, “What is hybrid cloud?” on page 7
- 1.3, “Hybrid cloud customer scenarios and use cases” on page 9

© Copyright IBM Corp. 2015. All rights reserved.

1.1 Business challenges and motivations for hybrid clouds

In this section, we cover the business challenges and drivers that motivate companies to adopt hybrid cloud solutions. In addition, we look at how customer expectations for a personalized experience when interacting with companies has driven systems of engagement and hybrid cloud solutions.

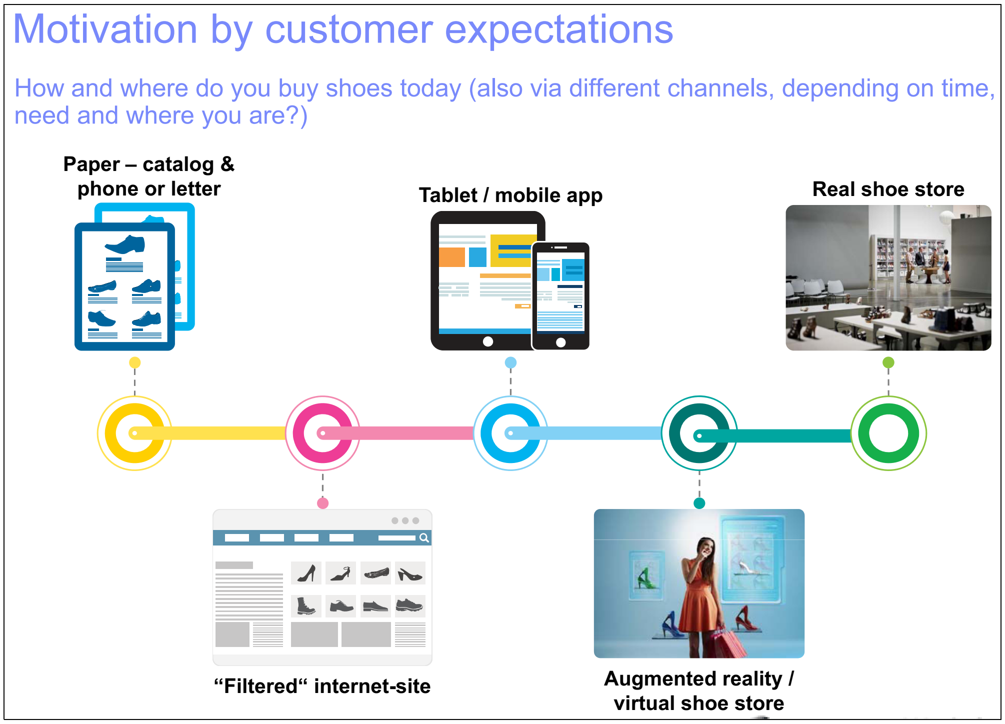
1.1.1 Business challenges and customer expectations

In today's market, the following business challenges motivate companies to look at hybrid cloud solutions:

- Lowering the total cost of ownership (TCO)
- Speed to market
- Lower level of effort for operations
- Reclaim control of projects lost to shadow IT
- Easy on-ramp to delivering new systems

In addition to these key business drivers, which motivate companies to adopt hybrid cloud solutions, many companies are motivated by new customer expectations. Many customers expect to be treated as individuals when interacting with a company. The need for a personalized experience has driven the need for the collection of both more historical and real-time information, as well as advanced analytics. Furthermore, customers now demand an easier interaction with companies with the ability to serve themselves. Finally, customers expect companies to support interactions over their smartphones with consistent experiences across all channels. This means that company systems must identify customers in context and coordinate the session state.

An example of this new customer expectation is the interaction with a retail store (see Figure 1-1 on page 5). Today, customers expect to be able to view and purchase shoe products through a paper catalog, phone, letter, Internet site, tablet, mobile application, virtual store, and real brick-and-mortar location. In addition, customers expect the experience to be personalized and in context with their last interaction with the company. Thus, customers expect the store to suggest products based on their previous purchases and browsing history.



Motivation by customer expectations

How and where do you buy shoes today (also via different channels, depending on time, need and where you are?)

Paper – catalog & phone or letter

Tablet / mobile app

Real shoe store

“Filtered“ internet-site

Augmented reality / virtual shoe store

Figure 1-1   Motivation by customer expectations

1.1.2 Lowering the total cost of ownership

Hybrid cloud solutions enable businesses to lower total cost of ownership by driving down the time to deploy and maintain a hybrid cloud solution. Although consulting services might be used to build a hybrid cloud solution, the total amount of consulting time that is needed for hybrid cloud solutions is significantly less than traditional solutions that require a large application build versus the reuse of existing microservices.

When evaluating the total cost of ownership for a solution, many companies evaluate the following factors:

- User-dependent basic charges
- Storage capacity for the developer team
- Inbound data transfer
- Outbound data transfer
- Provider internal data transfer
- Extra user data storage capacity
- Extra user document storage capacity
- Queries to the application programming interface
- Sent emails
- Database
- Secured logins
- Connections with other providers’ applications

In addition, accurate total cost of ownership assessments consider the mix of cloud services that the application uses. Furthermore, it is important to understand the typical load on the application and the possible variations on load.

1.1.3 Speed to market and on-ramp to deliver new applications

Before hybrid cloud solutions, large applications were the standard and tended to be difficult to maintain, modify, and become productive quickly. These large applications tend to create long development cycles, which increases the time or speed to market. In addition, large applications make the build, test, and deploy process occur for every small change because the application is a single unit.

To overcome the large application approach in the hybrid cloud world, application design focuses on partitioning the application into small *microservices* (for example, by stitching different services shown in blue in Figure 1-2). This approach allows each microservice to work independently as well as part of a large application.

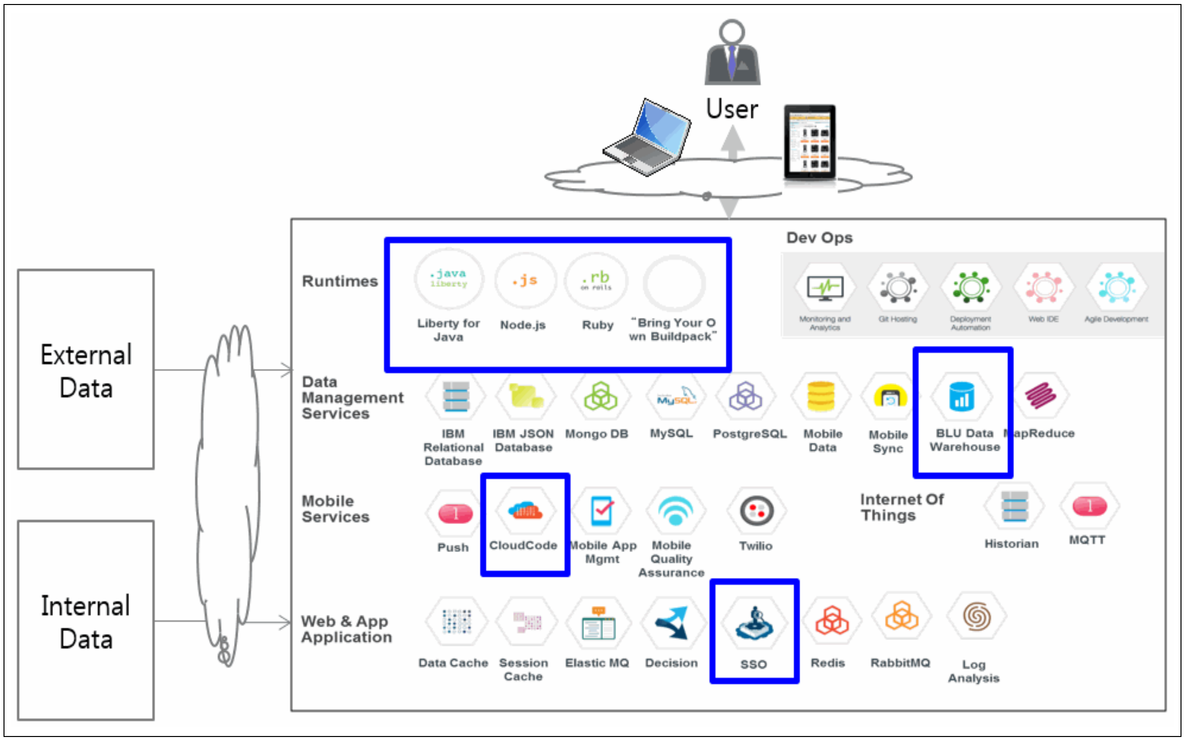


Figure 1-2 Microservice application design allows for quick time to market versus traditional large application design

This microservices approach immediately benefits business owners who get to see quicker times to market for applications and rapid response to customer and market needs.

More information about the microservices approach: For more information about the microservices approach, refer to *Microservices from Theory to Practice: Creating Applications in IBM Bluemix Using the Microservices Approach*, SG24-8275.

1.1.4 Lower level of effort for operations

In a traditional IT operational model, 80% of effort is focused on maintenance of existing systems. This leaves only 20% of effort for innovation. This operating model leads to slow response to the business needs. In a hybrid cloud operating model, day-to-day maintenance becomes a commodity that is automated as part of the hybrid cloud management platform. This drives operational costs down for IT and allows IT to focus on operating more like a business that manages vendor selection, packaging, pricing, delivery, and billing.

In order for IT to provide these services efficiently to the business, it must provide a self-service interface to order services and solutions. This need drives a new emphasis on packaging, cost, and IT supply chain management, which provides a continuous delivery model across both internal and external providers.

This operating model for hybrid cloud also puts a new focus on IT organizations' ability to offer choice to the user that unifies the operations of a solution, independent of how it is sourced. This allows the IT organization to act as a broker of services that can negotiate for the best services and price. In addition, it allows the expansion of the services and solutions offered to the user to expand quickly as business needs demand because IT no longer does the maintenance of the individual services and solutions. *In this operating model, IT only acts as the broker of best-of-breed services and solutions at the best possible price*.

1.1.5 Reclaim control of projects lost to shadow IT

The concept of *shadow IT* or IT resources procured outside the control of IT, opens a company to many challenges. This includes meeting data protection, privacy, audit, and compliance requirements. In the past, IT has dealt with shadow IT by trying to shut it down. However, this has not worked and now many IT departments are taking the approach of becoming a hybrid cloud service broker to take control of projects that have used shadow IT in the past. In order to successfully convert projects to a hybrid cloud solution, IT must provide a robust catalog of services at a lower cost and higher quality than a business unit can get from shadow IT. Also, the IT organization offering the services itself needs to act in an agile way, so responding fast to requests for change. By providing this robust catalog of services, many projects willingly convert over to the IT-provided hybrid cloud solution without resistance. In addition to saving the company money, this approach allows IT to put in the proper security and compliance checks into the hybrid cloud platform without impacting business innovation.

For more information about Shadow IT, refer to the following post published on the IBM Thoughts On Cloud blog "Go ahead, invite shadow IT to the party. Here's how" at the following site:

http://www.thoughtsoncloud.com/2015/04/go-ahead-invite-shadow-it-to-the-party-heres-how

1.2 What is hybrid cloud?

To fully understand what hybrid cloud is, we must first understand what cloud is. This section provides an introduction to cloud in the context of the characteristics of and differences between service and deployment models. With an understanding of these models, we can then begin to understand what we can do to integrate them and create a hybrid cloud or a hybrid IT model.

During the last few years, new cloud infrastructures have been developed and improved from multiple sources and companies, but the main idea of cloud remains the same: Ubiquitous network access to a pool of computing resources.

The main objectives of these clouds environments are to improve agility and productivity with performance, reliability, and scalability. At the same time, getting cost reduction and standardization without giving up on security.

Because companies have different service requirements, it was necessary to develop different clouds infrastructures to meet all of them. We have the following major service models to meet different services:

- Infrastructure as a service (IaaS)

  Infrastructure as a service is the basic method for most of the cloud providers at this moment. It lets you request the infrastructure that you need to develop your own cloud solution based on some infrastructure options such as processor, memory, storage, and network.

- Platform as a service (PaaS)

  Platform as a service provides a platform that allows customers to develop, run, and manage web applications without the complexity of building and maintaining the infrastructure typically associated with developing and launching an application.

- Software as a service (SaaS)

  The software as a service method allows you to use the required software in cloud that is priced as “pay-per-use”. This eliminates the requirements to manage the infrastructure, software license, and maintenance costs required to run your solution.

Figure 1-3 visually explains the differences between the cloud service models against the traditional on-premises solution.

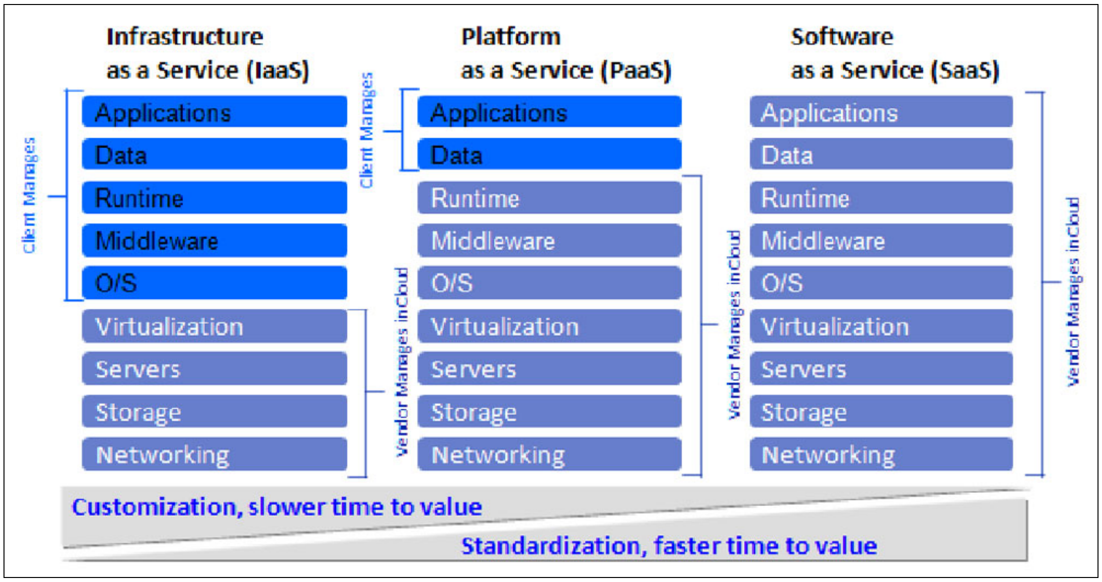


Figure 1-3 Cloud service models

Sometimes, organizations have complex and aggressive security requirements. Thus, different cloud deployment models were created, the most common of which are described here:

- Public Cloud

  The public cloud deployment model uses public network and shared compute, storage, and network resources to serve all customers. Because of that, security configurations and infrastructure are shared and often could not be changed to support a specific customer requirement.

- Private Cloud

  The private cloud deployment model relies on a dedicated infrastructure built to act as a cloud environment to operate for a specific organization. This model allows the organization to use their own specific infrastructure, security, and connectivity requirements.

The hybrid concept was created to solve some of these specific requirements. The idea is to use the best of each model (public and private). The difference between hybrid cloud and hybrid IT is the existence of connectivity to a non-cloud environment. In other words, *when you connect multiple cloud environments (public and private) to develop a solution, you create a hybrid cloud environment. When you connect multiple clouds together with some dedicated non-cloud infrastructure in your premises, like your existing environment, you built a hybrid IT model.*

Figure 1-4 shows the differences between hybrid cloud and hybrid IT.

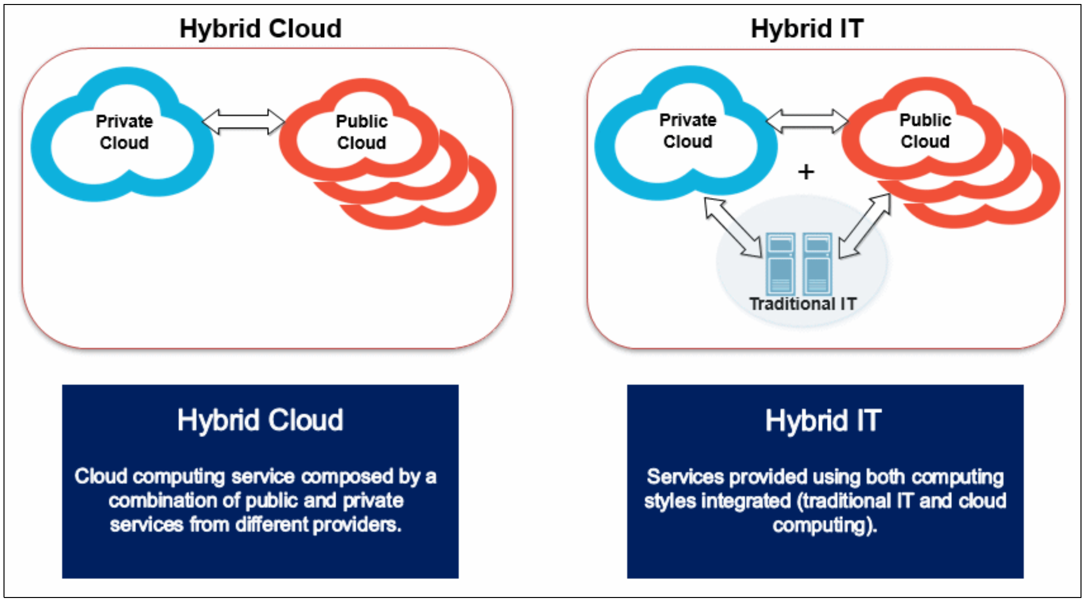

Hybrid Cloud
Private Cloud
Public Cloud
Hybrid IT
Private Cloud
+
Public Cloud
Traditional IT
Hybrid Cloud
Cloud computing service composed by a combination of public and private services from different providers.
Hybrid IT
Services provided using both computing styles integrated (traditional IT and cloud computing).

Figure 1-4   Hybrid cloud versus hybrid IT

1.3 Hybrid cloud customer scenarios and use cases

In this section, we cover how hybrid cloud solutions solve the business challenges of several industries. The goal in this section is to showcase how different industries benefit from hybrid cloud solutions.

1.3.1 Telecommunications industry background

In the telecommunications industry, communications service providers (CSPs) are facing declining voice revenue and have to find new services to help grow their Average Revenue Per User (ARPU). In order to increase the ARPU, CSPs have strategically worked to identify services that complement their broadband offering in such a way that consumers will not turn to other providers. This is even more important in mature markets where the market for equipping new households is limited. In addition, CSPs have started to introduce different forms of services by leveraging digital content like IP television, video on demand, or music on demand.

Furthermore, gaming is another form of digital content that CSPs are starting to use as well. Cloud gaming is a service by which games can be provided instantly, on-demand to consumers.

1.3.2 Telecommunications example CompanyC background

To showcase the business challenges and hybrid cloud solutions that a typical telecommunications company might be experiencing, we are sharing a case study about a telecommunications company, which we refer to as *CompanyC*. CompanyC is a mobile network operator (MNO) and service provider based in Moscow in the highly competitive Russian market. Established in 2001, it is a relatively new entrant to the market with high ambitions to compete in the Russian mobile market. It is a privately owned company with significant venture capitalist investment from Dubai in the Middle East. The investment team from Dubai is keen to see CompanyC gain market share and improve its profitability before being floated on the Russian Stock Exchange. No date has been set for the flotation, though outsiders expect this to take place in the next 2-4 years.

CompanyC’s primary market is the delivery of mobile services to consumers, fixed broadband services to homes and businesses, and an increasingly important corporate account business for unified telecommunications, and basic hosting services. Most of its mobile services (89%) are delivered in Western Russia (centered around Moscow), and some services are delivered in Ukraine and Poland. Coverage of services is concentrated in the major towns and cities. Most of the mobile transmission infrastructure is rented from its biggest competitors, though it does own its own broadband infrastructure, corporate network infrastructure, and some mobile transmission capabilities within Moscow. It has recently rented two new data centers outside Moscow, where it hosts the operations center for its telecoms delivery, both mobile and fixed, as well as providing capacity for its growing hosting services delivery ambitions.

1.3.3 CompanyC IT landscape

Today, CompanyC uses two data centers. The first data center is in Moscow East, and the second data center is in Domodedovo, with data replication between the two data centers. Both act as the primary, with failover capability to the other data center.

In the data centers, they host their core Business Support Systems on traditional distributed systems. In addition, CompanyC is invested heavily in the transition to client/server as well as a strong web presence, which is built on a combination of Power and Sun based UNIX systems. These systems host various databases including their primary customer portal, which is currently integrated with their core billing system.

1.3.4 CompanyC business initiatives

The management at CompanyC would like the ability to provide longer term customers to have a higher ARPU, and buy more high-margin products. The management at CompanyC realizes that revenue depends on having competitive quantity and quality of products offered to the marketplace, and the reach of those products. New products are required to retain competitive standing. Over time, the price of products decreases due to competition; thus, new products and retention are required to maintain revenue.

CompanyC faces a serious challenge to deliver on these new types of applications because their existing infrastructure, although efficient, is not agile. It sometimes takes a few weeks to a month to get these new applications online, and in today's connected world, the loss of time to market has made them a "me too" company in terms of social reach. That is about to change.

1.3.5 Solutions and actions by CompanyC CIO

In order to drive innovation of new production offerings to provide longer-term customers to have a higher ARPU, the CIO has made a decision to push new development to cloud-based platforms. The CIO has criteria of high availability, outsourcing of platform management, and the connection of two data centers to the cloud in a secure, manageable, and auditable way.

The CIO has decided to go with IBM SoftLayer® as the cloud vendor for a number of reasons. First, CompanyC IT needs more than virtual machines (VMs) and some storage. They need to manage applications that can be hosted only on bare-metal servers. In addition, they also want an agile, next generation, application development environment that allows CompanyC's line-of-business developers the freedom to innovate quickly but in a way that allows them to ensure that proper business and governance requirements are met. The IBM Bluemix offering has many of the features the company is looking for in the next generation platform.

Another consideration for CompanyC is that the labor pool for new developers includes developers who are familiar with technologies that they will want to work with and be productive with, which also keeps them relevant in the labor market.

The following solutions will be implemented to address the business initiatives:

- Video Streaming Service
- Gaming Streaming Service
- Music Streaming Service

First, CompanyC wants to implement the Video Streaming Service. The goal for this project is to use customers current pay-per-view activity to determine ideal video steaming content that is appealing to each customer. See Chapter 4, "Connecting to an enterprise database of record" on page 71 for implementation of this scenario.

The second initiative for CompanyC is to implement a gaming streaming service. This service uses applications that were developed in docker containers on-premises and CompanyC wants to connect those docker containers to docker containers running in the public cloud. See Chapter 5, "Connecting IBM Containers with on-premises Docker" on page 91 for implementation of this scenario.

Finally, CompanyC wants to streamline the order delivery process to optimize their music streaming service. This includes connecting on-premises data and exposing it to customers in a mobile application. See Chapter 9, "Mobile hybrid scenario: Secure Gateway, Connect & Compose, and DataWorks" on page 219 for implementation of this scenario.

1.3.6 Retail industry background

The retail industry is composed of companies that sell consumer goods or services in small or individual lots for direct consumption by the purchaser through multiple distribution channels to earn profit. The retail market is made up of several market segments, which include drug stores, grocery stores, discount general merchandise, speciality apparel, and department stores. Although each market segment sells different goods to different consumers, their internal organization structure and business challenges tend to be similar.

Retail companies need to understand their customers better and discover ways to optimize operations in order to drive profit in an increasing competitive industry. Many retailers today focus on three key business initiatives. First, they focus on providing a superior shopping experience. This includes personalized experiences that enable customers to shop in store and across channels when and how a customer chooses. Next, they focus on the creation of demand-driven merchandising and supply chain. This means providing fully integrated and consumer insight-driven merchandising and supply chain that align products and services with shopper profile demand. Finally, they focus on operational excellence, which includes integrated systems and processes that achieve operational efficiencies, analytical insights, and fast development and deployment of new capabilities.

1.3.7 Retail example CompanyB background

To showcase the business challenges and hybrid cloud solutions that a typical retail company might be experiencing, we share a case study about a retail company, which we refer to as CompanyB. CompanyB has provided quality food distribution for almost 50 years. Started by a small family in Ohio, it served a local market that quickly created a name for itself. In addition to providing freshly baked bread to the community it served, they also expanded products and offerings to include jams, coffees, and teas. The two brothers that founded the company developed a vision of delivering quality products on time and at a fair price. With this as their simple mantra, success followed them.

As their business expanded and they hired additional people to prepare, transport, and deliver goods, they also found that they also had to begin finding ways to process orders, invoice customers, and efficiently deliver their goods as the territory they served grew. Today, CompanyB operates in all 50 states and has expanded its food products to include all of the items that you would find in a traditional grocery store.

Their mission statement continues to this day, “Delivering quality products on time and at a fair price.” In addition to quality products, the brothers developed a sophisticated distribution and delivery system that manages 3500 tractors, 45,000 trailers and employs approximately 4000 drivers. Managing the logistics of this distribution fleet has been successful because CompanyB also focuses on best-of-breed solutions to manage their business. Information technology (IT) is the bedrock of their on-time and fair price part of their organization. CompanyB is considering how to expand their offerings to a more global footprint.

1.3.8 CompanyB IT landscape

The IT department at CompanyB has been invested in computing technology from early on in their journey. Today, they run three data centers across the country with IT reaching into the stores and distribution centers as well.

In the data centers, they host their core Business Support Systems on IBM z™ Systems. They are running approximately three z Systems mainframes in each data center. Supporting applications are hosted on IBM CICS® and IBM z/OS® DB2 systems. They also invested heavily in the transition to client/server as well as a strong web presence and have also included IBM POWER® and Sun based UNIX systems. These systems host various databases as well as SAP systems that help them optimize their logistics, customer campaigns, and other customer satisfaction efforts targeted at improving reach to customers.

Their communications network has moved to localized broadband endpoints in their stores and distribution centers, which has allowed them to retire most of the previous systems that were based on satellite.

1.3.9 CompanyB business initiatives

The current management at CompanyB is interested in more aggressively reaching its customers with information about sales and convenience services. For sales, CompanyB wants to take advantage of social communication channels like Facebook, Twitter, and Instagram to quickly alert customers to specials that are managed at a local level. In addition, they want to use mobile endpoints to improve their interaction with customers. For instance, the marketing department has created a new offering that tracks items commonly purchased by customers and has created the “virtual shopping list.”

In concept, the virtual shopping list tracks customer spending and determines patterns over time of repeating purchases of common items like toothpaste, sugar, and other household consumables. For customers that are running the CompanyB application called the *baguette* on their smartphone, CompanyB sends push notifications to the customers to help remind them of items that are most likely in need of. The application also provides a discounting option to encourage customers to get these items during their current visit.

CompanyB faces a serious challenge to deliver on these new types of applications as their existing infrastructure, although efficient, is not agile. It sometimes takes a few weeks to a month to get these new applications online, and in today’s connected world the loss of time to market has made them a “me too” company in terms of social reach. That is about to change.

1.3.10 Solutions and actions by the CompanyB CIO

The CIO has made a decision to push new development to cloud-based platforms. Some of his criteria are enterprise-grade availability, outsourcing of platform management, and the connection of his three data centers to the cloud in a secure, manageable, and auditable way. He has seen first hand what happens to companies where data is compromised, and CompanyB holds customer satisfaction as a high-priority item. The CIO is also looking for ways to reduce his capital expenditures and convert that spend into operational costs that will help address profitability as requested by the CFO. This means renting servers and services rather than building them in-house.

The CIO decided to go with IBM SoftLayer as his cloud vendor for a number of reasons. First, CompanyB IT department needs more than VMs and some storage. They need to manage some applications that can only be hosted on bare metal servers. In addition, they also want an agile, next generation, application development environment that allows CompanyB’s line-of-business developers freedom to innovate quickly but in a way that allows them to ensure that proper business and governance requirements are met. The Bluemix offering by IBM looks interesting and has many of the features that she is looking for in the next generation platform.

One other consideration that CompanyB is taking into account is that the labor pool for new developers is full of developers that are familiar with technologies that they will want to work with and be productive with, which also keeps them relevant in the labor market.

The following solutions are implemented to address the business initiatives:

- Mobile purchase order approvals
- Virtual shopping list
- Predictive analytics

First, CompanyB wants to implement the mobile purchase order approvals. The goal for this project is to be able to approve purchase orders inside the SAP system directly from mobile phones. The goal is to reduce time to deliver and purchase any order and also give mobility to the management level. With this application, CompanyB wants to speed up the purchase process and reduce the costs during the supply chain process. See Chapter 6, “Connecting Bluemix applications to your local (on-premises) enterprise SAP system” on page 131 for implementation of this scenario.

The second initiative for CompanyB is to implement the virtual shopping list. Leveraging decades of data on its IBM z Systems platform, the developers are actively working to group and analyze previous purchases. See Chapter 7, “Exposing CICS transactions with z/OS Connect” on page 157 for implementation of this scenario.

Finally, CompanyB is looking for ways to analyze previous purchasing decisions so that they can quickly connect its customers with new products and offerings by making the customers aware of them first to keep the customers engaged as well as maintain CompanyB’s competitive position as a leader in food distribution. See Chapter 8, “Watson Analytics in hybrid cloud using Secure Gateway and DataWorks” on page 195 for implementation of this scenario.

2

Hybrid cloud architectures: Three pillars of integration

In this chapter, we introduce the three pillars of integration for hybrid cloud architectures for applications. Delving into how each pillar functions and what businesses should expect from each of these three pillars can help you gain an understanding of how to plan your own hybrid cloud applications. For each hybrid cloud architecture, we provide a general overview, an example architecture and breakdown, and a client scenario with a set of requirements and concerns.

This chapter has the following sections:

- 2.1, “Cloud integration: An introduction to the three pillars” on page 16
- 2.2, “Pillar 1: API-centric hybrid cloud integration” on page 16
- 2.3, “Pillar 2: Data-centric hybrid cloud integration” on page 18
- 2.4, “Pillar 3: Event-centric hybrid cloud integration” on page 20

© Copyright IBM Corp. 2015. All rights reserved.

2.1 Cloud integration: An introduction to the three pillars

In the previous chapter, you gained a better understanding of the hybrid cloud, including its features, purpose, and strengths. With that in mind, we now look at the three main pillars of hybrid cloud applications and what role they play in application development. CompanyC and CompanyB both have unique development processes, infrastructure, requirements, and concerns. We look at both companies to achieve a better understanding of what each company should consider to drive success with their respective hybrid cloud infrastructure and applications. From there, we use these requirements, goals, and concerns to lay out several application scenarios for each company in 1.3, “Hybrid cloud customer scenarios and use cases” on page 9.

2.2 Pillar 1: API-centric hybrid cloud integration

As time passes and the cloud industry progresses, so too do hybrid cloud architectures. The cloud is entering a point where applications are composed from various application programming interfaces (APIs) purchased from various service providers and used on more traditional systems. This API-driven development allows businesses to achieve higher levels of agility by accessing various services as needed and by paying for their usage in a flexible manner. As the cloud grows in popularity, the number of services available to business also increases in number. Applications are starting to use more cloud-based APIs, and being able to properly manage and integrate with these services is important for driving an application’s success.

2.2.1 API-centric architecture example

For this example, we have a grocery store chain that has several enterprise on-premises web services for its grocery products. The company wants to expose these web services as APIs. Each of these APIs returns various information based on the product information provided. The APIs also return more or less information based on the access rights of the user or application making the request as follows:

- Price: The cost of a specific product the store offers:
  - No charge for internal use and no charge for open source applications.
  - Charges per query for third-party applications that generate revenue.
- Stock: The quantity of a specific product available for purchase within the store:
  - No charge for both internal and external applications and users.
  - Provides internal users and applications with the full quantity of an item in stock for the store.
  - Third-party API requests return the exact number of a product in stock if there are fewer than 40 items left in stock. Otherwise, it just returns in-stock.
- Nutrition: The nutrition information for food products that are available within the store:

  No charge for both internal and external applications and users.
- Purchase order history: Returns information regarding the purchase order history for a product the store offers, including dates, quantities, and cost:

  Is for internal applications and users only and should not be made available for third-party services or external users.

Figure 2-1 illustrates this scenario.

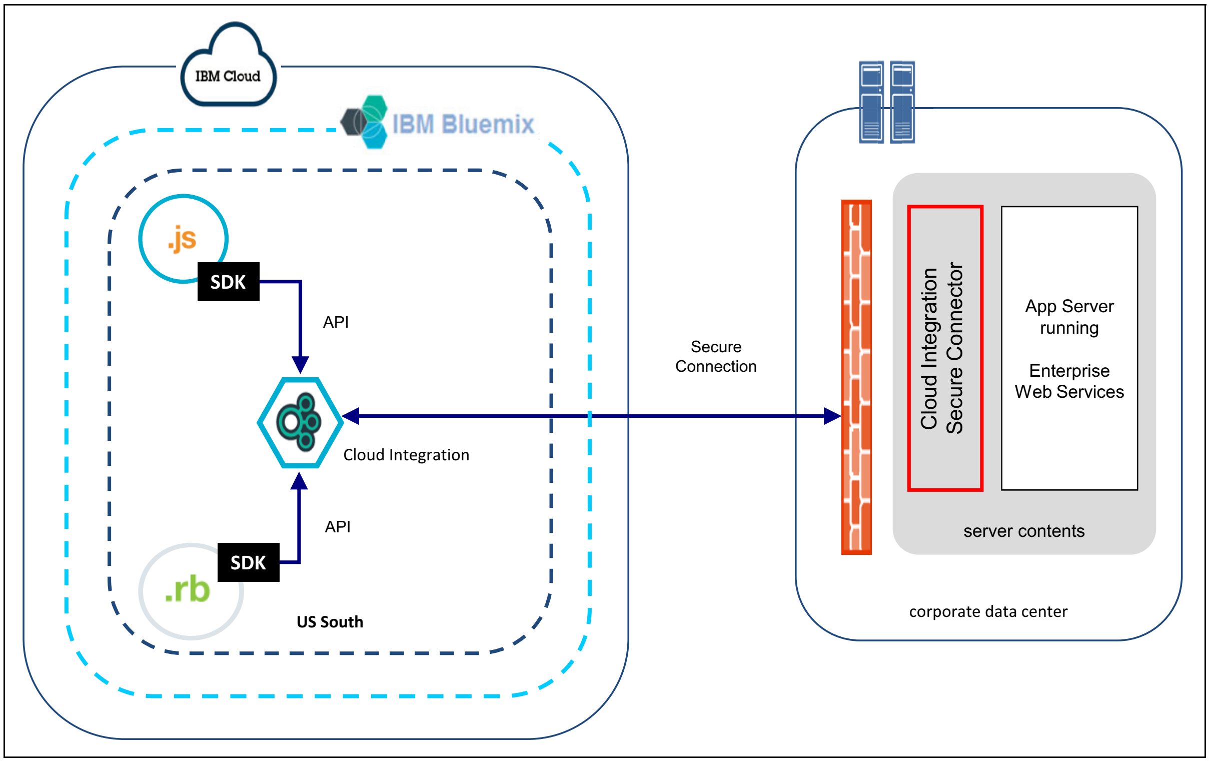


Figure 2-1 Exposing multiple enterprise on-premises web services as APIs

2.2.2 API-centric requirements

The telecommunications company, CompanyC, is excited to begin using the capabilities of a hybrid cloud infrastructure. They hope that their new API-centric infrastructure allows them to bring their applications to market faster by bringing in new services and integrating them with their data. By using APIs and hybrid cloud, CompanyC wants to drive the creation of various new applications to bring to market to stay competitive. Thus, the company wants to ensure that integration to and from cloud software as a service (SaaS) is simple and painless.

Expecting to generate and market APIs of their own, CompanyC also expects the capability to easily expose the functions of their own applications. Although this ease of use is important, CompanyC also wants a hybrid cloud solution that keeps their data secure because the data is exposed to the cloud. Thus, the solution that they need must maintain total compliance with their security standards.

CompanyC has the following company goals:

- Simplify integration to and from cloud services (SaaS).
- Easily expose application functions through APIs.
- Easily connect and work with multiple cloud-based services.
- Protect company data to and from the cloud.

2.2.3 API-centric concerns

As CompanyC transitions to a hybrid cloud infrastructure, they have a set of concerns about the transition and want to ensure that their cloud solution addresses these potential areas of concern. Like many companies moving to the cloud, the loss of direct control over all aspects of their applications is a big change. CompanyC wants to ensure that they are able to maintain a high level of control over their APIs, including how applications are called as well as whether they are allowed or denied access based on several factors. CompanyC also has several service level agreements in place and needs to ensure that they can continue to support these commitments to their clients easily, while retaining the ability to develop new functions.

CompanyC has the following API-centric concerns:

- Many businesses developing applications that expose functionality through an API want greater control over access to API calls.
- Development teams express a need for contextual control over several factors for API calls, such as:
  - Device type
  - Device ID
  - User credentials
  - Geolocation
- As the number of devices connecting to applications expands, such as within the Internet of Things, a system that can scale to handle millions of devices and users becomes necessary.
- Security of data is a top priority when working with APIs in the cloud. Protecting data, devices, and applications from unauthorized access attempts and other external threats through a secure gateway protects businesses from disaster.
- Many businesses with API-centric hybrid cloud implementations demand greater insight around their application, which can be delivered by tracking API usage through logging and monitoring services.
- Along with APIs come API lifecycles, managing updates to an API, and a business's service level agreements. It is the job of a trusted service provider to ensure that they can manage and support their customers throughout an API's lifecycle and into the next.

2.3 Pillar 2: Data-centric hybrid cloud integration

When developing applications in the cloud, it is a fairly common need to move data from place to place. If your cloud-hosted applications are pulling data from an on-premises database for example, it is important to know that your data is protected. This is especially true when dealing with sensitive information, such as financial and personal data, or information that is marked confidential by your company. The data-centric pillar of hybrid cloud integration focuses on just that, moving data throughout the cloud to where it is needed in a safe and efficient manner. Companies that plan to move their data across the cloud need to understand that when data is out of their hands, they release all control over its security measures. Encrypting data before any form of transfer, and in a way that still allows for fine grained, file level control, without negatively impacting performance marks a proper data-centric solution.

2.3.1 Data-centric architecture example

A mobile application developer has two mobile applications that in general have a similar userbase. Each mobile application has its own database for storing user information. The newer application has a database hosted in the cloud; the other application uses a large on-premises database. After receiving feedback from clients, the company wants to allow users to migrate a profile to the other application instead of needing to create a new account from scratch. One problem the company faces is that its older application has a large stable userbase that is difficult to move off-premises due to its size and complexity, and its newer application is growing too fast and fluctuates greatly and thus needs to stay in the cloud. The company has decided to keep the two user databases separate while still allowing the user to create a user account from an existing one within the other application. Afterward, the two accounts would be kept in sync where any profile information overlapped. The company has put security as a high priority here because users' accounts contain user names, emails, passwords, and usually credit card information, and for any of this information to leak or become accessible to hackers would mean disaster for the company's reputation.

Figure 2-2 provides a data-centric architecture example.

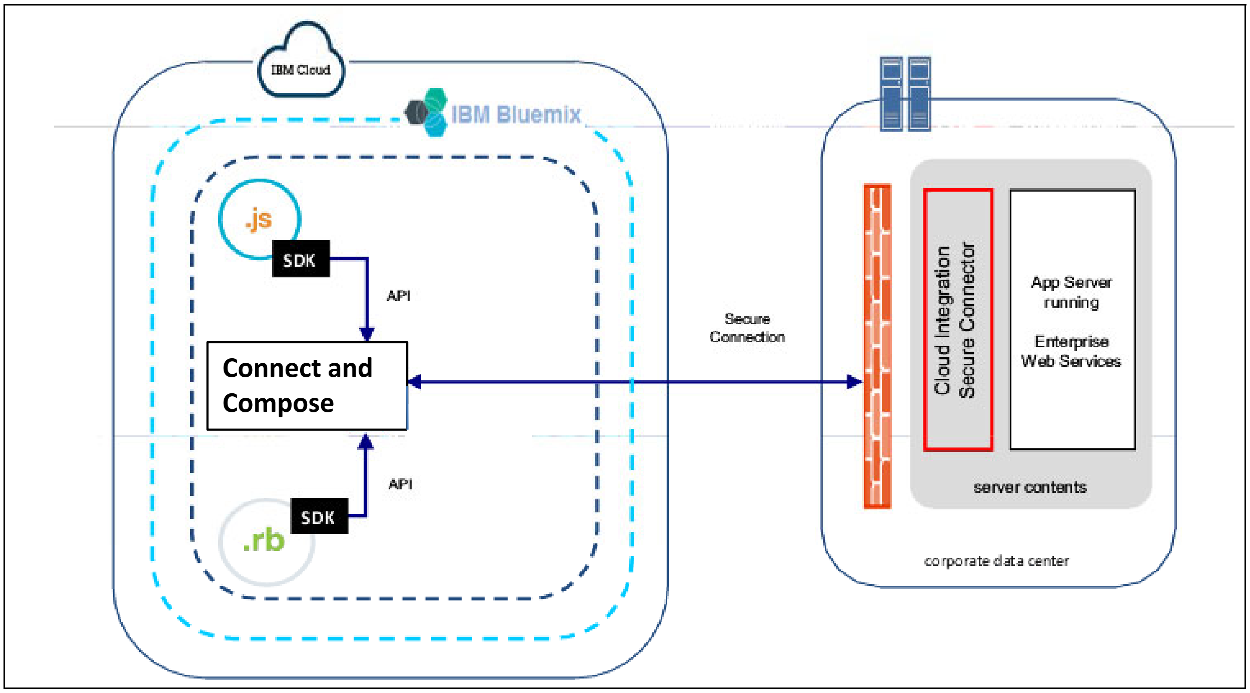


Figure 2-2 Database data migration and synchronization with hybrid cloud

2.3.2 Data-centric requirements

With its new business initiatives, CompanyC is expanding its infrastructure into the cloud. Part of this transition involves connecting its two established private data centers to this new hybrid cloud infrastructure they are creating. With this expansion comes serious concerns for CompanyC. They have a long, tried, and tested history of protecting the data of their clients. As they expose their data to the cloud, they need to ensure that it is protected with proper encryption methods, while also keeping their data easily accessible to their various users and applications when needed.

CompanyC has the following requirements for this environment:

- A process of firewalling data that allows for the transfer of sensitive data across multiple untrusted environments without a need for network security with a protection profile maintained by the payload.
- To comply with company security requirements even when their environment is a multi-tenant environment that is run by a third-party organization, proper security measures need to be in place.

2.3.3 Data-centric concerns

Now that CompanyC is going to have data flowing through the cloud and thus, through third-party services, they want to be sure that their data is safe even inside an environment where they have little to no control over security measures. A rogue administrator with access to sensitive files has become a real concern. And with access to sensitive files, such as earnings reports or the personal information of clients and employees, serious harm could be done to their business. Although CompanyC wants administrators and other privileged users to be able to perform their operations on their data, such as backing up sensitive files, they do not necessarily want to allow those privileged users to be able to access those files in a readable format. Because of this need for heightened security around sensitive files, CompanyC also wants logging and metrics on them.

They need to track access to their files and recognize if a user’s access patterns seem suspicious or abnormal:

- Protect highly sensitive data against privileged users such as system, network, and domain administrators.
- Privileged users should be able to locate, back up, and manage sensitive data without being able to access the information within.
- Protect data with fine-grained access controls with detailed monitoring, auditing, and reporting of access attempts to sensitive files.
- With the use of pattern recognition with data access logging, suspicious data access of files can be reported to users or administrators.

2.4 Pillar 3: Event-centric hybrid cloud integration

The event-centric pillar of hybrid cloud integration revolves around applications that are running based on events or messages. The number of data sources and destinations involved in creating solutions are growing at a rapid rate. The new event-centric approach to hybrid cloud aims to simplify the task of managing these messages while increasing the overall reliability of the system. Event-centric applications work so well in the cloud due to the varying intensity and frequency of events. These fluctuations fit well into a cloud infrastructure that can dynamically scale to fit those needs. An event-centric approach cuts down on communication overhead for an application, thus helping to speed up the development process. The fact that resources can be easily added, removed, and reused in the cloud also makes for a much more agile development process.

2.4.1 Event-centric application architecture example

Figure 2-3 displays an example architecture for an event-centric application.

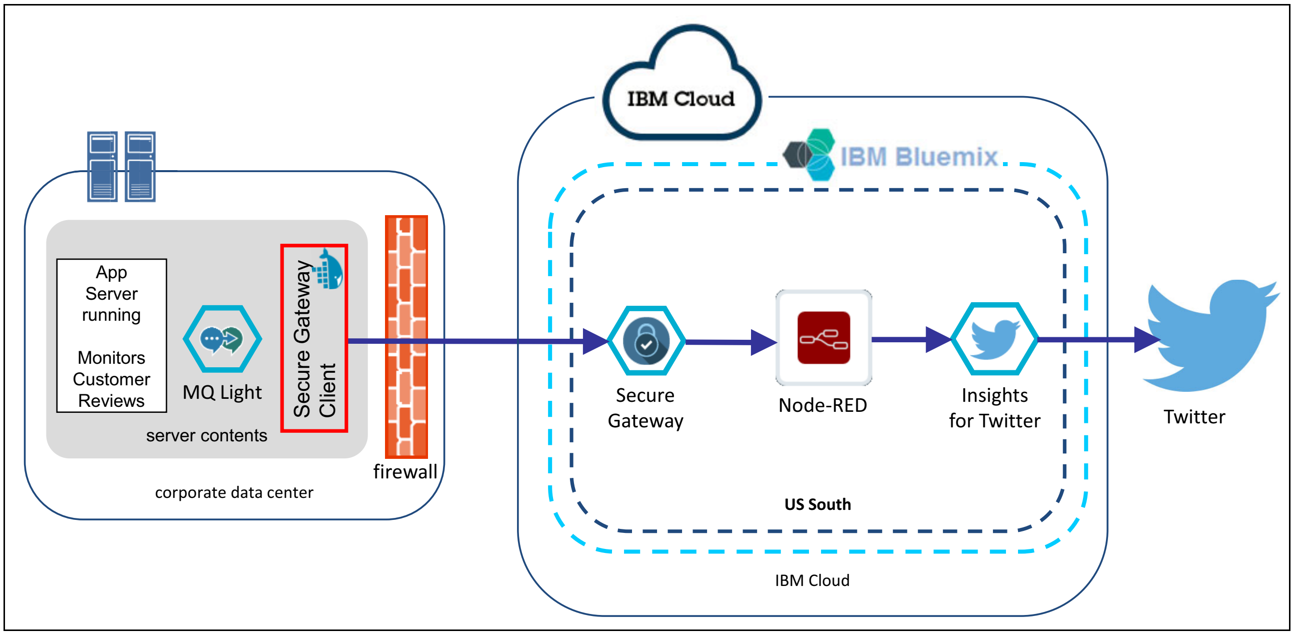


Figure 2-3 Event-centric example architecture diagram

The example client had an existing system for handling product reviews on a bare metal on-premises system. They wanted to develop an experimental application that would take incoming reviews and by using analytics, determine which product reviews were positive and post them to Twitter. The client doesn't want to change their current architecture. However, the client strongly believes that this application should be agile and flexible in resources and, as such, wants to move development to the cloud.

They chose a hybrid cloud architecture with IBM Bluemix.

In this scenario:

- MQ Light and Node-RED boilerplate are used to exchange information in the form of messages from on-premises systems and IBM Bluemix.
- Secure Gateway Client and Secure Gateway provide secure data connection between on-premises systems and IBM Bluemix.
- Insights for Twitter is used to enrich Tweets with sentiment and other insights for multiple languages, based on deep natural language processing algorithms from IBM Social Media Analytics.

2.4.2 Event-centric requirements

CompanyB is striving to take advantage of the various possibilities of event-centric hybrid cloud applications. With their business initiative to integrate more of their company's services and applications with social media, event-centric applications seem like a great fit. Now that their data is pushed and pulled from so many locations, orchestration of that flow of information stands to grow vastly in complexity. Feeding that data into various data stores as well as analytic applications needs to be as simple as possible while staying secure.

CompanyB’s CIO has also put forth initiatives to bring down costs of IT and resource costs of development. By simplifying the overhead of communication for their various applications, CompanyB hopes to create more flexible and affordable IT through event-centric solutions. As for development, CompanyB expects to be able to add and drop resources easily between development projects and need the ability to scale those resources to fit the needs of development. Making their development process more agile through the power of cloud.

CompanyB has the following goals:

- Highly cost-effective, responsive, and elastic IT
- Flexible dynamic consumption of resources to avoid large overhead
- Easy to manage and deploy cloud orchestration for clouds of varying types
- The ability to view, monitor, automate, and control off-premise and on-premises components of a hybrid cloud
- Speed time-to-market of applications by separating and simplifying server, storage, and networking capabilities
- Allow for easier application experimentation and relocation of resources for scrapped projects or applications

2.4.3 Event-centric concerns

Although moving to the cloud offers a great reduction in costs and a more agile development, CompanyB does have some concerns about the implementation of event-centric cloud applications. For starters, they have a wide array of data sources with various formats and transfer protocols. They want to be sure that their cloud solutions are robust enough to handle all that variation. Likewise, with so many sources and destinations, the solutions they choose need to simplify the process of properly connecting all that data drastically. Their developers need more time to focus on development for CompanyB’s new agile business initiatives, and they want every edge possible in shortening their time-to-market. As such, their cloud solutions need to automatically handle as much of the permissions, security, and overhead of data messaging as possible.

With several existing and planned applications, each with their own set of needs, CompanyB also needs its event-centric solutions to be highly configurable, as well as reliable:

- Currently, development for CompanyB is conflicted by their strive for connectivity, which directly causes strain on their resources and costs when innovating in development. They need their event-centric solutions to help these two goals coexist.
- Handling various data sources and applications can get highly complicated. Event-centric solutions need to ensure that things are kept simple for developers to give them enough time to innovate in development.
- CompanyB has thousands of data sources and different forms of messaging that they want to integrate into their applications. Their tooling needs to be able to handle such variance.
- Along with high configurability, there exists various levels of security and authorization within CompanyB’s various messages, which all need to be handled in a reliable manner.

3

Introduction to IBM provided hybrid cloud services and products

In Chapter 2, “Hybrid cloud architectures: Three pillars of integration” on page 15, we saw three domains or pillars of cloud integrations that are important for a hybrid cloud. We provided users with an understanding of architectures and patterns that lay the foundation. In this chapter, we look at specific IBM services and products that can be used to seamlessly integrate different cloud environments in a robust and efficient manner. We can put these services and products under two categories. The first category is services hosted on IBM Bluemix. The other is stand-alone IBM products. We provide a definition, a description, and use cases to help users understand which services or products are applicable for their scenarios or problems. As discussed, we are relating these products to use cases in later chapters so that you can read the relevant chapters to further understand the functionality of the product.

This chapter has the following sections:

- 3.1, “Services hosted on IBM Bluemix” on page 24
- 3.2, “Other IBM products” on page 56

© Copyright IBM Corp. 2015. All rights reserved.

3.1 Services hosted on IBM Bluemix

Bluemix integration services offer a range of composable services to connect to data and endpoints (application programming interfaces (APIs) or services) behind the firewall securely and easily. These endpoints can be easily exposed and managed as APIs for consumption by omni-channel applications. In-flow data cleansing and movement ensure that applications have the correct and relevant data. In this section, we provide an overview of these Bluemix services and explain use cases. You can use these *building blocks* to compose a solution to connect disparate clouds incrementally.

Services on Bluemix can be categorized in three main areas:

- API-centric
- Data-centric
- Event-centric

Refer to Chapter 2, "Hybrid cloud architectures: Three pillars of integration" on page 15 for a detailed explanation of these categories.

In the following sections, we cover Bluemix provided services that fit within these categories.

Additional references on Bluemix integration services: You can refer to the following as additional references on hybrid cloud:

- *5 Things to Know about Hybrid Cloud*

  https://www.ibm.com/developerworks/community/blogs/5things/entry/5_Things_to_Know_about_Hybrid_Cloud?lang=en

- *Creating Hybrid Clouds with IBM Bluemix Integration Services*, REDP-5270

  http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/redp5270.html?Open

3.1.1 API-centric service: IBM API Management

The API Economy is where companies (providers) expose their internal digital business assets or services in the form of APIs to third parties (consumers) with the goal of unlocking additional business value through the creation of new assets. Figure 3-1 illustrates the API Economy value chain described.

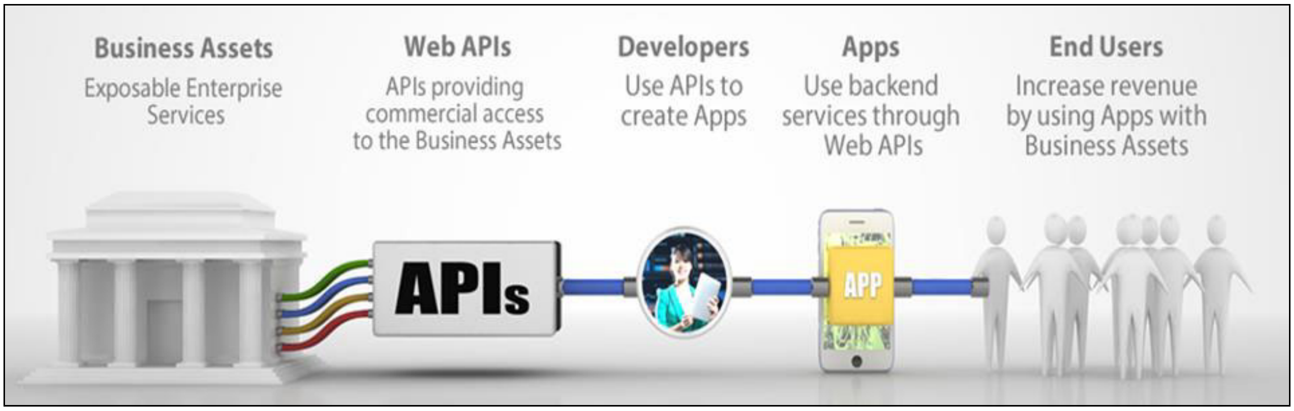

Business Assets
Exposable Enterprise Services
Web APIs
APIs providing commercial access to the Business Assets
Developers
Use APIs to create Apps
Apps
Use backend services through Web APIs
End Users
Increase revenue by using Apps with Business Assets
APIs
APP

Figure 3-1 The API Economy value chain

The IBM API Management solution provides a single, comprehensive solution to design, secure, control, publish, monitor, and manage APIs. It delivers the following sets of capabilities:

- An API developer can create, secure, control, deploy, analyze, and manage SOAP and Representational State Transfer (REST) APIs and services for internal or external consumption quickly through a single console.
  - Policy-driven assembly that enables custom rate-limiting scenarios, which are enforced by the API Gateway (IBM DataPower®).
  - Fine-grained ability to set API quotas and rate limits.
  - Visibility and control with ability to deploy across runtime environments.
  - Ability to easily create different versions of an API and understand where those versions are deployed, such as in test, staging, or production.
  - Ability to search for, add custom labels to, and mark favorite APIs and services for easier discovery in API Manager to work with APIs.
  - Ability to notify within API Manager to understand who is working on the APIs and any issues the APIs might be having.
  - Support for SOAP-based web services.
  - Discovery of web services from WebSphere Service Registry and Repository (WSRR).
  - Ability to create an API by assembling REST-based and SOAP-based services.
  - Role-based access for viewing and working with the APIs.
  - Support for OAuth 2.0, an open standard for authorization.
- An API business owner can advertise, market, socialize, and sell APIs as a product in developer communities worldwide:
  - Ability to have more than one developer portal in order to support more than one developer community (private, partner, and public), with controlled visibility.
  - Ability to create API plans, which treat the APIs as product offerings, allows several APIs and resources per plan.
  - Improved application developer management, with the ability to send an email to all application developers in a particular community.
- An application developer (also API consumer) can explore API documentation and provision application keys:
  - Developers can register their application, select API entitlement levels, and monitor their API usage.
  - A single ID allows a developer to be a member of multiple API Management communities.
  - Allows role-based access for viewing APIs.
- IT Operations can easily manage and upgrade the API Environment using existing investments in DataPower with the ability to monitor and scale without disruption to service:
  - Simplified operations environment for API assembly using DataPower Gateway.
  - Improved environment console experience.
  - Improved tenant management with the introduction of organization and owners.
  - Integration with enterprise authentication systems by using Lightweight Directory Access Protocol (LDAP) that enables administrators to streamline the user login process.

- Simplification of deployment architecture to only two tiers:
  - Gateway tier: DataPower appliance (virtual or physical appliance).
  - Management tier: API Management virtual appliance.

IBM API Management is available in four different deployment options. Table 3-1 on page 26 compares the different options. For more information, see the following sites:

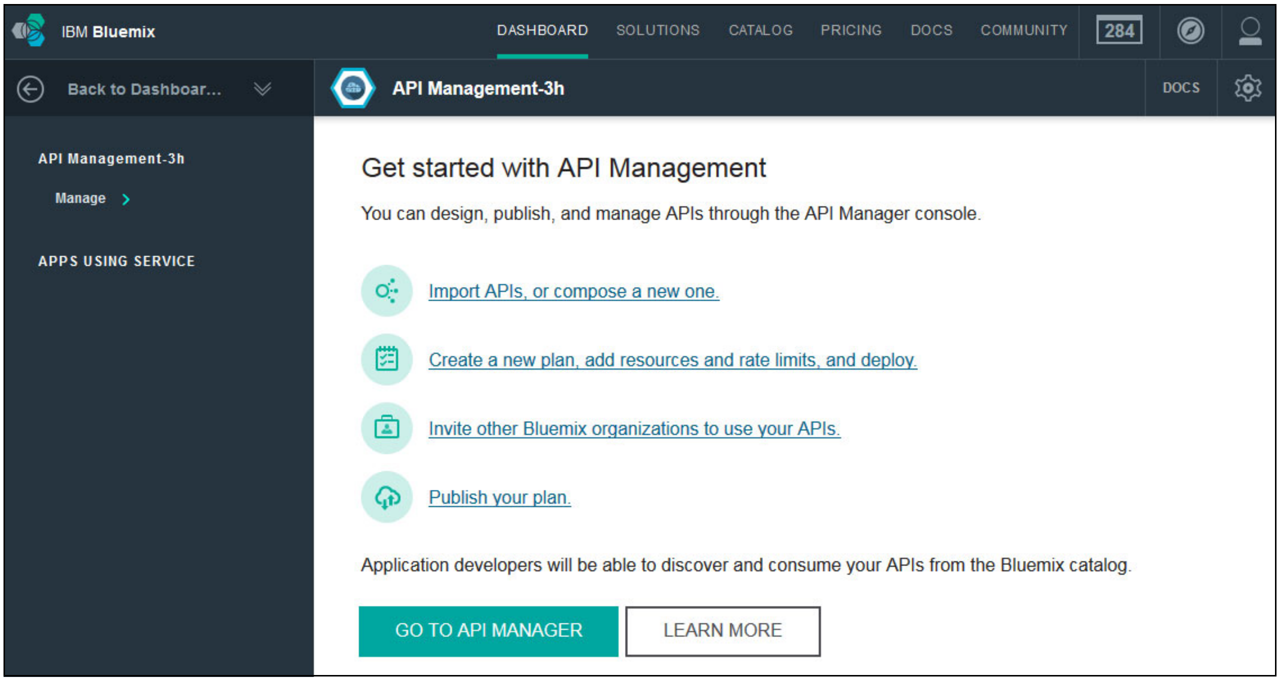
- `http://www.ibm.com/software/products/en/api-management`
- `http://www.ibm.com/support/knowledgecenter/SSZFB2_3.0.1/mapfiles/ic_service_home.html`

Table 3-1 Comparing the IBM API Management deployment options

Offering name	**Deployment model**	**Offering management**	**Infrastructure details**	**Capabilities**
IBM API Management	On-premises	Customer owned and managed	► Can be installed on VMWare, Xen, and PureApplication System ► Requires IBM DataPower Gateway	Provides functionalities to define, secure, control, version, and publish API, and monitor API usage
IBM API Management on Cloud	Off-premises, SaaS	IBM owned and managed	IBM SoftLayer	Provides functionalities to define, secure, control, version, and publish API, and monitor API usage
IBM Bluemix API Management	Off-premises, SaaS	IBM owned and managed	IBM SoftLayer	Provides functionalities to define, secure, control, version, and publish API, monitor API usage, and share APIs with Bluemix developers
IBM Bluemix Dedicated API Management	Off-premises, SaaS	IBM owned and managed	IBM SoftLayer	Provides functionalities to define, secure, control, version, and publish API, monitor API usage, and share APIs with Bluemix developers

The rest of this section focuses on the IBM API Management Service in Bluemix.

When you create an instance of the API Management service in Bluemix, you are presented with the launch page shown in Figure 3-2 on page 27. From this launch page, you can launch and go to the API Manager.



Get started with API Management

You can design, publish, and manage APIs through the API Manager console.

Import APIs, or compose a new one.

Create a new plan, add resources and rate limits, and deploy.

Invite other Bluemix organizations to use your APIs.

Publish your plan.

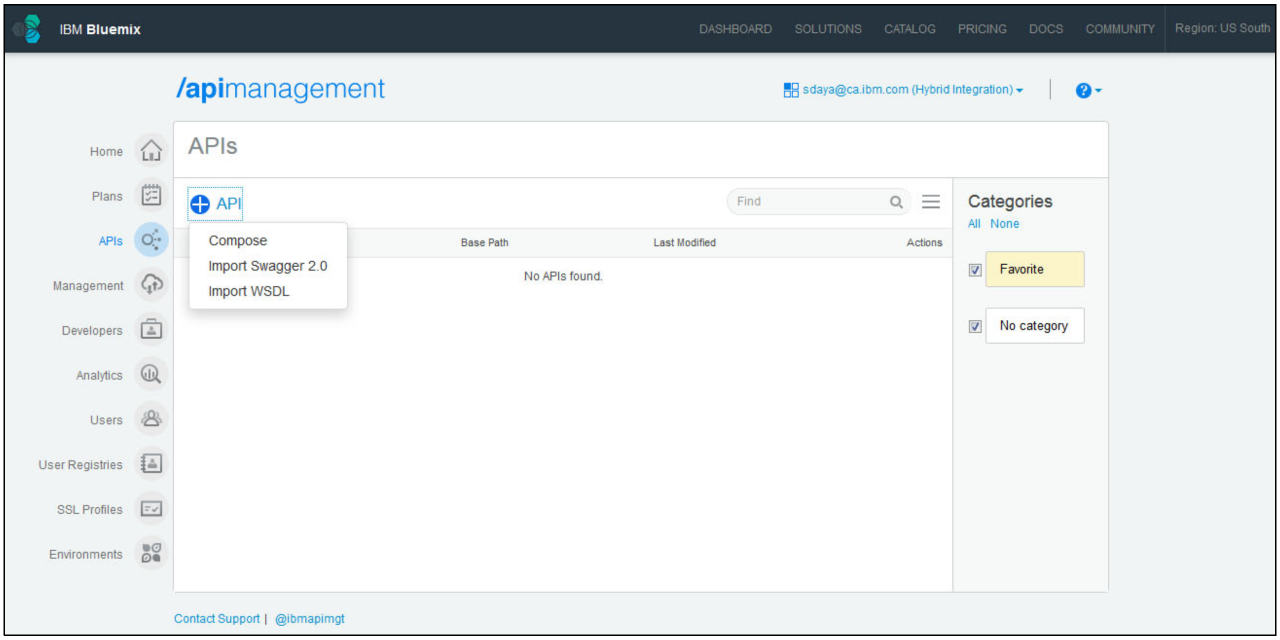
Application developers will be able to discover and consume your APIs from the Bluemix catalog.

Figure 3-2 API Management launch page

The API Manager is shown in Figure 3-3 on page 28. Following are the tasks that you typically need to perform to get started with API Manager:

1. Create an API
2. Create a plan
3. Add the API to the plan
4. Stage the plan
5. Test the API

Figure 3-3 on page 28 shows the APIs section of the API Manager. You see that to create an API, you can compose the API manually, or import a Swagger file representing a REST service, or import a Web Services Description Language (WSDL) file representing a SOAP-based web service. The links on the left enable you to navigate to other sections of API Manager.



No APIs found.

Figure 3-3 API Manager: Compose an API or import a Swagger or WSDL file

For illustrating the typical tasks and windows, we imported a Swagger file that represents an Airport Status service published by the Federal Aviation Administration (FAA). More details can be found at the following URL:

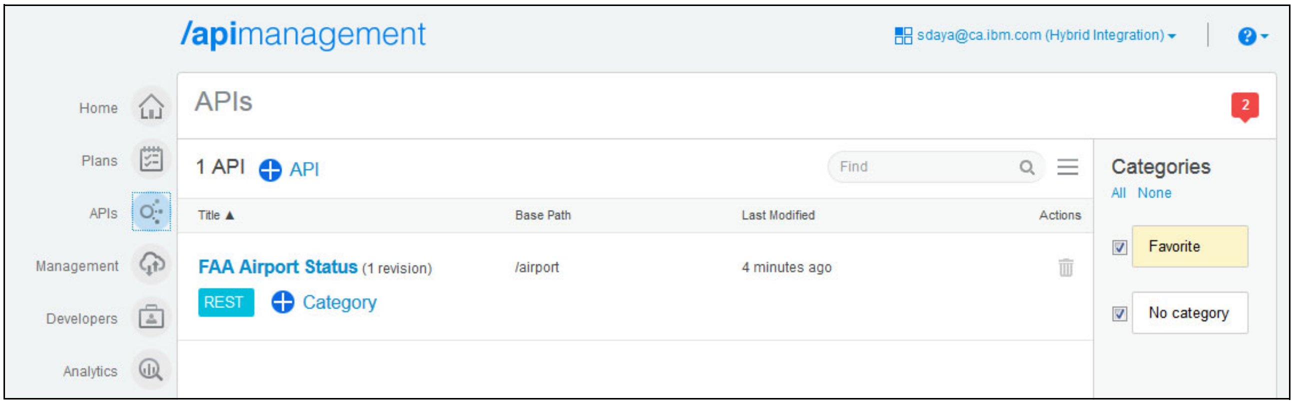
http://services.faa.gov/docs/services/airport/#airportStatus

Example 3-1 contains the Swagger file that we created. You can import this file and try it.

Example 3-1 Swagger file for FAA Airport Status service

```
{
    "swagger": "2.0",
    "info": {
        "title": "FAA Airport Status",
        "description": "",
        "version": "1"
    },
    "host": "services.faa.gov",
    "basePath": "/airport",
    "paths": {
        "/status/{iata}": {
            "get": {
                "summary": "",
                "operationId": "",
                "parameters": [
                    {
                        "default": null,
                        "description": "",
                        "name": "iata",
                        "required": true,
                        "type": "string",
                        "in": "path"
                    },
                    {
                        "default": "json",
                        "description": "",
                        "name": "format",
                        "required": true,
                        "type": "string",
                        "in": "query"
                    }
                ],
                "responses": {
                    "200": {
                        "schema": {},
                        "description": ""
                    }
                }
            }
        }
    },
    "definitions": {}
}
```

Figure 3-4 shows the APIs section of API Manager where the created APIs are listed.



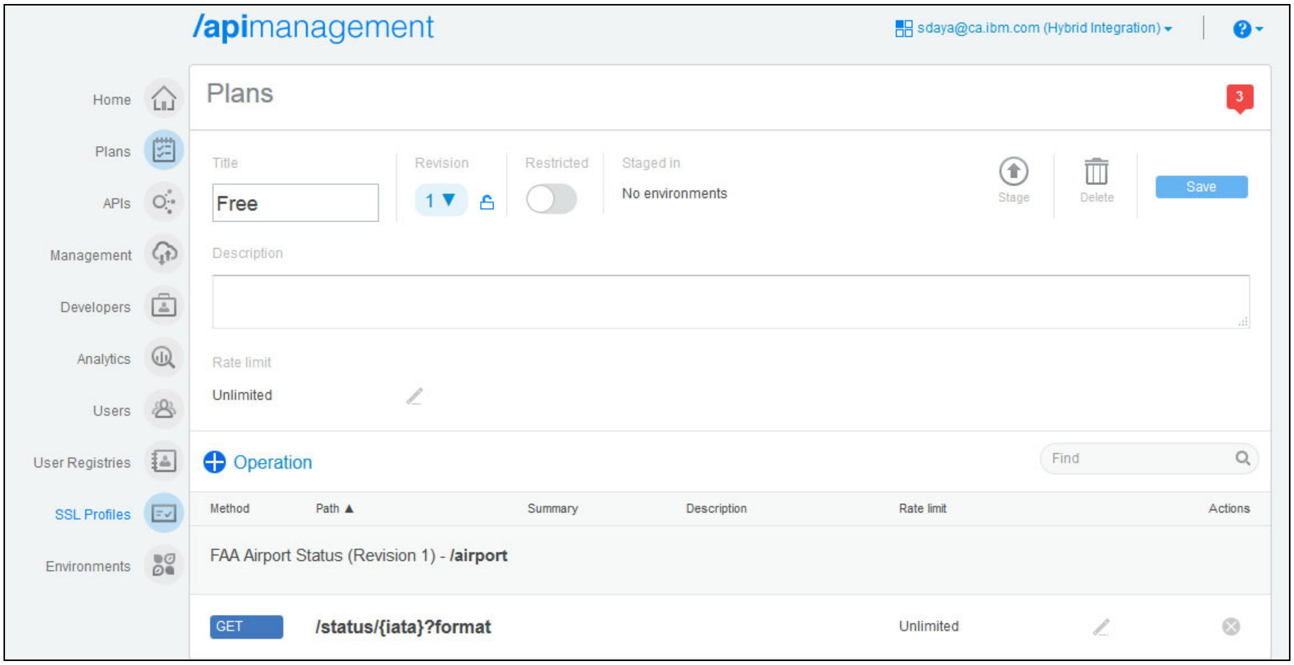
Title ▲	Base Path	Last Modified	Actions
FAA Airport Status (1 revision) REST Category	/airport	4 minutes ago	

Figure 3-4 API Manager: List of APIs

Before you can use an API, you need to add it to a plan. A plan treats APIs as product offerings, allowing several APIs and resources to be included in a plan. With a plan you can perform the following functions:

- Include multiple APIs and resources per plan
- Version your plans
- Apply entitlement by plan or resource
- Fine-grained control over plan deployment
- Enforce hard or soft limits

Figure 3-5 shows a *Free Plan* that has been created and the FAA Airport Status API has been added to that plan.



Plans

Title: Free

Revision: 1

Restricted

Staged in: No environments

Description

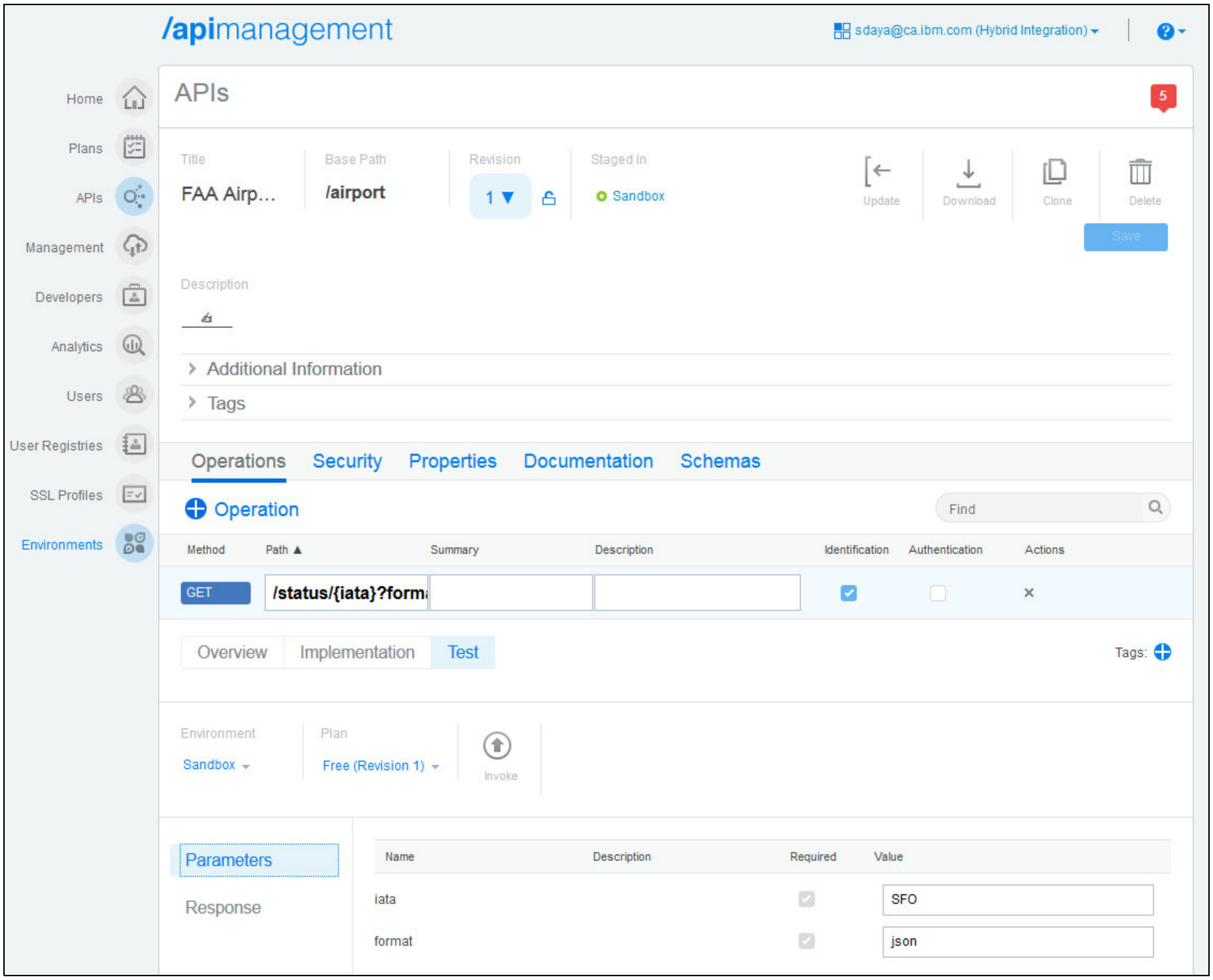
Rate limit: Unlimited

Method	Path ▲	Summary	Description	Rate limit	Actions
FAA Airport Status (Revision 1) - **/airport**					
GET	**/status/{iata}?format**			Unlimited	

Figure 3-5 API Manager: Created plan with API added

At this point, the plan is still not available for use. APIs and plans exist as authored artifacts visible in the API Manager. To become available to consumers, APIs must be deployed to an environment, and published to some or all organizations. An environment has an associated runtime capability. By default, a provider organization has a sandbox environment. Apps are registered to consume APIs via a selected plan, which determines the API quota. The plan needs to be staged into an environment.

When a plan is staged in an environment, you can test and start using the APIs that are included in the plan. You need to edit the API to access the test feature. In edit mode, you can change any configuration, such as security or add additional documentation for consumers of the API. Figure 3-6 on page 31 shows the Free Plan staged in the default *sandbox* environment.



Title: FAA Airp…
Base Path: /airport
Revision: 1
Staged in: Sandbox

Method	Path ▲	Summary	Description	Identification	Authentication	Actions
GET	/status/{iata}?form			☑	□	×

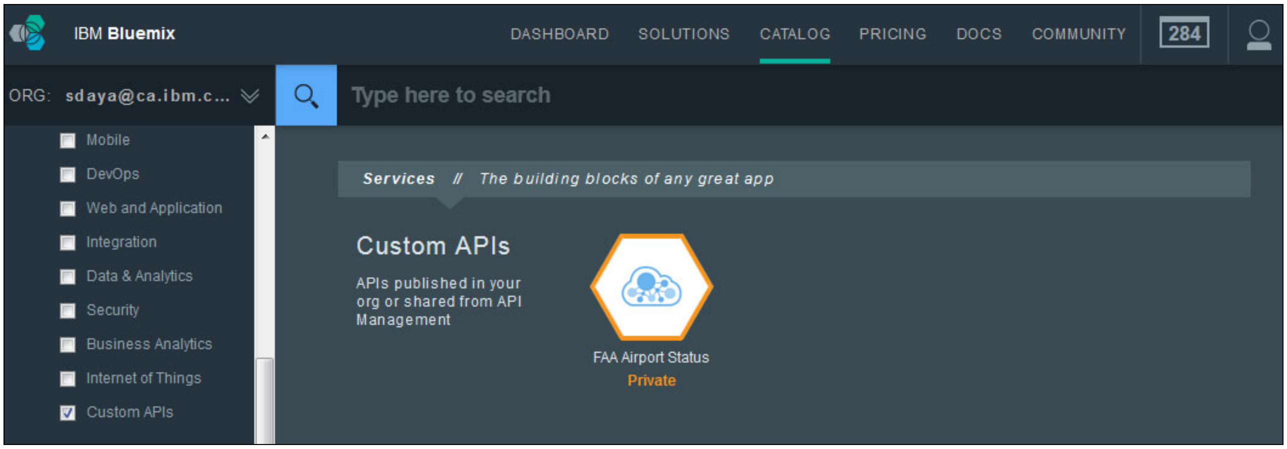
Environment: Sandbox
Plan: Free (Revision 1)

Name	Description	Required	Value
iata		☑	SFO
format		☑	json

Figure 3-6 API Manager: Testing an API

From an API security standpoint, you have two aspects. First, the identity of the application that is consuming the API, and second, the identity of the users of that application that is consuming the API. The identity of the application can simply be established by using a client ID or by using a client ID along with a client secret. For authentication/authorization of users of the application, basic authentication or OAuth can be configured.

The API is now also available as a Custom API in the Bluemix Catalog as shown in Figure 3-7. The API can now be instantiated, bound to an application, and used with the same developer experience as any of the other services in the Bluemix Catalog.



APIs published in your org or shared from API Management

Figure 3-7 APIs shared from API Management appear as Custom APIs in the Bluemix Catalog

Following are some common use cases for API Management:

- Provide omni-channel access to business information for accelerating internal app development.
- Collaborate with business partners faster, in an open but secure and managed way, while providing a complete self-service experience.
- Power Mobile apps with enterprise business logic to innovate and provide high value to customers: Digital transformation focus.
- Power Internet of Things (IoT) apps with enterprise information to drive innovation: Digital transformation focus.
- Centrally manage the consumption of business logic, across the enterprise, for both Systems of Record and Systems of Engagement.
- Publish APIs publicly to drive innovation, tap into broad developer ecosystem, and promote brand.
- Extend brand reach from Systems of Record to bridge to Systems of Engagement.
- Provide secure composite services in the cloud.
- Provide managed access to third-party cloud services to app dev teams to achieve centralized governance and cost optimization.
- Enable new business channels by monetizing enterprise data.

5 Things to Know about API Management in Bluemix blog post: For more information about API Management, check out *5 Things to Know about API Management in Bluemix*:

https://www.ibm.com/developerworks/community/blogs/5things/entry/5_Things_to_Know_about_API_Management_in_Bluemix?lang=en

3.1.2 API-centric service: Secure Gateway

The IBM Secure Gateway service in Bluemix provides a secure way to access your on-premises or cloud data from your application running in Bluemix over a secure passage that is the gateway. The basic scenario for Bluemix to on-premises integration is the integration between the new born on the cloud Systems of Engagement (SOE) and the existing legacy Systems of Record (SOR). Data is typically accessed from a SOR, such as a database or an application/web service.

The Secure Gateway works by using a client on the on-premises side to connect to your Bluemix organization, as shown in Figure 3-8.

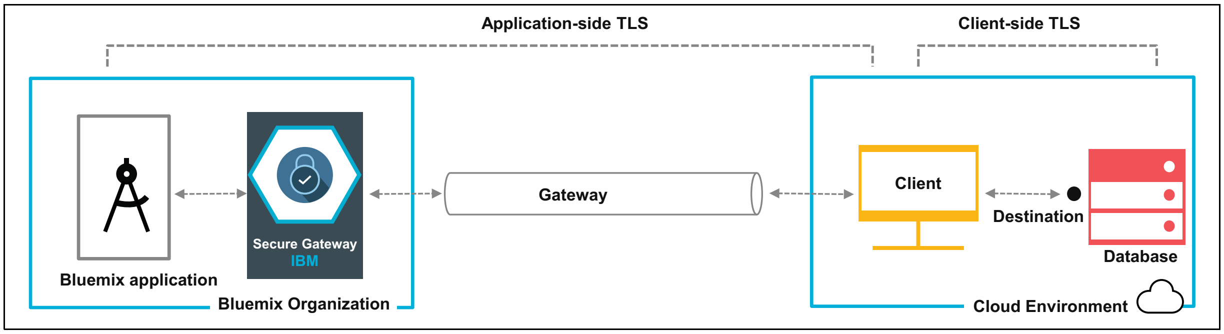


Figure 3-8 Secure Gateway high-level architecture overview

Table 3-2 defines some key terms.

Table 3-2 Key terms for Secure Gateway

Term	Definition
Client	The process that establishes the on-premises or cloud side of the gateway and forwards requests to the destinations.
Gateway	The tunnel between your Bluemix app and on-premises or cloud environment.
Destination	The point at which your on-premises data can be accessed.
Application-side TLS	Security between your Bluemix app and on-premises or cloud client.
Client-side TLS	Security between the client and on-premises or cloud destination/data.

The basic steps to set up and use the Secure Gateway service are as follows:

1. Provision a Secure Gateway service and bind it to your application.
2. Create a gateway (Name It).
3. Connect the gateway to a client (Connect It).
4. Add a destination to the gateway (Add Destinations).
5. Use the destination in your application.

Note: You can provision only one Secure Gateway service per space. You can have multiple Bluemix applications bind to the same Secure Gateway instance.

The gateway contains the configuration information for establishing a tunnel between the Secure Gateway client running in your environment and Bluemix. When adding a gateway, you can choose to enforce increased security over who is able to start a gateway. An optional security token can be provided when they connect the Secure Gateway client.

Figure 3-9 shows the Add Gateway window of the Secure Gateway showing the three steps to set up the gateway and destination.

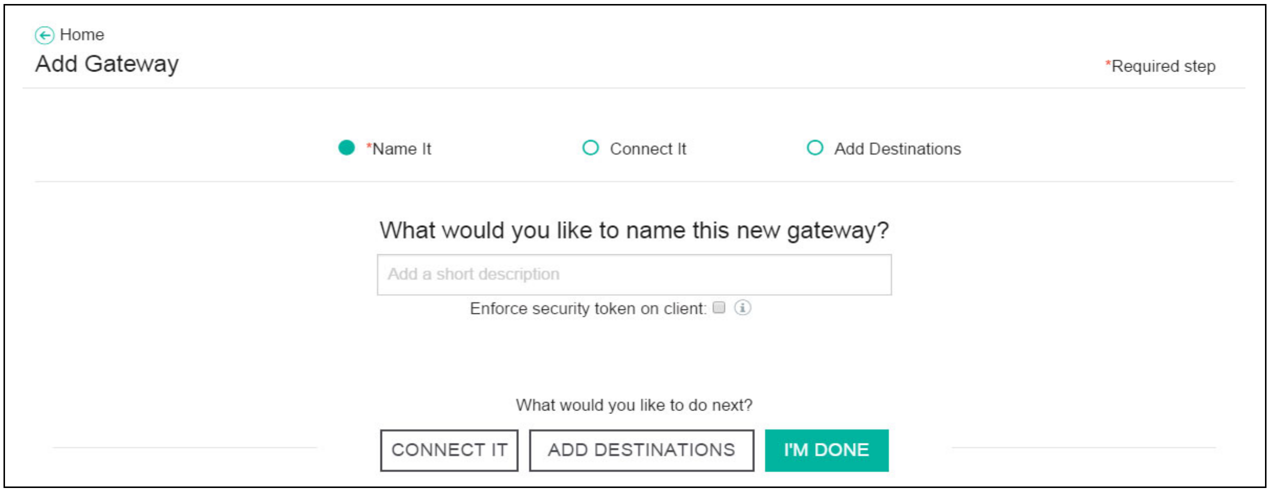


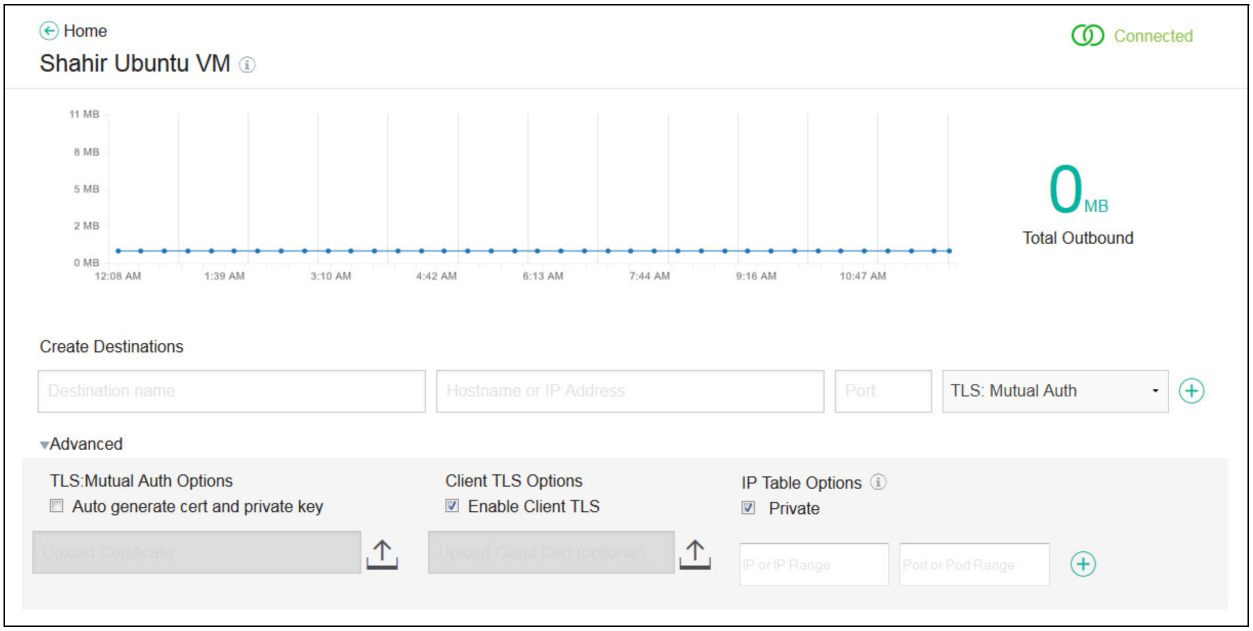
Figure 3-9 Add Gateway window

When you name your gateway, you can connect your gateway to a client. At the time of this writing, the following two options were available for the client:

- A Docker container running the secure gateway client. Docker provides a convenient *run anywhere* option.
- IBM DataPower providing an appliance-optimized solution with the same base features as the Docker client, but with additional security enforcement capabilities.

The gateway client is a process that runs in the on-premises or cloud side of the gateway. It has network visibility to the Bluemix application and to the destinations. Multiple destinations might need multiple clients. The gateway client initiates a connection with the gateway in Bluemix and forwards requests from the gateway to the destinations.

When you connect your gateway to a client using one of these listed clients, you can add a destination to your gateway. Figure 3-10 on page 35 shows the window to create a destination. The destination specifies how to connect to the system resource that is a SOR, database of record, and so on. You need only one destination per system resource. The destination consists of a name, host name or IP address, the port, and protocol.



Shahir Ubuntu VM

0 MB Total Outbound

Create Destinations

TLS: Mutual Auth

▾Advanced

TLS:Mutual Auth Options
□ Auto generate cert and private key

Client TLS Options
☑ Enable Client TLS

IP Table Options
☑ Private

Figure 3-10 Create Destination window

Each destination can optionally use Transport Layer Security (TLS) protocol:

- Application-side TLS

  Secures access between the Bluemix app and the cloud environment client.

- Client-side TLS

  Secures access between the cloud environment client and the destination.

- The two can be set independently

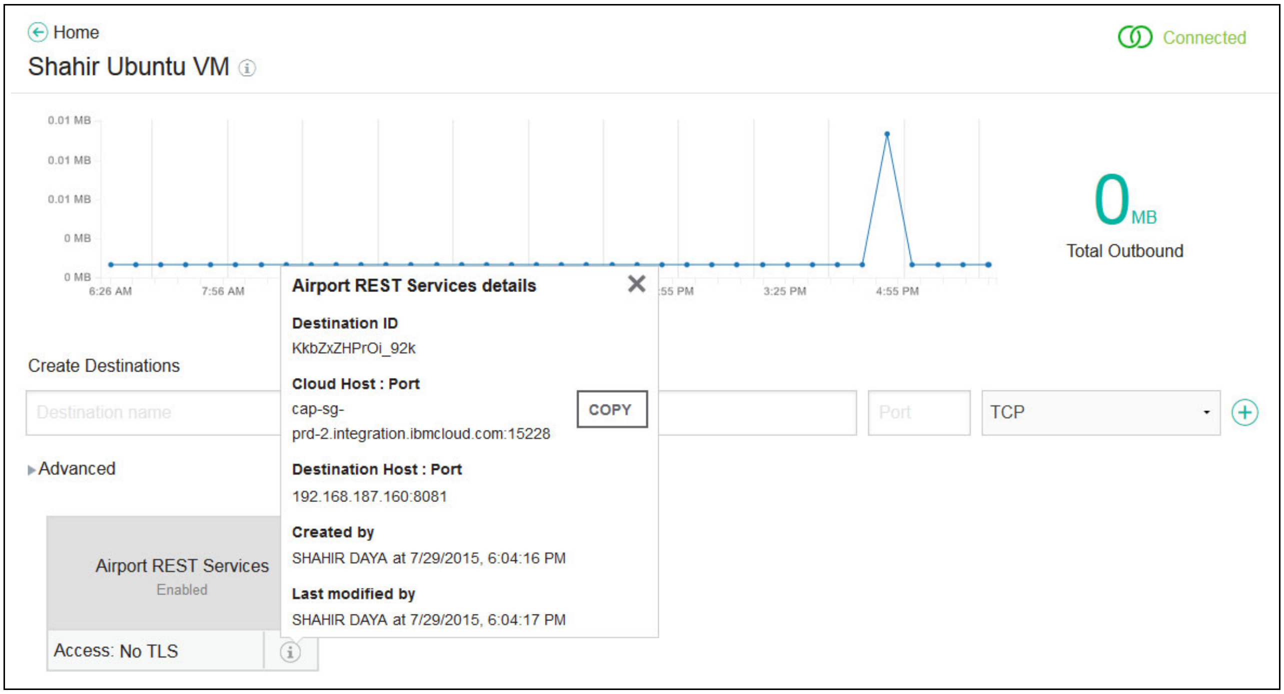
For application-side security, the following protocol options apply:

- TCP
  - No TLS: No certificates, no encryption
  - No authentication is provided
  - Bluemix application communicates directly to the gateway
- TLS Server Side
  - TLS is enabled
  - Secure gateway generates a certificate to prove its authority
  - You need to accept the server certificate into your Bluemix application truststore
- TLS Mutual Auth
  - Option 1: Auto-generate certificate and private key
  - Option 2: Upload existing keys to the gateway
- HTTP
- HTTPS

For client-side security, you can Enable Client TLS. Client TLS is required for connecting to an HTTPS backend. The destination’s certificate is verified against known certificate authorities. If the certificate (PEM) is self-signed, it must be attached to the cloud environment client. Attaching the certificate can be done during the creation and edit of the destination or via the Secure Gateway REST API.

Additionally, the IP Table Options can be used to limit the IP addresses or ports that can access the destination. The IP or port number entered into the IP table must be the external IP address that the Secure Gateway server sees, not the local IP address of the machine.

Figure 3-11 shows the details of a destination. The Cloud Host:Port is used by the application to connect securely to the destination.



Airport REST Services details

Destination ID
KkbZxZHPrOi_92k

Cloud Host : Port
cap-sg-prd-2.integration.ibmcloud.com:15228

Destination Host : Port
192.168.187.160:8081

Created by
SHAHIR DAYA at 7/29/2015, 6:04:16 PM

Last modified by
SHAHIR DAYA at 7/29/2015, 6:04:17 PM

Figure 3-11 Destination details contain Cloud Host and Port

Figure 3-12 shows a typical simple use case. An SOE web application developed in Java Platform, Enterprise Edition runs in a Java Liberty run time in Bluemix. The web application needs to access data that is stored in a MySQL database that is hosted in a corporate data center. In this case also, the Secure Gateway service is used to securely connect from the Public Bluemix to the corporate data center to enable making JDBC calls over a secure channel.

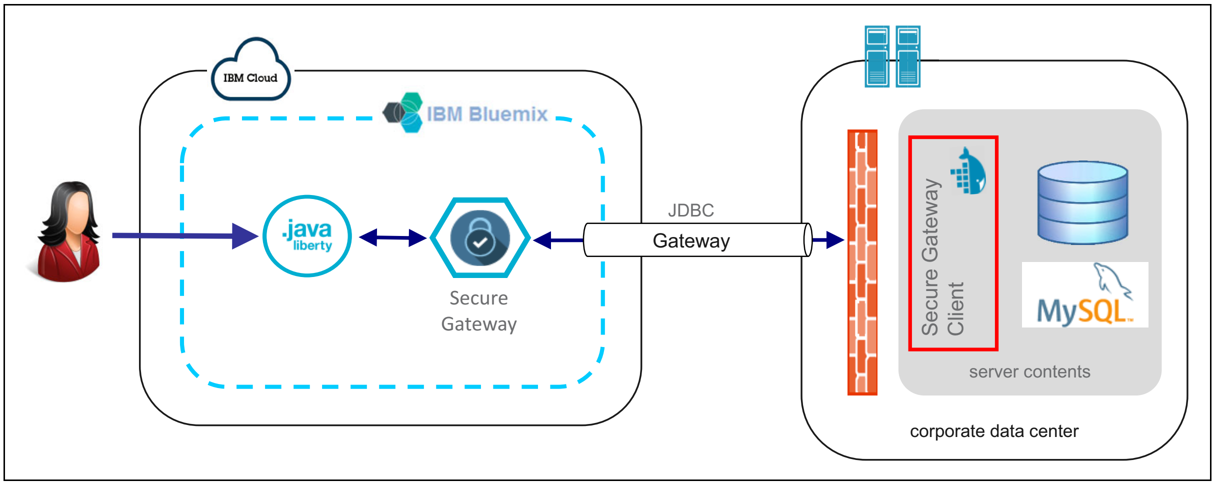


Figure 3-12 Bluemix SOE web application requiring integration with an on-premises relational database

Figure 3-13 shows a more complex use case for the Secure Gateway. An SOE mobile application uses Bluemix as the Mobile Backend as a Service (MBaaS). An IBM Cloudant® NoSQL DB service provides data storage for the mobile application. A NodeJS run time provides the necessary orchestration of service calls to fulfill the needs of user-interface interactions. The mobile application needs to call a SOAP-based web service that resides in the corporate data center. The Secure Gateway service is used to securely connect from the Public Bluemix to the corporate data center. SOAP requests go through the Secure Gateway over the secured channel.

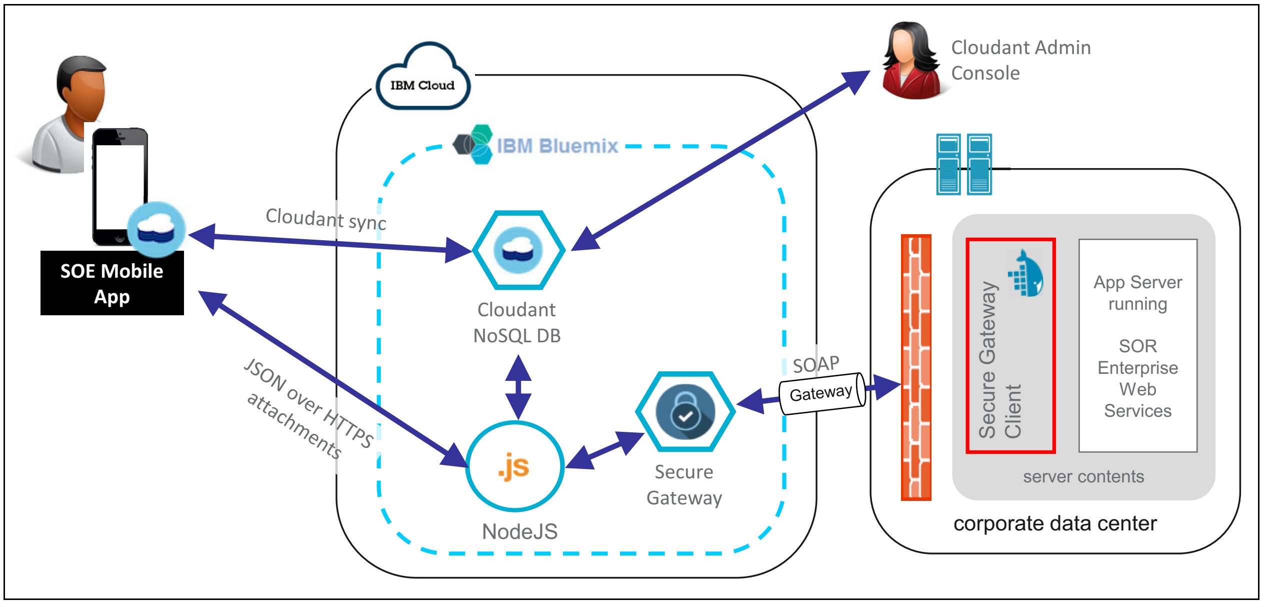


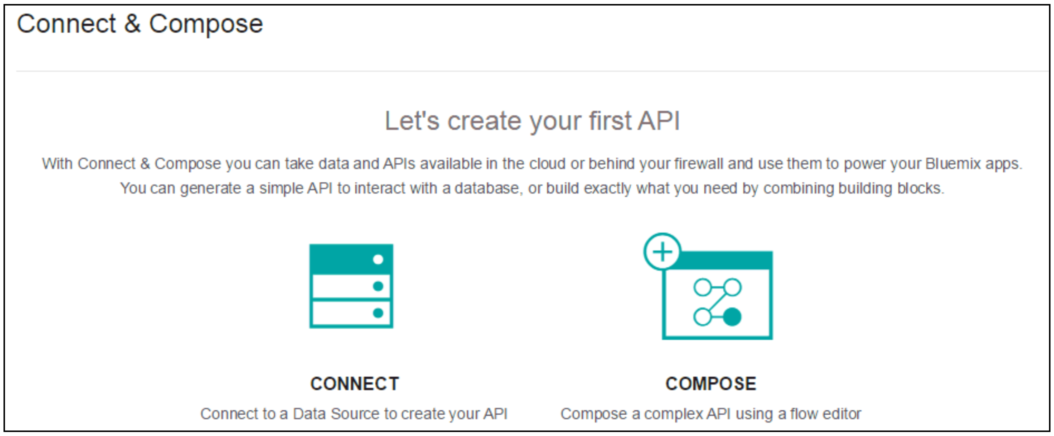
Figure 3-13 Bluemix based MBaaS requiring integration with a SOAP-based on-premises web service

In summary, the Secure Gateway service brings hybrid integration capability to your Bluemix environment. It provides secure connectivity from Bluemix to other applications and data sources running on-premises or in other clouds.

3.1.3 API-centric service: Connect & Compose

Connect & Compose is a new service offering in Bluemix for API creation. As shown in Figure 3-14 on page 38, the service offers to create API in two ways:

- Connect to and create a Representational State Transfer API to interact with a single source of data or service.
- Compose by using a flow editor and API that performs complex functionalities in each API call, such as relay further requests to backend services, sending email, interacting with persistence, and various third-party services.



Connect & Compose

Let's create your first API

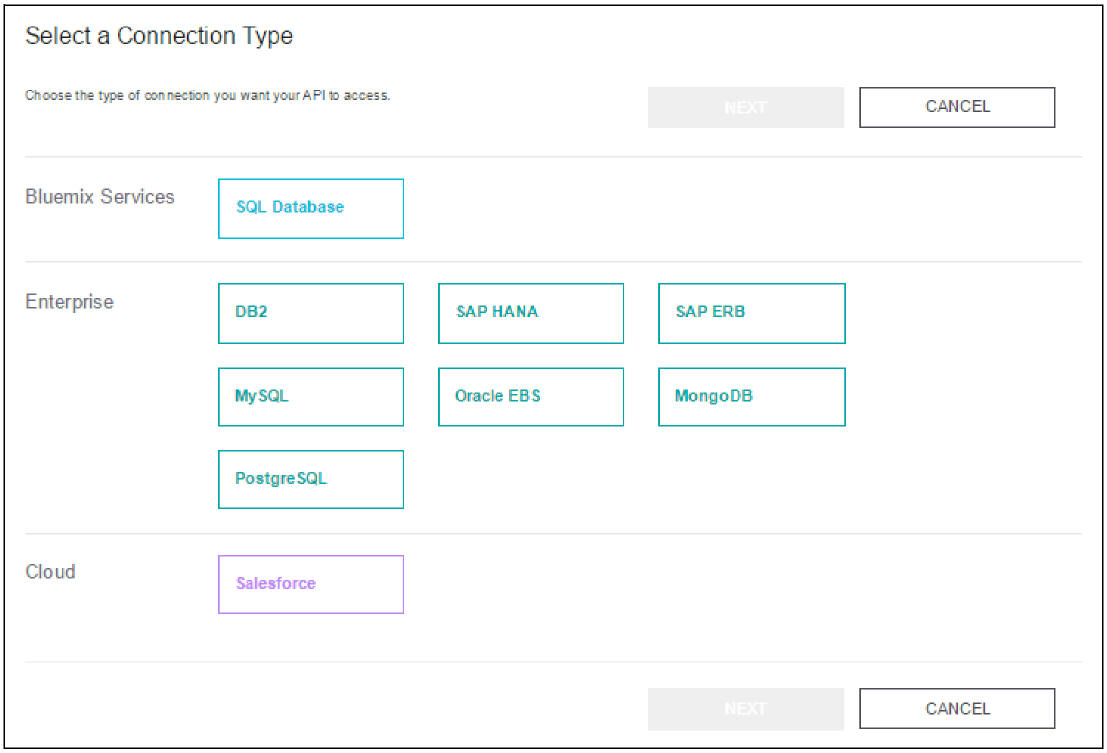
With Connect & Compose you can take data and APIs available in the cloud or behind your firewall and use them to power your Bluemix apps. You can generate a simple API to interact with a database, or build exactly what you need by combining building blocks.

