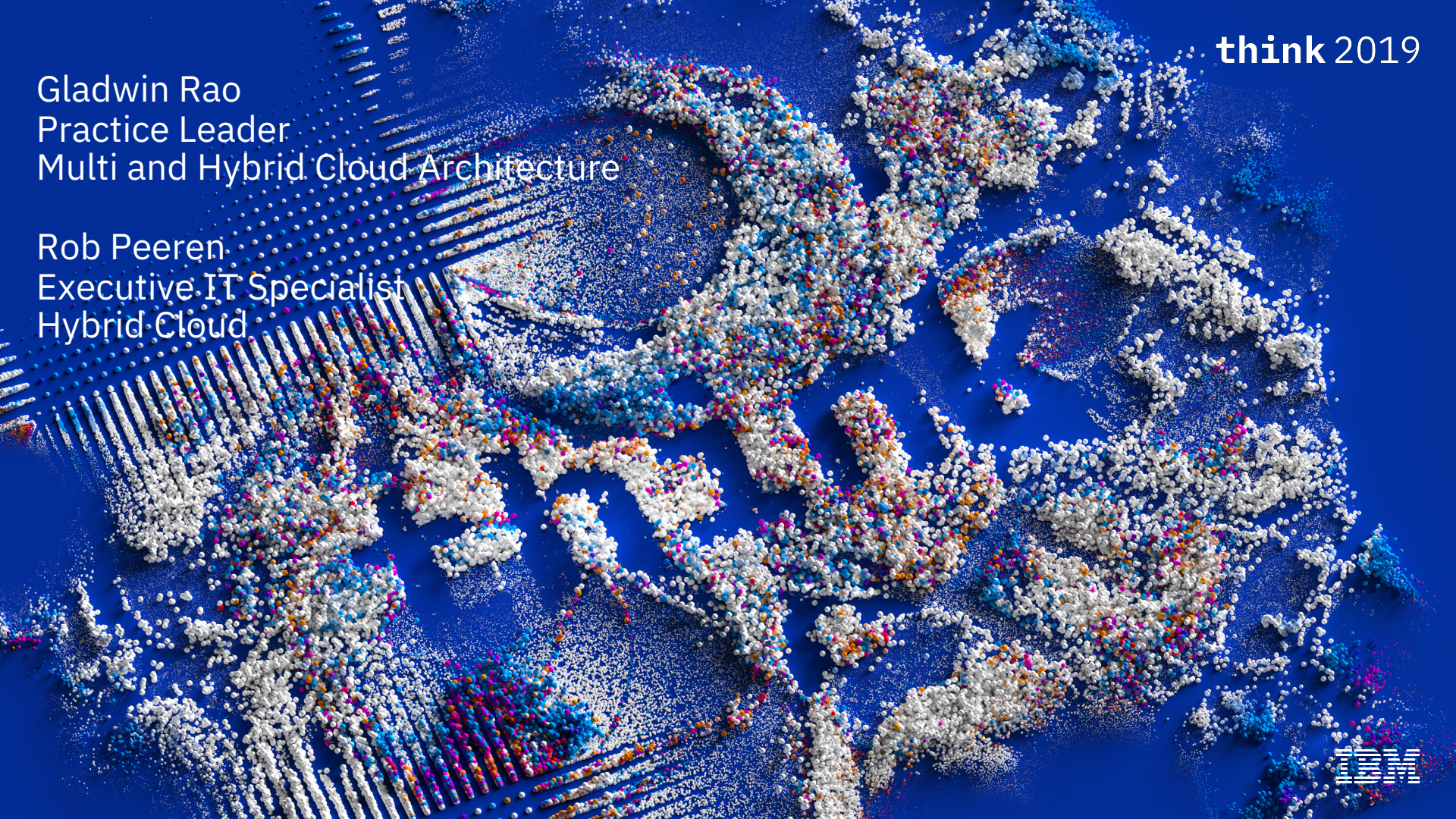


think 2019

Gladwin Rao
Practice Leader
Multi and Hybrid Cloud Architecture

Rob Peeren
Executive IT Specialist
Hybrid Cloud

IBM

Agenda

Part One **15**
Why Modernization
Approaches to Modernization
Key Practices

Part Two **15**
Practical demonstration of Modernization
Tools to accelerate

Enterprise Applications that have been migrated to the cloud

20%

Applications can propel you forward or hold you back from innovation and agility. Organizations modernizing their applications are about twice as successful at digital transformation. But today, only 20% of workloads have been modernized and moved to the cloud.

Why Modernization Matters

Applications that will need to be modernized to take advantage of Cloud Computing

80%

APIs, microservices and containers are becoming the standard, the remaining 80% of workloads require new thinking about cloud-native and beyond.
As enterprises evaluate existing infrastructure and investments, they require new skills, more flexible and open architectures, developer-first strategies, advanced integration, and governance.

Enterprise
Cloud Journey

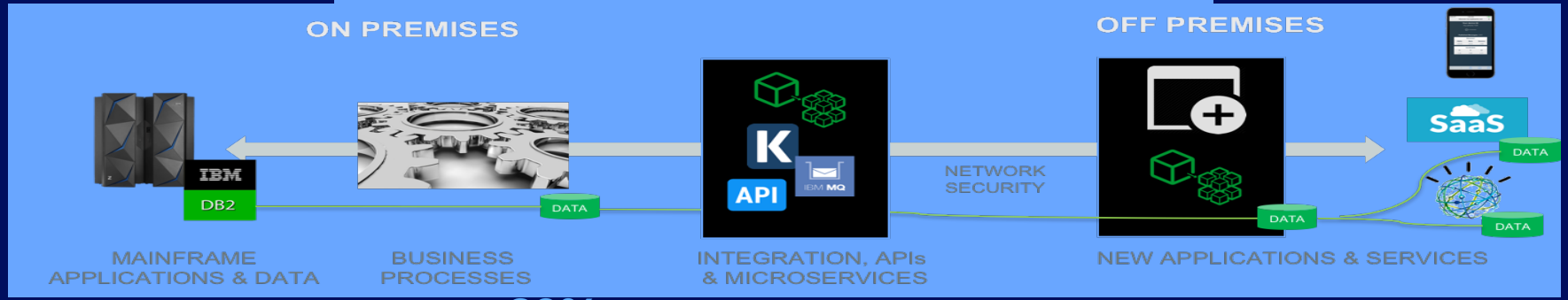
Why are clients embarking on a cloud journey?

Enabler for Innovation and Agility

Why is Application Modernization so difficult?

Understanding the enterprise cloud adoption challenges and opportunity to modernize…

Enterprise Technology Landscape



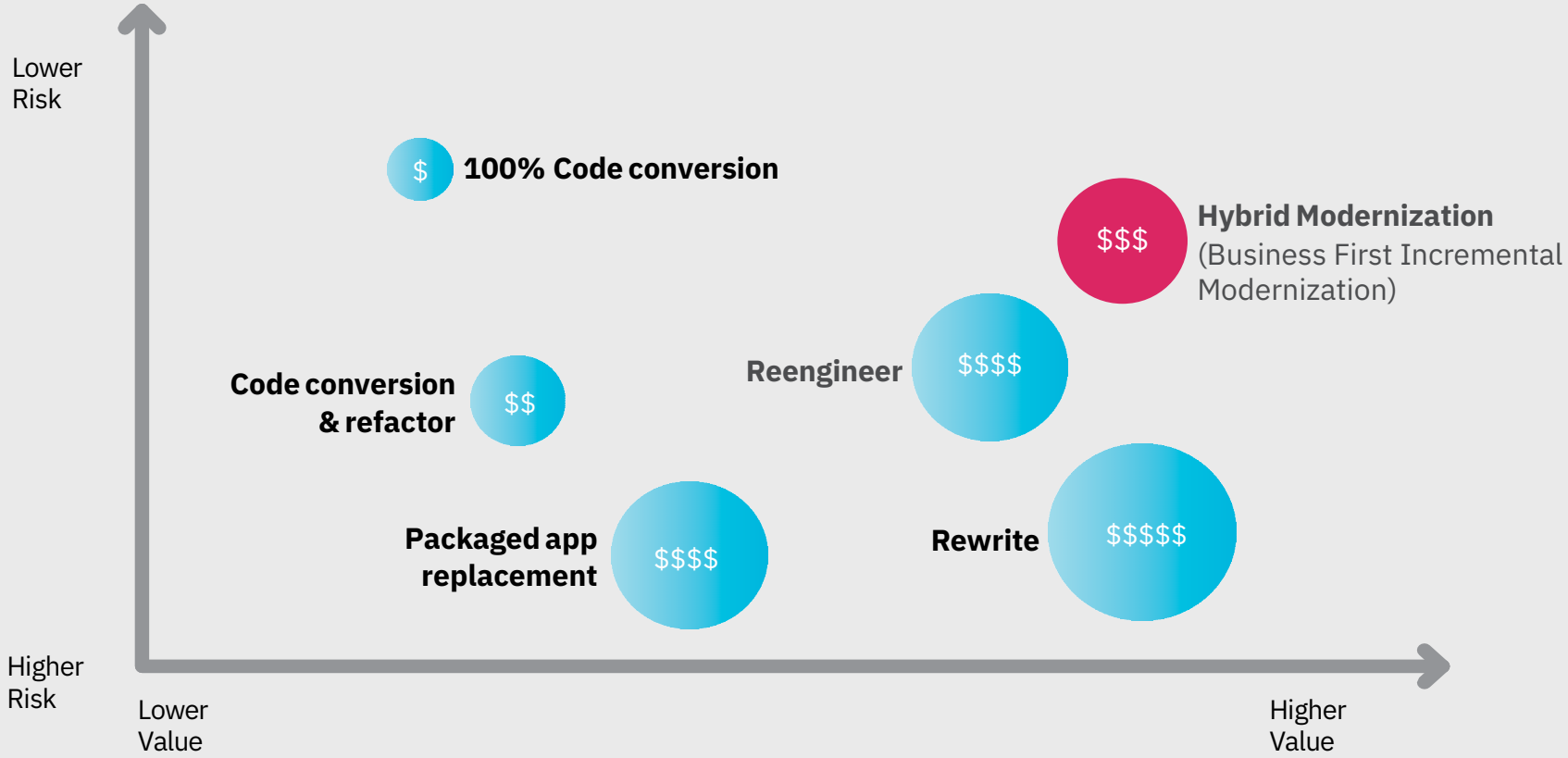
80% of enterprise workloads have yet to migrate to the cloud

Cloud Services Scope

- Legacy Transformation
- Application and Data Integration
- Digital & Cloud Native Development
- Hybrid Platform Services (DevOps, Multi-cloud Management and Operations)

Hybrid | **Multi-cloud** | **Open** | **Secure** | **Management**

Modernization can take many forms



Application Modernization

How do we approach Application Modernization?

Journey to Cloud

IBM Services for Cloud Strategy (*Advise on Cloud*)	IBM Services for Cloud Migration (*Move to Cloud*)	IBM Services for Cloud Development (*Build for Cloud*)	IBM Services for Cloud Management (*Manage on Cloud*)
• Enterprise Cloud Strategy • Cloud Architecture Design • Cloud Security and Compliance Strategy	• Cloud Migration • **Cloud Modernization** ✓ Legacy ✓ Data ✓ Enterprise Applications	• **Cloud Application Development and DevOps** • **Multicloud Infrastructure Development** • **Enterprise Application Integration**	• DevOps Monitoring and Management • Enterprise Application Management • Managed Multicloud Services Platform • Multicloud Management Services • Managed Infrastructure as a Service • Cloud Security and Resiliency

Our view of Application Modernization

Move out of Data Centre& Infrastructure Management

Reduce Tech Debt & Move the Core

Transform IT & Culture

Lift & Shift → **Remediate / Re-platform** → **Re-factor** → **Re-architect** → **Cloud Native**

20%

In pockets largely driven by LoBs

80%

The **real work** is in moving existing core applications to the cloud in a way that maximizes the benefits of Cloud

Lift & Shift

Move to compatible workload to Cloud for:

- DataCentre Savings
- Elastic or seasonal workloads
- Begin the Cloud Journey with quick results
- Begin culture changes

Remediate

Remediate Application while migrating to Cloud:

- Reduce technical debt by eliminating EoL components
- Application level migrations vs workload migration
- Introduce next gen elements such as Containers

Refactor

Refactor Applications while migrating to Cloud:

- Componentize application into foundational, strategic and innovative
- Optimize for cloud
- Embed security into code refactoring
- Applications begin to be more cloud aware

Re-architect

Refactor Applications to take advantage of Cloud:

- Leverage the ecosystems where it makes sense
- Re-architect your application for resiliency and scalability
- Applications to be more cloud native