CONNECT
Connect to a Data Source to create your API
COMPOSE
Compose a complex API using a flow editor

Figure 3-14 Connect & Compose landing

Connect

The Connect option in Connect & Compose provides a RESTful interface to interact with a growing option of data source and services in Bluemix, on-premises, and on cloud. Figure 3-15 shows supported sources as of this publication.



Select a Connection Type

Choose the type of connection you want your API to access.

Figure 3-15 Connector options at different locations

To start, the user selects where and the type of source to connect to. For databases on Bluemix, it is required that the service to be provisioned and bound to your app, before Connect & Compose can connect to it. Similarly, Secure Gateway service is required to access enterprise sources behind a corporate firewall or in a secured network.

With successful connection to a data source, the user then selects the model to which the API acts on. Depending on the type of source that the user is connecting to, Connect & Compose allows users to exclude optional parameters and REST endpoints during the API creation process. This is a valuable feature when the user only wants to externalize non-destructive endpoints. A sample swagger is shown for the user to check the API formation before save. With the host and port provided by Connect & Compose after creation, the user can now interact with the API via REST calls.

Compose

The more robust option to create an API with complex functionality is with the Compose option. API composition is delegated to a Node-RED composition interface. The interface allows user to compose an API by dragging and connecting wanted nodes. Figure 3-16 is a sample flow how an API can be composed to perform multiple functions within one call (extra payload handling function nodes are required to wrap services nodes in most cases shown, but are eliminated to provide a generic concept). All four APIs initiate in an http request, and ends with an http response:

- First API endpoint accepts a POST request with location information, retrieves Google Places data, checks the weather, and pushes the result to email, twitter, and pinterest.
- Second API endpoint accepts a GET request. The function node can provide adjustments, such as putting query parameters into payload, or adjusting and reformatting input values. The adjustment is then relayed to the SAP node to invoke a function that has been configured. The returned data is then uploaded to Dropbox, and submitted for debugging at the same time.
- Third API endpoint accepts a GET request and passes the information to a bus schedule service, and adds the information to Google Calendar.
- Fourth API endpoint demonstrates a chain of http requests, which ends with persisting the final result into an SAP Hana database.

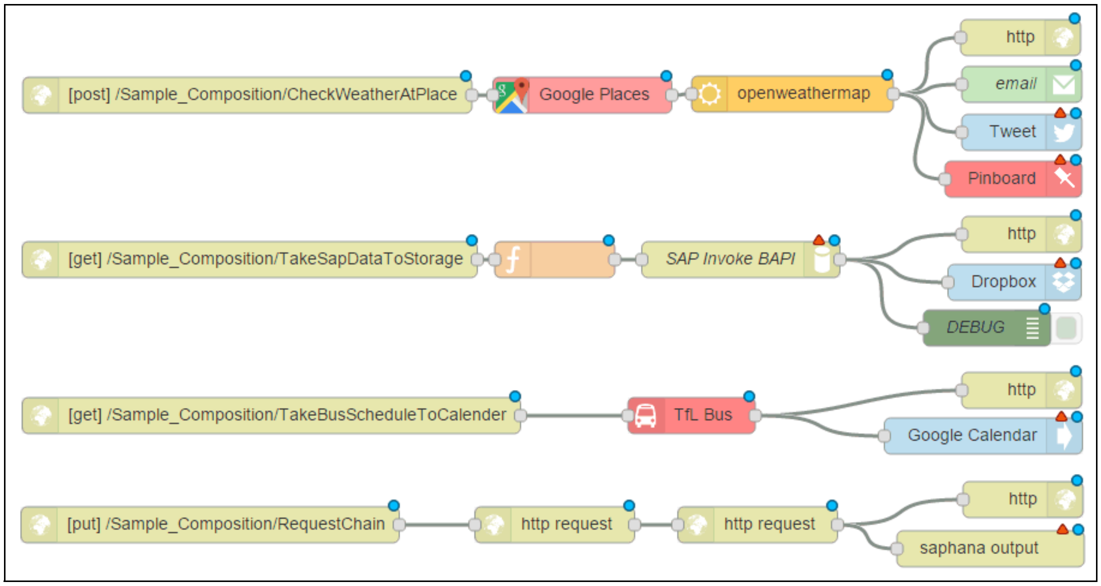
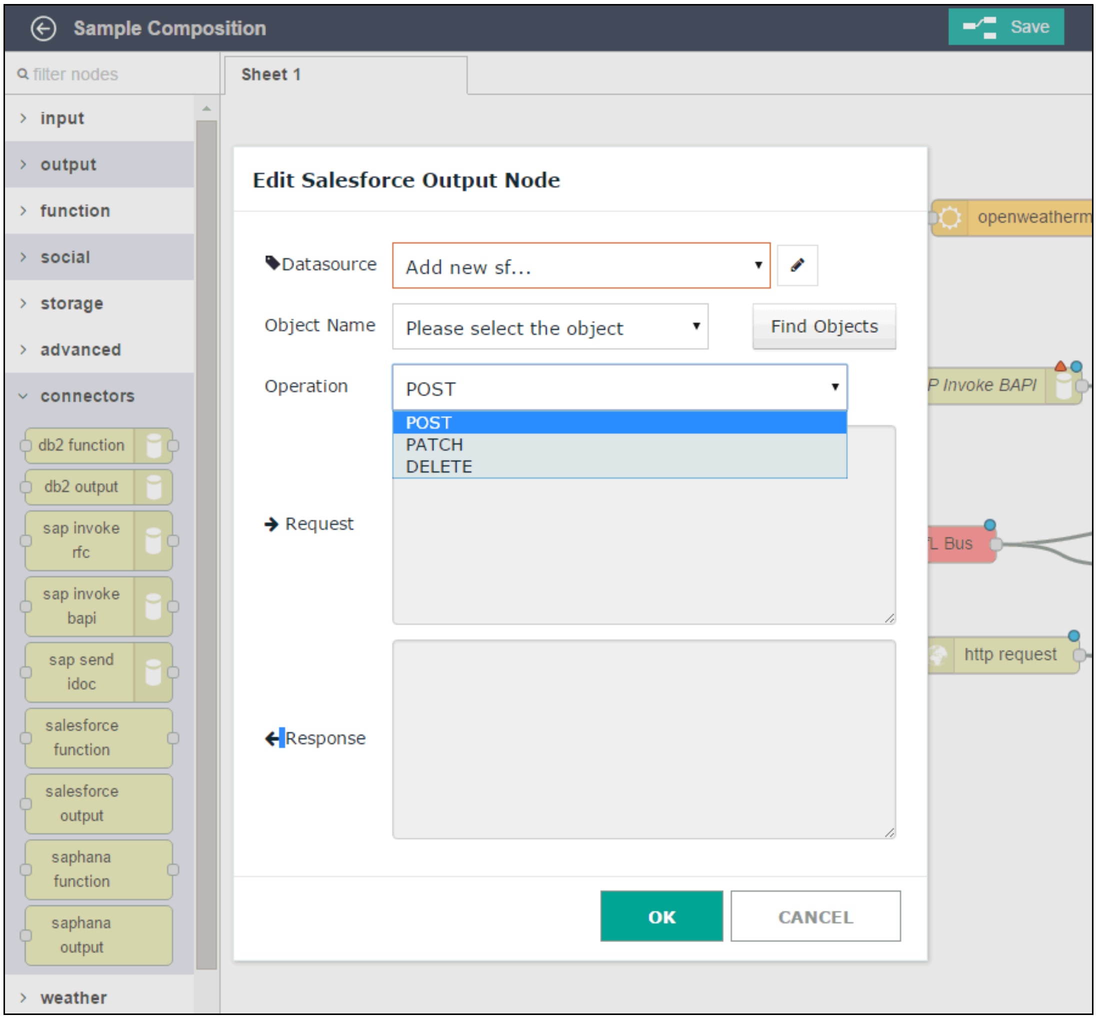

[post] /Sample_Composition/CheckWeatherAtPlace
Google Places
openweathermap
http
email
Tweet
Pinboard
[get] /Sample_Composition/TakeSapDataToStorage
SAP Invoke BAPI
http
Dropbox
DEBUG
[get] /Sample_Composition/TakeBusScheduleToCalender
TfL Bus
http
Google Calendar
[put] /Sample_Composition/RequestChain
http request
http request
http
saphana output

Figure 3-16 Sample Composition with various types of nodes

As long as there is a node to support it, there is also no limitation on what sources to act on in the composition flow editor, which distinguishes it from the Connect counterpart. Figure 3-17 shows a sample node called *salesforce output*. This node provides the functionality to interact with data stored on Salesforce database. Notice also in the left panel shown in Figure 3-17, Salesforce-related nodes come in two options. The *salesforce output* node performs POST, PATCH, and DELETE operations, and provides no output, does end the flow. The *salesforce function* node alternatively, performs GET operations, and provides output that can be passed on to the next node in the flow diagram.



Edit Salesforce Output Node

Datasource: Add new sf...

Object Name: Please select the object

Operation: POST

Figure 3-17 Configuration for salesforce output node

Table 3-3 shows the categories and their respective nodes available as of the time of this publication.

Table 3-3 Category of nodes

Category	Nodes
input	inject, catch, http, websocket
output	debug, http response, websocket
function	function, template, delay, trigger, comment, http request, switch, change, range, csv, html, json, xml, rbe

Category	Nodes
social	email, twitter, delicious, foursquare, swarm, google plus, google places, google calender, instagram, pinboard
storage	amazon s3, box, dropbox
advanced	feedparse
connectors	db2, sap, salesforce, saphana
weather	forecastio, openweathermap, wunderground
location	google geocoding, google directions
Google	google calender
transport	tfl underground, tfl bus

Using APIs

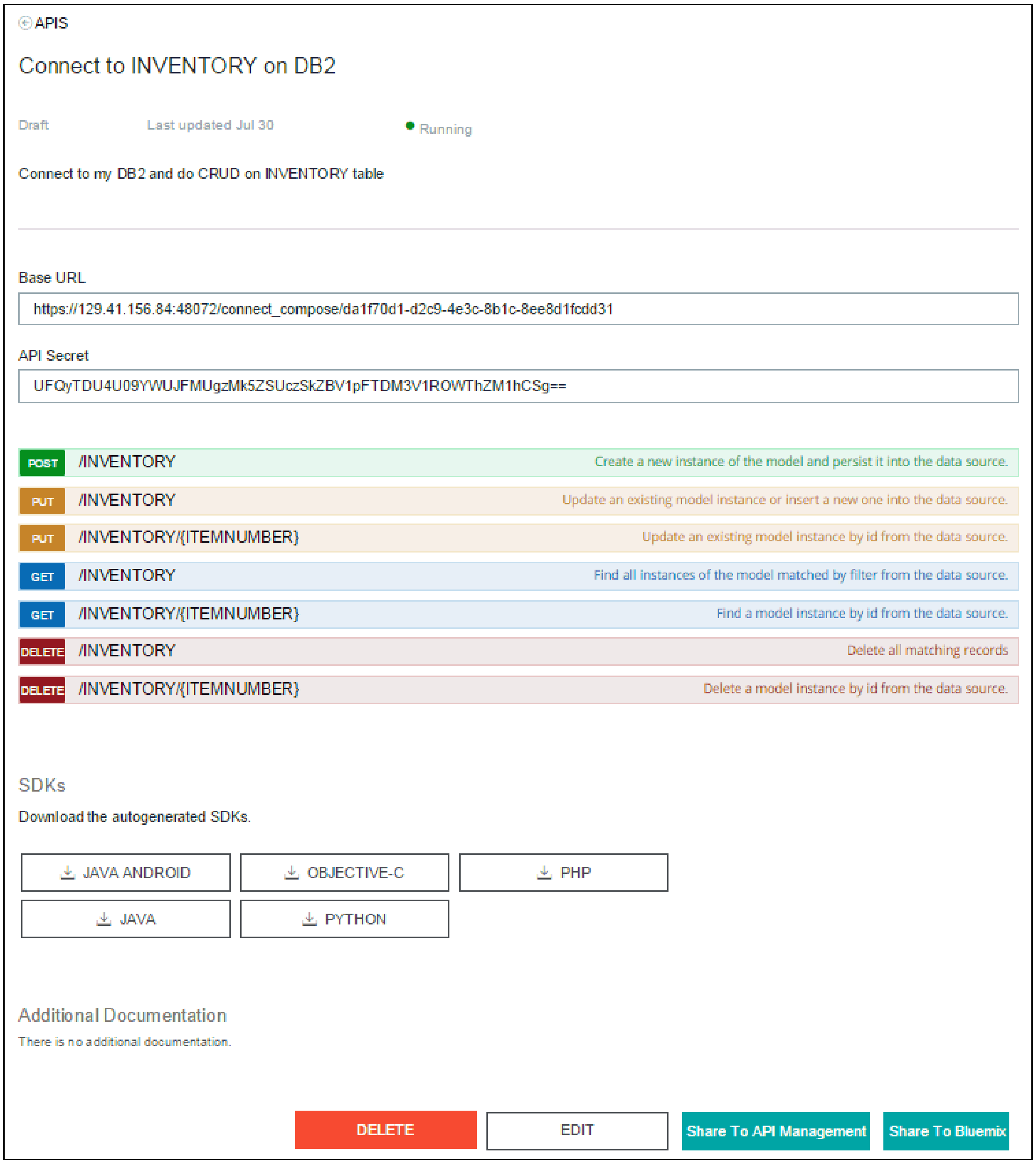
When an API is created in Connect & Compose, the API becomes immediately available on the host and port provided. Figure 3-18 on page 42 shows detailed information about the new API.

The top section of the page displays the general information about the API, including shared name, state, time stamp, and API running status. The running status is only an indicator of whether the API is available. The user is responsible to keep all connections from the API to the data source (such as Secure Gateway, virtual private network, and the data source itself) running.

The second section of the API details page shows the base Uniform Resource Locator for the API, and the access key to include in the header when sending requests to the API.

The third section includes an interactive Swagger UI that allows the user to visualize and test the API.

The fourth section provides SDK package downloads for ease of development in different languages. If the user uploaded more documentation for this API during the creation process, they are also displayed for download.



Connect to INVENTORY on DB2

Draft   Last updated Jul 30   ● Running

Connect to my DB2 and do CRUD on INVENTORY table

Base URL

https://129.41.156.84:48072/connect_compose/da1f70d1-d2c9-4e3c-8b1c-8ee8d1fcdd31

API Secret

UFQyTDU4U09YWUJFMUgzMk5ZSUczSkZBV1pFTDM3V1ROWThZM1hCSg==

POST	/INVENTORY	Create a new instance of the model and persist it into the data source.
PUT	/INVENTORY	Update an existing model instance or insert a new one into the data source.
PUT	/INVENTORY/{ITEMNUMBER}	Update an existing model instance by id from the data source.
GET	/INVENTORY	Find all instances of the model matched by filter from the data source.
GET	/INVENTORY/{ITEMNUMBER}	Find a model instance by id from the data source.
DELETE	/INVENTORY	Delete all matching records
DELETE	/INVENTORY/{ITEMNUMBER}	Delete a model instance by id from the data source.

SDKs

Download the autogenerated SDKs.

Additional Documentation

There is no additional documentation.

Figure 3-18 Sample API created from connecting to DB2

For advanced utilization of the API, user has the option to share to Bluemix, share to API Management, or both. When an API created in Connect & Compose is shared to Bluemix, the user obtains the ability to bind the API to an app in the same space. The API detail is also available for all users in the Bluemix Space to see. Alternatively, sharing the created API to API Management allows the API to be managed with more policies and all the functionalities that API Management has to offer.

Important: At the time of writing this book Connect & Compose was not publicly available, so some of this information might slightly change when the service becomes generally available.

3.1.4 Data-centric service: IBM DataWorks

IBM DataWorks is a service on Bluemix that can be used for data load, migration, refinement, transformation, and gaining insights from the data. As depicted in Figure 3-19, DataWorks helps the IT team, developers, and business users to get access to meaningful and appropriate data and provides the tools and APIs to perform the jobs. Data can be collected from various sources (web, mobile, social media, enterprise systems, document repository, IoT, and so on) and processed by DataWorks for use by various applications as shown in the diagram.

IBM DataWorks provides useful data services, such as:

- Load data
- Provision masked data
- Profile data
- Classify data
- Secure on-premises load to cloud targets and cleanse addresses

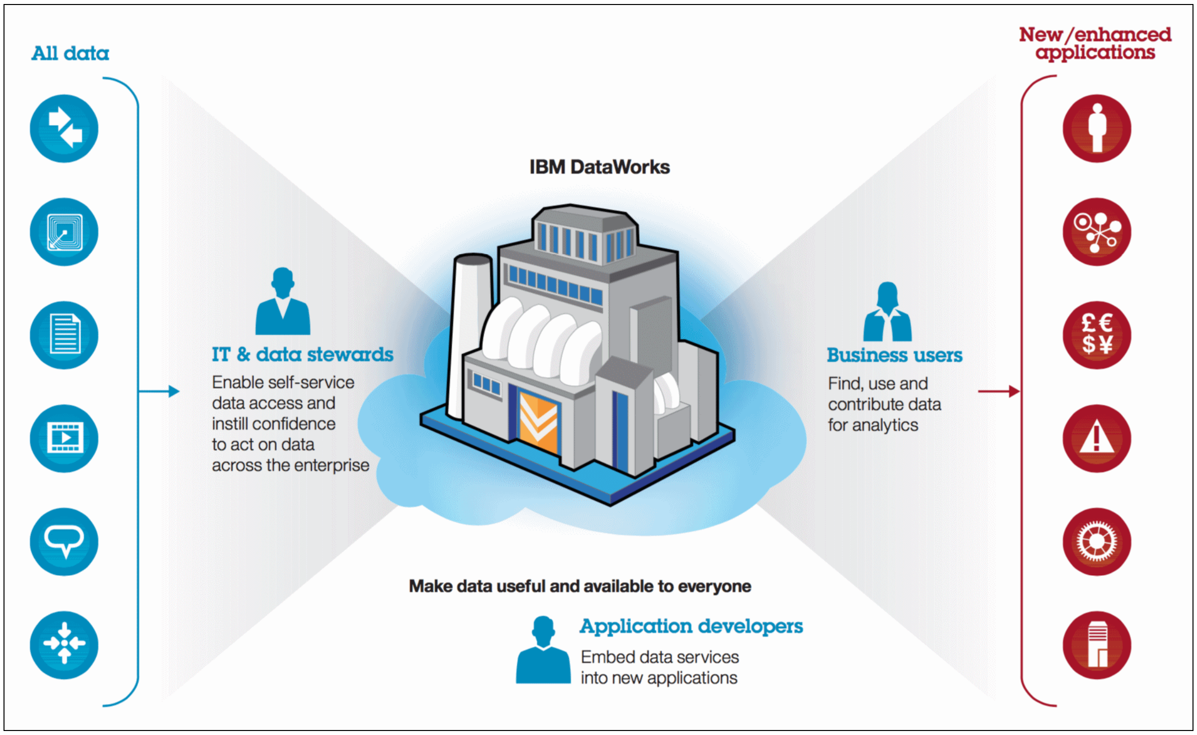

All data
New/enhanced applications
IBM DataWorks
IT & data stewards
Enable self-service data access and instill confidence to act on data across the enterprise
Business users
Find, use and contribute data for analytics
£€ $¥
Make data useful and available to everyone
Application developers
Embed data services into new applications

Figure 3-19 IBM DataWorks providing data services, capabilities, and insights to IT team, developers, and business users

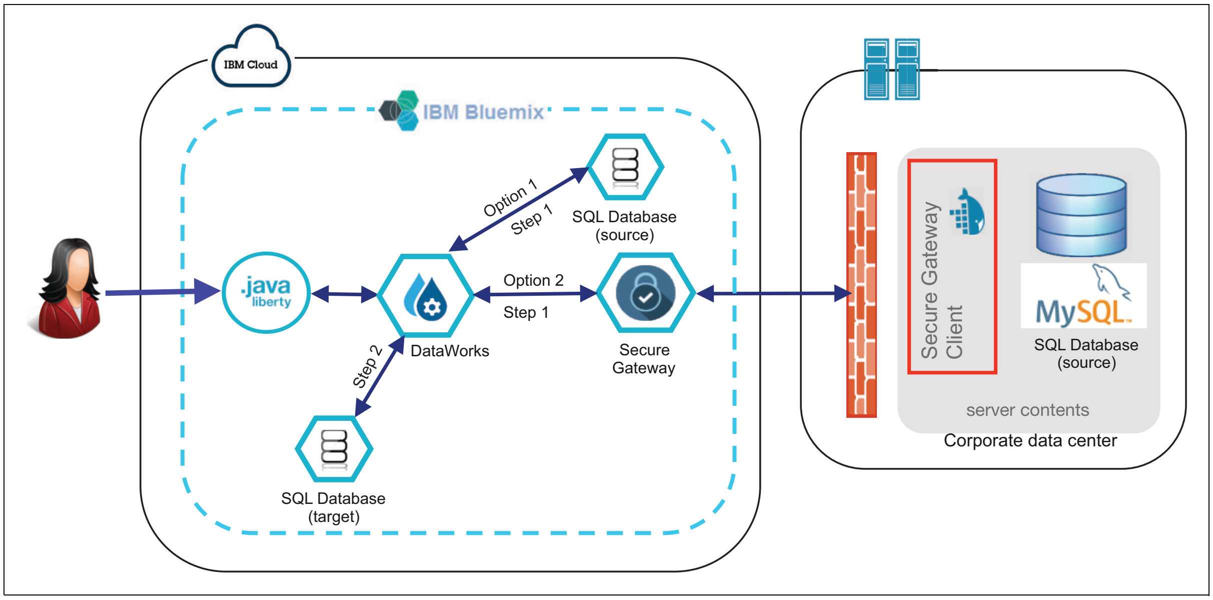


Figure 3-20 DataWorks use case in combination with Secure Gateway to access an on-premises database or without a Secure Gateway to access a public database

Figure 3-20 shows conceptually how DataWorks service can connect a user or an application to a public database or an enterprise database behind a firewall. Although the datagram indicates a Java application, the use case is not restricted to an application of a particular type. It can be an application using any programming language. A developer can use DataWorks Forge to get data from one data source (for example, SQL DB), shape the data and finally put the refined data in another database (maybe another SQL DB). An application can use the DataWorks APIs to load data, cleanse address, and profile the data. The two options in terms of the presence of a Secure Gateway are:

- Option 1

  In this scenario, an application or a user is connected to a public database (for example, SQL service hosted on Bluemix) through DataWorks. DataWorks service provides the APIs or DataWorks Forge manipulates the data (step 1) and then copies data from source to target (step 2). The data in the target is ultimately used by the application.

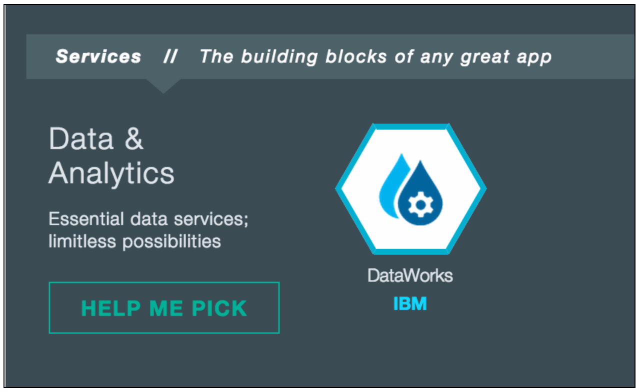
- Option 2

  In this scenario, an application or a user is connected to an on-premises database (for example, MySQL as shown in the diagram) through DataWorks. DataWorks service provides the APIs or DataWorks Forge manipulates the data (step 1) and then copies data from source to target (step 2). The data in the target is ultimately used by the application.

DataWorks service is composed of two offerings: IBM DataWorks Forge and DataWorks APIs.

IBM DataWorks Forge

IBM DataWorks Forge service on Bluemix can be used to find, shape, or transform data and deliver the refined data to applications and systems for seamless integration. A series of screen captures in this section gives you an overview of this service. For detailed step-by-step instructions, refer to Chapter 8, “Watson Analytics in hybrid cloud using Secure Gateway and DataWorks” on page 195. Developers can create a DataWorks service instance by choosing the DataWorks service from Bluemix catalog, as shown in Figure 3-21 on page 45.



Services // The building blocks of any great app

Data & Analytics

Essential data services; limitless possibilities

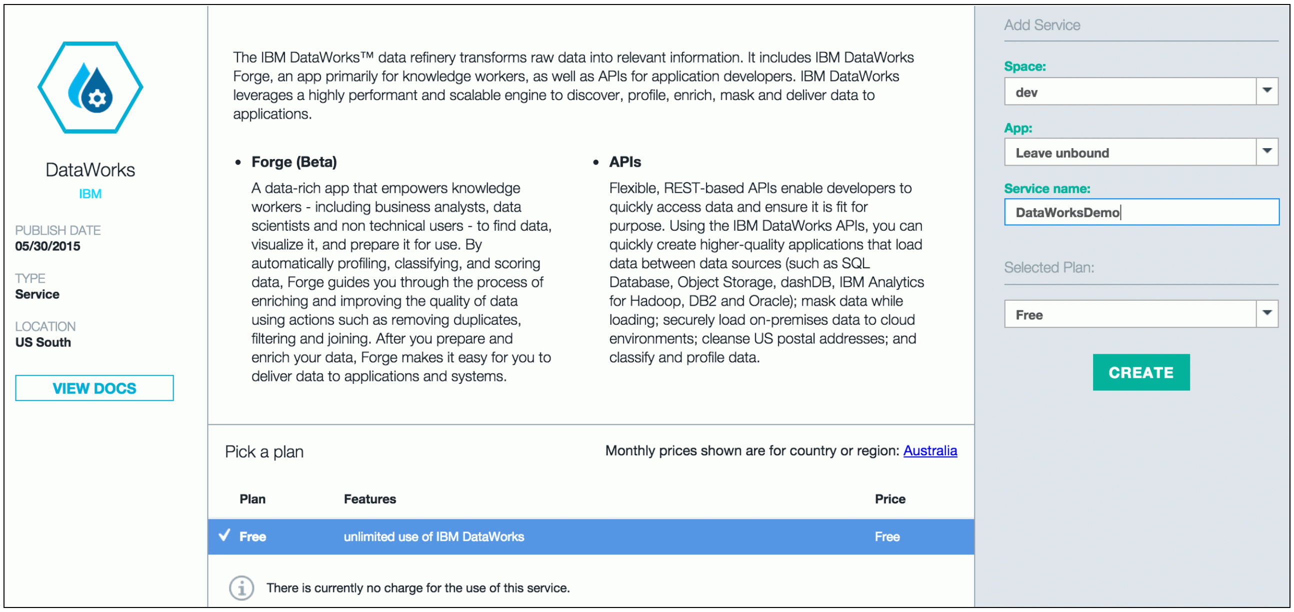
HELP ME PICK

DataWorks
IBM

Figure 3-21 IBM DataWorks service on Bluemix Catalog

Figure 3-22 shows an example of creating a DataWorks service. We chose **dev** for **Space**:, **Leave unbound** for **App:**, and **DataWorksDemo** for **Service name**:. You can choose the appropriate parameters for your Bluemix environments:

1. If you want to bind this service to an application, you can choose one from the **App:** dropdown list.



DataWorks
IBM

PUBLISH DATE
05/30/2015

TYPE
Service

LOCATION
US South

VIEW DOCS

The IBM DataWorks™ data refinery transforms raw data into relevant information. It includes IBM DataWorks Forge, an app primarily for knowledge workers, as well as APIs for application developers. IBM DataWorks leverages a highly performant and scalable engine to discover, profile, enrich, mask and deliver data to applications.

- **Forge (Beta)**

  A data-rich app that empowers knowledge workers - including business analysts, data scientists and non technical users - to find data, visualize it, and prepare it for use. By automatically profiling, classifying, and scoring data, Forge guides you through the process of enriching and improving the quality of data using actions such as removing duplicates, filtering and joining. After you prepare and enrich your data, Forge makes it easy for you to deliver data to applications and systems.

- **APIs**

  Flexible, REST-based APIs enable developers to quickly access data and ensure it is fit for purpose. Using the IBM DataWorks APIs, you can quickly create higher-quality applications that load data between data sources (such as SQL Database, Object Storage, dashDB, IBM Analytics for Hadoop, DB2 and Oracle); mask data while loading; securely load on-premises data to cloud environments; cleanse US postal addresses; and classify and profile data.

Pick a plan

Monthly prices shown are for country or region: Australia

Plan	Features	Price
✓ Free	unlimited use of IBM DataWorks	Free

ⓘ There is currently no charge for the use of this service.

Add Service

Space: dev

App: Leave unbound

Service name: DataWorksDemo

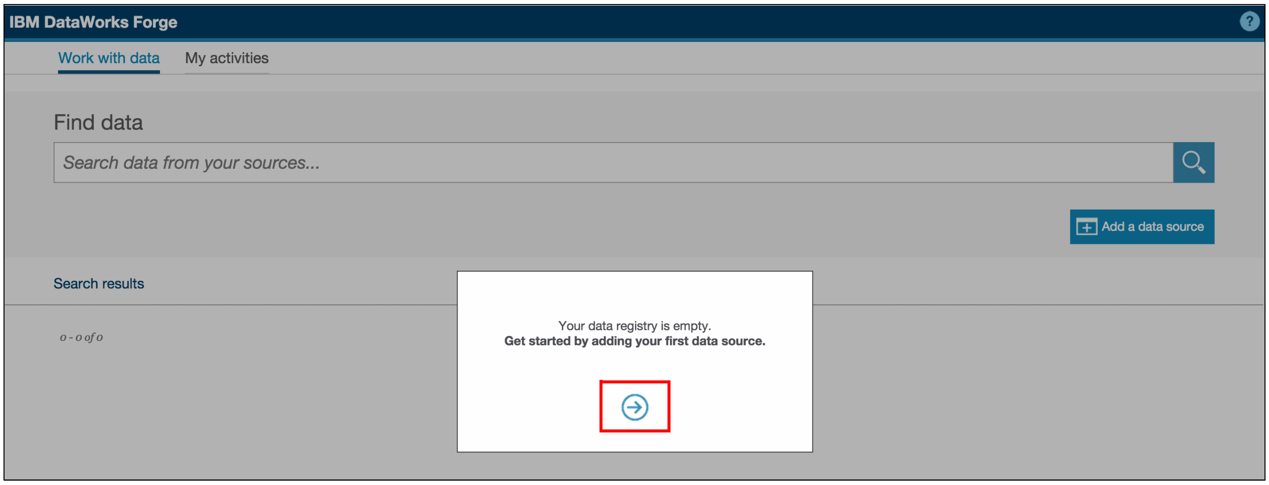
Selected Plan: Free

CREATE

Figure 3-22 IBM DataWorks contains DataWorks Forge and APIs

2. After the DataWorks service is created, choose **DataWorks Forge**, which leads to a window similar to Figure 3-23. Because this is created for the first time, the data registry is empty. Choose the **arrow** in the prompt to add the first data source. A window similar to Figure 3-24 that shows the currently supported data sources is displayed. Choose **SQL Database** here.

   Table 3-4 on page 47 lists the source and target data sources combination supported by DataWorks Forge.



Your data registry is empty.
Get started by adding your first data source.

Figure 3-23 DataWorks Forge prompting to add the first data source

3. As a demonstration here, we add SQL database services from Bluemix, which has been populated with a database containing a table named **visitors**. This database is used as the source database. You can easily create a `.csv` file with relevant data and then use SQL database data import features to populate a table in the database.

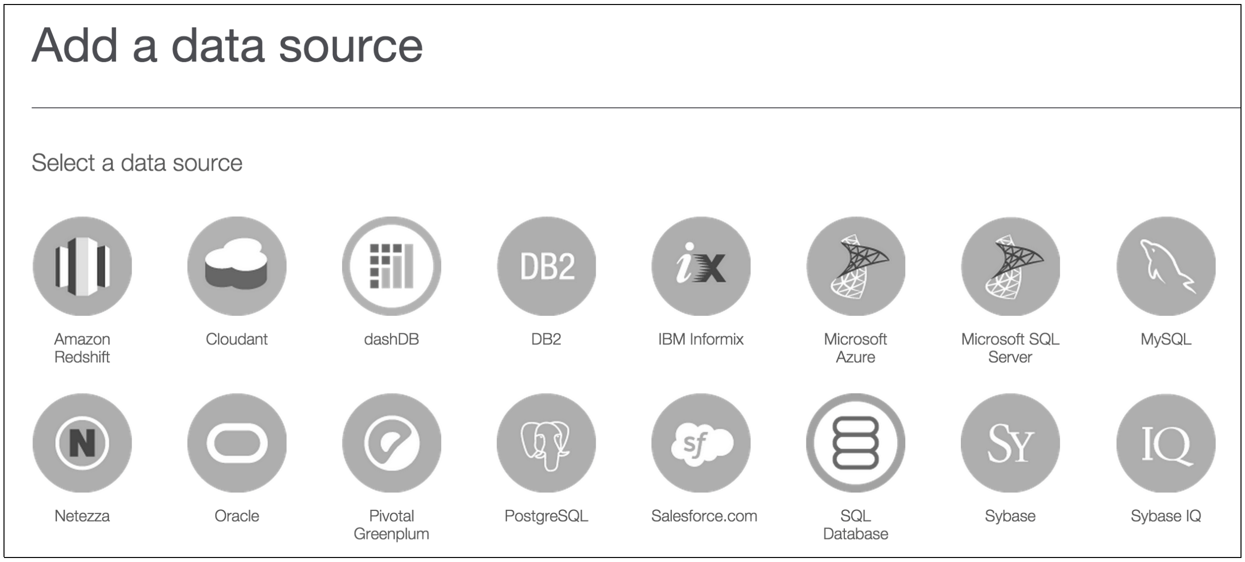


Figure 3-24 Currently supported data sources for DataWorks

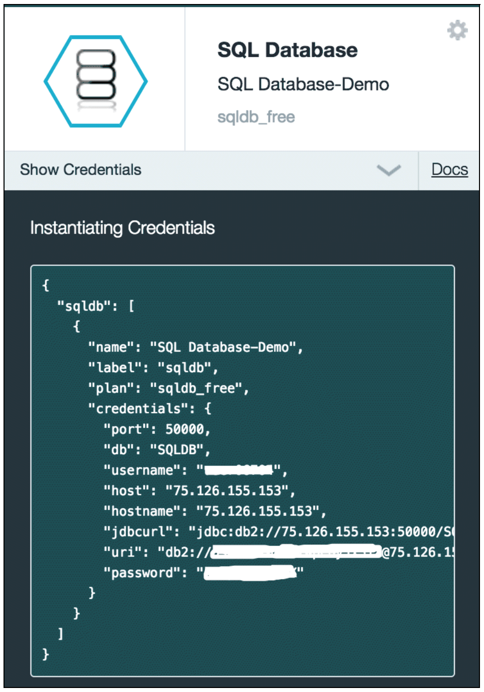
As mentioned earlier, Figure 3-24 shows currently supported data sources by DataWorks Forge. This is presented in a tabular form in Table 3-4 on page 47.

Table 3-4 DataWorks Forge supports the following source and target

Source	Target
Amazon Redshift	
Cloudera Impala	
IBM Cloudant NoSQL DB	
IBM dashDB™	
IBM DB2	
IBM Informix®	
IBM Netezza®	IBM Cloudant NoSQL DB
IBM SQL Database	IBM DashDB
Microsoft Azure	IBM SQL Database
Microsoft SQL Server	IBM Watson™ Analytics
MySQL	
Oracle	
Pivotal Greenplum	
PostgreSQL	
Salesforce.com	
Sybase	
Sybase IQ	

4. Next, as shown in Figure 3-26 on page 49, either select an existing connection (for example, by selecting *connection_to_demo_sql from the table*) or choose **Add a connection** and then complete the details for the table.

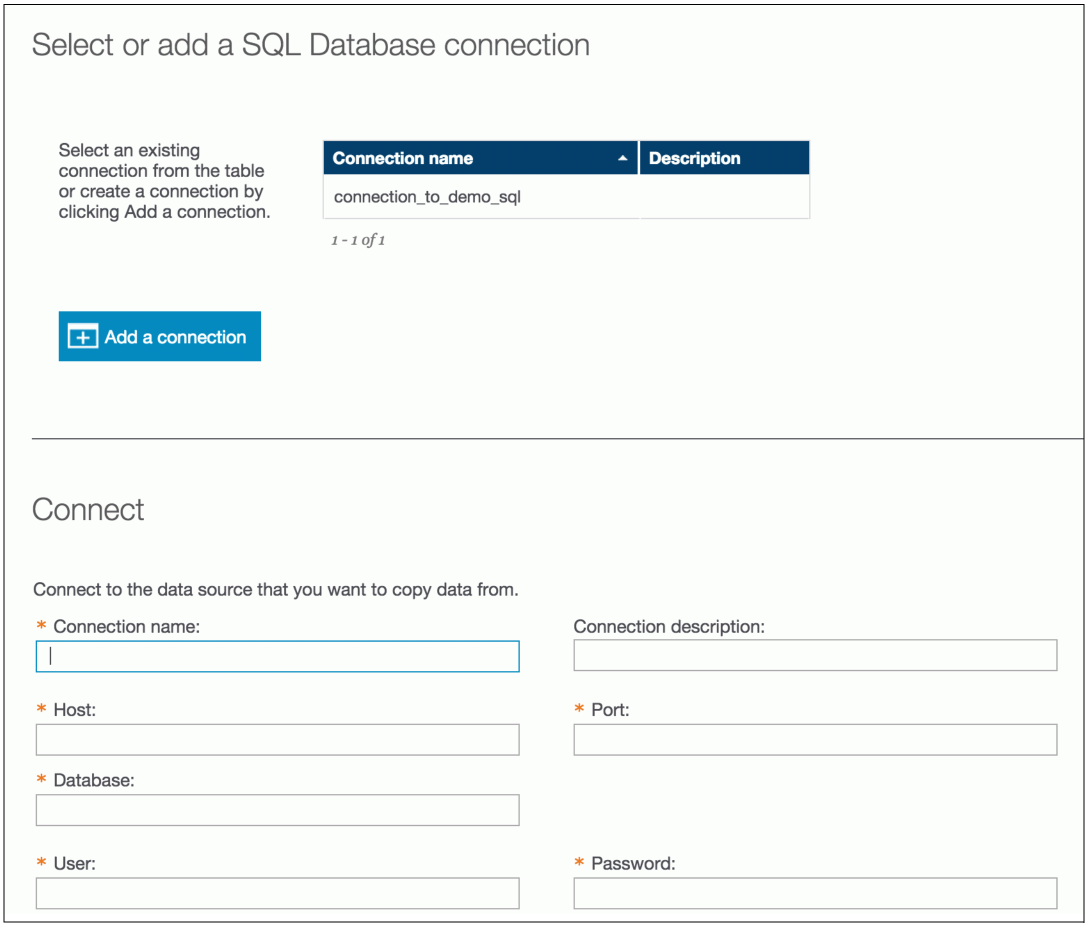
Note: If you are connecting to a Bluemix database service, for example SQL Service, get the **Host**, **Port**, **User**, **Password** and so on, from VCAP_SERVICES variables. See Figure 3-25 as an example.



```
{
  "sqldb": [
    {
      "name": "SQL Database-Demo",
      "label": "sqldb",
      "plan": "sqldb_free",
      "credentials": {
        "port": 50000,
        "db": "SQLDB",
        "username": "[redacted]",
        "host": "75.126.155.153",
        "hostname": "75.126.155.153",
        "jdbcurl": "jdbc:db2://75.126.155.153:50000/S
        "uri": "db2://[redacted]@75.126.1
        "password": "[redacted]"
      }
    }
  ]
}
```

Figure 3-25 VCAP_SERVICES variables for SQL database service

5. We have obfuscated **`username`**, **`uri`**, and **`password`** fields here in Figure 3-25. For your SQL service on Bluemix, find the parameters from VCAP_SERVICES variables and use these to create a new connection.



Select or add a SQL Database connection

Select an existing connection from the table or create a connection by clicking Add a connection.

Connection name	Description
connection_to_demo_sql	

1 - 1 of 1

Connect

Connect to the data source that you want to copy data from.

* Connection name:

Connection description:

* Host:

* Port:

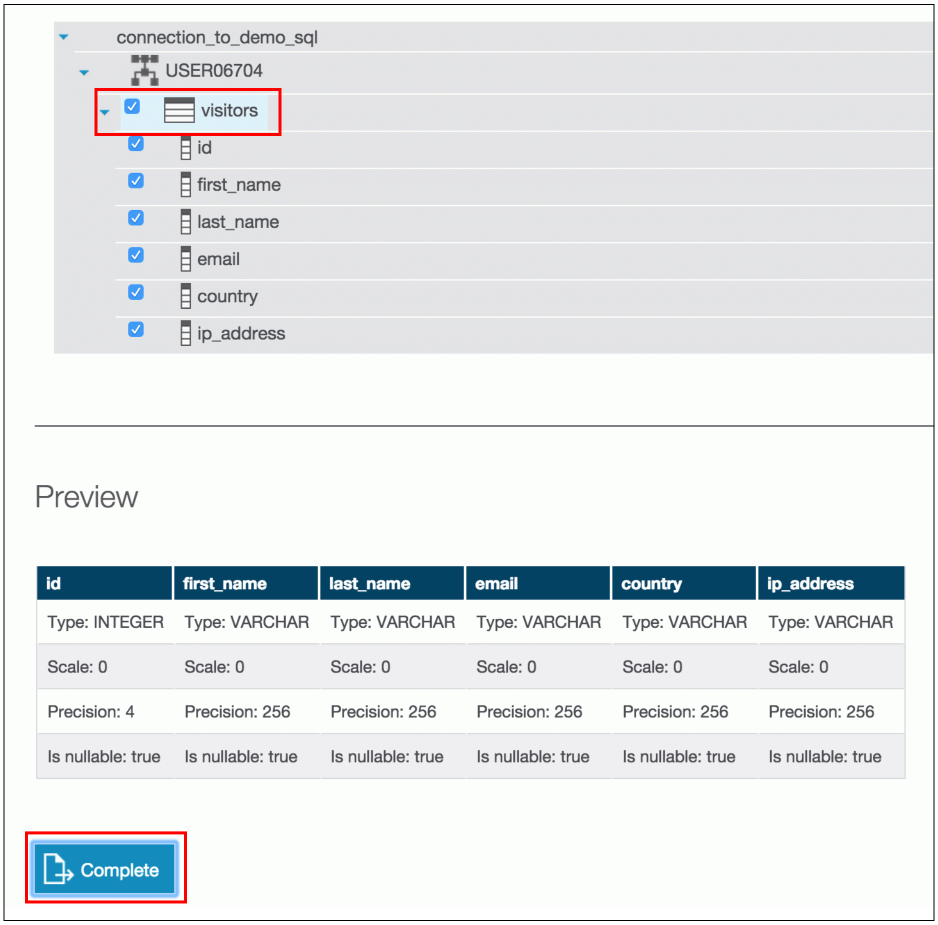
* Database:

* User:

* Password:

Figure 3-26 Adding a SQL connection

6. The next step is to choose the table and choose **Complete** as shown in Figure 3-27.



id	first_name	last_name	email	country	ip_address
Type: INTEGER	Type: VARCHAR	Type: VARCHAR	Type: VARCHAR	Type: VARCHAR	Type: VARCHAR
Scale: 0	Scale: 0	Scale: 0	Scale: 0	Scale: 0	Scale: 0
Precision: 4	Precision: 256	Precision: 256	Precision: 256	Precision: 256	Precision: 256
Is nullable: true	Is nullable: true	Is nullable: true	Is nullable: true	Is nullable: true	Is nullable: true

Figure 3-27 Selecting the intended database table from database schema

7. When the data is ready for IBM DataWorks Forge to use, we choose **Shape** to shape (sort, filter, or join), as shown in Figure 3-28.

> **Screen captures:** The intent of the screen captures shown here is mainly to introduce the features. The detailed step-by-step instructions are provided in Chapter 8, “Watson Analytics in hybrid cloud using Secure Gateway and DataWorks” on page 195.

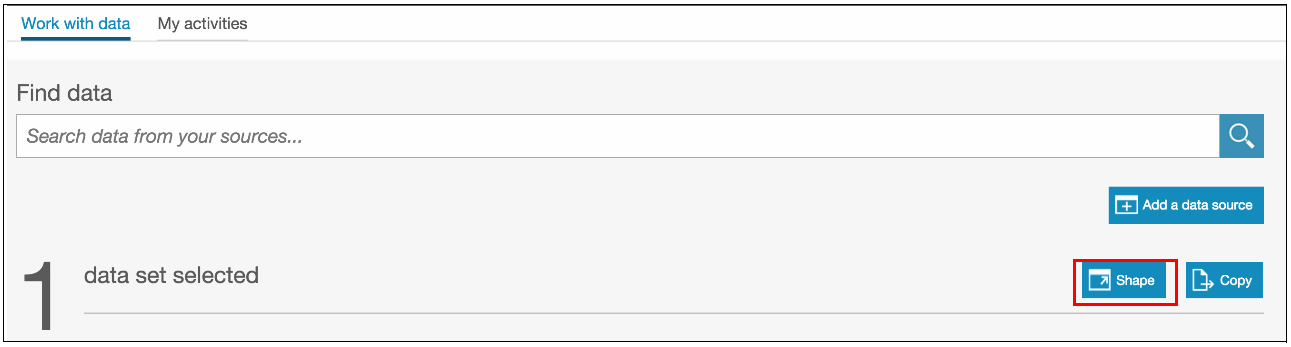


Figure 3-28 DataWorks Forge allows you to shape the data and copy the shaped data to a target database

Figure 3-29 shows the overall quality, size, and type of the data, and issues identified with the data.

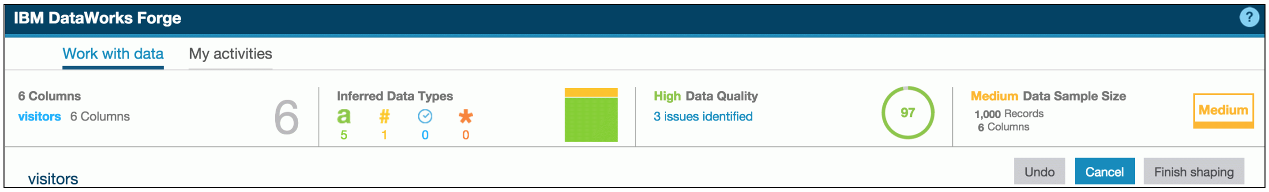


Figure 3-29 Overall quality of the data

DataWorks Forge also allows you to look at the quality metrics for each column, as indicated in Figure 3-30.

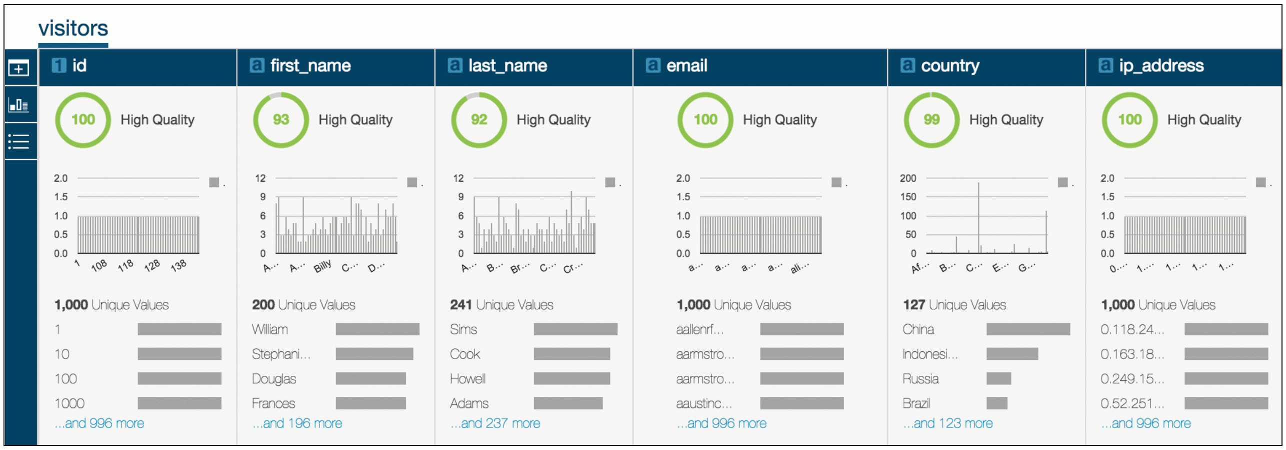


Figure 3-30 Column metrics from DataWorks Forge

8. When the developers or IT team or the business users are happy with the quality of data after shaping (sorting, filtering, or joining), it is time for this shaped data to be copied to a target database as indicated by Figure 3-31 and Figure 3-32 on page 52.

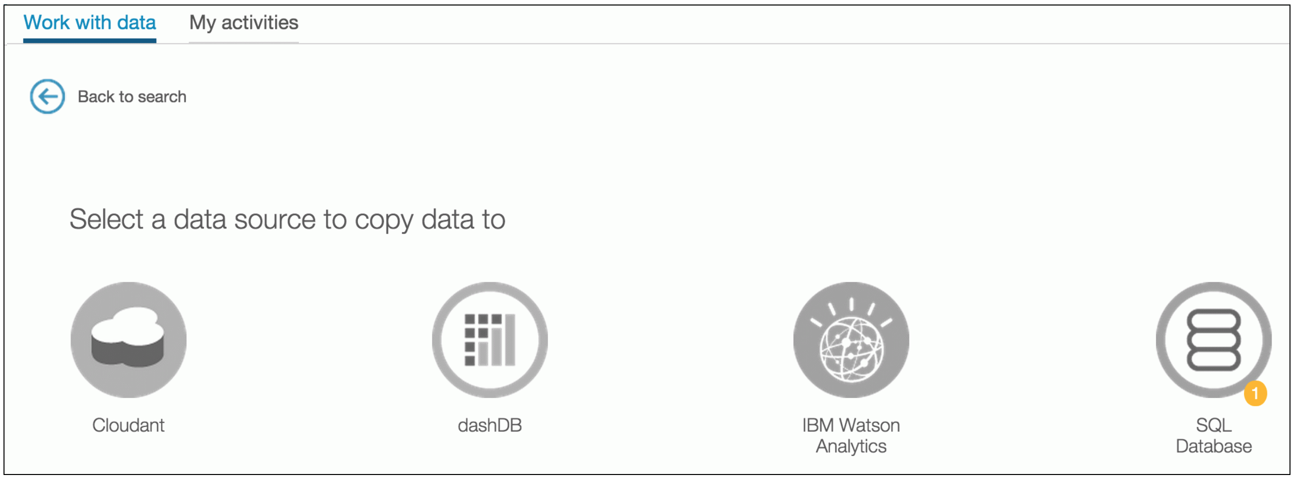
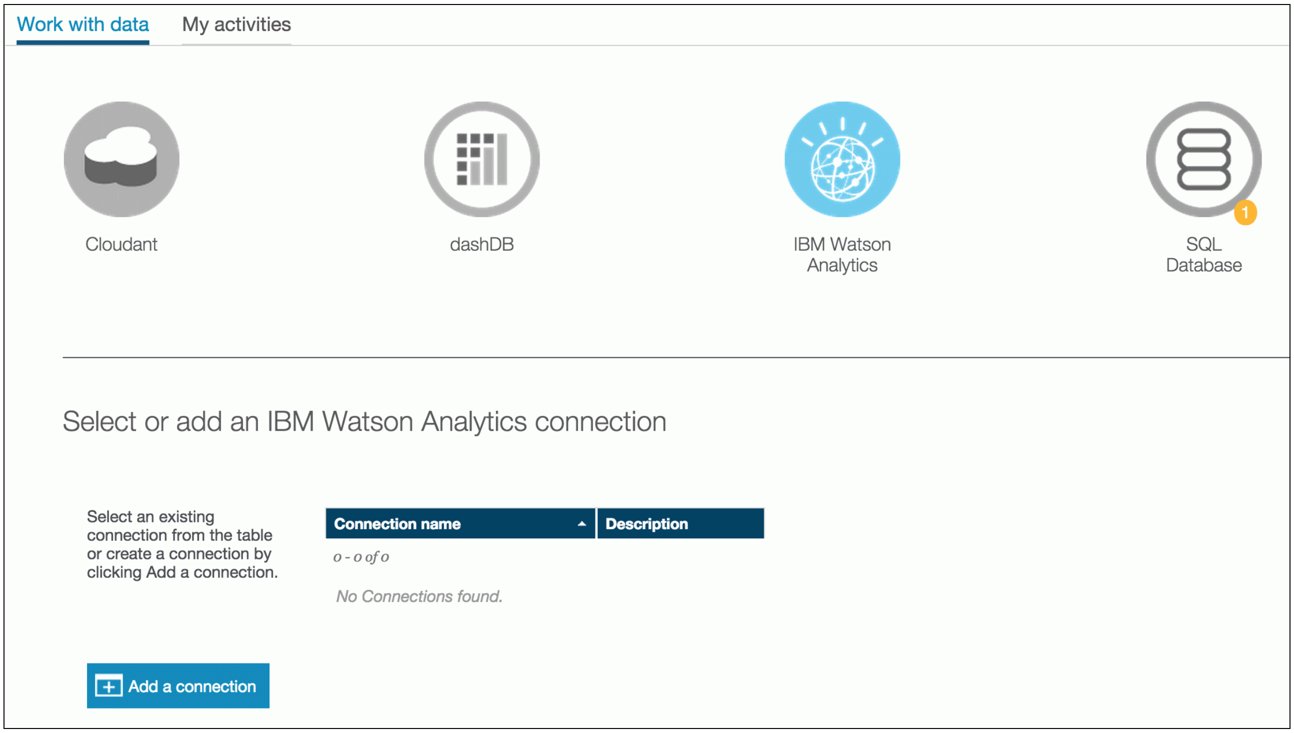


Figure 3-31 Selecting a target database to copy the shaped data to



Select an existing connection from the table or create a connection by clicking Add a connection.

0 - 0 of 0

No Connections found.

Figure 3-32 Selecting Watson Analytics as the target

9. A developer might choose Watson Analytics in this instance. Choose **IBM Watson Analytics** as shown in Figure 3-32, and select **Add a connection** to complete the relevant details.

   Thus, IBM DataWorks Forge allows a user to easily find data, shape it, and copy it to a target data source. This was a quick demonstration to introduce you to DataWorks Forge.

IBM DataWorks APIs

IBM DataWorks provide three APIs. These are discussed in this section.

- Data Load

  Data Load REST API is useful for loading data from cloud and on-premises database via Secure Gateway and then finally provisioning the data to a target. The following source and target data sources are supported as mentioned in Table 3-5 on page 53.

Table 3-5 DataWorks APIs support the following source and target

Source	Target
Amazon Redshift	
Amazon S3(CSV files)	
IBM Analytics for Apache Hadoop	
Apache Hive	
IBM Cloudant NoSQL DB	
IBM DashDB	
IBM DB2	
IBM Informix	
IBM Netezza	IBM Analytics for Apache Hadoop
IBM Object Storage for Bluemix version 1 (CSV files)	IBM Cloudant NoSQL DB
IBM SQL Database	IBM DashDB
Microsoft Azure	IBM SQL Database
Microsoft SQL Server	IBM Watson Analytics
MySQL	
Oracle	
Pivotal Greenplum	
PostgreSQL	
Salesforce.com	
Sybase	
Sybase IQ	

- Address Cleansing

  Address Cleansing API is useful for standardizing US addresses. It expects five address input fields and returns extra address information when input parameters match Coding Accuracy Support System (CASS) reference files containing United States Postal Service (USPS) standard addresses.

  Note: Address Cleansing API is only available in the United States Bluemix instance.

  Request body of the REST call supports 64 characters maximum for `address1` and `address2` fields and 42 characters for other input fields. Up to 100 addresses can be included for verification with each API call.

Example 3-2 shows the input parameters for an address request.

Example 3-2 Input parameters for an address request

```
{
"addresses": [{
"zipcode": "27709",
"state": "NC",
"address1": "3039 East Cornwallis Rd",
"address2": "",
"city": "Research Triangle Park"
}]
}
```

- Data Profiling

  The Data Profiling API classifies, analyzes, and validates data. This API is useful for understanding the content and structure of the data. This API can be explained with an example. Suppose CompanyC acquired CompanyB. CompanyC requires to integrate CompanyB's data assets with their existing data assets. A developer from CompanyC has been tasked to analyze the data from CompanyB in this regard. The developer can use the Data Profiling API to understand the following information:

  – Type of the data in various fields of a database table or document, for example, date, numeric, string, and so on.

  – Length and format of the fields.

  – Whether fields contain data, such as customer names, credit card numbers, telephone numbers, and so on.

  The Data Profiling API supports the following formats and sources as shown in Table 3-6.

Table 3-6 Formats and sources supported by DataWorks Data Profiling API

Data format	**Source**
Comma-separated value (CSV) file	► Amazon Simple Storage Service (S3) ► SoftLayer Object Storage
Delimited file	► Amazon Simple Storage Service (S3) ► SoftLayer Object Storage
Relational table	► SQL Database ► dashDB

In summary, IBM DataWorks service on Bluemix helps with data loading, shaping, profiling, and standardizing by means of DataWorks Forge and DataWorks APIs.

5 Things to Know about IBM DataWorks blog post: For more information about IBM DataWorks, check out *5 Things to Know about IBM DataWorks*:

https://www.ibm.com/developerworks/community/blogs/5things/entry/5_Things_to_know_about_IBM_DataWorks?lang=en

3.1.5 Event-centric services: Message Hub and Event Hub

Various applications software and services provide capabilities for publishing events when certain conditions occur. To develop innovative composable applications in a hybrid environment, it makes more economical sense to use existing functionality provided by third parties. However, building the integration points involves much work and effort and as new services are available, more development work is needed.

A better solution is to use the concept of connectors that are configurable for a particular service. The configuration could include information such as credentials, API Keys, and end points. As an example for such external services, Twitter Inc.® provides an API to receive a stream of tweets based on some search criteria; likewise, Salesforce.com enables publishing messages when specific events occur or data changes within the system. A messaging system that supports the publish/subscribe mechanism helps downstream applications to consume these events in a reliable manner.

Figure 3-33 depicts the Event Hub and Message Hub integration architecture.

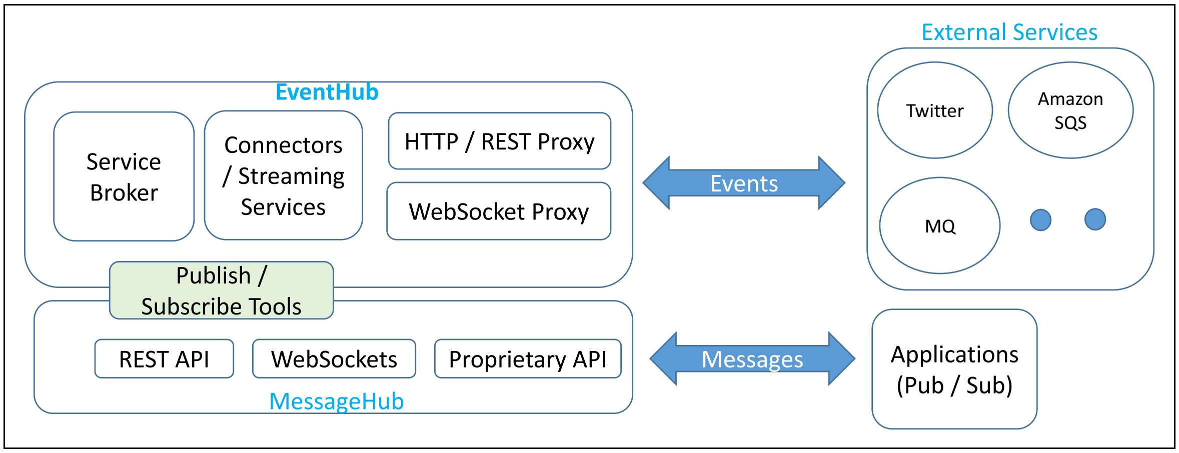


Figure 3-33 Event Hub and Message Hub integration architecture

Event Hub provides intelligent connectors for these services in addition to more traditional publish/subscribe tooling found in products like MQ Light, RabbitMQ, and so on. Each Event Hub connector is built against a specific service or technology (for example, Twitter, Cloudant, Force.com, MQ Light, and so on), and provides configuration options for entering connection credentials, search parameters, and so on.

Event Hub aggregates events from configured services into a single publish/subscribe service called *Message Hub* (available from the Bluemix Catalog), which then provides several mechanisms for subscribing to configured event streams. Subscription methods can include REST APIs, and other proprietary APIs.

After a connector has been configured, it is initialized and can begin receiving data from the target system. When the data begins flowing across the Message Hub topic, cloud-connected services can begin consuming the data by using one of the mechanisms provided.

Streams ultimately have simple lifecycles. After initialization, they can be stopped, restarted, or deleted. Event Hub is broken up into several components:

- Service broker
- IBM Container Extensions broker
- WebSocket proxy
- HTTP/REST proxy

- Premium connectors and Streaming services

Event Hub is built atop Message Hub service. Message Hub provides the backend for managing large volumes of publish/subscribe messages using a high-performance, robust, distributed system cluster. Event Hub uses this highly distributed architecture.

Event Hub requires the following steps:

1. When a new event stream is created (the connection between some service and the Event Hub system), this “connector” first establishes a connection to the Event Hub service broker in order to provide a “heartbeat” status for each stream, as well as enable passing status and debug messages back to the broker.
2. The connector establishes a connection to the target system and begins listening for events generated by that system.
3. When a new event is received, it is passed via a third concurrent connection to the Message Hub service.

> **Important:** At the time of writing this book Event Hub and Message Hub were not publicly available, so some of these details might slight change when these services become generally available.
>
> ITSO Redbooks team is going to publish a companion book titled “*Hybrid Cloud Event Integration: Integrate Your Enterprise and Cloud with Bluemix Integration Services*, SG24-8281 in 1Q 2016. This book will focus exclusively on hybrid cloud scenarios with Bluemix using Event Hub and Message Hub.

3.2 Other IBM products

In this section, we describe other IBM products that are not part of IBM Bluemix offerings. These products fall under the following categories:

- Utility software
- Physical device
- Virtual appliance
- Cloud-based solution

3.2.1 z/OS Connect

z/OS provides critical business value with systems such as CICS and IBM IMS™ to the largest enterprises across all industries. Those systems are often backed by DB2 and store vast amounts of business transactional and other data. Systems of Engagement (SOE) like mobile applications need to be able to access the transactions and data on IBM z Systems™. The question is how to connect the cloud environment to the core z Systems environment in a way that is consistent and manageable.

Modern cloud applications use RESTful services and JSON Data. z/OS Connect is software function written by IBM that runs inside an instance of WebSphere Liberty Profile z/OS, and uses existing connector technology to get to the backend systems. It is supplied as part of WebSphere Application Server z/OS, CICS, or IMS. At a high level, it exposes z/OS applications via a REST/JSON interface making consumption of those services consistent with other services that mobile/cloud developers are used to consuming.

Figure 3-34 on page 57 shows where x/OS Connect fits in-between the SOE running in the cloud and the Systems of Record (SOR) such as CICS and IMS running on z Systems.

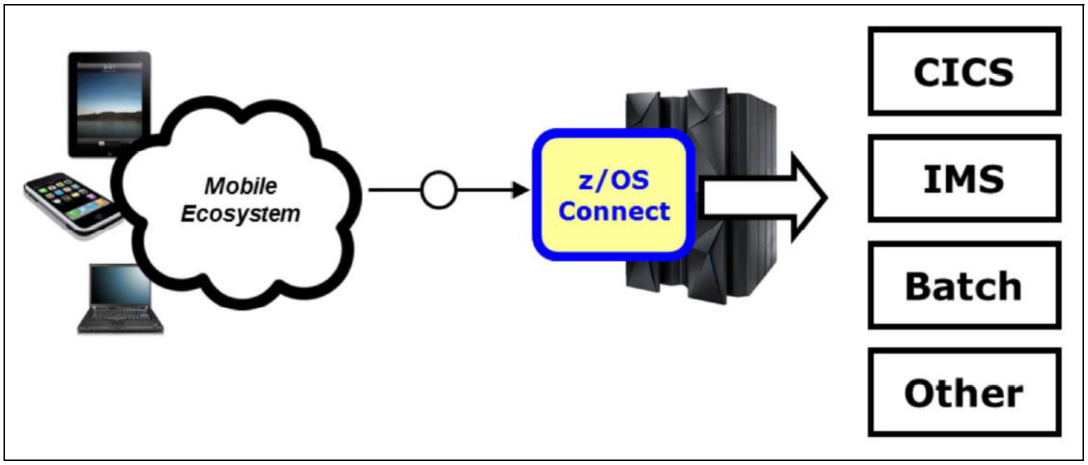


Figure 3-34 Where z/OS Connect fits in

The following list provides the features of z/OS Connect:

- There is no additional cost to use z/OS Connect, and it is packaged with WebSphere Application Server, CICS, and IMS software products.
- Built on Liberty Profile for z/OS, which means it is lightweight and dynamic. It can run as a z/OS Started Task, which means it can run within your current operational procedures and routines.
- Provides a RESTful API and accepts JSON data payloads.
- Configurable so that you control what backend programs or applications are exposed and accessible.
- Provides a discovery function so application developers can query for a list of configured services as well as query for details about the services.
- Capable of performing data conversion from JSON to the data format required by the backend configured service.
- Optional authorization checking using the System Authorization Facility (SAF) to allow or deny users access to z/OS Connect services. SAF is the System Authorization Facility element of z/OS. Its purpose is to provide the interface between those products requesting security services and the external security manager installed on the z/OS system.
- Optional activity recording using System Management Facilities (SMF) to track requests by date and time, bytes sent and received, and response time. SMF collects and records system and job-related information that your installation can use in the following functions:
  - Billing users
  - Reporting reliability
  - Analyzing the configuration
  - Scheduling jobs
  - Summarizing direct-access volume activity
  - Evaluating data set activity
  - Profiling system resource use
  - Maintaining system security

> **Key point:** z/OS Connect is a REST/JSON interface to a z/OS LPAR. It provides a common and consistent REST/JSON interface to many different backend systems.

You can achieve the same goal of accessing CICS or other z/OS systems without z/OS Connect. So why use z/OS Connect? Here are some benefits to using z/OS Connect:

- Provides a common and consistent entry point for mobile access to one or many backend systems.
- Written in Java, so it runs on specialty engines.
- Shields backend systems from requiring awareness of RESTful Uniform Resource Identifiers (URIs) and JavaScript Object Notation (JSON) data formatting.
- Provides point for authorization of user to invoke backend service.
- Provides point for capturing usage information using SMF.
- Simplifies front-end functions by allowing them to pass RESTful and JSON rather than be aware of or involved in data transformation.

Figure 3-35 provides a summary of the key features and capabilities of z/OS Connect.

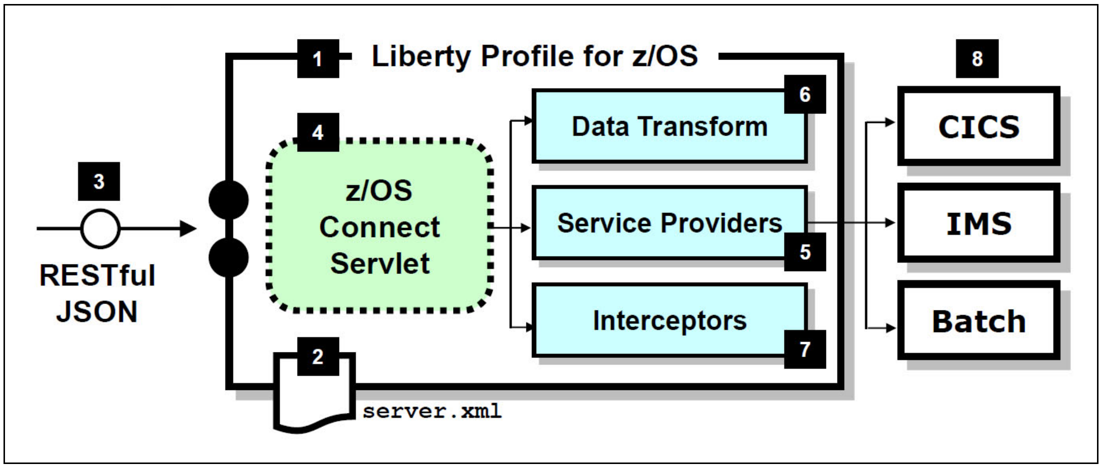


Figure 3-35 Summary of z/OS Connect key features and capabilities

Following is an explanation of the diagram:

1. z/OS Connect is a software function that runs in Liberty Profile for z/OS.
2. z/OS Connect is described and configured in the Liberty `server.xml` file.
3. z/OS Connect is designed to accept RESTful URIs with JSON data payloads.
4. One part of z/OS Connect is a servlet that runs in Liberty Profile z/OS.
5. A Service Provider is software that provides the connectivity to the backend system.
6. z/OS Connect provides the ability to transform JSON to the layout required by backend.
7. Interceptors are callout points where software can be invoked to do things such as SAF authorization and SMF activity recording.
8. Initially, the backend systems supported are CICS, IMS, and Batch.

Figure 3-36 shows the high-level application architecture of a typical mobile SOE application that needs to execute CICS transactions on z/OS. The application uses the Mobile Cloud boilerplate and leverages the Mobile Data, Mobile Quality Assurance, and Push Notifications services. A NodeJS run time provides orchestration of service calls mapping to the UI interactions in the mobile application.

On the z/OS side, z/OS Connect exposes CICS transactions via a REST/JSON interface. The Secure Gateway service is used to securely connect from Bluemix to the client's z/OS mainframe. A DataPower Gateway virtual appliance v7.2 running on x86 provides access into the corporate data center.

The API Management service in Bluemix makes the z/OS Connect exposed REST/JSON interface available as APIs in the Bluemix Catalog over the Secure Gateway. The NodeJS run time makes calls to these APIs.

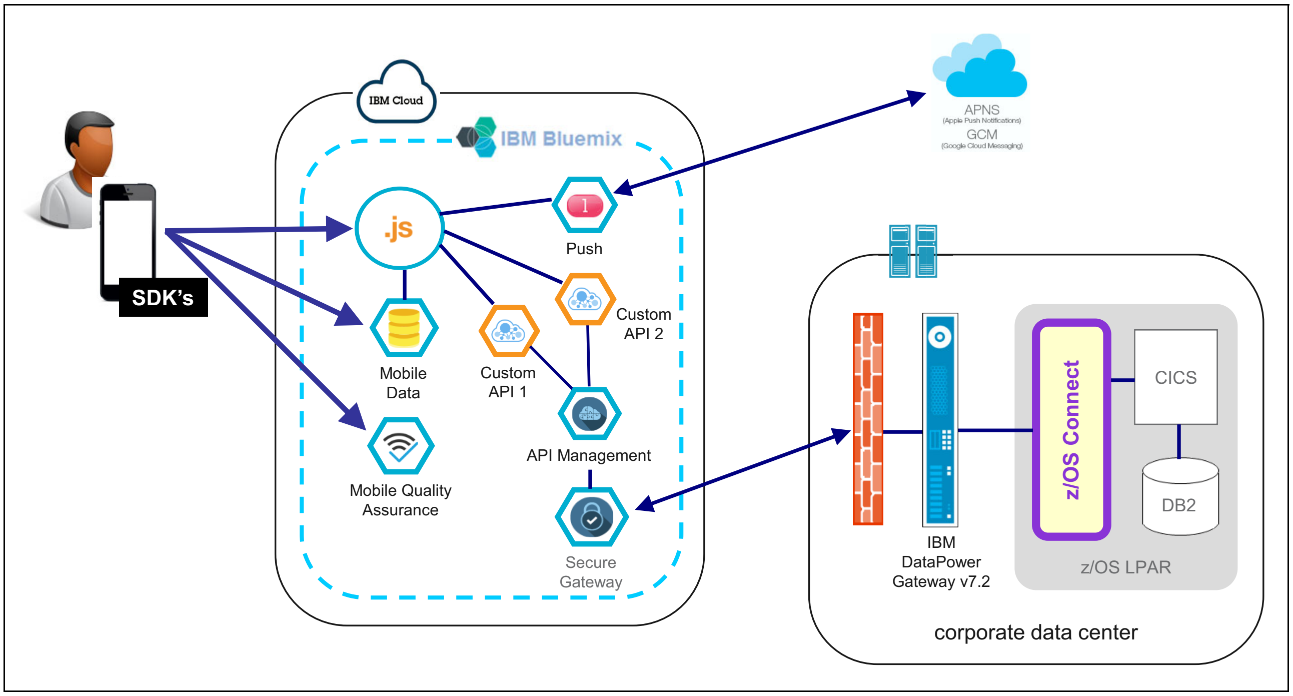


Figure 3-36 Mobile app using an existing CICS transaction using z/OS Connect

z/OS Connect is designed to enable better and more manageable connectivity between mobile systems and your backend z/OS programs, applications, and systems. It provides a consistent front end interface for mobile systems using REST and JSON, and it shields backend systems from having to understand those protocols and formats. It enables enterprises to leverage their existing investments in z/OS in modern applications and accelerate the development of these applications using the Bluemix platform.

5 Things to Know about z/OS Connect blog post: Refer to *5 Things to Know about z/OS Connect*:

https://www.ibm.com/developerworks/community/blogs/5things/entry/5_Things_to_Know_about_z_OS_Connect?lang=en.

3.2.2 IBM WebSphere Cast Iron Integration

Enterprises today have a wide variety of packaged, commercial off the shelf (COTS), and custom applications running in their environment. There is much investment that went into these applications that are not necessarily for cloud consumption. However, the benefits of cloud rely on the enterprises' ability to integrate their current applications with the new services and take advantage of them to make their business more efficient and serve their customers.

The primary objective that enterprises are looking to achieve from hybrid cloud at the enterprise level is the ability to integrate applications that are geographically separated but accessible through network in a secure fashion. The locations where applications reside could be on-premises behind a client firewall, in the cloud service provider environment like a hosted private cloud, or a third-party data center. They are looking to tap into the external application services that cloud service providers are able to offer to gain agility and innovate faster and be a disruptor in the marketplace.

In a hybrid cloud environment, specifically between on-premises environment and external public/private clouds, there is a need for easier application integration. IBM WebSphere Cast Iron® Cloud Integration solutions are designed to connect hybrid environments composed of cloud and on-premises applications integrating with SaaS and mobile applications. The Cast Iron portfolio includes process integration, data integration, UI mashups, and combinations.

IBM solution to achieve application integration in the hybrid cloud comes in three different deployment models:

- WebSphere DataPower Cast Iron Appliance XH40: This is a self-contained physical appliance. The appliance can be installed and managed within a local data center. It provides all the functionality needed to connect the cloud and on-premises applications.
- WebSphere Cast Iron Hypervisor Edition: This is a virtual appliance that can be installed on client's existing servers using virtualization technology. Virtual appliances allow better utilization of hardware and faster response to demands for newly deployed systems on the real appliance, thus helping to reduce the costs of both hardware and software operation and maintenance. Deploying virtual images can help businesses reduce the potential for errors and enable the rapid deployment of working systems for development, test, or production because virtual images are built with known, stable, and tested configurations.
- WebSphere Cast Iron Live: A complete multi-tenant cloud offering to connect cloud and on-premises applications. WebSphere Cast Iron Live enables clients to design, run, and manage integrations in the cloud without any infrastructure footprint onsite.

Figure 3-37 on page 61 shows the different configurations that are available for integrating the Cast Iron appliance. The figure also depicts the four different ways to integrate cloud native applications with the system of record (SOR) residing in the client data center:

1. The System of Engagement applications running in the public or hosted private cloud can access SOR by using the Cast Iron live SaaS solution over secure network connection leveraging the APIs.
2. A Cast Iron hardware appliance XH40 can be installed in the client data center and integrated into the network infrastructure and provide connectivity to the SOR client applications.
3. A Cast Iron virtual appliance for XH40 can be installed in the client data center on a virtual machine and integrated into the network infrastructure and provide connectivity to the SOR client applications.
4. Cloud Native applications can use the Bluemix PaaS Cloud Integration Services to connect to client SOR backend applications.

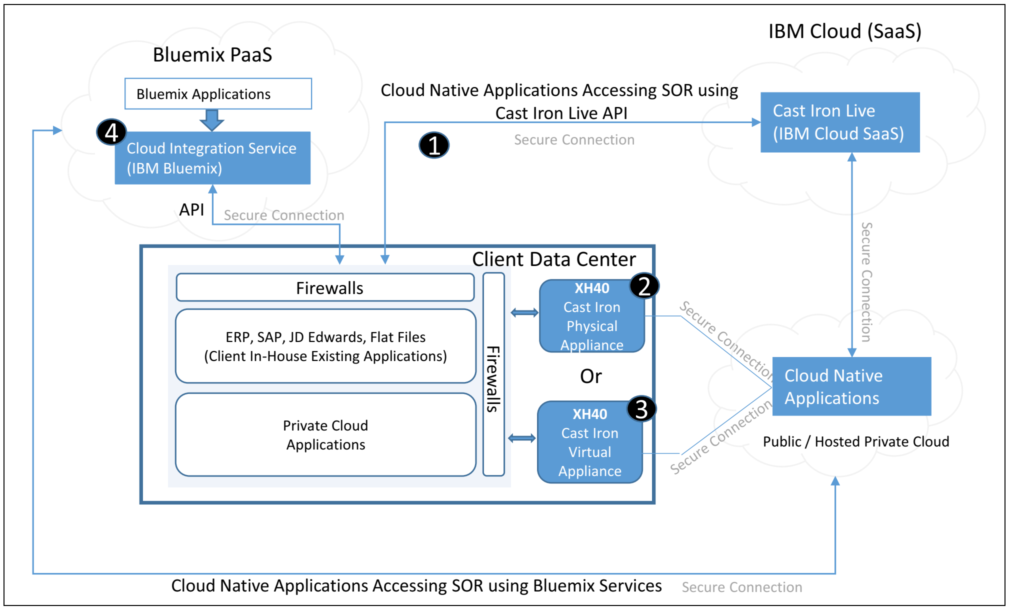

IBM Cloud (SaaS)
Bluemix PaaS
Bluemix Applications
4
Cloud Integration Service (IBM Bluemix)
API
Secure Connection
Cloud Native Applications Accessing SOR using Cast Iron Live API
1
Secure Connection
Cast Iron Live (IBM Cloud SaaS)
Client Data Center
Firewalls
ERP, SAP, JD Edwards, Flat Files (Client In-House Existing Applications)
Private Cloud Applications
Firewalls
XH40 Cast Iron Physical Appliance
2
Or
XH40 Cast Iron Virtual Appliance
3
Secure Connection
Secure Connection
Secure Connection
Cloud Native Applications
Public / Hosted Private Cloud
Cloud Native Applications Accessing SOR using Bluemix Services
Secure Connection

Figure 3-37 Cast Iron integration options

An integration solution must bridge the gap between the on-premises existing systems and new cloud applications, platform, and infrastructure, providing a rapid and easy-to-use method of setting up the integrations. IBM WebSphere Cast Iron provides a solution that meets the challenge of integrating cloud applications with on-premises systems, cloud applications-to-cloud applications, and on-premises to on-premises applications.

With WebSphere DataPower Cast Iron Appliance XH40 supporting the on-premises environments, IBM continues to enhance its capabilities to address the needs of hybrid environments. It provides rapid access to application data for application developers and driving towards an on-premises API Management solution. The API Management solution includes the most important features that are necessary to support the three most significant players in the API economy: Application developers, business owners, and IT personnel.

WebSphere DataPower Cast Iron Appliance XH40 is a 1U, 4.4 cm (1.75 in.), rack-mountable appliance, which is designed to fit into industry-standard racks, and can be attached to the network via Ethernet.

Clients are looking for a platform that can enable rapid, configuration-based, no-coding construction of integration among SaaS applications, on-premises packaged applications, custom applications, and generic systems such as databases, files, and web services. The Cast Iron “configuration, not coding” approach helps numerous companies go live with their SaaS application initiatives in days or weeks, in contrast to a traditional coding approach, which could easily take many months. The term licensing pricing options help enable clients to get started at a low cost and grow into larger cloud integration projects over time. Following are the key features that Cast Iron Cloud Integration makes available:

- Connect enterprise-grade packaged applications, such as large ERP and CRM applications, and enable access to the target application contents with no coding required.
- Support for a wide range of integration paradigms. Allows connectivity and data integration between salesforce.com applications and other applications.

- Synergy with other IBM integration products: Companies with significant application integration investment can use Cast Iron as the cloud gateway, allowing SaaS applications to become part of the larger integration framework.
- Ability to collaborate with IBM Mobile Foundation: Cast Iron works cooperatively with IBM Mobile Foundation to externalize enterprise data and processes in mobile applications.

WebSphere DataPower Cast Iron Appliance XH40 provides the following comprehensive integration capabilities (Table 3-7).

Table 3-7 WebSphere DataPower Cast Iron Appliance XH40 integration capabilities

Category	Feature	Benefit
Integration	Cloud Data Migration	Allows data handling using the data cleansing and data migration capabilities from existing applications to cloud applications in real time.
Integration	Cloud data synchronization	Provides connectivity, workflow, and transformation features, enabling coordination or orchestration of integration processes across multiple applications in real time. SaaS and cloud users can view data that is hidden in applications, minimizing any duplicate entries and maximizing decision making and productivity.
Integration	Integration in visualization (UI mashups)	Provides the capability to mashup information from disparate sources and to display this information using the native user interface of a cloud application in real time.
Integration	Integration that drives mobile applications	Supports mobile applications by harnessing data and processes from other parts of the enterprise.
Connectivity	Connectors	Comes with built-in connectivity to hundreds of cloud, packaged, and proprietary on-premises applications, including ERP, CRM, databases, web services, and flat files. WebSphere DataPower Cast Iron Appliance XH40 is completely self-contained and includes everything needed to complete integrations in one place.

Category	Feature	Benefit
Connectivity	Endpoints	This approach to integration makes no distinction between local and remote applications because they establish connectivity to the endpoints by using native application protocols. The advantage to this approach is that no additional adapters are required, and there is nothing to install or change at the endpoints.
Connectivity	Customization	Connector Development Kit, provided with this offering, allows skilled developers to use WebSphere Cast Iron to add connectors to custom applications. With connectors built by using the Connector Development Kit, users get the same development experience on IBM WebSphere Cast Iron Studio as with the built-in connectors. Cast Iron Studio is a downloadable development tool that can be used to design, test, and publish integration projects to a WebSphere Cast Iron Integration appliance.
Reusability	Template Integration Processes (TIPs)	This online library contains templates for many of the most common cloud integration scenarios. These templates encode common connectivity, known mapping, as well as best practice integration workflows. Templates provide a question-and-answer-based wizard that walks users through a common integration scenario, customizing as needed. Often, users simply complete the credentials, customize mappings, test, and deploy. Cast Iron clients and partners can create their own wizard-driven, reusable templates with the point-and-click TIP Development Kit and make them available to the entire WebSphere Cast Iron Cloud Integration user community.
Packaging options	Editions	Standard, Enterprise, Development

For the latest information about Cast Iron Cloud Integration, refer to the following URL:

http://www.ibm.com/software/products/en/castiron-cloud-integration

3.2.3 IBM DataPower Gateway

IBM DataPower Gateway appliances are security and integration gateways that help secure, control, integrate, and optimize the delivery of the full range of mobile, web, API, SOA, cloud, and business-to-business (B2B) applications and services. As illustrated in Figure 3-38, DataPower Gateway can reside either in a DMZ or trusted zone or both. In the DMZ, it serves the purpose of security gateway for web services or applications or APIs and B2B partner gateway. Besides these, DataPower Gateway provides network caching, intelligent content routing, and load distribution functionalities. In the trusted domain, DataPower Gateway appliances offer the functionalities of internal security enforcement, integration, runtime SOA governance, web service management, and integration with legacy services.

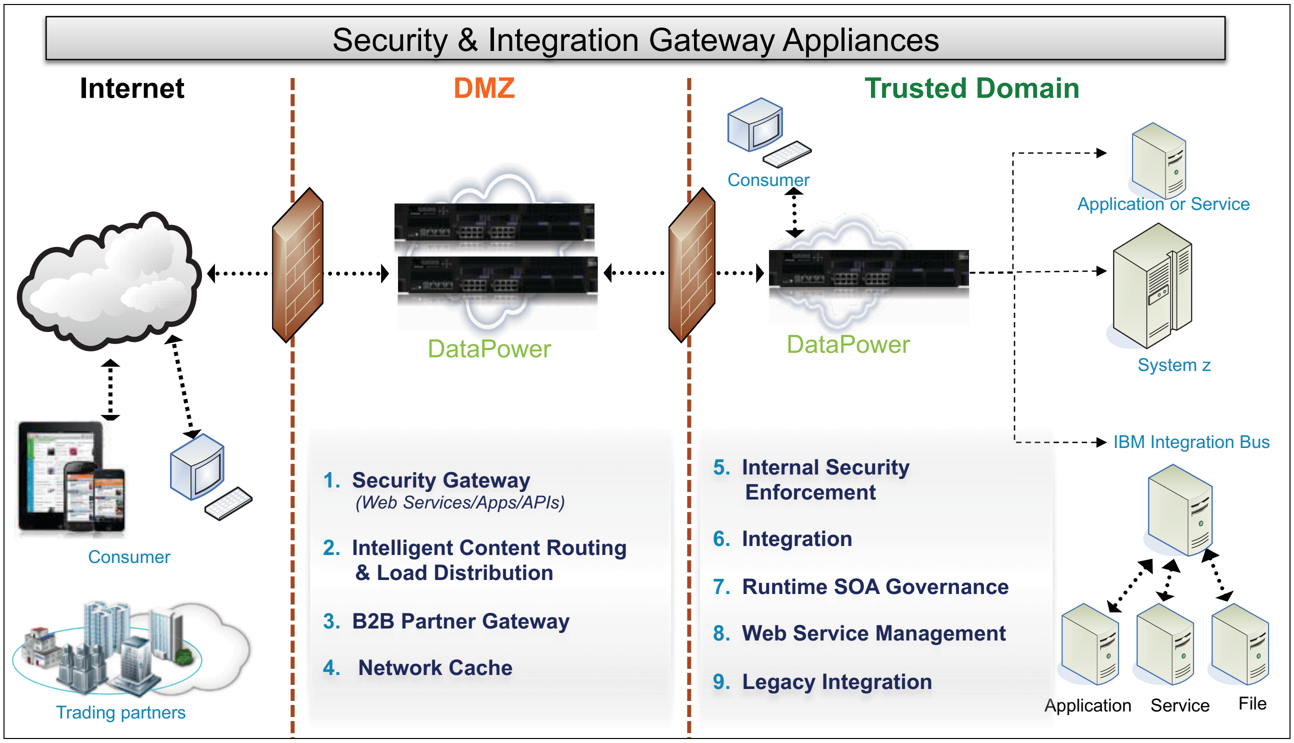


Figure 3-38 DataPower Gateway appliance

In summary, IBM DataPower Gateways are dedicated network devices that help with the following functions:

- *Secure*: Secure and protect back-end systems from harmful workloads and unauthorized users and applications. It does so by offering the following capabilities:
  - Authentication, authorization, auditing
  - Security token translation
  - Threat protection
  - Schema validation
  - Message filtering and semantics validation
  - Message digital signature
  - Message encryption

- *Integrate:* Convert payloads, bridge transports, and connect to existing services at wire-speed. The capabilities are:
  - Any-to-any message transformation
  - Transport protocol bridging
  - Message enrichment
  - Database connectivity
  - Mainframe connectivity
  - B2B trading partner connectivity
- *Control*: Limit and shape traffic based on service level agreements, and route based on message content. The features are:
  - Service level management
  - Quota enforcement, rate limiting
  - Message accounting
  - Content-based routing
  - Failure rerouting
  - Integration with governance and management platforms
- *Optimize*: Improve response times, reduce load on back-end systems, and intelligently distribute load. The capabilities are:
  - SSL/TLS offload
  - Hardware accelerated crypto-operations
  - JSON, XML offload
  - JavaScript, JSONiq, XSLT, XQuery acceleration
  - Response caching
  - Intelligent load distribution

Figure 3-39 depicts how IBM DataPower Gateway consolidates the previously mentioned four functionalities in just one device.

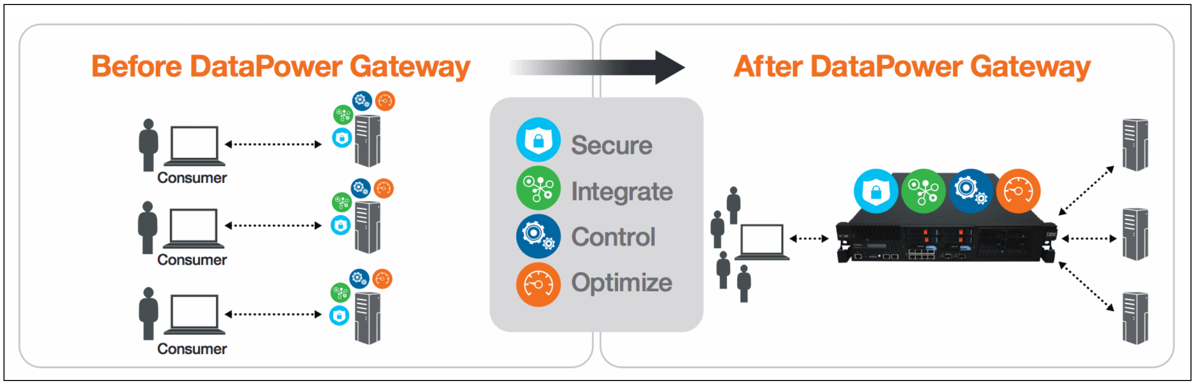


Figure 3-39 IBM DataPower Gateway providing four functionalities in one device

IBM DataPower Gateway is the consolidation of WebSphere DataPower Service Gateway XG45, WebSphere DataPower Integration Appliance XI52, and WebSphere DataPower B2B Appliance XB62. It is available in two form factors:

- IBM DataPower Gateway appliance
- IBM DataPower Gateway Virtual Edition (IDG VE)

IBM DataPower Gateway Virtual Edition provides deployment flexibility with multiple options to deploy to on-premises and cloud environments. This deployment flexibility reduces the costs for development and test environments. IDG VE can run on VMware or Citrix XenServer hypervisor on x86 servers, IBM PureApplication System W1500 and W2500, and SoftLayer dedicated server or bare metal instance. It offers the ability to upgrade and downgrade firmware similar to physical appliances. Configuration of the appliance can be migrated seamlessly between physical and virtual type. It supports SoftLayer instances and Amazon EC2 deployment.

IBM DataPower Gateways are modular. Additional functionalities through modules can be added to both form factors (physical or virtual) as needed. Base firmware provides mobile, web, and SOA workload support. Extra modules are:

- Integration module
- B2B module
- IBM Security Access Manager (ISAM) proxy module
- Application Optimization module
- TIBCO EMS module

Integration module

Integration module extends the capabilities of the IBM DataPower Gateway by providing seamless integration with legacy systems. It provides protocol transformations at network speeds, and supports the most common application protocols including: XML, JSON, REST, SOAP, HTTP, HTTPS, IBM MQ. Integration module offers the following capabilities:

- Offload and optimization capabilities for XML, JSON, REST, SOAP, and binary workloads.
- Mapping between formats such as XML, SOAP, JSON, REST, binary, and COBOL.
- Content and context-based routing and data enrichment facilities.
- Adapters to other connectivity options (IBM MQ, FTP, IMS Connect, TIBCO EMS, and so on).
- Database connectivity for DB2, Oracle, Microsoft, and others.

B2B module

This module is used as a high-throughput, secure entry point for trading partners in the DMZ. This gateway consolidates B2B trading partner connectivity and transaction management, and reduces heterogeneous integration challenges. B2B module is an add-on to standard IBM DataPower Gateway. Besides all features and benefits of DataPower Gateway, B2B module offers:

- Support for B2B messaging protocols, such as AS1, AS2, AS3, and ebMS.
- Includes message mediation, transformation, and database connectivity features as provided by the Integration module, except for IMS support.
- Support for B2B documentation formats, such as EDI-X12, EDIFACT, XML, and Binary documents.
- On-appliance transformation of data.
- Trading partner profiles, covering variables like multiple business locations, business identifications, and partner-specific certification and policy handling information.
- B2B Viewer tool for monitoring and managing B2B communications and transfers.
- A resend facility if a partner loses a message.
- Ability to accept large payloads from partners.
- Specific integration with IBM MQ Managed File Transfer capability.

IBM Security Access Manager proxy module

ISAM module provides enhanced web and mobile access control security. It offers the following benefits:

- Highly scalable reverse proxy for user access control and web single sign-on.
- Integrated enforcement point for context-based access policies for IBM Security Access Manager for Mobile.
- Compliance reporting and security intelligence integration with IBM Security QRadar® SIEM (see http://www.ibm.com/software/products/en/qradar-siem).
- Integrated DataPower Gateway monitoring, event and message logs for improved serviceability, and a single operations and management view.
- Pattern-based configurations, by using the DataPower Gateway pattern console, to simplify the process of creating new Access Management proxy instances.

Application Optimization module

Application Optimization module is useful for enhanced load balancing algorithms within the DMZ. It offers the following benefits:

- Routing of traffic and optimization of the full capacity of the bandwidth available in the customer’s infrastructure.
- High availability support for the network traffic enabling multiple DataPower appliances to operate as a single logical system, which eliminates any single appliance as a point of failure in the DMZ.
- Sophisticated routing within the application zone, tight integration with WebSphere servers, and functionalities as an on-demand router for multiple WebSphere servers or clusters.

TIBCO EMS module

This is an add-on module to IBM DataPower Gateway to provide integration capability to TIBCO EMS application servers.

Chapter 5, “Connecting IBM Containers with on-premises Docker” on page 91, presents a scenario involving IBM POWER8®, Secure Gateway, and DataPower appliance. In this scenario, an open source enterprise resource planning (ERP) system hosted in IBM Containers on Bluemix accesses the data from a PostgreSQL database residing in a Docker container on a POWER8 server. This POWER8 server and a DataPower Virtual Appliance are behind a firewall. A secure connection is made from the ERP application running on Bluemix to PostgreSQL database through Secure Gateway service on Bluemix and DataPower Gateway appliance behind the firewall.

Part 2

Hybrid cloud scenarios with IBM Bluemix

In this part, we cover several hybrid cloud scenarios with Bluemix. Each scenario is described with step-by-step instructions so that you can implement similar scenarios in your environment as well.

© Copyright IBM Corp. 2015. All rights reserved.

4

Connecting to an enterprise database of record

This chapter shows how to connect an application running in Bluemix to an enterprise database of record running in a private data center. This is a hands-on tutorial that you can perform by using your computer and a Bluemix account. The tutorial documentation explains the steps in enough detail that you can follow along without having to perform the steps.

After reading this chapter, you will know how to use the Secure Gateway service in Bluemix to create connections between Bluemix applications and resources outside of Bluemix.

This chapter contains the following sections:

© Copyright IBM Corp. 2015. All rights reserved.

4.1 Solution overview

The solution that we build in this tutorial is intentionally simple. The solution connects an application running in Bluemix to an enterprise database of record running outside of Bluemix. The focus is on how to configure and use Secure Gateway, not on the sophistication of the application or the database.

You can perform this tutorial by using your computer and a Bluemix account. Rather than requiring you to have access to an existing data center hosting a production enterprise database of record, this tutorial simulates one using a MySQL database running in a virtual machine hosted by Bluemix. This enables you to experience installing Secure Gateway without requiring access to a data center.

The techniques shown in this tutorial using this architecture can easily be applied to true production Java applications and enterprise databases running in private data centers. The existing Java application is migrated to Bluemix for the operational efficiencies of cloud computing. Meanwhile, the enterprise database remains in the data center so that other existing applications can continue to access it locally. Secure Gateway is installed in Bluemix and in the data center and configured to connect the Java application to the enterprise database. This tutorial shows how to perform these steps.

Figure 4-1 shows the architecture of the solution that we will build. The sample application is a Java program that enables the user to manipulate the data in an SQL relational database, which is typical functionality of many existing Java applications. This program runs in Bluemix in a Liberty for Java run time. The database stores the Java application's data. It is a SQL database, as is typically used by many existing Java applications. For this tutorial, the data center is simulated by a virtual machine and the SQL database is a MySQL server. The Secure Gateway service in Bluemix is used to connect the Java run time to the MySQL database.

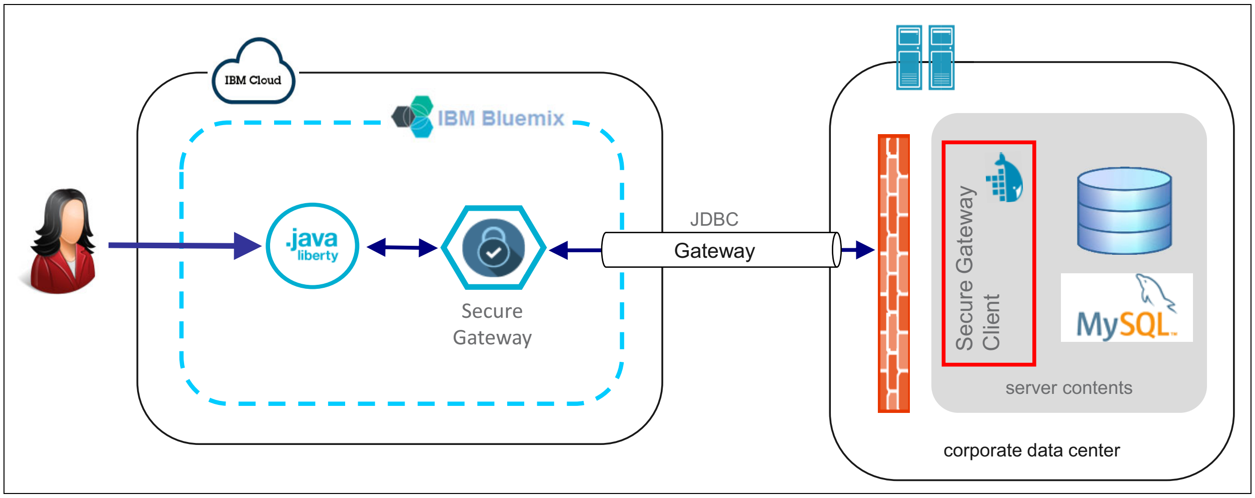


Figure 4-1 Architecture of the solution

Let's get started with the tutorial.

> **Business case for this scenario:** For more information about the business case for this scenario and the company profile, see 1.3.5, "Solutions and actions by CompanyC CIO" on page 11.

4.2 Step-by-step implementation

Because this tutorial is self-contained and can be performed with minimal existing assets, it requires a fair bit of setup. The tutorial leads you through the following setup:

- *Eclipse*: Those who have already developed a Java application for Bluemix will most likely already have the Eclipse setup. For those who have not, or whose installation might not include all of the optional tools that will be helpful, the tutorial explains how to set up the Eclipse IDE for developing applications for Bluemix.
- *Virtual machine:* The tutorial uses a virtual machine running MySQL to simulate an enterprise database hosted in a data center. The tutorial explains how to create the VM, install Docker, and install MySQL initialized with the database needed for the Java application.
- *Create sample app:* The tutorial needs a sample application to connect to the database. We'll use one already included in Bluemix, from the Java DB Web Starter boilerplate.
- *Import sample app:* When we've downloaded the starter code, we'll load the project into Eclipse.
- *Deploy sample app:* We'll deploy the Eclipse project to our local Liberty server to see the application from Bluemix run locally.

4.2.1 Install and configure Eclipse

To perform these exercises, you need Eclipse installed on your computer (desktop or notebook). You also need a Liberty server that's configured for Eclipse to use it, the WebSphere Developer Tools for Eclipse, and the Bluemix Tools for Eclipse.

After this tutorial is completed, this local development environment will also be useful for developing, testing, and deploying future Liberty profile applications, including deploying them to Bluemix Liberty for Java run time.

Install Eclipse

To install Eclipse, you need to first install the Java JDK. Then, you can install and run Eclipse.

> **Note:** Although Eclipse runs with the JRE, the Java EE IDE needs the JDK to compile Java code.

1. Download and install the latest Java JDK. The download for the current version, JDK 8, is available at Java SE Development Kit 8 Downloads:

   http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

2. Download and install the latest version of Eclipse for Java EE. The download for the current version, codenamed *Mars*, is available at Eclipse IDE for Java EE Developers:

   http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/marsr

3. Run the IDE by following the directions in "Running Eclipse" in *Eclipse documentation - Current Release*:

   http://help.eclipse.org/mars/topic/org.eclipse.platform.doc.user/tasks/running_eclipse.htm

Eclipse is now installed so you can run it.

Install Liberty server

To test your app's database connection, you need to run it in a Liberty server. The Liberty server is convenient to install into Eclipse.

1. Download and install Liberty into Eclipse. The download is available at Download Liberty profile in Eclipse:

   `https://developer.ibm.com/wasdev/downloads/liberty-profile-using-eclipse/`

> **Documentation:** The Liberty profile is documented in "Liberty profile overview" and elsewhere in the *WebSphere Application Server (Distributed and IBM i operating systems), Version 8.5.5* documentation:
>
> `http://www.ibm.com/support/knowledgecenter/SSEQTP_8.5.5/com.ibm.websphere.wlp.doc/ae/cwlp_about.html`

The Liberty server is now installed. This document refers to the installation directory as *<LIBERTY_ROOT>*.

Install WebSphere Developer Tools

The WebSphere Developer Tools make it easier to use Eclipse to deploy applications to WebSphere Application Server, including the Liberty profile:

1. Download and install the WebSphere Developer Tools. The download is available at WebSphere Developer Tools for Eclipse:

   `https://developer.ibm.com/assets/wasdev/sites/wasdev/pages/WebSphere-Developer-Tools-for-Eclipse-Luna.html`

> **Documentation:** The WebSphere Developer Tools are documented in the *IBM WebSphere Developer Tools, Version 8.5.5* documentation:
>
> `http://www.ibm.com/support/knowledgecenter/SSHR6W_8.5.5/com.ibm.websphere.wdt.doc/topics/welcome_wdt.htm`

The WebSphere Developer Tools are now installed in Eclipse.

Install Bluemix Tools

The Bluemix Tools make it easier to use Eclipse to deploy applications to Bluemix, especially the Liberty for Java run time. You can download IBM Eclipse Tools for Bluemix at:

`https://developer.ibm.com/assets/wasdev/sites/wasdev/pages/IBM-Eclipse-Tools-for-Bluemix.html`

> **Documentation:** Bluemix Tools are documented in "Managing applications: Deploying apps with IBM Eclipse Tools for Bluemix" in the Bluemix documentation:
>
> `https://www.ng.bluemix.net/docs/manageapps/eclipsetools/eclipsetools.html`

4.2.2 Create simulated data center

To demonstrate Secure Gateway, we need a system of record running in a private data center. One of those is difficult to download and install. So for the purposes of these exercises, we simulate one using a MySQL database running in a virtual machine. We create a virtual machine, install Docker, install MySQL running in a Docker container, and initialize the database with some sample data that the Java application needs.

Create VM

We create a virtual machine to simulate a private data center. That VM can run anywhere if it has a public IP address (in any cloud provider or on your local computer). For these exercises, we use the virtual machines capability in Bluemix:

1. Create an SSH keypair, as documented in "*Creating web applications: Creating a virtual machine: Configuring an SSH security key in a VM: Creating an SSH security key to access a VM*" in the Bluemix documentation:

   `https://www.ng.bluemix.net/docs/virtualmachines/vm_index.html#vm_create_ssh_key`

   Specifically:

   – We call ours `todokey`.

   – In UNIX/Linux: Run **`ssh-keygen -t rsa -f todokey`**

   – In Microsoft Windows: Use PuTTY. You can download it here:

     `http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html`

2. Create a VM on Bluemix, as documented in "*Creating web applications: Creating a virtual machine: Creating a VM in a public cloud*" in the Bluemix documentation:

   `https://www.ng.bluemix.net/docs/virtualmachines/vm_index.html#vm_create_public_cloud`

   Specifically:

   – To create the VM, use the settings shown in Table 4-1. For the settings that have default values, use those values.

   – To specify the security key, select **Add Key** to import your `todokey` key.

Table 4-1 Virtual machine creation settings

Property	Value	Default
VM Cloud	IBM Cloud Public	default
Initial instances	1	default
Assign public IP addresses	Select (yes)	default
VM image	Ubuntu 14.04	default
VM group name	`AAA_To_Do`	
VM size	m1.small	default
Security Key	`todokey`	
Network	private	default

3. When the VM is created, make note of its public IP address, whose form is 129.xxx.xxx.xxx. (The other IP address, 192.168.xxx.xxx, is private.) We refer to this public IP address as <*virtual machine's IP address*>.
4. Log in to the virtual machine. The image has a user that is predefined for logging in remotely; it's `ibmcloud`. Use the authentication key specified when creating the VM:

   ```
   $ ssh -i todokey ibmcloud@<virtual machine's IP address>
   ```

You now have a running Ubuntu VM and you can log in to it.

Install Docker

As you see later, Secure Gateway requires a client to be installed in the data center, and the simplest client implementation to install is one that is already installed in a Docker client (that is, the Docker client for Secure Gateway). Therefore, our VM needs the Docker run time installed so that we can later install the Secure Gateway client. When you use Secure Gateway with a real system of record, you need to install Docker on a host in your data center so that it can run the Secure Gateway client.

Later, we also install a MySQL database. To simplify that installation, we use MySQL that's already installed in a Docker container. So another reason we need our VM to run the Docker run time is so that it can run the MySQL container.

For details, see the following sites:

- "Installing Docker on Ubuntu" explains how to install Docker:

  https://docs.docker.com/installation/ubuntulinux

- Bluemix also documents installing the Docker run time. See "Services: Secure Gateway: Docker" in the Bluemix documentation:

  https://www.ng.bluemix.net/docs/services/SecureGateway/sg_021.html#sg_003

Log in to your VM using Secure Shell (SSH), as described previously, and complete the following commands:

1. Before installing any software, ensure that your Ubuntu installation is running the latest version of all of its packages. Run this command:

   ```
   $ sudo apt-get update
   ```

2. Install the Docker package:

   ```
   $ wget -qO- https://get.docker.com/ | sh
   ```

3. Verify that Docker is installed correctly:

   ```
   $ sudo docker run hello-world
   ```

4. When hello-world runs correctly, part of the output should say:

   ```
   Hello from Docker.
   This message shows that your installation appears to be working correctly.
   ```

When you can run hello-world successfully, your VM has the Docker run time installed and running correctly.

Install and configure MySQL

To simulate an enterprise database of record, we use a MySQL database with a small, simple data set. To initialize that database, we need a schema file and a data file.

Create the database files

Log in to your VM by using SSH, as described previously, and complete the following commands:

1. Create the directory for the database initialization files:

   ```
   $ mkdir ~/liberty-sql
   $ cd ~/liberty-sql
   ```

2. By using your favorite Linux text editor (such as `nano` or `vi`), create the file `todo-schema.sql` and insert the contents shown in Example 4-1.

   Example 4-1 Contents of todo-schema.sql

   ```
   DROP SCHEMA IF EXISTS todo;
   CREATE SCHEMA todo;
   USE todo;

   CREATE TABLE `TODOLIST` (
      `L_ID` INT(8) DEFAULT NULL,
      `C_NAME` VARCHAR(254) DEFAULT NULL
   ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
   ```

3. Create the file `todo-data.sql` and insert the contents shown in Example 4-2.

   Example 4-2 Contents of todo-data.sql

   ```
   USE todo;

   INSERT INTO `TODOLIST` (`L_ID`,`C_NAME`) VALUES (1, "sample entry #1");
   INSERT INTO `TODOLIST` (`L_ID`,`C_NAME`) VALUES (2, "sample entry #2");
   INSERT INTO `TODOLIST` (`L_ID`,`C_NAME`) VALUES (3, "sample entry #3");
   ```

4. Confirm that you have the files in the correct directory. It should look like this:

   ```
   ibmcloud@aaa-to-do-12345678:~$ ls -l /home/ibmcloud/liberty-sql/
   -rw-r--r-- 1 ibmcloud ibmcloud 227 Jan 1 12:00 todo-data.sql
   -rw-r--r-- 1 ibmcloud ibmcloud 190 Jan 1 12:00 todo-schema.sql
   ```

Tip: You can also **`cat`** each file to ensure that its contents look correct.

You now have the schema and data file that is needed to initialize the database when creating its Docker container.

Install the database

While still logged in to your VM using SSH:

1. Create the MySQL container instance and load the sample data from the two initialization files with this command:

   ```
   $ sudo docker run -d --name mysql-tutum -p 3306:3306 -v
   /home/ibmcloud:/home/ibmcloud -e MYSQL_PASS=passw0rd -e
   STARTUP_SQL="/home/ibmcloud/liberty-sql/todo-schema.sql
   /home/ibmcloud/liberty-sql/todo-data.sql" tutum/mysql
   ```

   Where:

   – `–d` runs the container in the background, not interactively

   – `mysql-tutum` is the name to give the container that is created from the image

- `3306:3306` forwards the MySQL port to make it accessible from the host OS's IP address
- `/home/ibmcloud:/home/ibmcloud` binds the directory to make the directory on the host OS available within the container
- `MYSQL_PASS` sets the password of the database's main user, in this example to `passw0rd`
- `STARTUP_SQL` tells the container to run the SQL files in the order specified via the space-separated list
- `tutum/mysql` is the name of the Docker image to create the container from

You now have a running Docker container named `mysql-tutum`. That container has a MySQL database server running in it, bound to port `3306`. The database server contains a database named `todo` that contains a table named `TODOLIST` that contains the sample data for a set of items in a To Do list.

4.2.3 Create sample app

This tutorial needs a sample application, a Java app that connects to a SQL database. Rather than create one from scratch, we use one that's built into Bluemix, created by the Java DB Web Starter boilerplate.

In the Bluemix Dashboard, perform the following steps:

1. Create a new space, name it **hybrid-cloud**.
2. Create a new instance of the **Java DB Web Starter** boilerplate.
   a. Select **Catalog**.
   b. In the Boilerplates section of the catalog, select **Java DB Web Starter**.
   c. On the boilerplate's Create page, specify the name as `AAA-java-sql`, where AAA is your initials or some other ID to make the application's name unique.
   d. Leave the other settings with their default values and press **Create**.
3. When the boilerplate's application run time has started, run the application:
   a. Select the link shown next to your app that is running, such as: `http://AAA-java-sql.mybluemix.net`
   b. A new browser window opens displaying the Java DB Web Starter home page. Experiment with adding, changing, and removing items in the To Do list. These items are stored in the SQL database.
4. Back in the Dashboard, it should still display the page for your application with Start Coding selected:
   a. Select **Download Starter Code** to download `AAA-java-sql.zip` to your local computer.
   b. Select **Download CF Command Line Interface** to download and install the CF CLI.

You've now seen the Java DB Web Starter app run in Bluemix and have downloaded its code. For good measure, we also installed the Cloud Foundry CLI, which later we'll use to upload your modified application back into Bluemix.

4.2.4 Import sample app

The starter code from Bluemix is conveniently packaged as an Eclipse project. Import that project into Eclipse so that you can review the sample app's code.

In Eclipse, import the **Java DB Web Starter** project, as documented in "Importing existing projects" in *Eclipse documentation - Current Release*:

http://help.eclipse.org/mars/topic/org.eclipse.platform.doc.user/tasks/tasks-importproject.htm

Specifically:

- Specify **Select archive file**.
- Select the starter code file you downloaded in 4.2.3, "Create sample app" on page 78, `AAA-java-sql.zip`.

This creates the liberty-IRDS project.

4.2.5 Deploy sample app to Liberty server

Not only can we edit the sample app's code in Eclipse, we can also use Eclipse to deploy the sample app to our Liberty server and run the sample app.

In Eclipse:

1. Add the sample app's project to the Liberty server, as documented in "Adding projects to a server" in *Eclipse documentation - Current Release*:

   http://help.eclipse.org/mars/topic/org.eclipse.wst.server.ui.doc.user/topics/twaddprj.html

   Specifically:

   - The project to add is liberty-IRDS.
   - The server to add it to is WebSphere Application Server Liberty Profile.
2. The Servers view now shows the project associated with the server.

For more information: For details, see "Servers view" in *Eclipse documentation - Current Release*:

http://help.eclipse.org/mars/topic/org.eclipse.wst.server.ui.doc.user/topics/rwrcview.html

At this point, we could start the server and start the app in the server, then, connect to the app and test it. There's not much point in doing that yet, however, because the app isn't connected to a database, the section of the web page that's supposed to display the To Do list in the database instead simply says "Error."

4.3 Connect directly to database

Now you'll set up the local Liberty server to connect to MySQL and test the app using that connection.

4.3.1 Add the MySQL driver jar

First: Download MySQL's JDBC driver, which is called *Connector/J*:

1. Go to the MySQL Download Connector/J page.

   https://dev.mysql.com/downloads/connector/j/

2. Download the archive for the latest version of the platform independent driver (*not the Microsoft Windows driver, since we will upload this driver to Bluemix*), unpack it, and get the JAR file. In this example, we downloaded v5.1.35, so the driver jar is:

   `mysql-connector-java-5.1.35-bin.jar`

3. In the directory structure of the Liberty server, add the MySQL driver jar, `mysql-connector-java-5.1.35-bin.jar`, to the server's `lib` directory by using the following steps:

   a. Go to the directory where your Liberty server is installed on disk, *<LIBERTY_ROOT>*.

   b. Go to the subdirectory `wlp/usr/servers/defaultServer`.

   c. Go to the subdirectory `lib`. (If `defaultServer/lib` doesn't exist, create it.)

   d. Copy or move the MySQL driver jar, `mysql-connector-java-5.1.35-bin.jar`, into `defaultServer/lib`.

The server's libraries now contain the database driver.

4.3.2 Configure server for MySQL

Modify the server configuration of the Liberty server, *WebSphere Application Server Liberty Profile*, to add a data source for the MySQL database and to set the app's context root.

First, let's review the application dependency that requires the data source.

In Eclipse, following these steps:

1. In the Enterprise Explorer, go to your project, `liberty-IRDS`.

   In that project, the `Java Resources` folder contains the application source code.

2. Go to `/liberty-IRDS/src/META-INF/persistence.xml` and open the file.
3. Notice the line defining a JTA data source, shown in Example 4-3.

Example 4-3 Defining a JTA data source

```
jta-data-source>java:comp/env/jdbc/mydbdatasource</jta-data-source>
```

This line makes an important declaration.

It binds the application's persistence unit, `openjpa-todo`, to a data source whose JNDI name is `jdbc/mydbdatasource`.

Therefore, the server needs to define a data source registered in JNDI as `jdbc/mydbdatasource`.

Next, add that data source to the server configuration.

In Eclipse, follow these steps:

1. In the Servers view, open the **Server Configuration [server.xml]** file associated with the server.
2. In the Server Configuration view, use either the Designer tab or the Source tab to add the Data Source and Shared Library shown in Example 4-4.
   - Notice that the library's fileset is configured to locate the MySQL driver jar.

Example 4-4 Data Source and Shared Library configuration

```
<dataSource jndiName="jdbc/mydbdatasource">
    <jdbcDriver libraryRef="mysql-connector"/>
    <properties URL="jdbc:mysql://<hostname>:<port>/todo?relaxAutoCommit=true" user="<user>"
password="<password>"/>
</dataSource>
<library id="mysql-connector" name="MySQL Connector" description="MySQL JDBC Driver">
    <fileset id="mysql-connector-jar" dir="${server.config.dir}/lib"
includes="mysql-connector-java-*.jar"/>
</library>
```

   In the URL property's value, substitute these variables:

   - *<hostname>*: The IP address of the virtual machine hosting the MySQL database: *<virtual machine's IP address>*
   - *<port>*: Set when we created the MySQL container: `3306`.
   - *<user>*: The default user in the MySQL container: `admin`
   - *<password>*: Set when we created the MySQL container: `passw0rd`

3. To make the app accessible via the root directory (and not a subdirectory like `liberty-IRDS`), add the web application configuration shown in Example 4-5.

Example 4-5 Web application configuration

```
<webApplication id="liberty-IRDS" name="liberty-IRDS" location="liberty-IRDS.war"
context-root="/"/>
```

4. As explained in the Console view, the URL for accessing the app is `http://localhost:9080/`.

The server is now configured with a data source for accessing the database, and the application's context root is set.

4.3.3 Test app using direct connection

Now run the app and confirm that it connects to MySQL correctly.

In Eclipse, follow these steps:

1. In the Servers view, confirm the server and app are started and synchronized. If not, start the server and confirm the app starts as well.

   The following items should show as started and synchronized:

   - WebSphere Application Server Liberty Profile
   - liberty-IRDS

2. The app runs a web GUI. To open the GUI, in Eclipse or in an external web browser, go to `http://localhost:9080/`.
3. The app's Java DB Web Starter web page opens. The area where the To Do list is displayed says "Please wait while the database is being initialized ..." while the database is accessed.
4. If the app connects to the database successfully, the To Do List then completes. The default items are sample entry #1, #2, and #3.

   If the app cannot connect and load the data successfully, the To Do area displays "Error."

The app is running locally using the MySQL database in the virtual machine.

4.4 Connect via Secure Gateway

Now let's configure a Secure Gateway that connects to the MySQL database, and configure the app to use that.

4.4.1 Configure a gateway and destination

We create an instance of the Secure Gateway service, then use it to configure a gateway and configure a destination in that gateway. The destination is the MySQL database running in the virtual machine.

For details, see "Services: Secure Gateway: Creating a Secure Gateway by using the Bluemix UI" in the Bluemix documentation:

`https://www.ng.bluemix.net/docs/services/SecureGateway/sg_022.html#sg_009`

In the Bluemix dashboard, following these steps:

1. Select your space, **hybrid-cloud**.
2. In the main view for the space, select **Use Services or APIs**.
3. In the Service Catalog, select **Secure Gateway**.
4. In the creation page for Secure Gateway, keep the default settings and select **Use**.
5. In the Secure Gateway page, select **Add Gateway**.
6. In the Add Gateway page, set the name to My Data Center and select **Add Destinations**.
7. In the Add Destinations page, add a destination with the settings shown in Table 4-2.
   - Complete the values and then press the plus sign (**+**) on the right.
   - The IP address and port are the same settings that you specified in the data source's URL in the Liberty server's configuration.

Table 4-2 Secure Gateway destination settings

Property	Value
Destination name	To Do Database
Hostname or IP address	*<virtual machine's IP address>*
Port	3306
Transport	TCP

8. A destination named To Do Database is now listed. Optional: Select destination information (the button looks like the letter i with a circle around it) to display the destination's configuration, as shown in Table 4-3 on page 83.

Table 4-3 Destination information

Property	Value
Name	To Do Database
Destination ID	aaaA0AAAA00_vfY
Cloud Host: Port	cap-sg-prd-1.integration.ibmcloud.com:15000
Destination Host: Port	<*virtual machine's IP address*>:3306
Created by	John Doe at 7/1/2015, 12:05:00 PM
Last modified by	John Doe at 7/1/2015, 12:05:01 PM

The gateway is a proxy that maps the cloud host and port to the destination host and port:

– The destination's host and port are the values that you specified when you created the destination. This is the target that the gateway connects to, the MySQL database.

– The cloud host and port are the values that an application uses to access the destination remotely.

You have now configured a gateway and destination for accessing the database.

4.4.2 Configure the gateway client

The gateway needs a client deployed in the data center hosting the database. The gateway client needs network access to both the database running in the data center and to the gateway running in Bluemix. Typically, this means that the gateway client should be installed on a host that is connected to the same network segment (that is, subnet or VLAN) as the host of the database, and that is connected to the Internet (or some other network connection to Bluemix).

The gateway client can run in a Docker container or in IBM DataPower. In this example, we use the Docker container. The virtual machine simulating our data center already has the Docker run time installed (we installed it in "Install Docker" on page 76), so we just need to deploy the Docker container to that Docker run time.

For details:

- We already installed the Docker run time in this tutorial (see "Install Docker" on page 76). For more details about installing the Docker run time, see "*Services: Secure Gateway: Docker*" in the Bluemix documentation:

  https://www.ng.bluemix.net/docs/services/SecureGateway/sg_021.html#sg_003

- Installing the Docker container for the Secure Gateway client is documented as part of setting up Secure Gateway. See "*Services: Secure Gateway: Creating a Secure Gateway by using the Bluemix UI*" in the Bluemix documentation:

  https://www.ng.bluemix.net/docs/services/SecureGateway/sg_022.html#sg_009

Starting the gateway client

First, get the Docker connect command for the gateway client. In the Bluemix dashboard, follow these steps:

1. Go to the Secure Gateway Dashboard for your space's Secure Gateway service instance.
2. One gateway is listed, named Virtual Machine. Optional: Select gateway information (the button looks like the letter i with a circle around it) to display the gateway's configuration, as shown in Table 4-4.

   Table 4-4 Gateway information

Property	Value
Name	Virtual Machine
Gateway ID	aaaA0AAAA00_prod_ng
Created by	John Doe at 7/1/2015, 12:00:00 PM
Last modified by	John Doe at 7/1/2015, 12:00:01 PM

3. From the Gateway Options menu (the button looks like a gear), select **Edit / View**.
4. In the Gateway page, notice that one destination is listed, To Do Database. Select **Connect Gateway**. The page displays: *How would you like to connect this new gateway?* With Docker selected, the page displays the command to install and run the gateway's Docker client, as shown in Example 4-6:
   – Notice that the parameter is the gateway's ID, shown in Table 4-4. This tells Bluemix what gateway the client needs to connect to.
   – Make a note of this Docker command. You need to run it when you're logged in to the VM.

Example 4-6 Command to run a gateway's Docker client

```
docker run -it ibmcom/secure-gateway-client aaaA0AAAA00_prod_ng
```

Second, install and run the gateway client in the virtual machine. From a command prompt:

1. Log into the virtual machine. Use the authentication key specified when creating the VM:

   ```
   $ ssh -i todokey ibmcloud@<virtual machine's IP address>
   ```

2. Verify that the Docker run time is running correctly:

   ```
   $ sudo docker run hello-world
   ```

3. Run the command that Bluemix gave us to install and run the gateway Docker client:

   ```
   $ sudo docker run -it ibmcom/secure-gateway-client aaaA0AAAA00_prod_ng
   ```

The gateway client is now running. It says it's connected, and the Gateway page says the gateway is connected. The shell used to run the client is now attached to the container and running the gateway client's shell, which displays when connections are open and closed.

Using the gateway client CLI

Optional: This section is a brief detour to look at the gateway client CLI. There are no tutorial steps in this section.

The command to run the gateway client container also connects the user to the gateway client CLI. The CLI shows logging information, such as when connections from the gateway are open and closed, as shown in Example 4-7 on page 85.

Example 4-7 Gateway client logging information

```
[2015-01-01 12:00:00.000] [INFO] Connection #1 is being established to 129.40.111.222:3306
[2015-01-01 12:01:00.000] [INFO] Connection #1 to 129.40.111.222:3306 was closed
[2015-01-01 12:02:00.000] [INFO] Connection #2 is being established to 129.40.111.222:3306
[2015-01-01 12:03:00.000] [INFO] Connection #2 to 129.40.111.222:3306 was closed
```

The CLI also has commands for managing the client. For the list of commands, run the help command:

```
cli> help
```

For more information: These commands are also documented in “Services: Secure Gateway: Secure Gateway client interactive command-line interface” in the Bluemix documentation:

https://www.ng.bluemix.net/docs/services/SecureGateway/sg_022.html#sg_010

When you log out of the virtual machine, the gateway client keeps running so that applications can continue to connect to destinations. How do you know that the client is still running? On the Bluemix end, in the Bluemix dashboard, the gateway view shows whether the gateway is connected.

On the client end, use Docker commands to see what the client container’s status is.

For more information: Docker commands are documented in “Using the command line” in the Docker documentation:

https://docs.docker.com/reference/commandline/cli

To see if the gateway client is running, run Docker’s process status command (Example 4-8).

Example 4-8 Docker process status command

```
$ docker ps
CONTAINER ID        IMAGE                          STATUS              NAMES
aa000000000a        ibmcom/secure-gateway-client   Up 4 days           gateway_client
a000aa00aaaa        tutum/mysql                    Up 2 weeks          mysql-tutum
```

This example shows a couple of details of interest:

- The container for the gateway client is the one named `gateway_client`. We know this because the container `gateway_client` was created from the image `ibmcom/secure-gateway-client`.
- The container `gateway_client` is running. Its status says “Up.”

To see all of the Docker containers, whether they're running or stopped, use the **all** option, as shown in Example 4-9.

Example 4-9 Docker process status all command

```
$ docker ps -a
CONTAINER ID        IMAGE                          STATUS                     NAMES
aa000000000a        ibmcom/secure-gateway-client   Exited (0) 7 seconds ago   gateway_client
a0a0000a0aa0        hello-world                    Exited (0) 2 weeks ago     serene_newton
a000aa00aaaa        tutum/mysql                    Up 2 weeks                 mysql-tutum
```

This example shows a couple of details of interest:

- The container gateway_client is stopped. Its status says "Exited."
- The gateway_client container stopped cleanly. Its exit code is 0.

To start an existing container, run Docker's start command:

```
$ docker start gateway_client
```

To stop a running container, run Docker's stop command:

```
$ docker stop serene_newton
```

Notice that example shows stopping the hello-world container. It doesn't show using the stop command to stop the gateway client's container. Stopping the gateway client by stopping its container is the equivalent of running kill -9 on the client process, which stops the client immediately and doesn't allow the client to stop cleanly. The result is a 137 exit code, as shown in Example 4-10.

Example 4-10 Docker stop causes error code 137

```
$ docker start gateway_client
gateway_client
$ docker stop gateway_client
gateway_client
$ docker ps -a
CONTAINER ID        IMAGE                          STATUS                     NAMES
aa000000000a        ibmcom/secure-gateway-client   Exited (137) 2 seconds ago gateway_client
a0a0000a0aa0        hello-world                    Exited (0) 2 weeks ago     serene_newton
a000aa00aaaa        tutum/mysql                    Up 2 weeks                 mysql-tutum
```

To stop a gateway client cleanly, use its CLI. If you lost the CLI connection to the gateway client, you can connect to the CLI again by running Docker's attach command:

```
$ docker attach gateway_client
```

Example 4-11 shows how to start the gateway client's container, connect to the client's CLI, and cause the client to shut down cleanly.

Example 4-11 Cleanly stopping a gateway client

```
$ docker start gateway_client
gateway_client
$ docker attach gateway_client

cli> quit
[2015-01-01 12:00:00.000] [FATAL] About to exit with code: 0
```

Now we've seen how to manage the gateway client using Docker commands and the gateway client CLI.

4.4.3 Configure application server for Secure Gateway

Now that we have our Secure Gateway service instance configured with a gateway and destination to connect to our MySQL database, we need to modify our app to use that gateway destination. Recall that the destination's settings, shown in Table 4-3 on page 83, include the value for Cloud Host: Port, which in this example is:

```
cap-sg-prd-1.integration.ibmcloud.com:15000
```

This is the URL that the app uses to access the database via the gateway destination.

In Eclipse, as we did in 4.3.2, "Configure server for MySQL" on page 80, modify the server configuration of the Liberty server to update the data source to use the gateway destination's URL.

1. In the Servers view, open the **Server Configuration [server.xml]** file that is associated with the server.
2. In the Server Configuration view, use either the Designer tab or the Source tab to edit the URL in the data source, this time with these values:
   a. *<hostname>*: The gateway destination's Cloud Host: `cap-sg-prd-1.integration.ibmcloud.com`
   b. *<port>*: The gateway destination's Cloud Port: 15000
   c. *<user>*: Unchanged: `admin`
   d. *<password>:* Unchanged: `passw0rd`

As the console shows, the server automatically updates with the new configuration.

4.4.4 Test app using gateway connection

Now let's run the local app and confirm that it correctly connects to MySQL via the gateway:

1. To run the app, go to `http://localhost:9080/`.
2. The app's Java DB Web Starter window opens and the To Do List data is displayed.
3. In the shell logged in to the virtual machine, the logging in the gateway client's CLI shows that a new connection was opened.

This confirms that the local app is able to connect to the database via the gateway.

4.5 Deploy to Bluemix

Now that we have the local app working correctly using the gateway, let's deploy it to Bluemix and confirm that it works correctly from there as well.

4.5.1 Deploy server and app to Bluemix

We need to not only deploy the app (its `libertyDBApp.war` file), but also its server configuration (the server's `server.xml` file and the database driver, mysql-connector-java-5.1.35-bin.jar). How can the server configuration be deployed along with the app?

Packaging a Liberty server

To deploy not just an app but its server configuration as well, we package the server with the app and deploy the package. There are two procedures for doing this:

- Command line
- Eclipse tools

Command line

This procedure is documented in “Creating web applications: Creating apps with Liberty for Java: Options for pushing Liberty applications” in the Bluemix documentation, under “Packaged Server”:

https://www.ng.bluemix.net/docs/starters/liberty/index.html#optionsforpushinglibertyapplications

The procedure uses a Liberty server command to package the server into an archive. That procedure is documented in “Packaging a Liberty profile server from the command line” in the *WebSphere Application Server (Distributed and IBM i operating systems), Version 8.5.5* documentation:

http://www.ibm.com/support/knowledgecenter/SSEQTP_8.5.5/com.ibm.websphere.wlp.doc/ae/twlp_setup_package_server.html

That procedure also uses the CF CLI, which you installed in 4.2.3, “Create sample app” on page 78 when you downloaded the sample app. Deploying applications using the CF CLI is documented in “Managing applications: Deploying applications by using the cf command” in the Bluemix documentation:

https://www.ng.bluemix.net/docs/manageapps/deployingapps.html#dep_apps

Eclipse tools

The IBM Eclipse Tools for Bluemix can also be used to deploy a packaged server. This procedure is documented in “CLI and Dev Tools: Deploying apps with IBM Eclipse Tools for Bluemix: Packaged server support” in the Bluemix documentation:

https://www.ng.bluemix.net/docs/manageapps/eclipsetools/eclipsetools.html#packagedserversupport

Perform the package and push

Here, we show how to package and push the server using the command line. This procedure shows more clearly the steps that the Eclipse tools perform for you.

In the directory structure of the Liberty server, package the server and then push it to Bluemix:

1. Go to the directory where your Liberty server is installed on disk, <*LIBERTY_ROOT*>.
2. Run the package command to package the server.

   ```
   $ wlp/bin/server package defaultServer --include=usr
   ```

3. If you have not already, log in the CF CLI to your Bluemix account, org, and space. Run the following command:

   ```
   $ cf login -a <API_URL> -u <username> -o <org> -s <space>
   ```

Where:

a. *<API_URL>*: The URL of the Cloud Foundry provider, which is Bluemix: `https://api.ng.bluemix.net`

b. *<username>*: Your email address

c. *<org>*: Your email address (assuming that your org in Cloud Foundry is named after your user)

d. *<space>*: The space in Cloud Foundry that you want your app deployed to: `hybrid-cloud`

4. Then, use the CF CLI to push the packaged server to Bluemix.

```
$ cf push <appname> -p wlp/usr/servers/defaultServer/defaultServer.zip
```

Where:

<appname>: The name you want your app to have in Bluemix. Needs to be unique so that its route will also be unique.

It will typically take the app 1 - 2 minutes to upload and several minutes to start. The deployment has finished when the CF CLI says the app has started and when the Bluemix dashboard says the app is running.

Review the deployed app

Optional: When the app is running in Bluemix, we can review it to see that our server configuration has been uploaded successfully.

In the Bluemix dashboard, follow these steps:

1. Navigate to the page for your application, *<appname>*, and select **Files and Logs**.
2. Navigate to `app/wlp/usr/servers/defaultServer/server.xml` and select the file.
3. Confirm that the `server.xml` file contains the customizations that we made to the server configuration back in 4.3.2, "Configure server for MySQL" on page 80 and 4.4.3, "Configure application server for Secure Gateway" on page 87, such as:
   a. A data source with the JNDI name `jdbc/mydbdatasource`.
   b. That the data source's URL property is in part `cap-sg-prd-1.integration.ibmcloud.com`.
   c. That the data source links to a library, `mysql-connector`, with a fileset for matches to `mysql-connector-java-*.jar`.
   d. A web application whose context root is the root directory.
4. Navigate to `app/wlp/usr/servers/defaultServer/lib`. Confirm that this directory contains the MySQL driver, such as `mysql-connector-java-5.1.35-bin.jar`.

This confirms that not only was the app deployed to Bluemix, but that its server configuration was also deployed.

4.5.2 Test app in Bluemix

Now run the app in Bluemix and confirm that it correctly connects to MySQL via the gateway:

1. To run the app, go to `http://<appname>.mybluemix.net/`.
2. The app's Java DB Web Starter window opens and the To Do List data is displayed.

3. In the shell logged in to the virtual machine, the gateway client shows that a new connection was opened.

This confirms that the Bluemix app is able to connect to the database via the gateway.

4.6 Conclusion

This tutorial has shown how to perform the following functions:

- Load an application into Eclipse and deploy it to a Liberty server.
- Configure the server with a data source so that the application can access its database.
- Configure the data source with a database that is running remotely and test that locally.
- Configure the data source to use Secure Gateway to access the remote database and test that locally.
- Deploy the application and its server configuration into Bluemix and test that.

With this knowledge, you can configure Secure Gateway to access systems of record in your data center, and configure your applications in Bluemix to use Secure Gateway.

Connecting IBM Containers with on-premises Docker

This chapter introduces the Bluemix container offering named *IBM Containers*. The chapter first describes the basic actions that you need to perform when working with containers. Then, it explains how IBM Containers can be integrated with your on-premises Docker environment.

At the end of the chapter, you will know how to set up a hybrid application running on IBM Container and your on-premises Docker environment.

This chapter has the following sections:

> **Recording of this scenario:** You can find the recording of this scenario at:
>
> `https://youtu.be/9XbzUqCvl3I`

© Copyright IBM Corp. 2015. All rights reserved.

5.1 IBM Containers overview

IBM Containers service enables the use of containers in a Bluemix environment.

5.1.1 IBM Containers architecture

IBM Containers offers managed-container services based on Docker technology. Containers that are created with IBM Containers can interact with other Bluemix services, such as:

- Cloud Foundry Apps
- Services & APIs
- Virtual Machines

IBM Containers architecture, as represented in Figure 5-1 on page 93, is composed of three main elements:

- *Private Bluemix repositories* for storing container images.

  > **Note:** In the rest of the book, this repository is referenced as *Bluemix Registry.*

- *Storage volumes* to ensure data persistency after the container deletion.
- *Containers* to run workloads. They can be ran in two different modes:
  - *Single container* mode is the standard form factor. Each container is executed independently of the others.
  - *Container group* mode runs multiple containers as an entity. Groups can scale up and down in size and have an *auto-recovery* feature to create a cluster of containers.

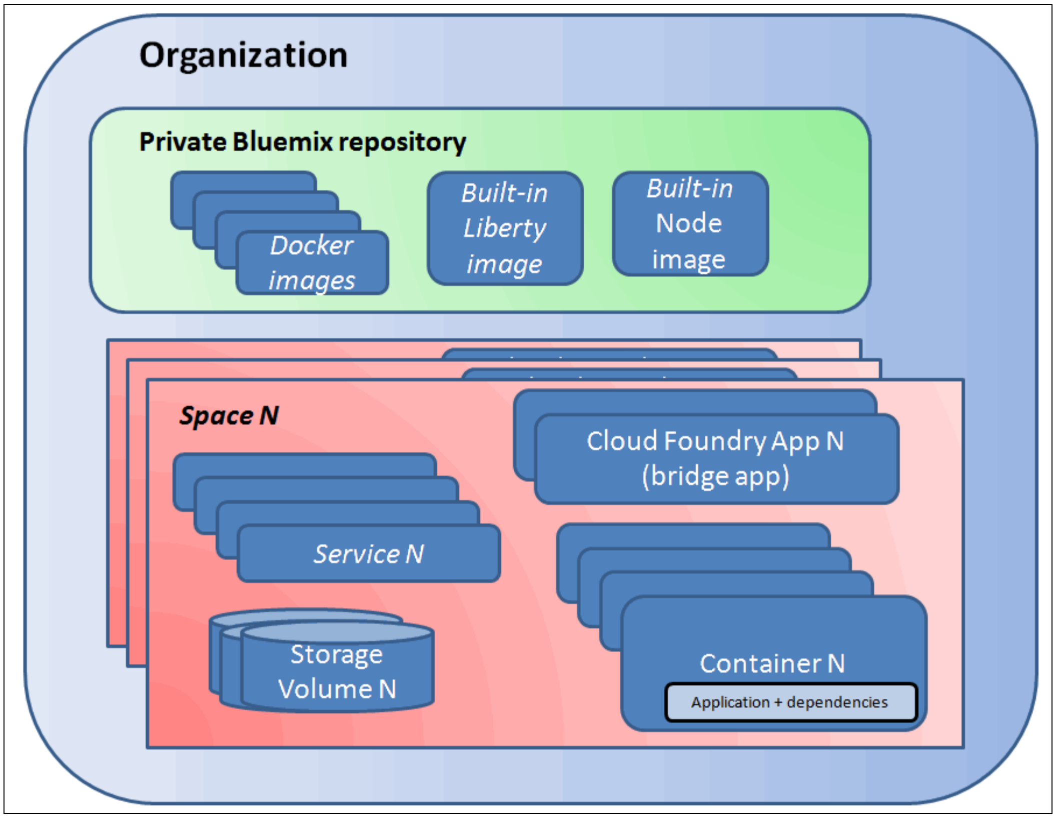


Figure 5-1 IBM Containers overall architecture in Bluemix

Single containers are similar to standard Docker containers. Thus, the following actions can be completed on IBM Containers single containers:

- Exposing ports of containers
- Linking containers internally
- Binding external IP addresses to containers

Container groups however are a concept specific to IBM Containers. A container group enables the creation of a scalable cluster of containers with high-availability thanks to the auto-recovery feature and with automatic load-balancing using Bluemix load-balancers.

The following actions can be performed on IBM Container groups:

- Exposing a specific port on all containers
- Scaling up or down the number of containers in a group
- Mapping a container group to a route for external access via Bluemix load-balancers

On both types of IBM Containers, the following actions can be performed:

- Binding containers to a Bluemix application
- Displaying containers monitoring data
- Displaying and querying containers logging data

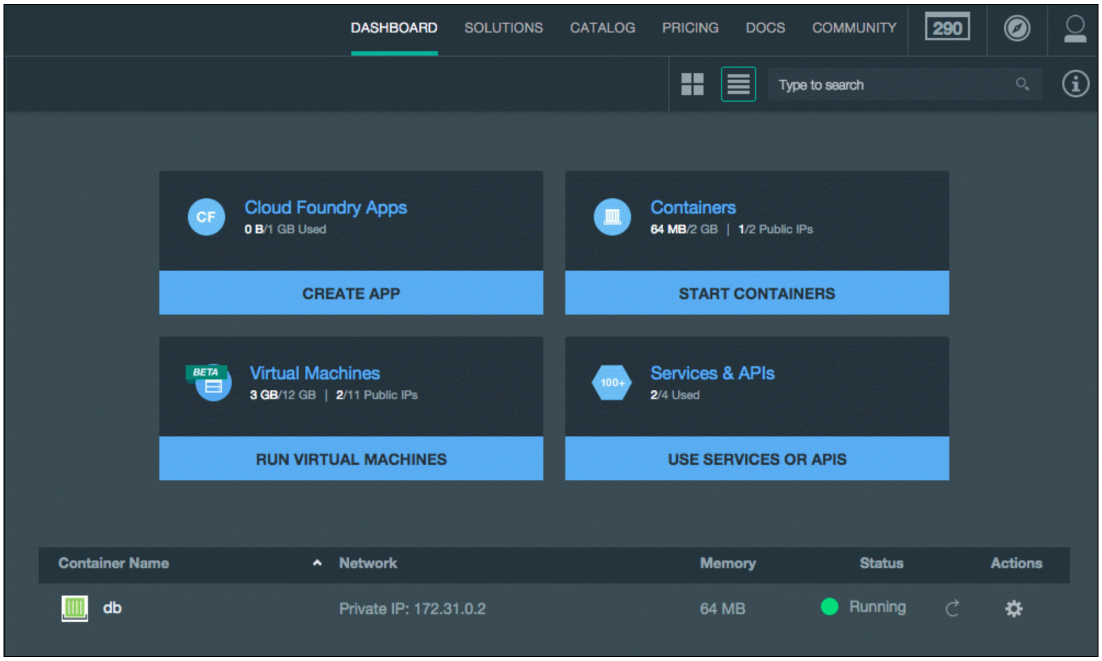
Note: Because this book focuses on integrating with Bluemix, we do not get into details about IBM Containers architecture. For more information, see the IBM Containers documentation:

`https://www.ng.bluemix.net/docs/containers/container_index.html#container_ov`

5.1.2 Creating your Bluemix registry

To start working with IBM Containers, log on to Bluemix and complete the following steps to create your Bluemix registry:

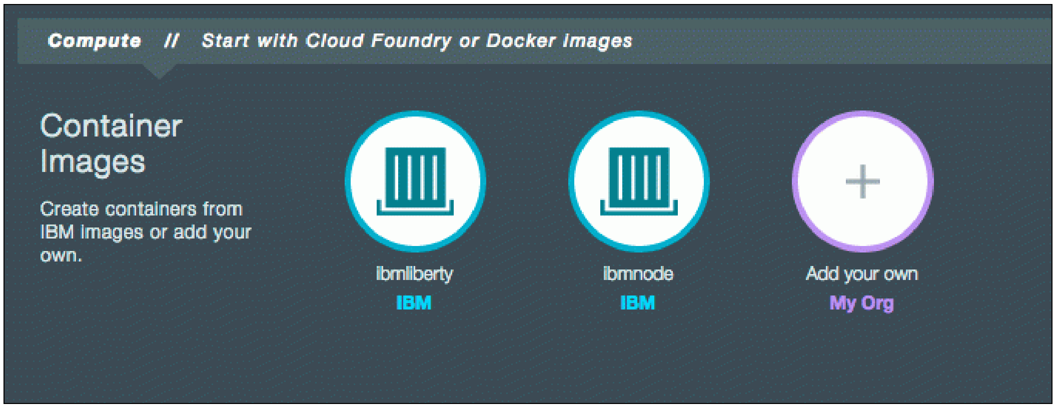
1. Click **Dashboard** to display a status of your currently running services (Figure 5-2):
   – Cloud Foundry Apps
   – Containers
   – Virtual Machines
   – Services & APIs



Container Name	Network	Memory	Status	Actions
db	Private IP: 172.31.0.2	64 MB	● Running	

Figure 5-2 Bluemix Dashboard

2. Click **Start Containers** to display the available images, as shown in Figure 5-3.

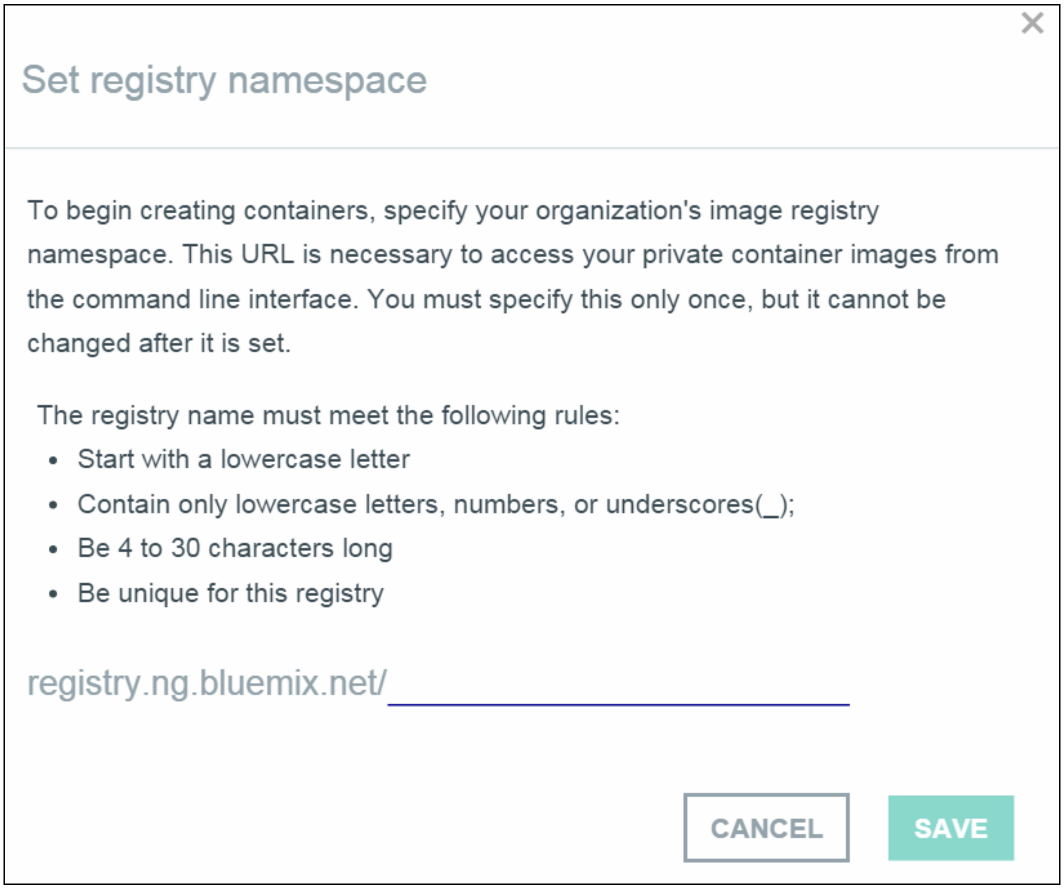


Create containers from IBM images or add your own.

Figure 5-3 IBM Containers image list

3. Click **ibmnode**. As first-time users of IBM Containers, Bluemix requires the creation of your own Bluemix registry as shown in Figure 5-4. Enter a name matching the rules listed on the panel and click **SAVE**.

> **Tip:** Choose the name of your registry cautiously because you are not able to change it afterward.



Set registry namespace

To begin creating containers, specify your organization's image registry namespace. This URL is necessary to access your private container images from the command line interface. You must specify this only once, but it cannot be changed after it is set.

The registry name must meet the following rules:

- Start with a lowercase letter
- Contain only lowercase letters, numbers, or underscores(_);
- Be 4 to 30 characters long
- Be unique for this registry

registry.ng.bluemix.net/

Figure 5-4 Bluemix registry creation

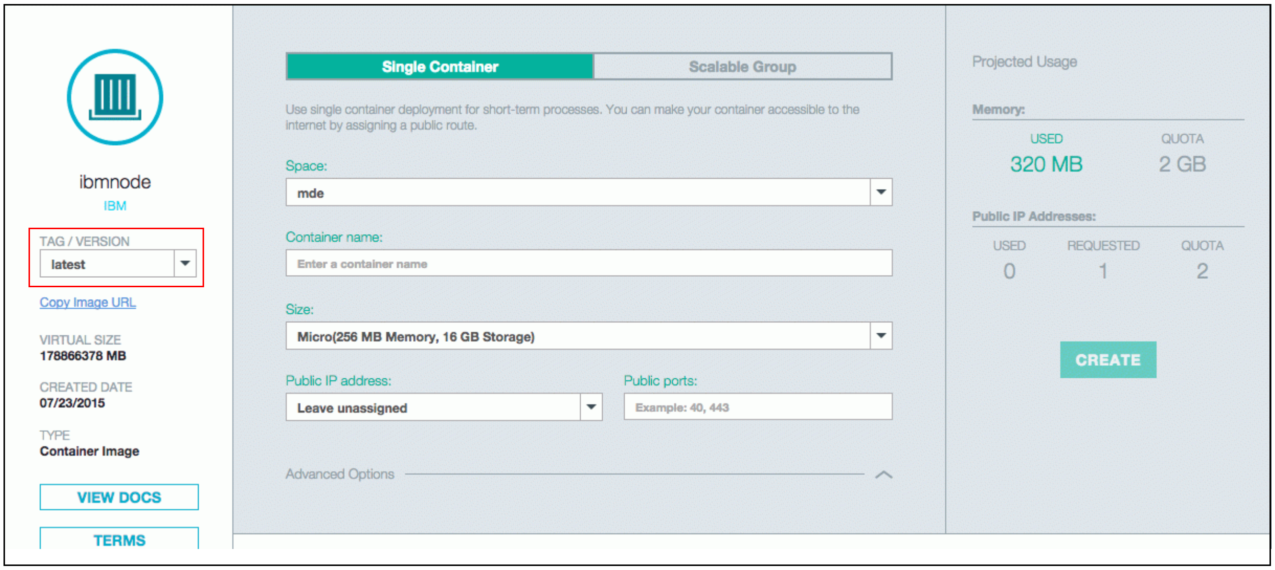
> **Note:** As stated in Figure 5-4, the registry is unique by organization and shared among users across spaces.

4. Your registry is now created. You can continue to 5.1.3, “Running a single container” on page 96 to learn how to run a single container.

5.1.3 Running a single container

To create a new single container, log on to Bluemix and complete the following steps:

1. Click **Dashboard** → **Start Containers** → **ibmnode**. The list of options is displayed as shown in Figure 5-5.



ibmnode
IBM

TAG / VERSION: latest

VIRTUAL SIZE: 178866378 MB

CREATED DATE: 07/23/2015

TYPE: Container Image

Use single container deployment for short-term processes. You can make your container accessible to the internet by assigning a public route.

Space: mde

Container name: Enter a container name

Size: Micro(256 MB Memory, 16 GB Storage)

Public IP address: Leave unassigned

Public ports: Example: 40, 443

Projected Usage

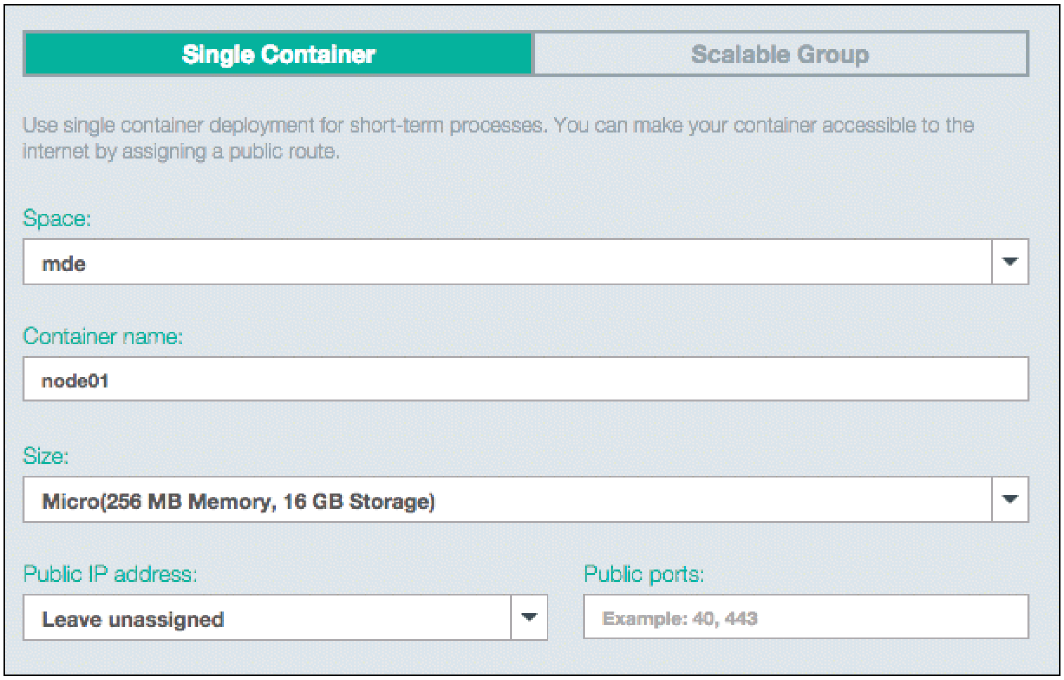
Memory:	USED	QUOTA
	320 MB	2 GB

Public IP Addresses:	USED	REQUESTED	QUOTA
	0	1	2

Figure 5-5 Single container creation form

Tip: When needing to create a container with a specific version tag, select the version required in the **TAG / VERSION** list on the left of the panel, as shown in Figure 5-5.

2. Ensure that **Single Container** is selected in green. Leave the default “Space” selected. And enter `node01` as the “Container name” as shown in Figure 5-6 on page 97.



Use single container deployment for short-term processes. You can make your container accessible to the internet by assigning a public route.

Space: mde

Container name: node01

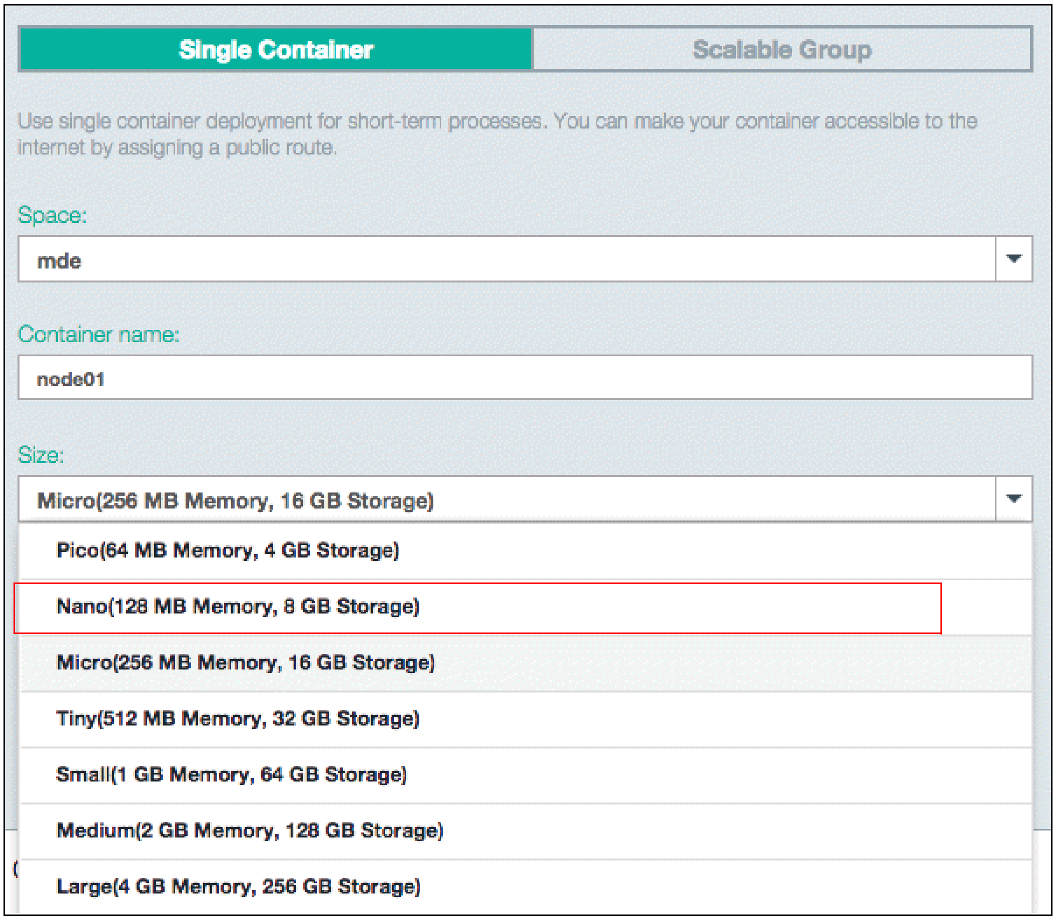
Size: Micro(256 MB Memory, 16 GB Storage)

Public IP address: Leave unassigned

Public ports: Example: 40, 443

Figure 5-6 Single container filled creation form

3. From the Size list, select **Micro** as shown in Figure 5-7 on page 98.


Single Container
Scalable Group

Use single container deployment for short-term processes. You can make your container accessible to the internet by assigning a public route.

Space: mde

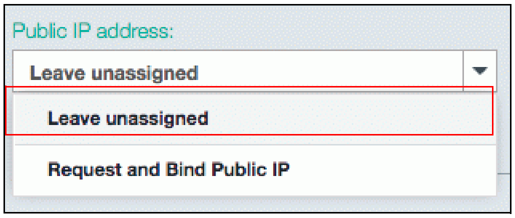
Container name: node01

Size: Micro(256 MB Memory, 16 GB Storage)

Pico(64 MB Memory, 4 GB Storage)
Nano(128 MB Memory, 8 GB Storage)
Micro(256 MB Memory, 16 GB Storage)
Tiny(512 MB Memory, 32 GB Storage)
Small(1 GB Memory, 64 GB Storage)
Medium(2 GB Memory, 128 GB Storage)
Large(4 GB Memory, 256 GB Storage)

Figure 5-7 Single container list of sizes

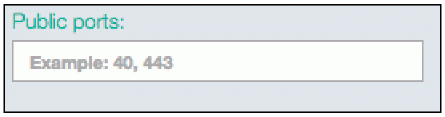
4. The public IP address can be requested and bound to the container to make it accessible from the web as shown in Figure 5-8. Select **Leave unassigned**.



Public IP address: Leave unassigned

Figure 5-8 Single container Public IP

5. When binding the public IP address, at least one port number must be entered in the Public ports field to be exposed externally as shown in Figure 5-9. Leave the Public ports field empty.



Public ports:

Figure 5-9 Single container public ports

Note: If multiple ports must be exposed, be sure to separate each number by a comma.

6. Expand “Advanced Options” to display the following options shown in Figure 5-10.
   - *Volumes* can be mapped to a container to access data on the host from the container or ensure data persistence after the container deletion.

     Tip: To learn more about volumes, refer to Bluemix documentation:

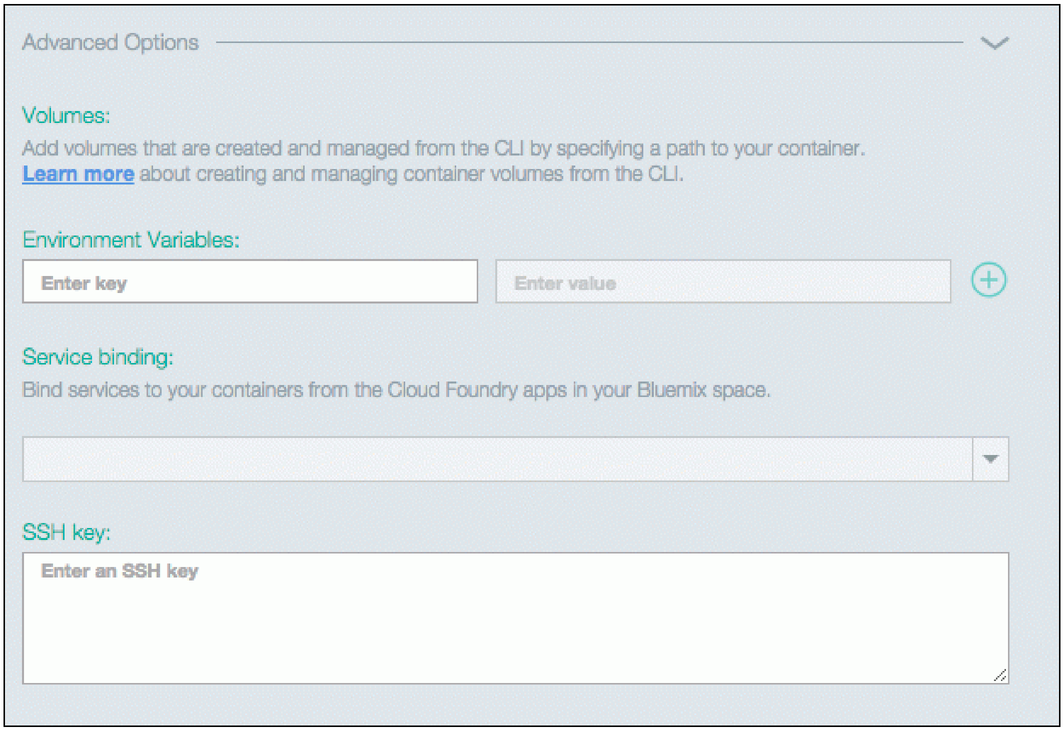
     https://www.ng.bluemix.net/docs/containers/container_optional.html#container_volumes

   - *Environment variables* are used to pass dynamic values to a container upon launch. Those values can then be reused inside the container as standard UNIX environment variables.
   - *Service binding* enables a container to be directly mapped to an existing Cloud Foundry application.

     Tip: To learn more about binding containers to applications, refer to Bluemix documentation:

     https://www.ng.bluemix.net/docs/containers/container_binding_ov.html

   - *SSH key* can be used to inject a public key in a container that is used to securely connect through SSH.



Advanced Options

Volumes:

Add volumes that are created and managed from the CLI by specifying a path to your container. Learn more about creating and managing container volumes from the CLI.

Environment Variables:

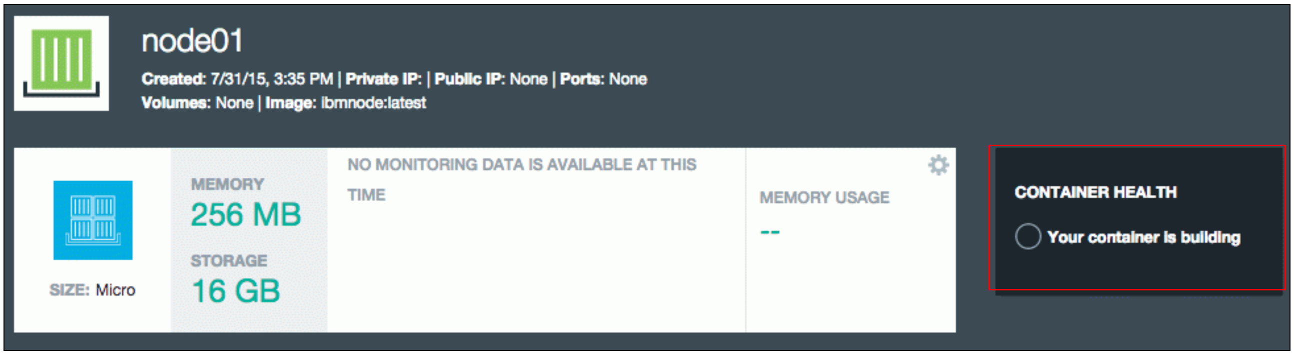
Service binding:

Bind services to your containers from the Cloud Foundry apps in your Bluemix space.

SSH key:

Figure 5-10 Single container advanced options

7. Click **CREATE** to launch the container. The container details and status are displayed as shown in Figure 5-11.



node01

Created: 7/31/15, 3:35 PM | **Private IP:** | **Public IP:** None | **Ports:** None
Volumes: None | **Image:** ibmnode:latest

SIZE: Micro

MEMORY 256 MB

STORAGE 16 GB

NO MONITORING DATA IS AVAILABLE AT THIS TIME

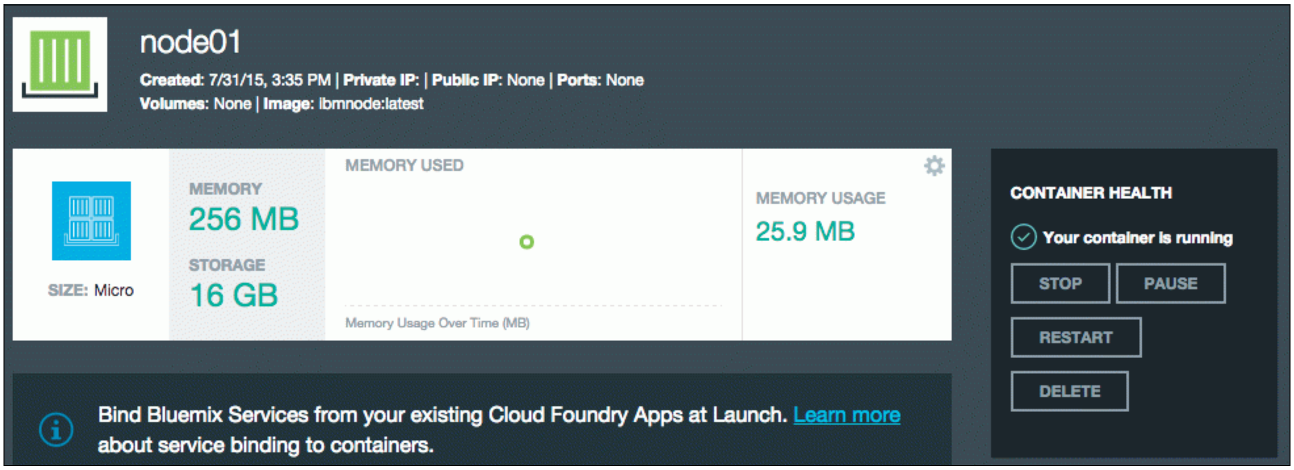
MEMORY USAGE --

CONTAINER HEALTH

Your container is building

Figure 5-11 Single container building view

8. After a few seconds, the container's status changes to `running`. The container's details and logs update accordingly as shown in Figure 5-12.



node01

Created: 7/31/15, 3:35 PM | **Private IP:** | **Public IP:** None | **Ports:** None
Volumes: None | **Image:** ibmnode:latest

SIZE: Micro

MEMORY 256 MB

STORAGE 16 GB

MEMORY USED

Memory Usage Over Time (MB)

MEMORY USAGE 25.9 MB

CONTAINER HEALTH

Your container is running

Bind Bluemix Services from your existing Cloud Foundry Apps at Launch. Learn more about service binding to containers.

Figure 5-12 Single container running view

9. By selecting the appropriate action, the container can be:

   – Stopped
   – Paused
   – Restarted
   – Deleted

10.Click the cog icon in the upper right of the white panel that is displaying the memory used by the container, and select **CPU Used**, as shown in Figure 5-13 to update the graphic as shown in Figure 5-14.

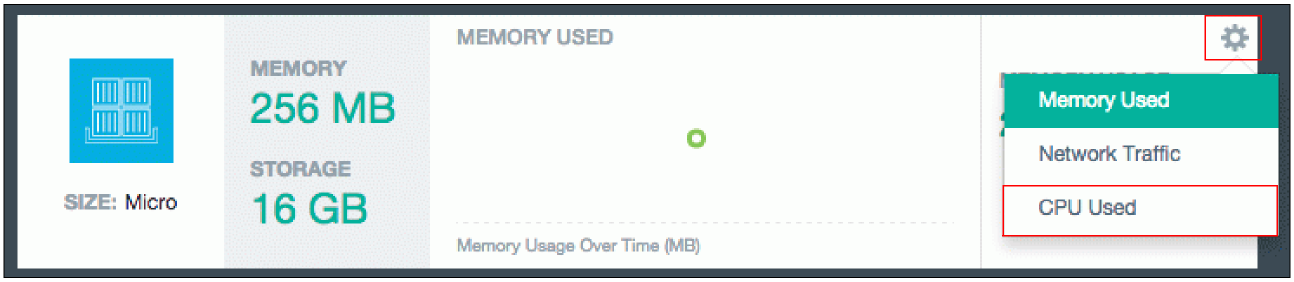


Figure 5-13 Single container simple used memory monitoring

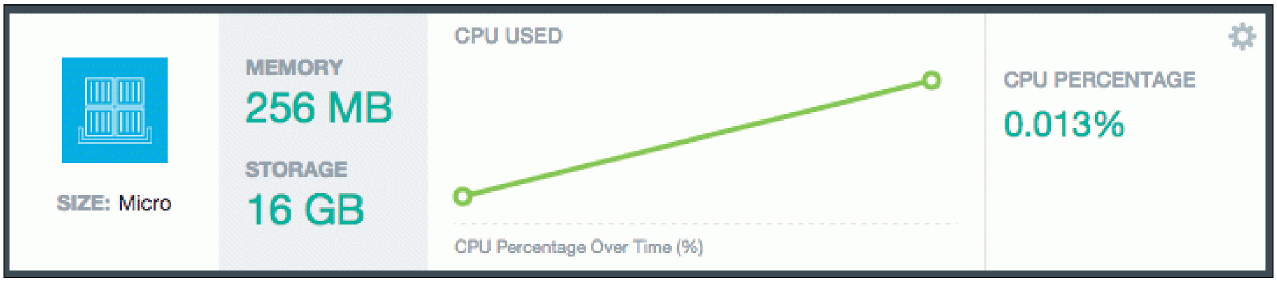


Figure 5-14 Single container simple used CPU monitoring

11.Click **Monitoring and Logs** on the left side of the window as shown in Figure 5-15. It displays a more detailed and historical view of monitoring data as shown in Figure 5-16 on page 102.

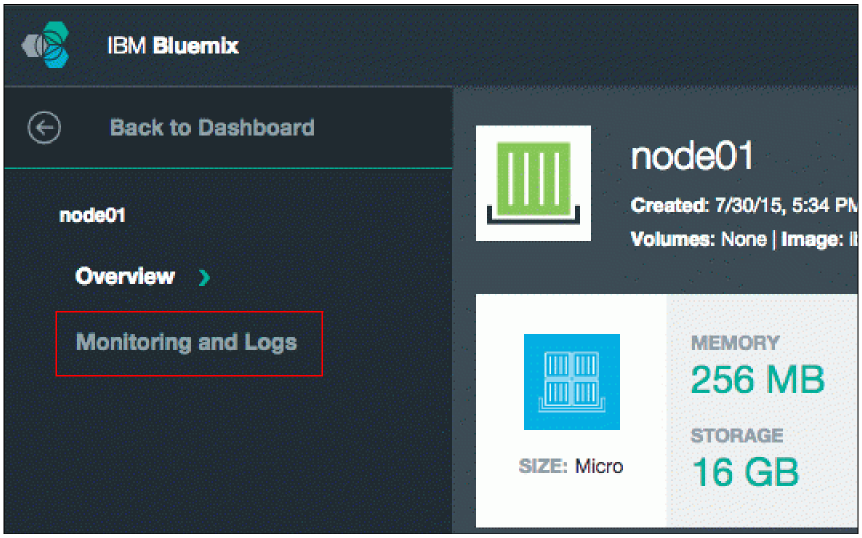


Figure 5-15 Monitoring and Logs

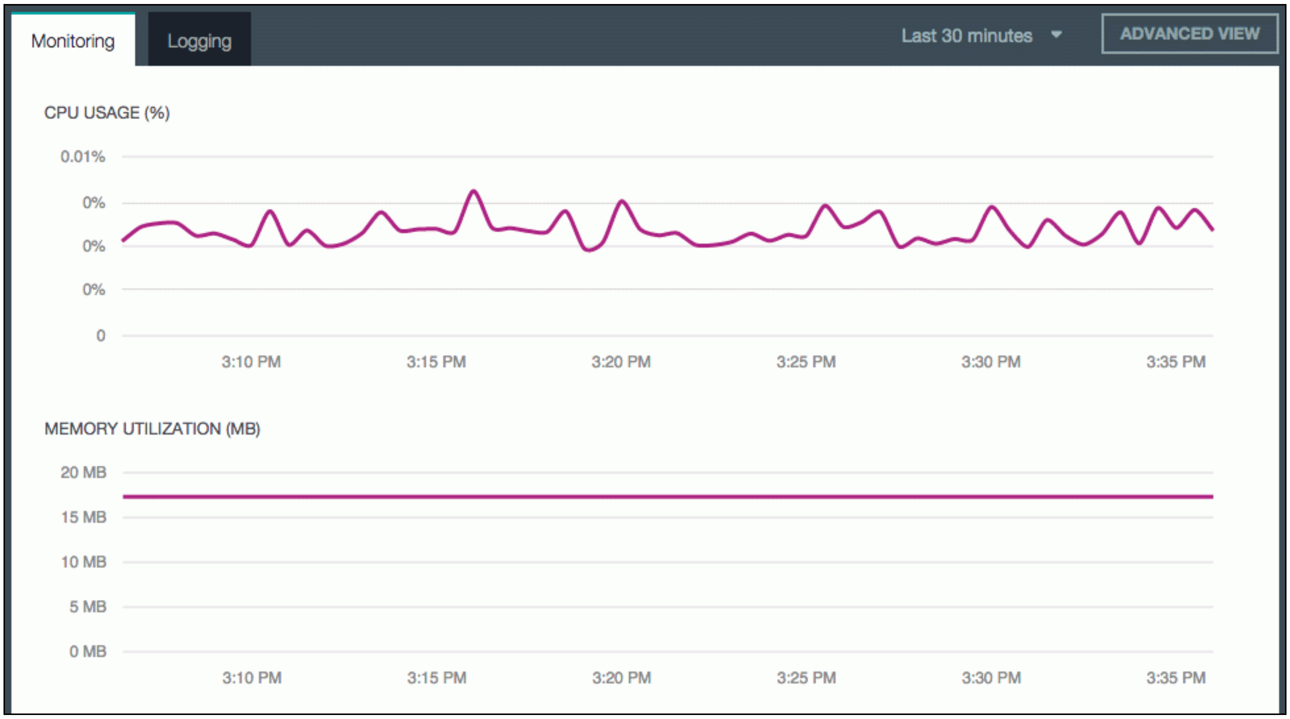


Figure 5-16 Single container monitoring charts

12. Click **ADVANCED VIEW** in the upper right corner of the window to display the Operations Dashboard as shown in Figure 5-17. This view provides advanced monitoring data and can be customized.

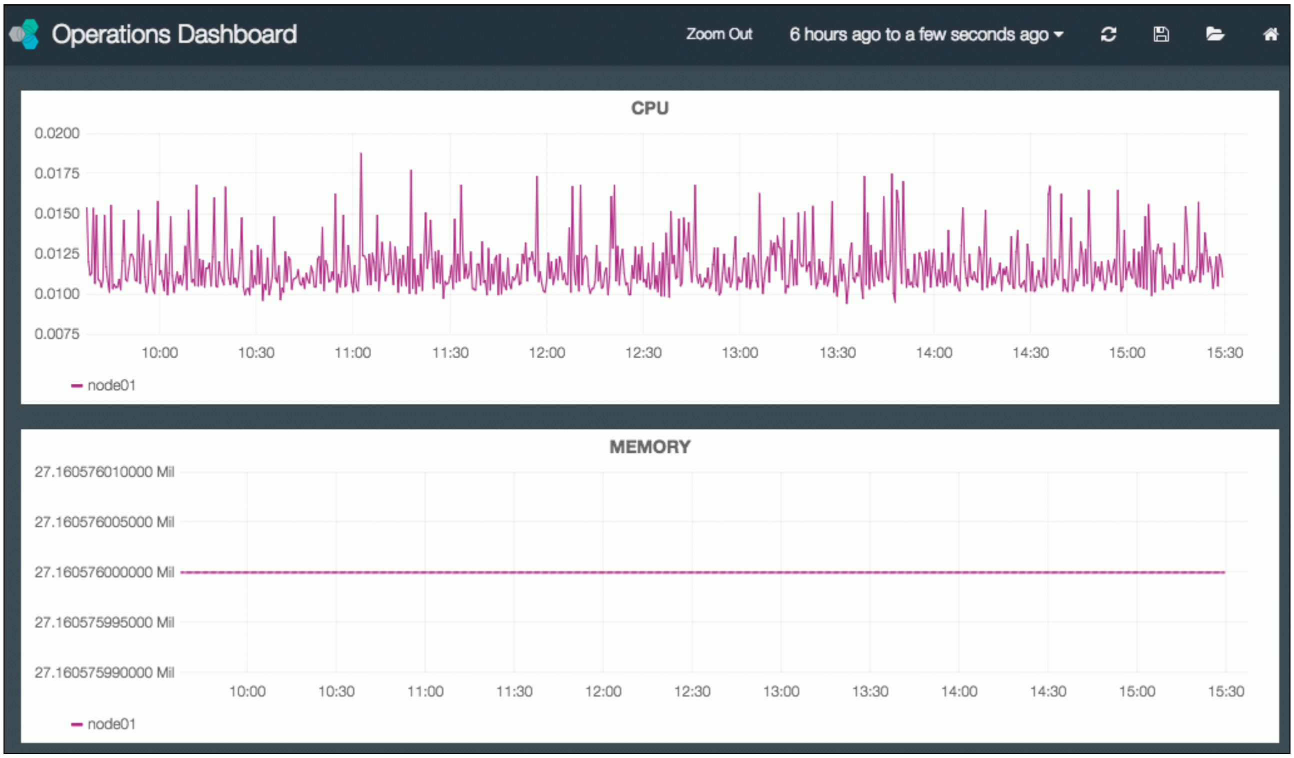
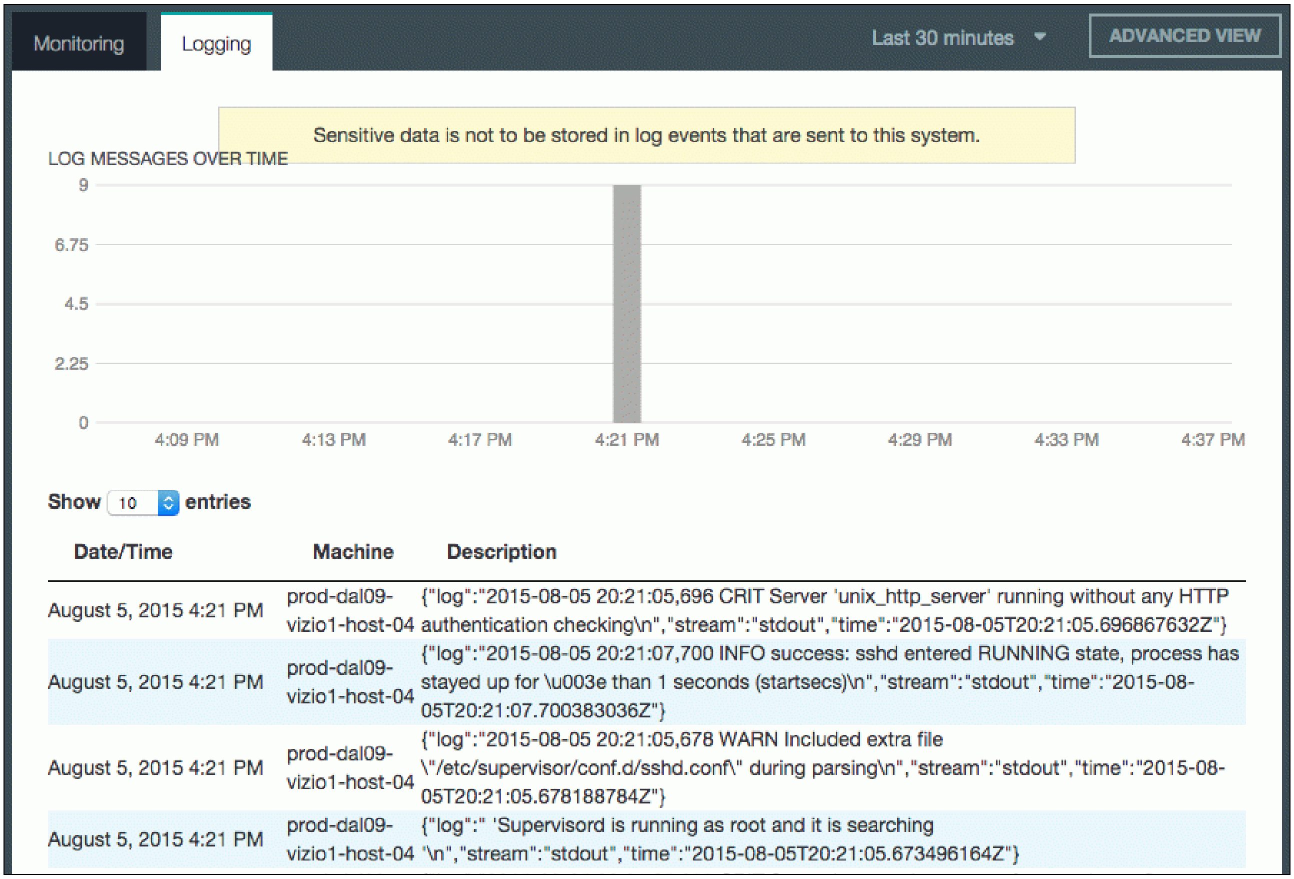


Figure 5-17 Single container operations dashboard

13. Click the **Logging** tab on top of the frame to display logs collected from the container as shown in Figure 5-18.

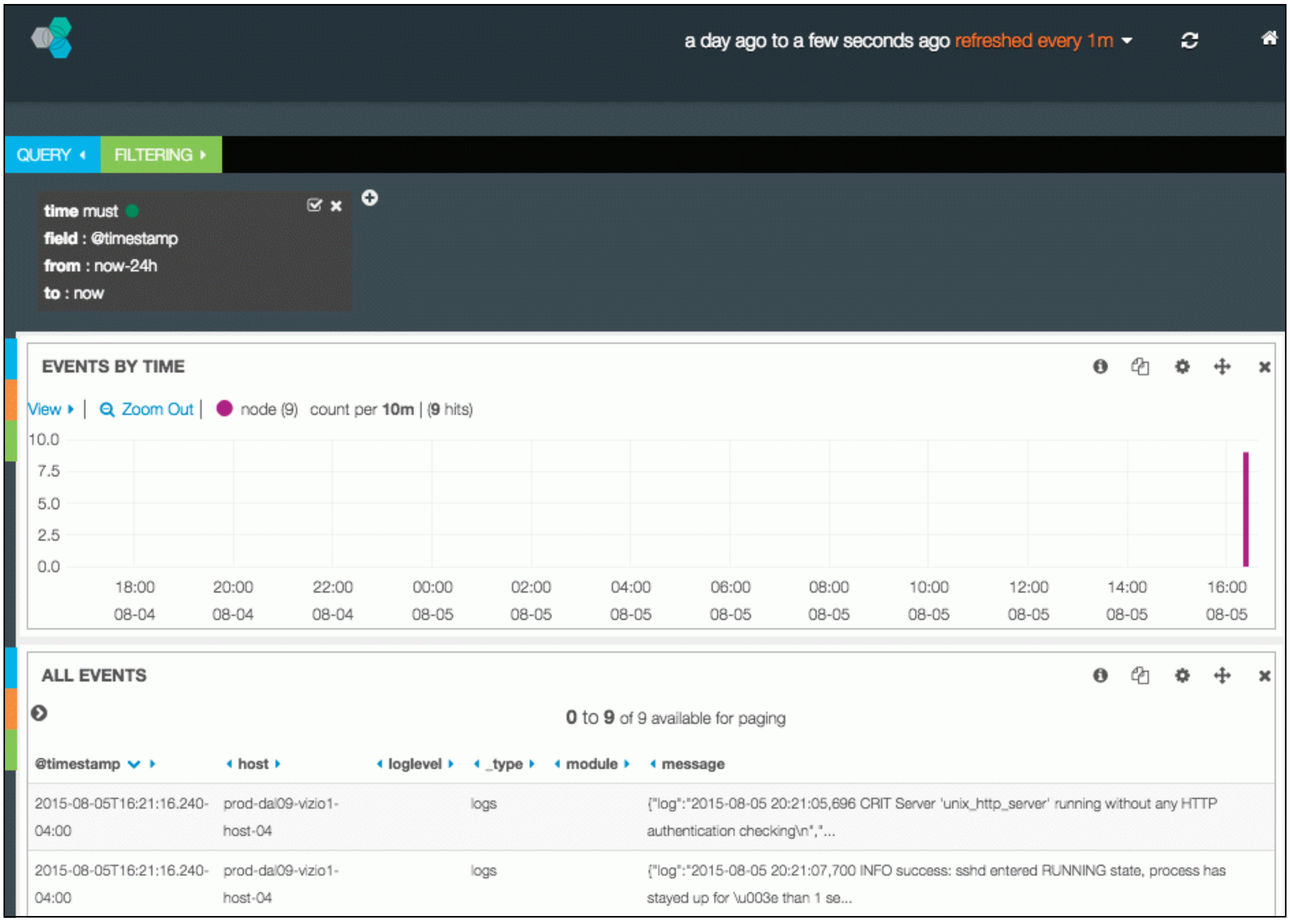


Sensitive data is not to be stored in log events that are sent to this system.

Date/Time	Machine	Description
August 5, 2015 4:21 PM	prod-dal09-vizio1-host-04	{"log":"2015-08-05 20:21:05,696 CRIT Server 'unix_http_server' running without any HTTP authentication checking\n","stream":"stdout","time":"2015-08-05T20:21:05.696867632Z"}
August 5, 2015 4:21 PM	prod-dal09-vizio1-host-04	{"log":"2015-08-05 20:21:07,700 INFO success: sshd entered RUNNING state, process has stayed up for \u003e than 1 seconds (startsecs)\n","stream":"stdout","time":"2015-08-05T20:21:07.700383036Z"}
August 5, 2015 4:21 PM	prod-dal09-vizio1-host-04	{"log":"2015-08-05 20:21:05,678 WARN Included extra file \"/etc/supervisor/conf.d/sshd.conf\" during parsing\n","stream":"stdout","time":"2015-08-05T20:21:05.678188784Z"}
August 5, 2015 4:21 PM	prod-dal09-vizio1-host-04	{"log":" 'Supervisord is running as root and it is searching '\n","stream":"stdout","time":"2015-08-05T20:21:05.673496164Z"}

Figure 5-18 Single container logging view

14. Click **ADVANCED VIEW** in the upper right corner of the window to display a more advanced set of tools as shown in Figure 5-19. This dashboard can be used to query and filter the logs.



time must
field : @timestamp
from : now-24h
to : now

0 to **9** of 9 available for paging

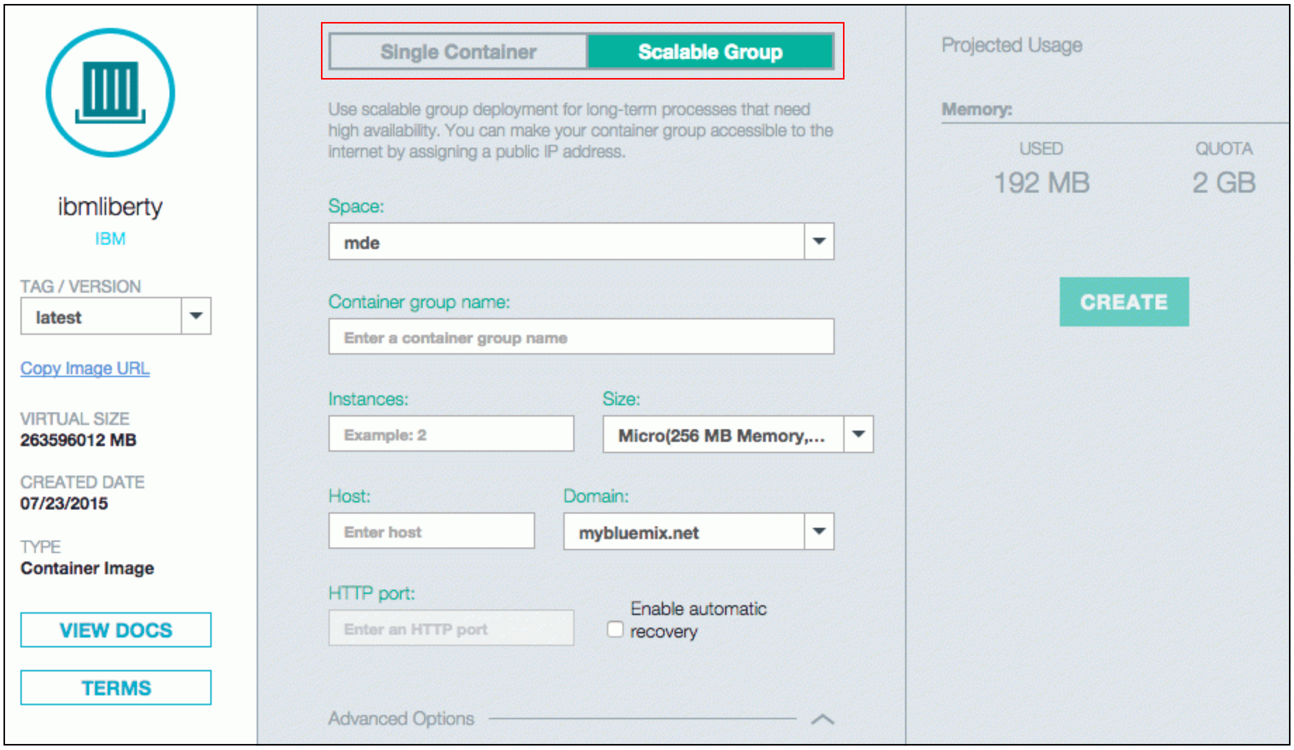
@timestamp	host	loglevel	_type	module	message
2015-08-05T16:21:16.240-04:00	prod-dal09-vizio1-host-04		logs		{"log":"2015-08-05 20:21:05,696 CRIT Server 'unix_http_server' running without any HTTP authentication checking\n","...
2015-08-05T16:21:16.240-04:00	prod-dal09-vizio1-host-04		logs		{"log":"2015-08-05 20:21:07,700 INFO success: sshd entered RUNNING state, process has stayed up for \u003e than 1 se...

Figure 5-19 Single container advanced logging view

5.1.4 Running a group of containers

To create a new group of containers, log on to Bluemix and complete the following steps:

1. Click **Dashboard** → **Start Containers** → **ibmliberty**. Select **Scalable Group** at the top of the panel to display the available options as shown in Figure 5-20 on page 105.



latest

VIRTUAL SIZE
263596012 MB

CREATED DATE
07/23/2015

TYPE
Container Image

Use scalable group deployment for long-term processes that need high availability. You can make your container group accessible to the internet by assigning a public IP address.

Space: mde

Container group name: Enter a container group name

Instances: Example: 2

Size: Micro(256 MB Memory,...

Host: Enter host

Domain: mybluemix.net

HTTP port: Enter an HTTP port

□ Enable automatic recovery

Memory:

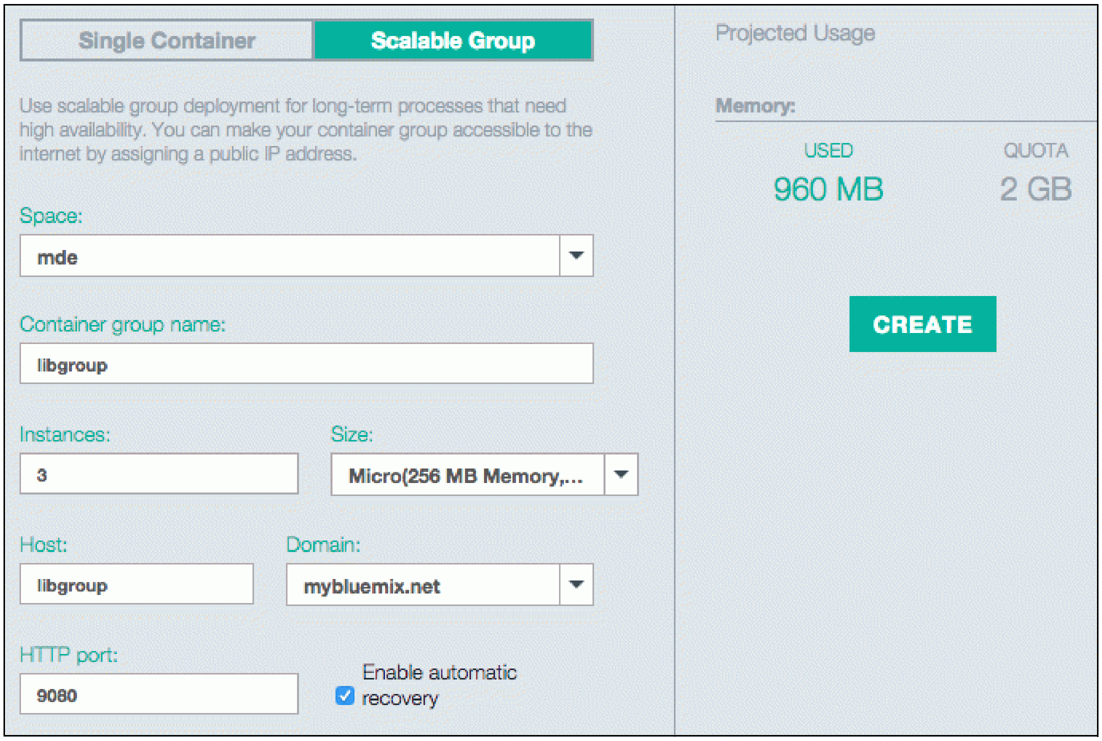
USED	QUOTA
192 MB	2 GB

Figure 5-20 Scalable group creation form

2. Complete the creation form with the following information as shown in Figure 5-21:
   a. Leave the default **Space** selected
   b. Enter `libgroup` in the **Container group name** field
   c. Enter `3` in the **Instances** field
   d. Leave the default `Micro` size in the **Size** list
   e. Enter *libgroup* in the **Host** field
   f. Leave the default **Domain** selected
   g. Select **Enable automatic recovery**
   h. Enter `9080` in the **HTTP port** field to be exposed

   > **Tip:** The automatic recovery feature automatically starts a new container if one of the scalable group fails. For more information about scalable groups, see the Bluemix documentation:
   >
   > `https://www.ng.bluemix.net/docs/containers/container_group_ov.html`

> **Note:** For more details about Advanced Options, refer to step 6 on page 99.



Use scalable group deployment for long-term processes that need high availability. You can make your container group accessible to the internet by assigning a public IP address.

Space: mde

Container group name: libgroup

Instances: 3

Size: Micro(256 MB Memory,...

Host: libgroup

Domain: mybluemix.net

HTTP port: 9080

☑ Enable automatic recovery

Projected Usage

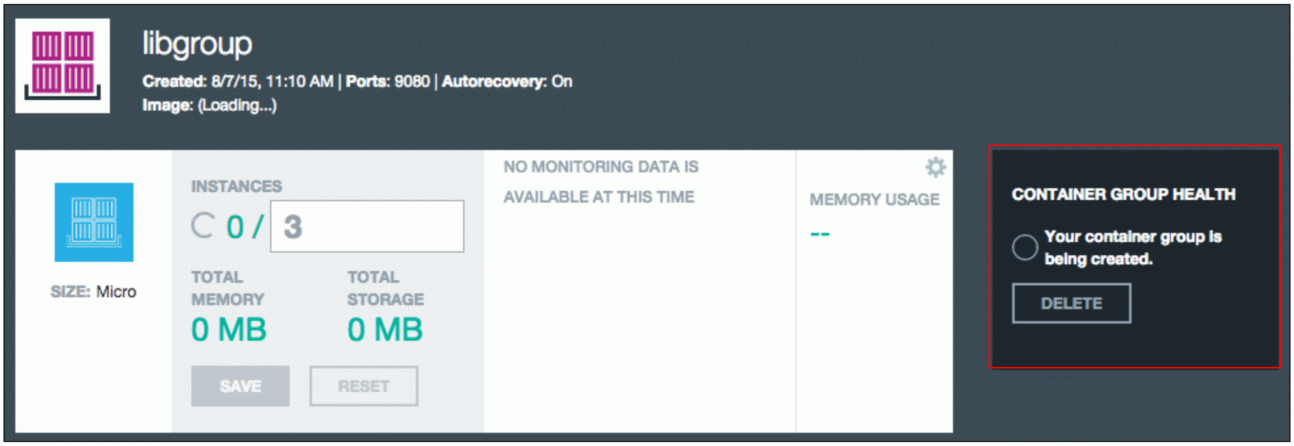
Memory:

USED 960 MB

QUOTA 2 GB

Figure 5-21 Scalable group completed creation form

3. Click **CREATE** to launch the group of containers. The scalable group details and status are displayed as shown in Figure 5-22 on page 107.



libgroup

Created: 8/7/15, 11:10 AM | Ports: 9080 | Autorecovery: On

Image: (Loading...)

Figure 5-22 Scalable group building view

4. After a minute, the status of the scalable group changes to `running`. The details and logs update accordingly as shown in Figure 5-23.

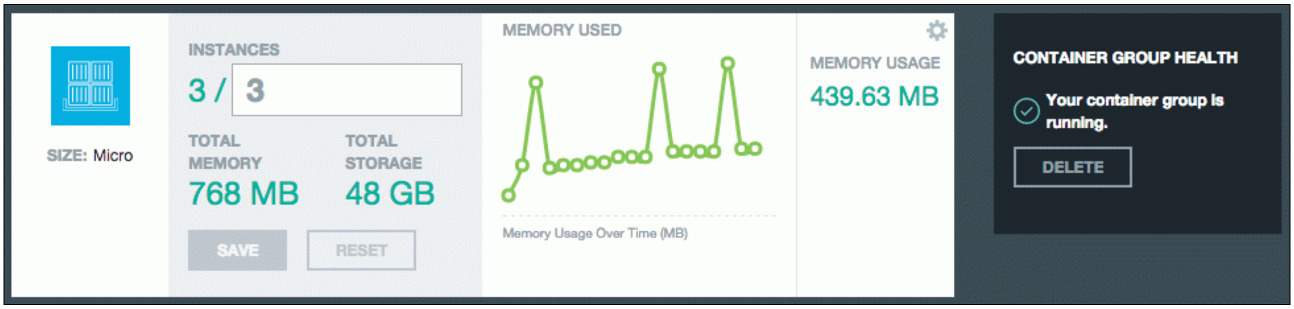


Figure 5-23 Scalable group running view

5. On the left of the window, click **Instances** as shown in Figure 5-24 to display the list of containers in `libgroup` scalable group (Figure 5-25 on page 108).

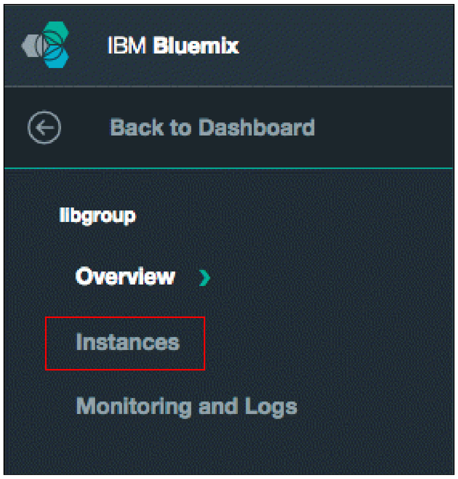
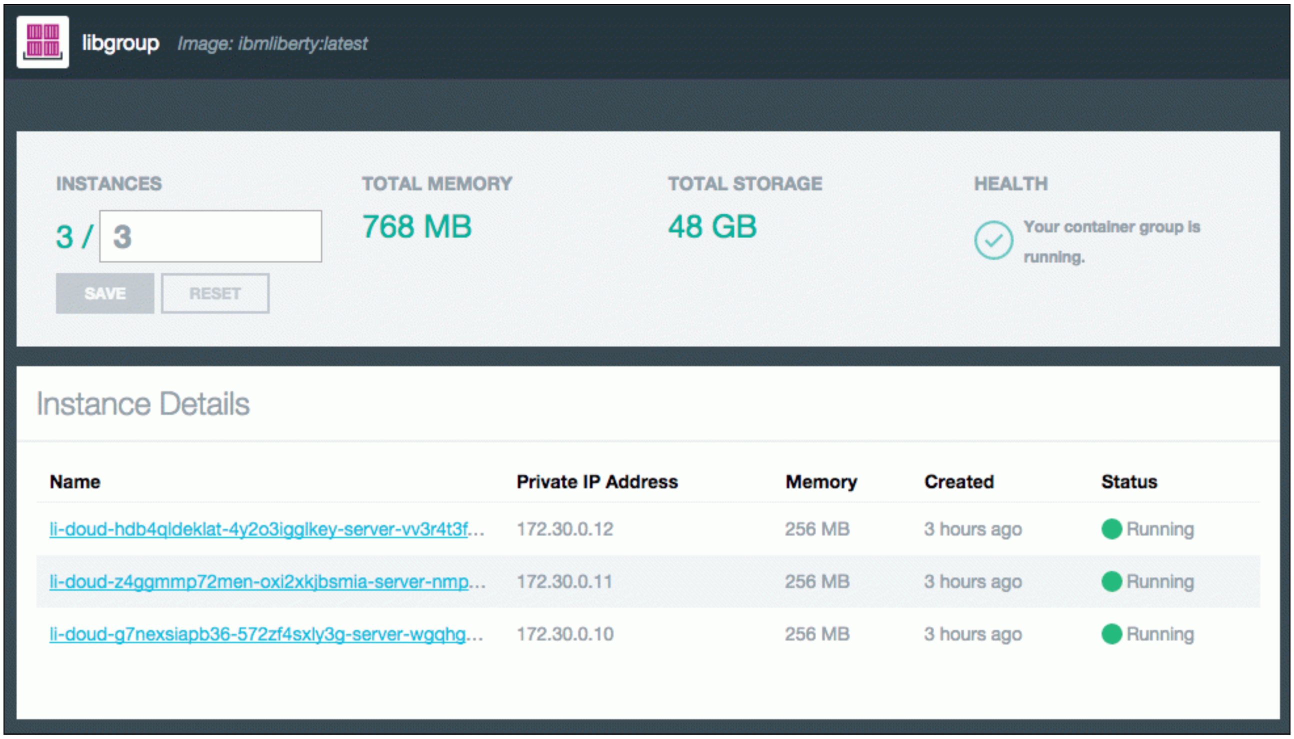


Figure 5-24 Scalable group menu



libgroup Image: ibmliberty:latest

INSTANCES 3 / 3

TOTAL MEMORY 768 MB

TOTAL STORAGE 48 GB

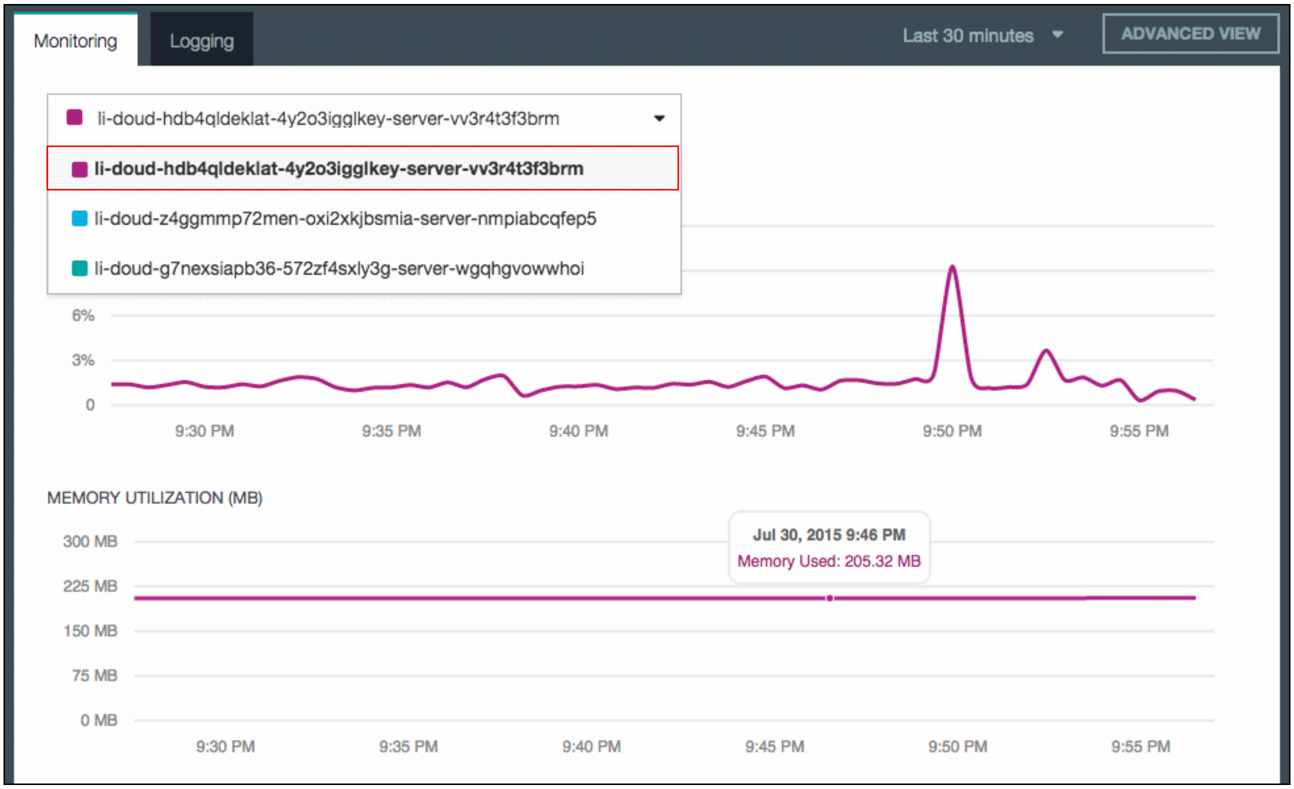
HEALTH Your container group is running.