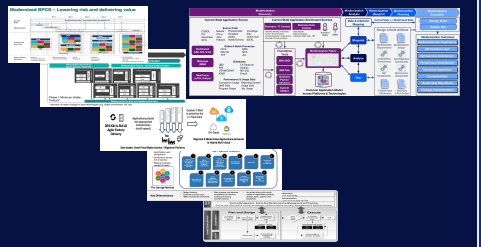
Re-Imagine

Re-Imagine your business and applications and

- Utilize Garage Methods to rewrite components aligned to your business
- Optimize your AppDev & Operations for extreme performance
- Built on Cloud differentiating applications

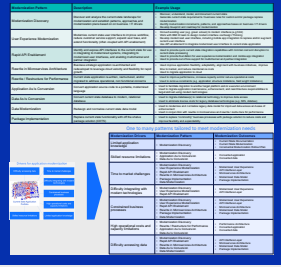
IBM Modernization Foundation Assets

Modernization Toolkit



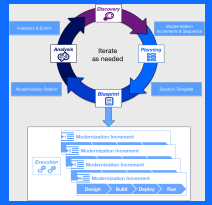
- Tailored integration of discovery and analysis tools for unique combination of current state technologies
- Centralized platform and technology agnostic application model for integrated discovery
- Customizable for addressing unique patterns and technologies
- Data science and analytics based providing scalability

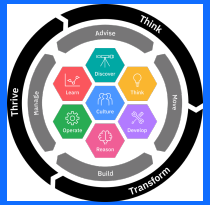
Modernization Patterns



- Common patterns to address any modernization drivers
- Business value optimized by tailoring one to many patterns to each modernization increment
- Can be applied for any target solution technology and platform

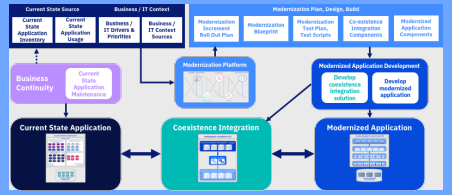
Modernization Method





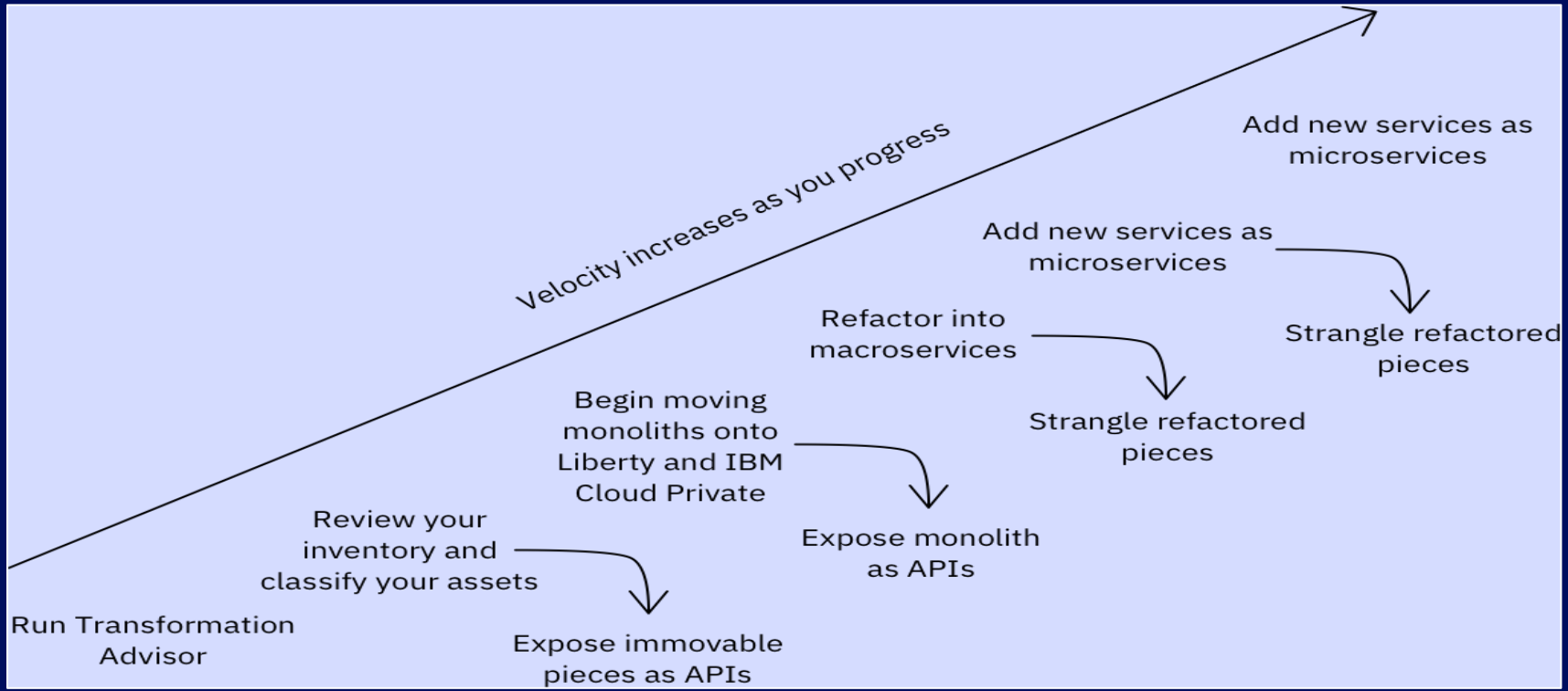
- Iterative, agile based modernization method integrated with modernization platform and patterns
- Method tailored to execute on required modernization patterns and outcomes
- Modernized functionality optimized to maximize business value and minimize technical complexity
- Increment discovery designed to drive scalable agile development and roll out

Coexistence Architecture



- Coexistence architecture isolates modernized state from current state
- Minimizes modifications to current state for supporting modernization roll out
- Common coexistence integration patterns tailored to for each increment
- Manages controlled decommission of current state
- Deployable on any technology

Modernization is a Journey



https://www.ibm.com/cloud/garage/architectures/application-modernization/overview

World's largest airline: Bottom-up business transformation

Client context

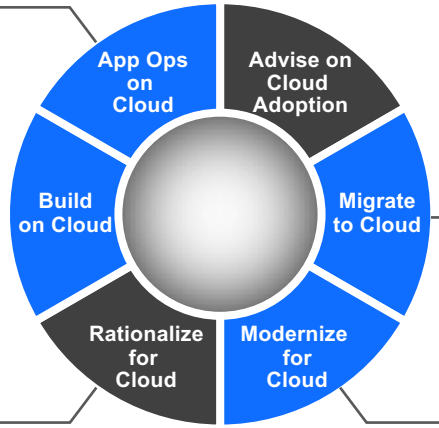
Client had three main asks:

1. Support migration of our data center to the IBM cloud
2. Move our monolithic applications to the cloud
3. Transform our organization into a high performing software delivery organization

IBM Solution

IBM Cloud

Implement End to End unified ITSM (IT Service Management) solution integrating both the customer's management and monitoring systems and IBM's leveraging IBM Cloud Innovate AppOps on Cloud model across multiple clouds and on premises systems.



Step 1
Migrate customer facing/business critical applications from 2 legacy data centers to IBM Cloud

Step 2
Incrementally modernize existing legacy applications to Cloud Native and Microservices using the Strangler Pattern and IBM Garage Method

Build new products using the IBM Garage Method including Design Thinking, Test Driven Development, Pair Programming, and SRE practices to deliver innovations at speed in an agile way

Results

6
Months to migrate 500+ servers from 2 Data Centers to IBM Cloud with no major outages

50%
Faster to get a new idea into the hands of customers using the IBM Garage Method compared to traditional approaches

1,500
Client applications targeted for cloud

©2019 IBM Corporation

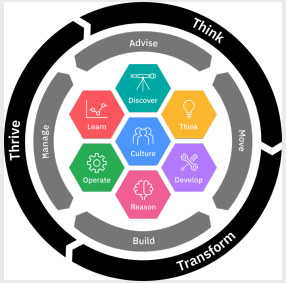
Application Modernization and Journey to Cloud

What are our credentials in this space and why IBM?

IBM Garage Method for Cloud powers all our cloud engagements

We have industrialized a common approach 'IBM Garage Method for Cloud' that is based on agile DevOps principles

IBM Garage Method for Cloud



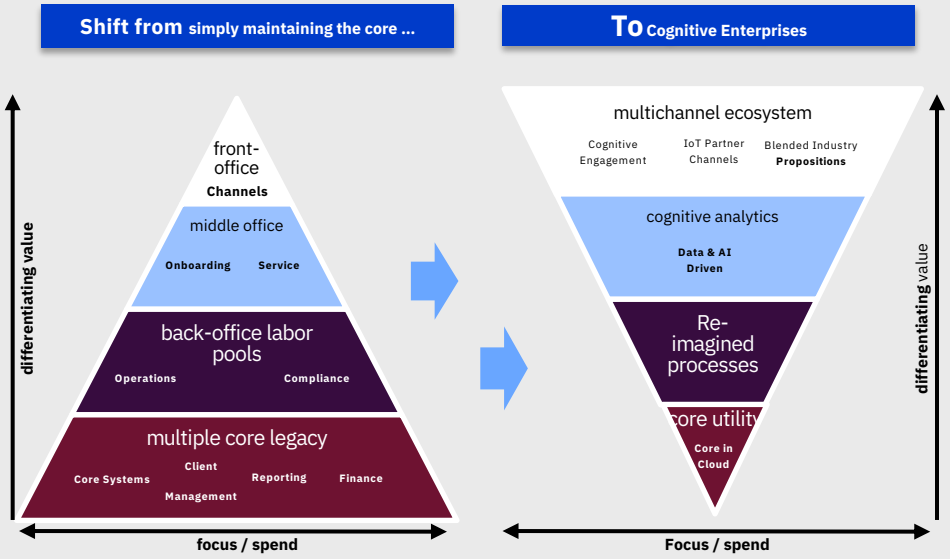
Garages deliver innovative, rapid transformation and lasting culture change

- Red Hat OpenShift
- Azure
- Amazon Web Services
- Google Cloud Platform
- IBM Cloud

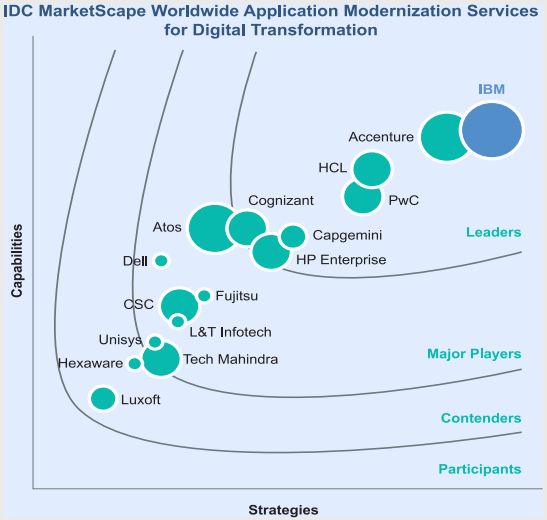
Advise	Move	Build	Manage
E2E advice, design, road mapping, and architectural services to help you navigate the complexity of your multi-cloud journey	Our Garage Method for Cloud employed to assist and accelerate execution; assisted by configurable and automated migration tooling supporting multiple cloud platforms Also execute the priorities to sunset applications and associated infrastructure through unique set of capabilities	High speed Application and Platform Engineering to build new capabilities and/or transform existing application and data portfolios to hybrid multi-cloud @scale	Multi/hybrid cloud DevOps and business service operation delivery through cloud platform architecture and engineering services
70+ IT operating and organizational engagements in last 2 years	75+ **modernization** engagements in the last two years	868 Build on Cloud engagements	45+ App Ops engagements in the last 2 years
1,000+ IBMers engaged in cloud advisory and architecture services	~3,000 IBMers engaged in migration activities	~8,000 IBMers engaged in Build for Cloud activities	1,200 IBMers engaged in app ops activities

1000+ Multi-cloud certified practitioners

IBM Leads this space - don’t just take our word for it



To maintain efficiency you will need to leverage cloud in a big way to transform your core capabilities, lower costs and shift focus to growth related imperatives



*Source IDC MarketScape WW App Mod Services 2015

IBM is well positioned to help clients in their Cloud transformation journey

1. In-depth understanding of the Application Landscape
2. Unparalleled expertise in managed cloud and infrastructure
3. Legacy and Middleware Systems Leadership
4. Red Hat – open, portable and most popular hybrid cloud choice

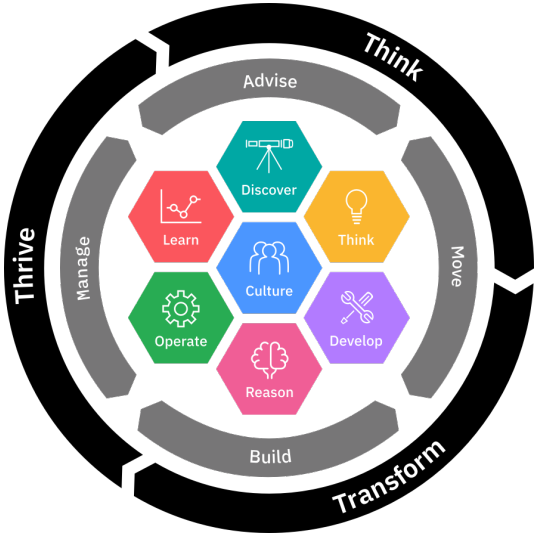
Application Modernization

Illustrate of an application modernization patterns?