Instance Details

Name	Private IP Address	Memory	Created	Status
li-doud-hdb4qldeklat-4y2o3igglkey-server-vv3r4t3f...	172.30.0.12	256 MB	3 hours ago	● Running
li-doud-z4ggmmp72men-oxi2xkjbsmia-server-nmp...	172.30.0.11	256 MB	3 hours ago	● Running
li-doud-g7nexsiapb36-572zf4sxly3g-server-wgqhg...	172.30.0.10	256 MB	3 hours ago	● Running

Figure 5-25 Scalable group instances list

6. On the left of the window, click **Monitoring and Logs** and select from the top list the instance to display as shown in Figure 5-26.



li-doud-hdb4qldeklat-4y2o3igglkey-server-vv3r4t3f3brm

li-doud-hdb4qldeklat-4y2o3igglkey-server-vv3r4t3f3brm

li-doud-z4ggmmp72men-oxi2xkjbsmia-server-nmpiabcqfep5

li-doud-g7nexsiapb36-572zf4sxly3g-server-wgqhgvowwhoi

Figure 5-26 Scalable group monitoring per instance

Note: For more details about Advanced View, see step 12 on page 102.

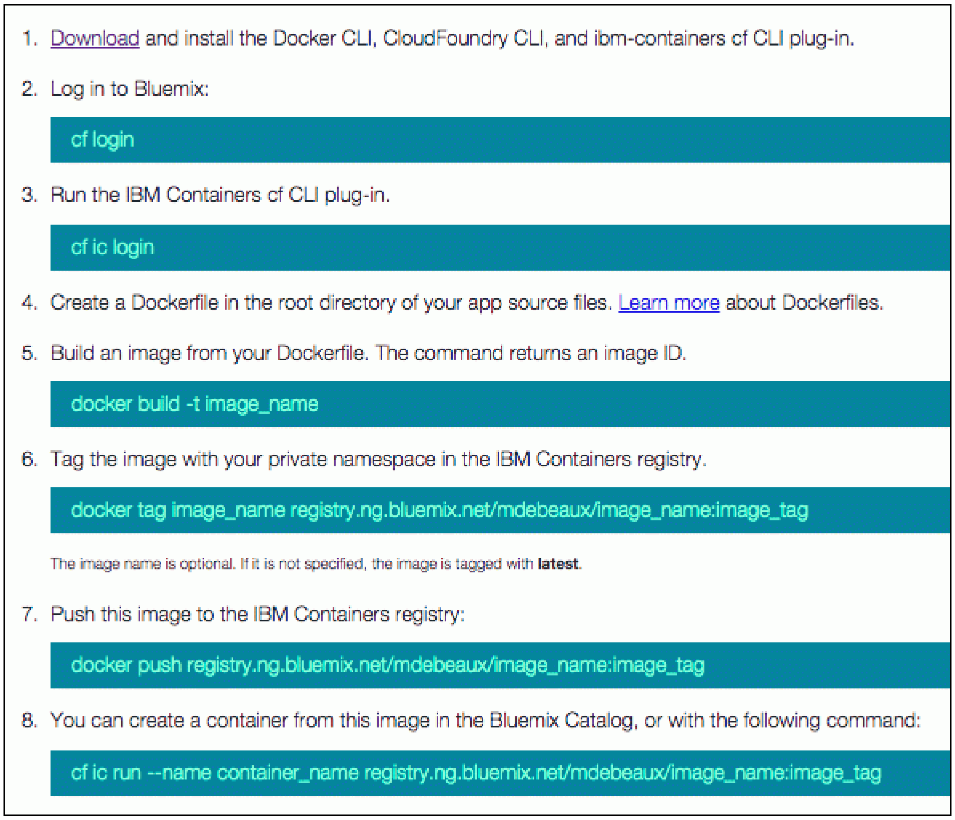
5.2 Running your own IBM Containers

This section presents the steps that are needed to be able to run your own images as containers on Bluemix.

5.2.1 Configuring your client to access IBM Containers

To configure your client and get access to IBM Containers, log on to Bluemix and complete the following steps:

1. Click **Dashboard** → **Start Containers** → **Add your own**. The list of steps are displayed as shown in Figure 5-27.



1. Download and install the Docker CLI, CloudFoundry CLI, and ibm-containers cf CLI plug-in.
2. Log in to Bluemix:

```
cf login
```

3. Run the IBM Containers cf CLI plug-in.

```
cf ic login
```

4. Create a Dockerfile in the root directory of your app source files. Learn more about Dockerfiles.
5. Build an image from your Dockerfile. The command returns an image ID.

```
docker build -t image_name
```

6. Tag the image with your private namespace in the IBM Containers registry.

```
docker tag image_name registry.ng.bluemix.net/mdebeaux/image_name:image_tag
```

The image name is optional. If it is not specified, the image is tagged with **latest**.

7. Push this image to the IBM Containers registry:

```
docker push registry.ng.bluemix.net/mdebeaux/image_name:image_tag
```

8. You can create a container from this image in the Bluemix Catalog, or with the following command:

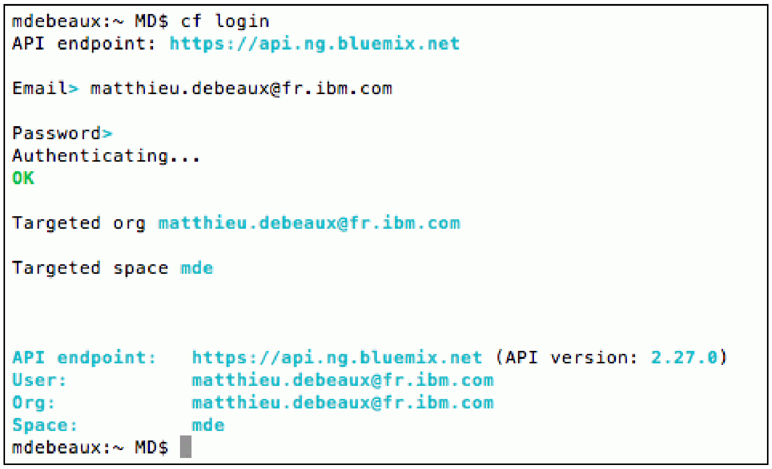
```
cf ic run --name container_name registry.ng.bluemix.net/mdebeaux/image_name:image_tag
```

Figure 5-27 List of actions to run private images in IBM Containers

2. Click **Download** and follow IBM Bluemix documentation to install:
   a. Docker CLI
   b. Cloud Foundry CLI
   c. IBM Containers Cloud Foundry plug-in

> **Note:** This installation process depends on your operating system (OS). Ensure that you follow the instructions matching your OS.

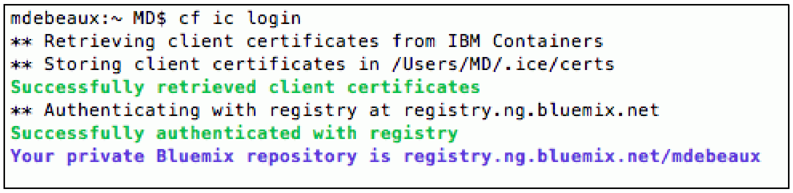
3. Open a terminal window, or command line (on Windows), and type **`cf login`** to login Bluemix as shown in Figure 5-28. When prompted, enter your **Email** and **Password**.



```
mdebeaux:~ MD$ cf login
API endpoint: https://api.ng.bluemix.net

Email> matthieu.debeaux@fr.ibm.com

Password>
Authenticating...
OK

Targeted org matthieu.debeaux@fr.ibm.com

Targeted space mde



API endpoint:   https://api.ng.bluemix.net (API version: 2.27.0)
User:           matthieu.debeaux@fr.ibm.com
Org:            matthieu.debeaux@fr.ibm.com
Space:          mde
mdebeaux:~ MD$
```

Figure 5-28   Log in to IBM Bluemix

4. Type **`cf ic login`** to log in to IBM Containers as shown in Figure 5-29.



```
mdebeaux:~ MD$ cf ic login
** Retrieving client certificates from IBM Containers
** Storing client certificates in /Users/MD/.ice/certs
Successfully retrieved client certificates
** Authenticating with registry at registry.ng.bluemix.net
Successfully authenticated with registry
Your private Bluemix repository is registry.ng.bluemix.net/mdebeaux
```

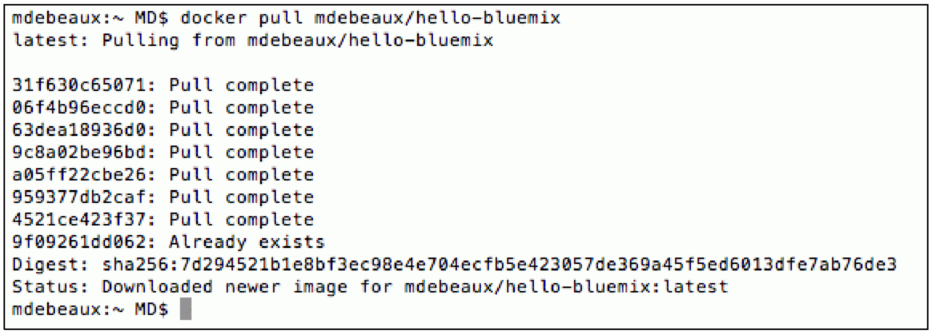
Figure 5-29   Log in to IBM Containers

5.2.2 Making your images available in IBM Containers

To make your private images available in IBM Containers, open a terminal or command window and complete the following steps:

> **Note:** In this section, we *pull* a simple image from Docker Hub. However, it is also possible to *build* an image from a Dockerfile. For more information about building Docker images, see the Docker documentation:
>
> https://docs.docker.com/reference/commandline/build

1. Type **`docker pull mdebeaux/hello-bluemix`** to retrieve the Docker image from Docker Hub, as shown in Figure 5-30 on page 111.

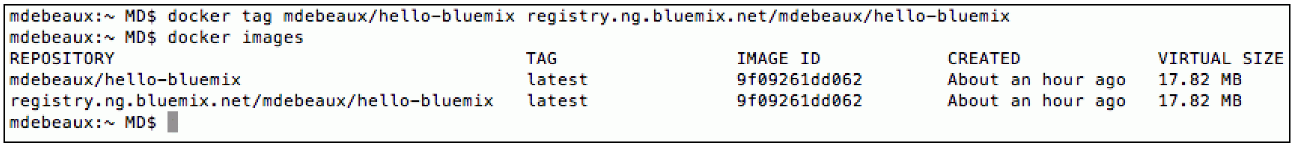


```
mdebeaux:~ MD$ docker pull mdebeaux/hello-bluemix
latest: Pulling from mdebeaux/hello-bluemix

31f630c65071: Pull complete
06f4b96eccd0: Pull complete
63dea18936d0: Pull complete
9c8a02be96bd: Pull complete
a05ff22cbe26: Pull complete
959377db2caf: Pull complete
4521ce423f37: Pull complete
9f09261dd062: Already exists
Digest: sha256:7d294521b1e8bf3ec98e4e704ecfb5e423057de369a45f5ed6013dfe7ab76de3
Status: Downloaded newer image for mdebeaux/hello-bluemix:latest
mdebeaux:~ MD$
```

Figure 5-30 Docker pull an image from Docker Hub

2. Type **`docker tag mdebeaux/hello-bluemix registry.ng.bluemix.net/`***your_registry_name***`/hello-bluemix`** to mark this image as pertaining to your Bluemix Registry as shown in Figure 5-31.

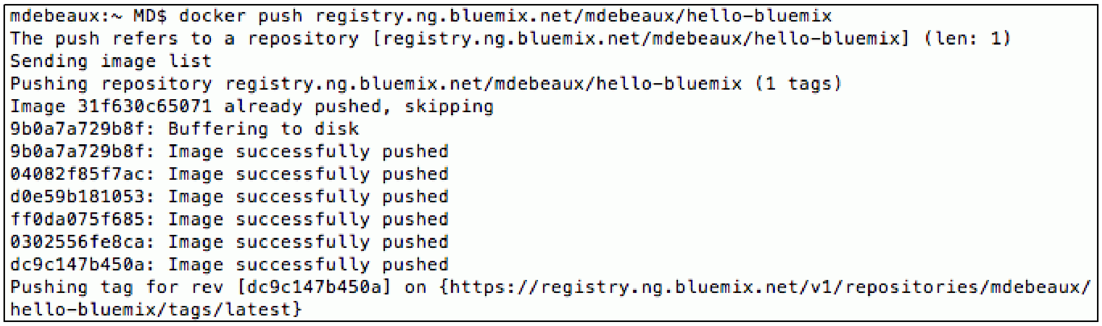
3. Type **`docker images`** to check that the image is correctly tagged as shown in Figure 5-31.



```
mdebeaux:~ MD$ docker tag mdebeaux/hello-bluemix registry.ng.bluemix.net/mdebeaux/hello-bluemix
mdebeaux:~ MD$ docker images
REPOSITORY                                       TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
mdebeaux/hello-bluemix                           latest              9f09261dd062        About an hour ago   17.82 MB
registry.ng.bluemix.net/mdebeaux/hello-bluemix   latest              9f09261dd062        About an hour ago   17.82 MB
mdebeaux:~ MD$
```

Figure 5-31 Docker tag an image for Bluemix Registry

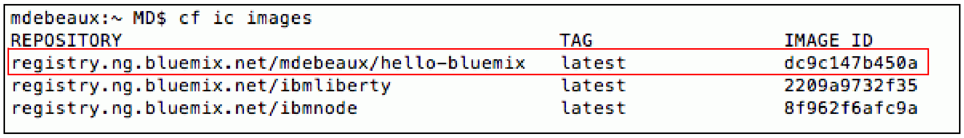
4. Type **`docker push registry.ng.bluemix.net/`***your_registry_name***`/hello-bluemix`** to upload the image to your registry as shown in Figure 5-32.



```
mdebeaux:~ MD$ docker push registry.ng.bluemix.net/mdebeaux/hello-bluemix
The push refers to a repository [registry.ng.bluemix.net/mdebeaux/hello-bluemix] (len: 1)
Sending image list
Pushing repository registry.ng.bluemix.net/mdebeaux/hello-bluemix (1 tags)
Image 31f630c65071 already pushed, skipping
9b0a7a729b8f: Buffering to disk
9b0a7a729b8f: Image successfully pushed
04082f85f7ac: Image successfully pushed
d0e59b181053: Image successfully pushed
ff0da075f685: Image successfully pushed
0302556fe8ca: Image successfully pushed
dc9c147b450a: Image successfully pushed
Pushing tag for rev [dc9c147b450a] on {https://registry.ng.bluemix.net/v1/repositories/mdebeaux/
hello-bluemix/tags/latest}
```

Figure 5-32 Docker push image to Bluemix Registry

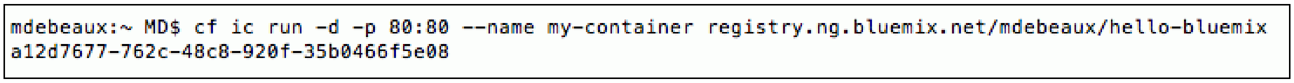
5. Type **`cf ic images`** to check that the image is now available in Bluemix Registry as shown in Figure 5-33.



```
mdebeaux:~ MD$ cf ic images
REPOSITORY                                       TAG                 IMAGE ID
registry.ng.bluemix.net/mdebeaux/hello-bluemix   latest              dc9c147b450a
registry.ng.bluemix.net/ibmliberty               latest              2209a9732f35
registry.ng.bluemix.net/ibmnode                  latest              8f962f6afc9a
```

Figure 5-33 IBM Containers image list

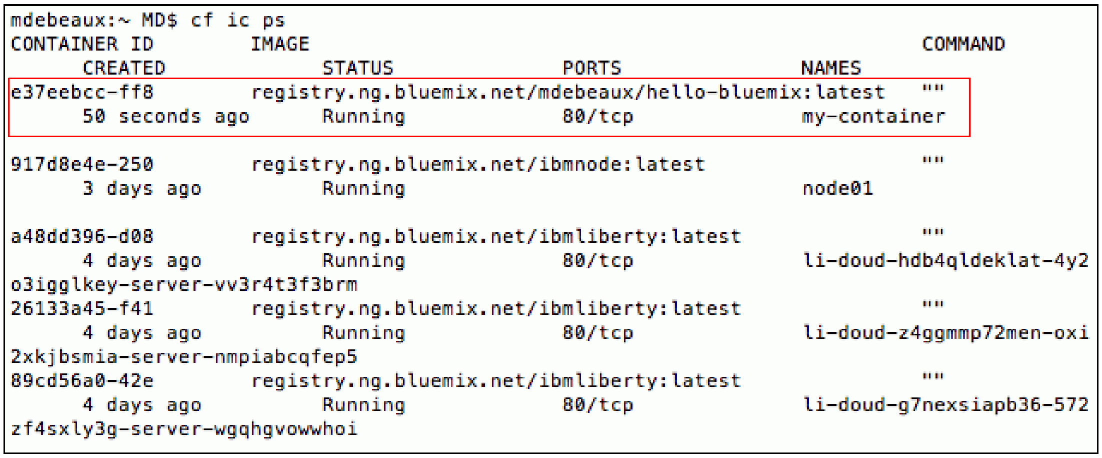
6. Type **`cf ic run -d -p 80:80 --name my-container registry.ng.bluemix.net/`***your_registry_name***`/hello-bluemix`** to launch the creation of a single container as a daemon and with port 80 exposed, as shown in Figure 5-34.



```
mdebeaux:~ MD$ cf ic run -d -p 80:80 --name my-container registry.ng.bluemix.net/mdebeaux/hello-bluemix
a12d7677-762c-48c8-920f-35b0466f5e08
```

Figure 5-34 IBM Containers run command

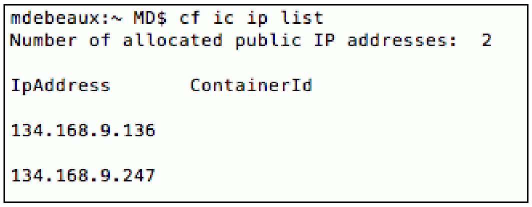
7. Type **`cf ic ps`** to list the running IBM containers as shown in Figure 5-35.



```
mdebeaux:~ MD$ cf ic ps
CONTAINER ID        IMAGE                                                   COMMAND
      CREATED             STATUS              PORTS               NAMES
e37eebcc-ff8        registry.ng.bluemix.net/mdebeaux/hello-bluemix:latest   ""
      50 seconds ago      Running             80/tcp              my-container

917d8e4e-250        registry.ng.bluemix.net/ibmnode:latest                  ""
      3 days ago          Running                                 node01

a48dd396-d08        registry.ng.bluemix.net/ibmliberty:latest               ""
      4 days ago          Running             80/tcp              li-doud-hdb4qldeklat-4y2
o3igglkey-server-vv3r4t3f3brm
26133a45-f41        registry.ng.bluemix.net/ibmliberty:latest               ""
      4 days ago          Running             80/tcp              li-doud-z4ggmmp72men-oxi
2xkjbsmia-server-nmpiabcqfep5
89cd56a0-42e        registry.ng.bluemix.net/ibmliberty:latest               ""
      4 days ago          Running             80/tcp              li-doud-g7nexsiapb36-572
zf4sxly3g-server-wgqhgvowwhoi
```

Figure 5-35 IBM Containers list running containers

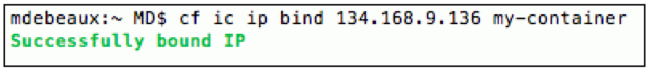
8. The container is now running and exposing port 80. List the available IPs to bind by typing **`cf ic ip list`** as shown in Figure 5-36.



```
mdebeaux:~ MD$ cf ic ip list
Number of allocated public IP addresses:  2

IpAddress       ContainerId

134.168.9.136

134.168.9.247
```

Figure 5-36 IBM Containers list IPs

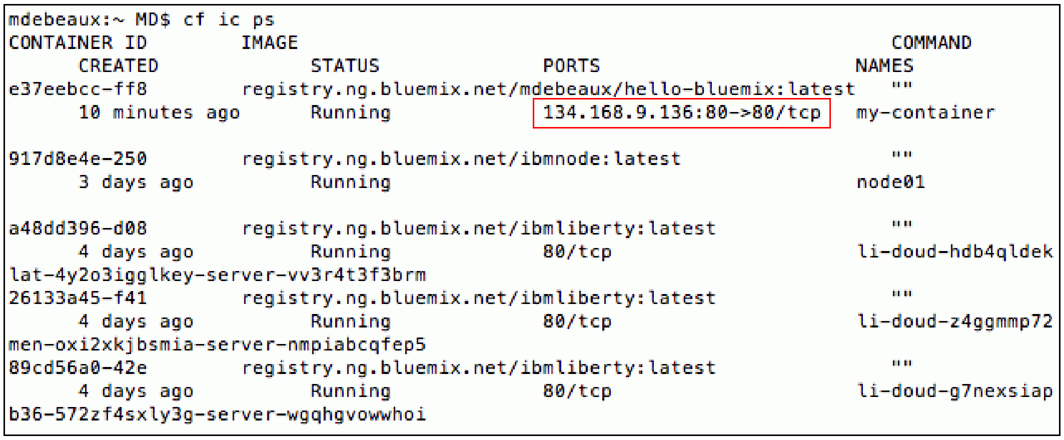
9. Select one of the IPs listed and type **`cf ic ip bind`** *your_ip* **`my-container`** to bind the select IP to the container as shown in Figure 5-37.



```
mdebeaux:~ MD$ cf ic ip bind 134.168.9.136 my-container
Successfully bound IP
```

Figure 5-37 IBM Containers bind IP to container

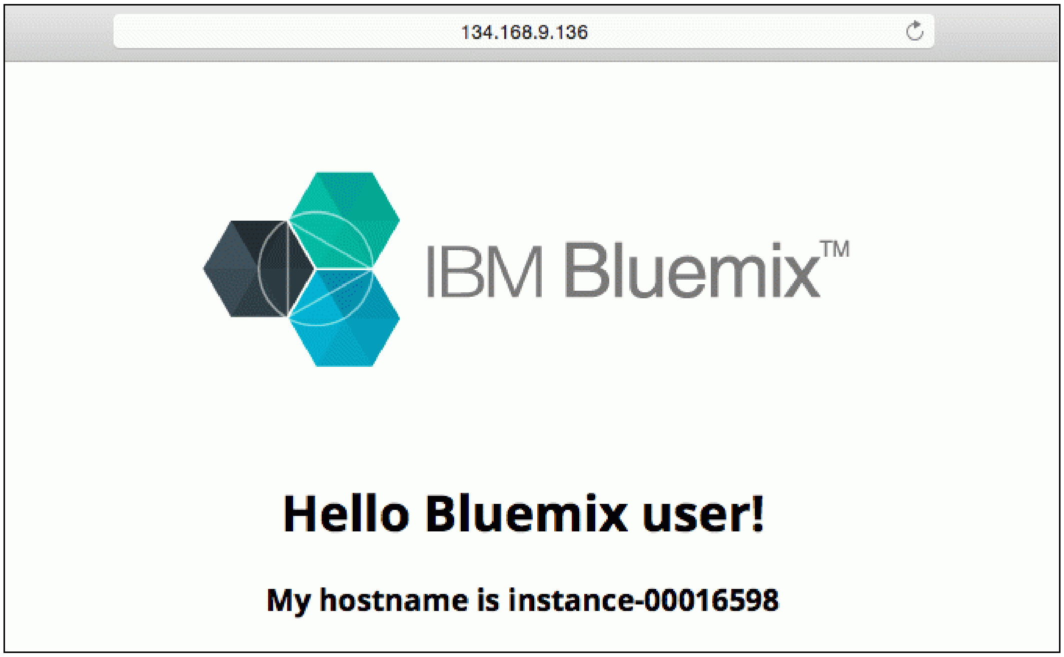
10. Type **`cf ic ps`** to check that the IP was correctly bound to the container as shown in Figure 5-38.



```
mdebeaux:~ MD$ cf ic ps
CONTAINER ID        IMAGE                                                   COMMAND
      CREATED             STATUS              PORTS                      NAMES
e37eebcc-ff8        registry.ng.bluemix.net/mdebeaux/hello-bluemix:latest   ""
      10 minutes ago      Running             134.168.9.136:80->80/tcp   my-container

917d8e4e-250        registry.ng.bluemix.net/ibmnode:latest                  ""
      3 days ago          Running                                        node01

a48dd396-d08        registry.ng.bluemix.net/ibmliberty:latest               ""
      4 days ago          Running             80/tcp                     li-doud-hdb4qldek
lat-4y2o3igglkey-server-vv3r4t3f3brm
26133a45-f41        registry.ng.bluemix.net/ibmliberty:latest               ""
      4 days ago          Running             80/tcp                     li-doud-z4ggmmp72
men-oxi2xkjbsmia-server-nmpiabcqfep5
89cd56a0-42e        registry.ng.bluemix.net/ibmliberty:latest               ""
      4 days ago          Running             80/tcp                     li-doud-g7nexsiap
b36-572zf4sxly3g-server-wgqhgvowwhoi
```

Figure 5-38 IBM Containers showing container with port exposed on external IP

11. Open a web browser and enter *your_ip* in the URL field to display the hello_bluemix page as shown in Figure 5-39.



Hello Bluemix user!

My hostname is instance-00016598

Figure 5-39 Hello Bluemix web page

Tip: For more information about IBM Container commands, refer to Bluemix documentation:

https://www.ng.bluemix.net/docs/containers/container_cli_reference_ov.html#container_cli_reference_cfic

5.3 Setting up CompanyB hybrid application

This section sets up CompanyB enterprise resource planning (ERP) environment.

5.3.1 Solution overview

CompanyB ERP environment is a two-tier application composed of:

- A web-front based on Odoo and running in a container on IBM Containers
- A database based on PostgreSQL running in a Docker container on POWER8

The two tiers are connected through a secure gateway connecting to an IBM DataPower Gateway as shown in the architecture schema in Figure 5-40.

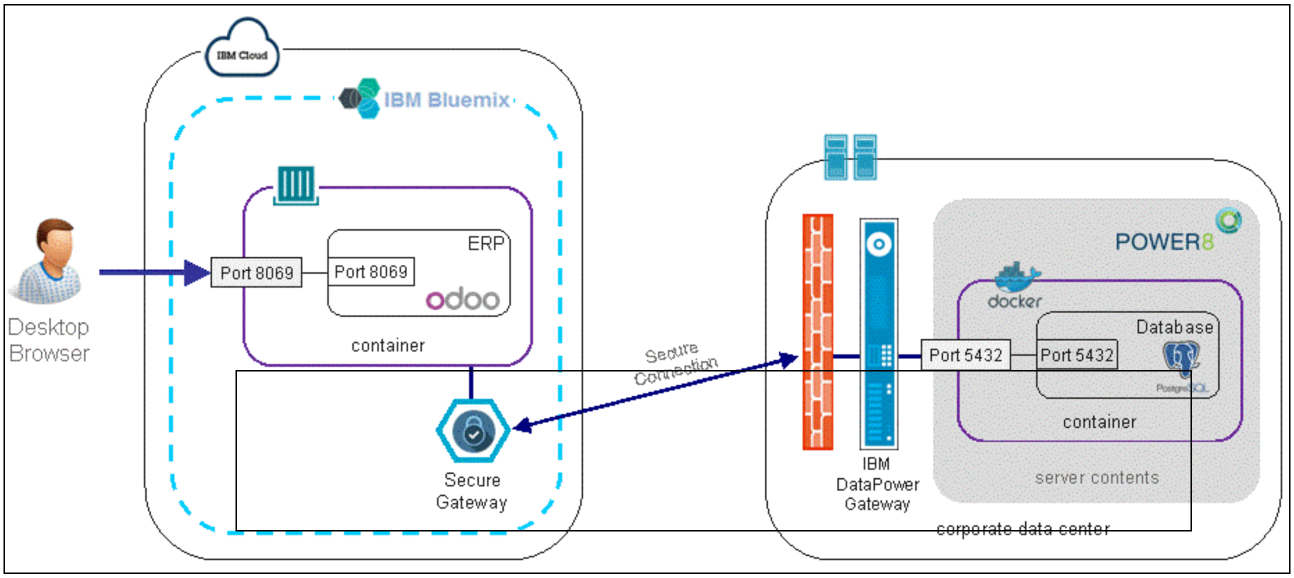


Figure 5-40 CompanyB hybrid application architecture

Business case for this scenario: For more information about the business case for this scenario and the company profile, see 1.3.5, “Solutions and actions by CompanyC CIO” on page 11.

5.3.2 Prerequisites

For this use case, we focus on integrating the existing on-premises Docker environment on Power Systems to IBM Containers on Bluemix. Therefore, the following pre-requirements must be met before going through the integration step-by-step process:

- IBM Containers on Bluemix set up with Bluemix Registry created

Tip: For more information about this step, refer to 5.1.2, “Creating your Bluemix registry” on page 94.

- POWER8 environment set up with Ubuntu 15.04 and Docker 1.5 (or more recent)
- IBM DataPower Gateway 7.2 set up

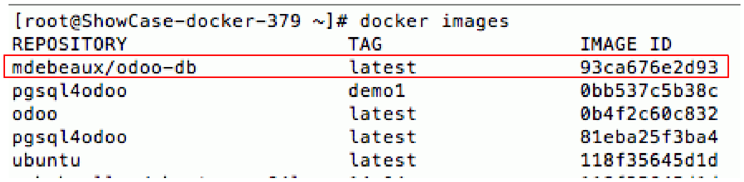
- Docker image based on PostgreSQL already built
- Docker image based on Odoo ready, waiting for connection parameters for build

> **Tip:** As a base for building the Odoo and PostgreSQL images, the following Docker Hub images can be used:
>
> Odoo: `https://registry.hub.docker.com/_/odoo`
>
> PostgreSQL: `https://registry.hub.docker.com/_/postgres`

5.3.3 Running the database container

To set up the PostgreSQL database in Docker on POWER8, log in as root on Ubuntu and complete the following steps:

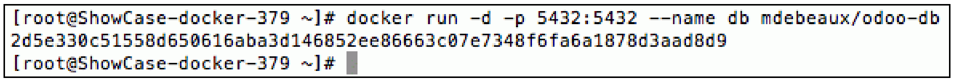
1. Type **`docker images`** to check that the database image is available as shown in Figure 5-41.



```
[root@ShowCase-docker-379 ~]# docker images
REPOSITORY                    TAG                  IMAGE ID
mdebeaux/odoo-db              latest               93ca676e2d93
pgsql4odoo                    demo1                0bb537c5b38c
odoo                          latest               0b4f2c60c832
pgsql4odoo                    latest               81eba25f3ba4
ubuntu                        latest               118f35645d1d
```

Figure 5-41 Docker images available on POWER8 system

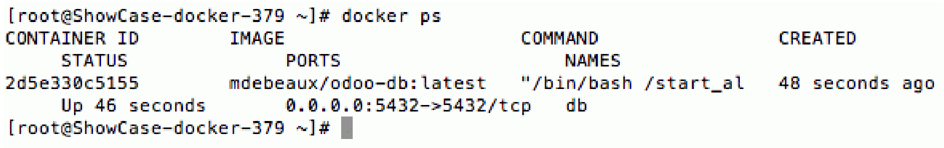
2. Type **`docker run -d -p 5432:5432 --name db`** *your_database_image_name* to launch the container as a daemon and exposing port 5432 as shown in Figure 5-42.



```
[root@ShowCase-docker-379 ~]# docker run -d -p 5432:5432 --name db mdebeaux/odoo-db
2d5e330c51558d650616aba3d146852ee86663c07e7348f6fa6a1878d3aad8d9
[root@ShowCase-docker-379 ~]#
```

Figure 5-42 Running PostgreSQL container on POWER8

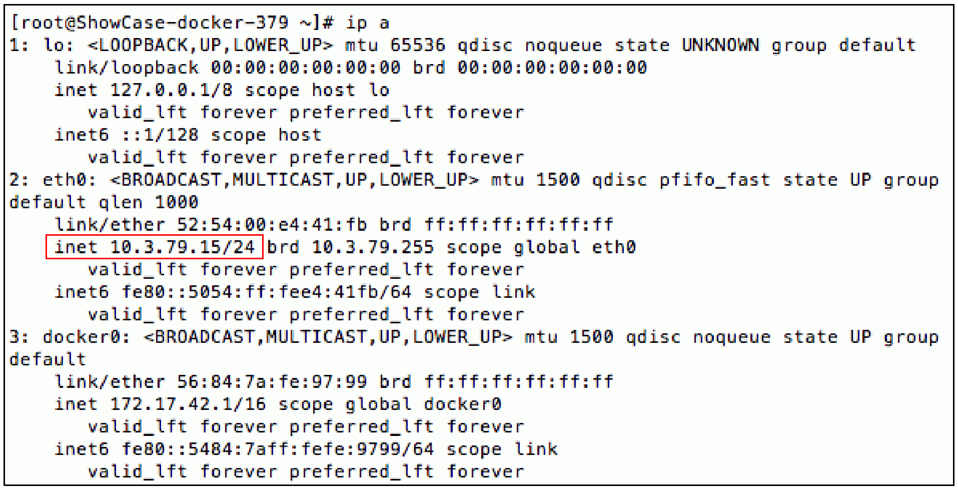
3. Type **`docker ps`** to check that the container is running as shown in Figure 5-43.



```
[root@ShowCase-docker-379 ~]# docker ps
CONTAINER ID        IMAGE                     COMMAND                CREATED
     STATUS              PORTS                     NAMES
2d5e330c5155        mdebeaux/odoo-db:latest   "/bin/bash /start_al   48 seconds ago
     Up 46 seconds       0.0.0.0:5432->5432/tcp    db
[root@ShowCase-docker-379 ~]#
```

Figure 5-43 Listing running containers on POWER8

4. Type **ip a** to display the list of available network interfaces as shown in Figure 5-44. Note the IP address of the network interface used to connect to the Internet because it will be needed in the next section.



```
[root@ShowCase-docker-379 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
    link/ether 52:54:00:e4:41:fb brd ff:ff:ff:ff:ff:ff
    inet 10.3.79.15/24 brd 10.3.79.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fee4:41fb/64 scope link
       valid_lft forever preferred_lft forever
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group
default
    link/ether 56:84:7a:fe:97:99 brd ff:ff:ff:ff:ff:ff
    inet 172.17.42.1/16 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 fe80::5484:7aff:fefe:9799/64 scope link
       valid_lft forever preferred_lft forever
```

Figure 5-44 Listing network interfaces on POWER8

5.3.4 Integrating with IBM Containers

To integrate with IBM Containers, we use the Secure Gateway service and IBM DataPower Gateway.

Configuring the Secure Gateway

To configure the Secure Gateway service, log on to Bluemix and complete the following steps:

1. Click **Catalog**, then enter `Secure Gateway` in the **Search** field as shown in Figure 5-45 on page 117. Click **Secure Gateway**.

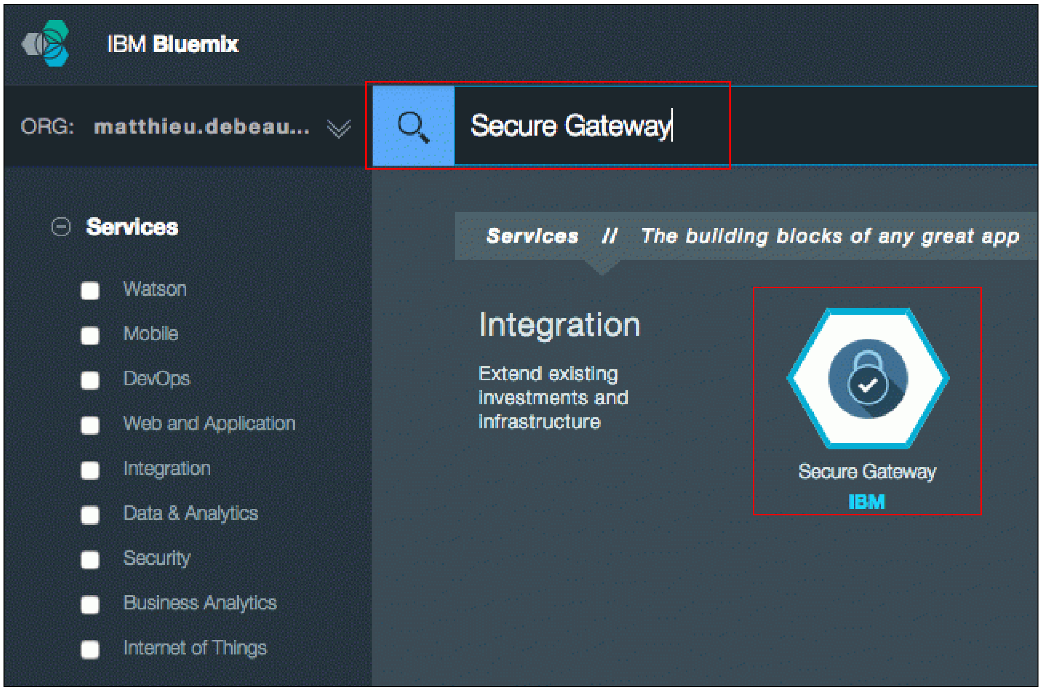
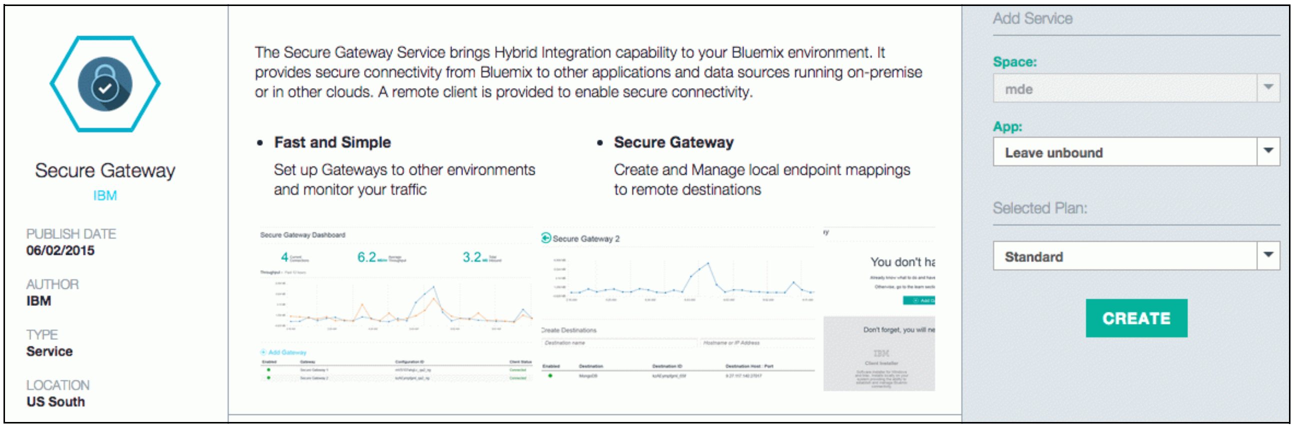


Figure 5-45 Bluemix catalog with Secure Gateway service

2. Click **CREATE** as shown in Figure 5-46.



PUBLISH DATE
06/02/2015

AUTHOR
IBM

TYPE
Service

LOCATION
US South

The Secure Gateway Service brings Hybrid Integration capability to your Bluemix environment. It provides secure connectivity from Bluemix to other applications and data sources running on-premise or in other clouds. A remote client is provided to enable secure connectivity.

- **Fast and Simple**
  Set up Gateways to other environments and monitor your traffic
- **Secure Gateway**
  Create and Manage local endpoint mappings to remote destinations

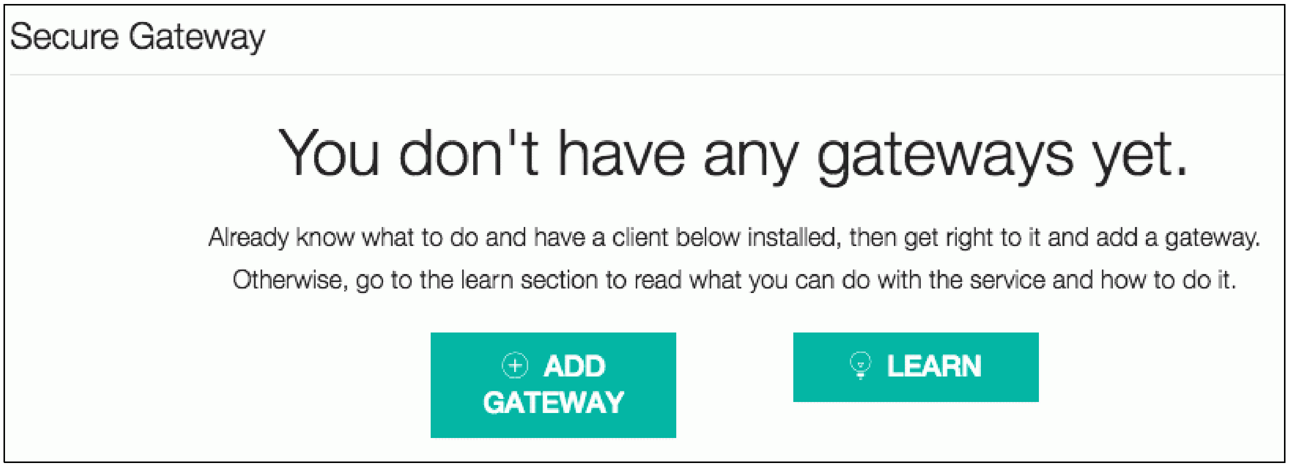
Space: mde

App: Leave unbound

Selected Plan: Standard

Figure 5-46 Secure Gateway service details

3. Click **ADD GATEWAY** to create a new gateway instance, as shown in Figure 5-47.



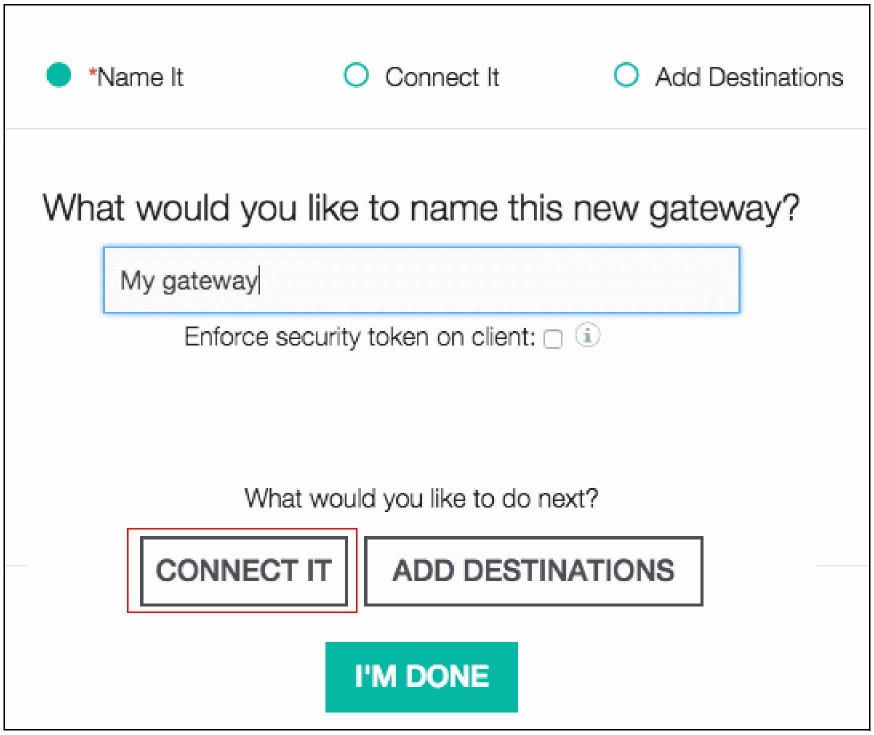
Secure Gateway

You don't have any gateways yet.

Already know what to do and have a client below installed, then get right to it and add a gateway. Otherwise, go to the learn section to read what you can do with the service and how to do it.

Figure 5-47 Secure Gateway landing page

4. Enter `My gateway` in the **What would you like to name this new gateway?** field, as shown in Figure 5-48. Click **CONNECT IT**.



What would you like to name this new gateway?

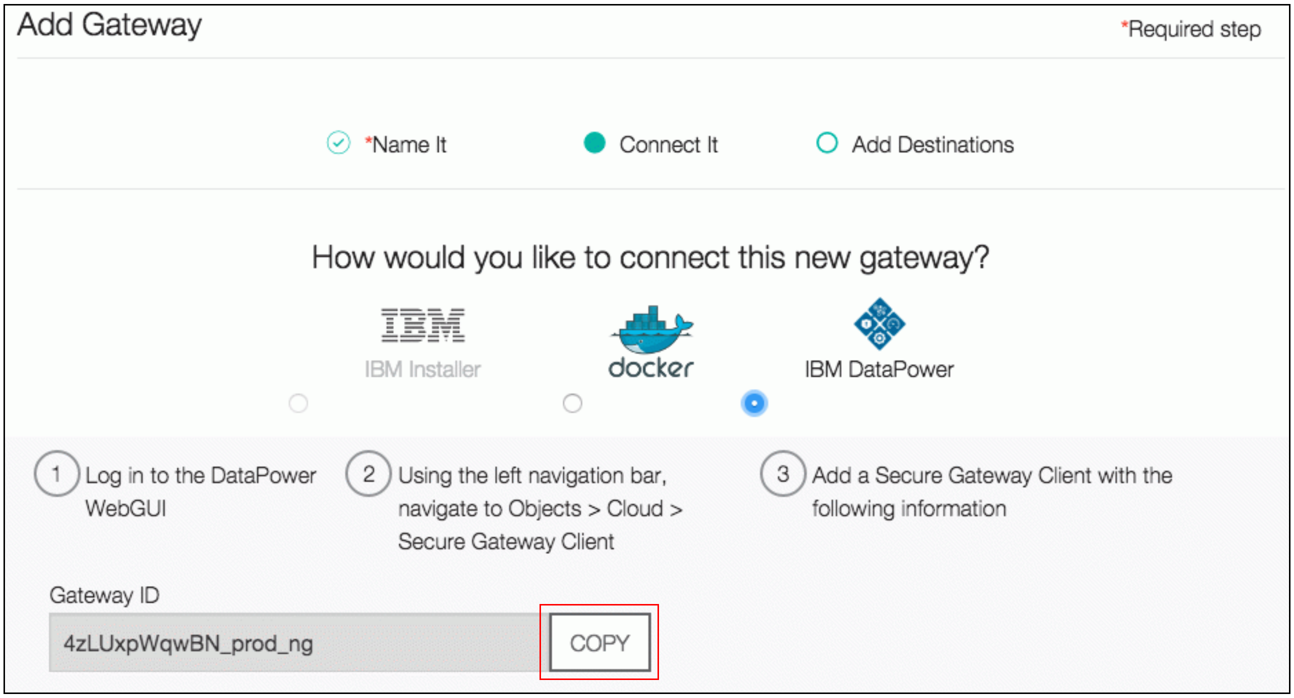
My gateway

Enforce security token on client: □

What would you like to do next?

Figure 5-48 Secure Gateway creation process

5. Click **IBM DataPower** as shown in Figure 5-49. Click **COPY** to copy the generated Gateway ID to clipboard.



Add Gateway

*Required step

✓ *Name It ● Connect It ○ Add Destinations

How would you like to connect this new gateway?

○ ○ ◉

1 Log in to the DataPower WebGUI

2 Using the left navigation bar, navigate to Objects > Cloud > Secure Gateway Client

3 Add a Secure Gateway Client with the following information

Gateway ID

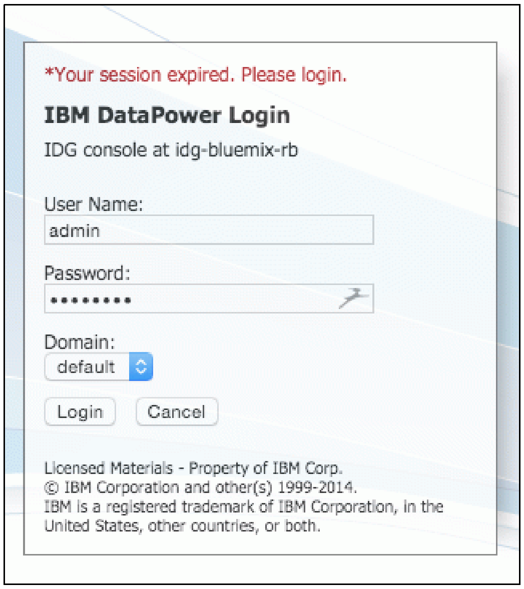
4zLUxpWqwBN_prod_ng

Figure 5-49 Secure Gateway connection process

Configuring the IBM DataPower Gateway

To continue the integration process, open a new web browser window, connect to IBM DataPower Gateway web interface, and complete the following steps:

1. Log in to the IBM DataPower Gateway web interface as shown in Figure 5-50.



*Your session expired. Please login.

IBM DataPower Login

IDG console at idg-bluemix-rb

User Name: admin

Password: ••••••••

Domain: default

Licensed Materials - Property of IBM Corp.
© IBM Corporation and other(s) 1999-2014.
IBM is a registered trademark of IBM Corporation, in the United States, other countries, or both.

Figure 5-50 IBM DataPower Gateway login page

2. Enter `secure gateway` in the Search field in the upper left corner of the window as shown in Figure 5-51. Click **Secure Gateway Client**.

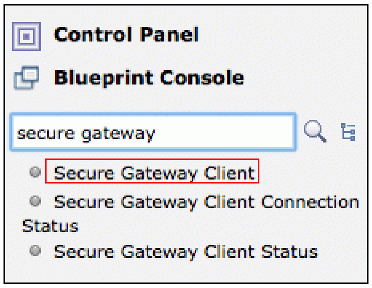
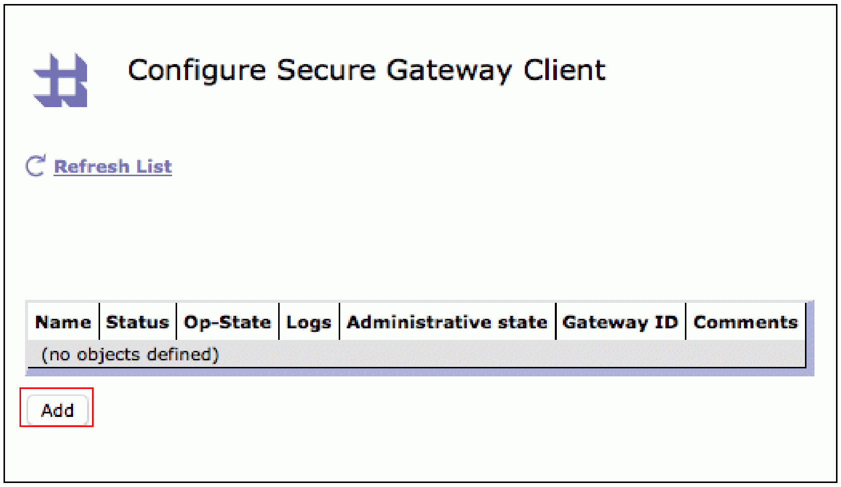

Control Panel
Blueprint Console
secure gateway
Secure Gateway Client
Secure Gateway Client Connection Status
Secure Gateway Client Status

Figure 5-51 IBM DataPower Gateway search

3. Click **Add** as shown in Figure 5-52.



Configure Secure Gateway Client

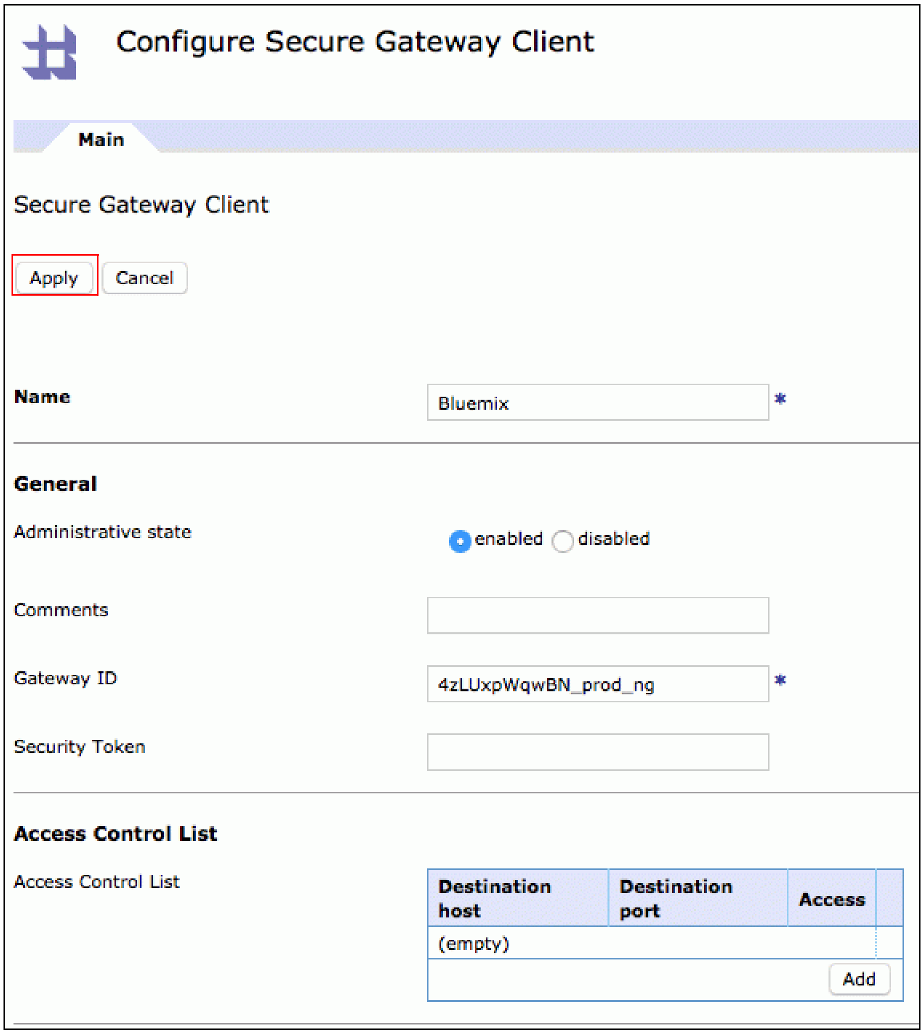
Refresh List

Name	Status	Op-State	Logs	Administrative state	Gateway ID	Comments
(no objects defined)						

Add

Figure 5-52 Secure Gateway Client list

4. Enter `Bluemix` in the Name field, paste the *Gateway ID* previously copied in the Gateway ID field, as shown in Figure 5-53. Click **Apply**.



Configure Secure Gateway Client

Secure Gateway Client

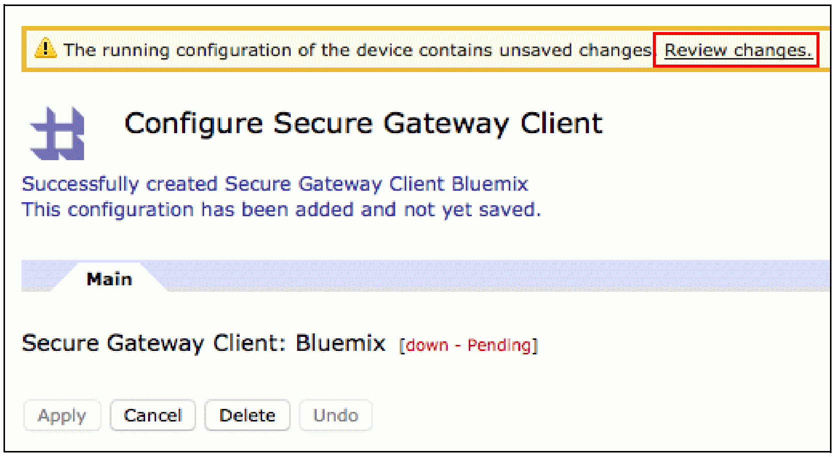
Name	Bluemix *
General	
Administrative state	● enabled ○ disabled
Comments	
Gateway ID	4zLUxpWqwBN_prod_ng *
Security Token	
Access Control List	

Access Control List

Destination host	Destination port	Access
(empty)		

Figure 5-53 Secure Gateway Client creation form

5. Click **Review changes** as shown in Figure 5-54.



⚠ The running configuration of the device contains unsaved changes. Review changes.

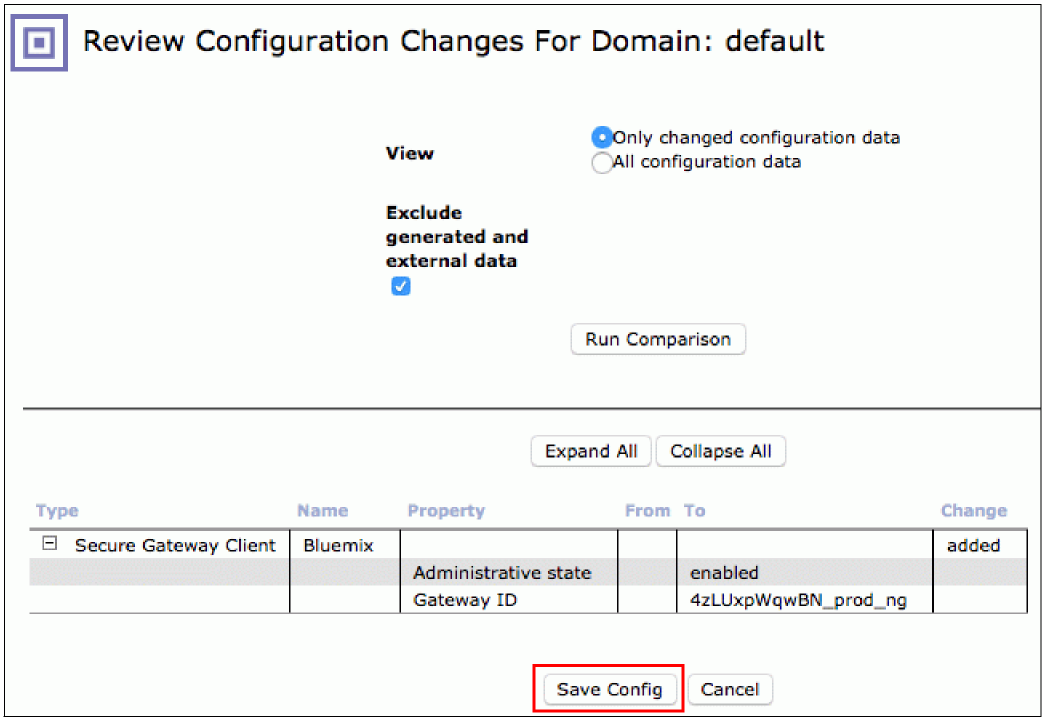
Configure Secure Gateway Client

Successfully created Secure Gateway Client Bluemix
This configuration has been added and not yet saved.

Secure Gateway Client: Bluemix [down - Pending]

Figure 5-54 Secure Gateway Client pending changes

6. Click **Save Config** as shown in Figure 5-55.



Review Configuration Changes For Domain: default

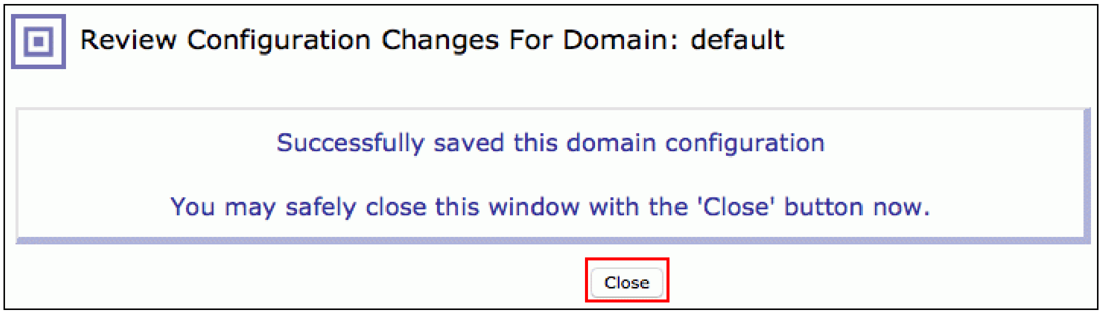
View: ● Only changed configuration data ○ All configuration data

Exclude generated and external data ☑

Type	Name	Property	From	To	Change
⊟ Secure Gateway Client	Bluemix				added
		Administrative state		enabled	
		Gateway ID		4zLUxpWqwBN_prod_ng	

Figure 5-55 IBM DataPower Gateway saving configuration

7. A confirmation message appears as shown in Figure 5-56. Click **Close** and close the web browser window.



Review Configuration Changes For Domain: default

Successfully saved this domain configuration

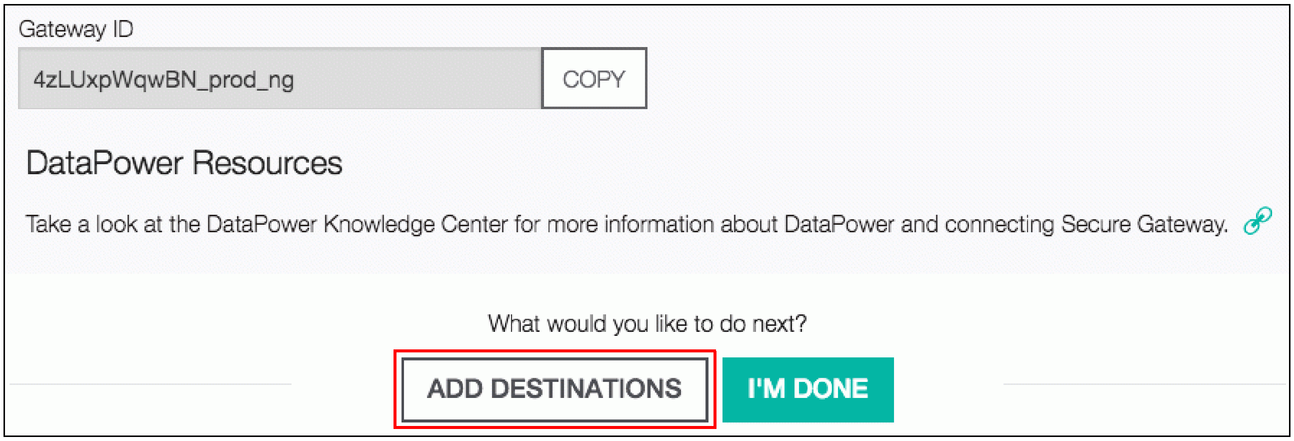
You may safely close this window with the 'Close' button now.

Figure 5-56 IBM DataPower Gateway configuration changes confirmation

Adding a destination in the Secure Gateway

To terminate the integration process, go back to the window displayed at the end of step 5 on page 119 and complete the following steps:

1. Click ADD DESTINATIONS as shown in Figure 5-57.



Gateway ID: 4zLUxpWqwBN_prod_ng

DataPower Resources

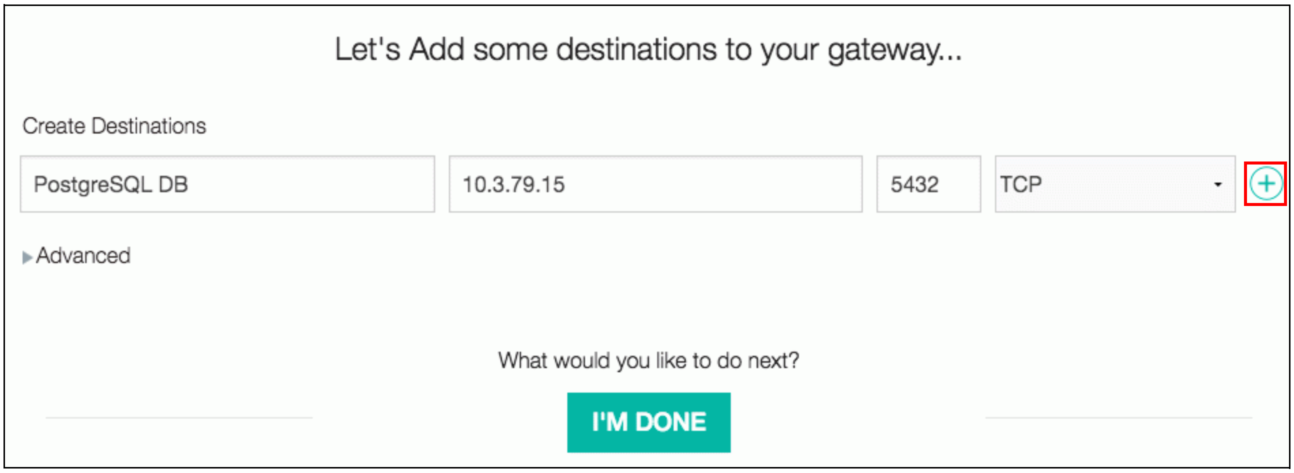
Take a look at the DataPower Knowledge Center for more information about DataPower and connecting Secure Gateway.

What would you like to do next?

Figure 5-57 Secure Gateway connection process continued

2. In the **Create Destinations** area, complete the following actions as shown in Figure 5-58:
   a. Enter `PostgreSQL DB` in the **Destination name** field.
   b. Enter *your_ubuntu_on_power_ip* that was retrieved in step 4 on page 116 in the **Hostname or IP Address** field.
   c. Enter `5432` in the **Port** field.
   d. Make sure `TCP` is selected in the list.
   e. Click the **plus** in the circle.

   > **Note:** For more information about **Advanced Options**, refer to Bluemix documentation:
   >
   > `https://www.ng.bluemix.net/docs/services/SecureGateway/sg_022.html#sg_009`



Let's Add some destinations to your gateway...

Create Destinations

PostgreSQL DB | 10.3.79.15 | 5432 | TCP

What would you like to do next?

Figure 5-58 Secure Gateway creating a new destination

3. Click **I'M DONE** to finish the gateway's creation as shown in Figure 5-59.

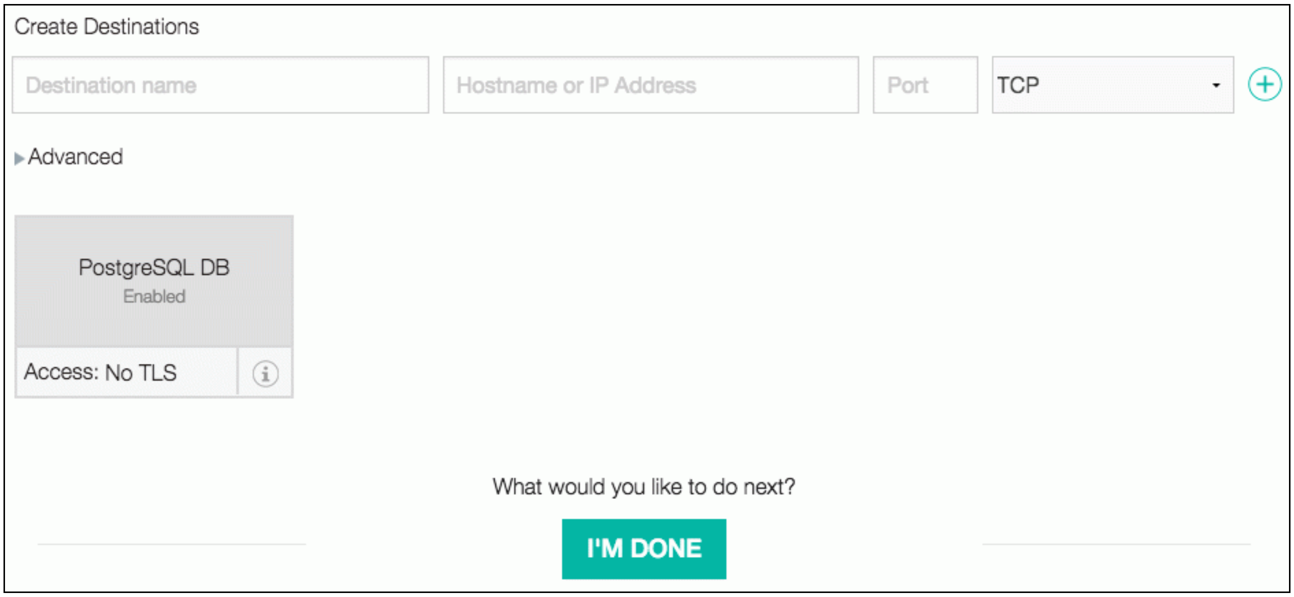


Figure 5-59 Secure Gateway final step

4. Check that the gateway's status displays "Connected" as shown in Figure 5-60.

Tip: This dashboard displays the gateway traffic in and out when in use.

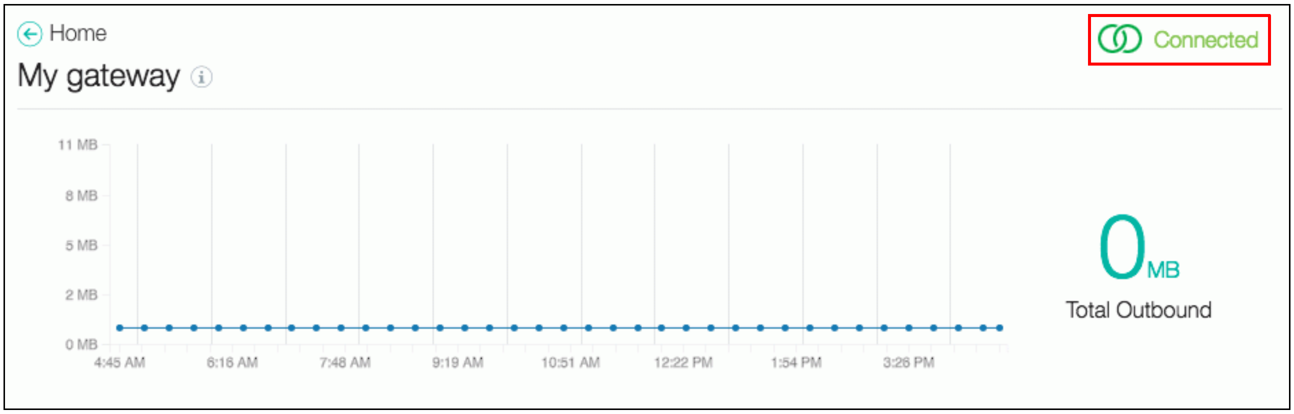


Figure 5-60 Secure Gateway status

5. Check that the destination's status displays "Enabled" as shown in Figure 5-61.

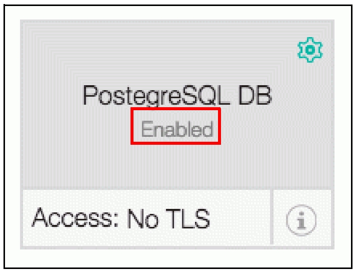
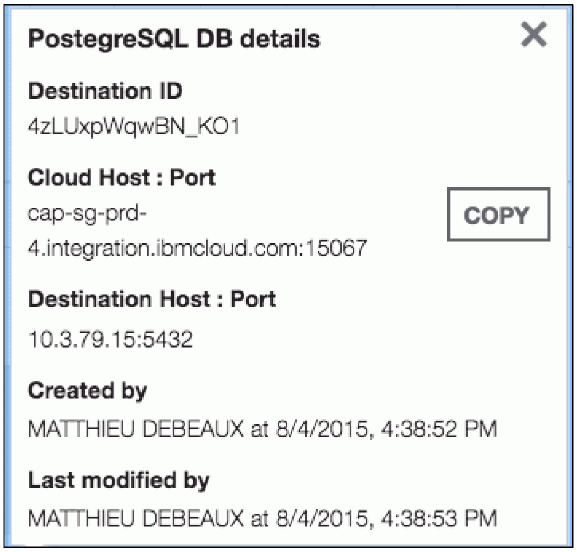


Figure 5-61 Secure Gateway destination status

6. Click the **round “i”** shaped icon to display the connection’s information as shown in Figure 5-62.



PostegreSQL DB details

Destination ID
4zLUxpWqwBN_KO1

Cloud Host : Port
cap-sg-prd-4.integration.ibmcloud.com:15067

Destination Host : Port
10.3.79.15:5432

Created by
MATTHIEU DEBEAUX at 8/4/2015, 4:38:52 PM

Last modified by
MATTHIEU DEBEAUX at 8/4/2015, 4:38:53 PM

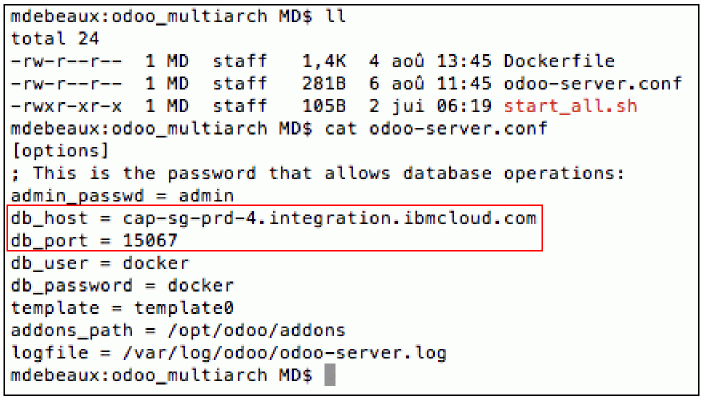
Figure 5-62 Secure Gateway destination information

7. Click **COPY** to copy to clipboard the generated **Cloud Host** and **Port**.

5.3.5 Running the ERP container

In order to finish the hybrid application setup, open a terminal or command window and complete the following steps:

1. Open the Odoo server configuration file that will be used to build your Odoo image and edit the following variables as shown in Figure 5-63:
   - **db_host**: Refers to the host that will be hosting the PostgreSQL database. Enter the *Cloud Host* name copied from the Secure Gateway destination in step 7.
   - **db_port**: Refers to the port that will be used to connect to the PostgreSQL database. Enter the *Port* number copied from the Secure Gateway destination in step 7.

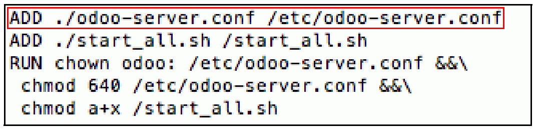


```
mdebeaux:odoo_multiarch MD$ ll
total 24
-rw-r--r--  1 MD  staff   1,4K  4 aoû 13:45 Dockerfile
-rw-r--r--  1 MD  staff   281B  6 aoû 11:45 odoo-server.conf
-rwxr-xr-x  1 MD  staff   105B  2 jui 06:19 start_all.sh
mdebeaux:odoo_multiarch MD$ cat odoo-server.conf
[options]
; This is the password that allows database operations:
admin_passwd = admin
db_host = cap-sg-prd-4.integration.ibmcloud.com
db_port = 15067
db_user = docker
db_password = docker
template = template0
addons_path = /opt/odoo/addons
logfile = /var/log/odoo/odoo-server.log
mdebeaux:odoo_multiarch MD$
```

Figure 5-63 Odoo configuration file

Note: The **db_user** and **db_password** values displayed depend on the configuration of your PostgreSQL database.

2. Open your **Dockerfile** to ensure that the Odoo server configuration file is copied to your container during the build as shown in Figure 5-64.



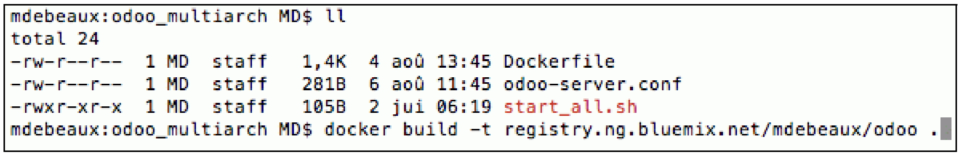
```
ADD ./odoo-server.conf /etc/odoo-server.conf
ADD ./start_all.sh /start_all.sh
RUN chown odoo: /etc/odoo-server.conf &&\
 chmod 640 /etc/odoo-server.conf &&\
 chmod a+x /start_all.sh
```

Figure 5-64 Dockerfile excerpt to add Odoo configuration file during build

Tip: For more information about configuring your Odoo container, refer to Odoo's documentation:

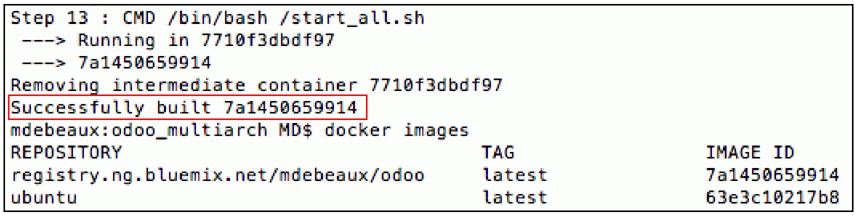
https://registry.hub.docker.com/_/odoo

3. Ensure that you are in the repository containing the Dockerfile for the Odoo image. Type **`docker build -t registry.ng.bluemix.net/`***`your_registry_name`***`/odoo .`** to build your image as shown in Figure 5-65 and the result of the build shown in Figure 5-66.



```
mdebeaux:odoo_multiarch MD$ ll
total 24
-rw-r--r--  1 MD  staff   1,4K  4 aoû 13:45 Dockerfile
-rw-r--r--  1 MD  staff   281B  6 aoû 11:45 odoo-server.conf
-rwxr-xr-x  1 MD  staff   105B  2 jui 06:19 start_all.sh
mdebeaux:odoo_multiarch MD$ docker build -t registry.ng.bluemix.net/mdebeaux/odoo .
```

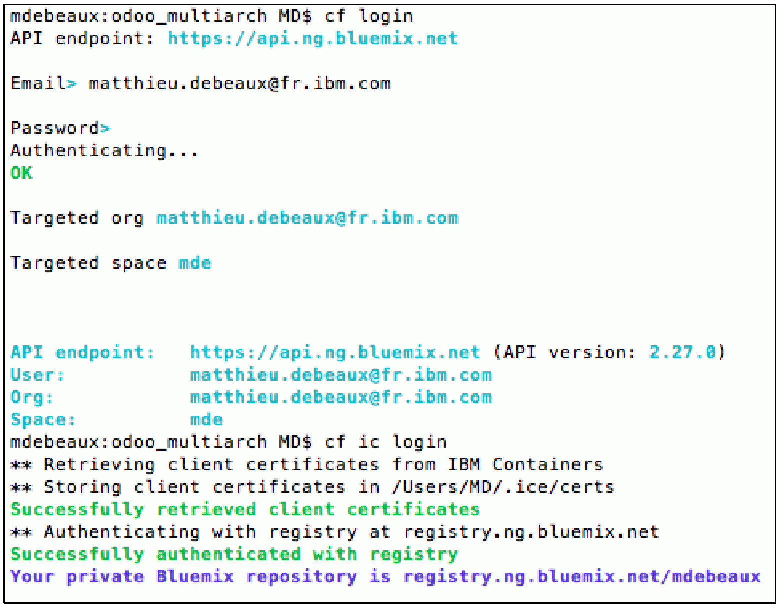
Figure 5-65 Building Odoo container



```
Step 13 : CMD /bin/bash /start_all.sh
 ---> Running in 7710f3dbdf97
 ---> 7a1450659914
Removing intermediate container 7710f3dbdf97
Successfully built 7a1450659914
mdebeaux:odoo_multiarch MD$ docker images
REPOSITORY                                 TAG                 IMAGE ID
registry.ng.bluemix.net/mdebeaux/odoo      latest              7a1450659914
ubuntu                                     latest              63e3c10217b8
```

Figure 5-66 Odoo container built

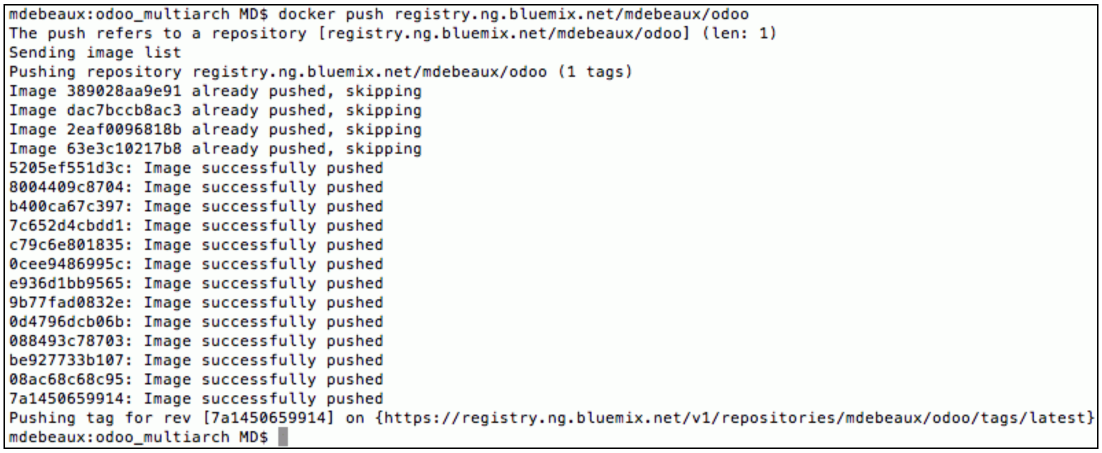
4. Log in to Bluemix and IBM Containers with command-line interface (CLI) by typing **`cf login`** and **`cf ic login`** as shown in Figure 5-67.



```
mdebeaux:odoo_multiarch MD$ cf login
API endpoint: https://api.ng.bluemix.net

Email> matthieu.debeaux@fr.ibm.com

Password>
Authenticating...
OK

Targeted org matthieu.debeaux@fr.ibm.com

Targeted space mde



API endpoint:   https://api.ng.bluemix.net (API version: 2.27.0)
User:           matthieu.debeaux@fr.ibm.com
Org:            matthieu.debeaux@fr.ibm.com
Space:          mde
mdebeaux:odoo_multiarch MD$ cf ic login
** Retrieving client certificates from IBM Containers
** Storing client certificates in /Users/MD/.ice/certs
Successfully retrieved client certificates
** Authenticating with registry at registry.ng.bluemix.net
Successfully authenticated with registry
Your private Bluemix repository is registry.ng.bluemix.net/mdebeaux
```

Figure 5-67   Logging in to Bluemix and IBM Containers

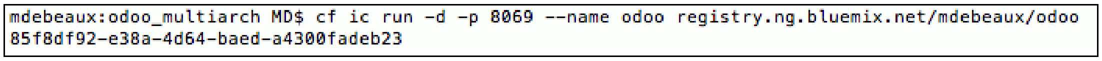
5. Type **`docker push registry.ng.bluemix.net/`***your_registry_name***`/odoo`** to push the newly built image to Bluemix Registry, as shown in Figure 5-68.



```
mdebeaux:odoo_multiarch MD$ docker push registry.ng.bluemix.net/mdebeaux/odoo
The push refers to a repository [registry.ng.bluemix.net/mdebeaux/odoo] (len: 1)
Sending image list
Pushing repository registry.ng.bluemix.net/mdebeaux/odoo (1 tags)
Image 389028aa9e91 already pushed, skipping
Image dac7bccb8ac3 already pushed, skipping
Image 2eaf0096818b already pushed, skipping
Image 63e3c10217b8 already pushed, skipping
5205ef551d3c: Image successfully pushed
8004409c8704: Image successfully pushed
b400ca67c397: Image successfully pushed
7c652d4cbdd1: Image successfully pushed
c79c6e801835: Image successfully pushed
0cee9486995c: Image successfully pushed
e936d1bb9565: Image successfully pushed
9b77fad0832e: Image successfully pushed
0d4796dcb06b: Image successfully pushed
088493c78703: Image successfully pushed
be927733b107: Image successfully pushed
08ac68c68c95: Image successfully pushed
7a1450659914: Image successfully pushed
Pushing tag for rev [7a1450659914] on {https://registry.ng.bluemix.net/v1/repositories/mdebeaux/odoo/tags/latest}
mdebeaux:odoo_multiarch MD$
```

Figure 5-68   Pushing Odoo image to Bluemix Registry

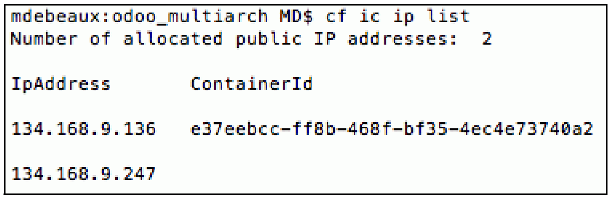
6. Type **`cf ic run -d -p 8069 --name odoo registry.ng.bluemix.net/`***your_registry_name***`/odoo`** to launch the Odoo container as shown in Figure 5-69.



```
mdebeaux:odoo_multiarch MD$ cf ic run -d -p 8069 --name odoo registry.ng.bluemix.net/mdebeaux/odoo
85f8df92-e38a-4d64-baed-a4300fadeb23
```

Figure 5-69   Launching Odoo container in IBM Containers

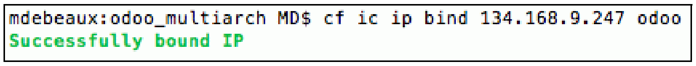
7. Type **cf ic ip list** to list the available IP addresses, as shown in Figure 5-70.



```
mdebeaux:odoo_multiarch MD$ cf ic ip list
Number of allocated public IP addresses:  2

IpAddress       ContainerId

134.168.9.136   e37eebcc-ff8b-468f-bf35-4ec4e73740a2

134.168.9.247
```

Figure 5-70 Listing available IP addresses

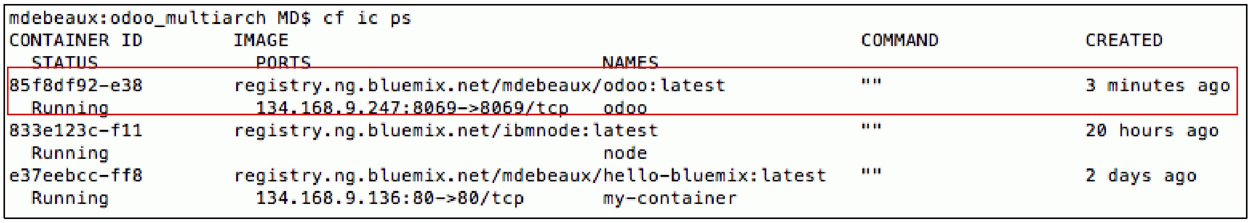
8. Select an IP address that is currently available and bind it to the Odoo container by typing **cf ic ip bind** *your_ip_address* **odoo**, as shown in Figure 5-71.



```
mdebeaux:odoo_multiarch MD$ cf ic ip bind 134.168.9.247 odoo
Successfully bound IP
```

Figure 5-71 Binding IP address with Odoo container

9. Type **cf ic ps** to check that the Odoo container is running and bound to an external IP on port 8069, as shown in Figure 5-72.



```
mdebeaux:odoo_multiarch MD$ cf ic ps
CONTAINER ID        IMAGE                                               COMMAND             CREATED
  STATUS              PORTS                          NAMES
85f8df92-e38        registry.ng.bluemix.net/mdebeaux/odoo:latest        ""                  3 minutes ago
  Running             134.168.9.247:8069->8069/tcp   odoo
833e123c-f11        registry.ng.bluemix.net/ibmnode:latest              ""                  20 hours ago
  Running                                          node
e37eebcc-ff8        registry.ng.bluemix.net/mdebeaux/hello-bluemix:latest ""                2 days ago
  Running             134.168.9.136:80->80/tcp       my-container
```

Figure 5-72 IBM Containers list of running containers

10. Open a web browser and go to *http://your_ip_address:8069* to display Odoo's landing page, as shown in Figure 5-73.

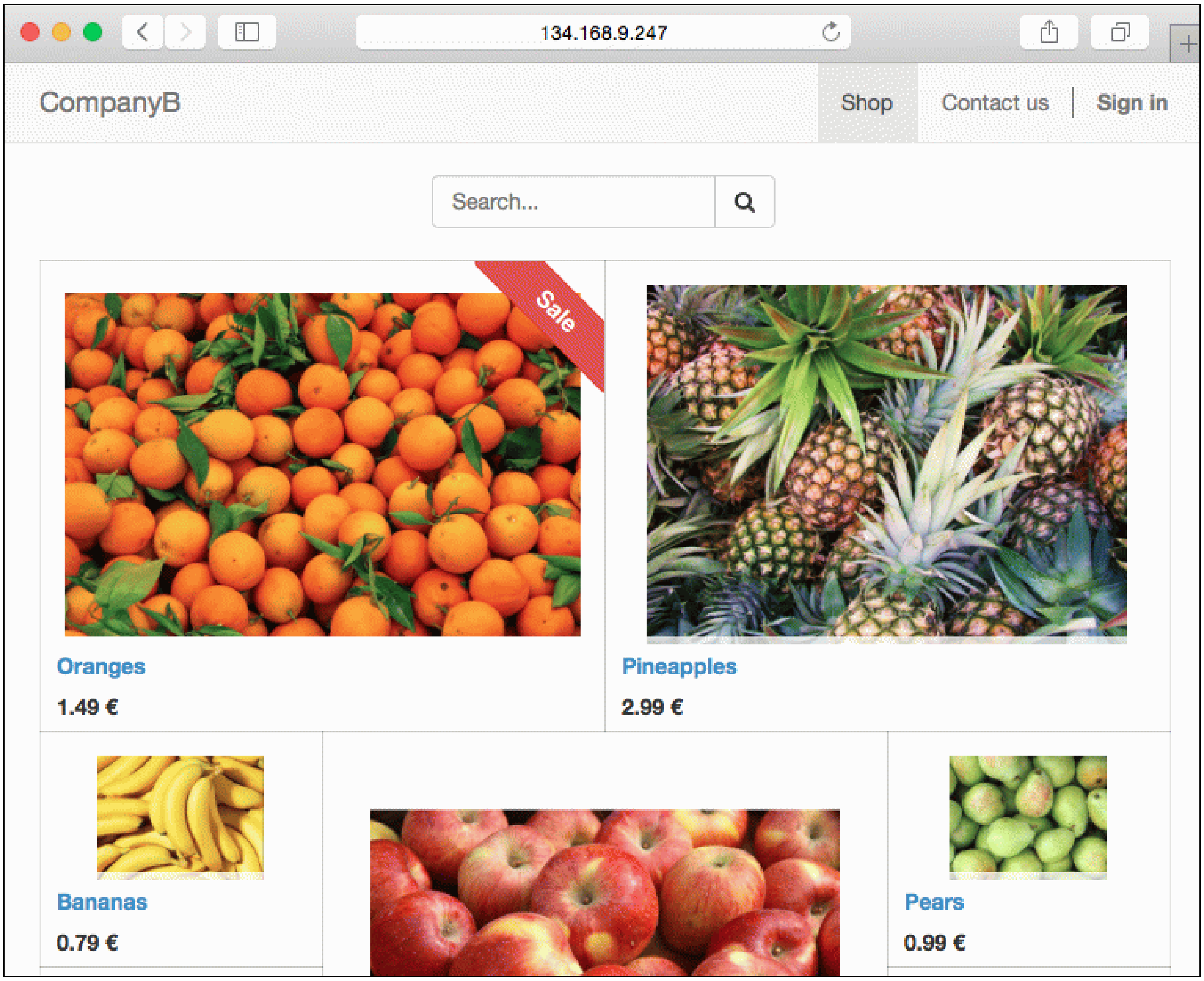


Figure 5-73   Odoo's web-front for CompanyB

Note: In this use case, Odoo is configured to present CompanyB sales website with pre-loaded data to display grocery items.

6

Connecting Bluemix applications to your local (on-premises) enterprise SAP system

This chapter covers the step-by-step process to connect your application on Bluemix to your local (on-premises) enterprise SAP system by using the Secure Gateway and Connect & Compose services. This chapter has the following sections:

- 6.1, “Solution overview” on page 132
- 6.2, “Step-by-step implementation” on page 134
- 6.3, “Testing the API” on page 149

Recording of this scenario: You can find the recording of this scenario at the following link:

`https://www.youtube.com/watch?v=QMRDg2ZxRj8&feature=youtu.be`

© Copyright IBM Corp. 2015. All rights reserved.

6.1 Solution overview

The main objective of the application covered in this scenario is to be able to approve purchase orders inside the SAP system directly from mobile phones. The goal is to reduce time to deliver and purchase any order and also give mobility to the management level. With this application, we can speed up the purchase process and reduce the costs during the supply chain process.

Business case for this scenario: For more information about the business case for this scenario and the company profile, refer to 1.3.10, "Solutions and actions by the CompanyB CIO" on page 13.

6.1.1 Infrastructure overview

This chapter shows how to create your application connected to a local (on-premises) SAP system and how to set up Secure Gateway and Connect & Compose the configuration of each one.

Figure 6-1 shows a high-level overview of the Bluemix services and the connectivity between them and the on-premises SAP infrastructure.

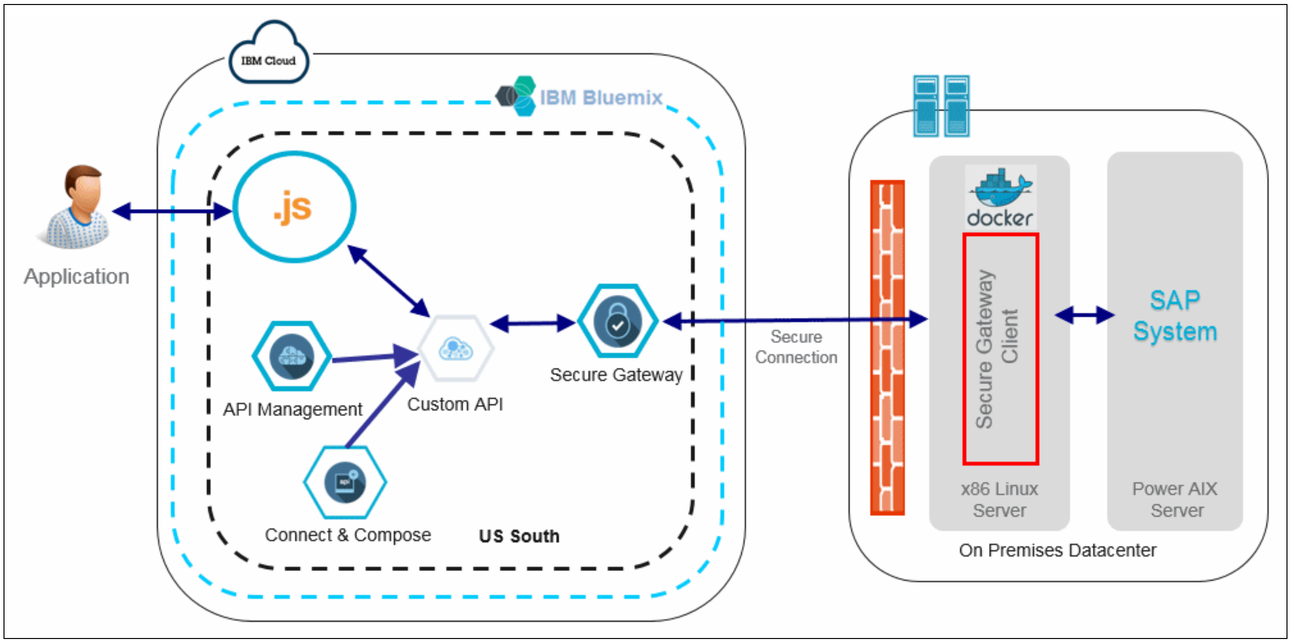


Figure 6-1 High-level infrastructure overview

The target SAP ECC (Enterprise Control Component) system resides in an on-premises datacenter running IBM AIX® on Power servers behind the customer firewall infrastructure.

To set up the secure connection between this infrastructure and the Bluemix application, we installed a secure gateway client on the same network by using a Docker container running inside an x86 Linux virtual machine.

With the secure connection setup done, we were able to create an application programming interface (API) to consume resources from the SAP system using the Connect & Compose Bluemix services. The Connect & Compose service connects to the Business Object Repository and easily bring all the SAP Business Application Programming Interfaces (BAPIs) to help you decide the most suitable API to use in your application. The Connect & Compose service has two more connections to the SAP system, totaling three:

- Remote Function Call (RFC)
- Business Object Repository (BOR)
- Application Link Enabling (ALE)

After you connect your SAP system to Bluemix by using the Connect & Compose service, you are able to save and share these new APIs with your Bluemix applications as well as the Bluemix API Management services. The API Management services allow you to manage all your APIs from one single point and also do the following activities:

- Check usage of each API call
- Run analytics on the API
- Create and approve API plans
- Execute user and roles management over API
- Create SLL profiles

When you are done with your infrastructure and logical connectivity setup, you are able to call your API by using any Software Development Kit (SDK) that is available on Bluemix. Following are the most commonly used SDKs:

- Liberty for Java (WebSphere)
- SDK for Node.js (JavaScript)
- SDK for GO
- SDK for PHP
- SDK for Python
- SDK for Ruby

6.1.2 Infrastructure requirements

This section shows all the required infrastructure, software versions, users, and passwords that are required to build this application.

SAP application side

The following lists the SAP application-side modules.

SAP ECC system with FI/CO modules

The target SAP ECC system needs to have Financial Accounting (FI) and Controlling (CO) modules installed to enable purchase order approvals. The SAP BAPI’s purchase order with the methods GetItemsForRelease and Release are available from SAP version 4.0A. You can add all available BAPIs into the SAP transaction BAPI.

SAP user, password, and client of this existing SAP ECC system

The SAP user must be created on the production SAP client. Most of the SAP production systems have only one SAP production client, but some development systems might have more than one. Each client has its own configuration information, users, and passwords. Ensure that you use the correct user, password, and client information.

SAP role and profile to check and approve purchase orders inside the SAP client

The SAP user must be created with the appropriate SAP profiles and roles to query and approve purchase orders on the systems. The required authorization object name is M_EINK_FRG, but you need to ask your system administration about existing SAP roles and profiles for managing the purchase orders.

SAP NetWeaver Gateway running on the SAP server

The SAP NetWeaver Gateway is required to set up the connection between the Bluemix and the SAP system using the Secure Gateway and Connect & Compose services. To set up the SAP NetWeaver Gateway in your SAP system, use the SAP transaction SPRO (SAP Project Reference Object) and then click **Display SAP Reference IMG**. The activation of the gateway can be found under the following path: **SAP Customizing Implementation Guide** → **SAP NetWeaver** → **Gateway** → **OData Channel** → **Configuration** → **Activate or Deactivate SAP NetWeaver Gateway**.

Secure Gateway Client

The following requirements are needed for the Secure Gateway Client.

Linux server

To set up the Docker engine, we need a Linux instance. In our scenario, we use Red Hat Enterprise Linux 7 to install the Docker engine. Docker engine is not available for all Red Hat Linux, for supported versions go to the following site:

`https://docs.docker.com/installation/`

Linux user and password or sudo

To install the Docker engine into a Linux Server, you must be root or at least have **`sudo`** privileges.

Docker engine

The procedure to install the Docker engine in each operating system is different. You can use the instructions on this page:

`https://docs.docker.com/installation/rhel`

Secure gateway client Docker image

To run the Secure Gateway client inside a Docker container connected to your Bluemix application, download the *ibmcom/secure-gateway-client* image and run it with the specified parameters from Bluemix. The full command that must be executed in your Docker engine is displayed during the Secure Gateway setup.

Bluemix

To start a Bluemix application, you need an account inside the IBM Cloud Bluemix.

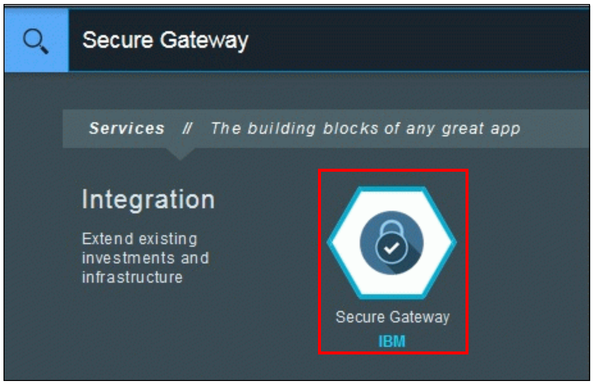
6.2 Step-by-step implementation

This section provides step-by-step instructions to set up the Secure Gateway and Connect & Compose services in order to expose your SAP BAPI transactions in your Bluemix application using the API calls.

6.2.1 Setting up the Secure Gateway service

These steps show how to set up the Bluemix Secure Gateway service and client. To execute this step, you need to have your Linux server with Docker engine prepared to run containers as described in the section “Secure Gateway Client” on page 134:

1. Log in to Bluemix and click **CATALOG**.
2. Search for Secure Gateway and click the image as shown in Figure 6-2.

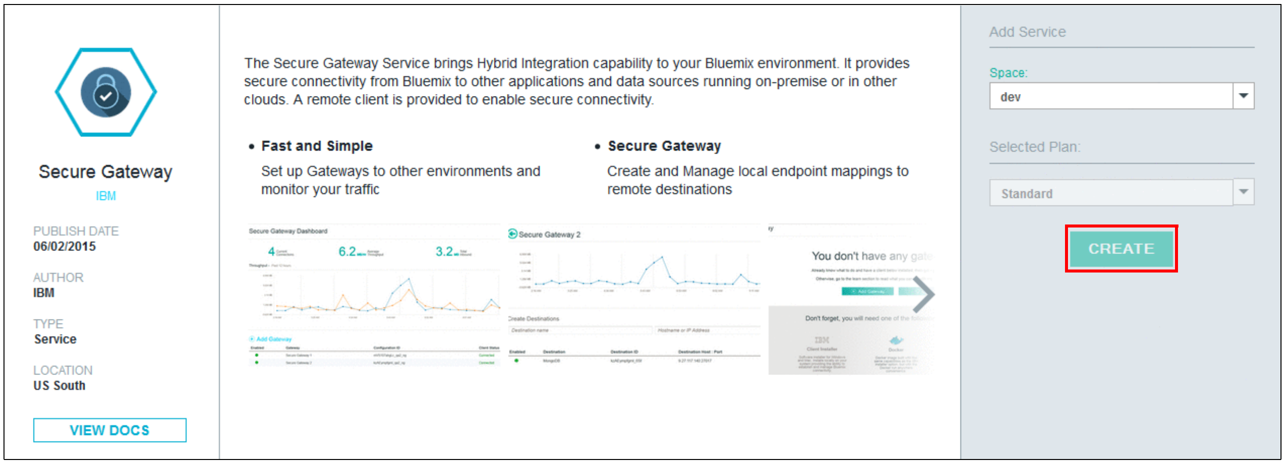


Secure Gateway

Services // The building blocks of any great app
Integration
Extend existing investments and infrastructure
Secure Gateway
IBM

Figure 6-2 Secure Gateway service

3. Select the space and plan and click **CREATE** as shown in Figure 6-3.



Secure Gateway
IBM

PUBLISH DATE
06/02/2015

AUTHOR
IBM

TYPE
Service

LOCATION
US South

The Secure Gateway Service brings Hybrid Integration capability to your Bluemix environment. It provides secure connectivity from Bluemix to other applications and data sources running on-premise or in other clouds. A remote client is provided to enable secure connectivity.

- **Fast and Simple**
  Set up Gateways to other environments and monitor your traffic
- **Secure Gateway**
  Create and Manage local endpoint mappings to remote destinations

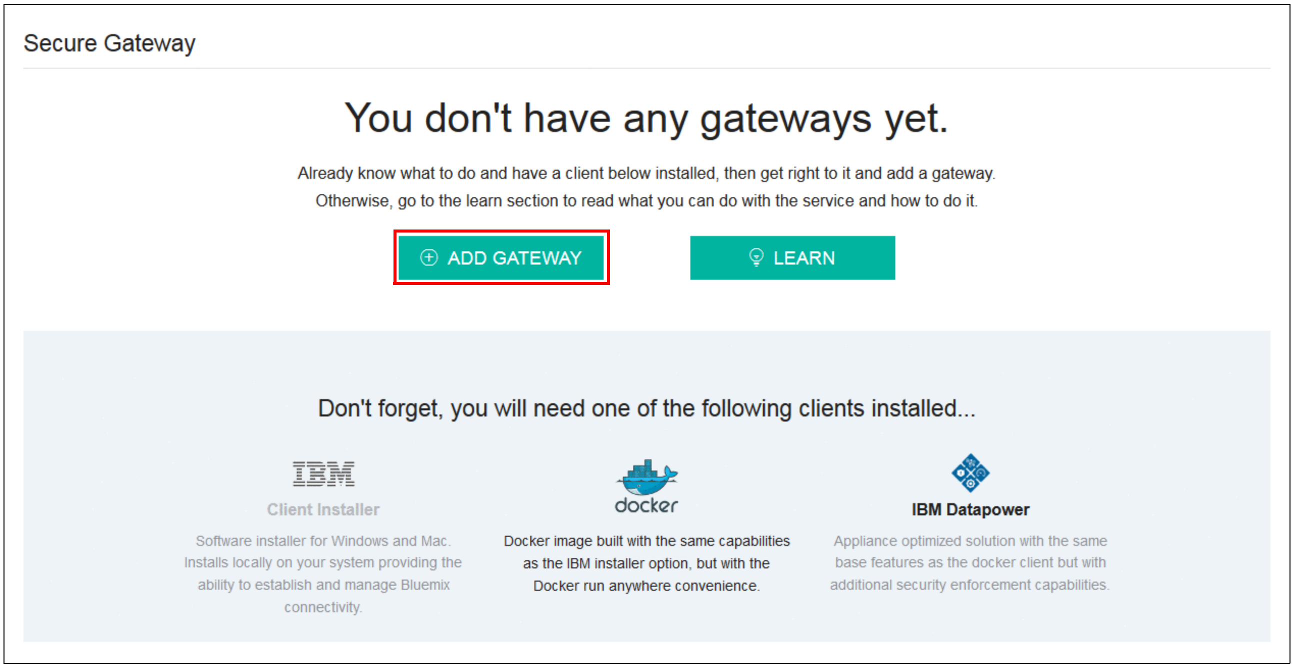
Add Service

Space: dev

Selected Plan: Standard

Figure 6-3 Secure Gateway plan options

4. To create your first gateway, click **ADD GATEWAY** as shown Figure 6-4 or click **LEARN** to see all the container options.



Secure Gateway

You don't have any gateways yet.

Already know what to do and have a client below installed, then get right to it and add a gateway.
Otherwise, go to the learn section to read what you can do with the service and how to do it.

Don't forget, you will need one of the following clients installed...

Client Installer
Software installer for Windows and Mac. Installs locally on your system providing the ability to establish and manage Bluemix connectivity.

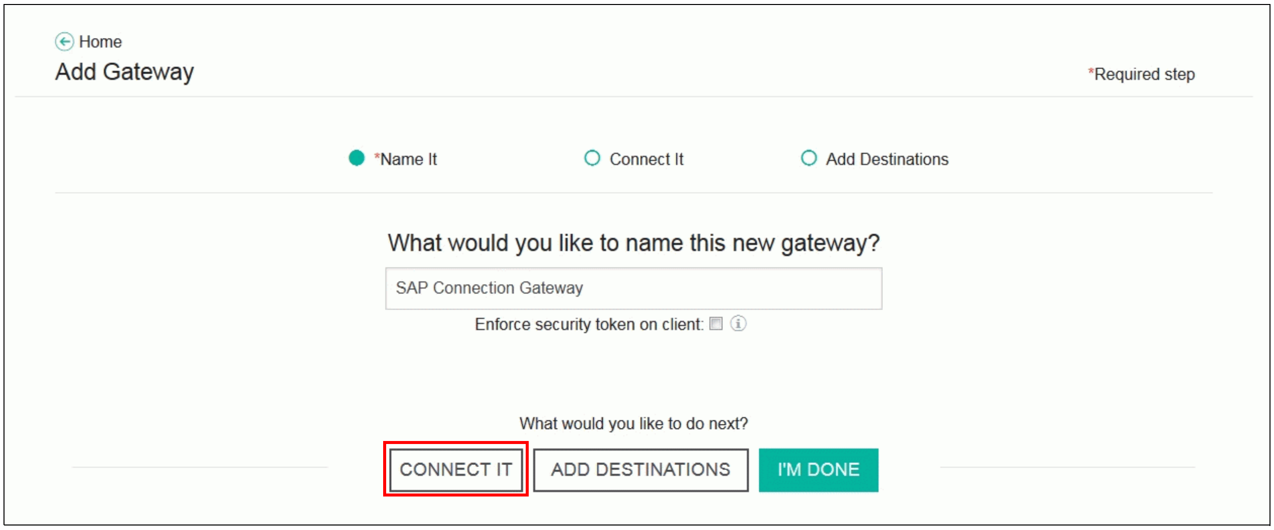
docker
Docker image built with the same capabilities as the IBM installer option, but with the Docker run anywhere convenience.

IBM Datapower
Appliance optimized solution with the same base features as the docker client but with additional security enforcement capabilities.

Figure 6-4 Secure Gateway first page

5. Define the gateway name and click **CONNECT IT** as shown in Figure 6-5.

> **Tip:** The Secure Gateway name defined here is required during the Connect & Compose setup procedure. Each Secure Connection setup has its own name.



Add Gateway

*Required step

● *Name It ○ Connect It ○ Add Destinations

What would you like to name this new gateway?

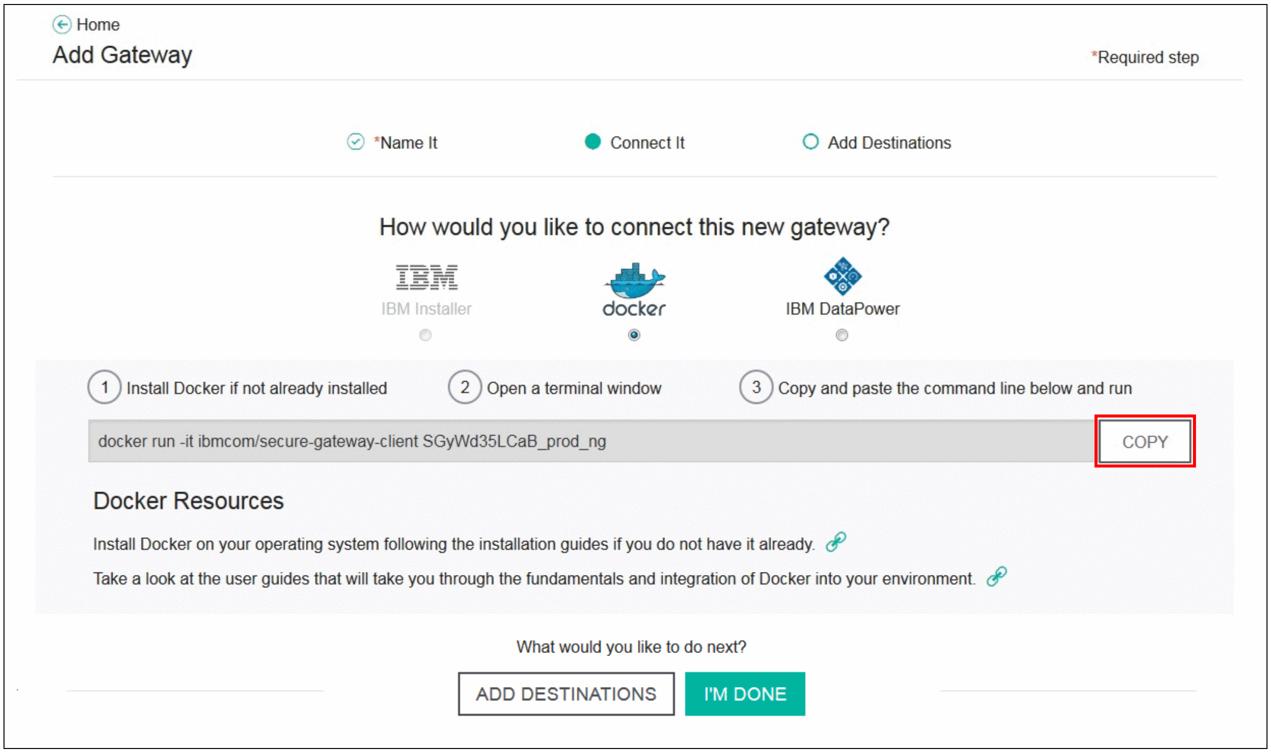
SAP Connection Gateway

Enforce security token on client: □

What would you like to do next?

Figure 6-5 Secure Gateway name definition

6. Copy the required Docker command by using **COPY** as shown in Figure 6-6.



Add Gateway

*Required step

How would you like to connect this new gateway?

1 Install Docker if not already installed

2 Open a terminal window

3 Copy and paste the command line below and run

```
docker run -it ibmcom/secure-gateway-client SGyWd35LCaB_prod_ng
```

Docker Resources

Install Docker on your operating system following the installation guides if you do not have it already.

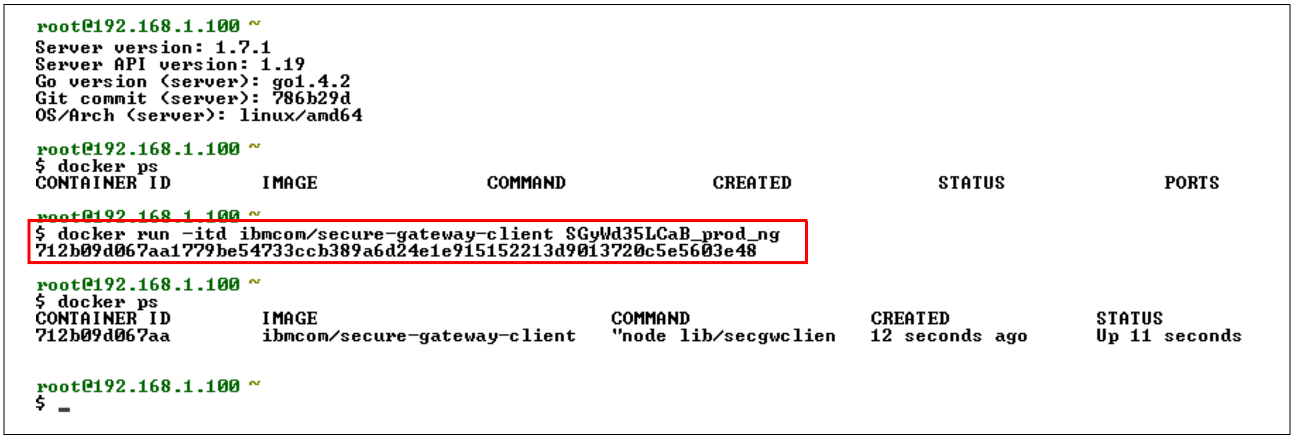
Take a look at the user guides that will take you through the fundamentals and integration of Docker into your environment.

What would you like to do next?

Figure 6-6 Secure Gateway client connection command

7. Run the **`Docker`** command inside your Linux image that has the Docker engine installed, as shown in Figure 6-7.

> **Tip:** The command provided by Bluemix starts the container and holds your Bourne Again SHell (bash) for display logs. You can use the **`-d`** option to run the container in the background, or replace **`-it`** by **`-itd`** in the command line provided.

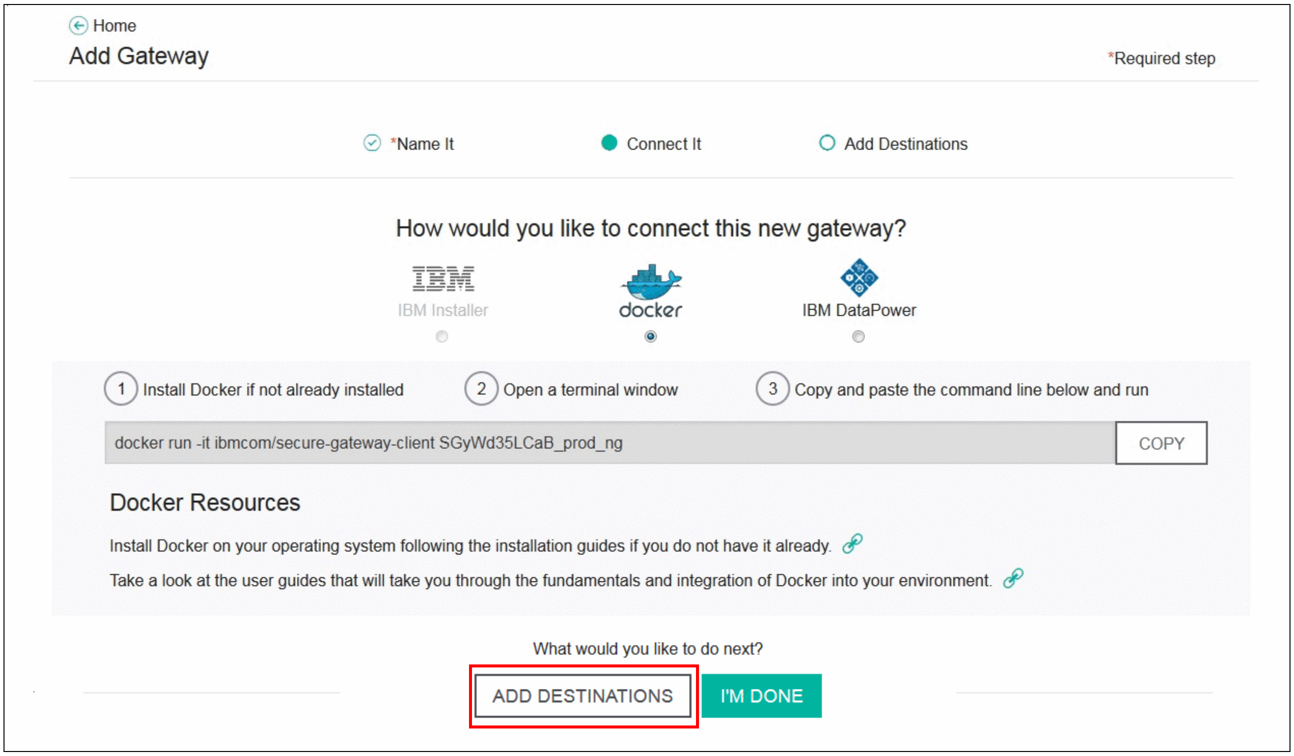


```
root@192.168.1.100 ~
Server version: 1.7.1
Server API version: 1.19
Go version (server): go1.4.2
Git commit (server): 786b29d
OS/Arch (server): linux/amd64

root@192.168.1.100 ~
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS

root@192.168.1.100 ~
$ docker run -itd ibmcom/secure-gateway-client SGyWd35LCaB_prod_ng
712b09d067aa1779be54733ccb389a6d24e1e915152213d9013720c5e5603e48

root@192.168.1.100 ~
$ docker ps
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS
712b09d067aa        ibmcom/secure-gateway-client   "node lib/secgwclien     12 seconds ago      Up 11 seconds

root@192.168.1.100 ~
$ _
```

Figure 6-7 Secure Gateway client Docker image start

> **Tip:** If you start the container in background mode, you can use the `docker ps` command to check the status.

8. You can add destinations to your gateway as soon as you get your client secure gateway container up and running. To do that, click **ADD DESTINATIONS**, as shown in Figure 6-8.



Home

Add Gateway

*Required step

How would you like to connect this new gateway?

1 Install Docker if not already installed
2 Open a terminal window
3 Copy and paste the command line below and run

```
docker run -it ibmcom/secure-gateway-client SGyWd35LCaB_prod_ng
```

Docker Resources

Install Docker on your operating system following the installation guides if you do not have it already.

Take a look at the user guides that will take you through the fundamentals and integration of Docker into your environment.

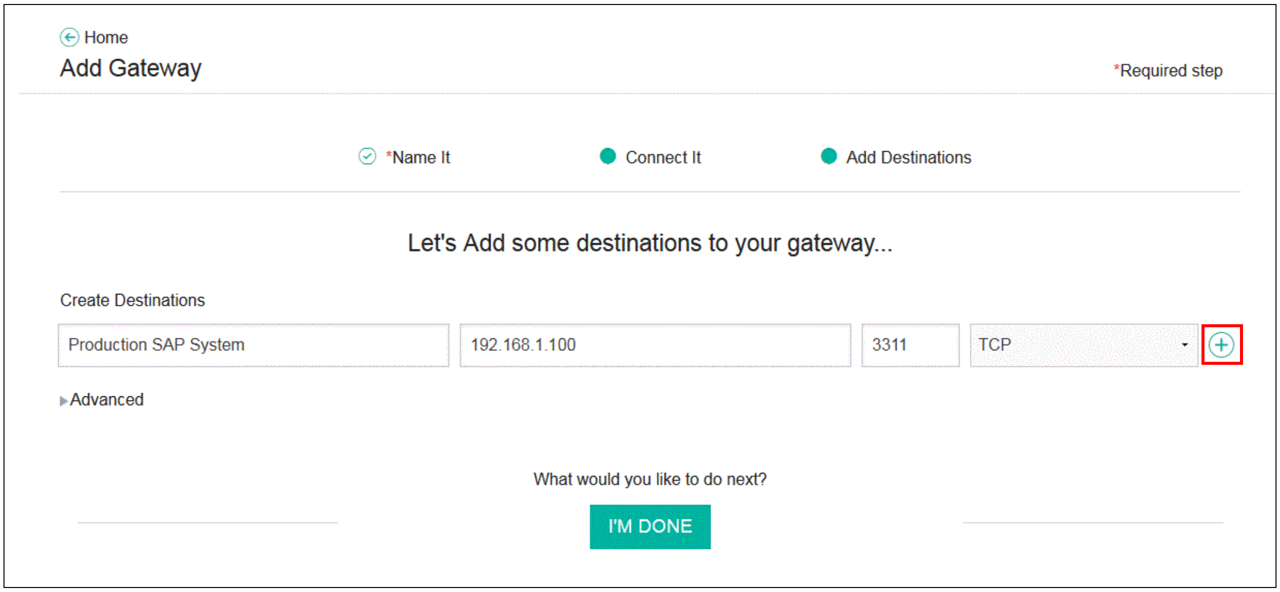
What would you like to do next?

Figure 6-8 Add destinations

9. On the Add Gateway configuration page, insert your destination name, IP address of your SAP system, and SAP gateway port of your SAP system. Select the authentication method that you want to use, and click the **green plus sign (+)** on the right side of the box, as shown in Figure 6-9 on page 139.

   The Add Gateway page allows you to create multiple destinations under the same gateway service. If you have plans to use more than one SAP system or include all SAP ECC landscape (development, quality, and production systems), you can add more destinations by using the plus sign on the left. If you want to create only one destination, you can click **I'M DONE**.

> **Tip:** SAP port convention for SAP Gateway is *33xx*, where xx is the SAP instance number of the system. You can usually find the SAP instance number inside the SAP Logon tool under properties of the target SAP system entry.



Home

Add Gateway

*Required step

✓ *Name It ● Connect It ● Add Destinations

Let's Add some destinations to your gateway...

Create Destinations

Production SAP System | 192.168.1.100 | 3311 | TCP ⊕

▸ Advanced

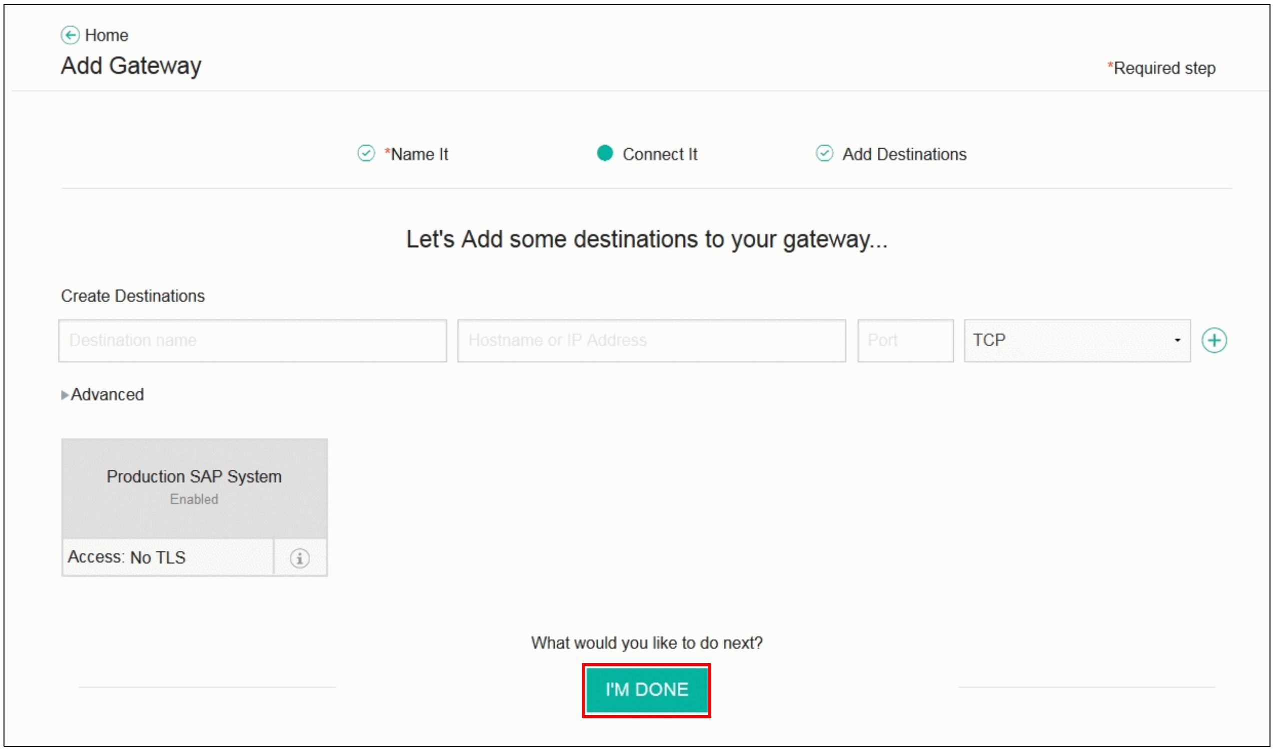
What would you like to do next?

I'M DONE

Figure 6-9 Secure Gateway destination setup

> **Important:** In this scenario, we used the TCP Authentication. For more secure connectivity, you can select the TLS Server side, Mutual Authentication.

10. The last step of the Secure Gateway service setup is to check all the destinations that you have created and confirm by using **I'M DONE**, as shown in Figure 6-10.



Home

Add Gateway

*Required step

✓ *Name It ● Connect It ✓ Add Destinations

Let's Add some destinations to your gateway...

Create Destinations

Destination name | Hostname or IP Address | Port | TCP ⊕

▸ Advanced

Production SAP System

Enabled

Access: No TLS ⓘ

What would you like to do next?

I'M DONE

Figure 6-10 Secure Gateway setup confirmation

11. At the end of the procedure, you are able to see the Secure Gateway dashboard. With this dashboard, you can check the current connections and inbound and outbound traffic flow of the destinations that you created, as shown in Figure 6-11.

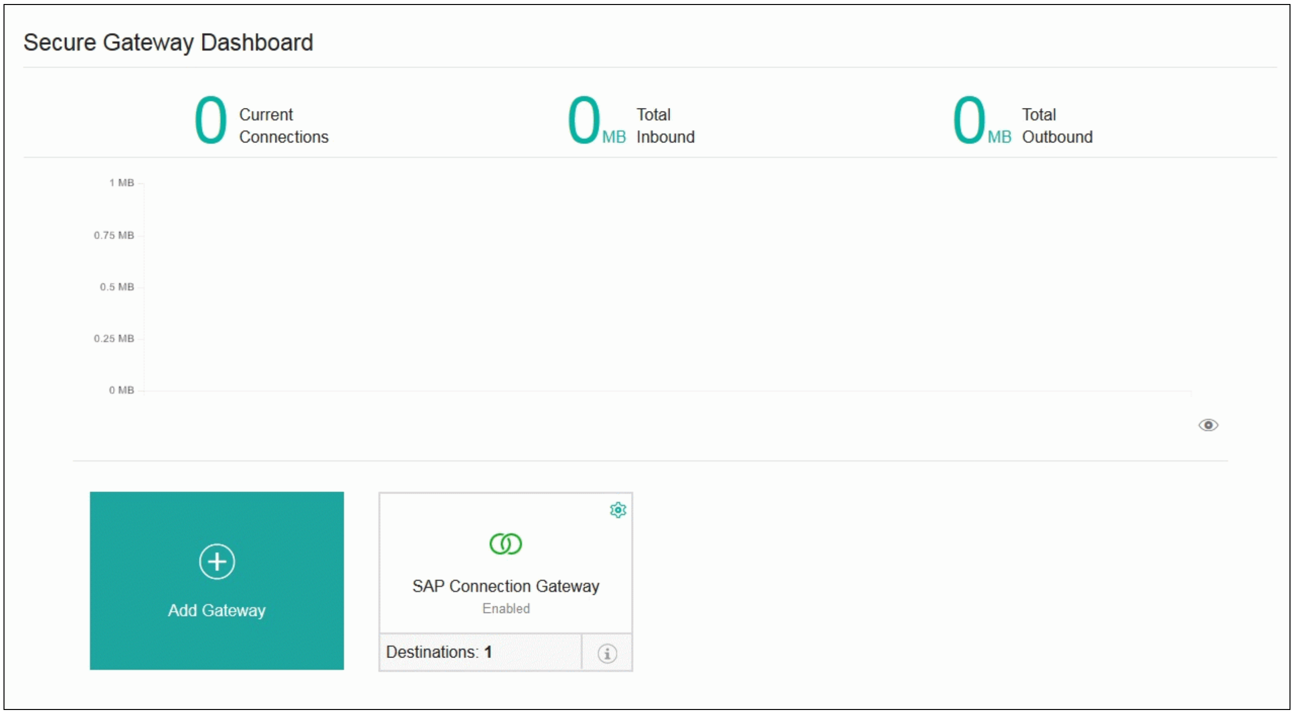


Figure 6-11 Secure Gateway dashboard

> **Important:** The connection status icon must be green to be working. If your connection status icon is red, you must re-execute the `docker` command. To see the exact command to run in order to connect your gateway, click the **red symbol** and then click **Connect Gateway** on the creation destination page**.**

6.2.2 Setting up the Connect & Compose service

These steps show you how to set up the Connect & Compose Bluemix service. To execute these steps, complete your Secure Gateway setup as described in section 6.2.1, “Setting up the Secure Gateway service” on page 135:

1. Log in to Bluemix and click **CATALOG**.
2. Search for Connect & Compose and click the image as shown in Figure 6-12 on page 141.

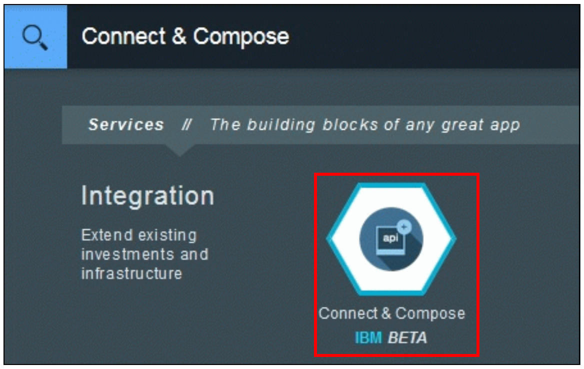
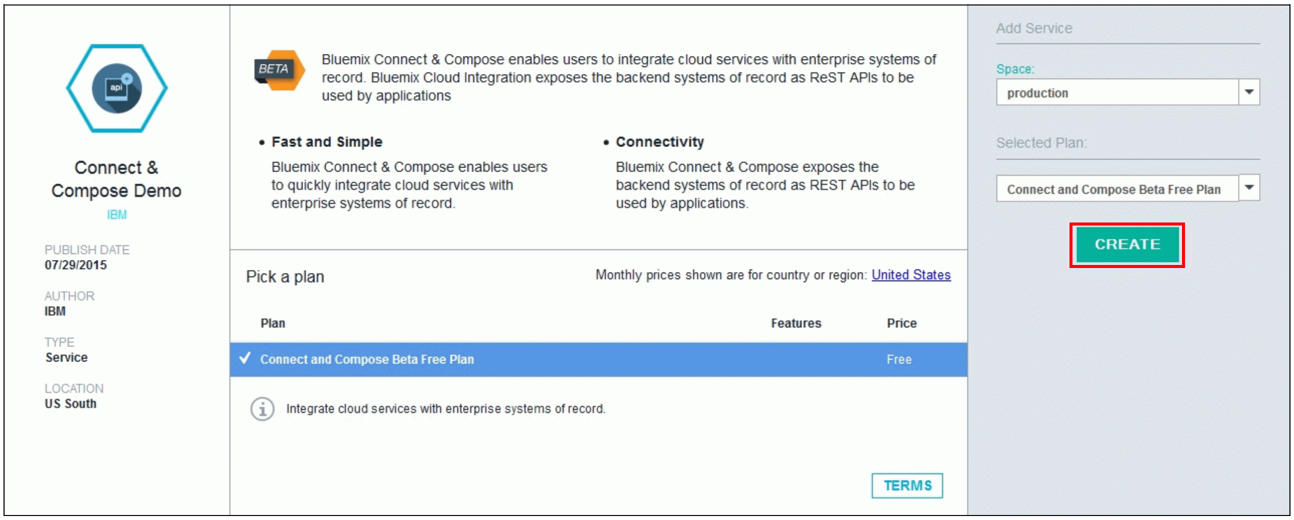

Connect & Compose
Services // The building blocks of any great app
Integration
Extend existing investments and infrastructure
api
Connect & Compose
IBM BETA

Figure 6-12 Connect & Compose service

3. Select the space and plan and click **CREATE**, as shown in Figure 6-13.



Connect & Compose Demo

IBM

PUBLISH DATE
07/29/2015

AUTHOR
IBM

TYPE
Service

LOCATION
US South

Bluemix Connect & Compose enables users to integrate cloud services with enterprise systems of record. Bluemix Cloud Integration exposes the backend systems of record as ReST APIs to be used by applications

- **Fast and Simple**
  Bluemix Connect & Compose enables users to quickly integrate cloud services with enterprise systems of record.
- **Connectivity**
  Bluemix Connect & Compose exposes the backend systems of record as REST APIs to be used by applications.

Pick a plan

Monthly prices shown are for country or region: United States

Plan	Features	Price
✓ Connect and Compose Beta Free Plan		Free

Integrate cloud services with enterprise systems of record.

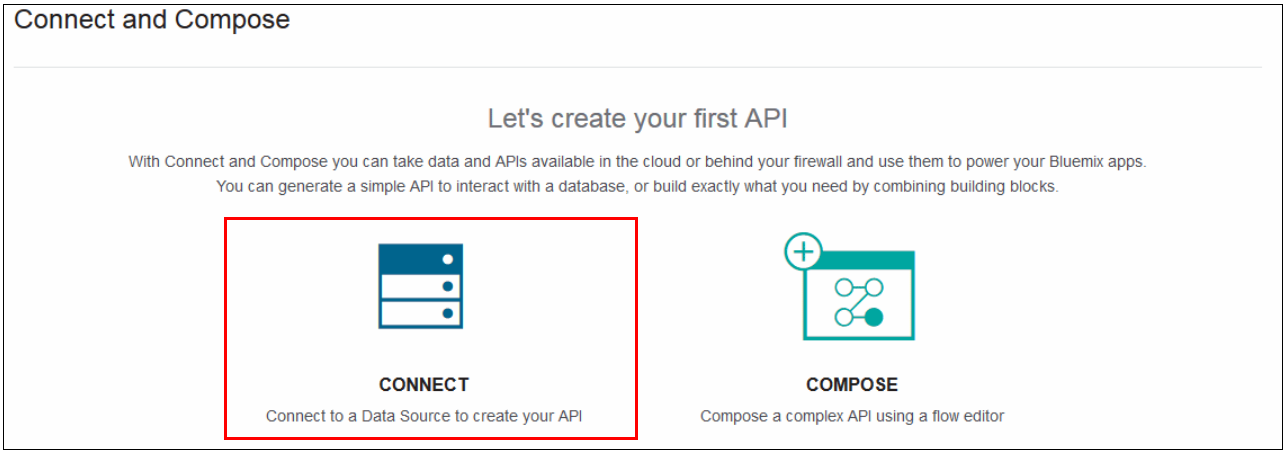
Add Service

Space: production

Selected Plan: Connect and Compose Beta Free Plan

Figure 6-13 Connect & Compose plans

4. On the first Connect & Compose panel, click **CONNECT** to start creating your first API, as shown in Figure 6-14.


Connect and Compose

Let's create your first API

With Connect and Compose you can take data and APIs available in the cloud or behind your firewall and use them to power your Bluemix apps. You can generate a simple API to interact with a database, or build exactly what you need by combining building blocks.

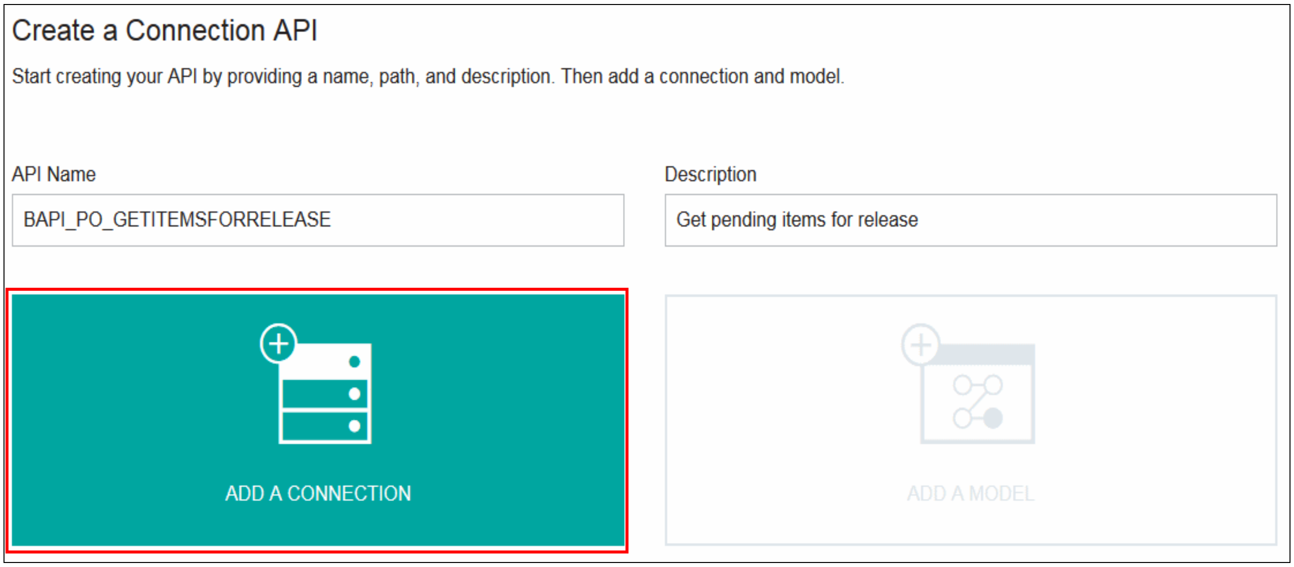
Connect to a Data Source to create your API

Compose a complex API using a flow editor

Figure 6-14 Connect & Compose first panel

5. Type the API name and a short description of the function call and click **ADD A CONNECTION** as shown in Figure 6-15.

> **Tip:** Only the API name is shared inside the Bluemix services. Use meaningful names that represent the objective of the API call.



Create a Connection API

Start creating your API by providing a name, path, and description. Then add a connection and model.

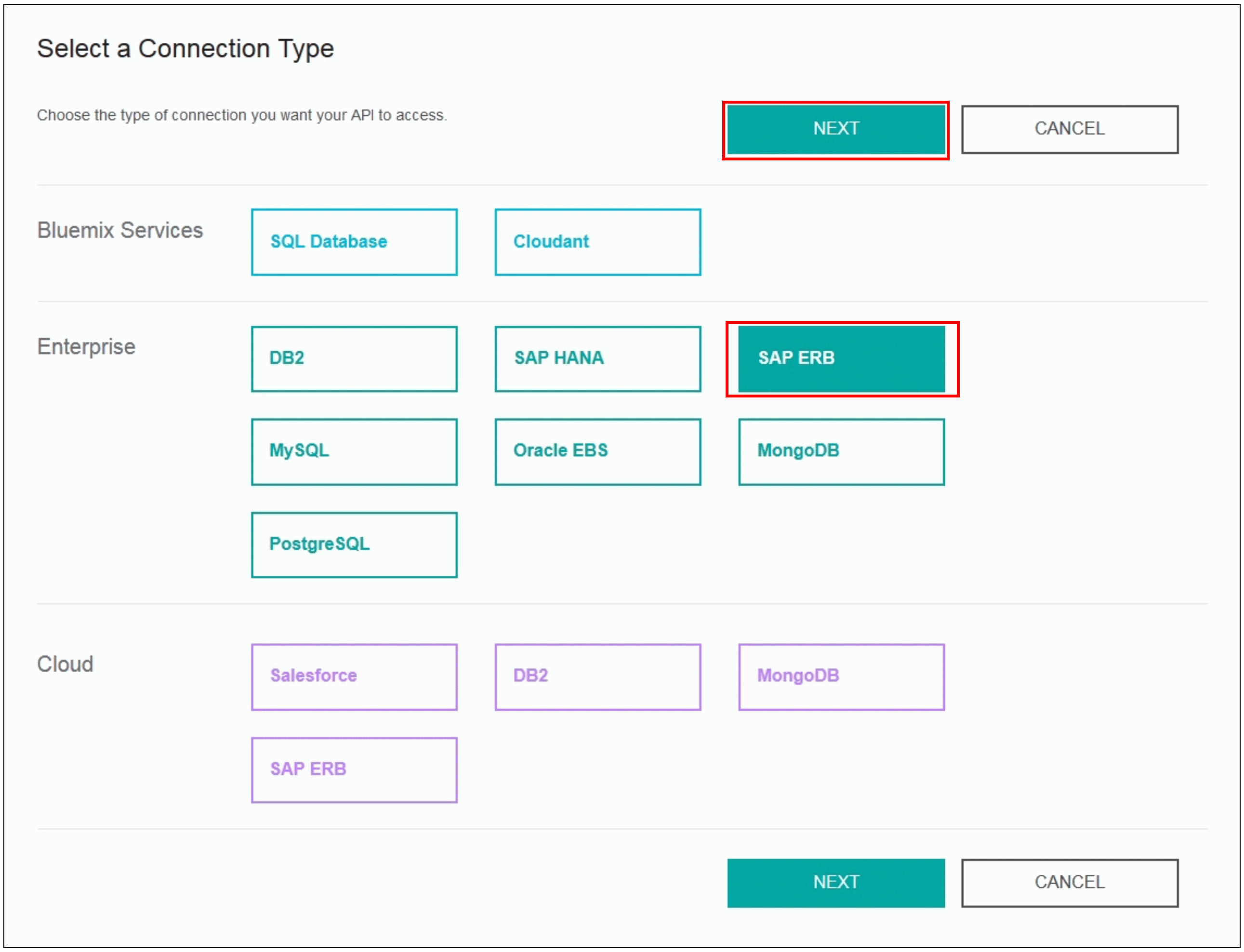
API Name: BAPI_PO_GETITEMSFORRELEASE

Description: Get pending items for release

Figure 6-15 Connect & Compose API name creation

The next page presents all types of connections that are available from the Connect & Compose service. You have two options for SAP systems connectivity: The first is *enterprise* (on-premises) and the second is *cloud* (on the Internet). In our scenario, we use an enterprise SAP system running on a dedicated network.

6. Select **SAP ERB** under the Enterprise section and click **NEXT**, as shown in Figure 6-16 on page 143.



Select a Connection Type

Choose the type of connection you want your API to access.

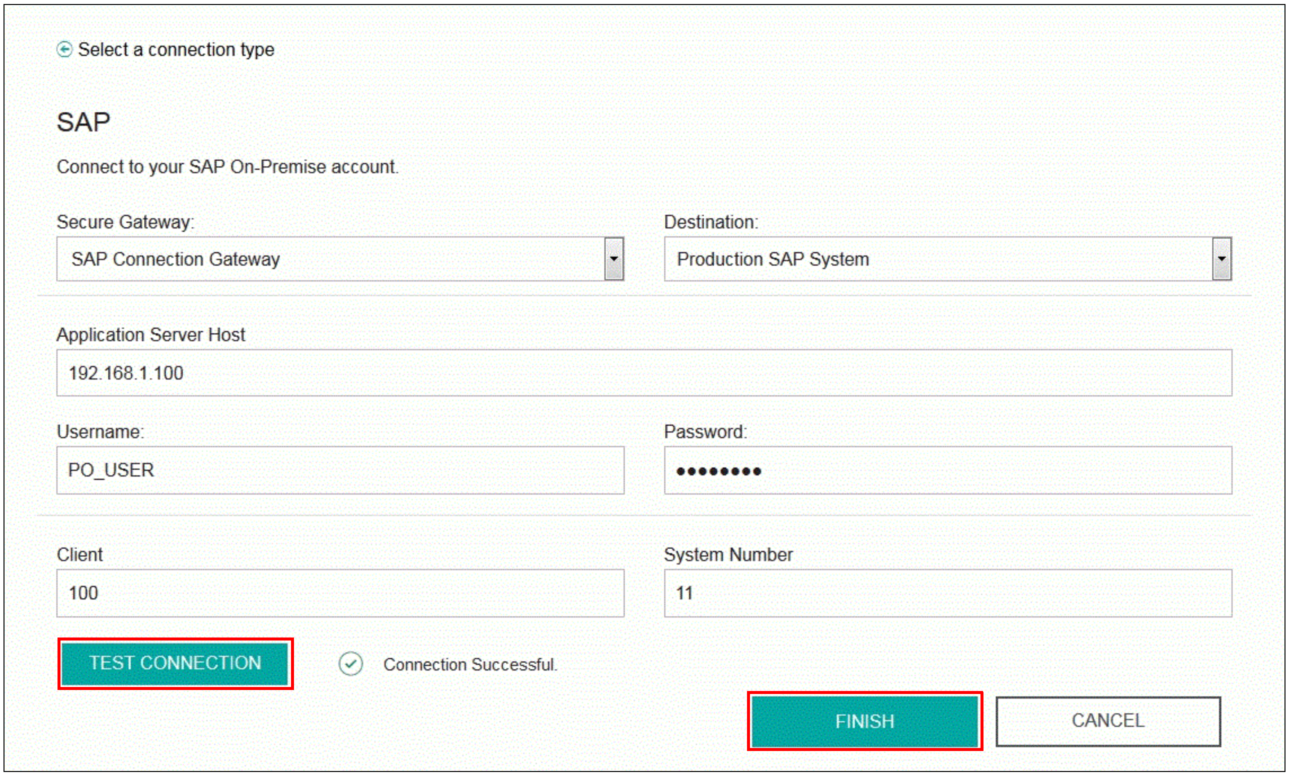
Figure 6-16 Connect & Compose options

On the connection type page, you must select and type all the required information to be able to establish the connection with the SAP system.

7. Select the name of the **Secure gateway** and **Destination** created in the Secure gateway service setup procedure.
8. Then, type the SAP Application Server Host IP Address, SAP user and password, SAP client, and SAP system number.
9. To test the connection between Bluemix and the target SAP system, click **TEST CONNECTION**.

10. If the connection was successfully executed, you see Connection Successful information, as shown in Figure 6-17. Click **FINISH**.

> **Tip:** If you had problems during the connection, try to log in to the SAP system by using the SAP Logon tool, using the same SAP client, user, and password provided here. Also, verify that the IP address and SAP system number of the target system are correct.



Select a connection type

SAP

Connect to your SAP On-Premise account.

Secure Gateway: SAP Connection Gateway

Destination: Production SAP System

Application Server Host: 192.168.1.100

Username: PO_USER

Password: ●●●●●●●●●

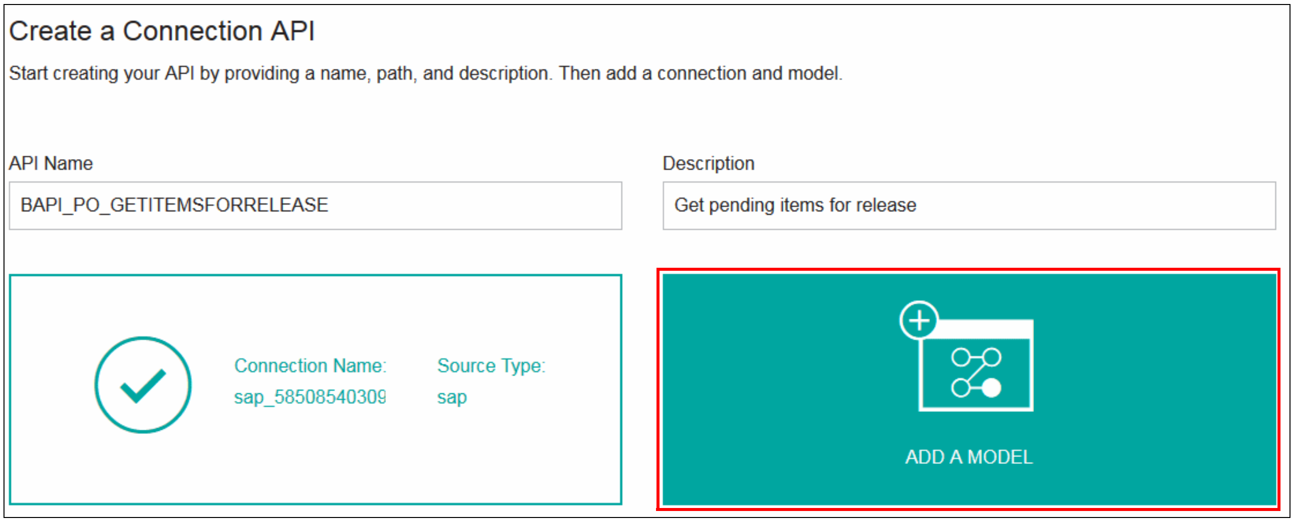
Client: 100

System Number: 11

✓ Connection Successful.

Figure 6-17 Connect & Compose SAP connection type

11. Now that the Connect phase is ready, click **ADD A MODEL** to interact with the SAP system functions as shown in Figure 6-18 on page 145.



Create a Connection API

Start creating your API by providing a name, path, and description. Then add a connection and model.

API Name: BAPI_PO_GETITEMSFORRELEASE

Description: Get pending items for release

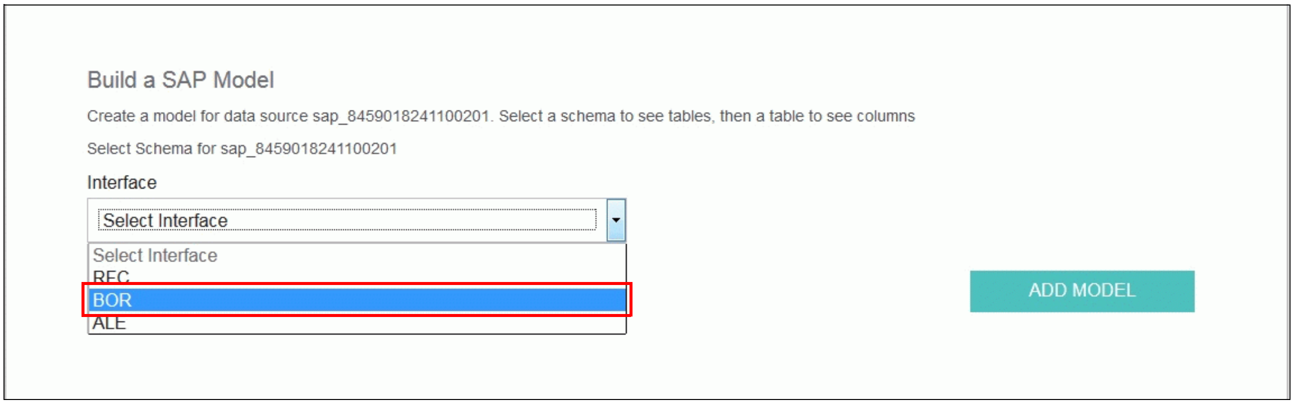
Connection Name: sap_58508540309

Source Type: sap

Figure 6-18 Connect & Compose API panel

The next page shows the three options to build your model with the SAP system. In our scenario, we used Business Object Repository (BOR) to easily bring in all SAP Business Application Programming Interfaces (BAPIs) into the Bluemix environment.

12. Select **BOR** to start building your model, as shown in Figure 6-19.



Build a SAP Model

Create a model for data source sap_8459018241100201. Select a schema to see tables, then a table to see columns

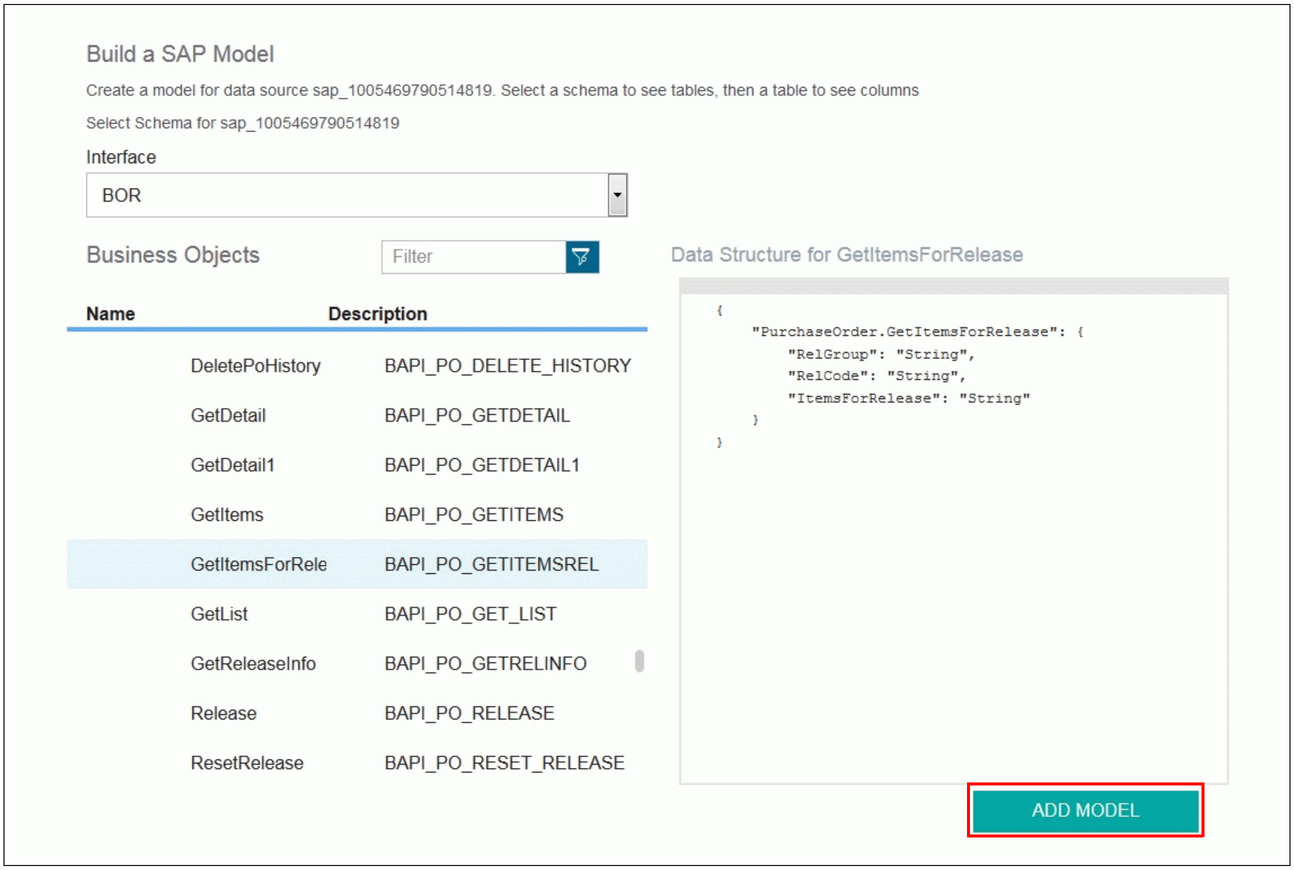
Select Schema for sap_8459018241100201

Interface: Select Interface

Figure 6-19 Connect & Compose possible SAP interfaces

At this moment, the Connect & Compose service connects to the target SAP system and brings all available business objects for selection. Our scenario uses the Purchase Orders business object to call GetItemForRelease and Release BAPI. When you select the wanted BAPI, you are able to see all required parameters to call that function on the right side of the panel.

13. To create the model, click **GetItemsForRelease** and click **ADD MODEL**, as shown in Figure 6-20.



Build a SAP Model

Create a model for data source sap_1005469790514819. Select a schema to see tables, then a table to see columns

Select Schema for sap_1005469790514819

Interface: BOR

Business Objects

Name	Description
DeletePoHistory	BAPI_PO_DELETE_HISTORY
GetDetail	BAPI_PO_GETDETAIL
GetDetail1	BAPI_PO_GETDETAIL1
GetItems	BAPI_PO_GETITEMS
GetItemsForRele	BAPI_PO_GETITEMSREL
GetList	BAPI_PO_GET_LIST
GetReleaseInfo	BAPI_PO_GETRELINFO
Release	BAPI_PO_RELEASE
ResetRelease	BAPI_PO_RESET_RELEASE

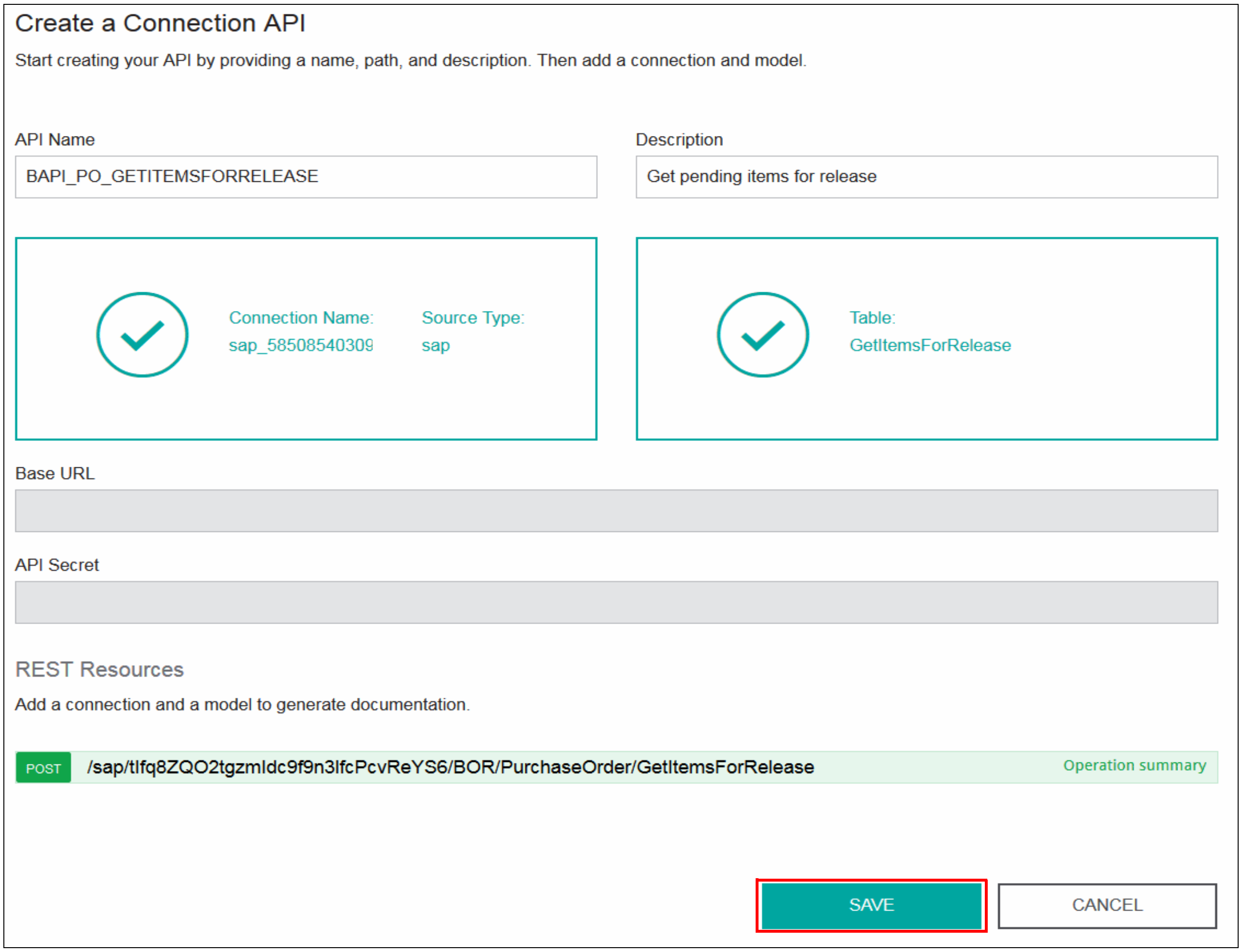
Data Structure for GetItemsForRelease

```
{
    "PurchaseOrder.GetItemsForRelease": {
        "RelGroup": "String",
        "RelCode": "String",
        "ItemsForRelease": "String"
    }
}
```

Figure 6-20 Connect & Compose BAPI details

14. Our scenario uses two APIs, one to get pending approval items (GetItemsForRelease) and the other for releasing the items (Release). To create the second API, execute all the steps from the beginning. When you finish this procedure, select **Release BAPI** instead of GetItemsForRelease in step 13.

15. To create the Connect & Compose service with the selected API, click **SAVE** as shown in Figure 6-21 on page 147.



Create a Connection API

Start creating your API by providing a name, path, and description. Then add a connection and model.

API Name: BAPI_PO_GETITEMSFORRELEASE

Description: Get pending items for release

Connection Name: sap_58508540309

Source Type: sap

Table: GetItemsForRelease

Base URL

API Secret

REST Resources

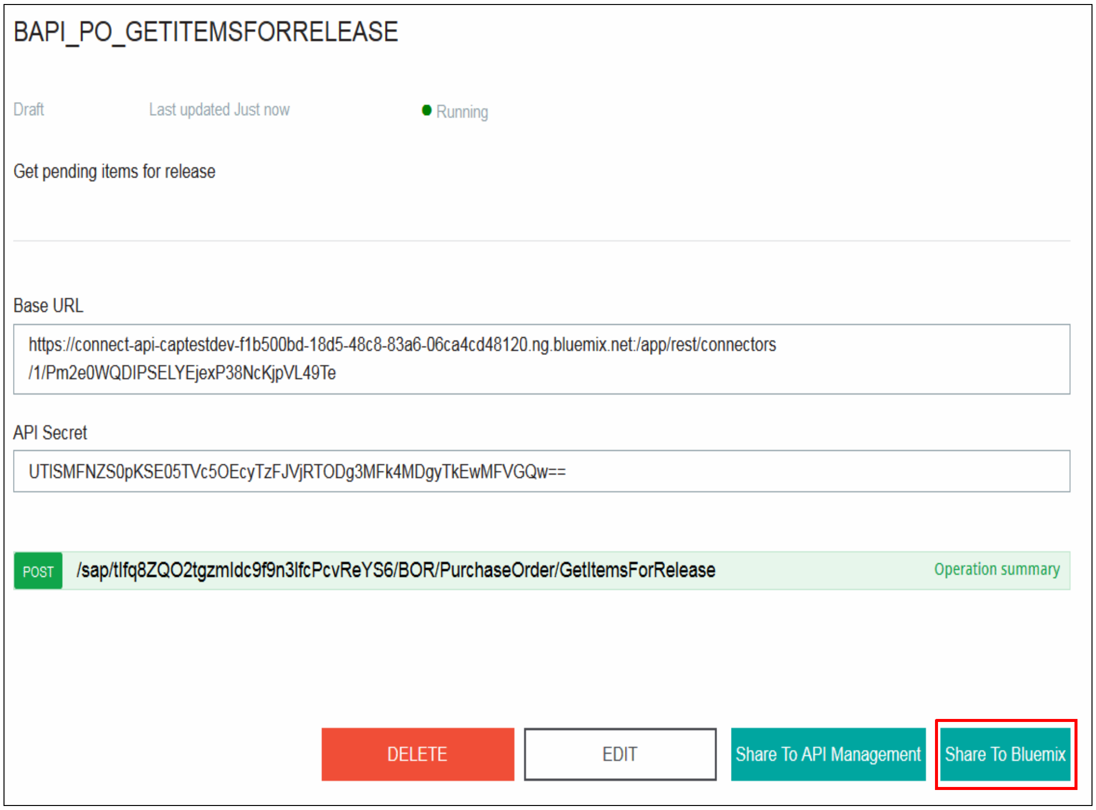
Add a connection and a model to generate documentation.

POST /sap/tlfq8ZQO2tgzmldc9f9n3lfcPcvReYS6/BOR/PurchaseOrder/GetItemsForRelease

Figure 6-21 Connect & Compose save panel

Now that you have saved your API, you can check the base API URL and the API secret used to call the API. To be able to bind the APIs with your application, you need to share your API with the Bluemix environment. You can also share your API with API Management if you want to manage all APIs from one single interface and create integrated plans with them.

16. Click **Share To Bluemix** as shown in Figure 6-22.



BAPI_PO_GETITEMSFORRELEASE

Draft Last updated Just now ● Running

Get pending items for release

Base URL

https://connect-api-captestdev-f1b500bd-18d5-48c8-83a6-06ca4cd48120.ng.bluemix.net:/app/rest/connectors
/1/Pm2e0WQDIPSELYEjexP38NcKjpVL49Te

API Secret

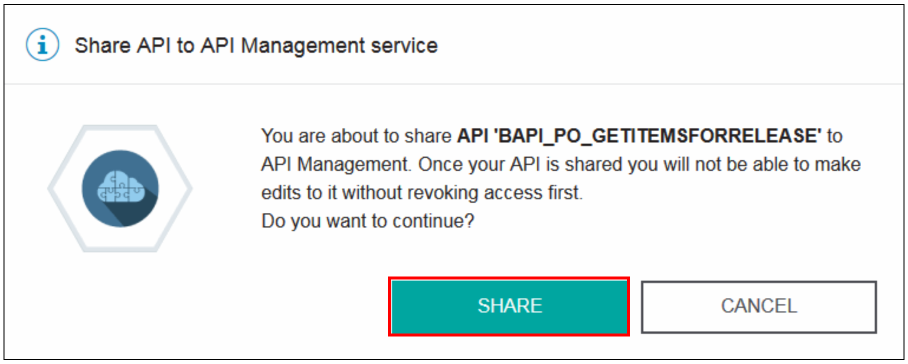
UTISMFNZS0pKSE05TVc5OEcyTzFJVjRTODg3MFk4MDgyTkEwMFVGQw==

POST /sap/tlfq8ZQO2tgzmIdc9f9n3lfcPcvReYS6/BOR/PurchaseOrder/GetItemsForRelease Operation summary

Figure 6-22 Connect & Compose share API Management panel

When you share your API with Bluemix, an extra service is created on your dashboard with your custom API.

17. To allow Bluemix to create the custom API, click **SHARE** as shown in Figure 6-23.

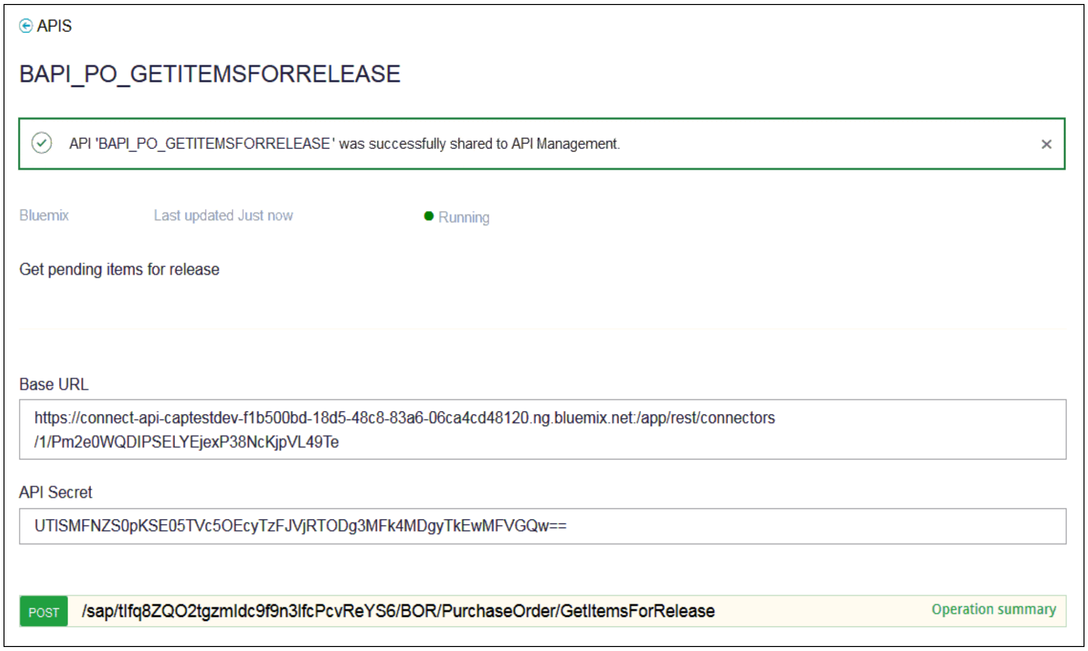


Share API to API Management service

You are about to share **API 'BAPI_PO_GETITEMSFORRELEASE'** to API Management. Once your API is shared you will not be able to make edits to it without revoking access first.
Do you want to continue?

Figure 6-23 Connect & Compose share API Management confirmation panel

After sharing the API with your Bluemix space as shown Figure 6-24 on page 149, you can start testing your API as described in the section 6.3, “Testing the API” on page 149.



APIS

BAPI_PO_GETITEMSFORRELEASE

API 'BAPI_PO_GETITEMSFORRELEASE' was successfully shared to API Management.

Bluemix

Last updated Just now

● Running

Get pending items for release

Base URL

https://connect-api-captestdev-f1b500bd-18d5-48c8-83a6-06ca4cd48120.ng.bluemix.net:/app/rest/connectors
/1/Pm2e0WQDIPSELYEjexP38NcKjpVL49Te

API Secret

UTISMFNZS0pKSE05TVc5OEcyTzFJVjRTODg3MFk4MDgyTkEwMFVGQw==

POST /sap/tlfq8ZQO2tgzmldc9f9n3lfcPcvReYS6/BOR/PurchaseOrder/GetItemsForRelease

Operation summary

Figure 6-24 Connect & Compose final panel

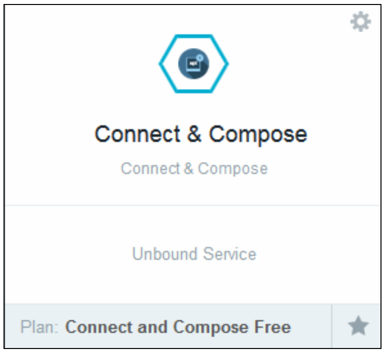
6.3 Testing the API

You can test your APIs directly from the Bluemix Swagger tool or by using your own REST client. In this section, we test the API calls by using the native Bluemix Swagger tool. To test each API, you must pass the correct data structure and the data expected by SAP.

Each SAP BAPI has its own required structure. You can check the structure of each BAPI by using the Connect & Compose service, as shown in the step 13 (Figure 6-20 on page 146) or inside the SAP transaction BAPI.

To test your API, follow these steps:

1. Log in to Bluemix and click **DASHBOARD**.
2. Click **Connect & Compose** service as shown in Figure 6-25.



Connect & Compose

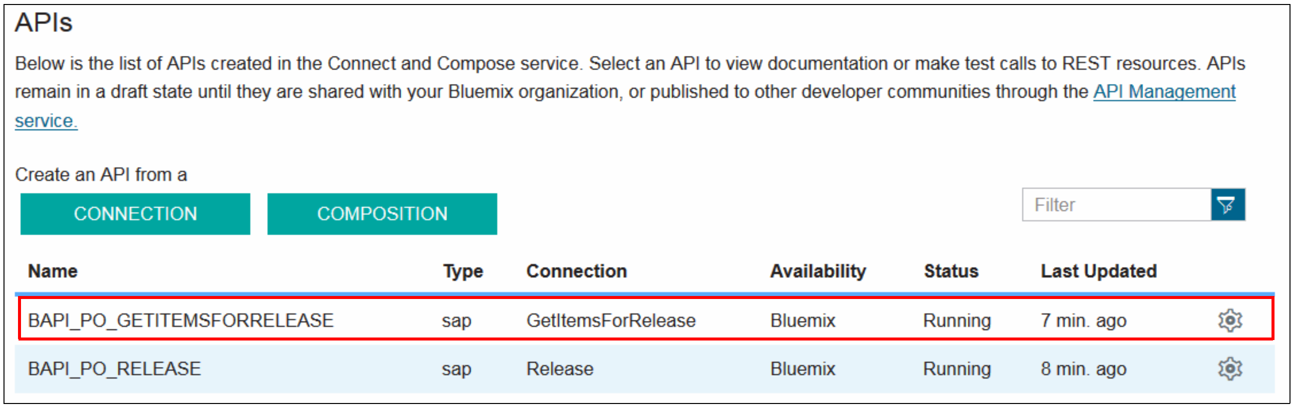
Connect & Compose

Unbound Service

Plan: Connect and Compose Free

Figure 6-25 Connect & Compose service

3. Select the API **BAPI_PO_GETITEMSFORRELEASE** as shown Figure 6-26.



APIs

Below is the list of APIs created in the Connect and Compose service. Select an API to view documentation or make test calls to REST resources. APIs remain in a draft state until they are shared with your Bluemix organization, or published to other developer communities through the API Management service.

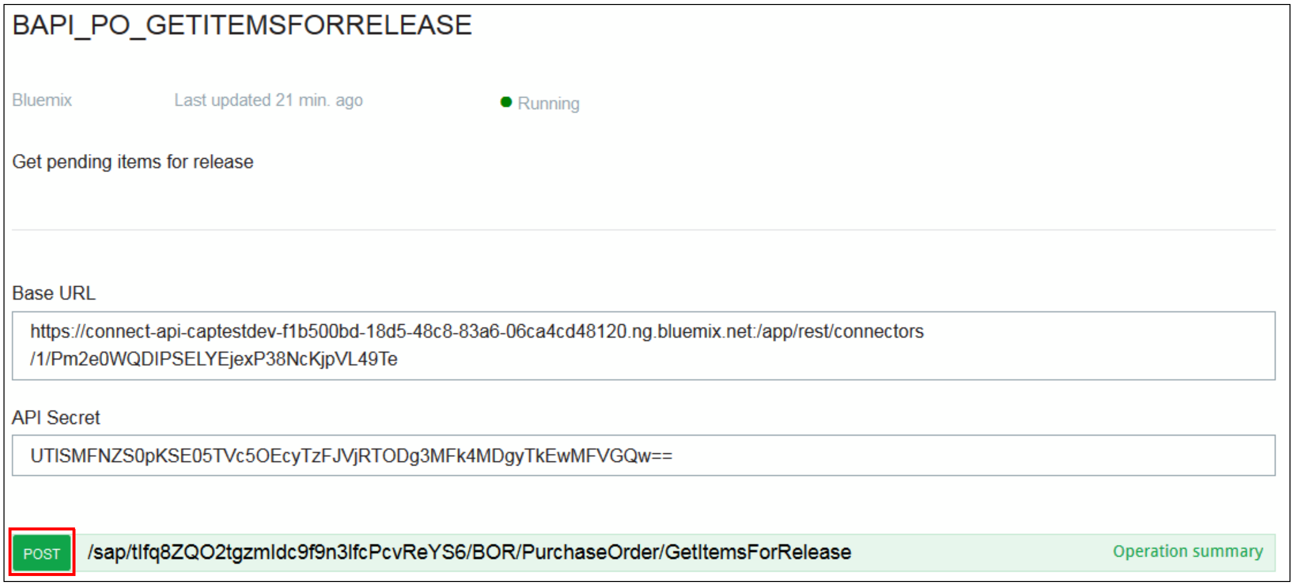
Create an API from a

Name	Type	Connection	Availability	Status	Last Updated
BAPI_PO_GETITEMSFORRELEASE	sap	GetItemsForRelease	Bluemix	Running	7 min. ago
BAPI_PO_RELEASE	sap	Release	Bluemix	Running	8 min. ago

Figure 6-26 Connect & Compose API list

On the next page, you are able to check the status of your API, the base URL, API Key, and post option.

4. To enter the required parameters for testing your API, click **POST**.



BAPI_PO_GETITEMSFORRELEASE

Bluemix Last updated 21 min. ago ● Running

Get pending items for release

Base URL

https://connect-api-captestdev-f1b500bd-18d5-48c8-83a6-06ca4cd48120.ng.bluemix.net:/app/rest/connectors/1/Pm2e0WQDIPSELYEjexP38NcKjpVL49Te

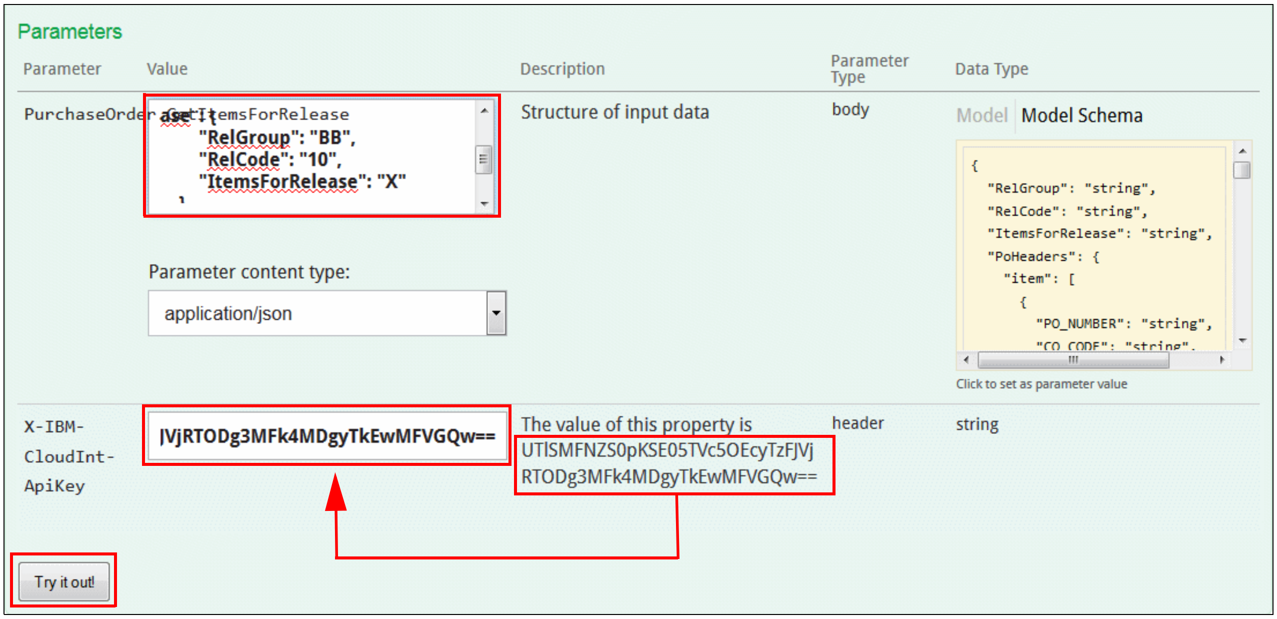
API Secret

UTISMFNZS0pKSE05TVc5OEcyTzFJVjRTODg3MFk4MDgyTkEwMFVGQw==

POST /sap/tlfq8ZQO2tgzmldc9f9n3lfcPcvReYS6/BOR/PurchaseOrder/GetItemsForRelease

Figure 6-27 Connect & Compose API detail

5. Insert the required structure and data inside the Parameter Value field on the upper left side of the page.
6. Copy the API Key from the **Description** field to **Value** field, as shown Figure 6-28.
7. Check that the content type parameter is: **application/json**.
8. Click **Try it Out!** to test the API.



Parameters

Parameter	Value	Description	Parameter Type	Data Type

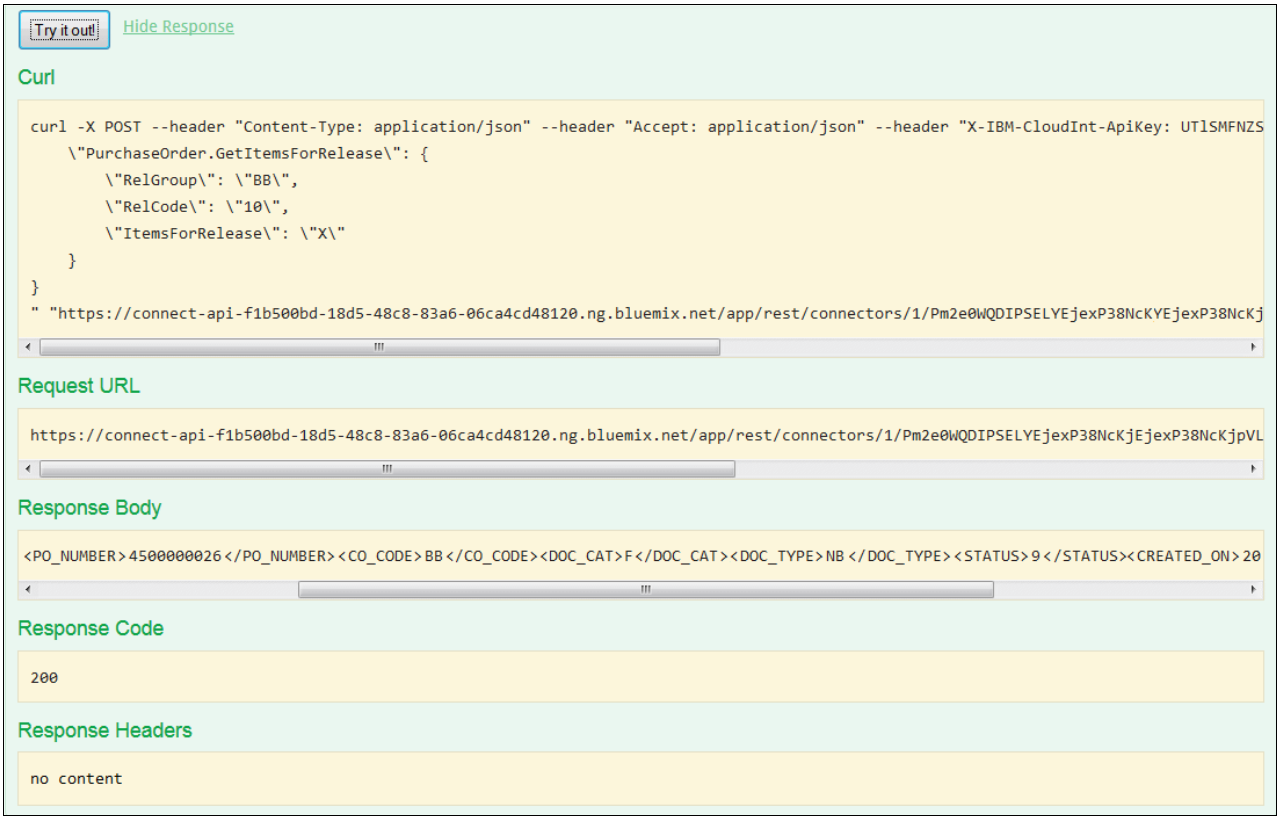
PurchaseOrderGetItemsForRelease	ase": {
	"RelGroup": "BB",
	"RelCode": "10",
	"ItemsForRelease": "X"
	}
	Parameter content type:
	application/json
		Structure of input data	body	Model Model Schema
				{
				"RelGroup": "string",
				"RelCode": "string",
				"ItemsForRelease": "string",
				"PoHeaders": {
				"item": [
				{
				"PO_NUMBER": "string",
				"CO_CODE": "string",
				Click to set as parameter value

X-IBM-CloudInt-ApiKey	JVjRTODg3MFk4MDgyTkEwMFVGQw==	The value of this property is UTlSMFNZS0pKSE05TVc5OEcyTzFJVjRTODg3MFk4MDgyTkEwMFVGQw==	header	string

Figure 6-28 Bluemix Swagger tool

> **Tip:** The data inside the structure must be correct to be able to get the response from SAP. You can check the existing pending approval order numbers, company codes, and release codes inside the SAP transaction ME23N or create a new order inside transaction ME21N.

9. The response code of the successful API test is `200 (OK)`. Inside the response body of the Swagger tool, you find the JSON data with all the requested data as shown in Figure 6-29.



Curl

```
curl -X POST --header "Content-Type: application/json" --header "Accept: application/json" --header "X-IBM-CloudInt-ApiKey: UTlSMFNZS
    \"PurchaseOrder.GetItemsForRelease\": {
        \"RelGroup\": \"BB\",
        \"RelCode\": \"10\",
        \"ItemsForRelease\": \"X\"
    }
}
" "https://connect-api-f1b500bd-18d5-48c8-83a6-06ca4cd48120.ng.bluemix.net/app/rest/connectors/1/Pm2e0WQDIPSELYEjexP38NcKYEjexP38NcKj
```

Request URL

```
https://connect-api-f1b500bd-18d5-48c8-83a6-06ca4cd48120.ng.bluemix.net/app/rest/connectors/1/Pm2e0WQDIPSELYEjexP38NcKjEjexP38NcKjpVL
```

Response Body

```
<PO_NUMBER>4500000026</PO_NUMBER><CO_CODE>BB</CO_CODE><DOC_CAT>F</DOC_CAT><DOC_TYPE>NB</DOC_TYPE><STATUS>9</STATUS><CREATED_ON>20
```

Response Code

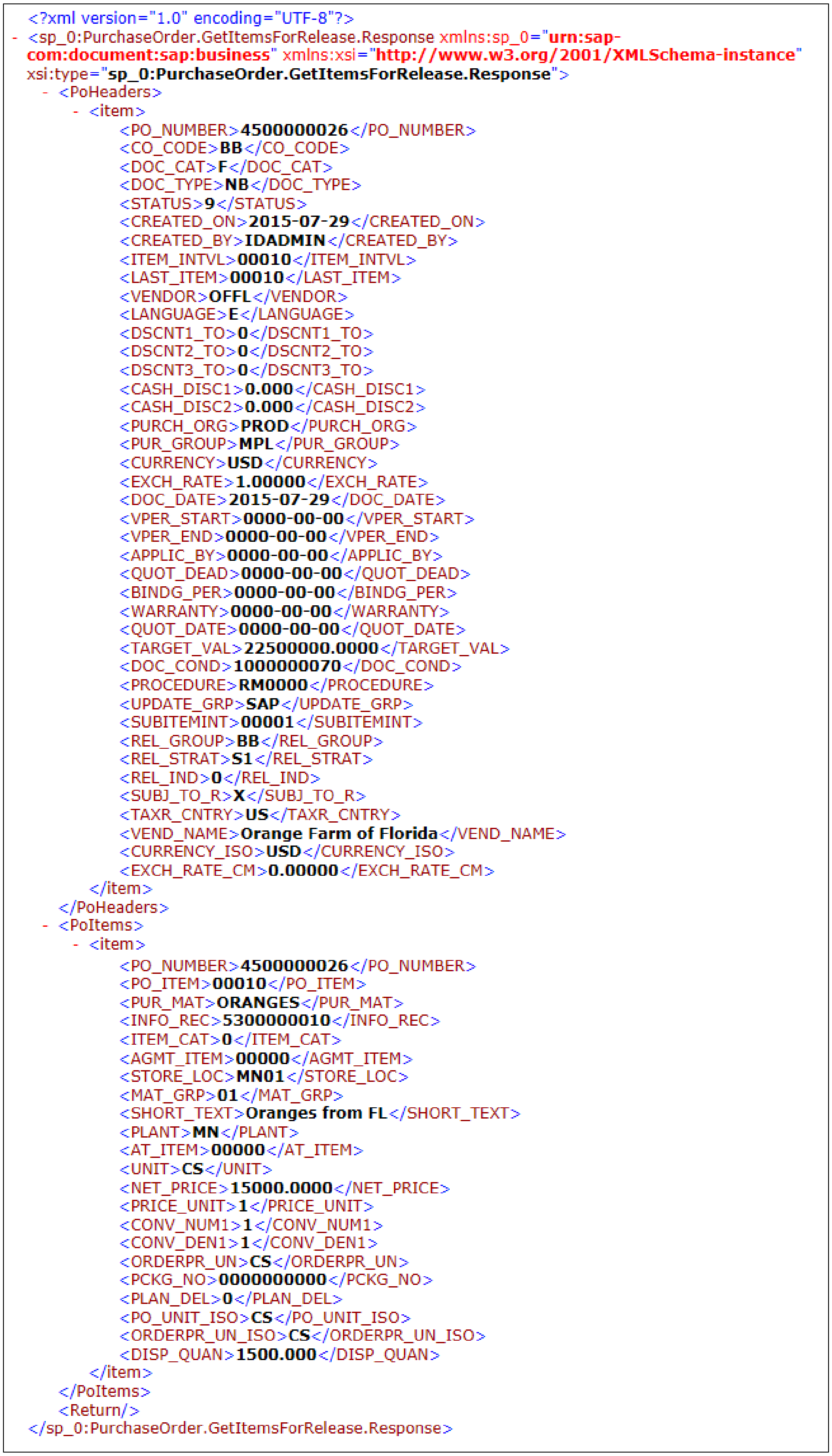
```
200
```

Response Headers

```
no content
```

Figure 6-29 API response

10. For a better view of the results, you can copy the data onto the Notepad application and save as an `.xml` file. You can open this file with any browser. By using this procedure, you have a better understanding of the fields and results as shown in Figure 6-30 on page 153.



```
  <?xml version="1.0" encoding="UTF-8"?>
- <sp_0:PurchaseOrder.GetItemsForRelease.Response xmlns:sp_0="urn:sap-
  com:document:sap:business" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="sp_0:PurchaseOrder.GetItemsForRelease.Response">
  - <PoHeaders>
    - <item>
        <PO_NUMBER>4500000026</PO_NUMBER>
        <CO_CODE>BB</CO_CODE>
        <DOC_CAT>F</DOC_CAT>
        <DOC_TYPE>NB</DOC_TYPE>
        <STATUS>9</STATUS>
        <CREATED_ON>2015-07-29</CREATED_ON>
        <CREATED_BY>IDADMIN</CREATED_BY>
        <ITEM_INTVL>00010</ITEM_INTVL>
        <LAST_ITEM>00010</LAST_ITEM>
        <VENDOR>OFFL</VENDOR>
        <LANGUAGE>E</LANGUAGE>
        <DSCNT1_TO>0</DSCNT1_TO>
        <DSCNT2_TO>0</DSCNT2_TO>
        <DSCNT3_TO>0</DSCNT3_TO>
        <CASH_DISC1>0.000</CASH_DISC1>
        <CASH_DISC2>0.000</CASH_DISC2>
        <PURCH_ORG>PROD</PURCH_ORG>
        <PUR_GROUP>MPL</PUR_GROUP>
        <CURRENCY>USD</CURRENCY>
        <EXCH_RATE>1.00000</EXCH_RATE>
        <DOC_DATE>2015-07-29</DOC_DATE>
        <VPER_START>0000-00-00</VPER_START>
        <VPER_END>0000-00-00</VPER_END>
        <APPLIC_BY>0000-00-00</APPLIC_BY>
        <QUOT_DEAD>0000-00-00</QUOT_DEAD>
        <BINDG_PER>0000-00-00</BINDG_PER>
        <WARRANTY>0000-00-00</WARRANTY>
        <QUOT_DATE>0000-00-00</QUOT_DATE>
        <TARGET_VAL>22500000.0000</TARGET_VAL>
        <DOC_COND>1000000070</DOC_COND>
        <PROCEDURE>RM0000</PROCEDURE>
        <UPDATE_GRP>SAP</UPDATE_GRP>
        <SUBITEMINT>00001</SUBITEMINT>
        <REL_GROUP>BB</REL_GROUP>
        <REL_STRAT>S1</REL_STRAT>
        <REL_IND>0</REL_IND>
        <SUBJ_TO_R>X</SUBJ_TO_R>
        <TAXR_CNTRY>US</TAXR_CNTRY>
        <VEND_NAME>Orange Farm of Florida</VEND_NAME>
        <CURRENCY_ISO>USD</CURRENCY_ISO>
        <EXCH_RATE_CM>0.00000</EXCH_RATE_CM>
      </item>
    </PoHeaders>
  - <PoItems>
    - <item>
        <PO_NUMBER>4500000026</PO_NUMBER>
        <PO_ITEM>00010</PO_ITEM>
        <PUR_MAT>ORANGES</PUR_MAT>
        <INFO_REC>5300000010</INFO_REC>
        <ITEM_CAT>0</ITEM_CAT>
        <AGMT_ITEM>00000</AGMT_ITEM>
        <STORE_LOC>MN01</STORE_LOC>
        <MAT_GRP>01</MAT_GRP>
        <SHORT_TEXT>Oranges from FL</SHORT_TEXT>
        <PLANT>MN</PLANT>
        <AT_ITEM>00000</AT_ITEM>
        <UNIT>CS</UNIT>
        <NET_PRICE>15000.0000</NET_PRICE>
        <PRICE_UNIT>1</PRICE_UNIT>
        <CONV_NUM1>1</CONV_NUM1>
        <CONV_DEN1>1</CONV_DEN1>
        <ORDERPR_UN>CS</ORDERPR_UN>
        <PCKG_NO>0000000000</PCKG_NO>
        <PLAN_DEL>0</PLAN_DEL>
        <PO_UNIT_ISO>CS</PO_UNIT_ISO>
        <ORDERPR_UN_ISO>CS</ORDERPR_UN_ISO>
        <DISP_QUAN>1500.000</DISP_QUAN>
      </item>
    </PoItems>
    <Return/>
  </sp_0:PurchaseOrder.GetItemsForRelease.Response>
```

Figure 6-30 XML response

6.3.1 Bind the API to your Bluemix application

To use the newly created API inside your Bluemix application, you must bind them together. To execute this step, you must have your application up and running inside Bluemix.

To bind your API, follow these steps:

1. Log in to Bluemix and click **DASHBOARD.**
2. Open the target Bluemix application as shown Figure 6-31.

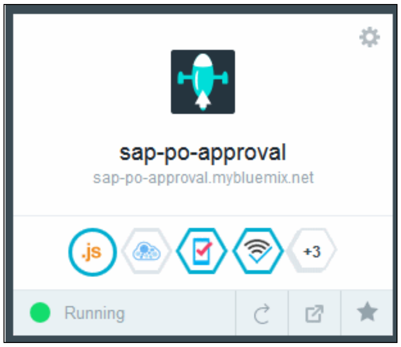


Figure 6-31 Bluemix application

3. Inside the application, click **BIND A SERVICE OR API** as shown in Figure 6-32.

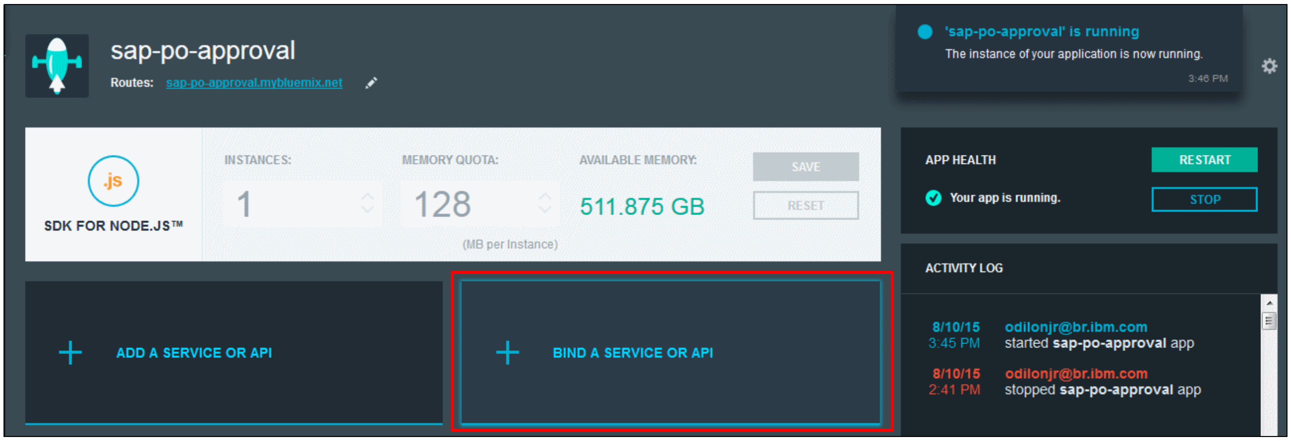
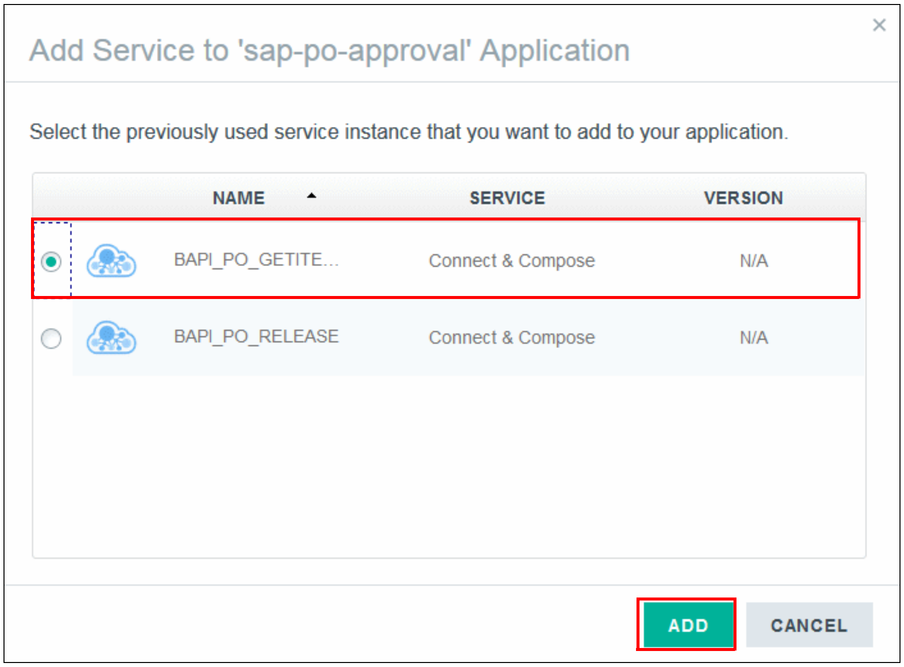


Figure 6-32 Bluemix application detail

4. Select the API **BAPI_PO_GETITEMSFORRELEASE** to bind, and click **ADD** as shown in Figure 6-33.



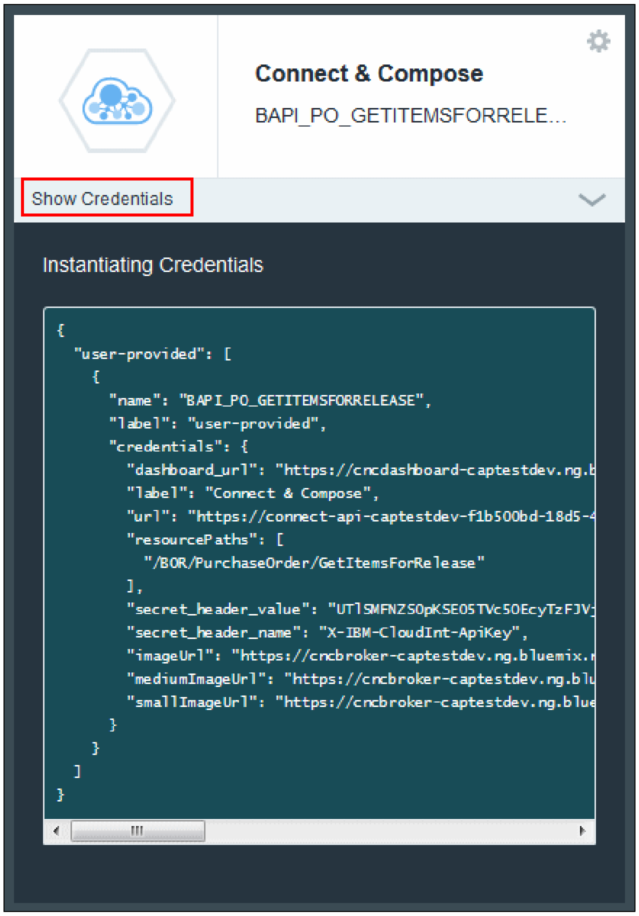
Add Service to 'sap-po-approval' Application

Select the previously used service instance that you want to add to your application.

	NAME	SERVICE	VERSION
●	BAPI_PO_GETITE...	Connect & Compose	N/A
○	BAPI_PO_RELEASE	Connect & Compose	N/A

Figure 6-33   API selection

5. To check the credentials of this API, click **Show Credentials** under your application API, as shown in Figure 6-34.



```
{
  "user-provided": [
    {
      "name": "BAPI_PO_GETITEMSFORRELEASE",
      "label": "user-provided",
      "credentials": {
        "dashboard_url": "https://cncdashboard-captestdev.ng.b
        "label": "Connect & Compose",
        "url": "https://connect-api-captestdev-f1b500bd-18d5-4
        "resourcePaths": [
          "/BOR/PurchaseOrder/GetItemsForRelease"
        ],
        "secret_header_value": "UTlSMFNZS0pKSE05TVc5OEcyTzFJVj
        "secret_header_name": "X-IBM-CloudInt-ApiKey",
        "imageUrl": "https://cncbroker-captestdev.ng.bluemix.r
        "mediumImageUrl": "https://cncbroker-captestdev.ng.blu
        "smallImageUrl": "https://cncbroker-captestdev.ng.blue
      }
    }
  ]
}
```

Figure 6-34 Credential detail of the API

This concludes testing of your SAP APIs from the Bluemix Swagger tool.

7

Exposing CICS transactions with z/OS Connect

This chapter describes how to build a cloud-native application in IBM Bluemix and connect it to an IBM CICS Transaction Server running in your data center.

At the end of the chapter, you will know how to set up IBM z/OS Connect for CICS Transaction Server, create Representational State Transfer (REST) application programming interfaces (APIs) for your existing CICS transactions, and connect your application in IBM Bluemix to your CICS Transaction Server.

This chapter has the following sections:

© Copyright IBM Corp. 2015. All rights reserved.

7.1 Solution overview

CompanyB is looking for ways to drive more sales at the store and increase their customer satisfaction. The line of business is proposing a “virtual shopping list,” which will remind customers that they need to purchase something.

For example, a customer is probably using pencil and paper to build their weekly shopping list. They think they have remembered everything, but after the shopping trip, the customer realizes they forgot milk. Now they run to the convenience store to buy the milk, rather than spending that money at CompanyB’s store.

With this app, customers will be reminded that they need milk at the store the next time they go based on the purchasing history. The new application sees that they have been buying milk every two weeks and it has been about two weeks since they last bought milk. The customer will now spend the money at CompanyB’s store and be happy he did not have to make an extra trip to get milk.

The line of business is working with IT to determine how to take advantage of the vast amounts of data stored on their systems. The data is stored on their z Systems platform. They trust their business transactions to IBM CICS Transaction Server for z/OS, the premier enterprise-grade, mixed-language application server for mission critical applications.

The line of business wants to see this application delivered as soon as possible. To accomplish this, they want to do their development as well as host the initial version in a cloud environment.

They have chosen Bluemix as the platform to build their application for their easy to use platform with integration technologies for connecting to on-premises systems of record.

Here is a proposed architecture for the solution in Figure 7-1.