Our Multicloud Migration & Modernization offering is enriched with Cloud Innovate™ E2E methodology

Cloud Innovate™ is End to End

- Full lifecycle
- Vertically integrated solution view
- Hybrid cloud transformation journey
- Secure lifecycle- strategy to operations



Cloud Innovate ensures

- Consistency
- Assimilation of best practices & experiences
- Standard set of Tools
- Efficiency factor

Cloud Innovate Methodology brings the IBM way to address Hybrid Cloud journey

IBM Cloud Azure aws Google ORACLE CLOUD

©2019 IBM Corporation

Thank You

Gladwin Rao
Practice Leader - Multi & Hybrid Cloud Architecture
Email: grao@ca.ibm.com

Rob Peeren
Executive IT Specialist - Multi & Hybrid Cloud
Email: robobob@ca.ibm.com

 ©2019 IBM Corporation

Application Modernization and Transformation

A Case-Study

Rob Peeren, B.Sc.

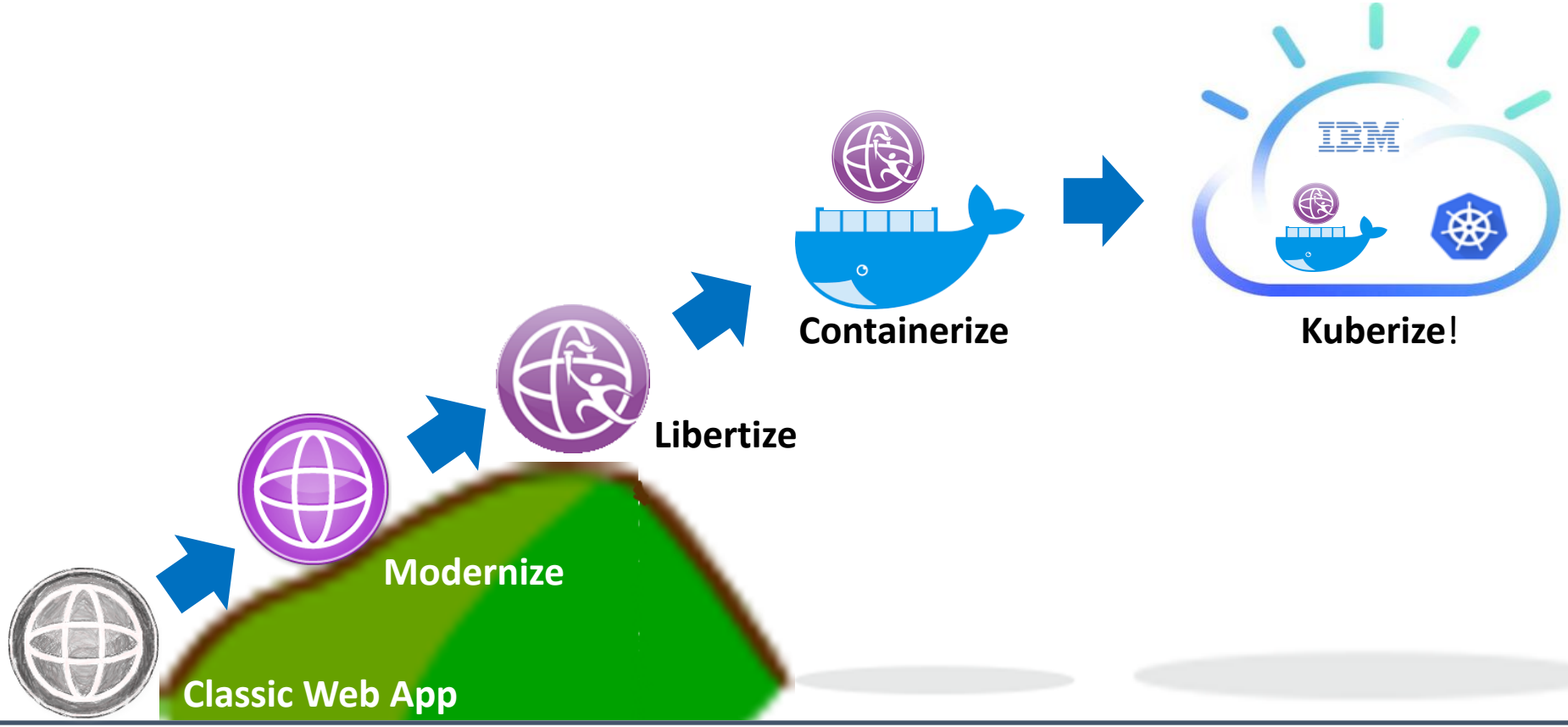
Executive IT Specialist

IBM Canada Ltd.

June 2019

Part 1: Modernization

The Application Modernization Blueprint



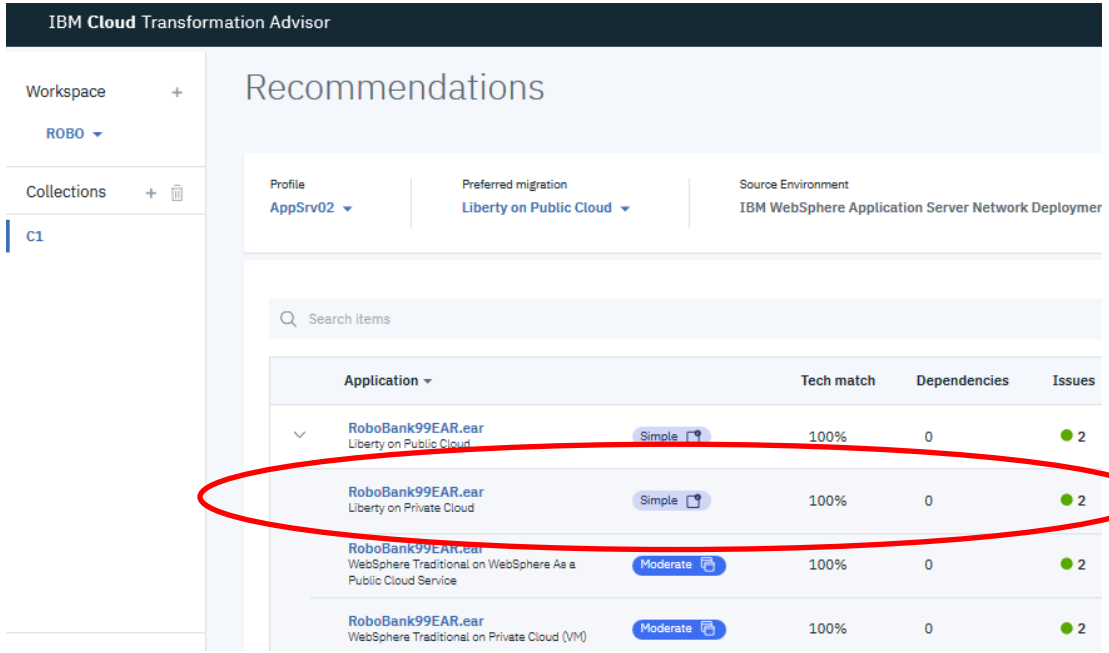
Step 1: Identify the Classic Application to Modernize

Transformation Advisor Identifies Candidates…

The IBM Transformation Advisor crawls through your inventory of JEE applications and generates a detailed report about the effort involved to migrate your applications into containers.

It provides line-by-line recommendations on code that is obsolete or deprecated.

Built by migration specialists over a 10 year span, it helps identify which applications can be easily migrated, which ones need some effort, and which ones would need refactoring.



The 20-Year Old Web Application…

This is an example of a web application built in the late 1990's.

It is a fairly simple JSP/Servlet/JDBC application that is also typical of the types of applications being written back then.

It originally ran on JDK 1.1.6 and IBM WebSphere Application Server 2.0!

TransferBean.java	7/22/1999 2:47 PM
StockTransactionServlet.java	8/19/1999 2:41 PM
StockTransactionBean.java	7/22/1999 2:48 PM
StockHistoryServlet.java	7/22/1999 3:00 PM
StockEngine.java	8/19/1999 2:49 PM
StockAnalysisServlet.java	7/22/1999 2:59 PM
QuoteServlet.java	8/19/1999 2:45 PM
QuoteEngine.java	8/19/1999 11:39 AM
QuoteBean.java	8/19/1999 12:44 PM
MutualFundEngine.java	8/19/1999 11:41 AM
MutualFundBean.java	7/22/1999 2:46 PM
InvestmentSummaryServlet.java	7/22/1999 2:59 PM
InitializeServlet.java	7/22/1999 2:45 PM
FundsTransferServlet.java	7/22/1999 2:59 PM
CustomerBean.java	8/25/1999 4:52 PM
AuthEngine.java	8/19/1999 11:40 AM
AccountSummaryServlet.java	7/22/1999 2:58 PM

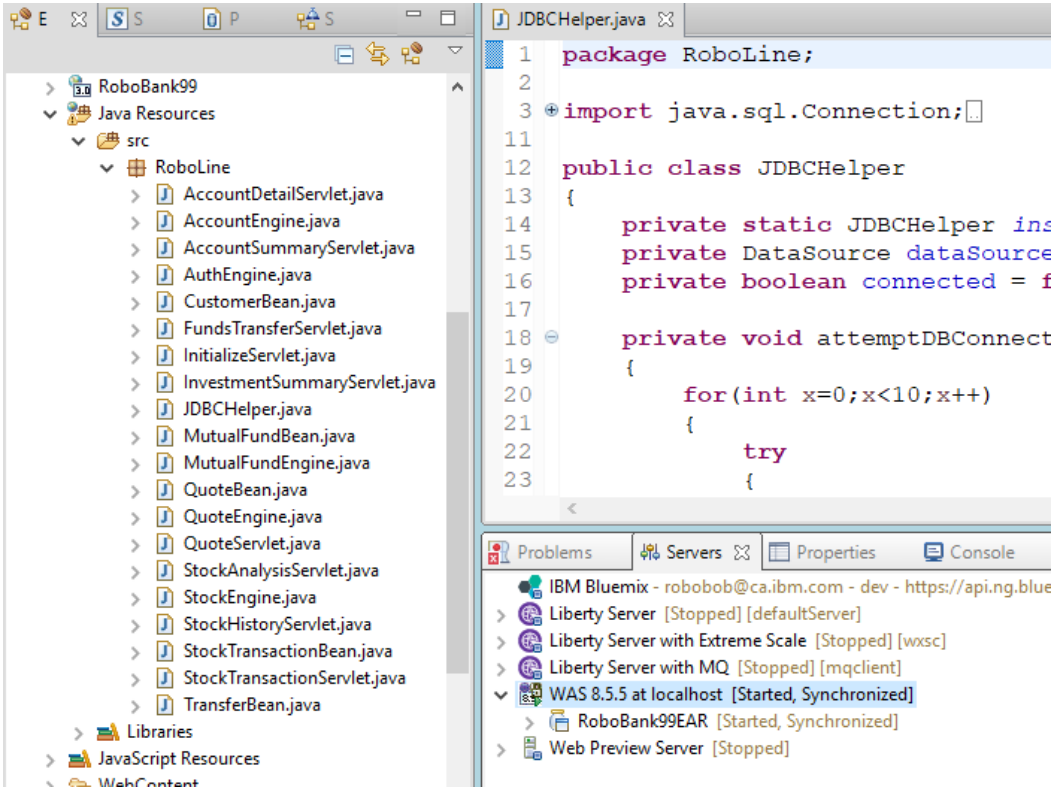
Step 2: Modernize

Import the Project into Modern Tooling…

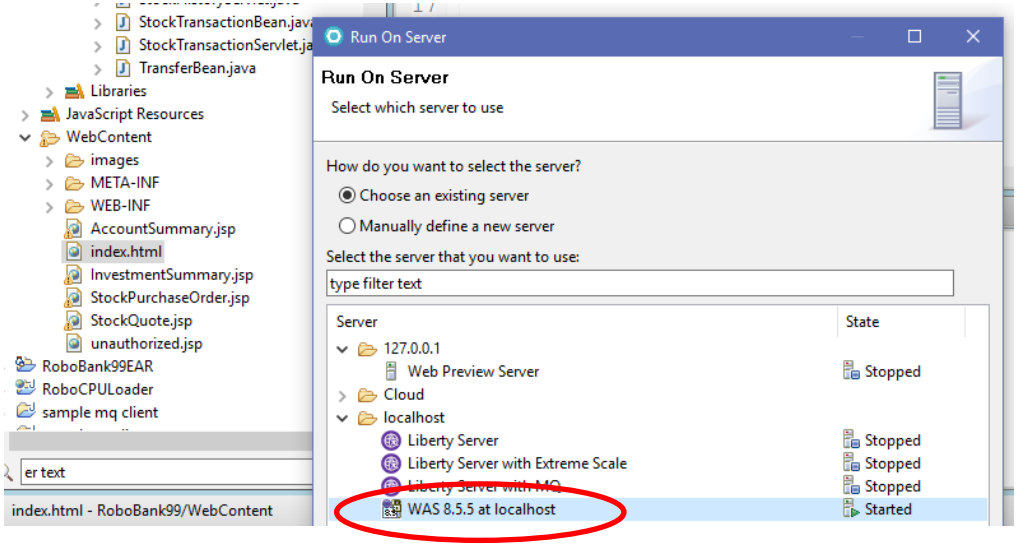
In this example, the original code is imported into Rational Application Developer (RAD) V8.5