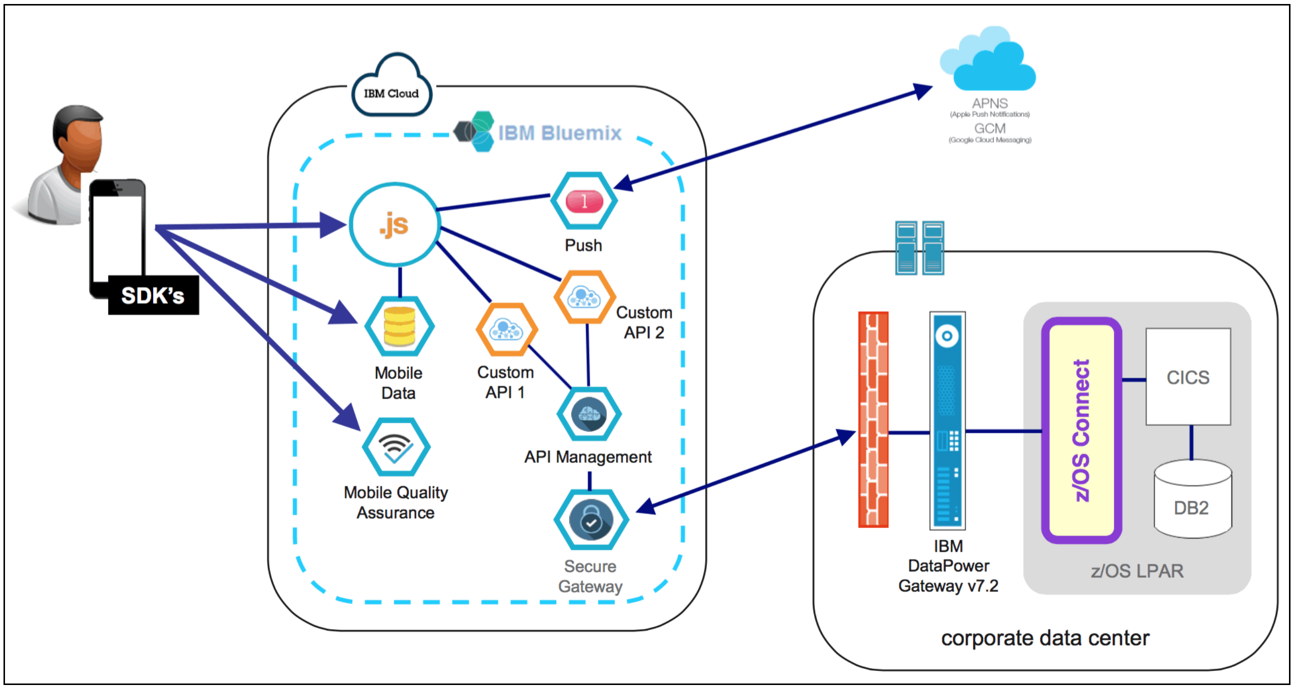


Figure 7-1 Proposed architecture of the virtual shopping list solution

CompanyB wants to reuse their proven programs running in CICS to get the data needed to provide recommendations to customers. z/OS Connect uses WebSphere Liberty Profile for z/OS to host the web application, which translates the REST APIs to a CICS transaction or program.

To connect their on-premises system of record, they use the Secure Gateway service in Bluemix, which connects via a DataPower gateway appliance.

Business case for this scenario: For more information about the business case for this scenario and the company profile, refer to 1.3.10, “Solutions and actions by the CompanyB CIO” on page 13.

This chapter describes how to configure z/OS Connect, set up the Secure Gateway service, configure DataPower, and build an application to get the data to come up with recommendations for their customers.

7.2 Step-by-step implementation

To implement this scenario, we first configure z/OS Connect for CICS, then connect IBM Bluemix to CICS using z/OS Connect, and finally use IBM API Management for the new APIs. In the following section, we describe these steps.

7.3 Configuring z/OS Connect for CICS

z/OS Connect for CICS is distributed as part of CICS Transaction Server V5.2 and later. You can configure z/OS Connect for CICS manually, and it is typically performed one time, with further configuration for individual services.

Before using this guide to configure z/OS Connect for CICS, you should be familiar with how to administer a CICS Transaction Server.

Enabling z/OS Connect should not affect the normal operation of the CICS Transaction Server.

To configure z/OS Connect, perform the following steps:

1. Create the Java virtual machine (JVM) server to host WebSphere Liberty Profile.
2. Configure the JVM server.
3. Create the pipeline for the existing CICS transactions to z/OS Connect.
4. Configure the WebSphere Liberty Profile.
5. Install the JVM server and pipeline.
6. Validate the configuration.

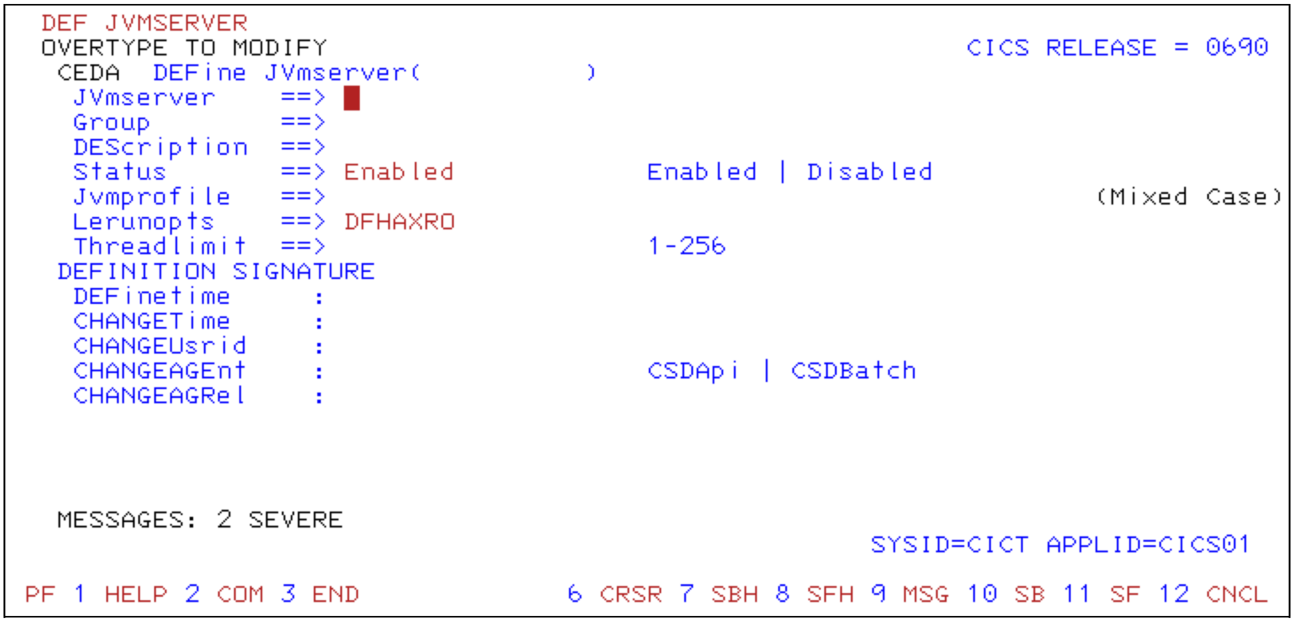
7.3.1 Creating the JVM server for z/OS Connect

It is important to consider whether you already have a WebSphere Liberty Profile JVM server that is configured in CICS. Although it is possible to host z/OS Connect and other unrelated services in the same WebSphere Liberty Profile environment, it is a good practice to configure a separate JVM server for the sole use of z/OS Connect. It is also a good practice to only have a single WebSphere Liberty Profile JVM server that is configured in any single CICS region. You can host z/OS Connect in its own CICS region, or group of CICS regions, and use the Distributed Program Link mechanism to call CICS programs in the application-owning CICS regions.

Enter the CICS terminal and use the CEDA transaction to define a JVM server:

```
CEDA DEF JVMSERVER
```

You are presented with the panel shown in Figure 7-2. The descriptions for the attributes of a JVM server are described in Table 7-1.



```
 DEF JVMSERVER
 OVERTYPE TO MODIFY                                                CICS RELEASE = 0690
  CEDA  DEFine JVmserver(          )
   JVmserver     ==>
   Group         ==>
   DEScription   ==>
   Status        ==> Enabled           Enabled | Disabled
   Jvmprofile    ==>                                                         (Mixed Case)
   Lerunopts     ==> DFHAXRO
   Threadlimit   ==>                   1-256
  DEFINITION SIGNATURE
   DEFinetime      :
   CHANGETime      :
   CHANGEUsrid     :
   CHANGEAGEnt     :                   CSDApi | CSDBatch
   CHANGEAGRel     :



  MESSAGES: 2 SEVERE
                                                                SYSID=CICT APPLID=CICS01

PF 1 HELP 2 COM 3 END           6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

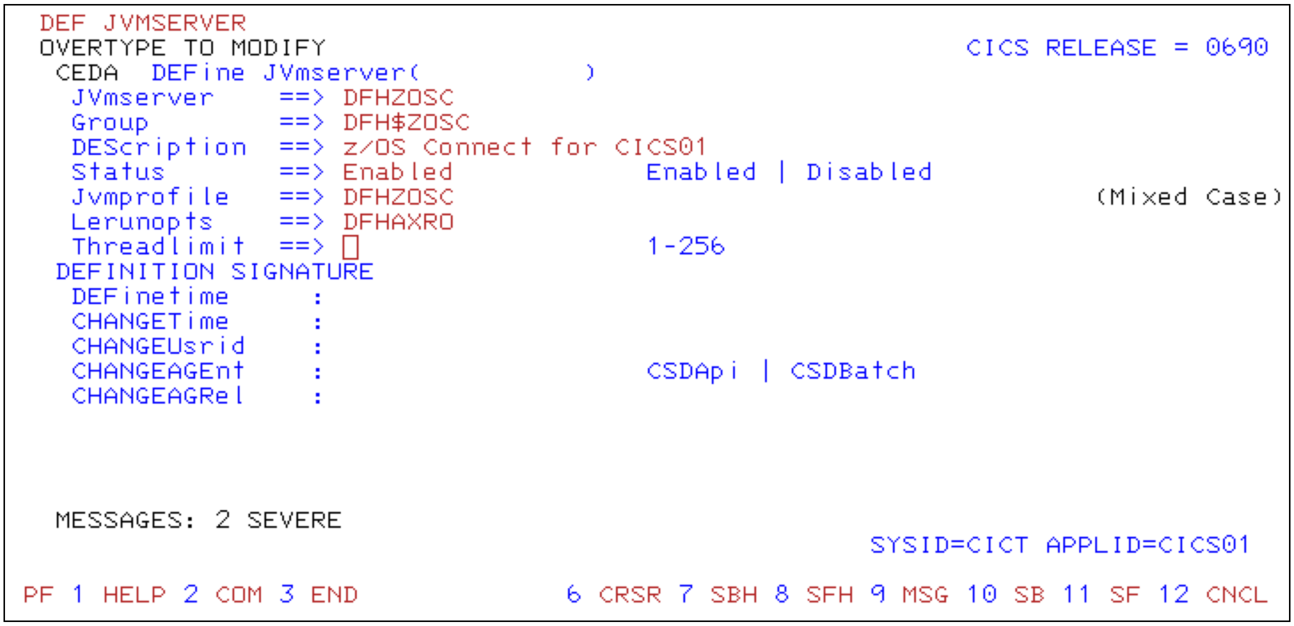
Figure 7-2 Defining a JVM server to host WebSphere Liberty Profile for z/OS Connect

Table 7-1 Description for the attributes of a JVM server

Attribute	Description
Jvmserver	Specifies the 1 - 8 character name of the JVMSERVER resource.
Description	The description text can be up to 58 characters in length. No restrictions apply to the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. If you use the **CREATE** command, for each single apostrophe in the text, code two apostrophes.
Group	Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed. The GROUP name can be up to 8 characters in length. Lowercase characters are treated as uppercase characters.

Attribute	Description
Jvmprofile	Specifies the 1 - 8 character name of the JVM profile containing the JVM options for running a JVM server. The file containing the JVM profile must have a file extension of .jvmprofile. For a JVM profile for a JVMSERVER resource defined by online resource definition, the file path is specified by the system initialization parameter **JVMPROFILEDIR**. For a JVM profile for a JVMSERVER resource defined by online resource definition, the file path is specified by the system initialization parameter.
Lerunopts	Specifies the 1 - 8 character name of the program that defines the runtime options for the IBM Language Environment® enclave. DFHAXRO is a supplied program that provides a set of default values. The source for DFHAXRO is in the *hlq.SDFHSAMP* library if you want to change the defaults for any of the Language Environment runtime options. If you want to use a different program, put the program in the *hlq.SDFHLOAD* library and specify the program name in uppercase characters.
Status	Specifies the initial status of the PIPELINE when it is installed. Valid values are ‘Enabled’ or ‘Disabled’
Threadlimit	Specifies the maximum number of threads that are allowed in the Language Environment enclave for the JVM server. Each thread runs under a T8 TCB. You can specify a limit in the range of 1 - 256 threads. If you specify a thread limit that exceeds the maximum of 2000 threads that is allowed for the CICS region, considering all other enabled and disabled JVMSERVER resources, CICS allocates the remaining threads up to 2000 to the resource as the thread limit value. If CICS is already at the maximum number of JVMSERVER threads, the resource installs in a disabled state.

Here are the values used in this scenario to create the JVM server in Figure 7-3.

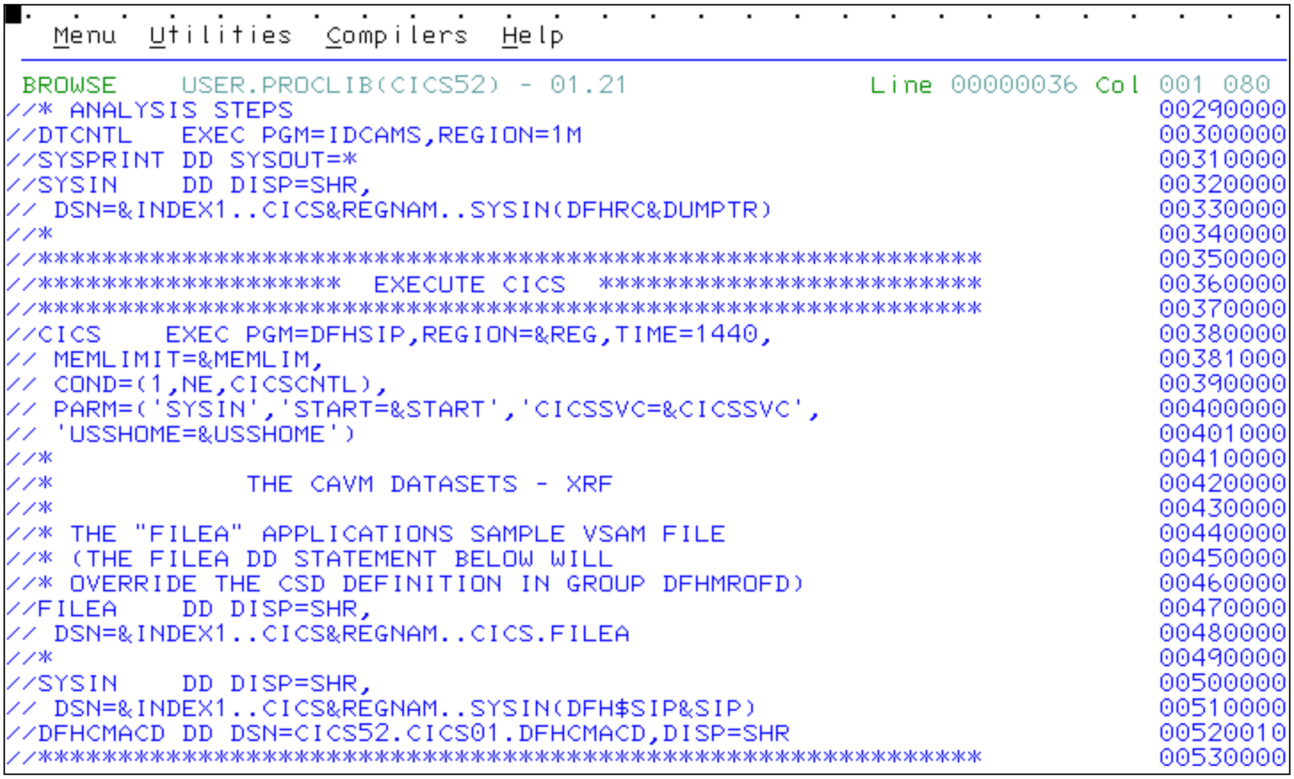


```
 DEF JVMSERVER
 OVERTYPE TO MODIFY                                                    CICS RELEASE = 0690
  CEDA  DEFine JVmserver(          )
   JVmserver     ==> DFHZOSC
   Group         ==> DFH$ZOSC
   DEScription   ==> z/OS Connect for CICS01
   Status        ==> Enabled            Enabled | Disabled
   Jvmprofile    ==> DFHZOSC                                                 (Mixed Case)
   Lerunopts     ==> DFHAXRO
   Threadlimit   ==> □                  1-256
  DEFINITION SIGNATURE
   DEFinetime      :
   CHANGETime      :
   CHANGEUsrid     :
   CHANGEAGEnt     :                    CSDApi | CSDBatch
   CHANGEAGRel     :



  MESSAGES: 2 SEVERE
                                                                 SYSID=CICT APPLID=CICS01

PF 1 HELP 2 COM 3 END                    6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 7-3 Values for creating the JVM server

7.3.2 Configuring the JVM server

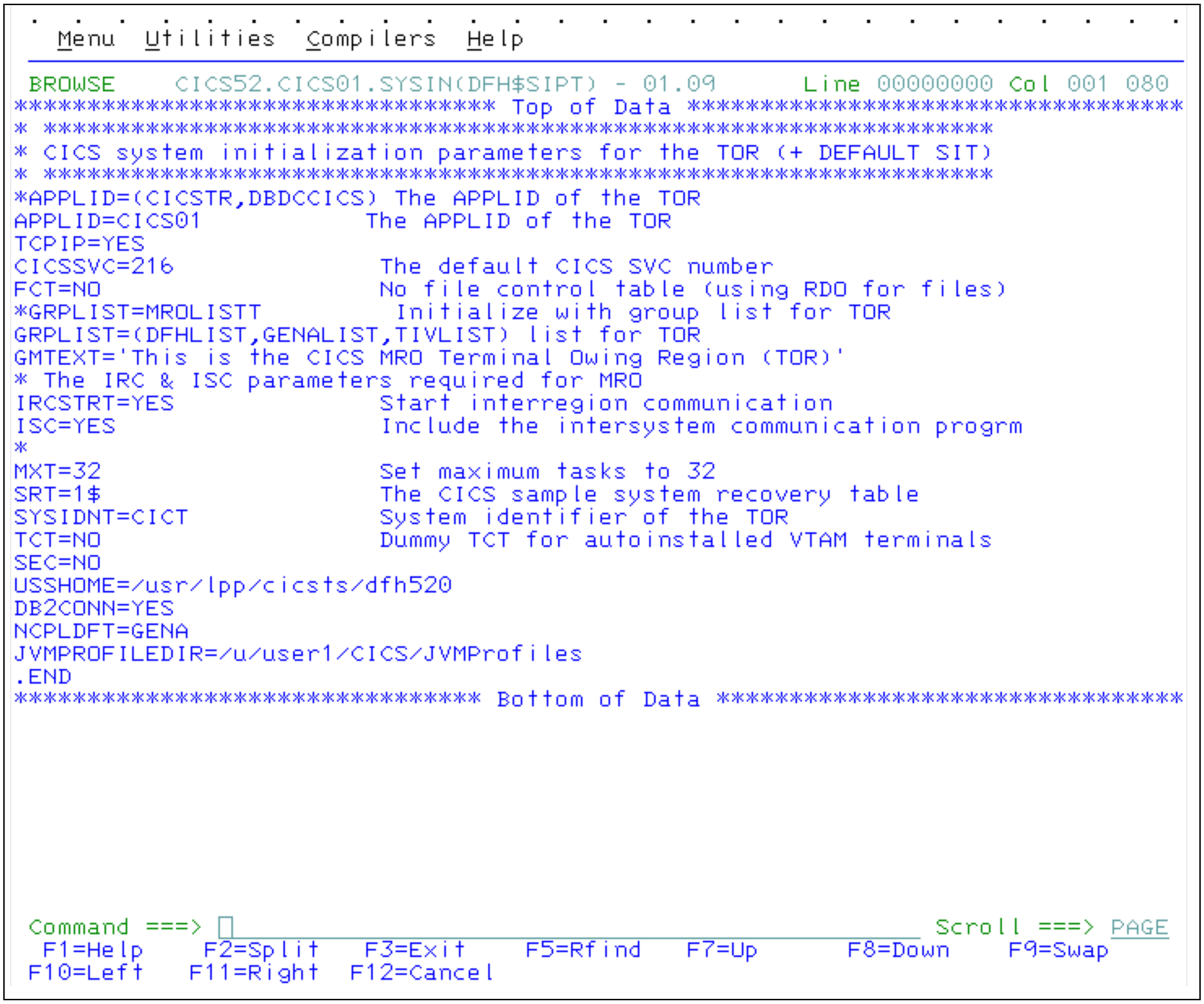
After the JVM is created, the `.jvmprofile` file needs to be created. The location where the `.jvmprofile` should be created is defined in the system initialization data set when CICS is started. The location of this data set is specified in the user PROCLIB of the CICS installation (see Figure 7-4).



```
 BROWSE    USER.PROCLIB(CICS52) - 01.21               Line 00000036 Col 001 080
//* ANALYSIS STEPS                                                      00290000
//DTCNTL   EXEC PGM=IDCAMS,REGION=1M                                    00300000
//SYSPRINT DD SYSOUT=*                                                  00310000
//SYSIN    DD DISP=SHR,                                                 00320000
// DSN=&INDEX1..CICS&REGNAM..SYSIN(DFHRC&DUMPTR)                        00330000
//*                                                                     00340000
//*************************************************************         00350000
//******************   EXECUTE CICS   ************************           00360000
//*************************************************************         00370000
//CICS    EXEC PGM=DFHSIP,REGION=&REG,TIME=1440,                        00380000
// MEMLIMIT=&MEMLIM,                                                    00381000
// COND=(1,NE,CICSCNTL),                                                00390000
// PARM=('SYSIN','START=&START','CICSSVC=&CICSSVC',                     00400000
// 'USSHOME=&USSHOME')                                                  00401000
//*                                                                     00410000
//*            THE CAVM DATASETS - XRF                                  00420000
//*                                                                     00430000
//* THE "FILEA" APPLICATIONS SAMPLE VSAM FILE                           00440000
//* (THE FILEA DD STATEMENT BELOW WILL                                  00450000
//* OVERRIDE THE CSD DEFINITION IN GROUP DFHMROFD)                      00460000
//FILEA    DD DISP=SHR,                                                 00470000
// DSN=&INDEX1..CICS&REGNAM..CICS.FILEA                                 00480000
//*                                                                     00490000
//SYSIN    DD DISP=SHR,                                                 00500000
// DSN=&INDEX1..CICS&REGNAM..SYSIN(DFH$SIP&SIP)                         00510000
//DFHCMACD DD DSN=CICS52.CICS01.DFHCMACD,DISP=SHR                       00520010
//*************************************************************         00530000
```

Figure 7-4 User PROCLIB for the CICS Transaction Server named CICS52. The system initialization data set for CICS is specified in SYSIN when CICS is started

Next, the data set needs to be modified to specify the path in zFS where the `.jvmprofile` should go in Figure 7-5. The data set needs to specify the JVMPROFILEDIR property with the path where the `.jvmprofile` should go.



```
 BROWSE    CICS52.CICS01.SYSIN(DFH$SIPT) - 01.09      Line 00000000 Col 001 080
********************************* Top of Data **********************************
* *****************************************************************
* CICS system initialization parameters for the TOR (+ DEFAULT SIT)
* *****************************************************************
*APPLID=(CICSTR,DBDCCICS) The APPLID of the TOR
APPLID=CICS01            The APPLID of the TOR
TCPIP=YES
CICSSVC=216               The default CICS SVC number
FCT=NO                    No file control table (using RDO for files)
*GRPLIST=MROLISTT          Initialize with group list for TOR
GRPLIST=(DFHLIST,GENALIST,TIVLIST) list for TOR
GMTEXT='This is the CICS MRO Terminal Owing Region (TOR)'
* The IRC & ISC parameters required for MRO
IRCSTRT=YES               Start interregion communication
ISC=YES                   Include the intersystem communication progrm
*
MXT=32                    Set maximum tasks to 32
SRT=1$                    The CICS sample system recovery table
SYSIDNT=CICT              System identifier of the TOR
TCT=NO                    Dummy TCT for autoinstalled VTAM terminals
SEC=NO
USSHOME=/usr/lpp/cicsts/dfh520
DB2CONN=YES
NCPLDFT=GENA
JVMPROFILEDIR=/u/user1/CICS/JVMProfiles
.END
******************************** Bottom of Data ********************************

 Command ===>                                               Scroll ===> PAGE
```

Figure 7-5 The system input parameter table is modified to specify JVMPROFILEDIR as /u/user1/CICS/JVMProfiles

A template can typically be found in zFS here:

```
/usr/lpp/cicsts/JVMProfiles/DFHWLP.jvmprofile
```

Note: The template file might be in a different directory depending on how the CICS Transaction Server was installed.

This file can be copied to the path specified by the JVMPROFILEDIR. The file name must be the same as the JVMPROFILE attribute, which was specified when the JVM server was created. In this scenario, the JVM profile directory was created here:

```
/u/user1/CICS/JVMProfiles/DFHZOSC.jvmprofile
```

The following properties should be modified to values that are applicable to the system running z/OS Connect:

- JAVA_HOME should be set to the location of your installed IBM Java SDK.
- WORK_DIR should be set to your choice of destination directory for messages, trace, and output from the JVM server. In this scenario, it has been set to /u/user1/CICS.
- WLP_INSTALL_DIR should be set to &USSHOME;/wlp.

> **Note:** The `-Dcom.ibm.cics.jvmserver.wlp.autoconfigure` property is purposely left uncommented, which means that the server.xml must be provided by the user. This is recommended for production environments.
>
> When the autoconfigure property is not set, the following properties are ignored from the `.jvmprofile` configuration file:
>
> ```
> -Dcom.ibm.cics.jvmserver.wlp.server.http.port.
> -Dcom.ibm.cics.jvmserver.wlp.server.host.
> -Dcom.ibm.cics.jvmserver.wlp.server.name.
> ```
>
> **Note:** The WLP_USER_DIR property is left uncommented, which means that it uses the default value, which is WORK_DIR/<APPLID>/<JVM server name>. In this scenario, the effective value for WLP_USER_DIR is /u/user1/CICS/CICS01/DFHZOSC.

7.3.3 Mapping the pipeline for existing CICS transactions to z/OS Connect

You can follow these steps to map the pipeline for existing CICS transactions to z/OS Connect.

Creating the pipeline in CICS

The first step in mapping a CICS transaction to z/OS Connect is to create a CICS pipeline.

To define the pipeline in CICS, it must know which JVM server to use as well as the directory where the web service bind files are located.

Creating the pipeline XML file for the JVM server

The JVM server for the pipeline must be defined in an XML file. A template for this file is found in zFS here:

```
/usr/lpp/cicsts/samples/pipelines/jsonzosconnectprovider.xml
```

> **Note:** The template file might be in a different directory depending on how the CICS Transaction Server was installed.

In this scenario, the template file was copied here:

```
/u/user1/CICS/pipeline/DFHZOSC_pipeline.xml
```

In this file, the value of the *jvmserver* property needs to be changed to match the name of the JVMSERVER used for z/OS Connect. In this scenario, the JVM server name is DFHZOSC. The pipeline XML file used in this scenario is in Example 7-1.

Example 7-1 The pipeline XML file that will be used in the definition of the pipeline

```
<?xml version="1.0"?>
<!--
<copyright
 notice="cics-lm-source-program"
 pids="5655-Y04"
 years="2013,2014"
 crc="2249679448" >


 Licensed Materials - Property of IBM

 5655-Y04

 (C) Copyright IBM Corp. 2014 All Rights Reserved.

 US Government Users Restricted Rights - Use, duplication or
 disclosure restricted by GSA ADP Schedule Contract with
 IBM Corp.

 @{[**]copyright.years=2014}


 </copyright>
-->
<provider_pipeline_json xmlns="http://www.ibm.com/software/htp/cics/pipeline">
  <jvmserver>DFHZOSC</jvmserver>
</provider_pipeline_json>
```

Creating the directory for the web service bind files

A set of web service bind files will be created for each program or transaction that is mapped to a web service.

The directory for these bind files must be manually created. In this scenario, this directory was created:

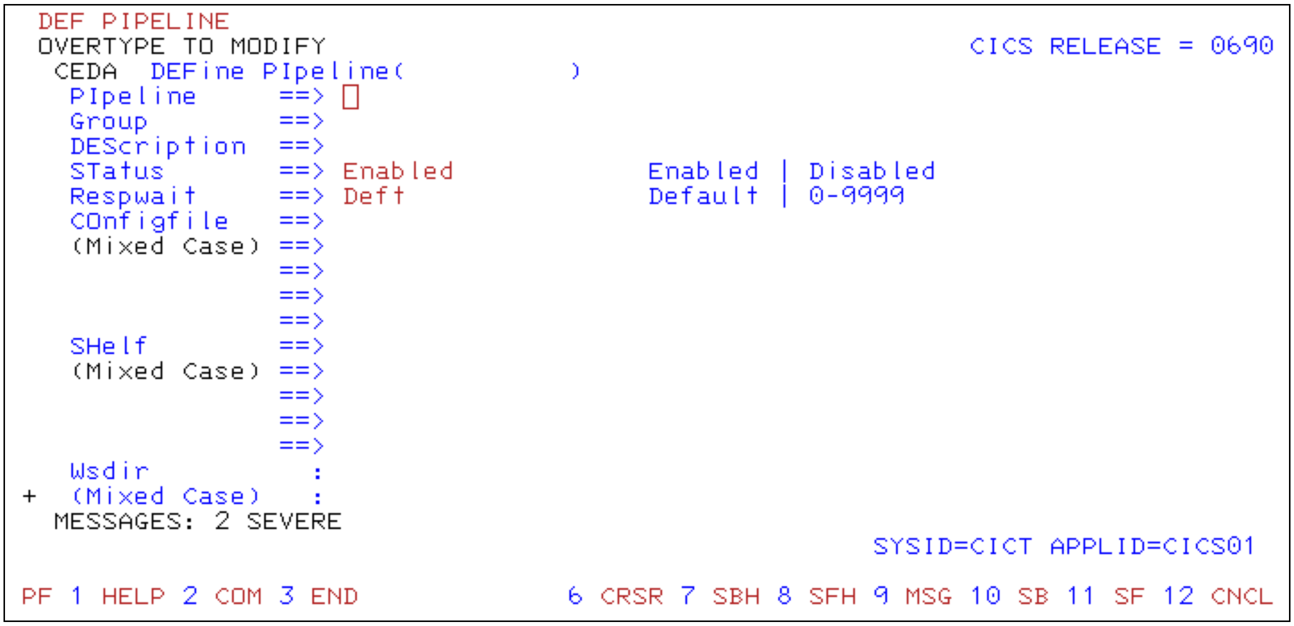
```
/u/user1/CICS/wsbind/
```

Defining the pipeline

After creating the pipeline XML file and the directory for the web service bind files, the pipeline can be defined in CICS. From a CICS terminal, use this command:

```
CEDA DEF PIPELINE
```

The panel for defining a pipeline is displayed as shown in Figure 7-6 on page 166. The attributes for the pipeline definition are described in Table 7-2 on page 166.



```
  DEF PIPELINE
  OVERTYPE TO MODIFY                                                CICS RELEASE = 0690
   CEDA  DEFine PIpeline(          )
    PIpeline      ==> □
    Group         ==>
    DEScription   ==>
    STatus        ==> Enabled            Enabled | Disabled
    Respwait      ==> Deft               Default | 0-9999
    COnfigfile    ==>
    (Mixed Case)  ==>
                  ==>
                  ==>
                  ==>
    SHelf         ==>
    (Mixed Case)  ==>
                  ==>
                  ==>
                  ==>
    Wsdir           :
 +  (Mixed Case)    :
   MESSAGES: 2 SEVERE
                                                              SYSID=CICT APPLID=CICS01

 PF 1 HELP 2 COM 3 END                     6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

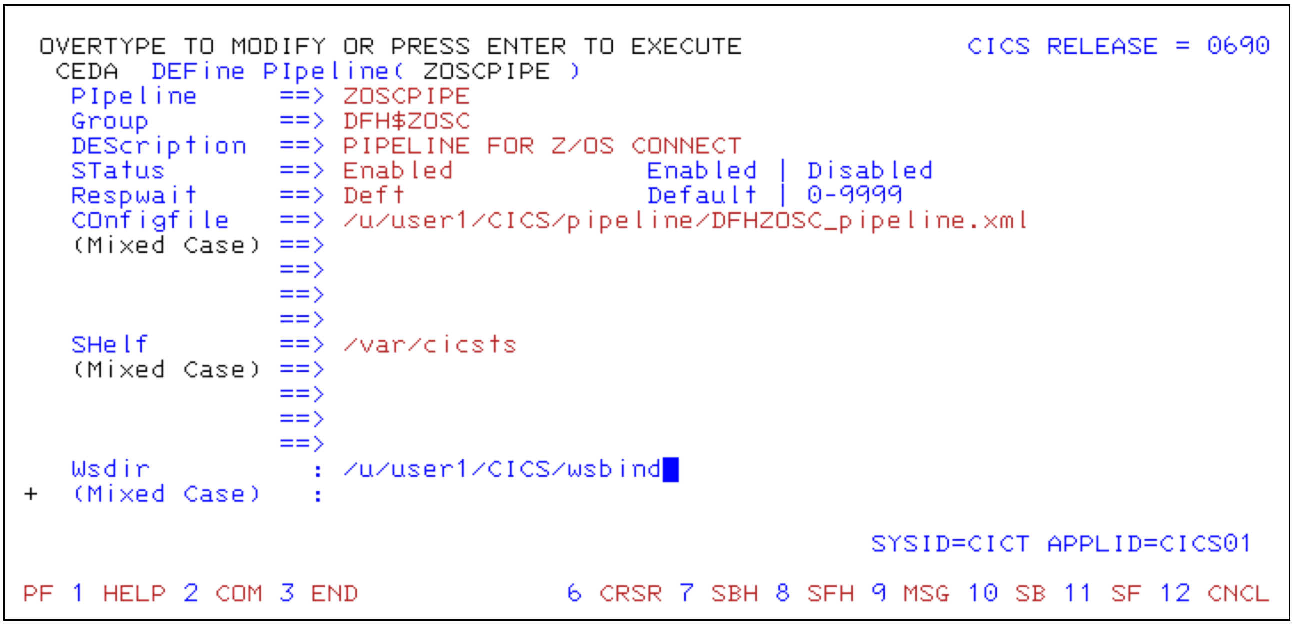
Figure 7-6 Defining a CICS pipeline for the web service

Table 7-2 Descriptions for the attributes of a pipeline

Attribute	Description
Configfile	Specifies the fully qualified or relative name of a z/OS UNIX file that contains information about the processing nodes that act on a service request, and on the response. A sample pipeline configuration file for z/OS Connect can be found in "Creating the pipeline XML file for the JVM server" on page 164.
Description	The description text can be up to 58 characters in length. No restrictions apply to the characters that you can use. However, if you use parentheses, ensure that for each left parenthesis there is a matching right one. If you use the **CREATE** command, for each single apostrophe in the text, code two apostrophes.
Group	Every resource definition must have a GROUP name. The resource definition becomes a member of the group and is installed in the CICS system when the group is installed. The GROUP name can be up to 8 characters in length. Lowercase characters are treated as uppercase characters.
Pipeline	Specifies the name of this PIPELINE. The name can be up to 8 characters in length.
Respwait	Specifies the number of seconds that an application program should wait for a response message from a remote web service. The value can range 0 - 9999 seconds. To use the default timeout value of the transport protocol, specify DEFT. The default timeout value is also used if you do not specify any value for RESPWAIT.

Attribute	Description
Shelf	Specifies the 1 – 255 character fully qualified name of a directory (a *shelf*, primarily for web service binding files) on z/OS UNIX. CICS regions into which the PIPELINE definition is installed must have full permissions to the shelf directory: Read, write, and the ability to create subdirectories. A single shelf directory can be shared by multiple CICS regions and by multiple PIPELINE definitions. Within a shelf directory, each CICS region uses a separate subdirectory to keep its files separate from those of other CICS regions. Within each region's directory, each PIPELINE uses a separate subdirectory. You should not attempt to modify the contents of a shelf that is referred to by an installed PIPELINE definition. If you do, the effects are unpredictable.
Status	Specifies the initial status of the PIPELINE when it is installed. Valid values are 'Enabled' or 'Disabled'
Wsdir	Specifies the 1 – 255 character fully qualified name of the web service binding directory (also known as the *pickup directory*) on z/OS UNIX. Each PIPELINE installed in a CICS region must specify a different web service binding directory. The web service binding directory contains web service binding files that are associated with a PIPELINE, and that are to be installed automatically by the CICS scanning mechanism. When the PIPELINE definition is installed, CICS scans the directory and automatically installs any web service binding files that it finds there. This happens regardless of whether the PIPELINE is installed in an enabled or disabled state. If you specify a value for the WSDIR attribute, it must refer to a valid z/OS UNIX directory to which the CICS region has at least read access. If not, any attempt to install the PIPELINE resource fails. If you do not specify a value for WSDIR, no automatic scan takes place on installation of the PIPELINE.

For the pipeline that is created in this scenario, the values are displayed in Figure 7-7.



```
 OVERTYPE TO MODIFY OR PRESS ENTER TO EXECUTE              CICS RELEASE = 0690
  CEDA  DEFine PIpeline( ZOSCPIPE )
   PIpeline      ==> ZOSCPIPE
   Group         ==> DFH$ZOSC
   DEScription   ==> PIPELINE FOR Z/OS CONNECT
   STatus        ==> Enabled             Enabled | Disabled
   Respwait      ==> Deft                Default | 0-9999
   COnfigfile    ==> /u/user1/CICS/pipeline/DFHZOSC_pipeline.xml
   (Mixed Case)  ==>
                 ==>
                 ==>
                 ==>
   SHelf         ==> /var/cicsts
   (Mixed Case)  ==>
                 ==>
                 ==>
                 ==>
   Wsdir           : /u/user1/CICS/wsbind
+  (Mixed Case)    :

                                                         SYSID=CICT APPLID=CICS01

PF 1 HELP 2 COM 3 END             6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 7-7 Define pipeline window

Create the mapping between JSON and COBOL program

The CICS JSON assistant is a set of batch utilities that creates a mapping between JSON schema and language structures. This mapping is used by CICS at run time to do the transformation between JSON and application data. The assistant supports rapid deployment of CICS applications for use in service providers and service requesters, with the minimum of programming effort.

When you use the JSON assistant for CICS, you do not have to write your own code for parsing inbound messages and for constructing outbound messages; CICS maps data between the JSON message and the application program's data structure.

In this scenario, the DFHLS2JS procedure is used to generate a JSON schema file from our COBOL program.

The DFHLS2JS JCL procedure is installed in the data set HLQ.XDFHINST, where HLQ is the high-level qualifier where CICS is installed.

The JCL used to create the mapping in this scenario is shown in Example 7-2. In this scenario, the program being called is BBIORD01 and the transaction is BBOI. There are three output files, the `.wsbind` file, the request JSON file, and the response JSON file. The three files are placed into the directory named:

```
/u/user1/CICS/wsbind
```

This directory is the same one that was created in "Creating the directory for the web service bind files" on page 165 and specified as a parameter when the pipeline was defined.

The URI that is used when the web service is called is /BBAPP/InquireOrder. This is arbitrary and the name that is chosen for this scenario.

Example 7-2 JCL that calls DFHLS2JS JSON assistant for CICS to create the proper mapping from JSON to the program's data structure

```
//LS2JS EXEC DFHLS2JS,
//  USSDIR='dfh520',
//  PATHPREF='',
//  JAVADIR='java/J7.1_64',
//  TMPDIR='/tmp',
//  TMPFILE=''
//INPUT.SYSUT1 DD *
LOGFILE=/u/user1/CICS/LS2JS_LGIJORD01.LOG
PDSLIB=USER1.BB12.SOURCE
REQMEM=SOAI001
RESPMEM=SOAI001
LANG=COBOL
PGMNAME=BBIORD01
URI=/BBAPP/InquireOrder
TRANSACTION=BBOI
PGMINT=COMMAREA
MAPPING-LEVEL=3.0
WSBIND=/u/user1/CICS/wsbind/getOrderDetails.wsbind
JSON-SCHEMA-REQUEST=/u/user1/CICS/wsbind/getOrderDetailsReq.json
JSON-SCHEMA-RESPONSE=/u/user1/CICS/wsbind/getOrderDetailsResp.json
/*
```

7.3.4 Configure WebSphere Liberty Profile for z/OS Connect

The WebSphere Liberty Profile for z/OS Connect needs to be configured by creating the `server.xml` file manually. A template of the `server.xml` file can be found here:

```
/usr/lpp/cicsts/wlp/templates/servers/defaultServer/server.xml
```

Note: The template file might be in a different directory depending on how the CICS Transaction Server was installed.

The template `server.xml` file should be copied here:

```
/u/user1/CICS/CICS01/DFHZOSC/wlp/usr/servers/defaultServer/server.xml
```

This path is based on the WLP_USER_DIR property in the `.jvmprofile` configuration file. In this scenario, the WLP_USER_DIR is `/u/user1/CICS/CICS01/DFHZOSC`. The rest of the directory structure is expected by the WebSphere Liberty Profile for the server.xml.

> **Note:** The `server.xml` file must be encoded in UTF-8. The **`EU`** command can be used under z/OS UNIX Directory List utility (ISPF menu option 3.17).

Example 7-3 is the `server.xml` file used for this scenario. It has been updated from the template file to add the following:

- The feature managers for cicsts:zOS-Connect-1.0
- The *httpEndpoint* was modified to specify the *httpsPort*
- The CICSEndpoint attribute
- The zOSConnectService attribute. The values are from the JCL that was executed to create the web service bindings in Example 7-2 on page 169

Example 7-3 The server.xml configuration file for WebSphere Liberty Profile for z/OS Connect

```
<?xml version="1.0" encoding="UTF-8"?><server description="CICS Liberty profile
sample configuration">

    <!-- Enable features -->
    <featureManager>
        <feature>cicsts:core-1.0</feature>
        <feature>jsp-2.2</feature>
        <feature>wab-1.0</feature>
        <feature>blueprint-1.0</feature>

      <feature>ssl-1.0</feature>
        <feature>cicsts:zosConnect-1.0</feature>
        <feature>appSecurity-2.0</feature>
    </featureManager>

    <!-- Default HTTP End Point -->
    <httpEndpoint host="*" httpPort="9080" httpsPort="9443"
id="defaultHttpEndpoint"/>

    <com.ibm.cics.wlp.zosconnect.CICSEndpoint
id="com.ibm.cics.wlp.zosconnect.CICSEndpointService"/>
s
    <zosConnectService invokeURI="/BBAPP/InquireOrder"
serviceName="getOrderDetails"
serviceRef="com.ibm.cics.wlp.zosconnect.CICSEndpointService"/>

    <!-- CICS Bundle Installed Applications -->
    <include location="${server.output.dir}/installedApps.xml"/>

    <!-- The following configuration controls how often server.xml
         is scanned for updates. The default is every 500ms which may
         cause excessive I/O and CPU cost on z/OS.
         The values shown below reduce the overhead while still
         providing a relatively timely detection of new applications
         that have been installed/removed via a CICS Bundle
         (WAR bundlepart). If you use CICS bundles to install Web
         Applications (WAR files) do not disable the polling.
    -->
    <config monitorInterval="5s" updateTrigger="polled"/>

    <!-- Further scanning is performed to detect application updates or
         addition/removal of applications to the dropins directory. If
         you are using CICS Bundles as the vehicle for Application
         deployment you should disable the dropins directory as shown.
         Further I/O and CPU reduction can be achieved by disabling
         the application scan. To effect changes to your applications
         while the server is still running, you should disable and
         re-enable the CICS bundle that contains the Web application.
         The pollingRate is only applicable when the updateTrigger is
         set to the 'polled' value.
         Consult the WebSphere Application Server Liberty Profile
         documentation for further information on these parameters.
    -->
    <applicationMonitor dropins="dropins" dropinsEnabled="false" pollingRate="5s"
updateTrigger="disabled"/>

   <ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore" sslProtocol="TLS"/>
<keyStore id="defaultKeyStore" password="defaultPassword"/>

</server>
```

7.3.5 Starting the JVM server and pipeline

Now, all the artifacts needed for our web service have been created and we can install the JVM server configured to z/OS Connect for CICS and the pipeline from z/OS Connect to our CICS transaction.

The JVM server should be installed before the pipeline is installed.

Installing the JVM server

To install the JVM server, use the CEDA transaction to open the JVM server with this command:

```
CEDA INSTALL GROUP(DFH$ZOSC) JVMSERVER(DFHZOSC)
```

Use the same **GROUP** and **JVMSERVER** parameters when the JVM server was defined in 7.3.1, “Creating the JVM server for z/OS Connect” on page 160.

Installing the pipeline

To install the pipeline, use the CEDA transaction to install the pipeline with this command:

```
CEDA INSTALL GROUP(DFH$ZOSC) PIPELINE(ZOSCPIPE)
```

Use the same **GROUP** and **PIPELINE** parameters when the pipeline was defined in 7.3.3, “Mapping the pipeline for existing CICS transactions to z/OS Connect” on page 164.

7.3.6 Accessing the REST API

Now that z/OS Connect is running and the web services have been mapped to the CICS transaction, z/OS Connect can be validated to ensure that it is running as well as the available web services it is providing.

Simply call this URL:

```
https://<server name or IP Address>:9443/zosConnect/services
```

The port 9443 is the httpsPort that was defined in the **server.xml** file in 7.3.4, “Configure WebSphere Liberty Profile for z/OS Connect” on page 169.

The output in this scenario looks like what is shown in Example 7-4.

Example 7-4 Output from REST API for validating z/OS Connect setup

```
{
   "zosConnectServices": [
      {
         "ServiceName":"getOrderDetails",
         "ServiceDescription":"DATA_UNAVAILABLE",
         "ServiceProvider":"com.ibm.cics.wlp.zosconnect.CICSEndpointServiceImpl",
      "ServiceURL":"https://cics-server:9443/zosConnect/services/getOrderDetails"
      }
   ]
}
```

7.4 Connecting IBM Bluemix to CICS

Bluemix has a service named *Secure Gateway service* to connect Bluemix to on-premises systems.

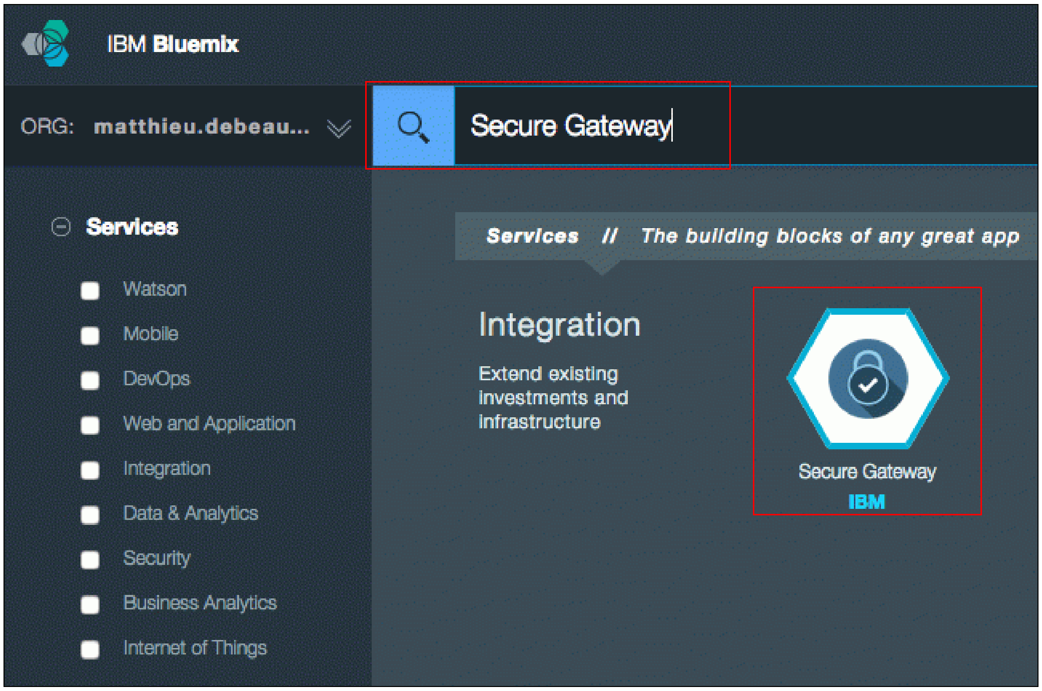
In this scenario, the Secure Gateway service is used to connect the application running in Bluemix to the CICS system via z/OS Connect.

The Secure Gateway service requires a client that can reach the z/OS Connect system. In this scenario, the client running on-premises is a Data Power appliance.

Configuring the Secure Gateway

To configure the Secure Gateway service, log on to Bluemix and complete the following steps:

1. Click **Catalog**. Then, enter `Secure Gateway` in the **Search** field as shown in Figure 7-8. Click **Secure Gateway**.



ORG: matthieu.debeau...

Secure Gateway

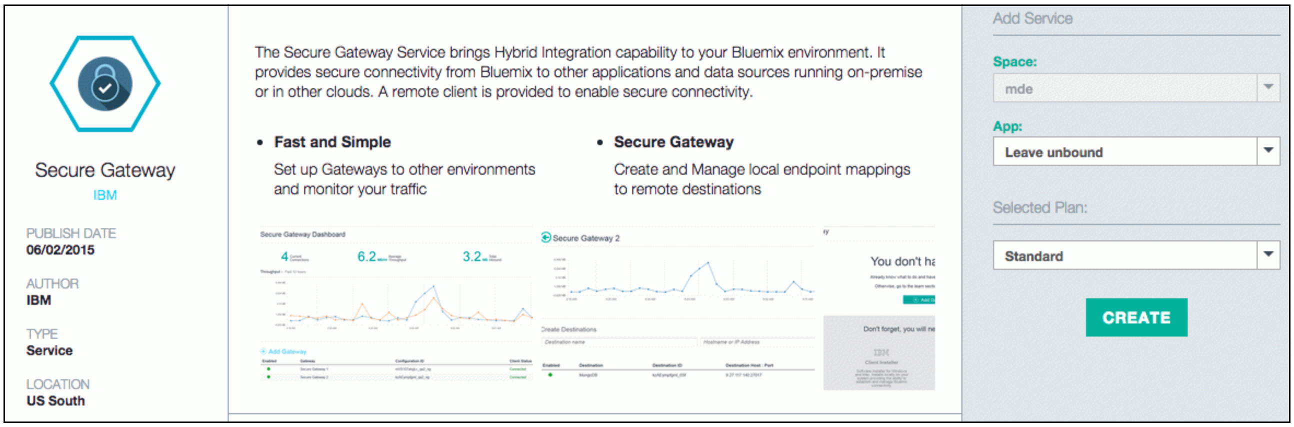
Services // The building blocks of any great app

Integration

Extend existing investments and infrastructure

Figure 7-8 Bluemix catalog with Secure Gateway service

2. Click **CREATE** as shown in Figure 7-9.



Secure Gateway
IBM

PUBLISH DATE
06/02/2015

AUTHOR
IBM

TYPE
Service

LOCATION
US South

The Secure Gateway Service brings Hybrid Integration capability to your Bluemix environment. It provides secure connectivity from Bluemix to other applications and data sources running on-premise or in other clouds. A remote client is provided to enable secure connectivity.

- **Fast and Simple**
  Set up Gateways to other environments and monitor your traffic
- **Secure Gateway**
  Create and Manage local endpoint mappings to remote destinations

Add Service

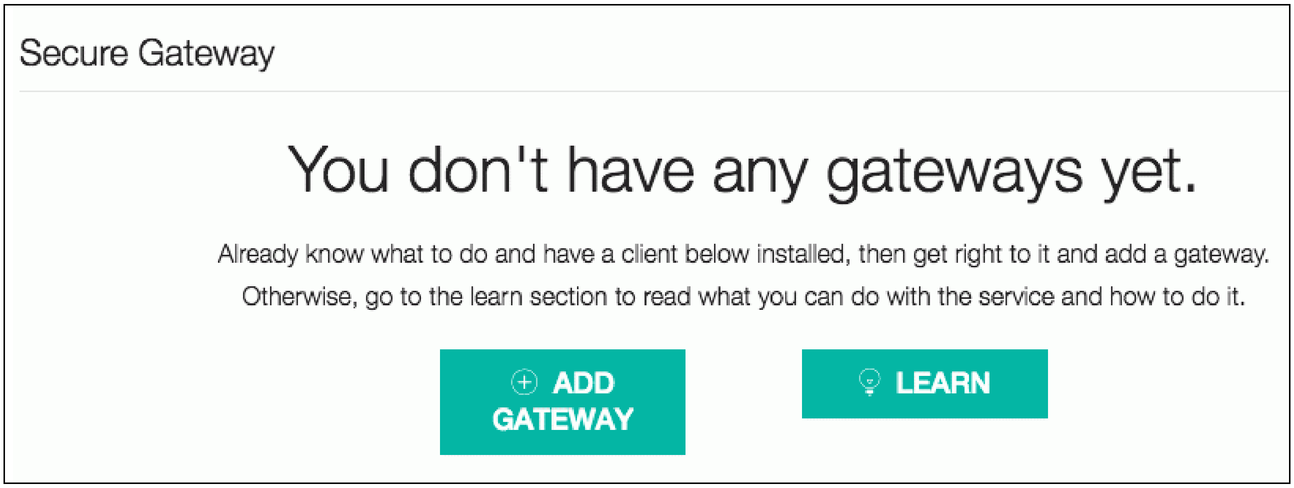
Space: mde

App: Leave unbound

Selected Plan: Standard

Figure 7-9 Secure Gateway service details

3. Click **ADD GATEWAY** to create a new gateway instance, as shown in Figure 7-10.



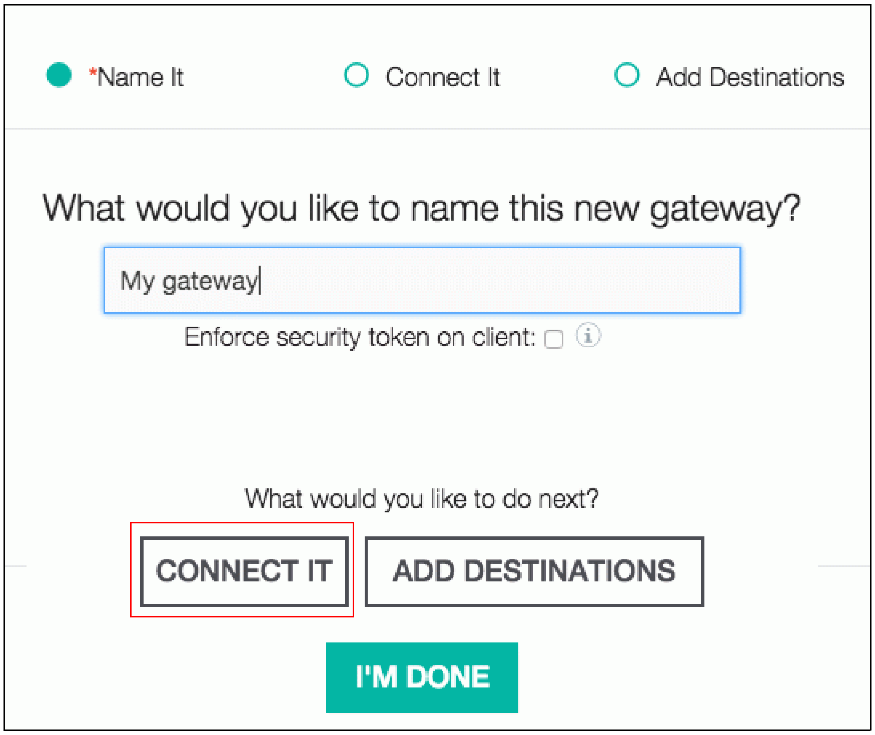
Secure Gateway

You don't have any gateways yet.

Already know what to do and have a client below installed, then get right to it and add a gateway. Otherwise, go to the learn section to read what you can do with the service and how to do it.

Figure 7-10 Secure Gateway landing page

4. Enter `My gateway` in the **What would you like to name this new gateway?** field, as shown in Figure 7-11. Click **CONNECT IT**.



What would you like to name this new gateway?

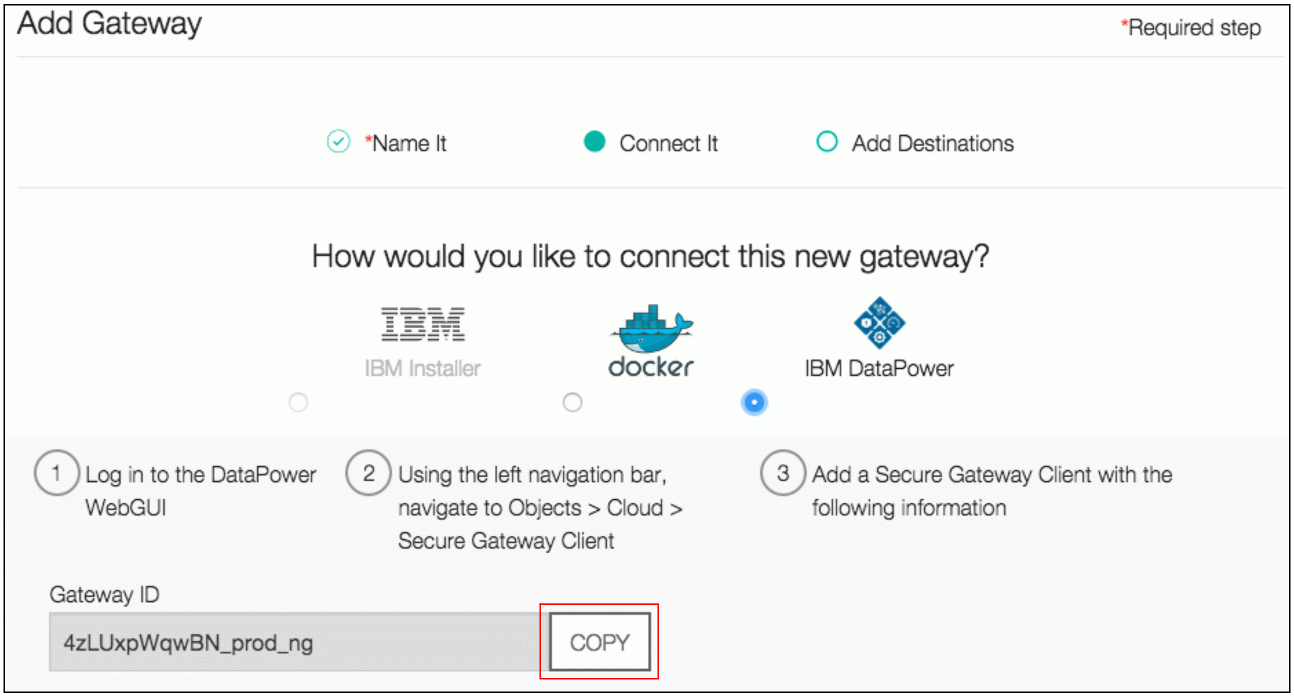
My gateway

Enforce security token on client: □

What would you like to do next?

Figure 7-11 Secure Gateway creation process

5. Click **IBM DataPower** as shown in Figure 7-12. Click **COPY** to copy the generated **Gateway ID** to clipboard.



Add Gateway

*Required step

How would you like to connect this new gateway?

1 Log in to the DataPower WebGUI

2 Using the left navigation bar, navigate to Objects > Cloud > Secure Gateway Client

3 Add a Secure Gateway Client with the following information

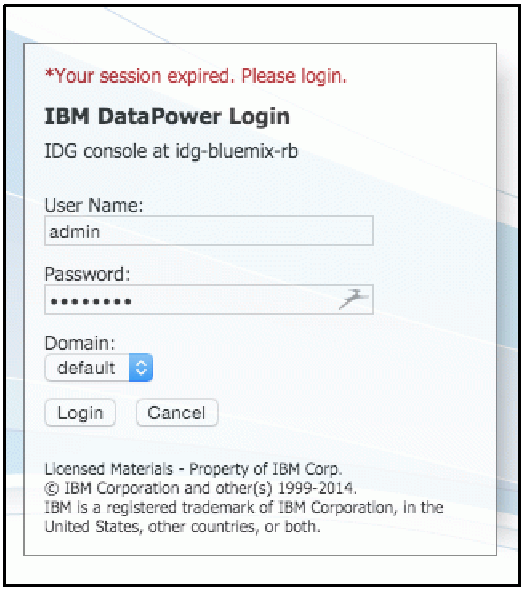
Gateway ID: 4zLUxpWqwBN_prod_ng

Figure 7-12 Secure Gateway connection process

Configuring the IBM DataPower Gateway

To continue the integration process, open a new web browser window, connect to IBM DataPower Gateway web interface, and complete the following steps:

1. Log in to IBM DataPower Gateway web interface as shown in Figure 7-13.



*Your session expired. Please login.

IBM DataPower Login

IDG console at idg-bluemix-rb

User Name: admin

Password: ••••••••

Domain: default

Licensed Materials - Property of IBM Corp.
© IBM Corporation and other(s) 1999-2014.
IBM is a registered trademark of IBM Corporation, in the United States, other countries, or both.

Figure 7-13 IBM DataPower Gateway login page

2. Enter `secure gateway` in the **Search** field in the upper left corner of the window, as shown in Figure 7-14. Click **Secure Gateway Client**.

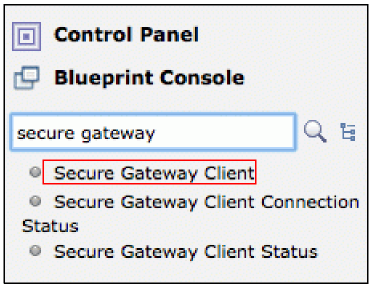
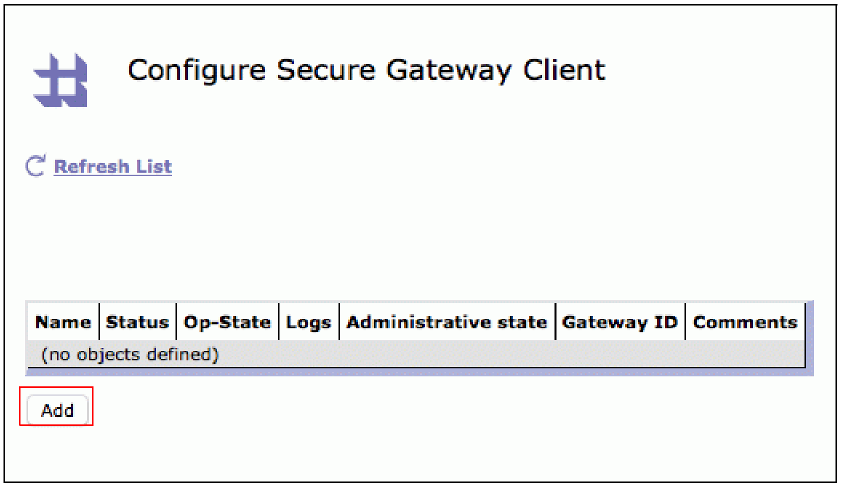

Control Panel
Blueprint Console
secure gateway
Secure Gateway Client
Secure Gateway Client Connection Status
Secure Gateway Client Status

Figure 7-14 IBM DataPower Gateway search

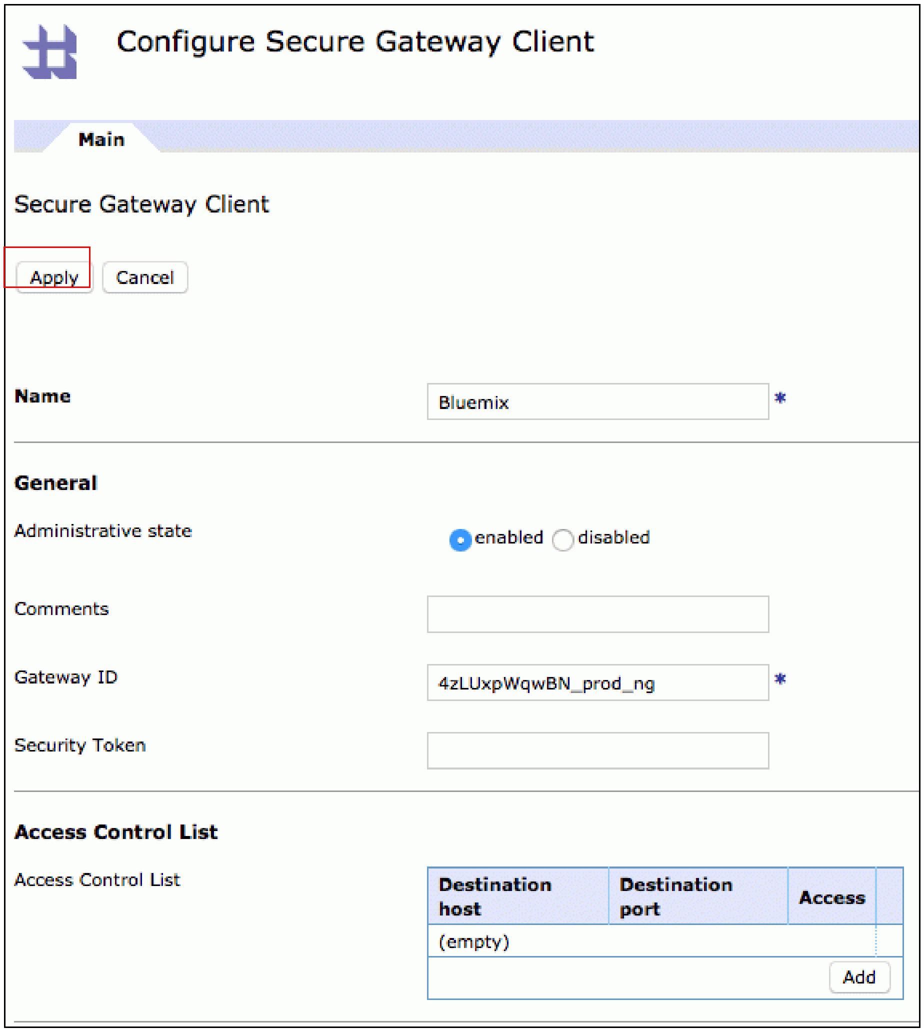
3. Click **Add** as shown in Figure 7-15.


Configure Secure Gateway Client

Name	Status	Op-State	Logs	Administrative state	Gateway ID	Comments
(no objects defined)						

Figure 7-15 Secure Gateway Client list

4. Enter `Bluemix` in the **Name** field. Paste the *Gateway ID* previously copied in the **Gateway ID** field, as shown in Figure 7-16. Click **Apply**.



Configure Secure Gateway Client

Secure Gateway Client

Name: Bluemix *

General

Administrative state: ● enabled ○ disabled

Comments:

Gateway ID: 4zLUxpWqwBN_prod_ng *

Security Token:

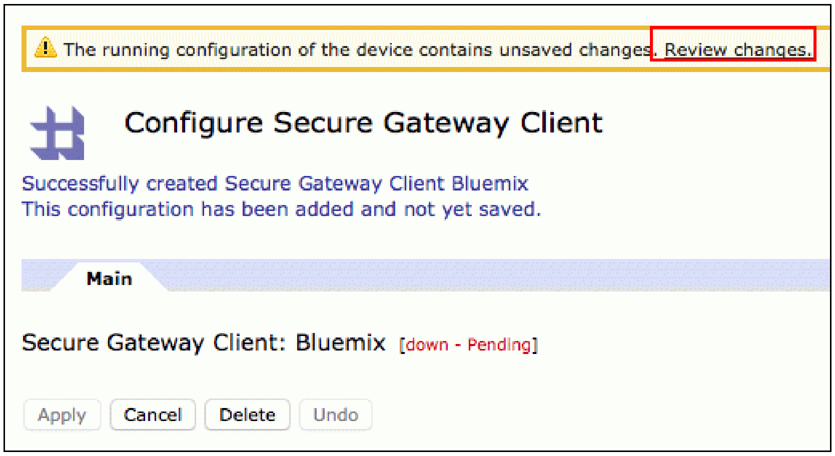
Access Control List

Access Control List:

Destination host	Destination port	Access
(empty)		

Figure 7-16 Secure Gateway Client creation form

5. Click **Review changes** as shown in Figure 7-17.



⚠ The running configuration of the device contains unsaved changes. Review changes.

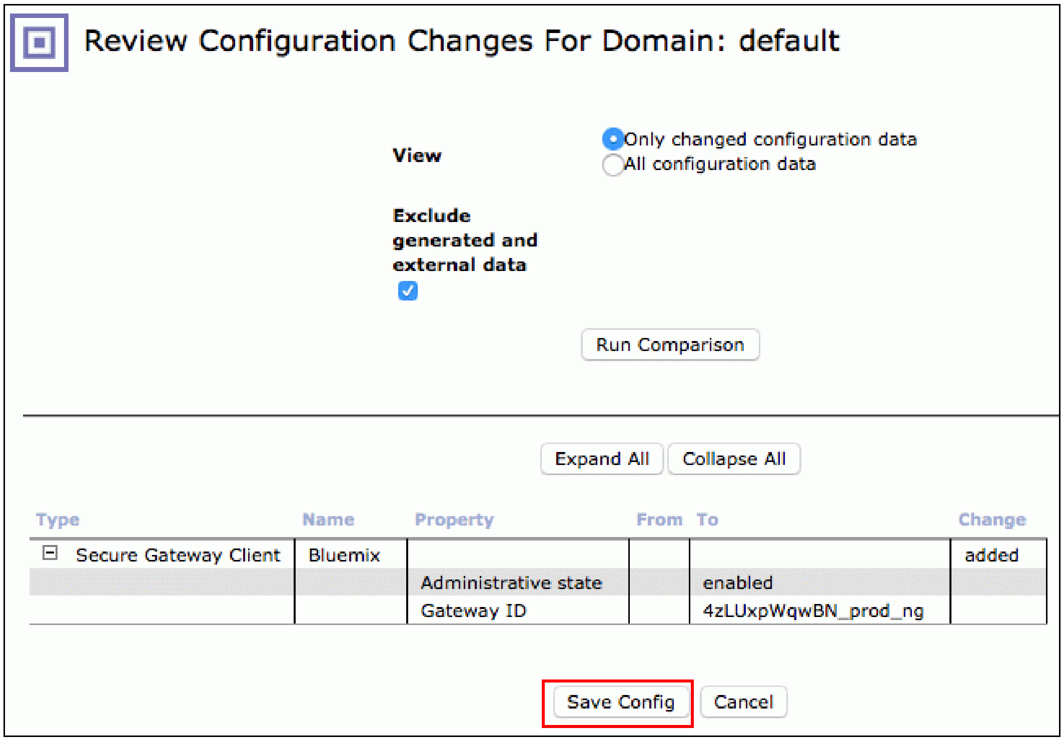
Configure Secure Gateway Client

Successfully created Secure Gateway Client Bluemix
This configuration has been added and not yet saved.

Secure Gateway Client: Bluemix [down - Pending]

Figure 7-17 Secure Gateway Client pending changes

6. Click **Save Config** as shown in Figure 7-18.



Review Configuration Changes For Domain: default

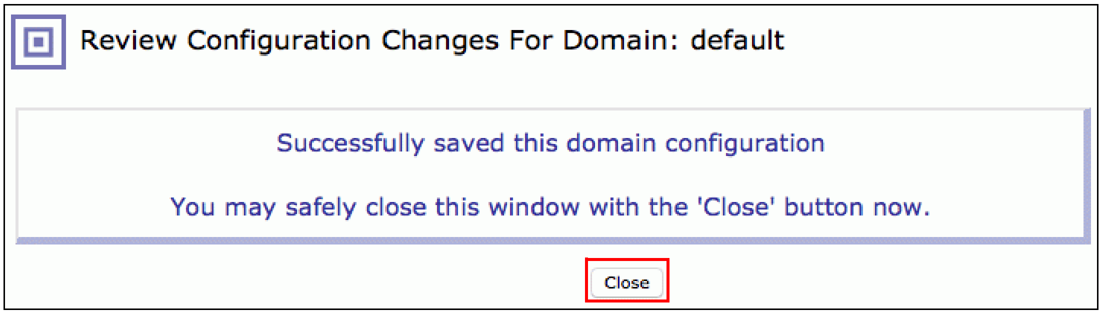
View: ● Only changed configuration data ○ All configuration data

Exclude generated and external data ☑

Type	Name	Property	From	To	Change
⊟ Secure Gateway Client	Bluemix				added
		Administrative state		enabled	
		Gateway ID		4zLUxpWqwBN_prod_ng	

Figure 7-18 IBM DataPower Gateway saving configuration

7. A confirmation message appears as shown in Figure 7-19. Click **Close** and close the web browser window.



Review Configuration Changes For Domain: default

Successfully saved this domain configuration

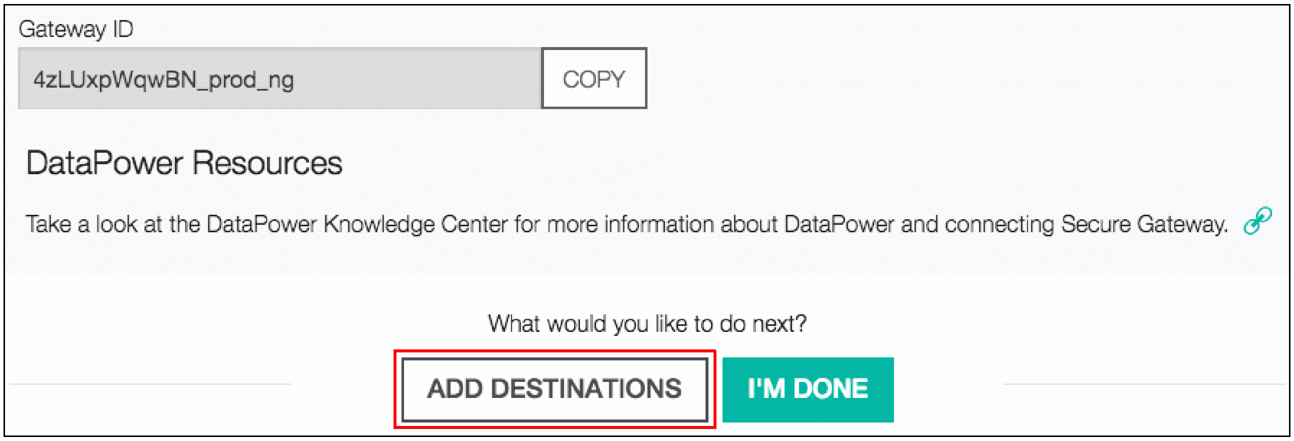
You may safely close this window with the 'Close' button now.

Figure 7-19 IBM DataPower Gateway configuration changes confirmation

Adding a destination in the Secure Gateway

To terminate the integration process, go back to the window displayed at the end of step 5 on page 175 and complete the following steps:

1. Click **ADD DESTINATIONS** as shown in Figure 7-20.



Gateway ID

4zLUxpWqwBN_prod_ng

DataPower Resources

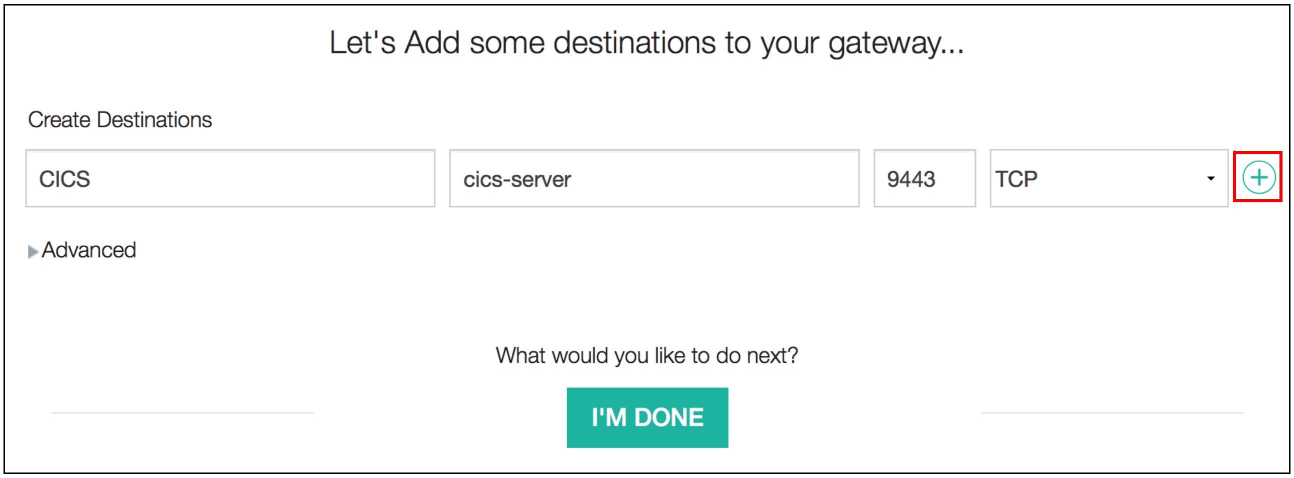
Take a look at the DataPower Knowledge Center for more information about DataPower and connecting Secure Gateway.

What would you like to do next?

Figure 7-20 Secure Gateway connection process continued

2. In the **Create Destinations** area, complete the following actions, as shown in Figure 7-21.
   a. Enter `CICS` in the **Destination name** field.
   b. Enter *cics_server_hostname* in the **Hostname or IP Address** field.
   c. Enter *httpsPort* of z/OS Connect in the **Port** field.
   d. Make sure `TCP` is selected in the list.
   e. Click the **round plus** shaped icon.

   > **Note:** For more information about **Advanced Options**, refer to the Bluemix documentation:
   >
   > `https://www.ng.bluemix.net/docs/services/SecureGateway/sg_022.html#sg_009`



Let's Add some destinations to your gateway...

Create Destinations

CICS

cics-server

9443

TCP

What would you like to do next?

Figure 7-21 Creating a new destination for the secure gateway

3. Click **I'M DONE** to finish the gateway's creation as shown in Figure 7-21.

4. This completes the setup of the Secure Gateway to our CICS Transaction Server. To validate that the connection is working, retrieve the cloud host name and port to use for connecting to our system. This can be done by opening the Secure Gateway service for the Bluemix space from the dashboard and drilling down to the gateway that was created to view the destinations.
5. Now select the **information** icon as shown in Figure 7-22. In the dialog box that is displayed, record the **Cloud Host:Port** field to use as the host and port name to connect to when validating the connection.

   Use this URL to test the connection:

   `https://<Cloud Host>:<Port>/zosConnect/services.`

   The result should be the same as shown in Example 7-4 on page 172.

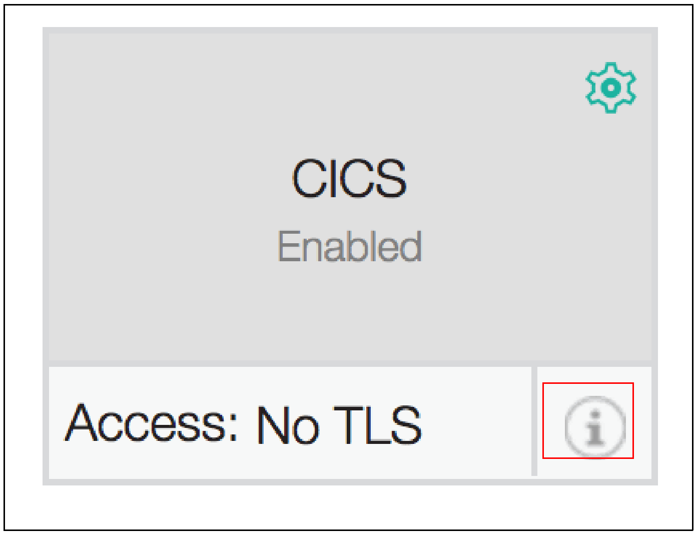


Figure 7-22 Destination for z/OS Connect

7.5 Using IBM API Management for your new APIs

When the Secure Gateway destination has been created and validated to z/OS Connect, the API Management service can be used to manage the lifecycle, access control, and traffic of the API.

7.5.1 Creating the IBM API Management Service

Follow these steps to create the IBM API Management Service:

1. From the catalog window as shown in Figure 7-23, choose **API Management**.

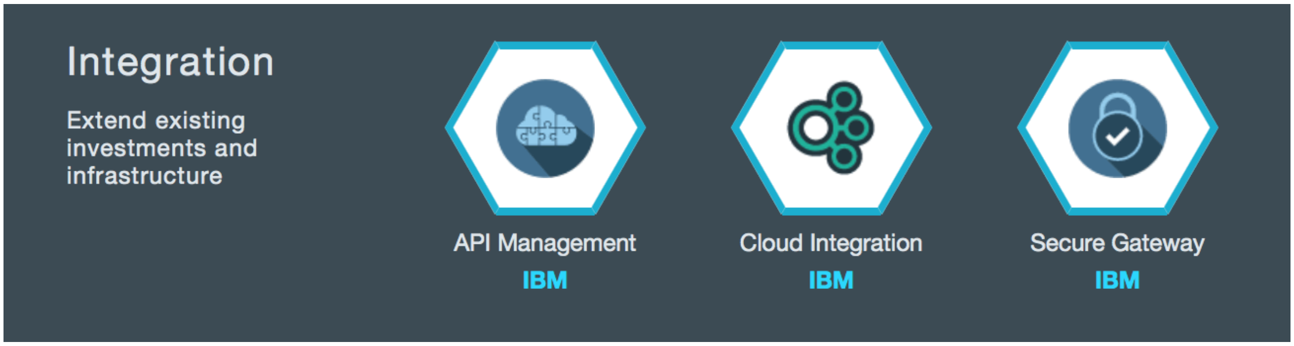
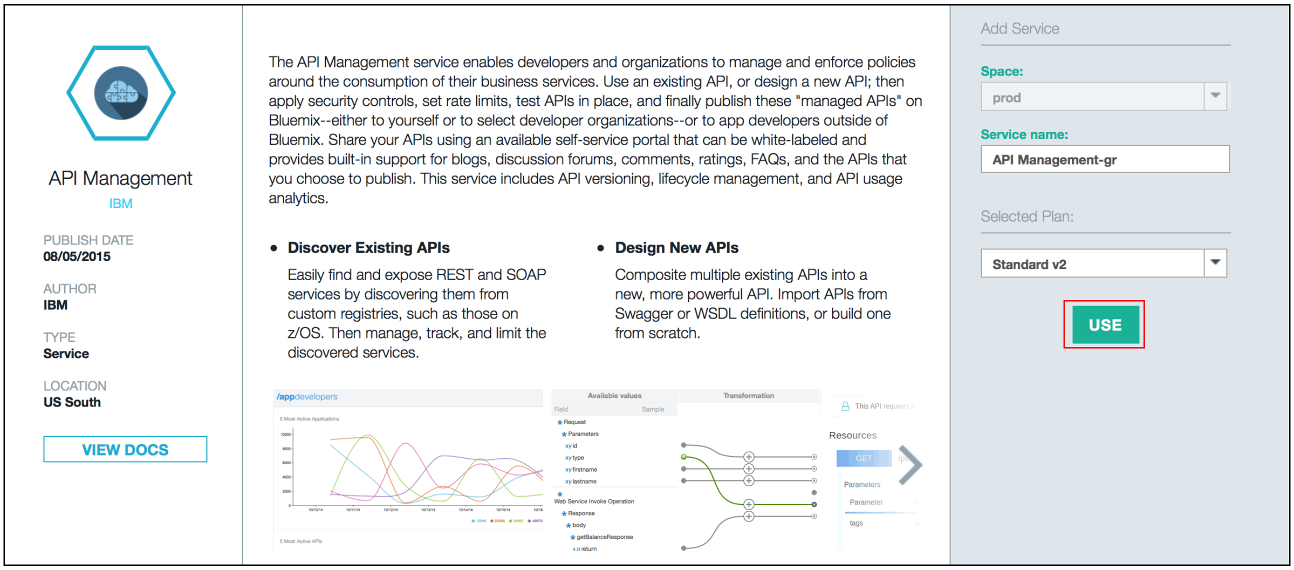


Figure 7-23 Catalog of the Integration services in Bluemix

2. Select the plan that best fits the intended usage. Currently, only one plan includes a free tier. When finished, click the **Use** button as shown in Figure 7-24.



API Management
IBM

PUBLISH DATE
08/05/2015

AUTHOR
IBM

TYPE
Service

LOCATION
US South

VIEW DOCS

The API Management service enables developers and organizations to manage and enforce policies around the consumption of their business services. Use an existing API, or design a new API; then apply security controls, set rate limits, test APIs in place, and finally publish these "managed APIs" on Bluemix--either to yourself or to select developer organizations--or to app developers outside of Bluemix. Share your APIs using an available self-service portal that can be white-labeled and provides built-in support for blogs, discussion forums, comments, ratings, FAQs, and the APIs that you choose to publish. This service includes API versioning, lifecycle management, and API usage analytics.

- **Discover Existing APIs**
  Easily find and expose REST and SOAP services by discovering them from custom registries, such as those on z/OS. Then manage, track, and limit the discovered services.
- **Design New APIs**
  Composite multiple existing APIs into a new, more powerful API. Import APIs from Swagger or WSDL definitions, or build one from scratch.

Add Service

Space: prod

Service name: API Management-gr

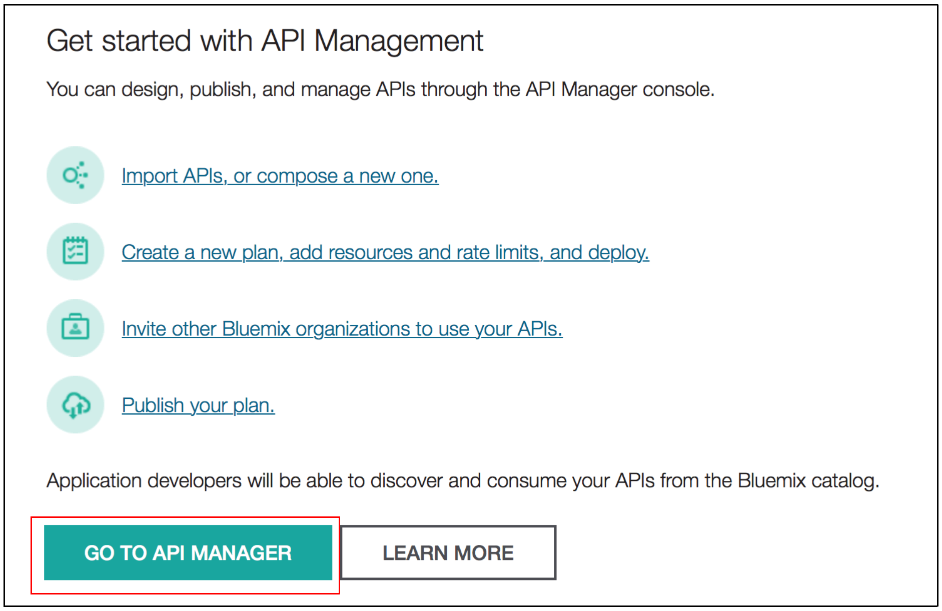
Selected Plan: Standard v2

USE

Figure 7-24 Select the appropriate plan for the API Management Service

The service should now be created and a panel with the launch button into the API Management console should appear, as shown in Figure 7-25.

3. Click **Go to API Manager** to open the API Management console.



Get started with API Management

You can design, publish, and manage APIs through the API Manager console.

Import APIs, or compose a new one.
Create a new plan, add resources and rate limits, and deploy.
Invite other Bluemix organizations to use your APIs.
Publish your plan.

Application developers will be able to discover and consume your APIs from the Bluemix catalog.

GO TO API MANAGER
LEARN MORE

Figure 7-25 Launchpad for the API Management service that was created

7.5.2 Using API Manager to build and publish a new API

The process for building and publishing an API involves four steps:

1. Create the API. This is where we define the URL and the mapping to our z/OS Connect server as well as documentation for the API.
2. Create an environment. This is the environment where the API will be run. The environment can be customized with a custom URL as well as other properties, such as developer portal URL, user registry, and permissions for who can do what in the environment. For example, who can publish a plan in the environment.
3. Create a plan. This is where the traffic management, such as rate limiting for the API can be set. Multiple plans can be created for the same API to allow for charging of different rates. In this example, only one plan is created.
4. Stage and publish the plan into an environment.

Creating the API

When creating an API, there are multiple options to define the API. The API can be defined in the API Manager via a wizard or by importing a Swagger 2.0 document or Web Service Description Language (WSDL) file. In this example, the API is defined with a Swagger document:

> **Note:** Swagger 2.0 is an open standard for defining and documenting APIs. To learn more about the Swagger 2.0, see `http://swagger.io`.

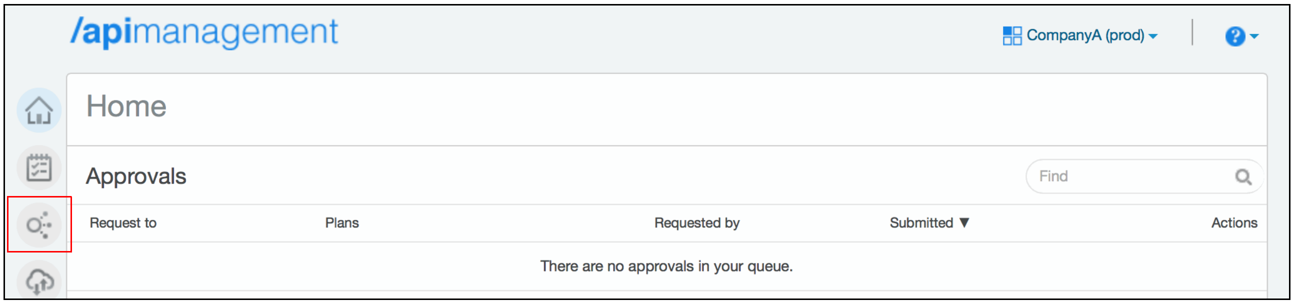
1. The Swagger document that is used for this example is shown in Example 7-5. The **Host** field in the example should refer to the Secure Gateway destination, which was described in “Adding a destination in the Secure Gateway” on page 179.

   The **basePaths** and **paths** fields come from the **URI** parameter of the JCL program on z/OS, as shown in Example 7-2 on page 169.

Example 7-5 Swagger 2.0 document used for representing the new API

```
{
    "swagger": "2.0",
    "info": {
        "title": "Virtual Shopping List",
        "description": "API to service the Virtual Shopping List App",
        "version": "1.0.0"
    },
    "host": "<Cloud Host>:<Port>",
    "schemes": [
        "https"
    ],
    "basePath": "/BBAPP",
    "produces": [
        "application/json"
    ],
    "paths": {
        "/InquireOrder": {
            "post": {
                "summary": "Retrieve shopping list of specified customer",
                "description": "This API is used for retrieving customer order
history via CICS running on premise on z/OS.",
                "responses": {
                    "200": {
                        "description": "Orders were successfully created",
                        "schema": {}
                    },
                    "default": {
                        "description": "",
                        "schema": {}
                    }
                }
            }
        }
    },
    "definitions": {
    }
}
```

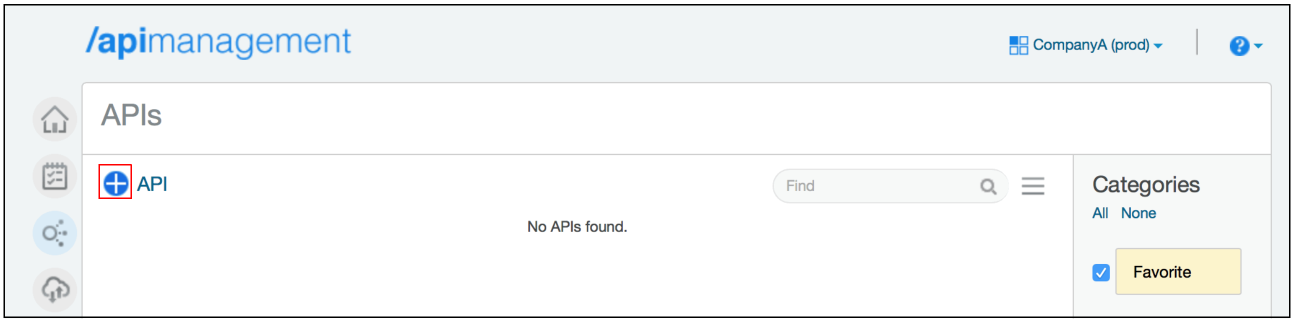
2. With the Swagger 2.0 document ready, go back to the API Manager console and go to the API view by selecting the **API icon** on the left, as shown in Figure 7-26.



There are no approvals in your queue.

Figure 7-26 API Manager home page

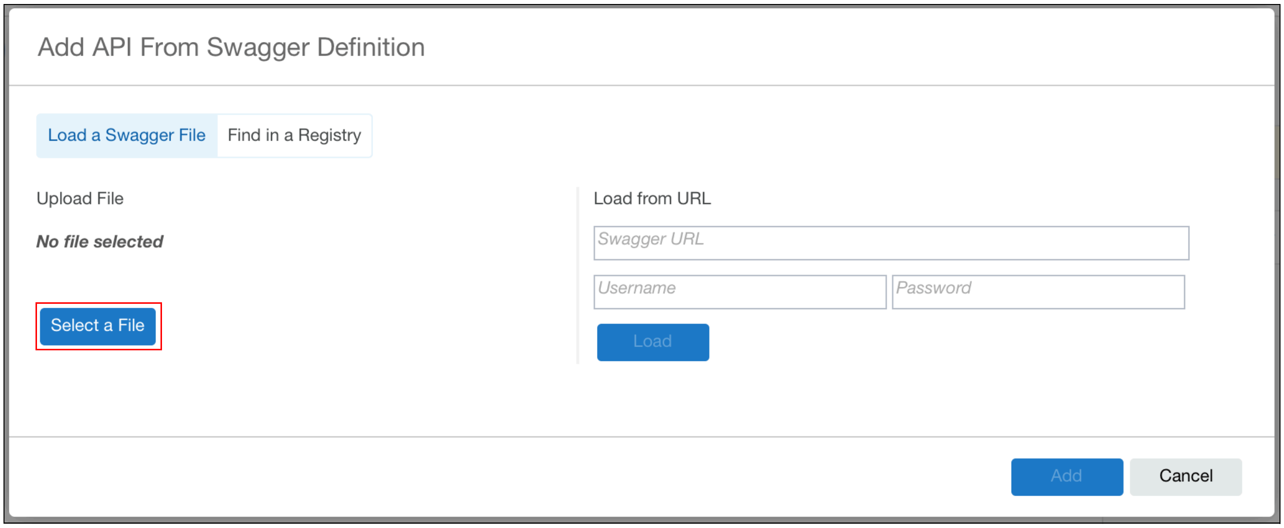
3. Click the **Add** icon as shown in Figure 7-27 to begin creating a new API. A menu should be displayed to select the method for defining the API. Select **Import Swagger 2.0** to import a swagger document.



No APIs found.

Figure 7-27 API list. Currently, there are no APIs. Use the Add button to create a new API

4. A dialog should be displayed to import a swagger definition as shown in Figure 7-28. Choose **Select a File** to choose the file containing the Swagger definition.

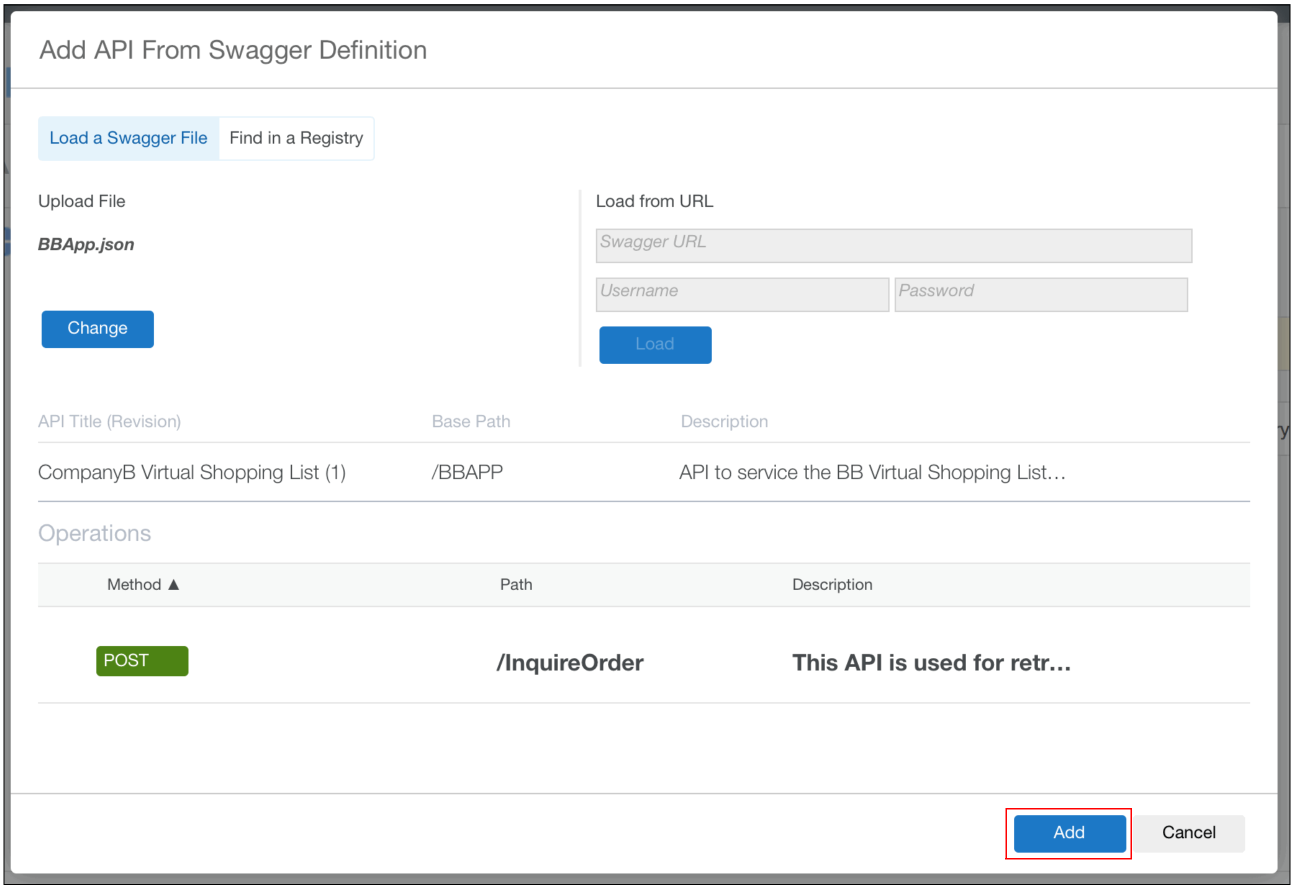


No file selected

Figure 7-28 Dialog for importing a file for a Swagger definition

When the file is selected, the service validates that the file is a valid Swagger file and displays the API that is created as shown in Figure 7-29.

5. Click **Add** to complete the process.



Add API From Swagger Definition

Load a Swagger File | Find in a Registry

Upload File

BBApp.json

Change

Load from URL

Swagger URL

Username | Password

Load

API Title (Revision)	Base Path	Description
CompanyB Virtual Shopping List (1)	/BBAPP	API to service the BB Virtual Shopping List…

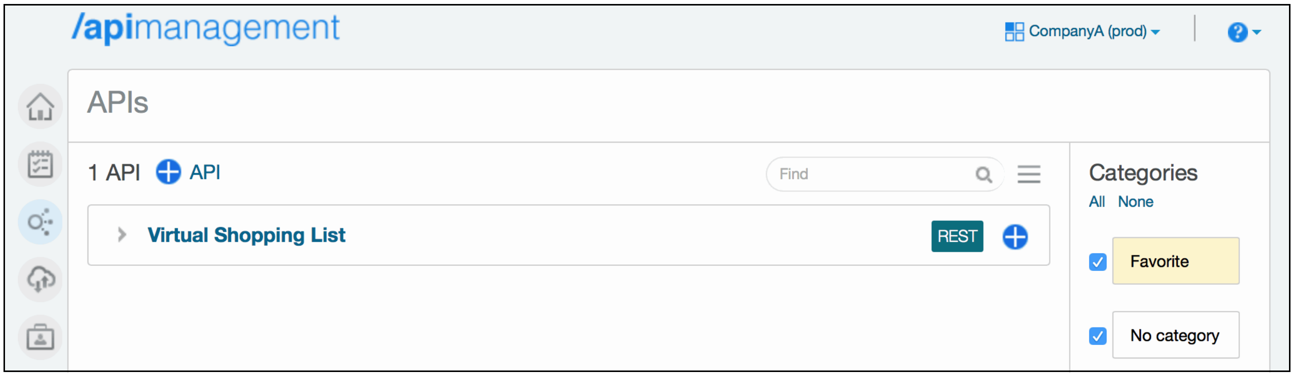
Operations

Method ▲	Path	Description
POST	**/InquireOrder**	**This API is used for retr…**

Add | Cancel

Figure 7-29 Importing a Swagger file

6. When the import is complete, the API is listed under APIs in the API Manager console, as shown in Figure 7-30.



/apimanagement

CompanyA (prod)

APIs

1 API | API

Find

Virtual Shopping List | REST

Categories

All None

☑ Favorite

☑ No category

Figure 7-30 The list of APIs now includes the new API imported from a Swagger file

Creating an environment

Environments are used to configure access to the APIs. Environments contain the information of the URL to use for accessing the APIs as well as the developer portal. Environments can be customized to allow only certain users to do things like publish an API.

1. To create an environment, click the **Environments** icon as shown in Figure 7-31.

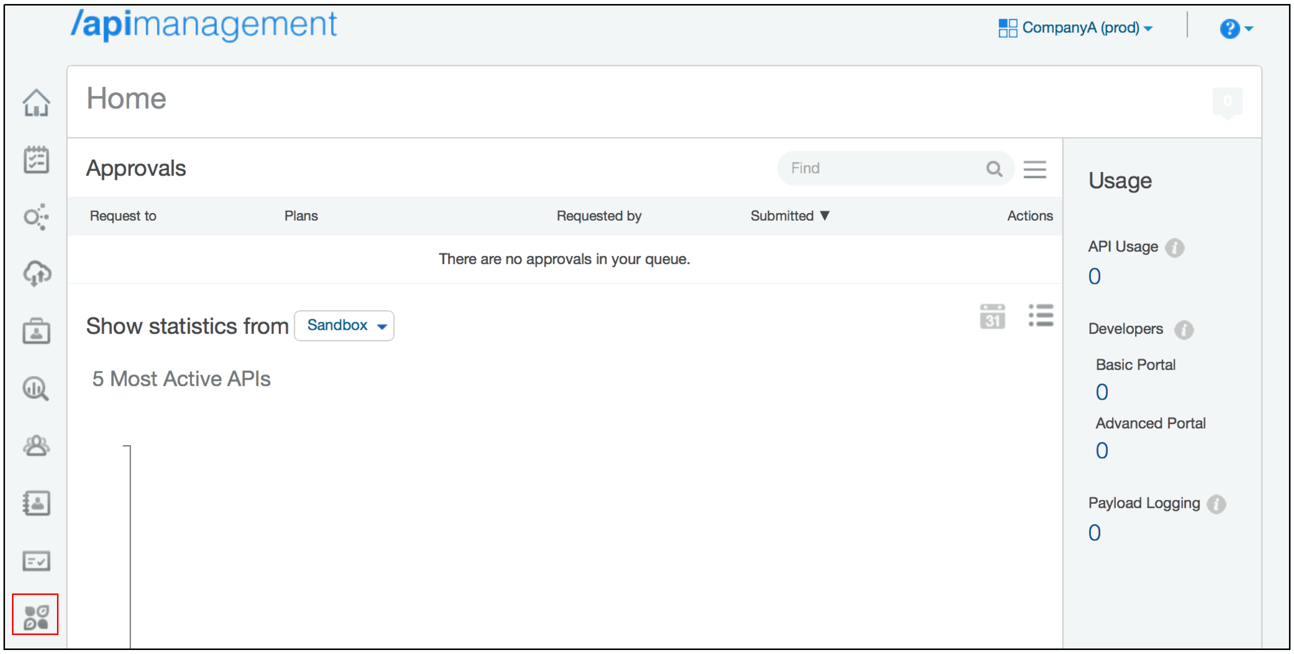


Figure 7-31 Select the Environments icon to create a new environment

By default, there is an environment that is created called **Sandbox**. It is recommended to create a separate environment for publishing new APIs, which is not a sandbox. A sandbox type environment automatically forces the staging and publishing of new plans.

2. Click the **Plus** button to create a new environment as shown in Figure 7-32.

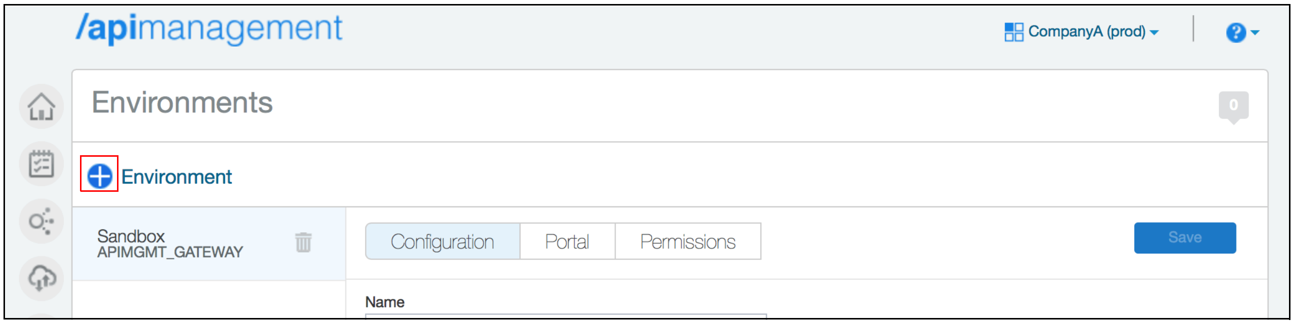
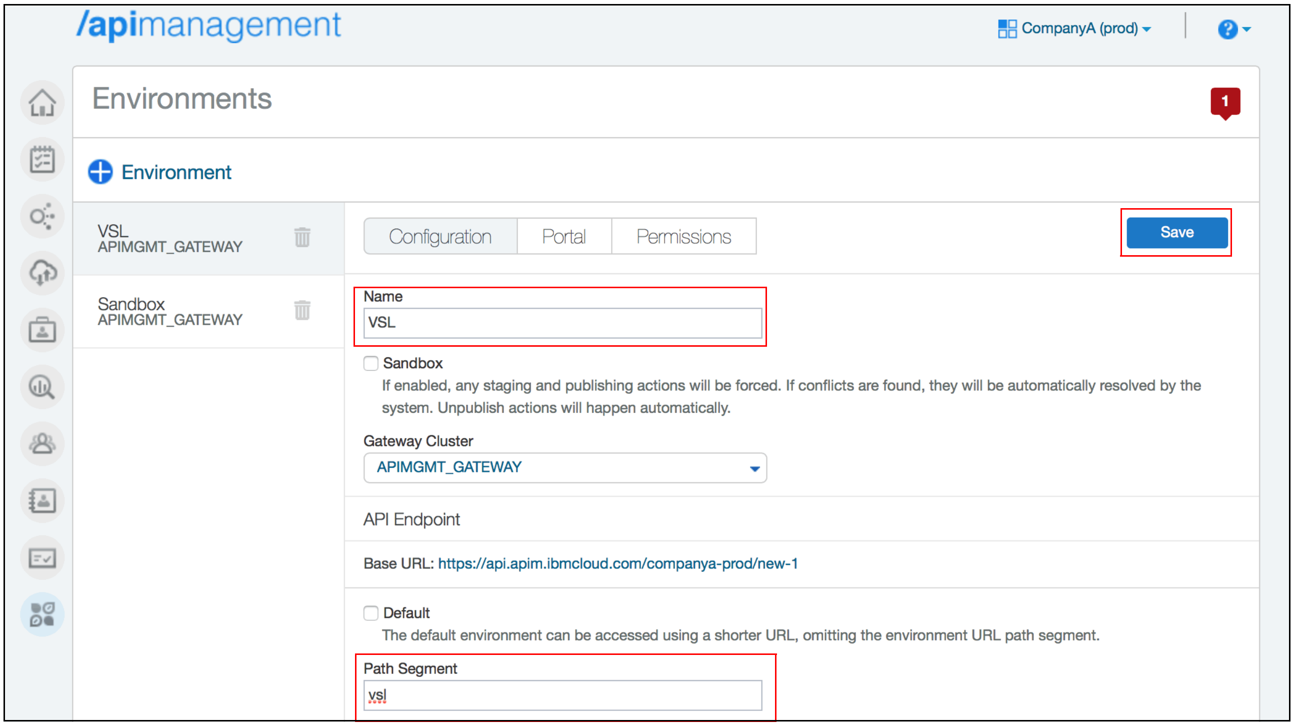


Figure 7-32 List of environments. A new environment should be created for this scenario

3. When the new environment is created, it can be updated. In this scenario, the default values for everything except the name and the path segment of the environment. When finished, click **Save** as shown in Figure 7-33.



VSL
APIMGMT_GATEWAY

Sandbox
APIMGMT_GATEWAY

Name
VSL

□ Sandbox
If enabled, any staging and publishing actions will be forced. If conflicts are found, they will be automatically resolved by the system. Unpublish actions will happen automatically.

Gateway Cluster
APIMGMT_GATEWAY

API Endpoint

Base URL: https://api.apim.ibmcloud.com/companya-prod/new-1

□ Default
The default environment can be accessed using a shorter URL, omitting the environment URL path segment.

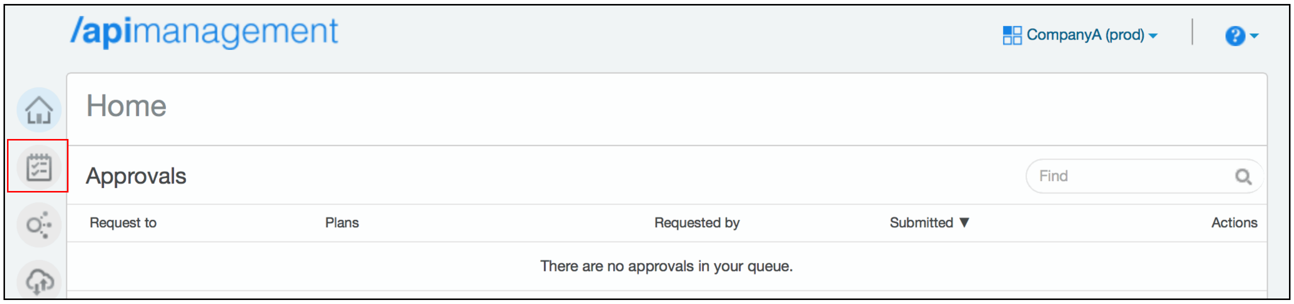
Path Segment
vsl

Figure 7-33 Modify the Name and Path Segment of the environment and click Save

Creating a plan to publish for the API

Plans are used for bundling a set of one or more APIs and used to determine how often an API can be used, for example, an API can be called only 20 times in 1 minute by a given user. This prevents the backend server from becoming overwhelmed with calls to our API.

1. In the API Manager console, select the **Plans** icon as shown in Figure 7-34.



Approvals

Request to	Plans	Requested by	Submitted ▼	Actions

There are no approvals in your queue.

Figure 7-34 Select the Plans icon to view the list of plans

2. There are two options for defining plans. In this scenario, a brand new plan is created. Click the **Add** icon to create a new plan as shown in Figure 7-35.

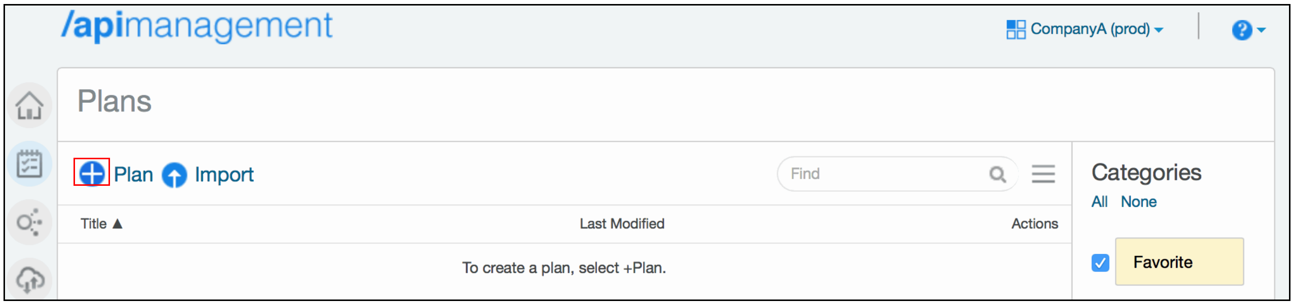
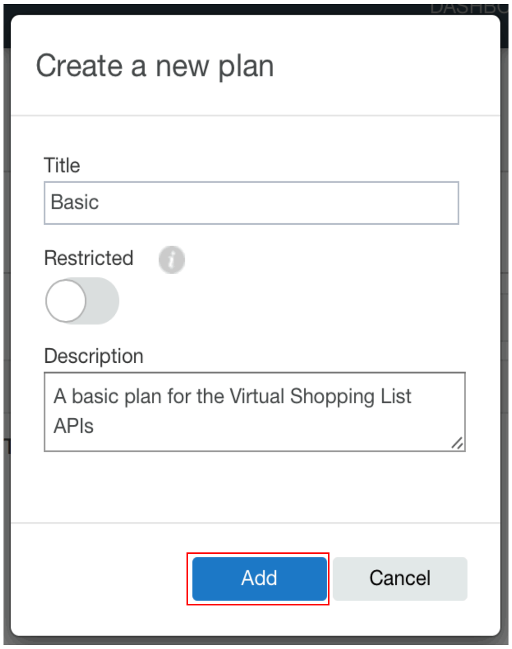


Figure 7-35 Plans list. There are no plans defined. Use the Add icon to create a new plan

3. A dialog is displayed. Enter the name and description of the new plan and click **Add** as shown in Figure 7-36. The plan can also be restricted by moving the slider across. Restricting a plan means that a developer needs approval from an administrator before using the plan. A plan can be restricted after it has been created as well.


Create a new plan

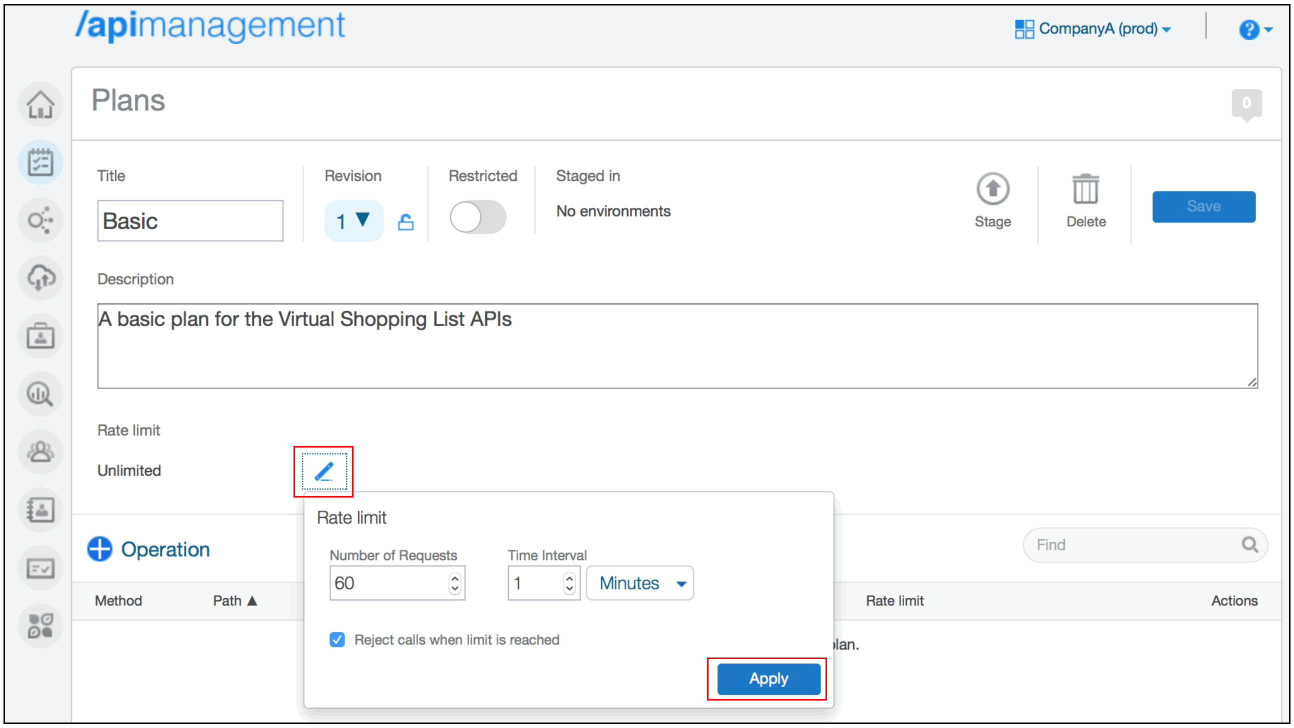
Title: Basic

Restricted

Description: A basic plan for the Virtual Shopping List APIs

Figure 7-36 Dialog for creating a new plan

4. Now that the plan is created, click the **Pencil** icon to update the plan to allow only 60 requests per minute, as shown in Figure 7-37. Click **Apply** to finish it.



Title: Basic
Revision: 1
Restricted: ○
Staged in: No environments

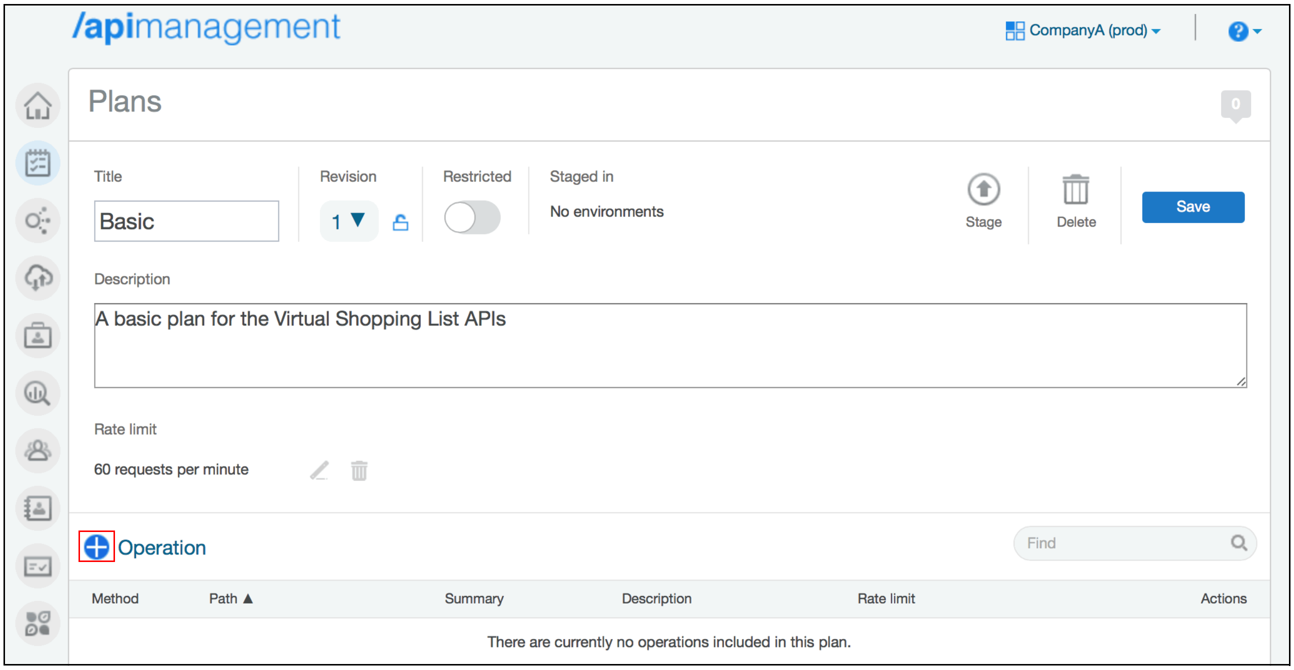
Description: A basic plan for the Virtual Shopping List APIs

Rate limit: Unlimited

Rate limit
Number of Requests: 60
Time Interval: 1 Minutes
☑ Reject calls when limit is reached

Figure 7-37 Update the plan to allow only 60 requests per minute

5. Next, the APIs that were previously imported into the API Manager should be added. This is done by pressing the **Add** icon, as shown in Figure 7-38.



Title: Basic
Revision: 1
Restricted: ○
Staged in: No environments

Description: A basic plan for the Virtual Shopping List APIs

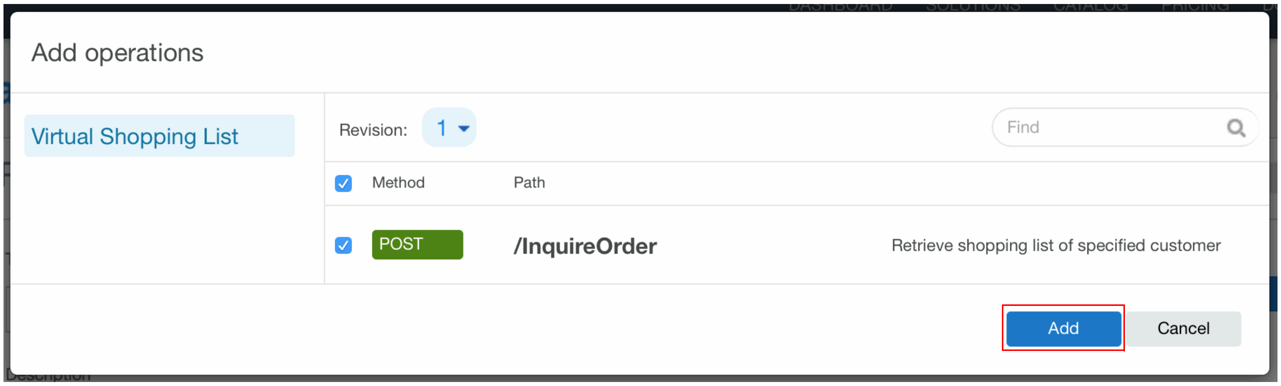
Rate limit: 60 requests per minute

Method	Path ▲	Summary	Description	Rate limit	Actions

There are currently no operations included in this plan.

Figure 7-38 Add the APIs to this plan

6. This opens a list of all the APIs to choose from as shown in Figure 7-39. Select all of the APIs to include in this plan and click **Add**.



Revision: 1

☑	Method	Path	
☑	POST	/InquireOrder	Retrieve shopping list of specified customer

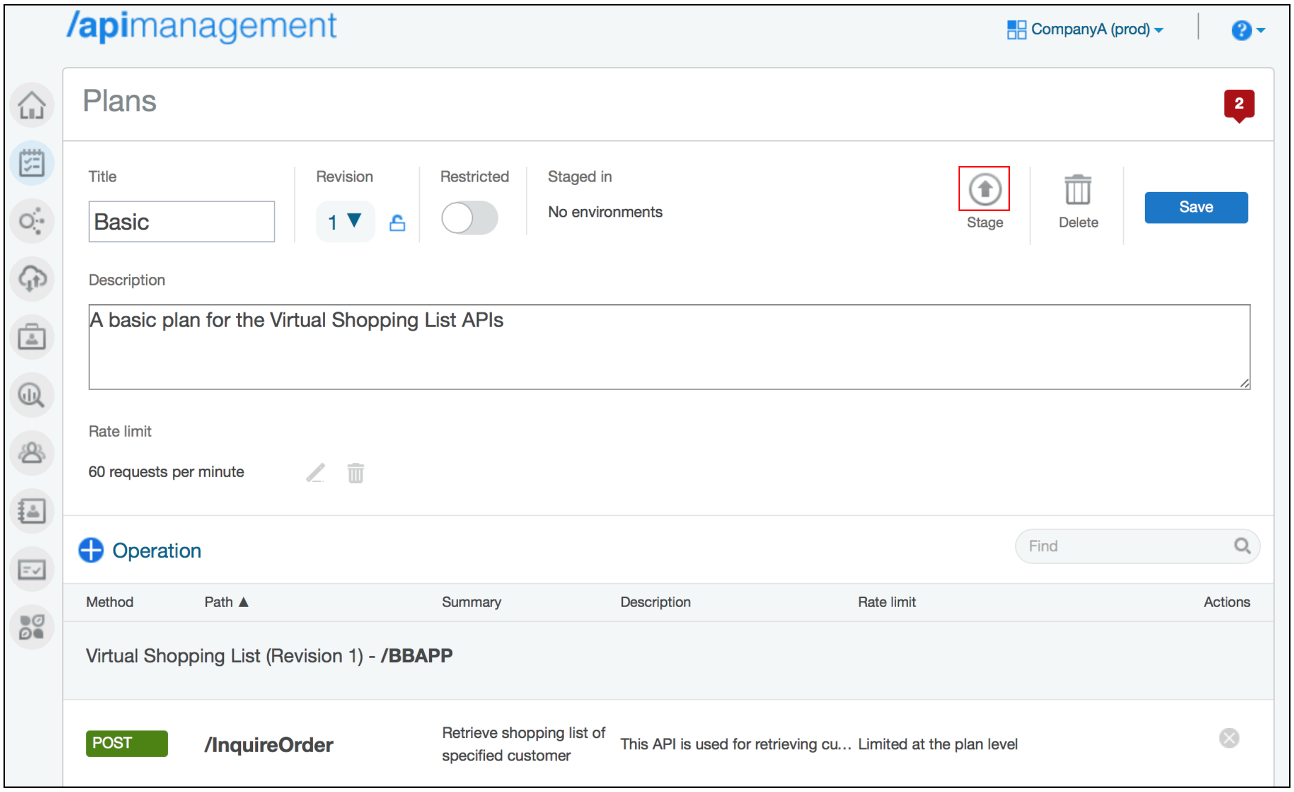
Figure 7-39 Select the APIs to be a part of this plan

7. After clicking **Add**, click **Save** to save the plan.

Staging and publishing the plan into an environment

Now that the plan is updated with the right policies and APIs, it can be staged and published:

1. In the details view of the new plan, click the **Stage** icon as shown in Figure 7-40. This opens the list of available environments. Choose the name of the environment that was created before.



Title: Basic

Revision: 1

Restricted

Staged in: No environments

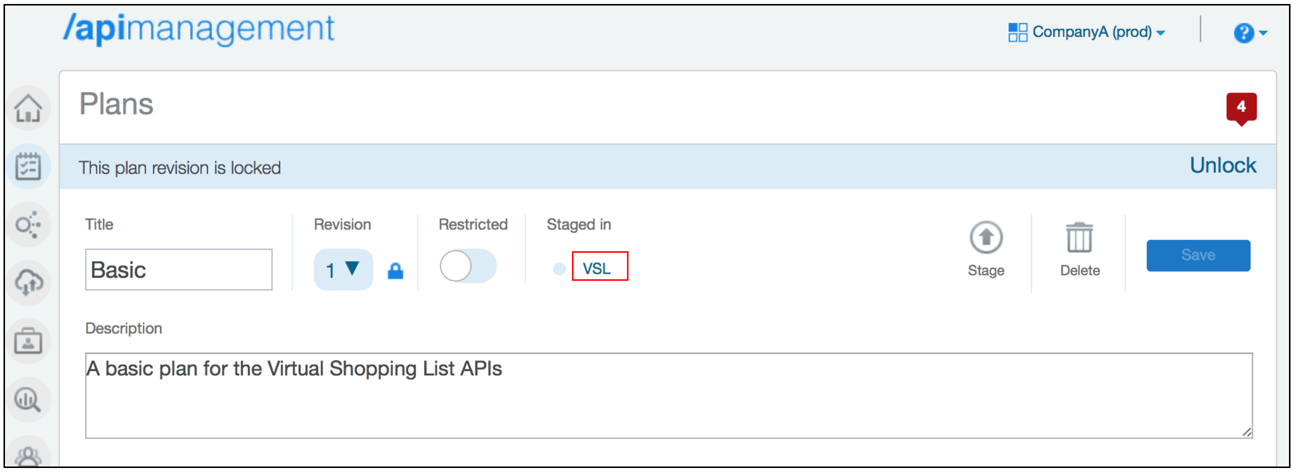
Description: A basic plan for the Virtual Shopping List APIs

Rate limit: 60 requests per minute

Method	Path ▲	Summary	Description	Rate limit	Actions
Virtual Shopping List (Revision 1) - /BBAPP					
POST	/InquireOrder	Retrieve shopping list of specified customer	This API is used for retrieving cu…	Limited at the plan level	

Figure 7-40 The Basic plan is ready to be staged to an environment

2. The plan should be staged and now it needs to be published. Click the name of the environment that the plan has been staged into to publish it, as shown in Figure 7-41.



This plan revision is locked

Title: Basic

Revision: 1

Restricted

Staged in: VSL

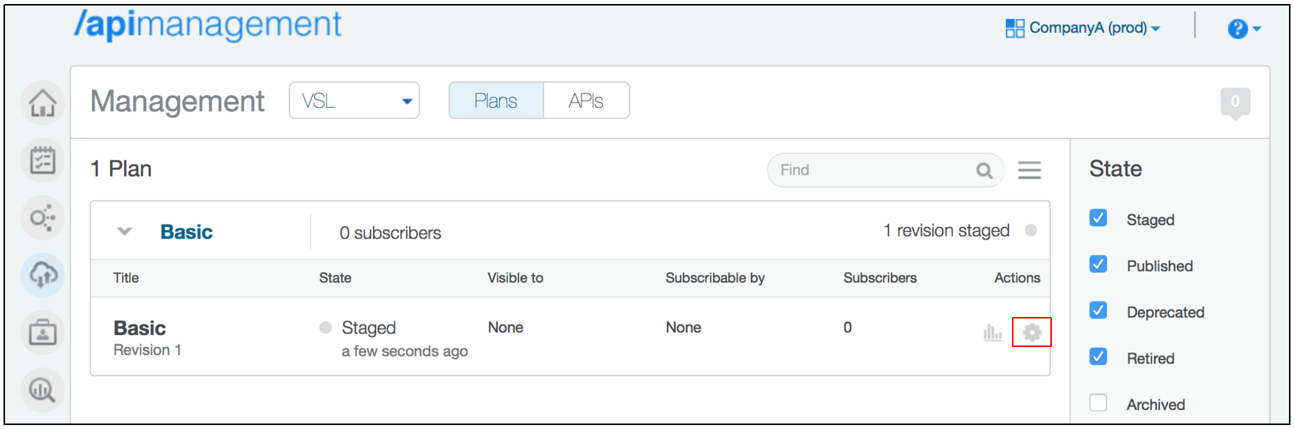
Description

A basic plan for the Virtual Shopping List APIs

Figure 7-41 The plan has been staged and is ready to be published

3. To publish the plan, select the **Gear** icon under Actions for the plan, as shown in Figure 7-42, and click **Publish**.

.



1 Plan

Basic 0 subscribers 1 revision staged

Title	State	Visible to	Subscribable by	Subscribers	Actions
Basic Revision 1	Staged a few seconds ago	None	None	0	

State

☑ Staged

☑ Published

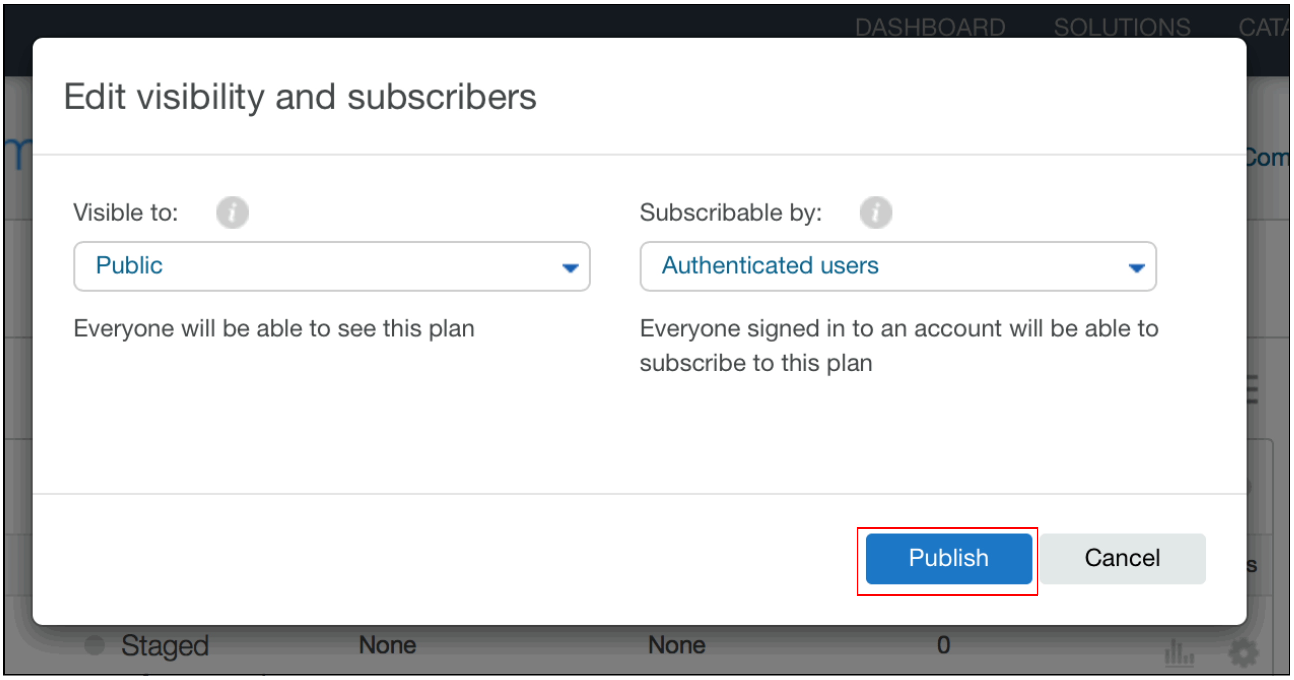
☑ Deprecated

☑ Retired

☐ Archived

Figure 7-42 The plan is ready to be published. Select the Gear icon to publish the plan

4. A dialog is displayed to select who is able to see this plan in the environment's developer portal, as shown in Figure 7-43. Click **Publish** to complete the process.



Edit visibility and subscribers

Visible to: Public

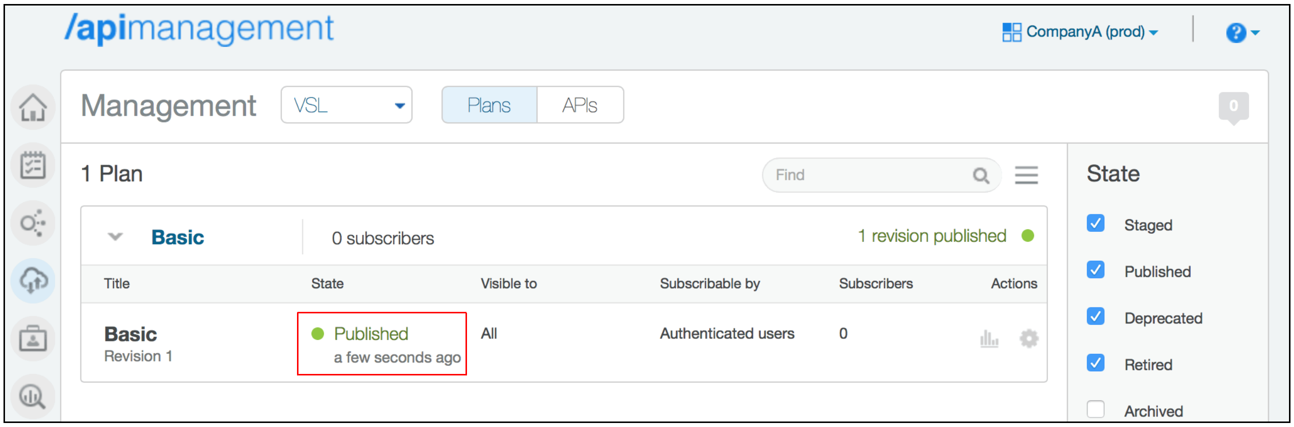
Everyone will be able to see this plan

Subscribable by: Authenticated users

Everyone signed in to an account will be able to subscribe to this plan

Figure 7-43 Determine which developers should have access in the developer portal

5. When the process is complete, we can see the plan has been published in the API Management console as shown in Figure 7-44. Now the plan is ready to be used in a Bluemix application.



1 Plan

Basic 0 subscribers 1 revision published ●

Title	State	Visible to	Subscribable by	Subscribers	Actions
Basic Revision 1	● Published a few seconds ago	All	Authenticated users	0	

State

☑ Staged
☑ Published
☑ Deprecated
☑ Retired
☐ Archived

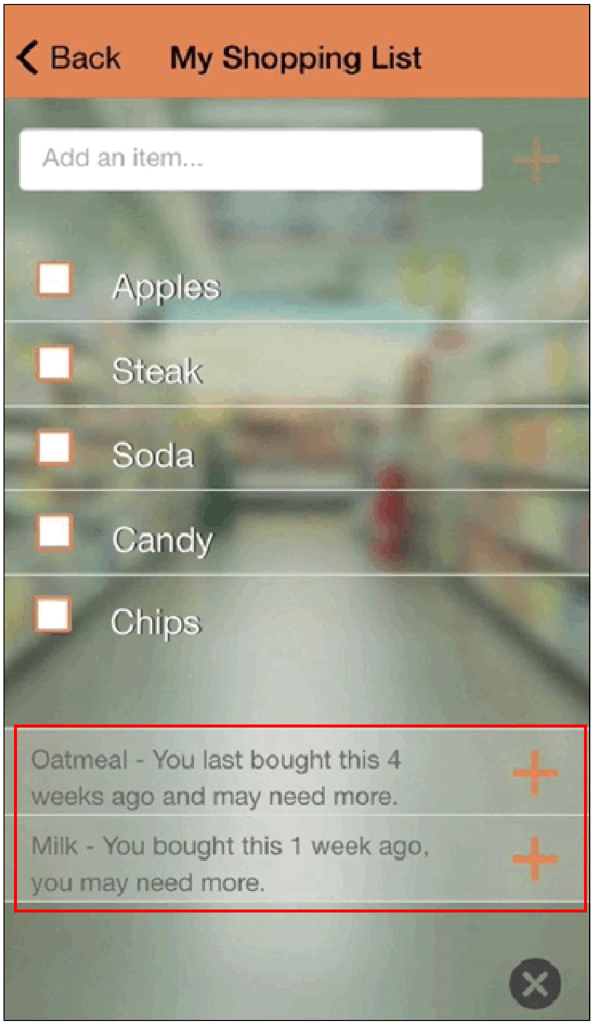
Figure 7-44 The plan is now published and ready to be used by Bluemix applications

7.6 Putting it all together

In this scenario, now that the API has been published to Bluemix, a new application can be created to consume this API, which has been published.

The application can be bound to the API, which has been created by the API Management service. A new token that is specific to the application is automatically created, which can be used by the application.

Figure 7-45 shows an image from their application.



□ Apples

□ Steak

□ Soda

□ Candy

□ Chips

Oatmeal - You last bought this 4 weeks ago and may need more.

Milk - You bought this 1 week ago, you may need more.

Figure 7-45   Shopping list with recommendations based on data from CICS on z/OS

CompanyB was able to build a new application leveraging their previous investments on z Systems to gain new insights into their customers and provide additional value to their customers.

They used z/OS Connect to transform their existing CICS transactions into an API, which was easily consumed by a mobile application. They were able to extend their datacenter into Bluemix by using the Secure Gateway service with a DataPower appliance to ensure a secure end-to-end connection.

Finally, by using Bluemix, CompanyB was also able to deliver their application quickly and it gave them the ability to respond quickly to fix issues in the application and the ability to easily roll out new features going forward.

8

Watson Analytics in hybrid cloud using Secure Gateway and DataWorks

This chapter shows how to ingest data from databases in cloud or on-premises servers into Watson Analytics service in IBM Cloud. The scenario described in this chapter takes selected data from the retailers transaction database to analyze the patterns of high selling products from its stores. This chapter is a hands-on tutorial that you can perform by using your computer and a Bluemix account. The tutorial documentation explains the steps in detail and is sufficient to understand the procedure without having to perform the implementation steps.

You learn how to use the Secure Gateway and DataWorks services in Bluemix, to create secure data transfer tunnel for Watson Analytics.

This chapter contains the following sections:

Recording of this scenario: You can find the recording of this scenario at the following link:

`https://youtu.be/2FZXYFEWUf4`

© Copyright IBM Corp. 2015. All rights reserved.

8.1 Solution overview

The solution created in this chapter shows how Watson Analytics service in IBM Cloud could be used for analyzing data in cloud or on-premises databases. As described in Section 1.3.7, “Retail example CompanyB background” on page 12, CompanyB is a local retailer and has its own private data center. The business requirement for CompanyB is to quickly analyze the high-selling products from their stores to enable their supply chain, without having their IT teams to spend time on setting their own analytics environment for this problem.

Business case for this scenario: For more information about the business case for this scenario and the company profile, see 1.3.10, “Solutions and actions by the CompanyB CIO” on page 13.

CompanyB uses an on-premises database as its secure data storage. Some data for this analytics is stored in transaction-related tables, which also contain other sensitive and confidential data. The objective is to set up a Watson Analytics environment with secure and limited visibility of their data.

You can perform this tutorial by using your computer and Bluemix account. This chapter simulates a database on-premises by using an IBM DB2 database. The techniques shown in this chapter can easily be applied to true production data hosted in cloud or on-premises databases.

Figure 8-1 on page 197 shows the architecture of the solution that will be created for CompanyB. The solution enables the data analyst to import data from an on-premises database into Watson Analytics. This integration is possible through the two services (Secure Gateway, DataWorks) in Bluemix. The Secure Gateway service provides a secure way to access cloud or on-premises data to applications running in over a secure passage that is the gateway. DataWorks is a service on Bluemix that can be used for data load, migration, refinement, transformation, and gaining insights from the data. Section 3.1.2, “API-centric service: Secure Gateway” on page 32 and section 3.1.4, “Data-centric service: IBM DataWorks” on page 43 in this book provide more details about these two services in Bluemix.

The database used in this chapter is IBM DB2 and hosts tables for storing customer transactions and other details. The Secure Gateway client is hosted on another machine that simulates a machine in a DMZ environment. The Secure Gateway client connects to Secure Gateway service in Bluemix to create a secure gateway tunnel for data transfer. After the tunnel is established, the DataWorks service can connect to the remote database and perform filters for the visibility of data to the Watson Analytics application. Now DataWorks can copy the data from an on-premises database to Watson Analytics service for further exploration and prediction analytics.

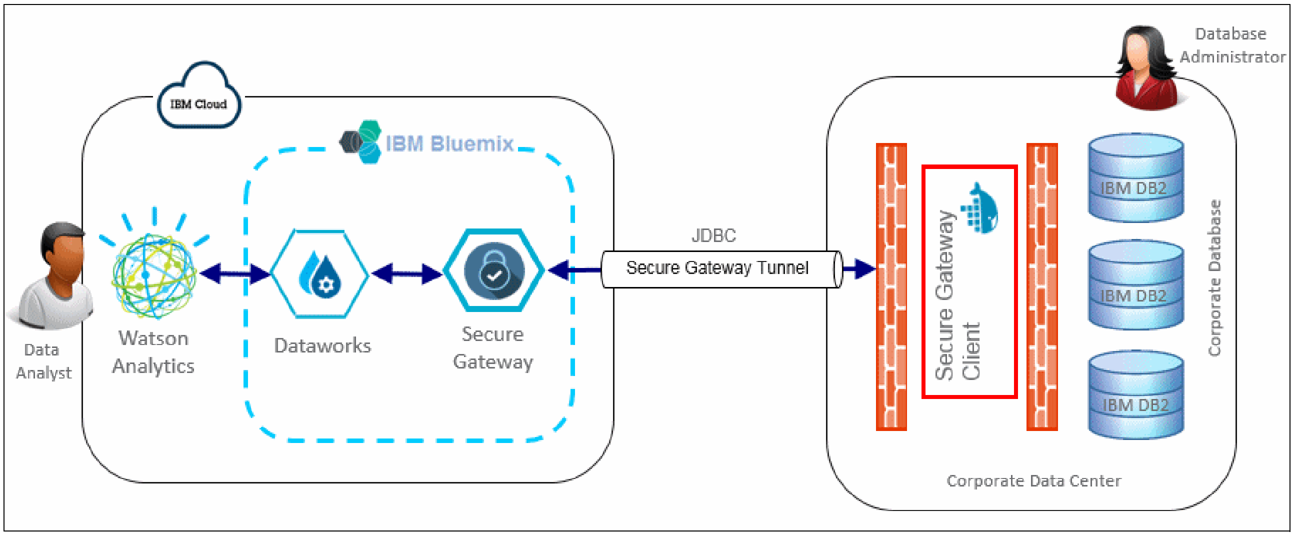


Figure 8-1 Architecture of the solution

8.2 Step-by-step implementation

This section explains the step-by-step procedure to create a solution for this problem. The following steps will be executed to set up this hybrid cloud scenario for Watson Analytics:

- **IBM DB2 setup**: This step explains creation of tables in the IBM DB2 database, understanding the sample data, and uploading the sample data in the table created by using a sample Java program. Refer to section 8.2.1, “Setting the IBM DB2 database for on-premises data” on page 198.
- **Secure Gateway client setup**: This section explains the installation of Docker and Secure Gateway client. Refer to section 8.2.2, “Setting the Secure Gateway client in the on-premises environment” on page 201.
- **Secure Gateway setup**: This section explains the creation and configuration of Secure Gateway service in Bluemix. Refer to section 8.2.3, “Setting Secure Gateway service in Bluemix” on page 202.
- **DataWorks setup**: This section explains the creation and configuration of DataWorks service in Bluemix. Refer to section 8.2.4, “Setting DataWorks service in Bluemix” on page 205.
- **WatsonAnalytic setup**: This section explains the registration and initial configuration with Watson Analytics service in IBM Cloud. Refer to section 8.2.5, “Create a Watson Analytics user in IBM Cloud” on page 206.

Note: You can download the .CSV file and Java program used in this scenario from the ITSO Redbooks FTP site. Refer to section “Locating the web material” on page 241 for downloading instructions.

8.2.1 Setting the IBM DB2 database for on-premises data

This step assumes that an IBM DB2 database is already installed and is configured to listen on a specific TCP port. We proceed by creating a database ITSODB for the retailer. Example 8-1 shows the command to create a database.

Example 8-1 Create database ITSODB

```
>db2 create db ITSODB
DB20000I  The CREATE DATABASE command completed successfully.
```

The database is successfully created and the next step is to create a table TRANSACTIONVIEW. This table stores data described in Table 8-1. The purpose of the table is to store customer purchases with transaction, product, and customer details.

Table 8-1 Data to be stored in the table TRANSACTIONVIEW

Name	Data type	Explanation
TRANSACTIONID	VARCHAR(9)	This column stores the transaction ID.
CUSTOMERID	VARCHAR(12)	This column stores the customer ID.
TRANSACTIONDATE	DATE	This column stores the transaction date.
PRODUCTID	VARCHAR(9)	This column store the product ID.
PRODUCTDESCRIPTION	VARCHAR(30)	This column stores the product description.
PRODUCTPRICE	INTEGER	This column stores the product selling price.
CUSTOMERNAME	VARCHAR(50)	This column stores the customer name.
CUSTOMEREMAIL	VARCHAR(70)	This column stores the customer email address.

Example 8-2 shows the commands to create table TRANSACTIONVIEW from the DB2 command shell.

Example 8-2 Create table TRANSACTIONVIEW

```
>db2 connect to ITSODB

   Database Connection Information

 Database server        = DB2/NT64 10.5.3
 SQL authorization ID   = DB2ADMIN
 Local database alias   = ITSODB

>db2

db2 => CREATE TABLE TRANSACTIONVIEW ( TRANSACTIONID VARCHAR(9), CUSTOMERID
VARCHAR(12), TRANSACTIONDATE DATE, PRODUCTID VARCHAR(9), PRODUCTDESCRIPTION
VARCHAR(30), PRODUCTPRICE INTEGER, CUSTOMERNAME VARCHAR(50), CUSTOMEREMAIL
VARCHAR(70))
DB20000I  The SQL command completed successfully.

db2 => describe table TRANSACTIONVIEW
```

```
                      Data type                     Column
Column name           schema    Data type name      Length     Scale Nulls
-------------------   --------- ------------------- ---------- ----- ------
TRANSACTIONID         SYSIBM    VARCHAR                      9     0 Yes
CUSTOMERID            SYSIBM    VARCHAR                     12     0 Yes
TRANSACTIONDATE       SYSIBM    DATE                         4     0 Yes
PRODUCTID             SYSIBM    VARCHAR                      9     0 Yes
PRODUCTDESCRIPTION    SYSIBM    VARCHAR                     30     0 Yes
PRODUCTPRICE          SYSIBM    INTEGER                      4     0 Yes
CUSTOMERNAME          SYSIBM    VARCHAR                     50     0 Yes
CUSTOMEREMAIL         SYSIBM    VARCHAR                     70     0 Yes
  8 record(s) selected.
```

The sample data is stored in the `TransactionView.csv` file. The sample data from the CSV file is processed by using a stand-alone Java program that is shown in Example 8-3, which inserts this data into the TRANSACTIONVIEW table using JDBC. Both the CSV file and Java program are made available on the ITSO FTP site. Refer to “Locating the web material” on page 241 for downloading instructions.

Example 8-3 Stand-alone Java program to insert data in ITSODB

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class TransactionView {

public static void main(String[] args) {

String serverName   = "0.0.0.0";
String serverPort   = "50000";
String databaseName = "ITSODB";
String userID       = "********";
String password     = "********";
String query        = null;
String url          = null;
Connection con      = null;
int counter         = 0;
String csvFile      = "TransactionView.csv";
BufferedReader br   = null;
String line         = "";
String cvsSplitBy   = ",";

System.out.println("********************************************************");
System.out.println("> INSERTING RECORDS INTO TABLE TRANSACTIONVIEW OF ITSODB");

try
{

Class.forName("com.ibm.db2.jcc.DB2Driver").newInstance();
url = "jdbc:db2://" + serverName + ":" + serverPort + "/"+databaseName;
con = DriverManager.getConnection(url,userID,password);
br = new BufferedReader(new FileReader(csvFile));

while ((line = br.readLine()) != null) {

String[] tx = line.split(cvsSplitBy);

query = "INSERT INTO TRANSACTIONVIEW (TRANSACTIONID, CUSTOMERID, "+
        "TRANSACTIONDATE, PRODUCTID, PRODUCTDESCRIPTION, PRODUCTPRICE,"+
        "CUSTOMERNAME, CUSTOMEREMAIL) VALUES "+ "("+"\'"+tx[0]+"\',"+
        "\'"+tx[1]+"\',"+"\'"+tx[2]+"\'"+",\'"+tx[3]+"\','"+tx[4]+"\',"+
        tx[5]+",\'"+tx[6]+"\',"+"\'"+tx[7]+"\'"+")";

Statement stmt = con.createStatement();
stmt.executeUpdate(query);
counter++;

}
}catch(Exception e){
e.printStackTrace();
}
finally {
try {
con.close();
} catch (SQLException e) {
e.printStackTrace();
}
}
System.out.println("> TOTAL OF"+counter+" RECORDS INSERTED INTO TRANSACTIONVIEW");
System.out.println("**********************************************************");

}
}
```

This Java program could be executed from the command line and produces the following results on the standard output console. Example 8-4 shows the execution of the Java program from command line.

Example 8-4 Execution of Java program from command line

```
> java TransactionView
*****************************************************************
> INSERTING RECORDS INTO TABLE TRANSACTIONVIEW OF ITSODB
> TOTAL OF 100000 RECORDS INSERTED INTO TRANSACTIONVIEW
*****************************************************************
```

Note: Ensure that the DB2 client JAR files are in CLASSPATH while executing this Java program.

Now the data is ready in the on-premises database and could be accessed by configuring the Docker client for Secure Gateway.

8.2.2 Setting the Secure Gateway client in the on-premises environment

Secure Gateway service in Bluemix requires a client to be installed in the on-premises environment, and the simplest client implementation to install is one that is already installed in a Docker client (Docker client for Secure Gateway). The on-premises server needs the Docker run time installed, and later the Secure Gateway client needs to be pulled into Docker run time. To install Docker, depending on the operating system, refer to the documentation in the following websites:

- “Installing Docker” explains how to install Docker:

  https://docs.docker.com/installation/

- Bluemix also documents installing the Docker run time. See “Services: Secure Gateway: Docker” in the Bluemix documentation:

  https://www.ng.bluemix.net/docs/services/SecureGateway/sg_021.html#sg_003

When Docker run time is installed, the Secure Gateway client should be pulled in the Docker run time. Example 8-5 shows the installation of the Secure Gateway client in Docker run time.

Example 8-5 Pull Secure Gateway client in Docker run time

```
$ docker pull ibmcom/secure-gateway-client
latest: Pulling from ibmcom/secure-gateway-client
0ec088cf2a05: Pull complete
798be516e56c: Pull complete
3520f279093f: Pull complete
dc8cb9033694: Pull complete
e19abe4bb3c4: Pull complete
427fb8b33dae: Pull complete
e791321f459d: Already exists
Digest: sha256:acbdaa37538bba312429a1443dfb153d84e95db5b4d02a59388cbb362b0b06de
Status: Downloaded newer image for ibmcom/secure-gateway-client:latest
```

Now the Secure Gateway client is set up in the on-premises environment and the next step is to set up Secure Gateway service in Bluemix.

8.2.3 Setting Secure Gateway service in Bluemix

The Secure Gateway service provides a secure way to access cloud or on-premises data from Bluemix application through a secure passage. To create an instance of Secure Gateway service, log in to the Bluemix dashboard:

1. In the Bluemix dashboard, select your space. Figure 8-2 shows the selection of Bluemix space **hybrid-cloud**.

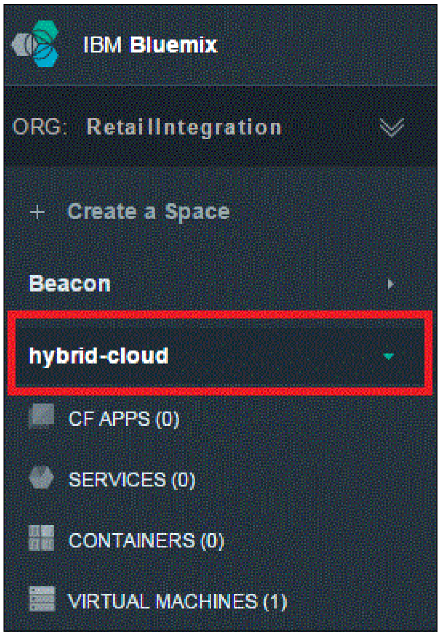


Figure 8-2 Select the space in Bluemix dashboard

2. In the main view of the space, select **Use Services or APIs**.
3. In the Service catalog, select Secure Gateway service, as shown in Figure 8-3.

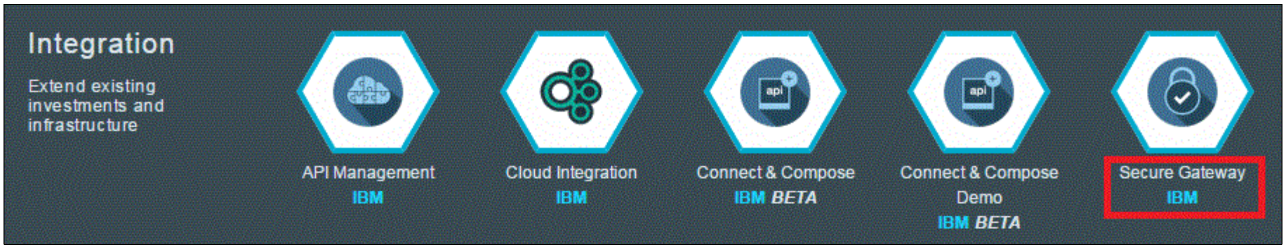
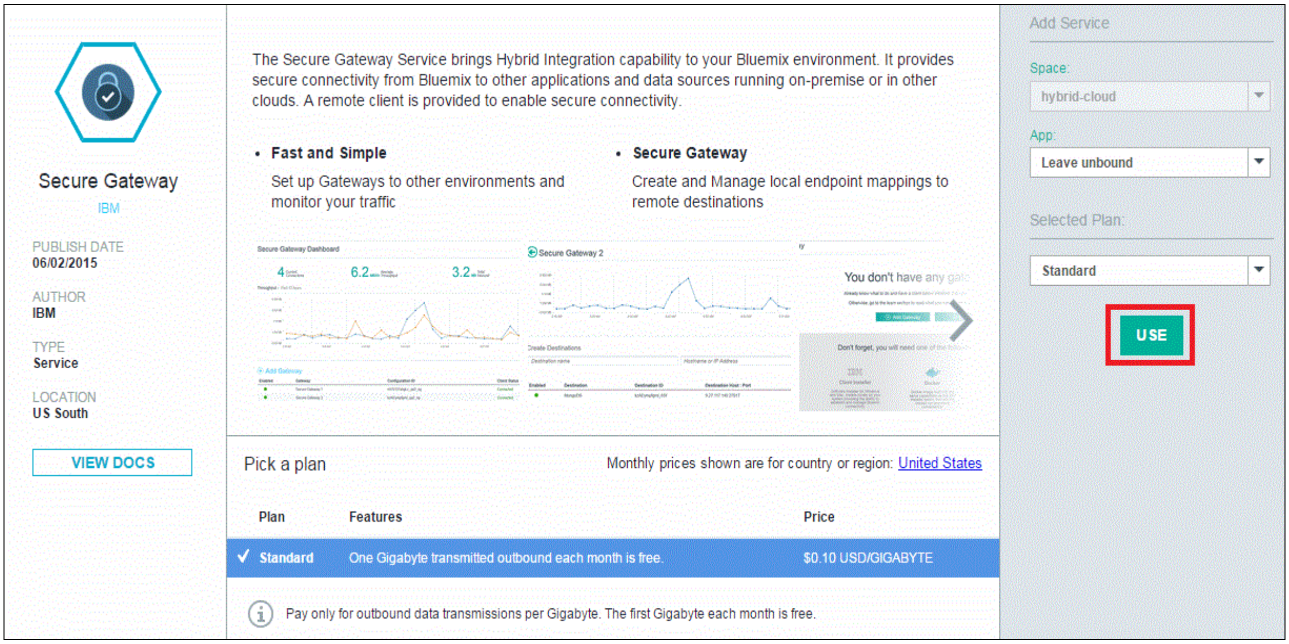


Figure 8-3 Secure Gateway service in Bluemix dashboard

4. Create an instance of Secure Gateway using the default setting and leaving it unbound, as shown in Figure 8-4.



Secure Gateway

IBM

PUBLISH DATE
06/02/2015

AUTHOR
IBM

TYPE
Service

LOCATION
US South

VIEW DOCS

The Secure Gateway Service brings Hybrid Integration capability to your Bluemix environment. It provides secure connectivity from Bluemix to other applications and data sources running on-premise or in other clouds. A remote client is provided to enable secure connectivity.

- **Fast and Simple**
  Set up Gateways to other environments and monitor your traffic
- **Secure Gateway**
  Create and Manage local endpoint mappings to remote destinations

Pick a plan

Monthly prices shown are for country or region: United States

Plan	Features	Price
✓ Standard	One Gigabyte transmitted outbound each month is free.	$0.10 USD/GIGABYTE

Pay only for outbound data transmissions per Gigabyte. The first Gigabyte each month is free.

Add Service

Space: hybrid-cloud

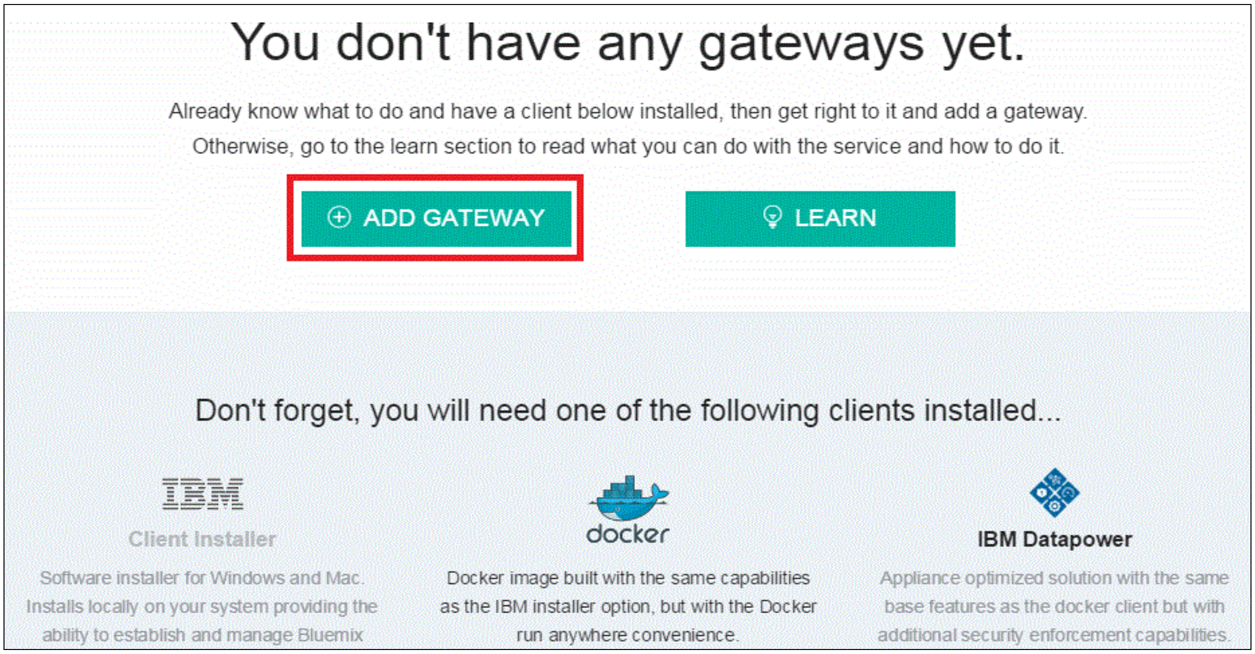
App: Leave unbound

Selected Plan: Standard

USE

Figure 8-4   Create an instance of Secure Gateway service in Bluemix

5. Rename the Secure Gateway service to **Retail DB Secure Gateway**.
6. In the dashboard, click the Retail DB Secure Gateway, which then takes you to the service landing page of Secure Gateway.
7. Click **ADD GATEWAY** to add a new gateway in this instance, as shown in Figure 8-5.



You don't have any gateways yet.

Already know what to do and have a client below installed, then get right to it and add a gateway.

Otherwise, go to the learn section to read what you can do with the service and how to do it.

ADD GATEWAY   LEARN

Don't forget, you will need one of the following clients installed...

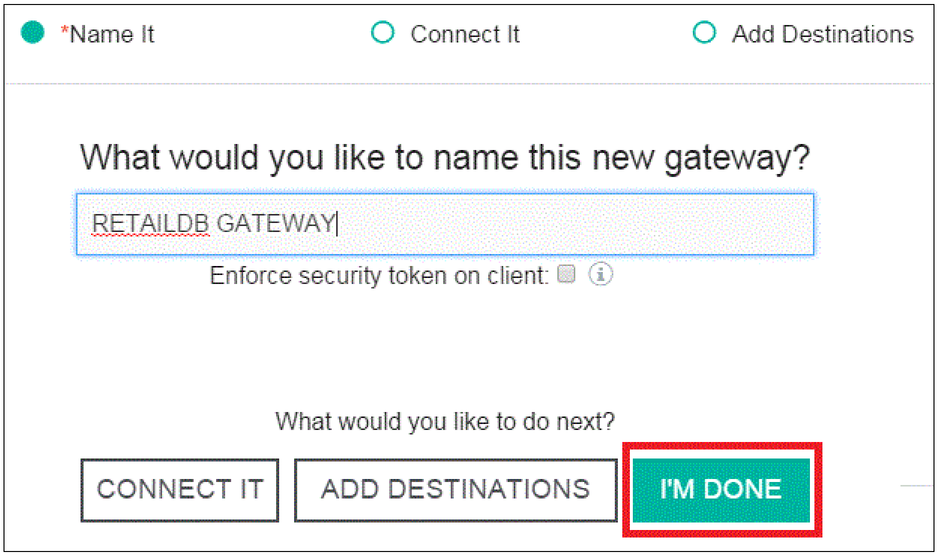
IBM Client Installer: Software installer for Windows and Mac. Installs locally on your system providing the ability to establish and manage Bluemix

docker: Docker image built with the same capabilities as the IBM installer option, but with the Docker run anywhere convenience.

IBM Datapower: Appliance optimized solution with the same base features as the docker client but with additional security enforcement capabilities.

Figure 8-5   Add a gateway in Retail DB Secure Gateway service

8. In the **Add Gateway** page, add a short description about the gateway and click **I'M DONE**, as shown in Figure 8-6.



● *Name It ○ Connect It ○ Add Destinations

What would you like to name this new gateway?

RETAILDB GATEWAY

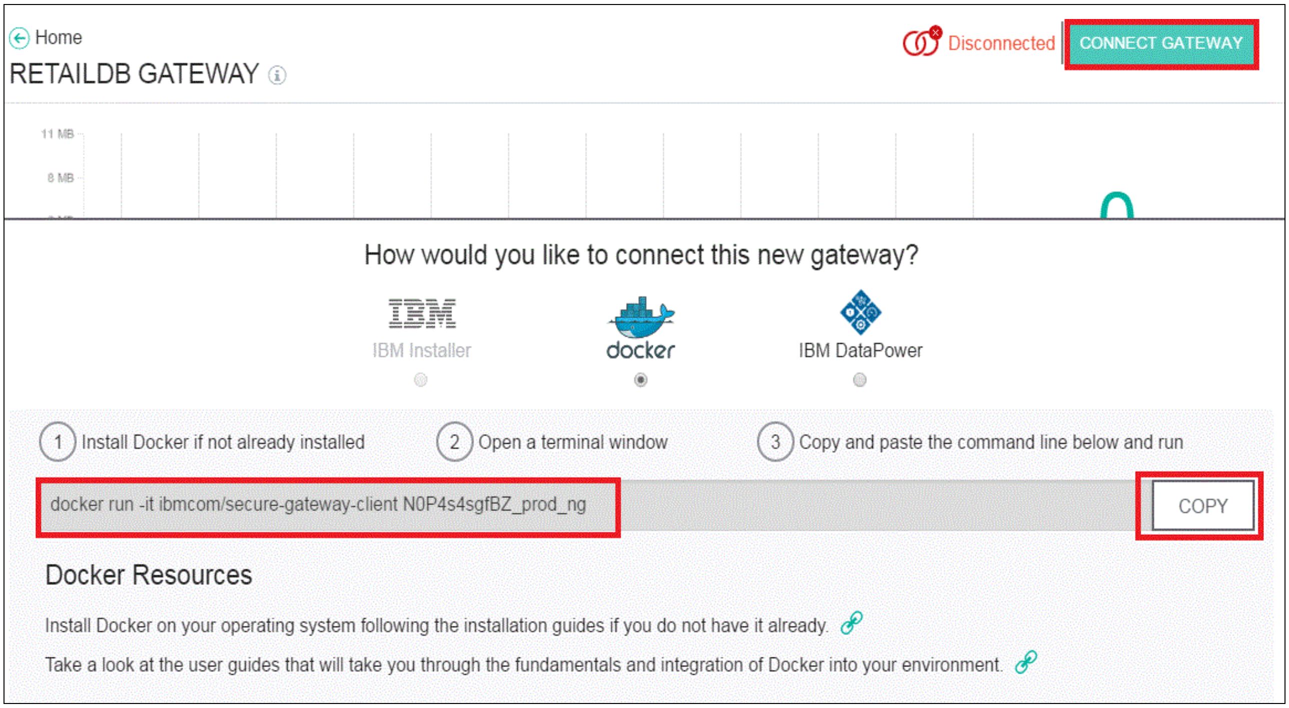
Enforce security token on client: □

What would you like to do next?

Figure 8-6 Complete gateway description

9. Click **RETAILDB GATEWAY** in the Secure Gateway dashboard and its status should show disconnected because none of the Secure Gateway clients are connected yet.

10. Click **Connect Gateway** to capture the command for executing the Secure Gateway client in Docker. Figure 8-7 shows how to capture the command.



How would you like to connect this new gateway?

1 Install Docker if not already installed

2 Open a terminal window

3 Copy and paste the command line below and run

```
docker run -it ibmcom/secure-gateway-client N0P4s4sgfBZ_prod_ng
```

Docker Resources

Install Docker on your operating system following the installation guides if you do not have it already.

Take a look at the user guides that will take you through the fundamentals and integration of Docker into your environment.

Figure 8-7 Command to run the Secure Gateway Docker client

11.Keep a reference of the following command because it is required to start the Secure Gateway Docker clients later.

> **Note:** docker run -it ibmcom/secure-gateway-client N0P4s4sgfBZ_prod_ng

Now the Secure Gateway service is created in Bluemix and the next step is to create an instance of DataWorks service.

> **Note:** A destination is not added in this gateway because the DataWorks service automatically creates a destination when trying to connect to the database in the on-premises server.

Also, refer to "*Services: Secure Gateway: Creating a Secure Gateway by using the Bluemix UI*" in the Bluemix documentation:

https://www.ng.bluemix.net/docs/services/SecureGateway/sg_022.html#sg_009

The next step is to create an instance of DataWorks service in Bluemix.

8.2.4 Setting DataWorks service in Bluemix

DataWorks in Bluemix equips users with trustworthy, useful, and relevant data. DataWorks Forge enables users to find data, shape it, and deliver it to applications and systems. To create an instance of DataWorks, log in to the Bluemix dashboard:

1. In Bluemix dashboard, select your space **hybrid-cloud**. See Figure 8-2 on page 202.
2. In the main view of the space, select **Use Services or APIs**.
3. In the Service catalog, select **DataWorks** service as shown in Figure 8-8.

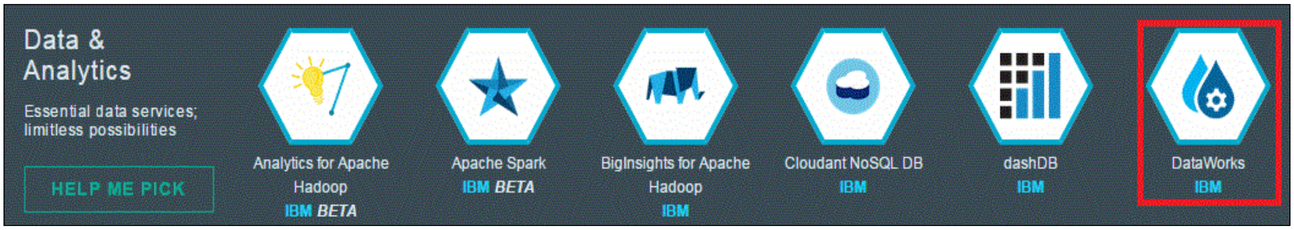
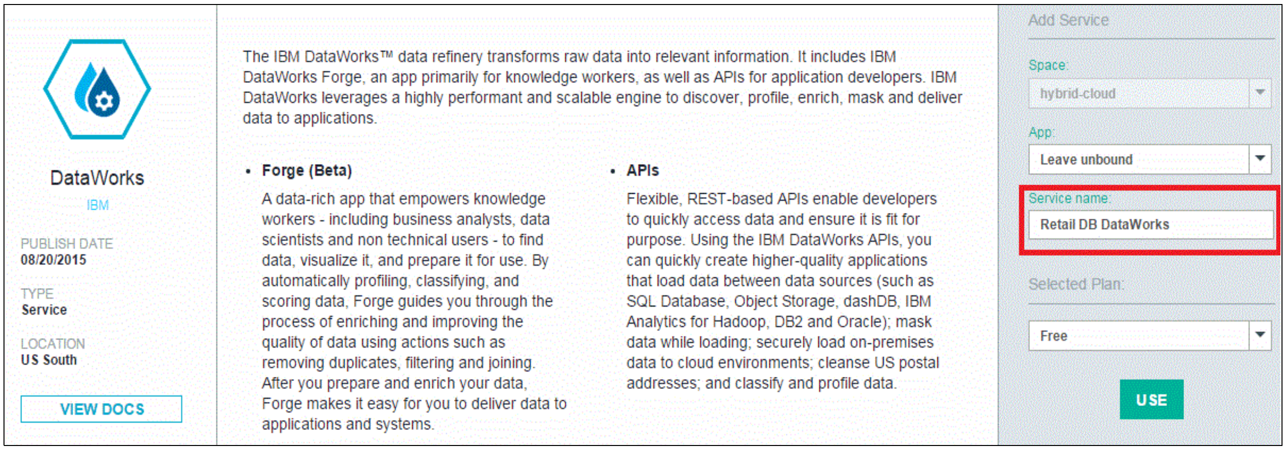

Data & Analytics
Essential data services; limitless possibilities
HELP ME PICK
Analytics for Apache Hadoop
IBM BETA
Apache Spark
IBM BETA
BigInsights for Apache Hadoop
IBM
Cloudant NoSQL DB
IBM
dashDB
IBM
DataWorks
IBM

Figure 8-8 DataWorks service in Bluemix dashboard

4. Create an instance of DataWorks by leaving the service unbound, and providing it a name **Retail DB DataWorks** as shown in Figure 8-9.



DataWorks

IBM

PUBLISH DATE
08/20/2015

TYPE
Service

LOCATION
US South

VIEW DOCS

The IBM DataWorks™ data refinery transforms raw data into relevant information. It includes IBM DataWorks Forge, an app primarily for knowledge workers, as well as APIs for application developers. IBM DataWorks leverages a highly performant and scalable engine to discover, profile, enrich, mask and deliver data to applications.

- **Forge (Beta)**

  A data-rich app that empowers knowledge workers - including business analysts, data scientists and non technical users - to find data, visualize it, and prepare it for use. By automatically profiling, classifying, and scoring data, Forge guides you through the process of enriching and improving the quality of data using actions such as removing duplicates, filtering and joining. After you prepare and enrich your data, Forge makes it easy for you to deliver data to applications and systems.

- **APIs**

  Flexible, REST-based APIs enable developers to quickly access data and ensure it is fit for purpose. Using the IBM DataWorks APIs, you can quickly create higher-quality applications that load data between data sources (such as SQL Database, Object Storage, dashDB, IBM Analytics for Hadoop, DB2 and Oracle); mask data while loading; securely load on-premises data to cloud environments; cleanse US postal addresses; and classify and profile data.

Add Service

Space: hybrid-cloud

App: Leave unbound

Service name: Retail DB DataWorks

Selected Plan: Free

USE

Figure 8-9 Create an instance of DataWorks service in Bluemix

5. Click **Retail DB DataWorks** service and it takes you to the landing page of DataWorks service.
6. The DataWorks Forge service is used for exporting data into Watson Analytics in section 8.5, “Exploring data in Watson Analytics” on page 215.

Also, refer to “*Services: DataWorks: Adding a DataWorks service to your application*” in the Bluemix documentation:

https://www.ng.bluemix.net/docs/services/dataworks1/index.html#t_adding_dataworks_service

The next step is to configure Watson Analytics service in IBM Cloud.

8.2.5 Create a Watson Analytics user in IBM Cloud

Watson Analytics in IBM Cloud is a breakthrough cloud analytics service for business users. Watson Analytics allows users to make business decisions with confidence by using data exploration predictive analytics features. It allows users to explore data with natural language. To get started with this analytic solution, you need to register for this service in IBM Cloud.

You can register for Watson Analytics service by using the following website:

https://www.ibm.com/ibmid/basic_register/register.html?watsonanalytics

When registered, use the following Watson Analytics website to access the service:

http://www.ibm.com/analytics/watson-analytics

> **Note:** This chapter uses the no-cost account of Watson Analytics. However, you can upgrade at any time based on the features and data to be analyzed. More details about Watson Analytics features and pricing model can be seen at the following website:
>
> https://www.ibm.com/marketplace/cloud/watson-analytics/us/en-us?trialId=500648376&cm_sp=ICE-_-WatsonAnalytics%20Trial-_-Buy

Now the setup for various components is complete. The next section explains how to start Secure Gateway client to enable data transfer from the on-premises database to Watson Analytics service in IBM Cloud.

8.3 Starting Secure Gateway client

The Secure Gateway service in Bluemix needs a Secure Gateway client to be running in the on-premises host. The Secure Gateway client hosted on an on-premises machine should have access to the on-premises database server and to Secure Gateway services hosted in Bluemix. Typically, this means that the Secure Gateway client should be installed on a machine that is connected to the same network segment (subnet or VLAN) as the host of the database and that is connected to the network connection having access to Bluemix. A good architecture should host the Secure Gateway client machine in a DMZ environment.

The Secure Gateway client can run in a Docker container or in IBM DataPower. This chapter uses Secure Gateway client in a Docker container. The machine simulating on-premises server in DMZ already has a Docker run time installed and the Secure Gateway client pulled into it. Refer to 8.2.2, “Setting the Secure Gateway client in the on-premises environment” on page 201 for more details.

Section 8.2.3, “Setting Secure Gateway service in Bluemix” on page 202 explained how to create an instance of Secure Gateway service in Bluemix. Step 10 on page 204 showed how to capture the command for starting the Secure Gateway client in Docker. The command shown in Example 8-6 is used to start this client from the on-premises host. This command contains the Secure Gateway ID, which tells the Secure Gateway service in Bluemix on which specific Secure Gateway service instance the client is trying to connect.

Example 8-6 Command to start the Secure Gateway client

```
docker run -it ibmcom/secure-gateway-client N0P4s4sgfBZ_prod_ng
```

Example 8-7 shows the output of the command shown in Example 8-6. This output shows that the Secure Gateway client is successfully connected to Secure Gateway instance N0P4s4sgfBZ_prod_ng.

Example 8-7 Secure Gateway client connected to service in Bluemix

```
$ docker run -it ibmcom/secure-gateway-client N0P4s4sgfBZ_prod_ng
IBM Bluemix Secure Gateway Client Version 1.2.2
************************************************************************************************
You  are running the  IBM Secure  Gateway Client for Bluemix. When you enter the provided docker
command the IBM Secure Gateway Client  for Bluemix automatically downloads as a Docker image and
is executed on your system/device. This is released under an IBM license. The  license agreement
for IBM  Secure Gateway Client for Bluemix is available at the following location:

http://www.ibm.com/software/sla/sladb.nsf/lilookup/986C7686F22D4D3585257E13004EA6CB?OpenDocument

Your use of the components of the package and  dependencies constitutes your acceptance  of this
license agreement. If you do  not want to accept the license, immediately quit  the container by
closing the  terminal  window or by  entering 'quit' followed by the ENTER key. Then, delete any
pulled Docker image from your device.
************************************************************************************************
>
[2015-08-28 20:06:26.843] [INFO] The Secure Gateway tunnel is connected
```

The command in Example 8-8 shows the Docker container is running in Docker run time.

Example 8-8 Secure Gateway client running in Docker run time

```
$ docker ps
CONTAINER ID       IMAGE                           STATUS          NAMES
91eb2d2ae546       ibmcom/secure-gateway-client    Up 2 minutes    reverent_feynman
```

Figure 8-10 shows the Secure Gateway Dashboard and the client connectivity status on **RETAILDB GATEWAY** changed from red to green.

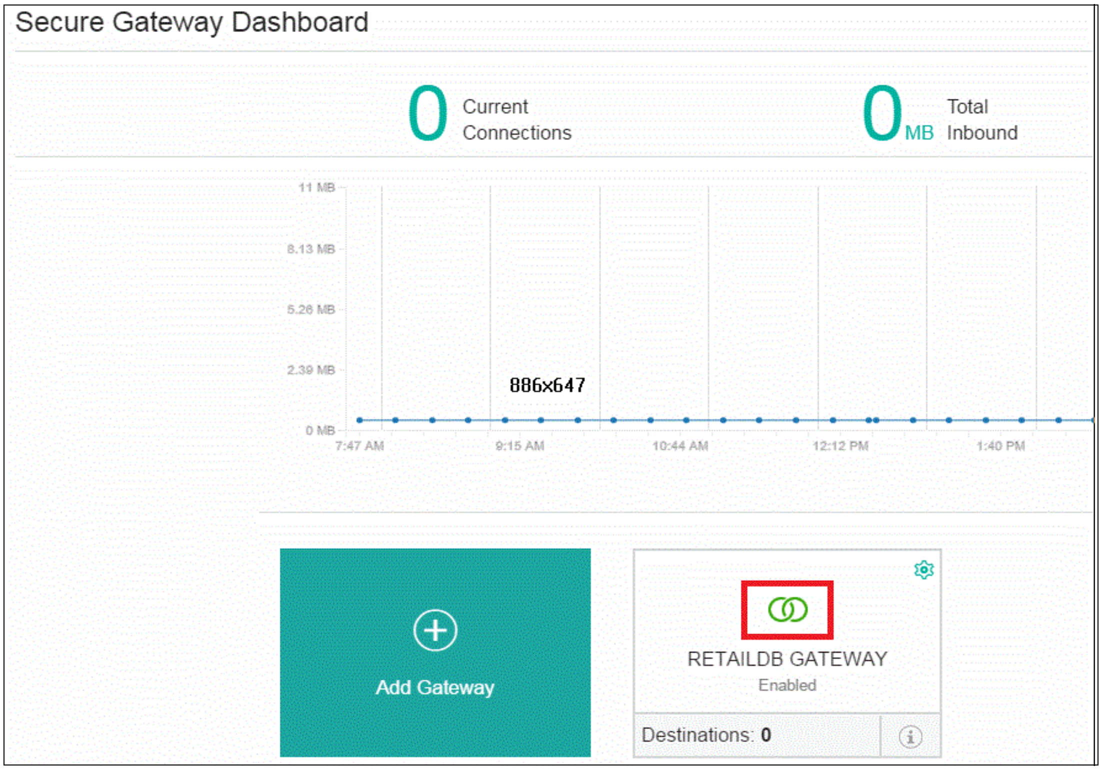


Figure 8-10 Secure Gateway client status in Secure Gateway Dashboard

Now we have the Secure Gateway client connected and the secure tunnel is established between the on-premises host and Bluemix. This system is now ready for transfer of data for Watson Analytics, however, before that, DataWorks needs to be configured to provide visibility of non-confidential data and then copy it to Watson Analytics. Steps to achieve this are executed in section 8.4, “Load data into Watson Analytics” on page 208.

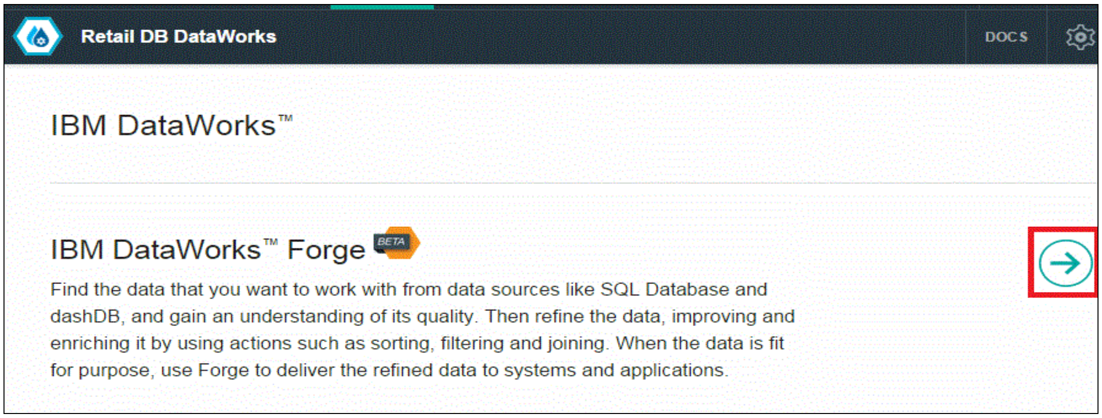
8.4 Load data into Watson Analytics

DataWorks Forge allows you to identify relevant data, transform the data to suit user needs, and load it to a system for use. The instance of DataWorks service created in section 8.2.4, “Setting DataWorks service in Bluemix” on page 205 is used to load data from the on-premises database into Watson Analytics. To read more about DataWorks Forge, see the following website:

https://www.ng.bluemix.net/docs/services/dataworks1/index-gentopic4.html

To use the DataWorks Forge service to load data into Watson Analytics, use the following steps:

1. In Bluemix dashboard, click the DataWorks service created earlier. This takes you to the **Retail DB DataWorks** dashboard. As shown in Figure 8-11, click **IBM DataWorks Forge** service to extract data from the on-premises database and load it into Watson Analytics.



Find the data that you want to work with from data sources like SQL Database and dashDB, and gain an understanding of its quality. Then refine the data, improving and enriching it by using actions such as sorting, filtering and joining. When the data is fit for purpose, use Forge to deliver the refined data to systems and applications.

Figure 8-11 IBM DataWorks Forge

2. In the IBM DataWorks Forge application, start by adding the IBM DB2 data source, as shown in Figure 8-12.

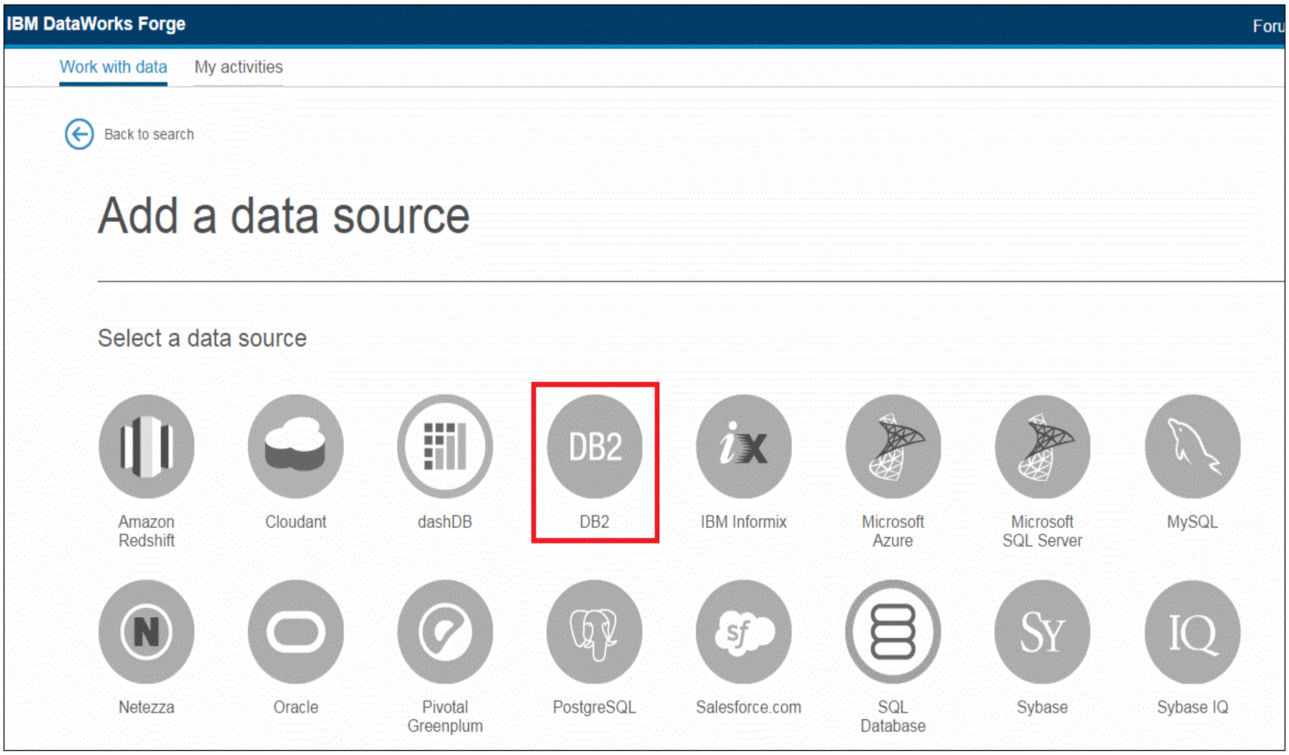


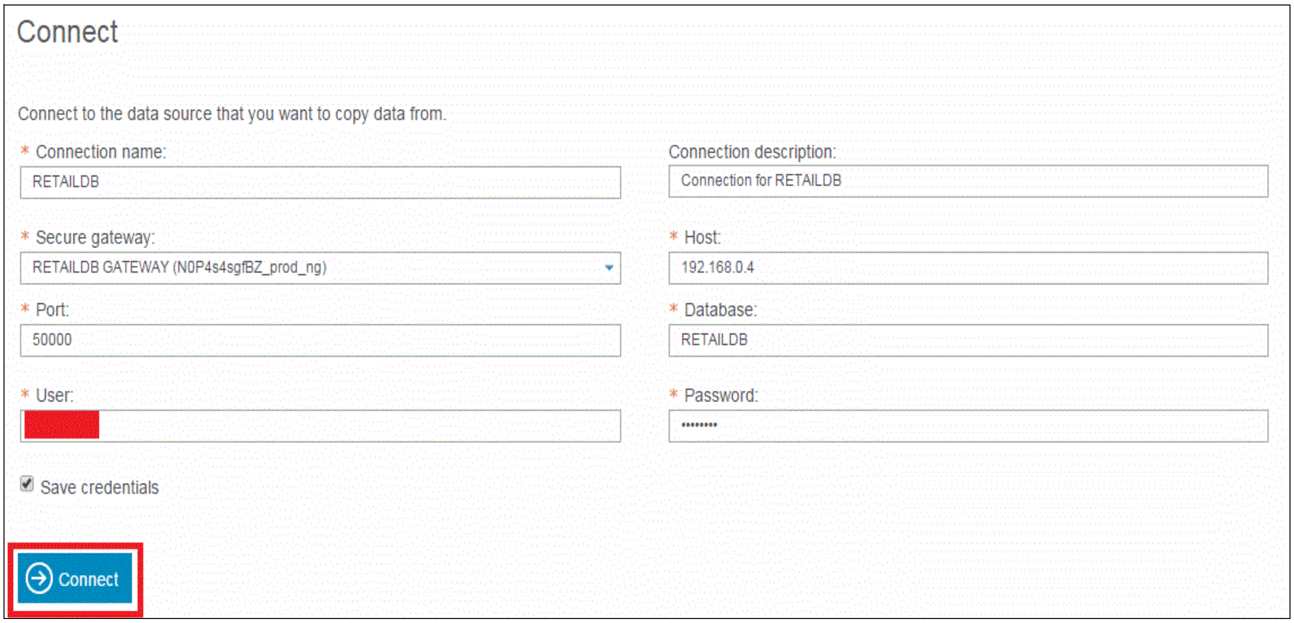
Figure 8-12 Add a DB2 data source in IBM DataWorks Forge

3. The next step is to create a connection for this data source by using the connection details provided in Table 8-2.

Table 8-2 Connection property, detail, and value for IBM DB2 data source

Property	Detail	Value
Connection name	Name of the connection and a destination with the same name will be created in the Secure Gateway	RETAILDB
Connection description	Description for this connection	Connection for RETAILDB
Secure gateway	Secure Gateway created for this service	Select the Gateway with gateway ID N0P4s4sgfBZ_prod_ng
Host	Host IP for on-premises DB2 server	192.168.0.4
Port	TCP port for DB2	50000
Database	Name of the database	RETAILDB
User	User name to access DB2	********
Password	Password for the user name	********

4. As shown in Figure 8-13, click **Connect** to establish connection to RETAILDB database after adding the connection details.



Connect

Connect to the data source that you want to copy data from.

* Connection name: RETAILDB	Connection description: Connection for RETAILDB
* Secure gateway: RETAILDB GATEWAY (N0P4s4sgfBZ_prod_ng)	* Host: 192.168.0.4
* Port: 50000	* Database: RETAILDB
* User: [redacted]	* Password: ••••••••

☑ Save credentials

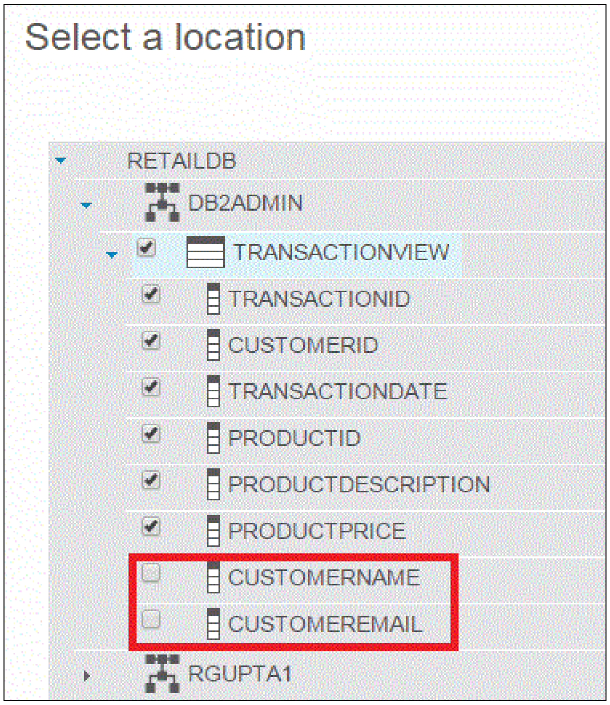
Figure 8-13 Establish connection to RETAILDB

5. The connection request is sent through the Secure Gateway and the Secure Gateway client logging shows information, such as when connections from the gateway are open and closed, as shown in Example 8-9.

Example 8-9 Gateway client logging information