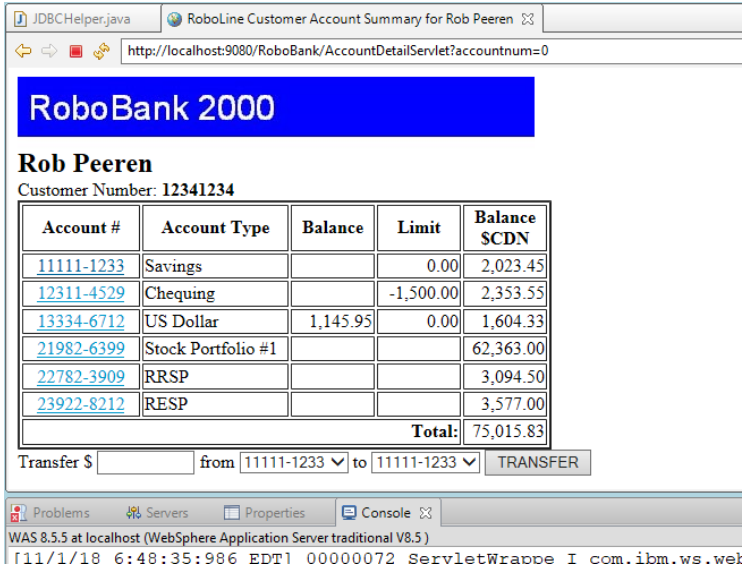
Based on recommendations from the Transformation Advisor, the code is quickly cleaned up by the developer (~1.5 days of effort in this case)

RAD includes a local copy of WebSphere Application Server (WAS) 8.5.5 that the application can be deployed to.



Deploy Updated Project to Modern App Server



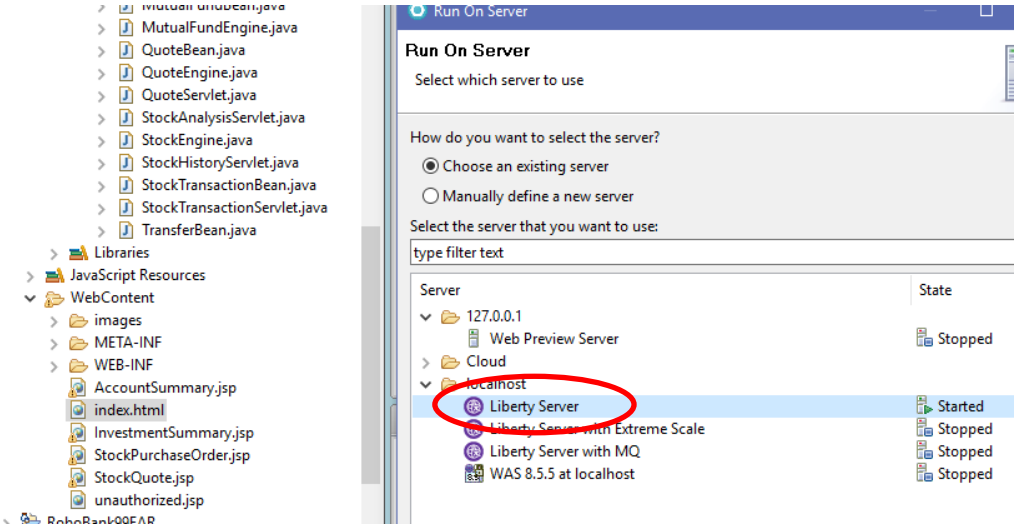


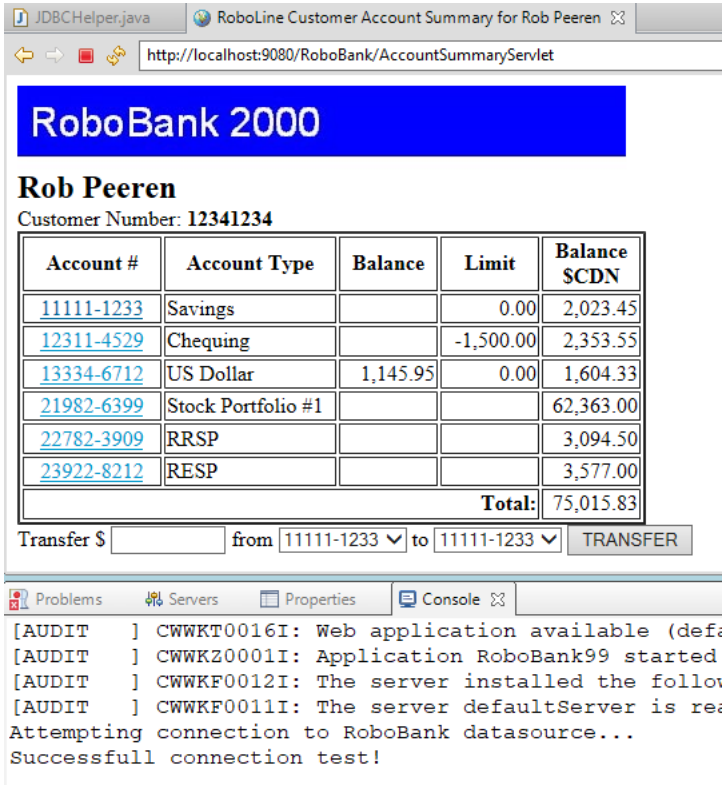
WAS 8.5.5 is made the run target of the application…

And it runs!

Step 3: Libertize

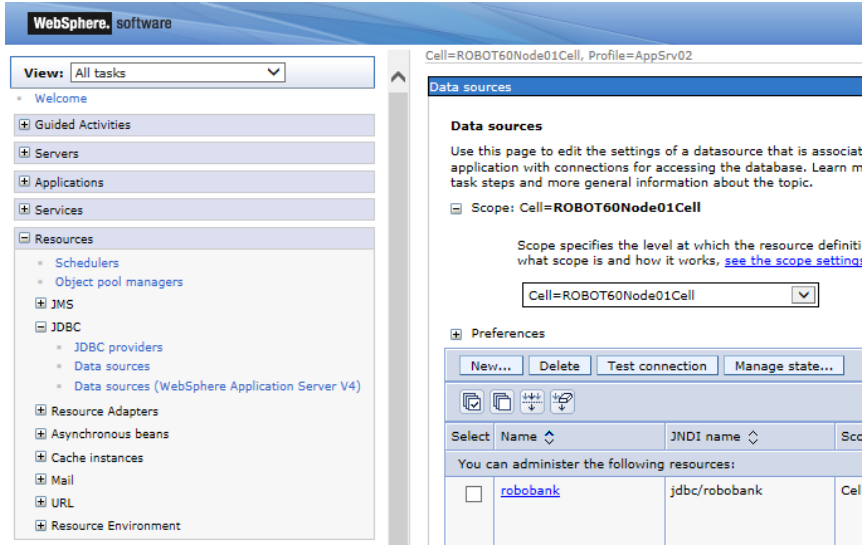
Repeat the Deploy to Liberty Application Server…





Liberty is now made the run target of the application, and it still works! No extra refactoring required.

What’s the Difference?




```xml
<server description="new server">
      <featureManager>
      <feature>jdbc-4.0</feature>
      <feature>jndi-1.0</feature>
      <feature>jsp-2.2</feature>
      <feature>localConnector-1.0</feature>
      <feature>servlet-3.1</feature>
      <feature>json-1.0</feature>
   </featureManager>

   <library id="DB2Lib">
     <fileset dir="${shared.resource.dir}/db2" includes="*.jar"/>
   </library>

   <dataSource jndiName="jdbc/robobank" transactional="true">
      <jdbcDriver libraryRef="DB2Lib"/>
      <properties.db2.jcc currentSchema="DB2" databaseName="robobank"
      <!-- <properties.db2.jcc  enableClientAffinitiesList="1" client
   </dataSource>

   <httpEndpoint httpPort="9080" httpsPort="9443" id="defaultHttpEndp
   <webApplication contextRoot="RoboBank" id="RoboBank99" location="R

   <applicationMonitor updateTrigger="mbean"/>
</server>
```

WebSphere Admin Console

WebSphere Liberty server.xml

Step 4: Containerize

Create Dockerfile

Containerizing an application requires the building of a Dockerfile, which essentially is a manifest of all the assets that are required to run your application within a container.

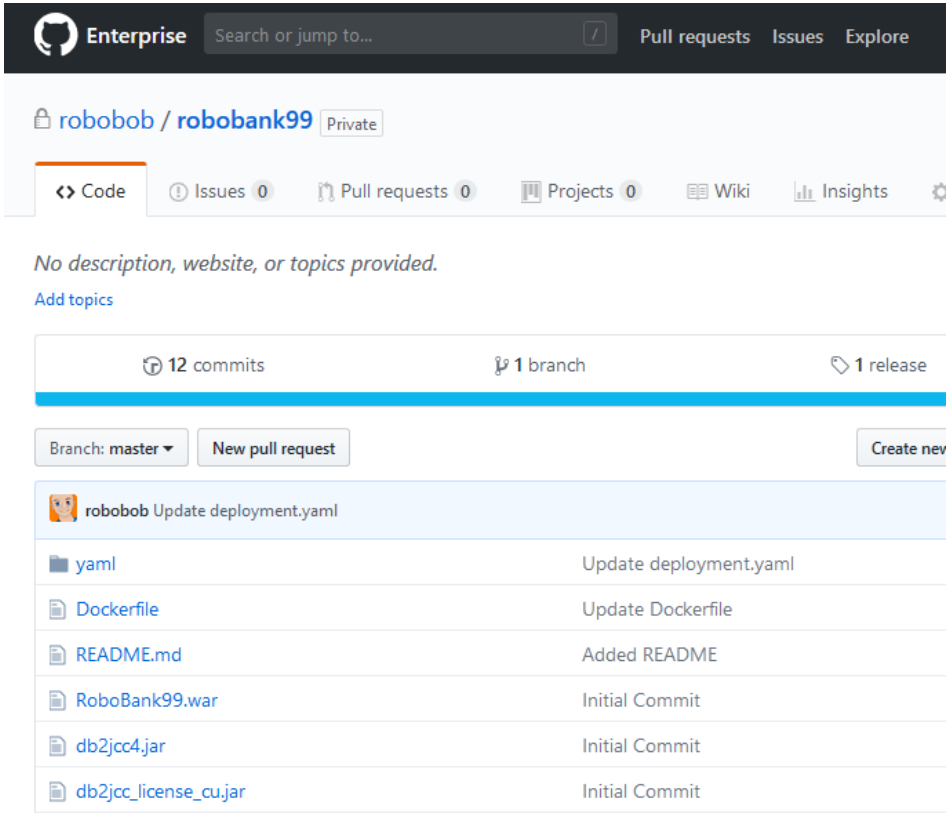
In this case, the web application WAR file, jdbc drivers, and other assets are copied into the container.

```
FROM websphere-liberty:webProfile7

USER root
RUN mkdir /opt/ibm/wlp/usr/shared/resources/db2
COPY runBank.sh /usr/local/bin
RUN chmod +x /usr/local/bin/runBank.sh
COPY db2jcc4.jar /opt/ibm/wlp/usr/shared/resources/db2
COPY db2jcc_license_cu.jar /opt/ibm/wlp/usr/shared/resources/db2
COPY server.xml /opt/ibm/wlp/usr/servers/defaultServer
COPY RoboBank99.war /opt/ibm/wlp/usr/servers/defaultServer/apps
RUN /opt/ibm/wlp/bin/installUtility install defaultServer
USER 1001


EXPOSE 9080 9443

CMD ["/usr/local/bin/runBank.sh"]
```

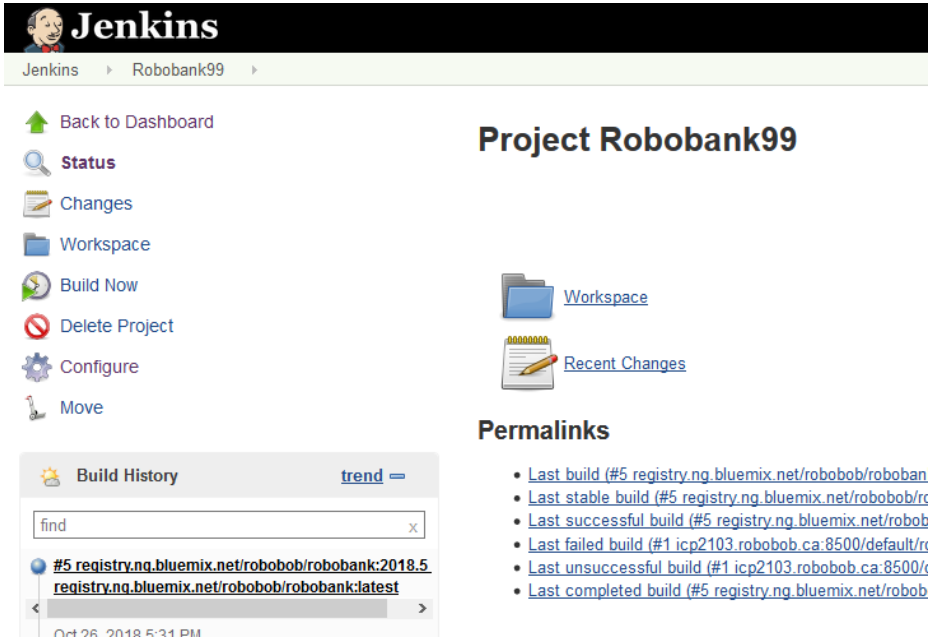
Push Assets to Source Code Control

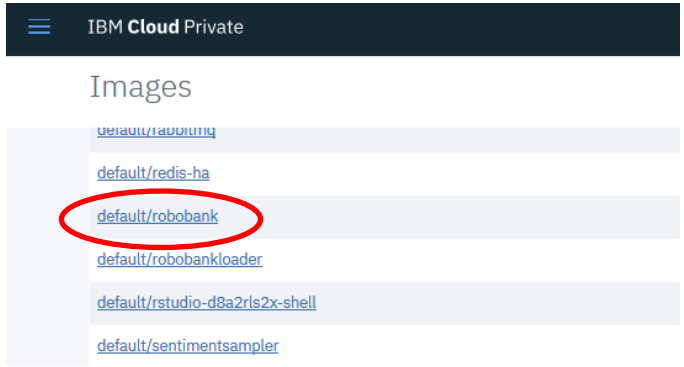
To avoid building the container manually every time a change is made, we can enable continuous integration/continuous delivery (CI/CD) pipelines by pushing all the assets into a source code control system (SCCS).

All of the projects assets, including the Dockerfile, are pushed here.



Use an Image Build Tool…





A tool like Jenkins can have a pipeline that pulls the assets from the SCCS, builds the image…

And then pushes that image into an image repository.

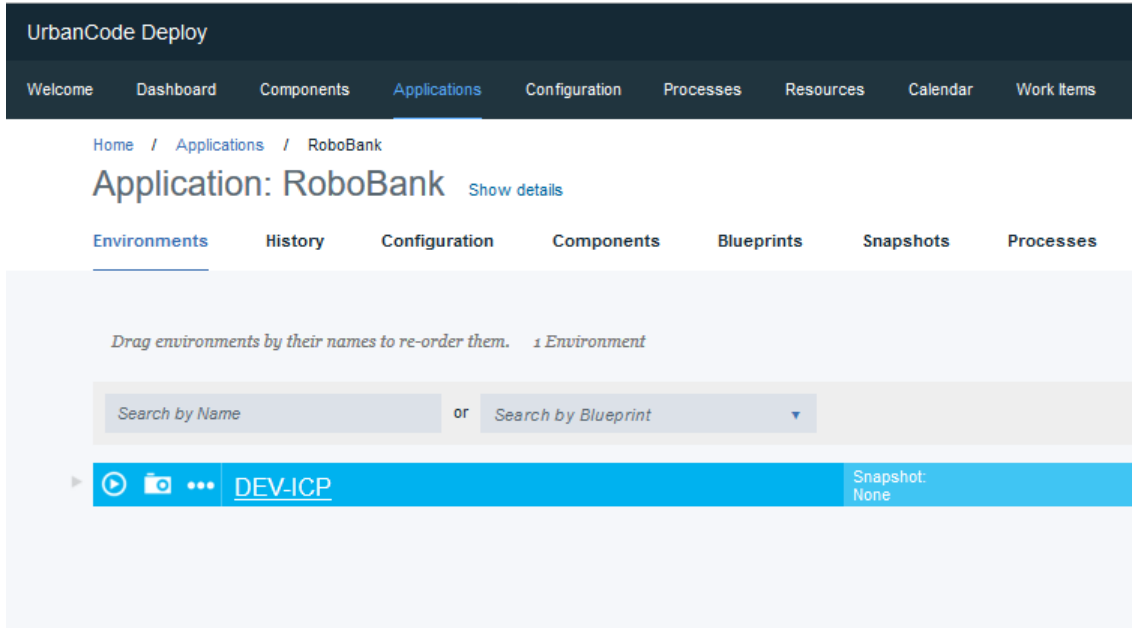
Step 5: Kuberize!

Use an Image Deployment Tool

Once built, we can use a deployment tool to pull the container image from the image repository and deploy into Kubernetes.

Jenkins can be used to deploy an container, but is limited to a single deployment target per pipeline.

A tool like UrbanCode Deploy can use a single pipeline to deploy into multiple Kubernetes targets.



Voila!

IBM **Cloud** Private

Deployments

Search items

20 items per page | 1-10 of 10 items

NAME	NAMESPACE	DESIRED	CURRENT	RE
minecraft-minecraft	default	1	1	1
robobank	default	1	1	1
sentimentsampler	default	1	1	1
robobankloader	default	4	4	4
ldapgui	default	1	1	1
jenkins-ibm-jenkins-dev	default	1	1	1
advisor-ibm-transadv-dev-couchdb	default	1	1	1

IBM Cloud Private | RoboLine Customer Account Sum

robobank.robobob.ca/RoboBank/Account

Most Visited | IBM | handy | ICP | IBM Cloud | IBM Europe

RoboBank 2000

Rob Peeren

Customer Number: **12341234**

Account #	Account Type	Balance	Limit	Balance $CDN
11111-1233	Savings		0.00	1,723.45
12311-4529	Chequing		-1,500.00	2,653.55
13334-6712	US Dollar	1,145.95	0.00	1,604.33
21982-6399	Stock Portfolio #1			62,363.00
22782-3909	RRSP			3,094.50
23922-8212	RESP			3,577.00
			Total:	75,015.83

Transfer $ _____ from 11111-1233 to 11111-1233 TRANSFER

Scroll down to see the details for Savings account 11111-1233

Action	Previous Balance	Amount	Current Balance
Deposit	1,923.45	500.00	2,423.45
Withdrawl	2,423.45	700.00	1,723.45

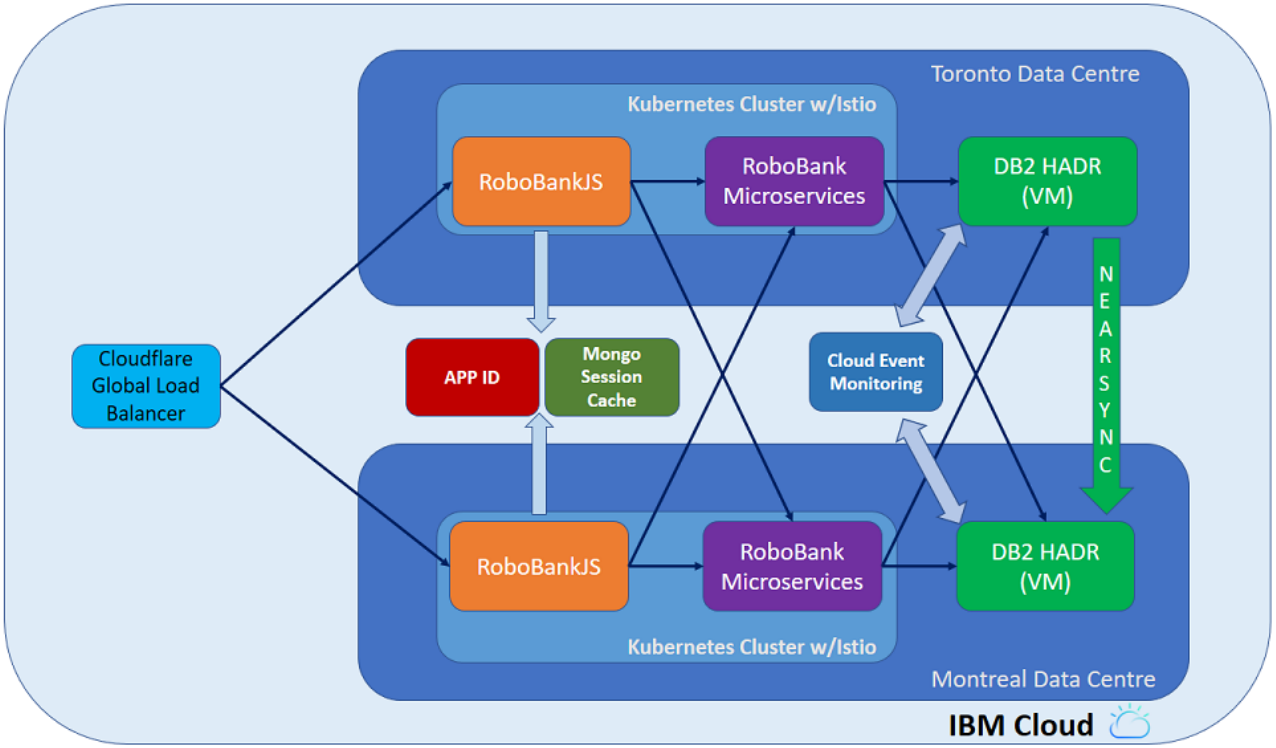
Once deployed, the application is available in the Kubernetes environment, in this case IBM Cloud Private.

Part 2: Transformation

Transforming Applications to Leverage Cloud

In many cases, not only is there a desire to make an application Container-capable, but also to transform that application to make use of modern programming techniques.

Refactoring applications to use microservices and Istio, introducing new programming languages such as NodeJS, and leveraging cloud services to manage complex tasks are all important drivers in cloud transformation.



Step 1: Leverage Cloud Tooling and Templates

Use a Cloud Template

A good place to start transforming your applications is to use code templates based on programming best practices.

IBM **Cloud** Catalog Docs Cookie Prefer

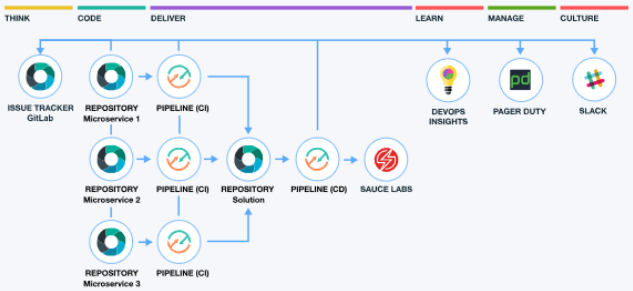
Develop and test microservices with Kubernetes and Helm

With this cloud-native toolchain, you use a sample to create an online store that consists of three microservices: a Catalog API, an Orders API, and a UI that calls both of the APIs. The toolchain is preconfigured for continuous delivery, source control, functional testing, issue tracking, online editing, and alert notification.

The Catalog and Orders API are backed by a Cloudant store. As part of deploying this application, a no-cost Cloudant service instance is created. For more information and terms, see the IBM Cloud catalog.

This toolchain uses tools that are part of the Continuous Delivery service. If an instance of that service isn't already in the selected organization, when you click **Create**, it is automatically added with the free Lite plan selected.

To get started, click **Create**.



Modify Template Code with your Business Logic

IBM Cloud DevOps templates generate all the stub code necessary to build out a particular style of application.

Developers populate the stub with their own business logic.

Additionally all the scripts needed to build and deploy the application are also generated, greatly reducing the time it takes for developers to transform their code and see it running.