```
[2015-08-30 00:38:51.160] [INFO] The Secure Gateway tunnel is connected
[2015-08-30 00:39:37.504] [INFO] Connection #0 is being established to 192.168.0.4:50000
[2015-08-30 00:39:43.832] [INFO] Connection #0 to 192.168.0.4:50000 was closed
[2015-08-30 00:41:38.130] [INFO] Connection #1 is being established to 192.168.0.4:50000
[2015-08-30 00:41:40.901] [INFO] Connection #1 to 192.168.0.4:50000 was closed
[2015-08-30 00:41:42.719] [INFO] Connection #2 is being established to 192.168.0.4:50000
[2015-08-30 00:41:46.900] [INFO] Connection #2 to 192.168.0.4:50000 was closed
```

6. Select the table and columns from which the data needs to be loaded into Watson Analytics. As shown in Figure 8-14, select columns in table **TRANSACTIONVIEW**, omitting two columns related to customer name and customer email, and then click **Complete**.

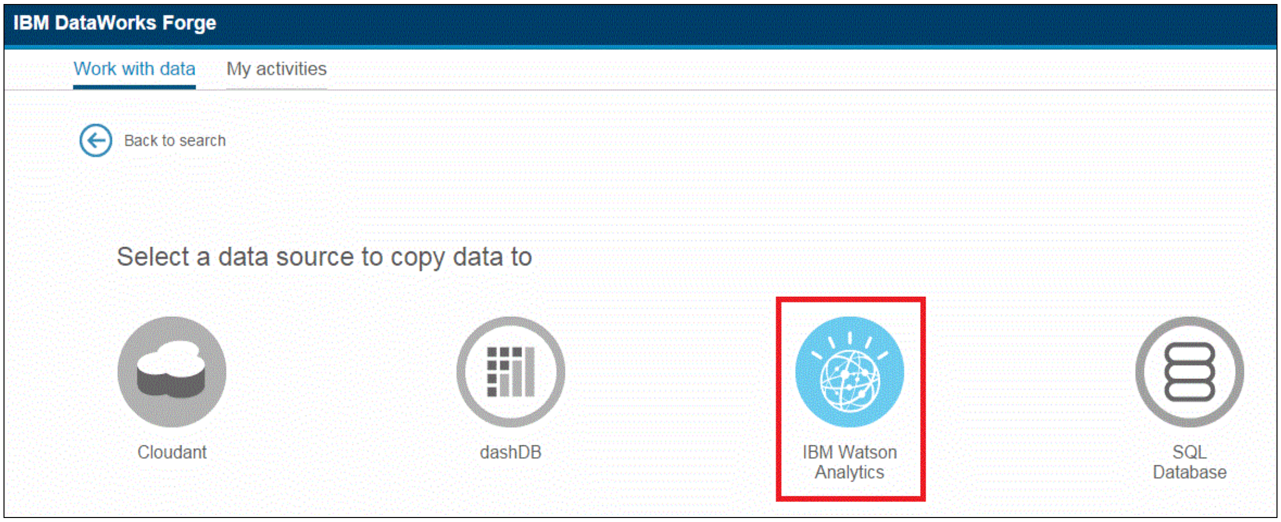


Select a location

- RETAILDB
  - DB2ADMIN
    - ☑ TRANSACTIONVIEW
      - ☑ TRANSACTIONID
      - ☑ CUSTOMERID
      - ☑ TRANSACTIONDATE
      - ☑ PRODUCTID
      - ☑ PRODUCTDESCRIPTION
      - ☑ PRODUCTPRICE
      - ☐ CUSTOMERNAME
      - ☐ CUSTOMEREMAIL
  - RGUPTA1

Figure 8-14 Select data to be loaded into Watson Analytics

7. Click **Copy** and then select the destination for copying this data. Select **IBM Watson Analytics** as shown in Figure 8-15.



Select a data source to copy data to

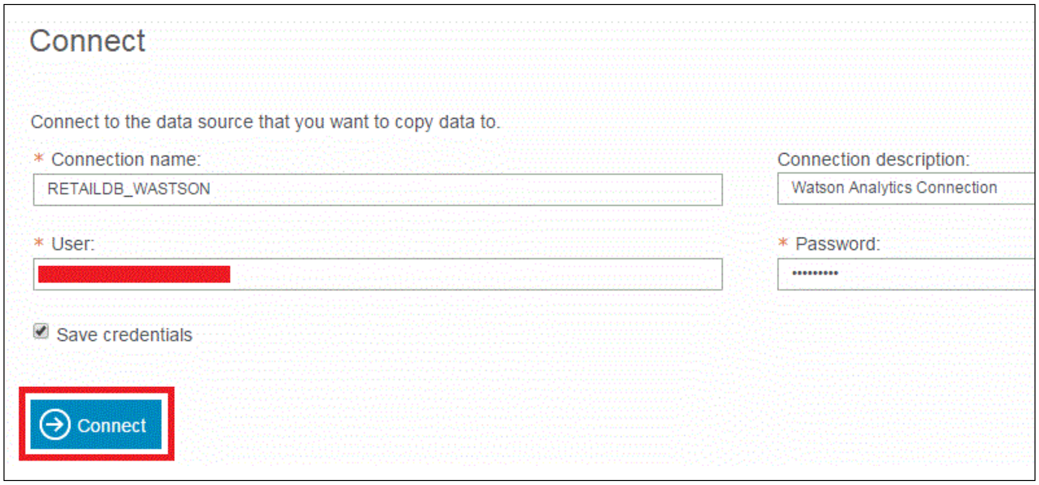
Figure 8-15 Select destination as Watson Analytics

8. Add the connection details for Watson Analytics by using the connection details provided in Table 8-3.

Table 8-3 Connection property, detail, and value for Watson Analytics service

Property	**Detail**	**Value**
Connection name	Connection name for Watson Analytics service	RETAILDB_WATSON
Connection description	Connection description	Watson Analytics Connection
User	User name for Watson Analytics service	***********************
Password	Password for Watson Analytics service	***********************

9. As shown in Figure 8-16, click **Connect** after adding the connection details for Watson Analytics service. Refer to 8.2.5, “Create a Watson Analytics user in IBM Cloud” on page 206 for registering with Watson Analytics service.



Connect

Connect to the data source that you want to copy data to.

* Connection name: RETAILDB_WASTSON

Connection description: Watson Analytics Connection

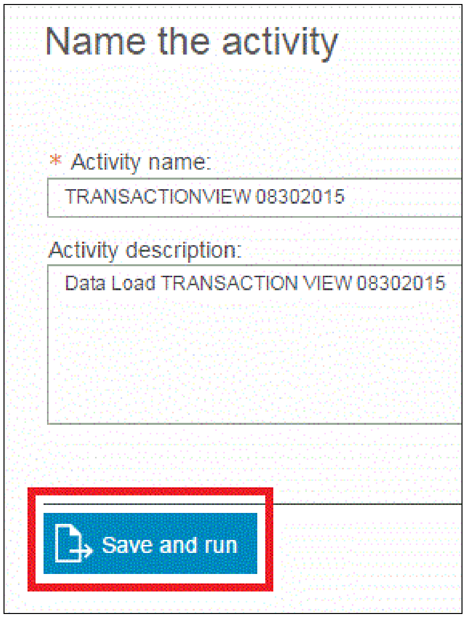
* User: [redacted]

* Password: •••••••••

☑ Save credentials

Figure 8-16 Establish connection to Watson Analytics service

10. Data can now be loaded into Watson Analytics service by creating an activity, as shown in Figure 8-17.



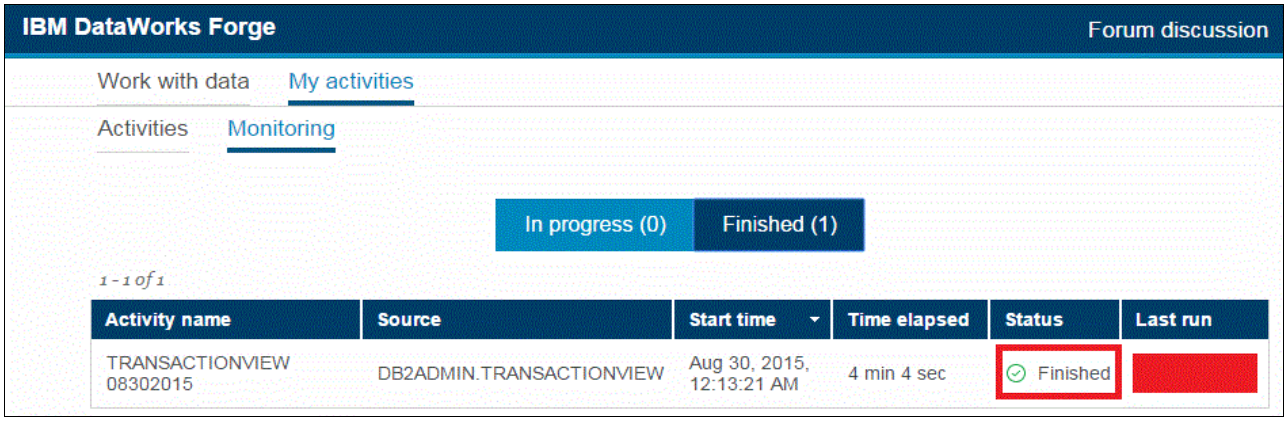
Name the activity

* Activity name: TRANSACTIONVIEW 08302015

Activity description: Data Load TRANSACTION VIEW 08302015

Figure 8-17 Watson Analytics activity

11. The Monitoring tab can be used to check the status of this activity. Figure 8-18 shows the successful completion of a data load in Watson Analytics service.



Activity name	Source	Start time	Time elapsed	Status	Last run
TRANSACTIONVIEW 08302015	DB2ADMIN.TRANSACTIONVIEW	Aug 30, 2015, 12:13:21 AM	4 min 4 sec	✓ Finished	[redacted]

Figure 8-18 Successful completion of data upload in Watson Analytics

12. Sign in to Watson Analytics dashboard by using the following URL to validate if the data is successfully loaded:

    http://www.ibm.com/analytics/watson-analytics

Figure 8-19 shows that a data set for the TRANSACTIONVIEW table from RETAILDB was successfully loaded into Watson Analytics. This data set can now be used for exploration, prediction, and assembling charts.

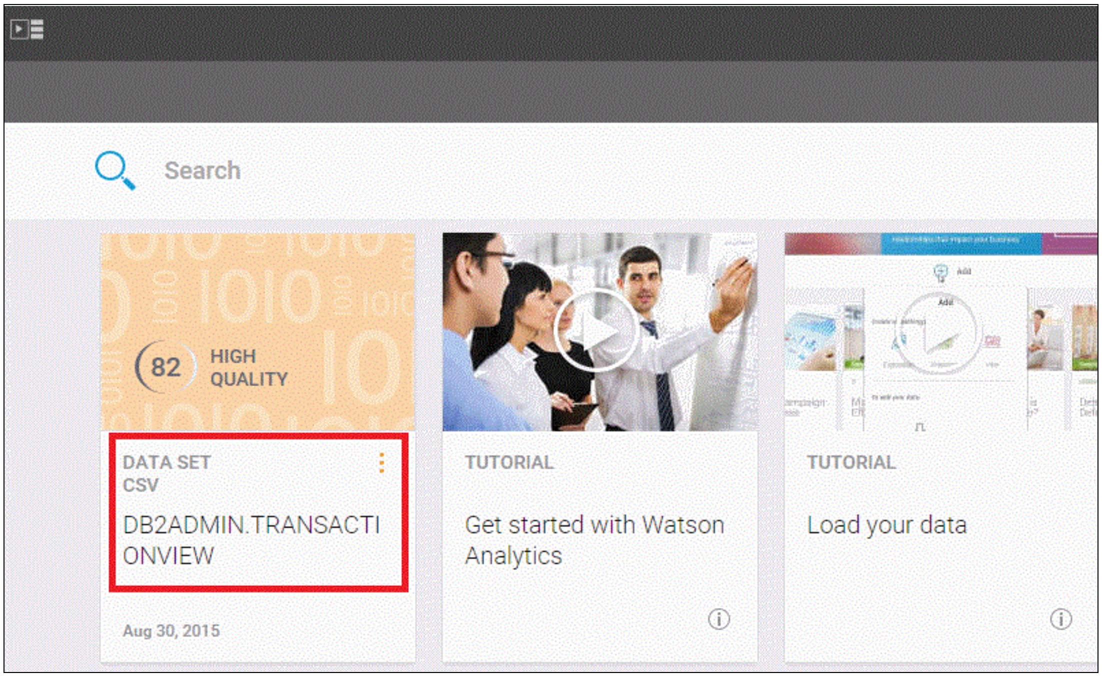
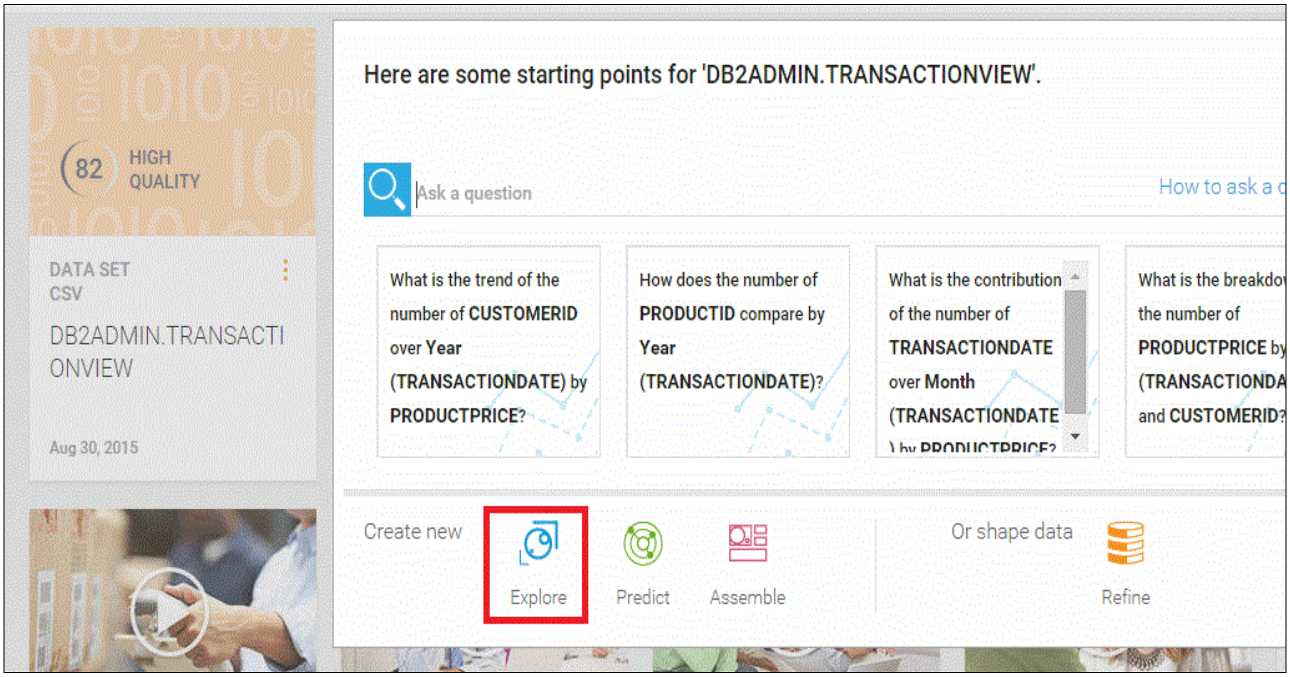


Figure 8-19 Data set available in Watson Analytics

8.5 Exploring data in Watson Analytics

This section shows how the data uploaded in Watson Analytics can be used to explore data in natural language:

1. Sign in to Watson Analytics dashboard and click data set **DB2ADMIN.TRANSACTIONVIEW** to create a new exploration, as shown in Figure 8-20.



Here are some starting points for 'DB2ADMIN.TRANSACTIONVIEW'.

What is the trend of the number of **CUSTOMERID** over **Year** **(TRANSACTIONDATE)** by **PRODUCTPRICE?**

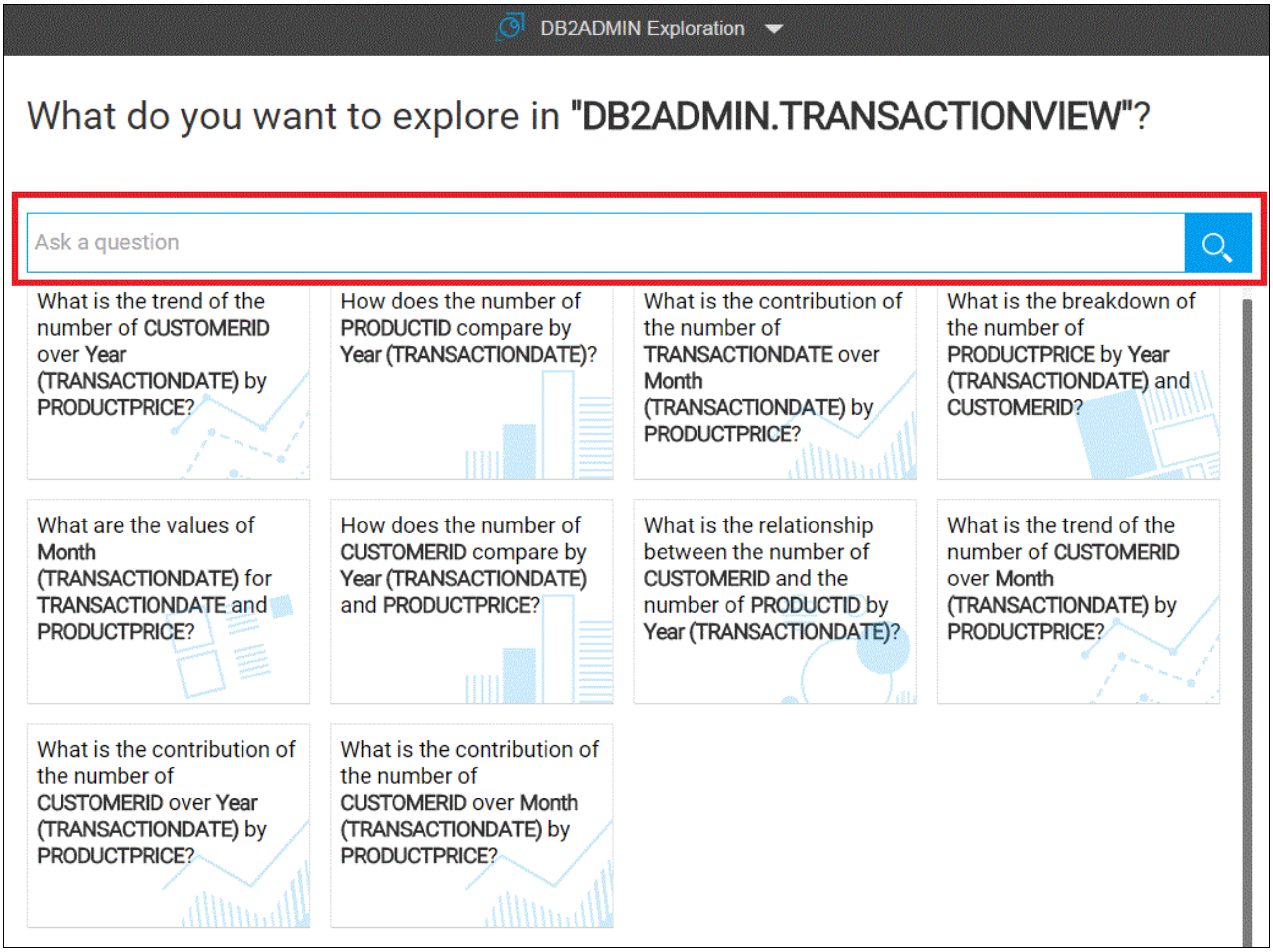
How does the number of **PRODUCTID** compare by **Year** **(TRANSACTIONDATE)?**

What is the contribution of the number of **TRANSACTIONDATE** over **Month** **(TRANSACTIONDATE** **) by PRODUCTPRICE?**

What is the breakdo the number of **PRODUCTPRICE by** **(TRANSACTIONDA** and **CUSTOMERID?**

Figure 8-20 Create a new exploration in Watson Analytics

2. As shown in Figure 8-21, a data analyst can type a question in natural language to explore the data set.



What do you want to explore in "DB2ADMIN.TRANSACTIONVIEW"?

What is the trend of the number of CUSTOMERID over Year (TRANSACTIONDATE) by PRODUCTPRICE?

How does the number of PRODUCTID compare by Year (TRANSACTIONDATE)?

What is the contribution of the number of TRANSACTIONDATE over Month (TRANSACTIONDATE) by PRODUCTPRICE?

What is the breakdown of the number of PRODUCTPRICE by Year (TRANSACTIONDATE) and CUSTOMERID?

What are the values of Month (TRANSACTIONDATE) for TRANSACTIONDATE and PRODUCTPRICE?

How does the number of CUSTOMERID compare by Year (TRANSACTIONDATE) and PRODUCTPRICE?

What is the relationship between the number of CUSTOMERID and the number of PRODUCTID by Year (TRANSACTIONDATE)?

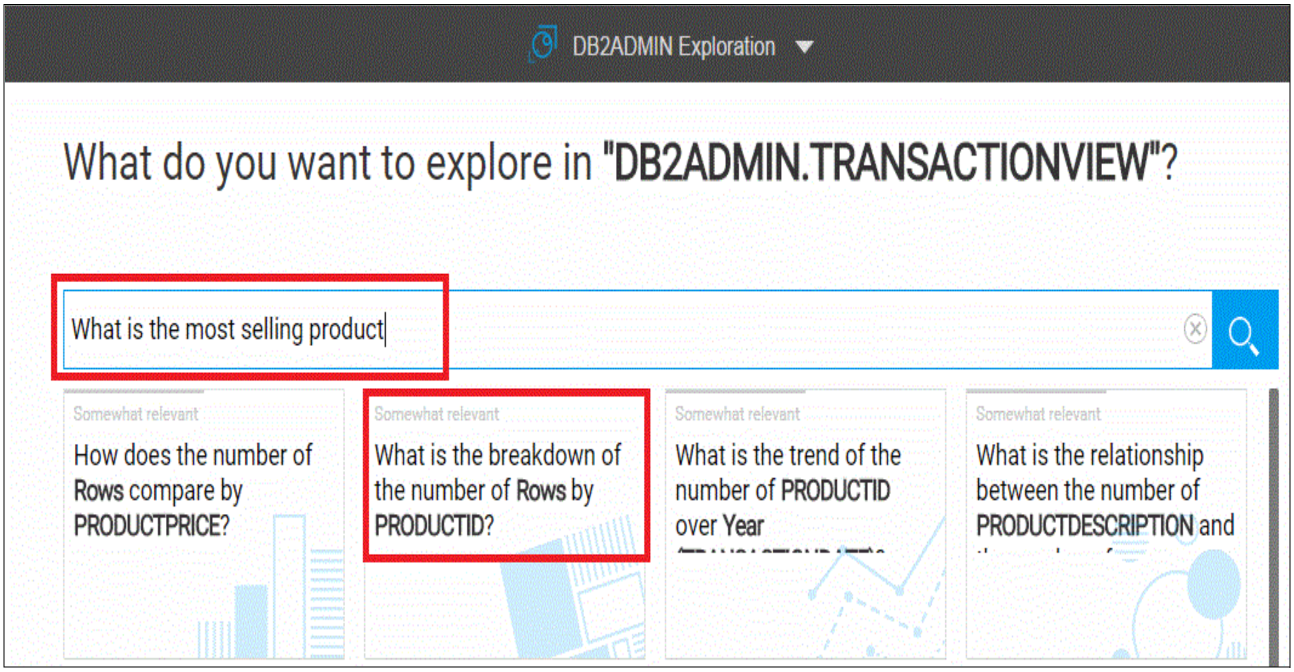
What is the trend of the number of CUSTOMERID over Month (TRANSACTIONDATE) by PRODUCTPRICE?

What is the contribution of the number of CUSTOMERID over Year (TRANSACTIONDATE) by PRODUCTPRICE?

What is the contribution of the number of CUSTOMERID over Month (TRANSACTIONDATE) by PRODUCTPRICE?

Figure 8-21 Explore data set in natural language

3. The data analyst wants to know the answer for the most selling product in retail stores, so the analyst types a question in natural language shown in Figure 8-22 on page 217. Watson Analytics responds and recommends exploration areas. Click the box showing the question **“What is the breakdown of the number of Rows by PRODUCTID?”**



What do you want to explore in "DB2ADMIN.TRANSACTIONVIEW"?

What is the most selling product

How does the number of Rows compare by PRODUCTPRICE?

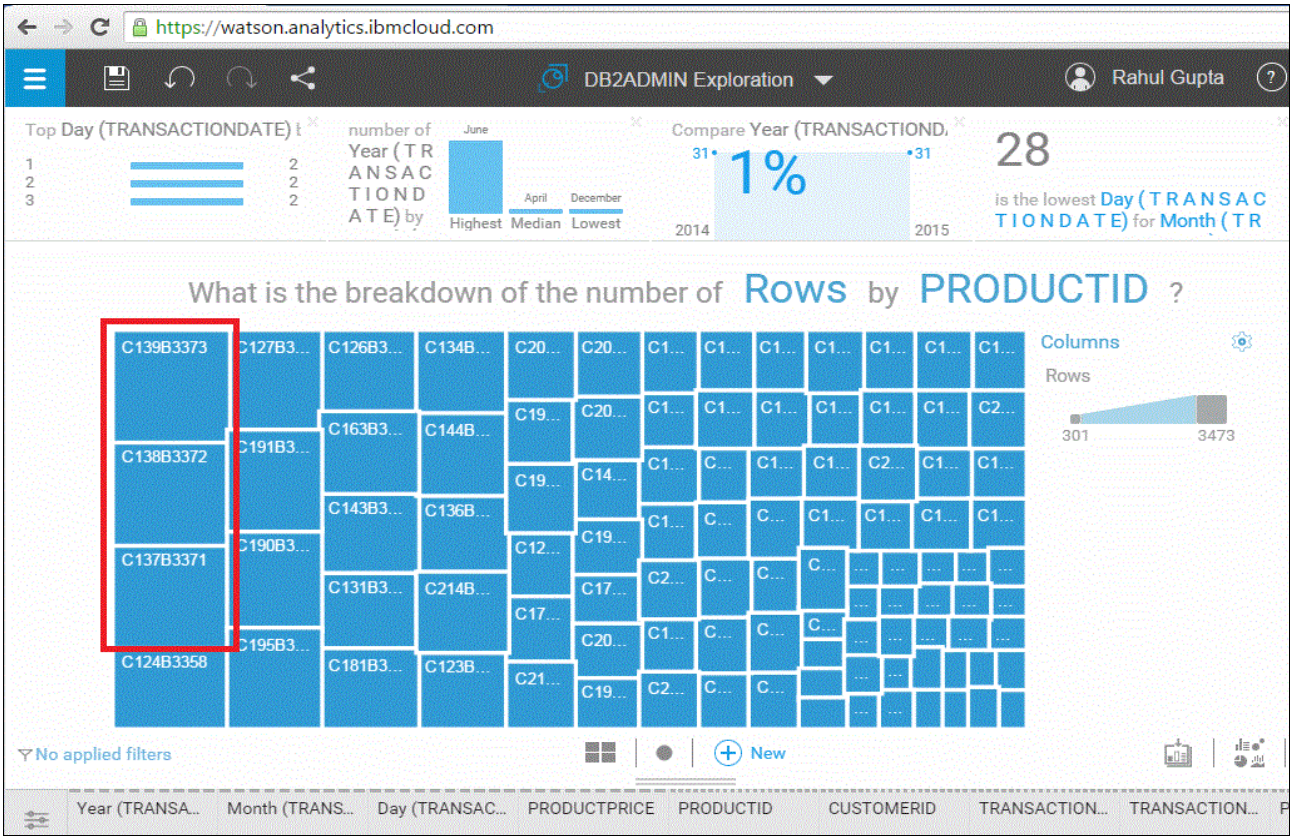
What is the breakdown of the number of Rows by PRODUCTID?

What is the trend of the number of PRODUCTID over Year

What is the relationship between the number of PRODUCTDESCRIPTION and

Figure 8-22 Ask a question in natural language

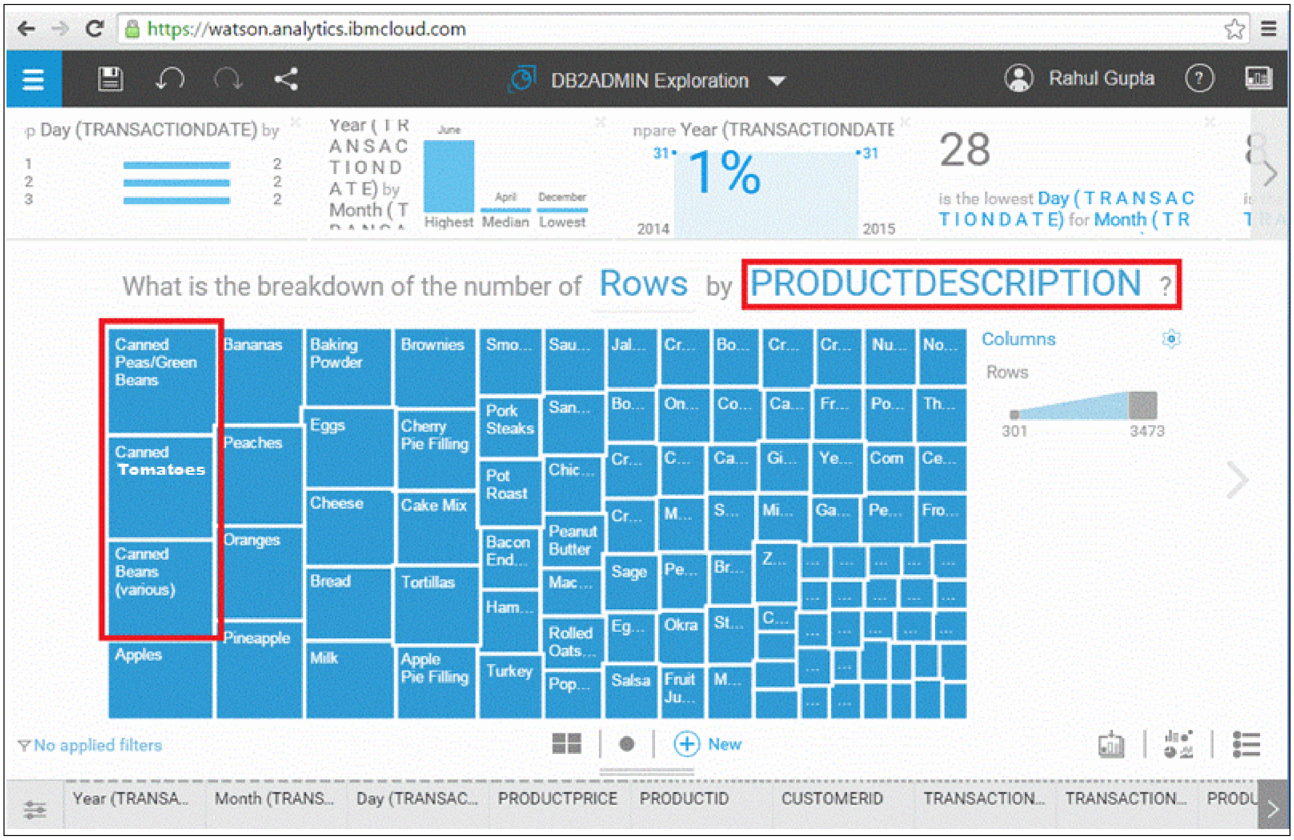
Figure 8-23 shows a treemap showing the highest selling product ID in the given data set of TRANSACTIONVIEW from RETAILDB.



What is the breakdown of the number of Rows by PRODUCTID ?

Figure 8-23 Treemap for highest selling product ID

4. The treemap can be re-created by changing the field from PRODUCTID to PRODUCTDESCRIPTION, as shown in Figure 8-24. The highest selling products are canned peas, canned tomatoes, and canned beans.



What is the breakdown of the number of Rows by PRODUCTDESCRIPTION ?

Figure 8-24 Treemap for the highest selling product

8.6 Conclusion

This chapter explained the step-by-step procedure to load data from cloud or on-premises databases into Watson Analytics service in IBM Cloud. It also provides instructions to perform the following actions:

- Load sample data in IBM DB2 databases.
- Configure the Secure Gateway service in Bluemix and the gateway client in on-premises server.
- Configure the DataWorks service in Bluemix.
- Use DataWorks Forge to load data in Watson Analytics.
- Explore data in natural language in Watson Analytics service.

With this knowledge, users can configure Secure Gateway and DataWorks to access data in on-premises databases and then analyze the data set in Watson Analytics service.

9

Mobile hybrid scenario: Secure Gateway, Connect & Compose, and DataWorks

This chapter provides a sample scenario where a company can use integration service offerings on Bluemix to create a mobile application (app) to use enterprise data.

This chapter has the following sections:

© Copyright IBM Corp. 2015. All rights reserved.

9.1 Solution overview

CompanyC is a local retailer that has served the area for years, and is looking to expand its area of services. After studying the market, CompanyC decides to try a pilot program to provide same day delivery service for loyal customers in the area. To accomplish the task efficiently, CompanyC needs a mobile web application to facilitate the process for employees to claim an order, pick up all the items in the order from different aisles, and deliver the order to the customer’s door.

Technology wise, CompanyC uses on-premises IBM DB2 as its secured data storage. The inventory table that is needed for this application contains several columns that are confidential, and must be excluded from any over-the-air exchanges.

Now that the business goal is clear, the following section explains how we can use Secure Gateway and various Connect & Compose offerings to create an application programming interface (API) that caters to each step of the delivery process.

> **Business case for this scenario:** For more information about the business case for this scenario and the company profile, see 1.3.5, “Solutions and actions by CompanyC CIO” on page 11.

9.1.1 Objectives

This scenario presents an example on the order delivery process by the employee of CompanyC:

1. **When EmployeeA claims an order, nobody should be able to claim it anymore:** We can create an API by using the connect feature from Connect & Compose, with the help of Secure Gateway, to retrieve and update the Order record to indicate that the order is under process. We need to make sure that this API is strictly retrieve and update only, and does not return more than the necessary information.
2. **EmployeeA should see a list of products ordered by aisle:** Similarly, we can create an API that retrieves product details from the Inventory table. To prevent leaking confidential information, the API limits the available parameter to just Name, Description, Product ID, and link to a product picture if available.
3. **EmployeeA should indicate product availability as he or she retrieves them:** By using the update function created in objective 1, EmployeeA should be able to indicate a product is retrieved or unavailable, and recalculate the order accordingly.
4. **When EmployeeA is done loading the goods, an email should be sent to the customer with estimated delivery time, and directions to the customer’s location should be displayed:** By using the compose feature in Connect & Compose, we can create an API that shows EmployeeA the directions to the location that the customer has indicated, as well as sending an email to the customer to expect delivery.
5. **When the order is received by the customer, EmployeeA should update order status and complete the transaction:** We again reuse the update function created in objective 1 to indicate in the database record that the goods have been delivered. We can also use the compose feature to bind the action to call another hypothetical external API that will process the credit card payment for the transaction.

9.2 Step-by-step implementation

We show how to use Bluemix to create a web application using NodeJS, and several hybrid cloud service offerings.

9.2.1 Creating a NodeJS Web APP on Bluemix

First, we create a NodeJS Web APP on Bluemix:

1. After logging in to Bluemix, in the **DASHBOARD** view, select the wanted space, and create a Cloud Foundry app.
2. The first window is shown in Figure 9-1. We select **WEB** for the type of app to create for more flexibility.

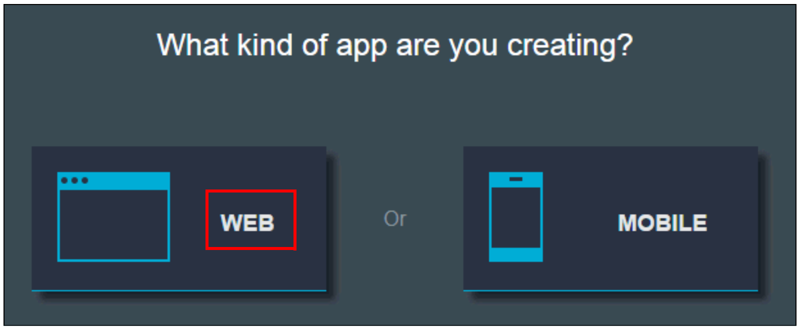
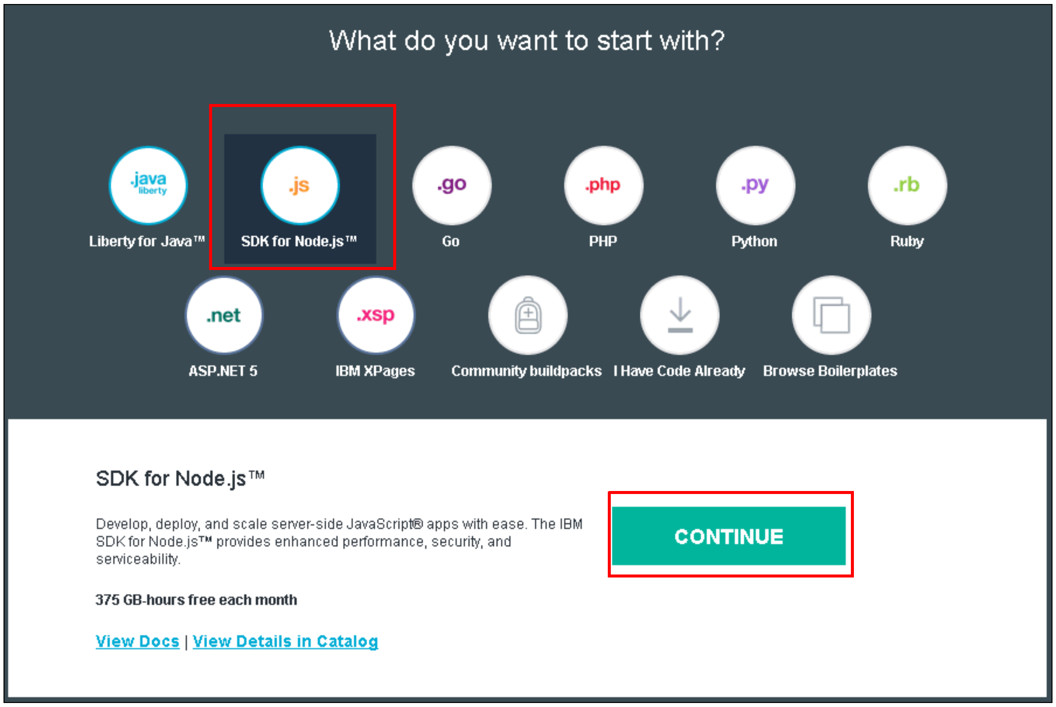


Figure 9-1 First step in creating a Cloud Foundry application

3. Next, as shown in Figure 9-2, we choose **SDK for Node.js**, and click **CONTINUE**.



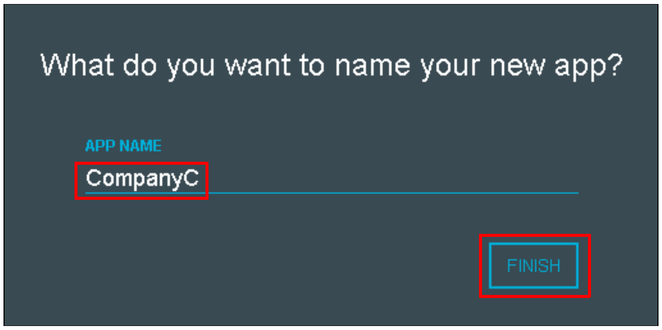
SDK for Node.js™

Develop, deploy, and scale server-side JavaScript® apps with ease. The IBM SDK for Node.js™ provides enhanced performance, security, and serviceability.

375 GB-hours free each month

Figure 9-2 Select the language for the application template

4. In Figure 9-3, we enter an app name, and click **FINISH**.

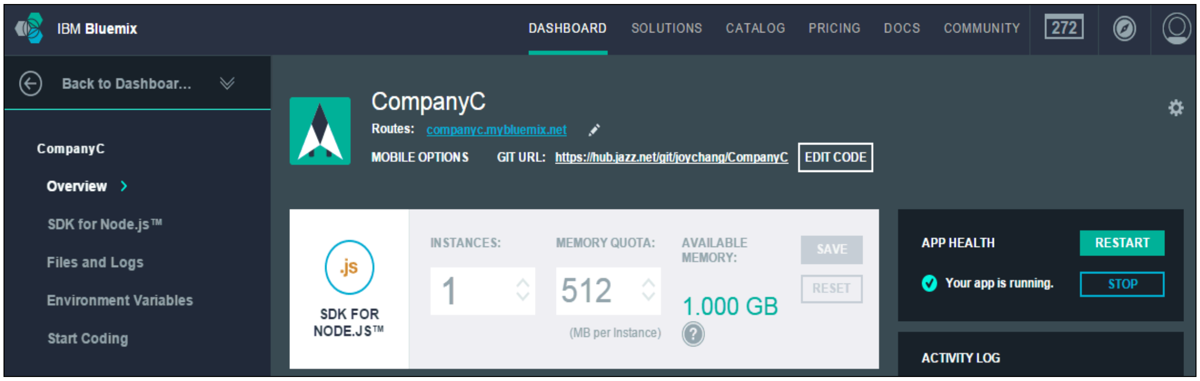


What do you want to name your new app?

APP NAME: CompanyC

Figure 9-3 Enter APP NAME

5. Bluemix then creates an application template and deploys it. When the process is done, the template application is accessible at `{application-name}.mybluemix.net`, in this case, *companyc.mybluemix.net*.
6. The setup wizard then shows three ways to work with the source code. Follow the instructions to set up according to your preference. This sample uses git. Figure 9-4 shows the app overview page. Notice that this page displays the app route, app status, and various options.



CompanyC

Routes: companyc.mybluemix.net

GIT URL: https://hub.jazz.net/git/joychang/CompanyC

INSTANCES: 1

MEMORY QUOTA: 512 (MB per Instance)

AVAILABLE MEMORY: 1.000 GB

Your app is running.

Figure 9-4 Application overview page

7. After setting up the application for git, Bluemix provides an online code editor. By clicking **EDIT CODE**, as shown in the middle of Figure 9-4, a web editor is shown to allow online edits of the template source code.

> **Tip:** If you are familiar with git, you can also clone the source to your local machine by using the regular git way.

9.2.2 Setting up Secure Gateway

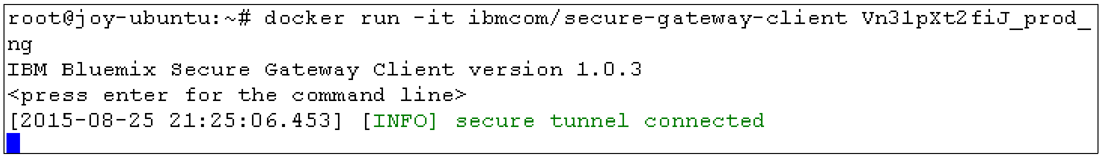
Because CompanyC hosts their database in an enterprise zone, it is necessary to establish a secured connection through the firewall:

1. First, provision the Secure Gateway service. The service can be found by typing in the `CATALOG` search bar, or filter by the Integration category.
2. From the Secure Gateway service landing page, click **ADD GATEWAY**. Give the Secure Gateway a name and click **CONNECT IT**. The name is used in Connect & Compose later.

3. As Figure 9-6 shows, there are currently two options to set up a Secure Connector. In this example, we used the Docker variety, which requires installation of Docker on a Linux machine that has network access to the on-premises database.

> **Instructions for Docker installation:** For Docker installation instructions, see the following site:
>
> `https://docs.docker.com/installation/`

4. Simply copy the command to run as administrator on the Docker machine, as shown in Figure 9-5.

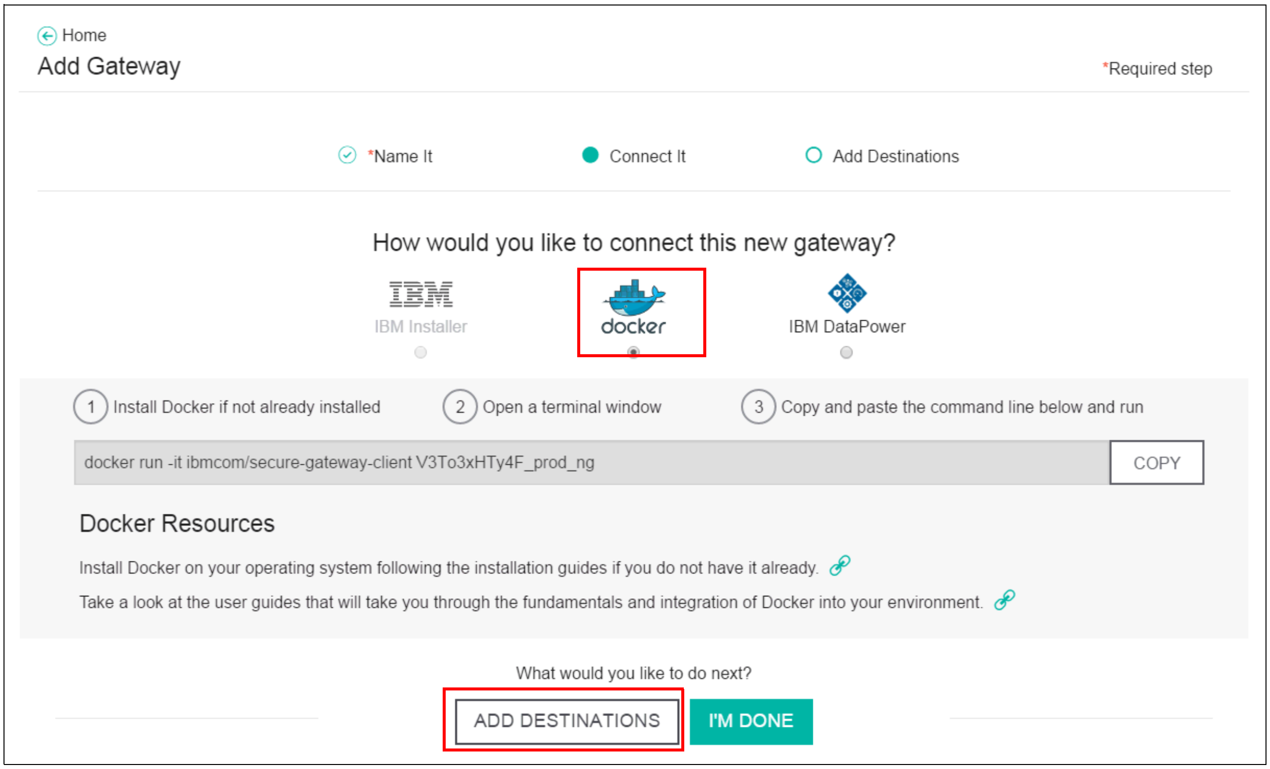


```
root@joy-ubuntu:~# docker run -it ibmcom/secure-gateway-client Vn31pXt2fiJ_prod_
ng
IBM Bluemix Secure Gateway Client version 1.0.3
<press enter for the command line>
[2015-08-25 21:25:06.453] [INFO] secure tunnel connected
```

Figure 9-5 Starting Secure Gateway Client on the Docker machine

You will see "*[INFO] secure tunnel connected"* on your Docker machine terminal, and a new panel on Secure Gateway service indicating that the gateway has been connected.

5. Proceed to click **ADD DESTINATIONS** at the bottom of the panel, as shown in Figure 9-6.



How would you like to connect this new gateway?

1 Install Docker if not already installed

2 Open a terminal window

3 Copy and paste the command line below and run

```
docker run -it ibmcom/secure-gateway-client V3To3xHTy4F_prod_ng
```

Docker Resources

Install Docker on your operating system following the installation guides if you do not have it already.

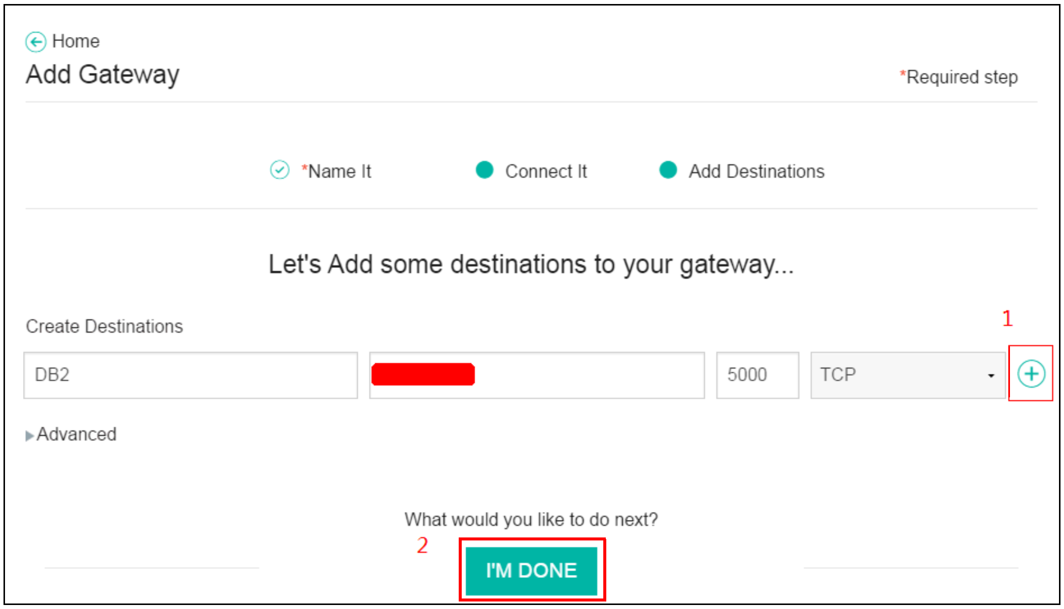
Take a look at the user guides that will take you through the fundamentals and integration of Docker into your environment.

What would you like to do next?

Figure 9-6 Secure Gateway setup information

6. Provide and note the name of the destination because this will be used in Connect & Compose later. Complete the host name and port information, then click **+** as shown in Figure 9-7. Ensure that the **+** has been clicked before clicking **I'M DONE**.

C



Home

Add Gateway

*Required step

*Name It   Connect It   Add Destinations

Let's Add some destinations to your gateway...

Create Destinations

1

DB2   [redacted]   5000   TCP

Advanced

What would you like to do next?

2

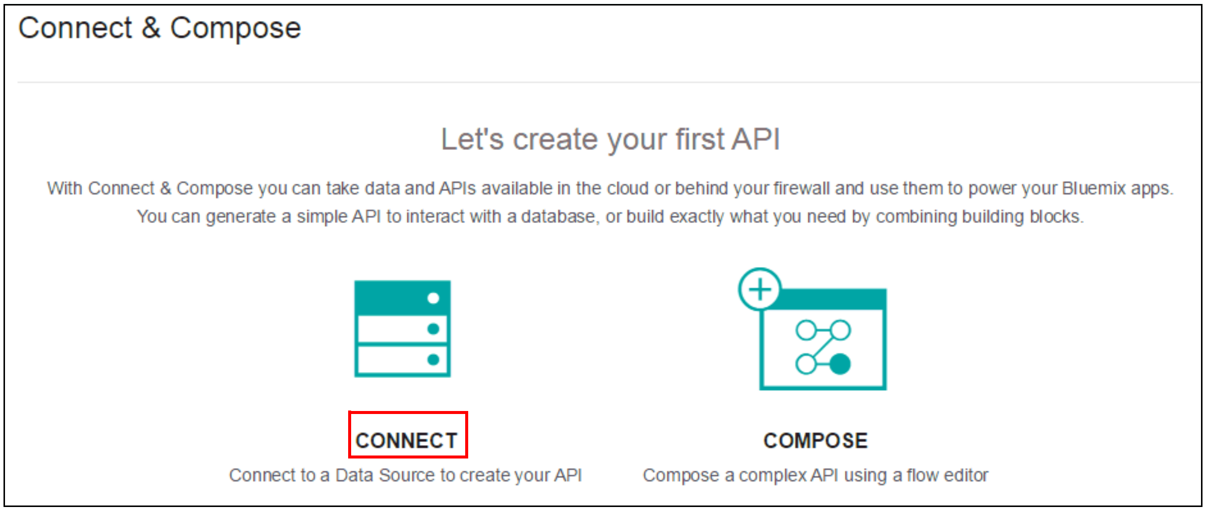
I'M DONE

Figure 9-7   Create a destination on the Secure Gateway client

9.2.3 Connecting an API to retrieve and update orders

Now that Secure Gateway is set up, we can use Connect & Compose to create a Representational State Transfer (REST) API to interact with the database. Figure 9-8 shows the landing page of Connect & Compose. We describe connecting to the Orders table in this section.

1. On the Connect & Compose landing page, click **CONNECT**.



Connect & Compose

Let's create your first API

With Connect & Compose you can take data and APIs available in the cloud or behind your firewall and use them to power your Bluemix apps. You can generate a simple API to interact with a database, or build exactly what you need by combining building blocks.

CONNECT

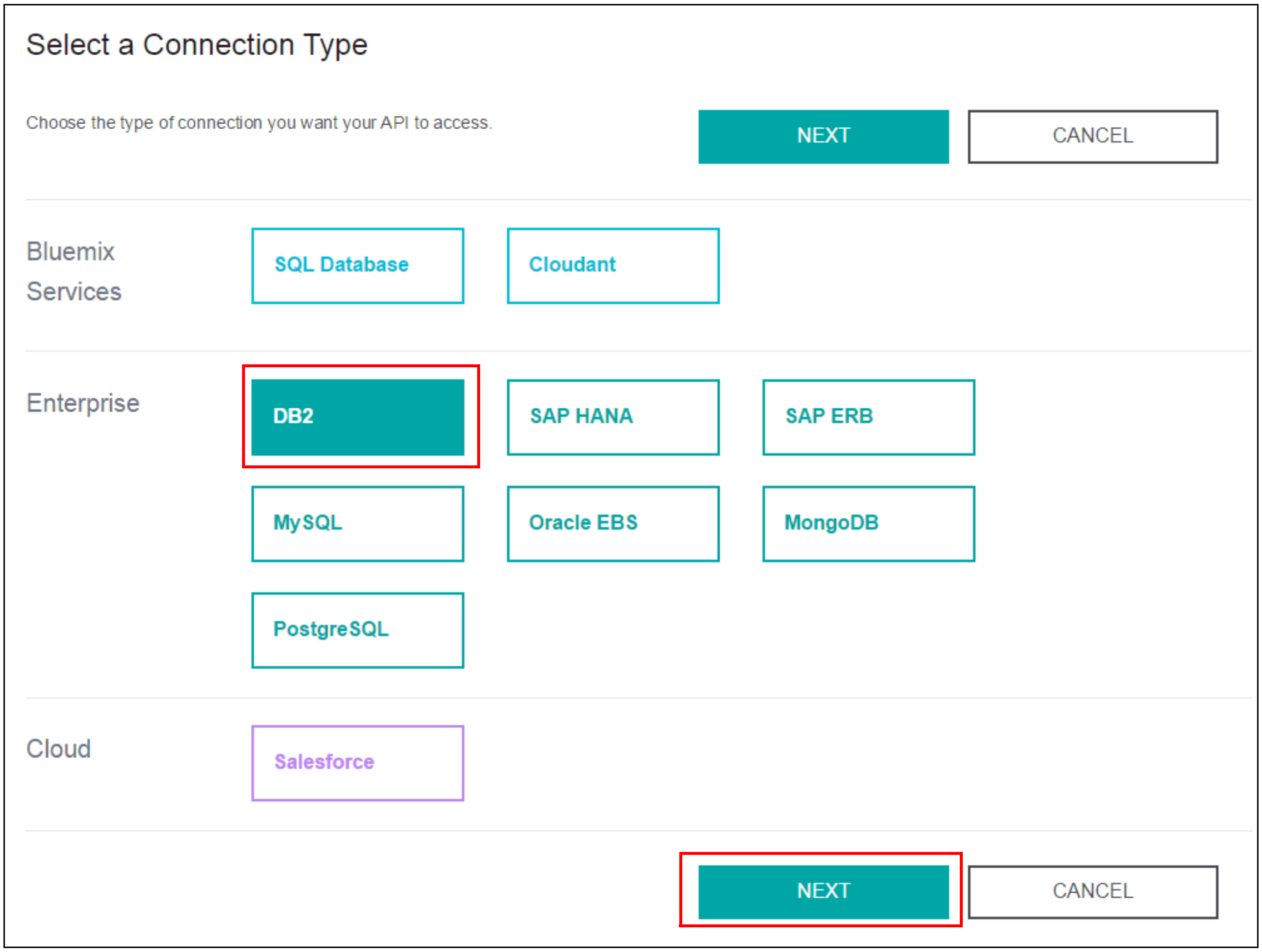
Connect to a Data Source to create your API

COMPOSE

Compose a complex API using a flow editor

Figure 9-8   Connect & Compose landing page

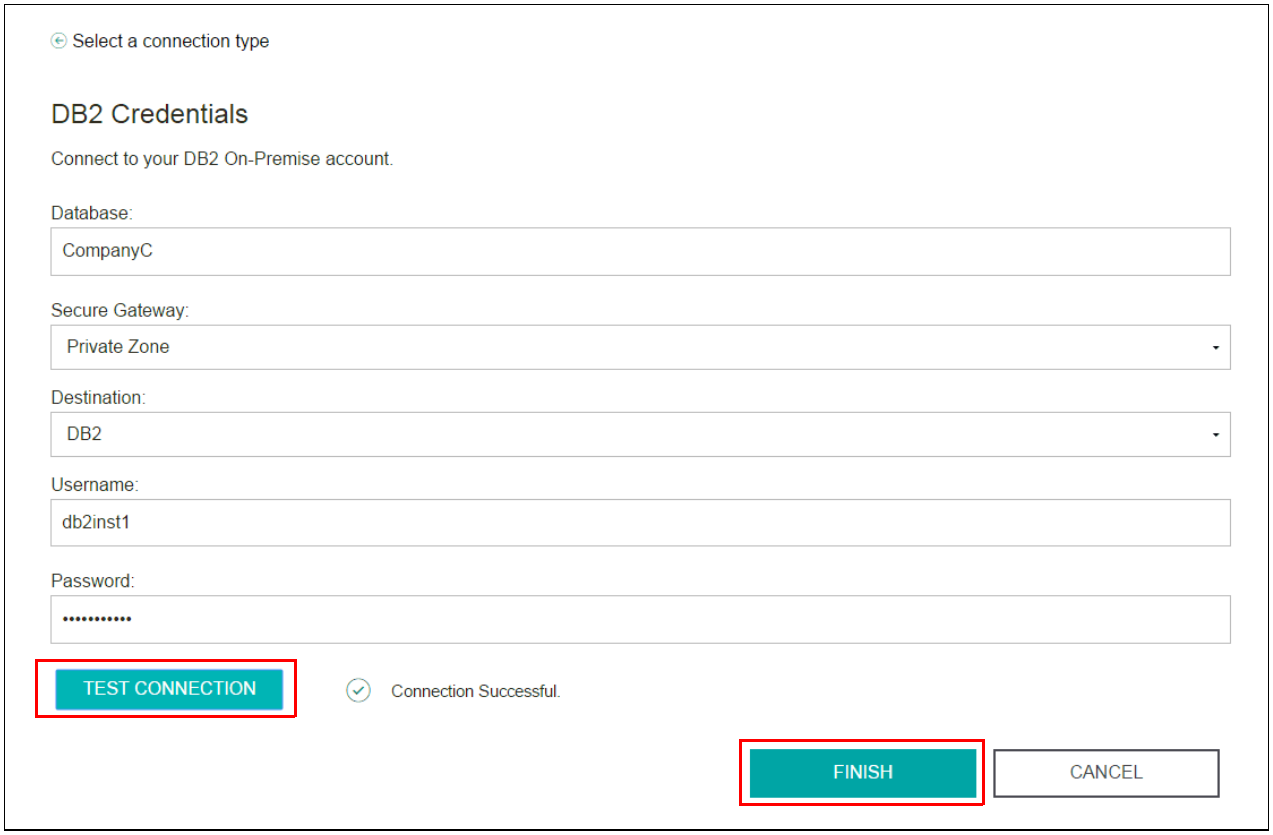
a. As shown in Figure 9-9, there are three different types of locations. Because CompanyC is using IBM DB2, we find **DB2** in the Enterprise location, and click **NEXT**.


Select a Connection Type

Choose the type of connection you want your API to access.

Figure 9-9 Connection types

2. Because Secure Gateway has already been set up to access our on-premises database in previous steps, Figure 9-10 shows a form to configure the connection. In the case where no Secure Gateway is set up, Connect & Compose prompts and provides options to redirect to the Secure Gateway service to set up first.
3. After inputting all the credentials, including selecting the Secure Gateway and Destination that we set up in previous steps, click **TEST CONNECTION**. Connect & Compose tests to see if the connection is established and the credentials are correct.
4. When "*Connection Successful"* is displayed, proceed and click **FINISH**.



Select a connection type

DB2 Credentials

Connect to your DB2 On-Premise account.

Database: CompanyC

Secure Gateway: Private Zone

Destination: DB2

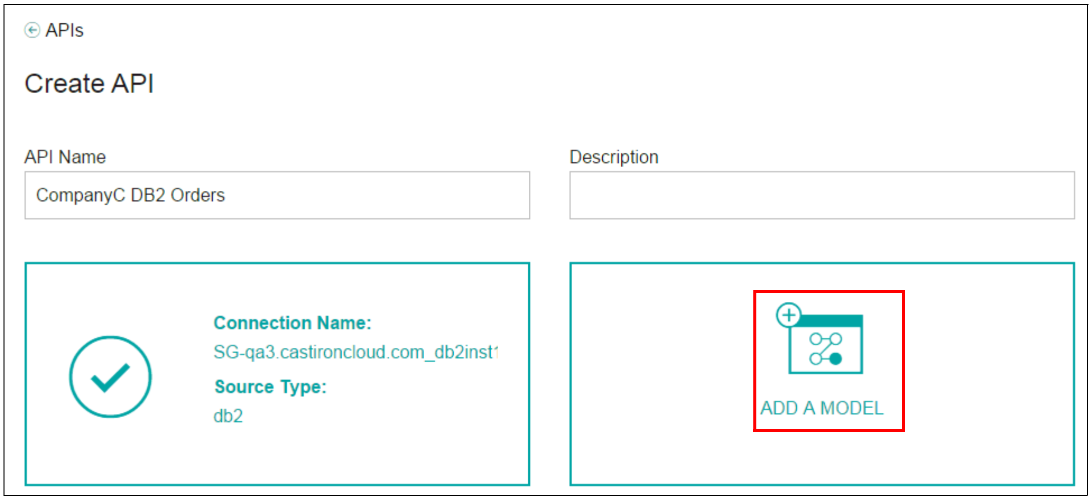
Username: db2inst1

Password: •••••••••••

Connection Successful.

Figure 9-10 DB2 credentials

5. The model selection box becomes active, as shown in Figure 9-11. Click **ADD A MODEL** to see the model panel in Figure 9-12 on page 228. This is where we select the table that we want to use.



APIs

Create API

API Name: CompanyC DB2 Orders

Description:

Connection Name: SG-qa3.castironcloud.com_db2inst1

Source Type: db2

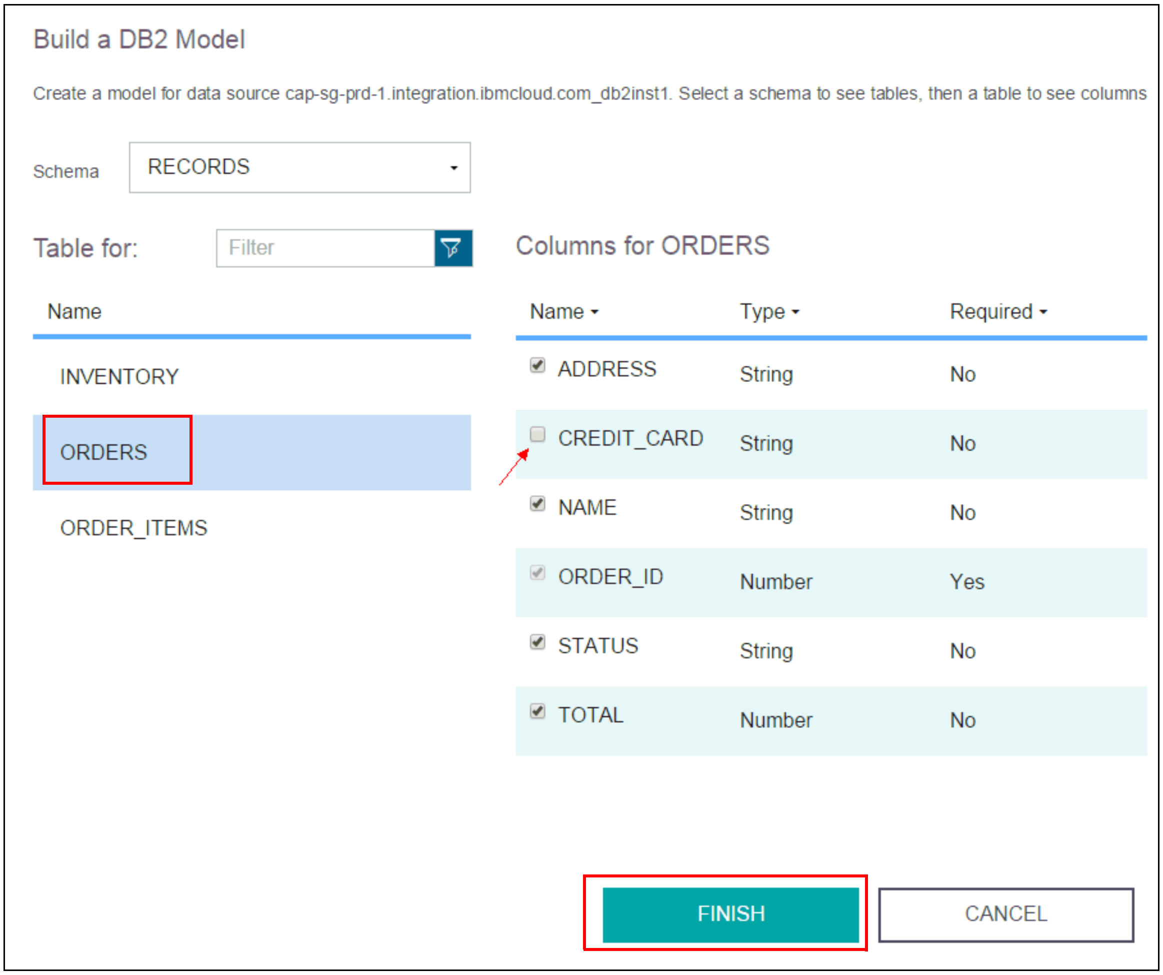
Figure 9-11 ADD A MODEL becomes active when a connection is created

6. First, select the **schema** from the drop-down list. Connect & Compose will then query for the list of tables.
7. Then, click the wanted table. For this section, we need to get **ORDERS**. Connect & Compose will then query for all the columns.

8. Finally, clear any columns that we do not want to expose with this API. As shown in Figure 9-12, CompanyC's employees should not see any payment information; therefore, the credit card is cleared.

> **Tip:** Any column that does not include a *NOT NULL* restraint can be cleared.

9. When you have completed your selection, click **FINISH**.



Build a DB2 Model

Create a model for data source cap-sg-prd-1.integration.ibmcloud.com_db2inst1. Select a schema to see tables, then a table to see columns

Schema: RECORDS

Table for: Filter

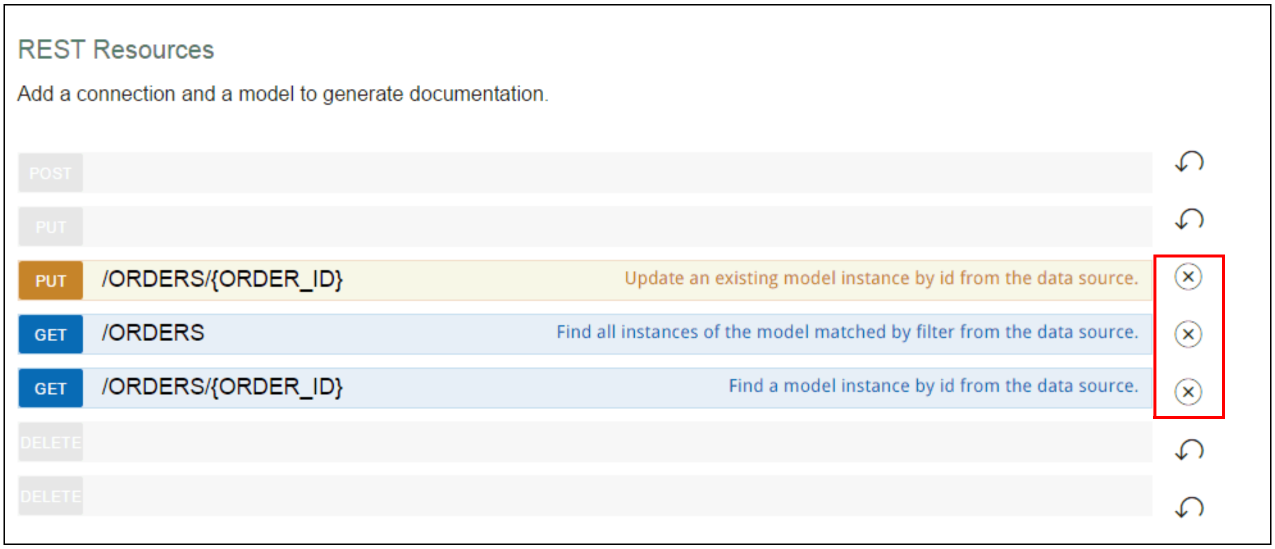
Name
INVENTORY
ORDERS
ORDER_ITEMS

Columns for ORDERS

Name	Type	Required
☑ ADDRESS	String	No
☐ CREDIT_CARD	String	No
☑ NAME	String	No
☑ ORDER_ID	Number	Yes
☑ STATUS	String	No
☑ TOTAL	Number	No

Figure 9-12 Model selection view. Clear properties to exclude from the API

10. After clicking **FINISH**, the pop-up module closes, and the REST resources are populated. For this API, we only want updates with primary key and retrieve functions. We can clear it by clicking the **X** to the right of the endpoints, as in Figure 9-13.



REST Resources

Add a connection and a model to generate documentation.

POST		
PUT		
PUT	/ORDERS/{ORDER_ID}	Update an existing model instance by id from the data source.
GET	/ORDERS	Find all instances of the model matched by filter from the data source.
GET	/ORDERS/{ORDER_ID}	Find a model instance by id from the data source.
DELETE		
DELETE		

Figure 9-13 Clear API endpoints that we do not want to include in the API

11. Click **SAVE** to create the API. There might be a pop-up window that indicates the API is undergoing creation. Because it might take a while to make available the API routes, feel free to explore other functionalities in the meantime.

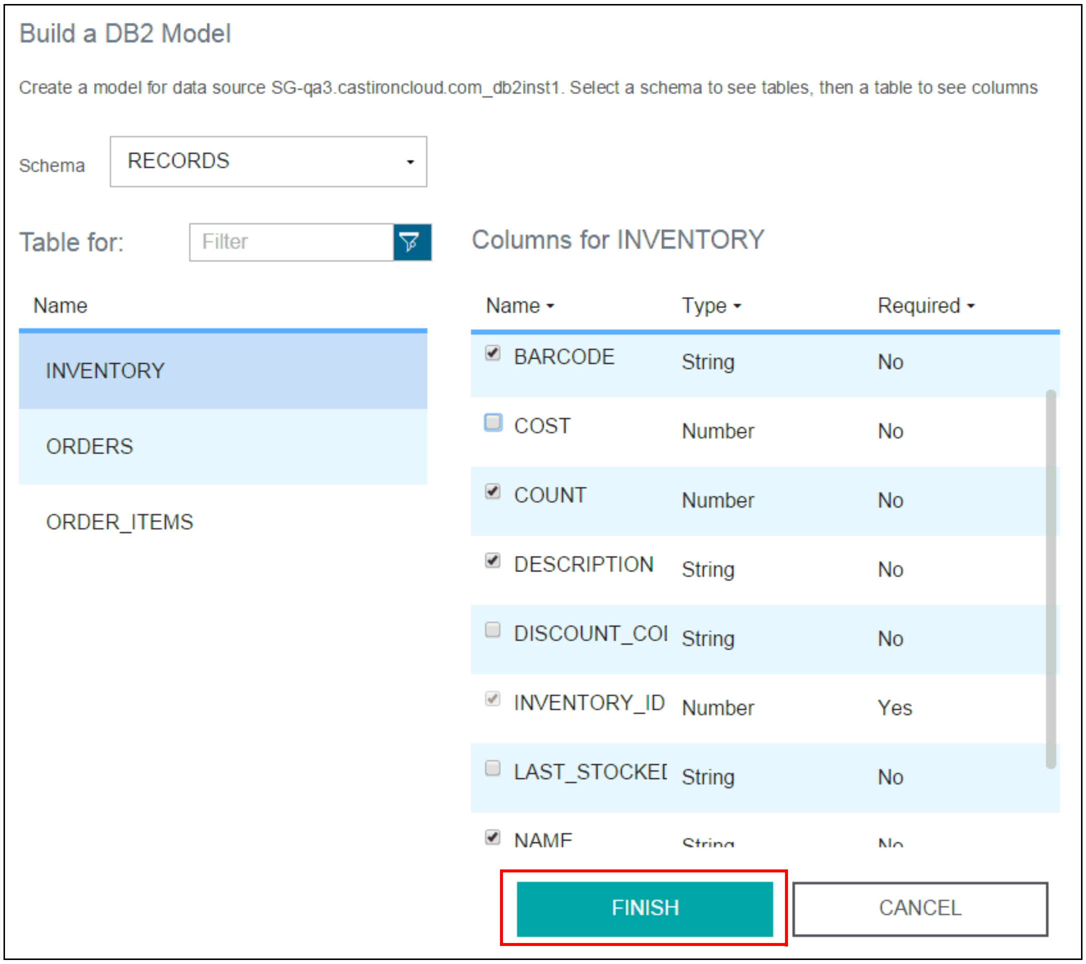
9.2.4 Connecting an API to retrieve product details

This operation has two parts to it because it involves two different tables in the database.

To create a connection to ordered items (order_items table), follow Steps 1 - 10 from the previous section (9.2.3, “Connecting an API to retrieve and update orders” on page 224). All of the properties are needed at step 8, and clear the same endpoints at step 9.

The second part of this operation is to create a connection to the Inventory table, which provides us with inventory and stock details. Again, follow Steps 1 - 10 from the previous section (9.2.3, “Connecting an API to retrieve and update orders” on page 224):

1. Figure 9-14 shows the models to select for Step 8, and Figure 9-15 on page 231 shows the endpoints that we use for Step 9.



Build a DB2 Model

Create a model for data source SG-qa3.castironcloud.com_db2inst1. Select a schema to see tables, then a table to see columns

Schema: RECORDS

Table for: Filter

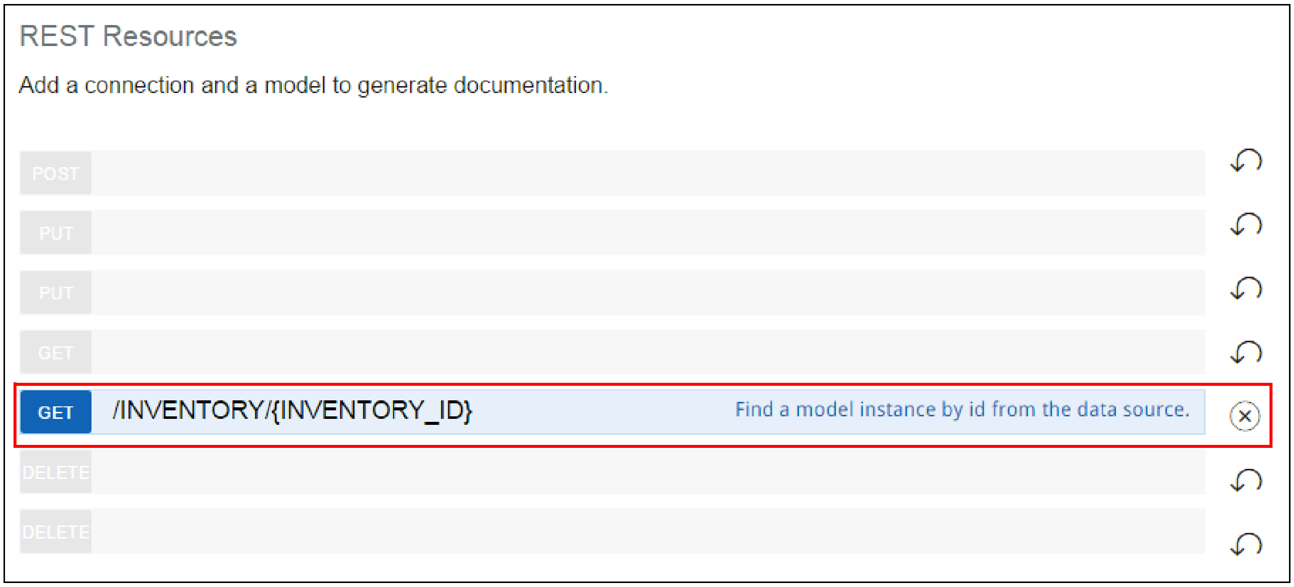
Name
INVENTORY
ORDERS
ORDER_ITEMS

Columns for INVENTORY

Name	Type	Required
☑ BARCODE	String	No
□ COST	Number	No
☑ COUNT	Number	No
☑ DESCRIPTION	String	No
□ DISCOUNT_COI	String	No
☑ INVENTORY_ID	Number	Yes
□ LAST_STOCKEI	String	No
☑ NAME	String	No

Figure 9-14 Only retain properties that are useful for the employees to prepare for delivery

2. The inventory data is only used to assist employees in CompanyC to find the products in the store. Therefore, as shown in Figure 9-15, we only retrieve by primary key function.



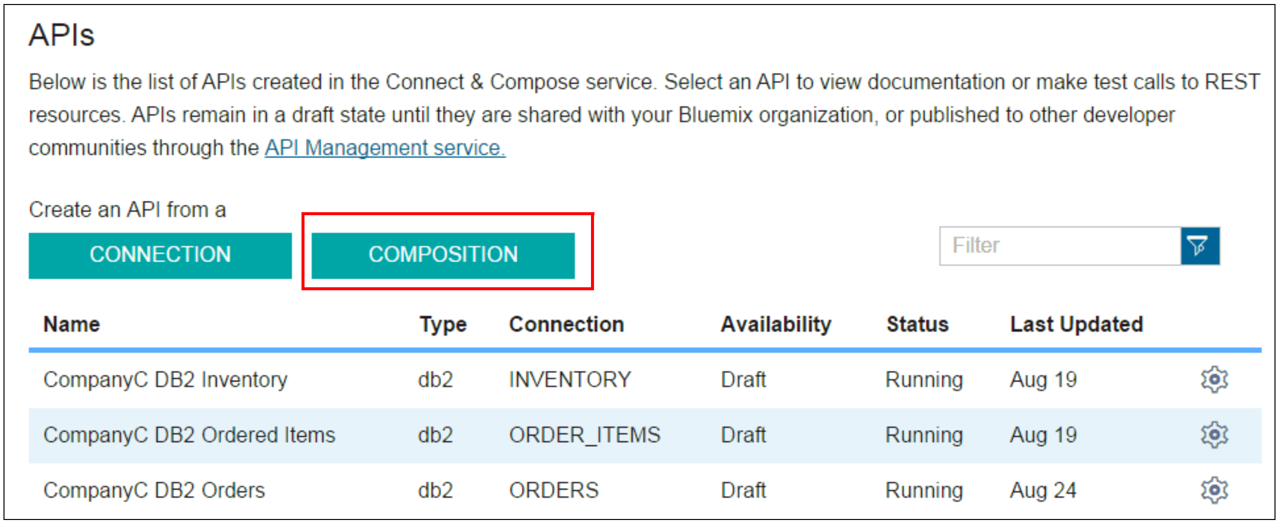
REST Resources

Add a connection and a model to generate documentation.

GET /INVENTORY/{INVENTORY_ID} Find a model instance by id from the data source.

Figure 9-15 Inventory is internal data, therefore we only want to retrieve information

3. After both APIs are created, we can then go to the landing page for Connect & Compose, shown in Figure 9-16. Next, we see the API creation page for Composition API in Figure 9-16. Click **COMPOSITION**.



APIs

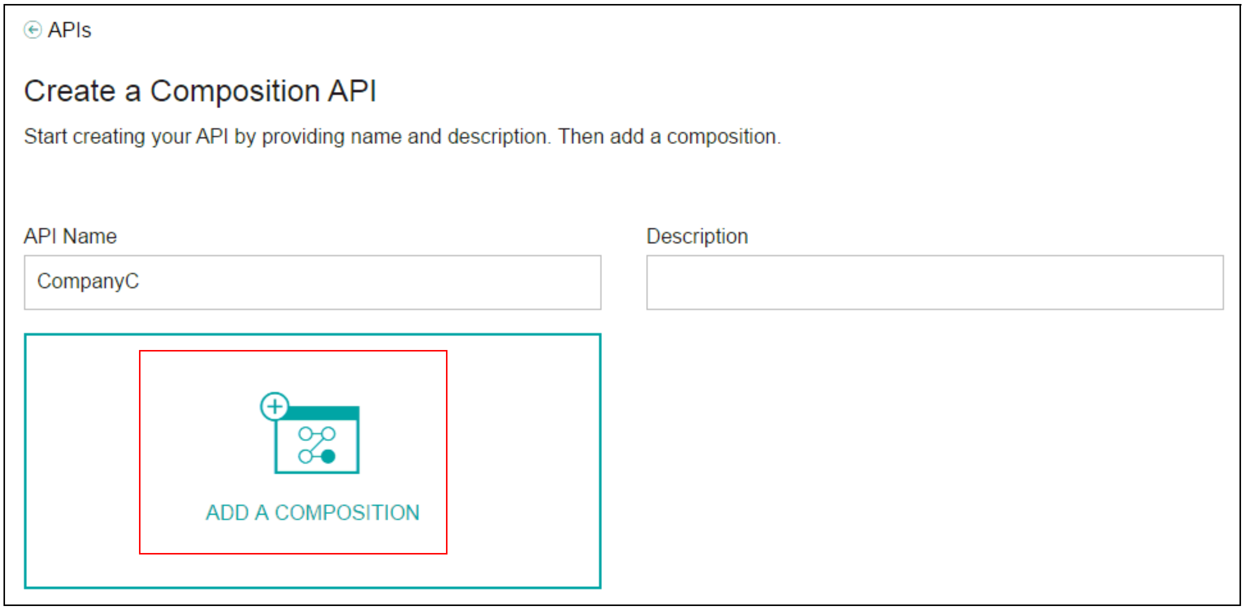
Below is the list of APIs created in the Connect & Compose service. Select an API to view documentation or make test calls to REST resources. APIs remain in a draft state until they are shared with your Bluemix organization, or published to other developer communities through the API Management service.

Create an API from a

Name	Type	Connection	Availability	Status	Last Updated
CompanyC DB2 Inventory	db2	INVENTORY	Draft	Running	Aug 19
CompanyC DB2 Ordered Items	db2	ORDER_ITEMS	Draft	Running	Aug 19
CompanyC DB2 Orders	db2	ORDERS	Draft	Running	Aug 24

Figure 9-16 Connect & Compose landing page with API listing

4. Give the composite API a name, and click **ADD A COMPOSITION**. See Figure 9-17.



APIs

Create a Composition API

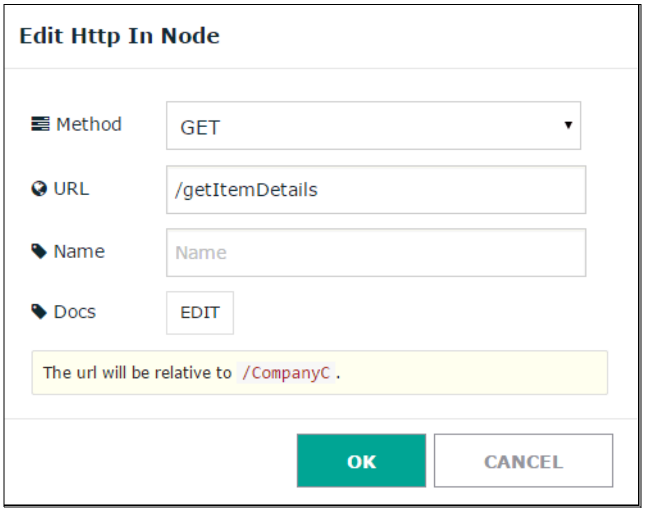
Start creating your API by providing name and description. Then add a composition.

API Name: CompanyC

Description:

Figure 9-17 Create a Composition API

5. A Node-RED composition editor starts. To put together the two APIs that just created, follow the next steps. To start with a REST endpoint, pull the *http* node from input, and *http response* from output on to the flow editor.
6. Double-click the **http node**, and you can see a panel to configure the node, as shown in Figure 9-18.



Edit Http In Node

Method: GET

URL: /getItemDetails

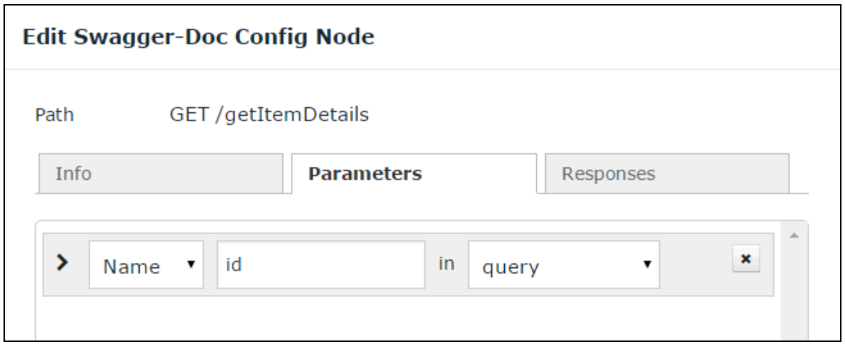
Name: Name

Docs: EDIT

The url will be relative to /CompanyC .

Figure 9-18 Configuration panel for http node

7. After completing the wanted URL, click **EDIT** next to Docs. As shown in Figure 9-19, you can configure the Swagger-Doc for the node. For our scenario, we need to receive the *id* from the query parameter.



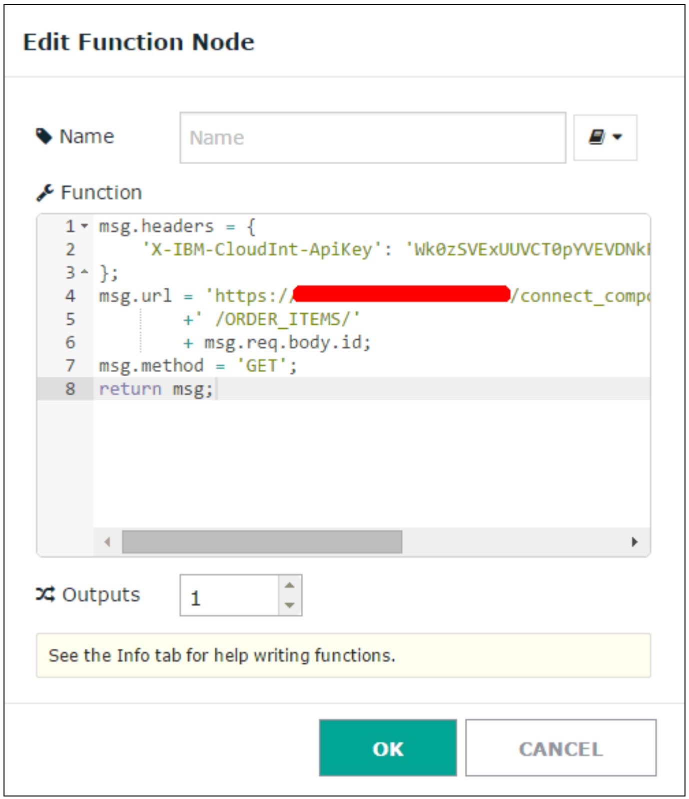
Edit Swagger-Doc Config Node

Path GET /getItemDetails

Name ▾ id in query ▾

Figure 9-19 Swagger-Doc configuration for the http node

8. To use the Ordered Items API that we created, pull *http request* node from under “Function”. To allow advanced configuration, pull a *function* node under the “Function” category. As shown in Figure 9-20, set the API key in the header.



Edit Function Node

Name: Name

Function

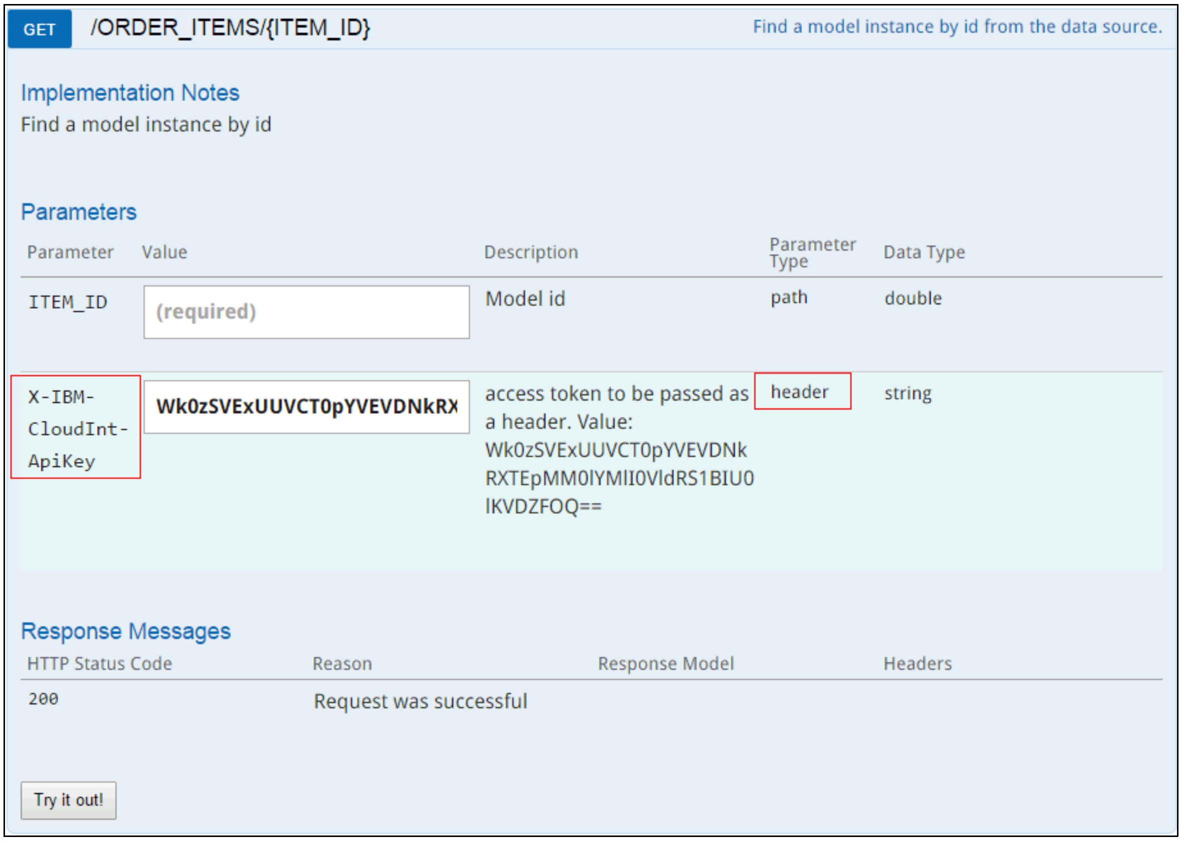
```
msg.headers = {
    'X-IBM-CloudInt-ApiKey': 'Wk0zSVExUUVCT0pYVEVDNk
};
msg.url = 'https:/[redacted]/connect_comp
        +' /ORDER_ITEMS/'
        + msg.req.body.id;
msg.method = 'GET';
return msg;
```

Outputs 1

See the Info tab for help writing functions.

Figure 9-20 Set header, url, and method with a Function node

The specific header key name can be found in the swagger document shown in Figure 9-21. The url consists of the base url, the base path to the endpoint, and item ID, which we obtain from the original request.



GET /ORDER_ITEMS/{ITEM_ID} Find a model instance by id from the data source.

Implementation Notes

Find a model instance by id

Parameters

Parameter	Value	Description	Parameter Type	Data Type
ITEM_ID	(required)	Model id	path	double
X-IBM-CloudInt-ApiKey	Wk0zSVExUUVCT0pYVEVDNkRX	access token to be passed as a header. Value: Wk0zSVExUUVCT0pYVEVDNkRXTEpMM0lYMlI0VldRS1BIU0lKVDZFOQ==	header	string

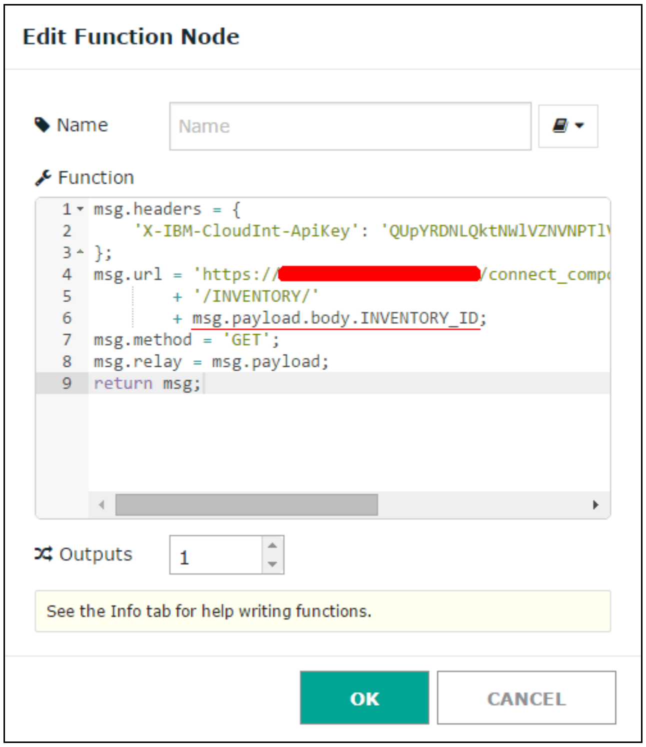
Response Messages

HTTP Status Code	Reason	Response Model	Headers
200	Request was successful		

Figure 9-21 Swagger document

9. Connect the initial http node to the function node, then to the http request.

10. Next, we use the *INVENTORY_ID* from the previous request to query for inventory details. Figure 9-22 shows how we get the *INVENTORY_ID* from a previous call. We also want to keep a record of the payload, and combine it with the payload from the next call for the final response.



```
msg.headers = {
    'X-IBM-CloudInt-ApiKey': 'QUpYRDNLQktNWlVZNVNPTl
};
msg.url = 'https://[redacted]/connect_comp
        + '/INVENTORY/'
        + msg.payload.body.INVENTORY_ID;
msg.method = 'GET';
msg.relay = msg.payload;
return msg;
```

1

See the Info tab for help writing functions.

Figure 9-22 Use INVENTORY_ID from previous request for the next request to Inventory API

11. Again, we drag another *http request* node and place after the *function* node.

12. Lastly, we use a functional node to merge two responses. See Figure 9-23 for the final flow graph.

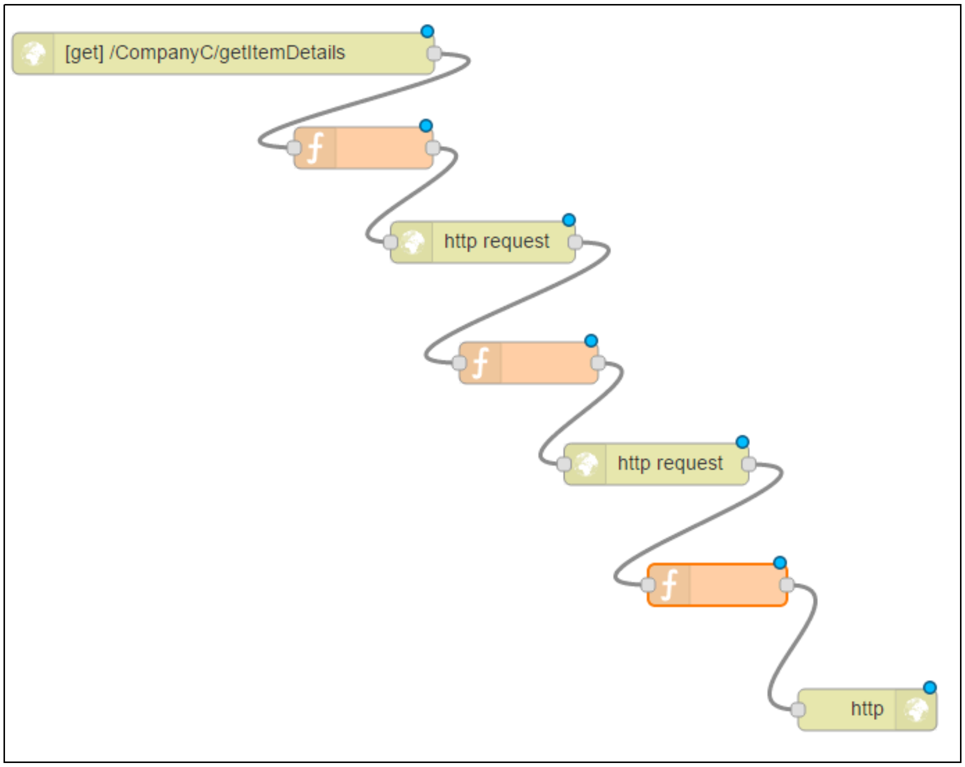


Figure 9-23 Full flow to get full item details

9.2.5 Using API composition with Google Directions and email service

When an employee is done loading the ordered goods, we need to update the order, find directions, and send an email to the customer with an estimated arrival time. Figure 9-24 shows the flow that we composed:

1. We start with an HTTP POST request that we call to pass in the order details.
2. On the top lane, we modify the payload and configure the request to make a call to the order's API's PUT (update) endpoint. Because the employee does not need the response, we just record it with a debug node.
3. On the bottom lane, we pull out the address, and form it into a proper payload to use Google Directions node, which in turn returns directions and estimated amount of time to reach the destination. We pass the full response from Google Directions service back to our application, and parse the estimated time to create an email for the customer.

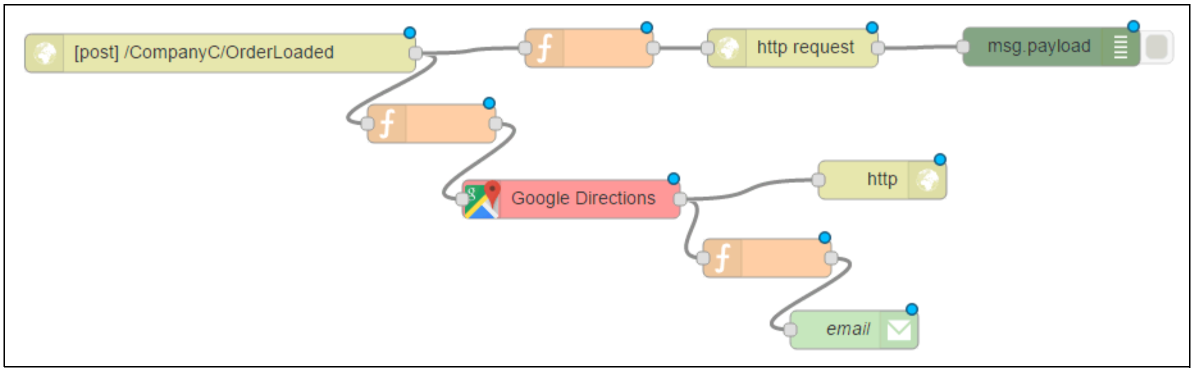


Figure 9-24 Composition flow for OrderLoaded endpoint

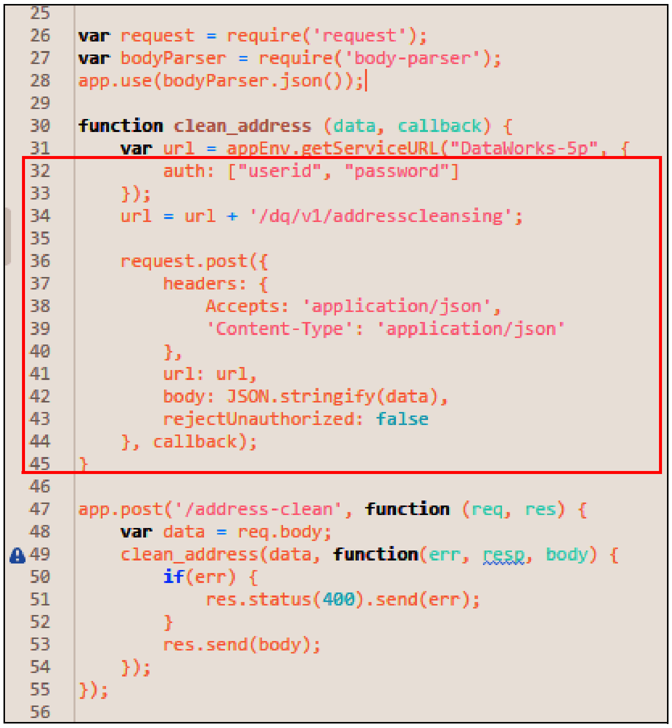
9.2.6 Using DataWorks for address cleansing

DataWorks service provides an Address Cleansing API that puts addresses in CASS format, which is useful for mailing future promotions.

CASS: CASS stands for Coding Accuracy Support System. It enables the United States Postal Service (USPS) to evaluate the accuracy of software that corrects and matches street addresses.

The following section describes how to integrate the Address Cleansing API to the application we created earlier:

1. First, in *package.json,* under dependencies, add *request* and *body-parser* modules.
2. Next, in *app.js,* after the *app* variable has been declared, get the two modules from Step 1, and use the body-parser middleware as shown in lines 26 - 28 in Figure 9-25.
3. Lines 32 - 45 in Figure 9-25 show the function that performs the call to DataWorks's API.



```
25
26  var request = require('request');
27  var bodyParser = require('body-parser');
28  app.use(bodyParser.json());
29
30  function clean_address (data, callback) {
31      var url = appEnv.getServiceURL("DataWorks-5p", {
32          auth: ["userid", "password"]
33      });
34      url = url + '/dq/v1/addresscleansing';
35
36      request.post({
37          headers: {
38              Accepts: 'application/json',
39              'Content-Type': 'application/json'
40          },
41          url: url,
42          body: JSON.stringify(data),
43          rejectUnauthorized: false
44      }, callback);
45  }
46
47  app.post('/address-clean', function (req, res) {
48      var data = req.body;
49      clean_address(data, function(err, resp, body) {
50          if(err) {
51              res.status(400).send(err);
52          }
53          res.send(body);
54      });
55  });
56
```

Figure 9-25 Code to integrate DataWorks API

4. Lines 47 - 55 implement a route that we can use to test that the function in Step 3 is working.
5. With this function, we can connect to another database table to store the customer's sanitized address information.

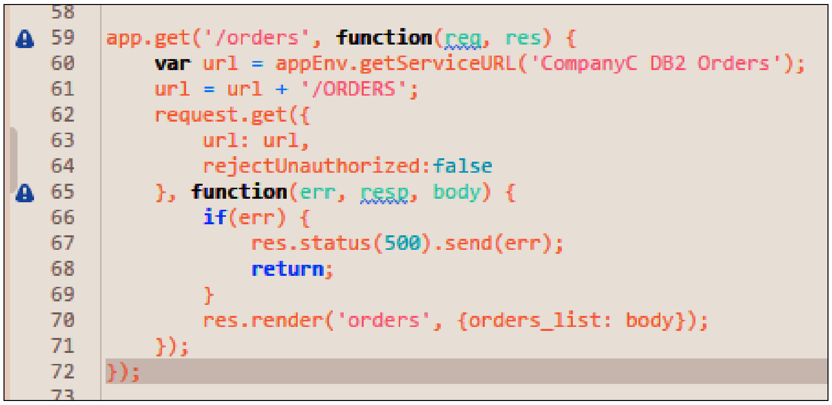
9.2.7 Creating a page to list all the orders

Routes of an app are used to serve contents to the user. We created many APIs to be used for this application. Here, we show the detailed steps to render a page with the list of orders. This list acts as the first step for an employee from CompanyC to start the delivery process. The rest of the application can be implemented in similarly:

1. Bind Orders API to app:
   a. On the landing page, find the API that is connected to the Orders table, click the **gear icon**, and select share to Bluemix.
   b. On Bluemix Dashboard, we see among Services listings, our API *CompanyC DB2 Orders*.
   c. Click **CompanyC** (our application), select **BIND A SERVICE OR API**. Select **CompanyC DB2 Orders** (our API), then click **ADD**.

d. The API credentials then become available in VCAP_SERVICES, and we can access it by using *appEnv.getServiceURL("CompanyC DB2 Orders").*

2. Create a route to serve the page. Figure 9-26 shows the route that we need to display our Orders listing page.



```
58
59 app.get('/orders', function(req, res) {
60     var url = appEnv.getServiceURL('CompanyC DB2 Orders');
61     url = url + '/ORDERS';
62     request.get({
63         url: url,
64         rejectUnauthorized:false
65     }, function(err, resp, body) {
66         if(err) {
67             res.status(500).send(err);
68             return;
69         }
70         res.render('orders', {orders_list: body});
71     });
72 });
73
```

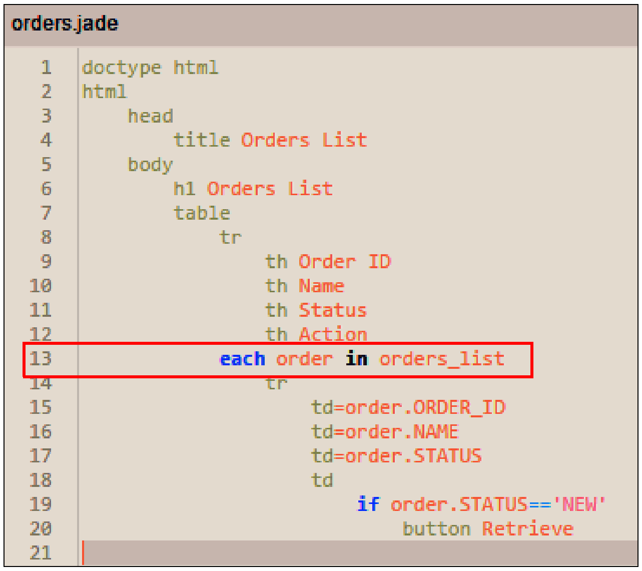
Figure 9-26 Route to display list of orders

a. Use Jade templates to populate the list of orders. Jade (jade-lang.com) is one of the popular template engines for NodeJS. To use Jade, first add the jade module to package.json.

b. Then, in app.js, configure the app to use jade as the view engine: `app.set(‘view engine’, ‘jade’);`

c. You can also set the folder from which views files are stored, views folder in this case, by using the following command: `app.set('views', __dirname + '/views');`

Figure 9-27 shows the basic template that corresponds to line 70 shown in Figure 9-26. Notice that the **`orders_list`** variable with the list of orders was passed in for rendering. The variable then became available in the template, as shown in line 13 in Figure 9-27.

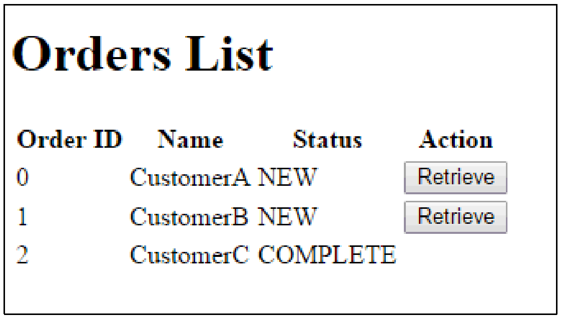


orders.jade

```
1  doctype html
2  html
3      head
4          title Orders List
5      body
6          h1 Orders List
7          table
8              tr
9                  th Order ID
10                 th Name
11                 th Status
12                 th Action
13             each order in orders_list
14                 tr
15                     td=order.ORDER_ID
16                     td=order.NAME
17                     td=order.STATUS
18                     td
19                         if order.STATUS=='NEW'
20                             button Retrieve
21
```

Figure 9-27 Jade template code for order listing

From looking at Figure 9-28 on page 240, you can see the page at `http://companyc.mybluemix.net/orders`. Notice how Jade handles conditional statements and iterative statements to accommodate impracticability of the data.



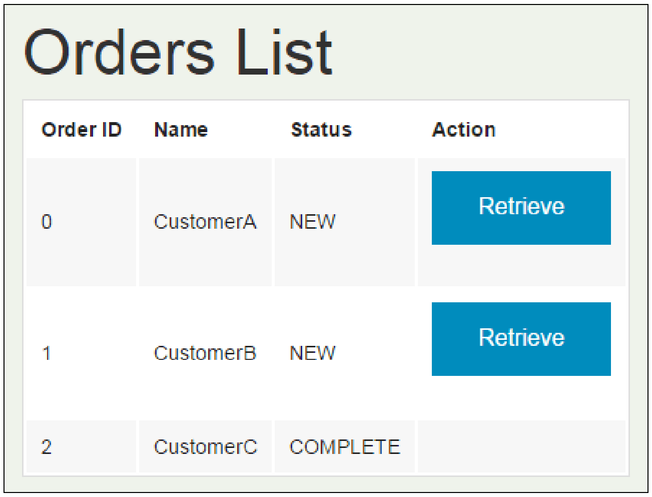
Orders List

Order ID	Name	Status	Action
0	CustomerA	NEW	Retrieve
1	CustomerB	NEW	Retrieve
2	CustomerC	COMPLETE	

Figure 9-28 Rendered view for the orders list

3. Use Cascading Style Sheets (CSS) to improve the visuals. Lastly, for the app to be visually pleasant, we use CSS stylesheets to touch up the page. To apply the styles in Figure 9-29, we used a third-party CSS style sheet from the ZURB Foundation:

   `http://foundation.zurb.com`



Orders List

Order ID	Name	Status	Action
0	CustomerA	NEW	Retrieve
1	CustomerB	NEW	Retrieve
2	CustomerC	COMPLETE	

Figure 9-29 Styled page for orders list

This concludes a simple implementation of an application using hybrid cloud services provided in Bluemix.

A

Additional material

This book refers to additional material that can be downloaded from the Internet as described in the following sections.

Locating the web material

The web material associated with this book is available in softcopy on the Internet from the IBM Redbooks web server. Point your browser at:

ftp://www.redbooks.ibm.com/redbooks/SG248277

Alternatively, you can go to the IBM Redbooks website at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG248277.

Using the web material

The additional web material that accompanies this book includes the following files:

File name	*Description*
SG248277.zip	CSV file and Java program used in the Watson Analytics scenario in Chapter 9, “Mobile hybrid scenario: Secure Gateway, Connect & Compose, and DataWorks” on page 219

System requirements for downloading the web material

The web material requires the following system configuration:

Hard disk space:	5 MB minimum
Operating System:	Windows/Linux

© Copyright IBM Corp. 2015. All rights reserved.

Downloading and extracting the web material

Create a subdirectory (folder) on your workstation, and extract the contents of the web material `.zip` file into this folder.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks publications

The following IBM Redbooks publications provide additional information about the topic in this document. Note that publications referenced in this list might be available in softcopy only.

- *Microservices from Theory to Practice: Creating Applications in IBM Bluemix Using the Microservices Approach*, SG24-8275
- *Creating Hybrid Clouds with IBM Bluemix Integration Services*, REDP-5270

You can search for, view, download or order this document and other IBM Redbooks materials at the following website:

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- IBM API Management product page:

  http://www.ibm.com/software/products/en/api-management
- IBM Cast Iron product page:

  http://www.ibm.com/software/products/en/castiron-cloud-integration
- IBM Security QRadar SIEM product page:

  http://www.ibm.com/software/products/en/qradar-siem
- Airport Status service published by the Federal Aviation Administration (FAA):

  http://services.faa.gov/docs/services/airport/#airportStatus
- WebSphere Developer Tools:

  http://www.ibm.com/support/knowledgecenter/SSHR6W_8.5.5/com.ibm.websphere.wdt.doc/topics/welcome_wdt.htm
- IBM Bluemix Tools:

  https://www.ng.bluemix.net/docs/manageapps/eclipsetools/eclipsetools.html
- Installing Docker on Ubuntu:

  https://docs.docker.com/installation/ubuntulinux
- Eclipse documentation - Current Release:

  http://help.eclipse.org/mars/topic/org.eclipse.wst.server.ui.doc.user/topics/twaddprj.html

© Copyright IBM Corp. 2015. All rights reserved.

- MySQL Download Connector/J page:

  https://dev.mysql.com/downloads/connector/j

- Docker commands:

  https://docs.docker.com/reference/commandline/cli

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Redbooks

Hybrid Cloud Data and API Integration

SG24-8277-00

ISBN 0738441090

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

SG24-8277-00

ISBN 0738441090

Printed in U.S.A.

Get connected

Redbooks

ibm.com/redbooks