EXPLORER

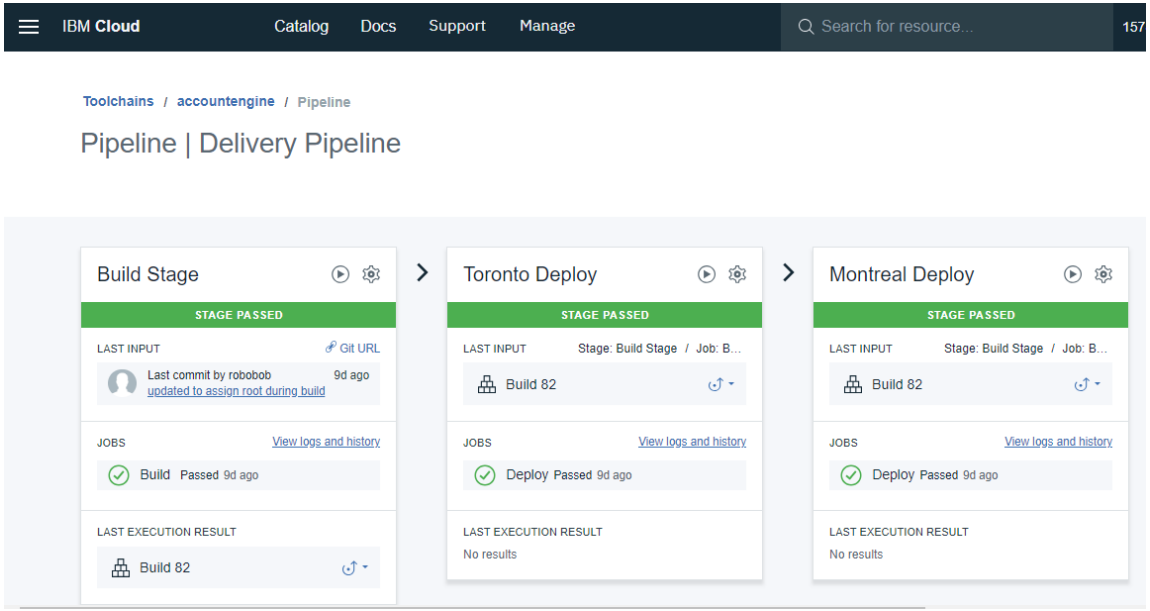
- OPEN EDITORS
- ROBOBANK2020 (WORKSPACE)
  - accountengine
    - .bluemix
    - .settings
    - chart
    - lib
    - manifests
    - src
      - main
        - java
          - application
            - rest
              - v1
                - AccountEngineMicroService.ja...
                - Example.java
              - JaxrsApplication.java
              - RootEndpoint.java
            - HealthEndpoint.java
            - robobank
              - AccountEngine.java
              - AuthEngine.java
              - CustomerBean.java
              - JDBCHelper.java
              - MutualFundBean.java
              - MutualFundEngine.java

AccountEngineMicroService.java

```
    @Produces(MediaType.APPLICATION_JSON)
    public HashMap<String,Float> rates()
    {
        return AccountEngine.getInstance().getExchangeRates();
    }

    @GET
    @Path("provision/{customerNumber}")
    @Produces(MediaType.APPLICATION_JSON)
    public HashMap<String,ArrayList> provision(@PathParam(value = "custom
    {
        CustomerBean customerBean = new CustomerBean();
        customerBean.setCustomerNumber(customerNumber);

        initialize();

        AccountEngine accountEngine = AccountEngine.getInstance();
        accountEngine.createNewCustomer(customerBean);

        return accountinfo(customerNumber);
    }

    @GET
    @Path("info/{customerNumber}")
    @Produces(MediaType.APPLICATION_JSON)
    public HashMap<String,ArrayList> accountinfo(@PathParam(value = "cust
    {
        System.out.println("***************  ROBO  ******************");
        System.out.println("Customer Number: "+customerNumber);
        System.out.println("***************  ROBO  ******************");
```

Optionally use Cloud-Native Toolchains

Existing DevOps toolchains can still be used, but cloud native toolchains can also be leveraged for users who don't want to manage and administer these tools themselves.



Deploy to a Canadian-based Kubernetes Service

IBM Cloud | Catalog | Docs | Support | Manage

Dashboard

RESOURCE GROUP: All Resources | CLOUD FOUNDRY ORG: All Organizations | CLOUD FOUNDRY SPACE: All Spaces | LOCATION: Al | CATEGORY

Clusters

Name
Montreal
Toronto

kubernetes | Search

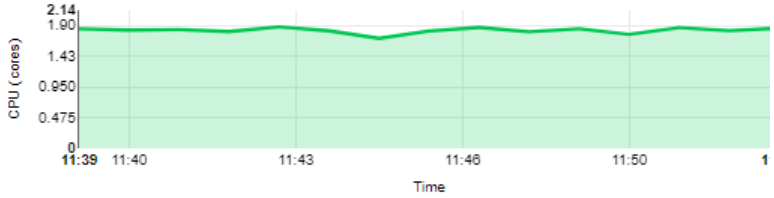
Cluster

- Cluster
- Namespaces
- Nodes
- Persistent Volumes
- Roles
- Storage Classes

Namespace: default

- Overview
- Workloads
- Cron Jobs
- Daemon Sets

CPU usage



Namespaces

Name	Labels
robobank	istio-injection: enabled

IBM Canada has cloud data centres in Toronto and Montreal to provide regional, in-country, HA workloads.

Step 2: Leverage Cloud Services

Global Load Balancing

Through IBM's CloudFlare partnership, IBM offers a global load balancing service providing up to five-9's availability to cloud-based workloads.

IBM Cloud | Catalog | Docs | Support | Manage

- Getting started
- Overview
- Metrics
- Security
- Reliability
  - Global Load Balancers
  - DNS
- Performance
- Plan

Dashboard /

RoboInternetServices

Resource Group: default Location: Global

Global Load Balancers

Protect your service from disruptions and outages with the Global Load Balancing service. You can distribute your traffic across multiple servers with a combination of origin pools, health checks, and a load balancer. Automatically respond to origin failures with active failover.

Manage | Health Check Events

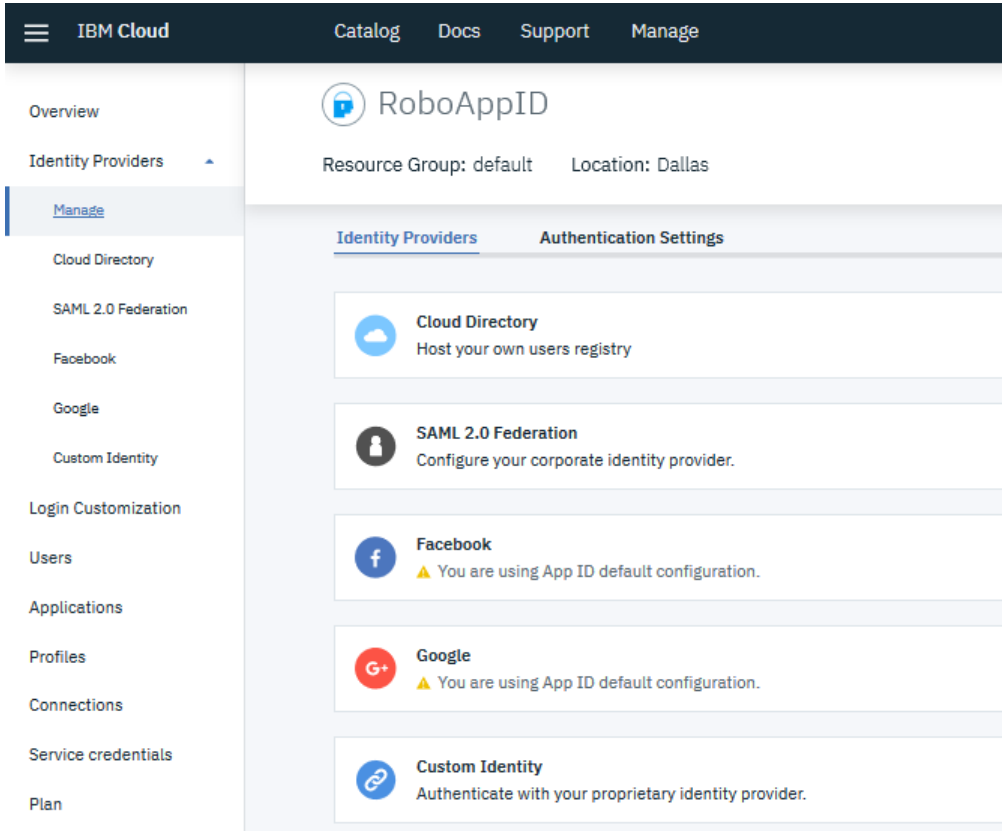
Load Balancers

Global load balancers help distribute your proxied traffic across servers within an origin pool, and based on the pool's health, fail over to a different pool if needed.

	HEALTH	HOSTNAME	AVAILABLE POOLS
>	Healthy	cashback-dc	1 of 1
>	Healthy	cashback	1 of 1
>	Healthy	cpuloader	1 of 1

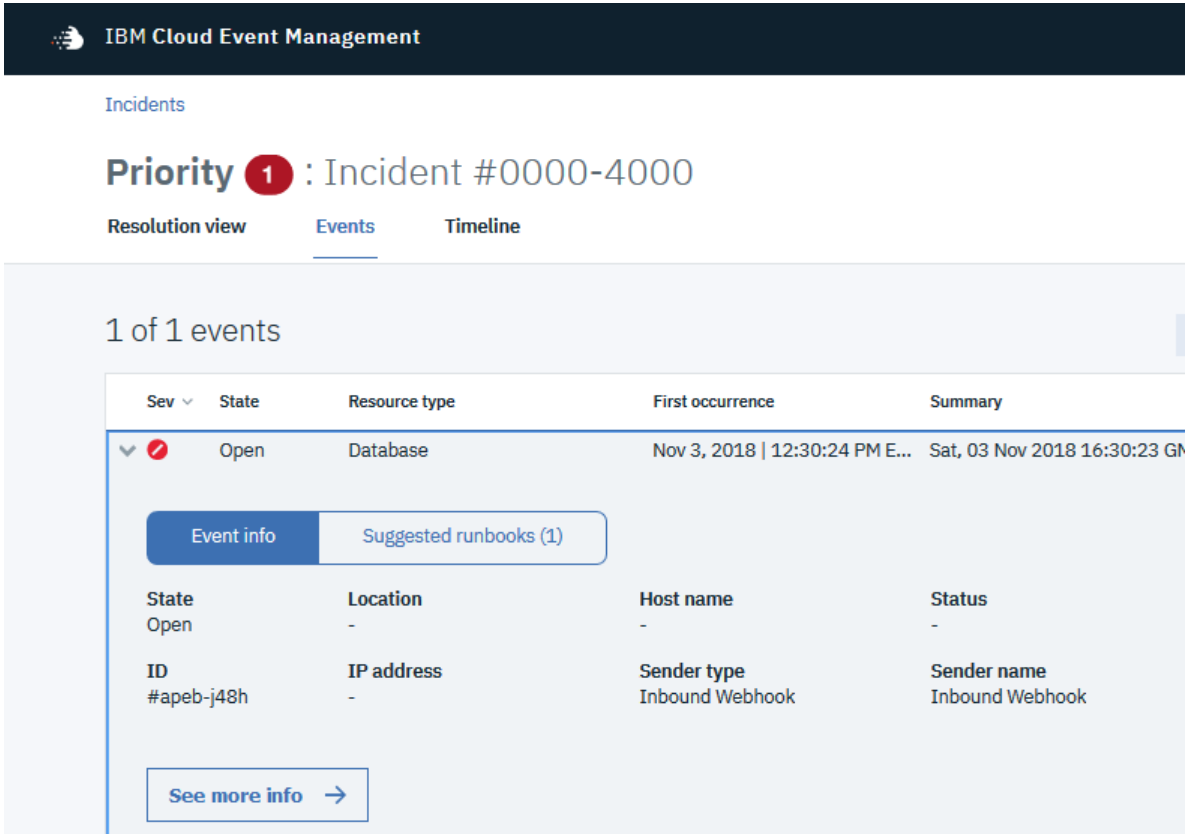
Cloud-based Authentication Service

IBM's AppID service enables the ability to farm out authentication to an external service.



Cloud-based Event Management

IBM's Cloud Event Management service enables the ability to monitor cloud applications and respond to situations either automatically or through human-oriented run books.



Canadian-based Infrastructure

Data centres in Toronto and Montreal ensure cloud based applications and associated data can reside in Canada and be highly available.

IBM **Cloud** Catalog Support Account Help

Devices
Storage
Network
Security
Services

icp31-1.robobob.ca	Virtual Server	Toronto 1
icp31-w1.robobob.ca	Virtual Server	Toronto 1
icp31-w2.robobob.ca	Virtual Server	Toronto 1
icp31-w3.robobob.ca	Virtual Server	Toronto 1
icp31-w4.robobob.ca	Virtual Server	Toronto 1
icp4.robobob.ca	Virtual Server	Toronto 1
icp5.robobob.ca	Virtual Server	Toronto 1
kube-mon01-cr751ef7ab97c74631944355e6(	Virtual Server	Montreal 1
kube-mon01-cr751ef7ab97c74631944355e6(	Virtual Server	Montreal 1
kube-mon01-cr751ef7ab97c74631944355e6(	Virtual Server	Montreal 1
kube-tor01-cr1e8c530c24e745ef91ca564518	Virtual Server	Toronto 1
kube-tor01-cr1e8c530c24e745ef91ca564518	Virtual Server	Toronto 1
kube-tor01-cr1e8c530c24e745ef91ca564518	Virtual Server	Toronto 1
montreal.db2.ibmcloud	Virtual Server	Montreal 1
toronto.db2.ibmcloud	Virtual Server	Toronto 1

Shameless Self-Promotion

Watch my Videos!

Application Modernization Part 1: Moving Old J2EE Apps into Docker and IBM Cloud Private
https://www.youtube.com/watch?v=XJ014-YowV8

Application Modernization Part 2: Java Workloads using IBM Middleware like MQ and IIB in IBM Cloud Private
https://www.youtube.com/watch?v=yn-j6-7KydA

Continuing the Modernization Journey: Part 1, Dev Practices in a Cloudified World
https://www.youtube.com/watch?v=BJ_rYuroQgU

Continuing the Modernization Journey: Part 2, Ops Practices in a Cloudified World
https://www.youtube.com/watch?v=dpo0RDJ2wqA

Application Modernization and Transformation

A Case-Study

Rob Peeren, B.Sc.

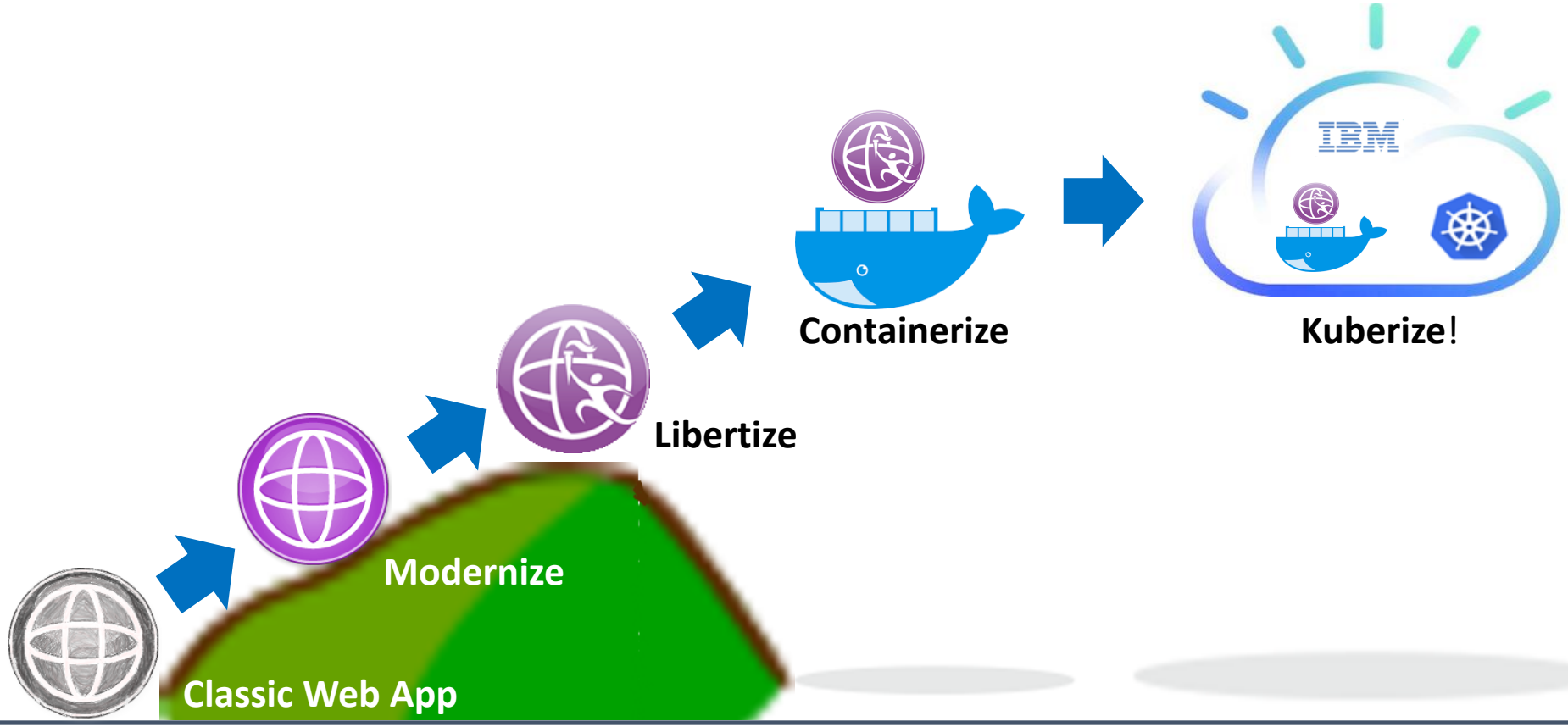
Executive IT Specialist

IBM Canada Ltd.

June 2019

Part 1: Modernization

The Application Modernization Blueprint



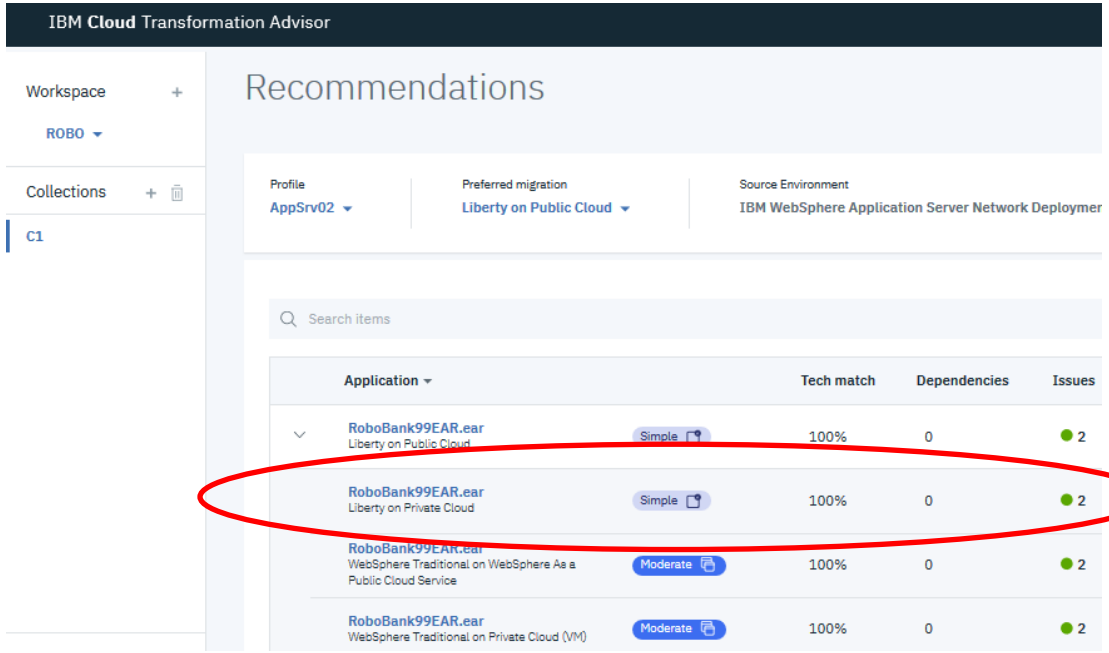
Step 1: Identify the Classic Application to Modernize

Transformation Advisor Identifies Candidates…

The IBM Transformation Advisor crawls through your inventory of JEE applications and generates a detailed report about the effort involved to migrate your applications into containers.

It provides line-by-line recommendations on code that is obsolete or deprecated.

Built by migration specialists over a 10 year span, it helps identify which applications can be easily migrated, which ones need some effort, and which ones would need refactoring.



The 20-Year Old Web Application…

This is an example of a web application built in the late 1990's.

It is a fairly simple JSP/Servlet/JDBC application that is also typical of the types of applications being written back then.

It originally ran on JDK 1.1.6 and IBM WebSphere Application Server 2.0!

TransferBean.java	7/22/1999 2:47 PM
StockTransactionServlet.java	8/19/1999 2:41 PM
StockTransactionBean.java	7/22/1999 2:48 PM
StockHistoryServlet.java	7/22/1999 3:00 PM
StockEngine.java	8/19/1999 2:49 PM
StockAnalysisServlet.java	7/22/1999 2:59 PM
QuoteServlet.java	8/19/1999 2:45 PM
QuoteEngine.java	8/19/1999 11:39 AM
QuoteBean.java	8/19/1999 12:44 PM
MutualFundEngine.java	8/19/1999 11:41 AM
MutualFundBean.java	7/22/1999 2:46 PM
InvestmentSummaryServlet.java	7/22/1999 2:59 PM
InitializeServlet.java	7/22/1999 2:45 PM
FundsTransferServlet.java	7/22/1999 2:59 PM
CustomerBean.java	8/25/1999 4:52 PM
AuthEngine.java	8/19/1999 11:40 AM
AccountSummaryServlet.java	7/22/1999 2:58 PM

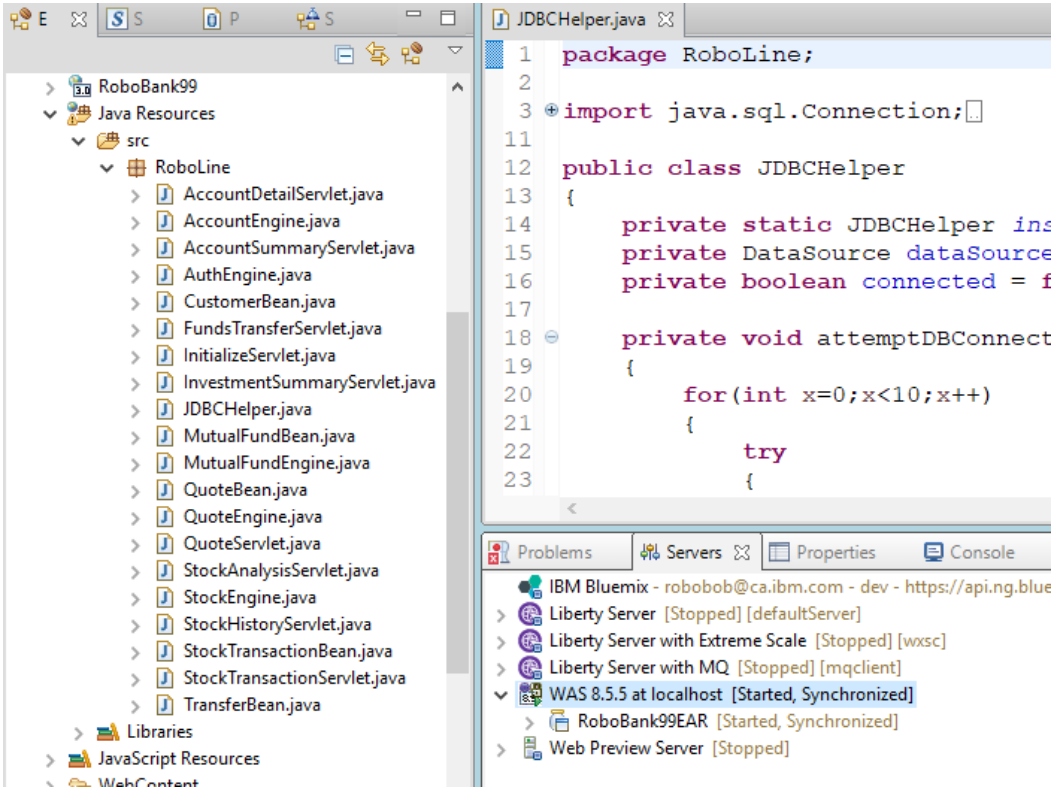
Step 2: Modernize

Import the Project into Modern Tooling…

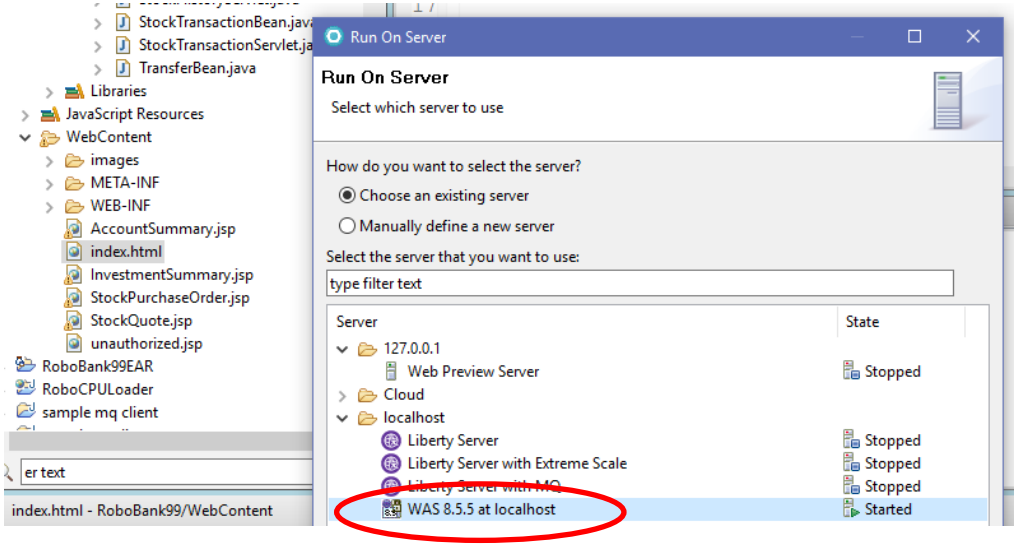
In this example, the original code is imported into Rational Application Developer (RAD) V8.5

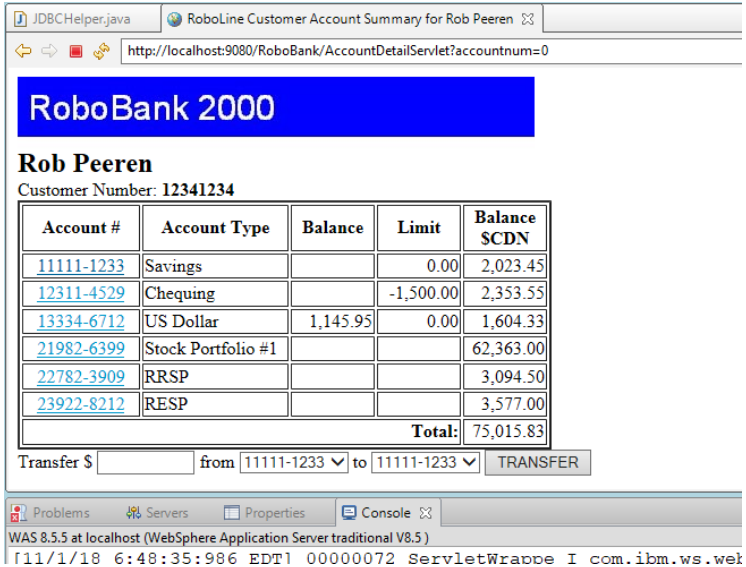
Based on recommendations from the Transformation Advisor, the code is quickly cleaned up by the developer (~1.5 days of effort in this case)

RAD includes a local copy of WebSphere Application Server (WAS) 8.5.5 that the application can be deployed to.



Deploy Updated Project to Modern App Server



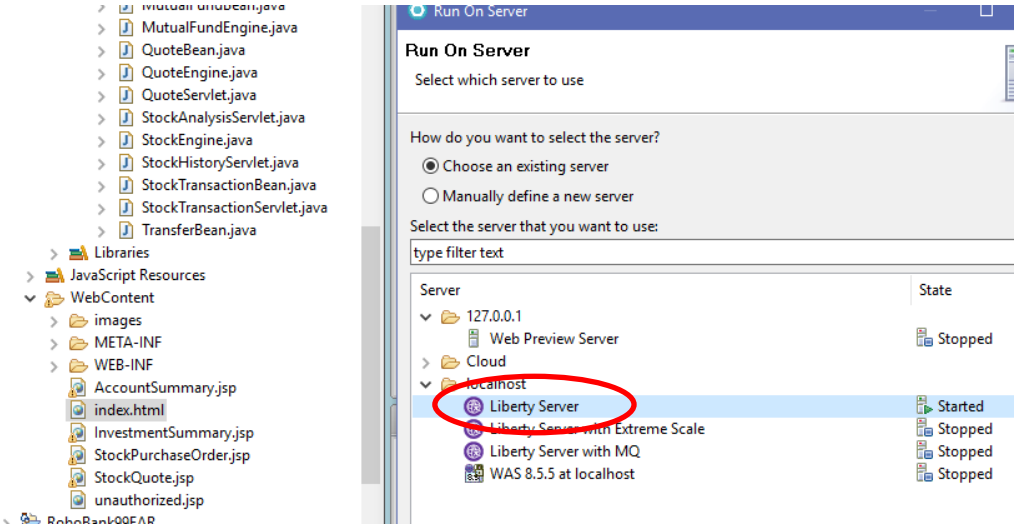


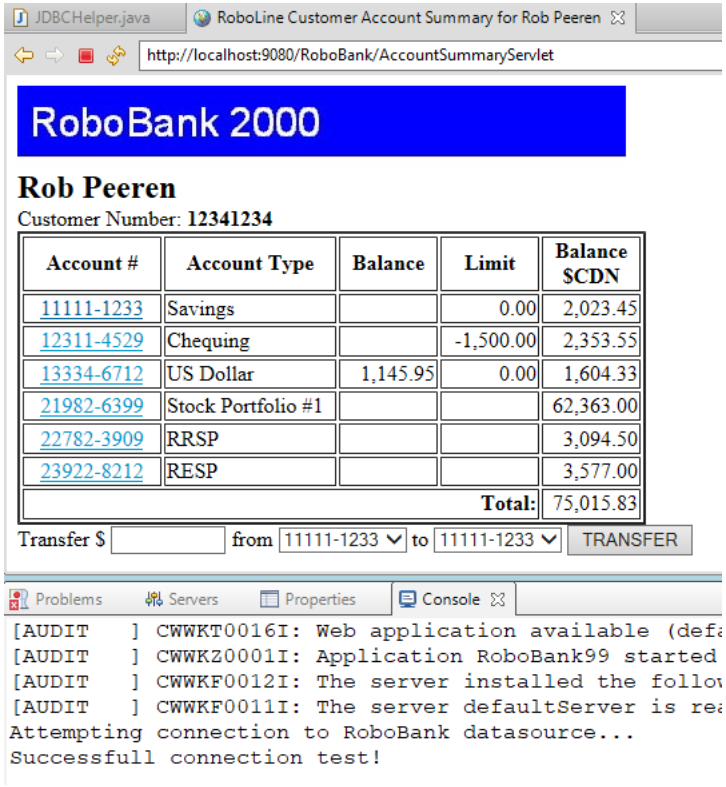
WAS 8.5.5 is made the run target of the application…

And it runs!

Step 3: Libertize

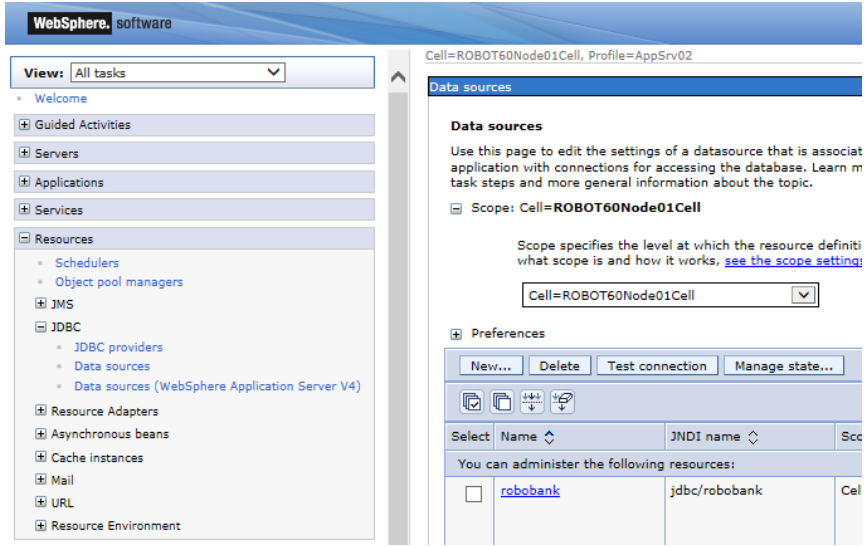
Repeat the Deploy to Liberty Application Server…





Liberty is now made the run target of the application, and it still works! No extra refactoring required.

What’s the Difference?




```xml
<server description="new server">
      <featureManager>
      <feature>jdbc-4.0</feature>
      <feature>jndi-1.0</feature>
      <feature>jsp-2.2</feature>
      <feature>localConnector-1.0</feature>
      <feature>servlet-3.1</feature>
      <feature>json-1.0</feature>
   </featureManager>

   <library id="DB2Lib">
     <fileset dir="${shared.resource.dir}/db2" includes="*.jar"/>
   </library>

   <dataSource jndiName="jdbc/robobank" transactional="true">
      <jdbcDriver libraryRef="DB2Lib"/>
      <properties.db2.jcc currentSchema="DB2" databaseName="robobank"
      <!-- <properties.db2.jcc  enableClientAffinitiesList="1" client
   </dataSource>

   <httpEndpoint httpPort="9080" httpsPort="9443" id="defaultHttpEndp
   <webApplication contextRoot="RoboBank" id="RoboBank99" location="R

   <applicationMonitor updateTrigger="mbean"/>
</server>
```

WebSphere Admin Console

WebSphere Liberty server.xml

Step 4: Containerize

Create Dockerfile

Containerizing an application requires the building of a Dockerfile, which essentially is a manifest of all the assets that are required to run your application within a container.

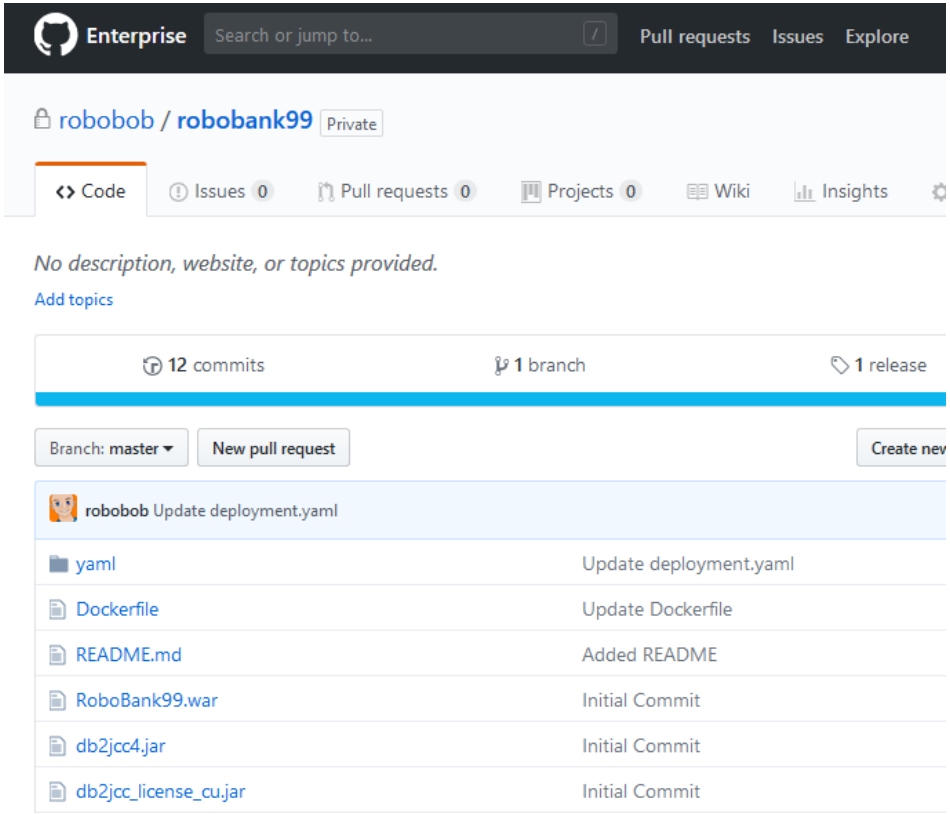
In this case, the web application WAR file, jdbc drivers, and other assets are copied into the container.

```
FROM websphere-liberty:webProfile7

USER root
RUN mkdir /opt/ibm/wlp/usr/shared/resources/db2
COPY runBank.sh /usr/local/bin
RUN chmod +x /usr/local/bin/runBank.sh
COPY db2jcc4.jar /opt/ibm/wlp/usr/shared/resources/db2
COPY db2jcc_license_cu.jar /opt/ibm/wlp/usr/shared/resources/db2
COPY server.xml /opt/ibm/wlp/usr/servers/defaultServer
COPY RoboBank99.war /opt/ibm/wlp/usr/servers/defaultServer/apps
RUN /opt/ibm/wlp/bin/installUtility install defaultServer
USER 1001


EXPOSE 9080 9443

CMD ["/usr/local/bin/runBank.sh"]
```

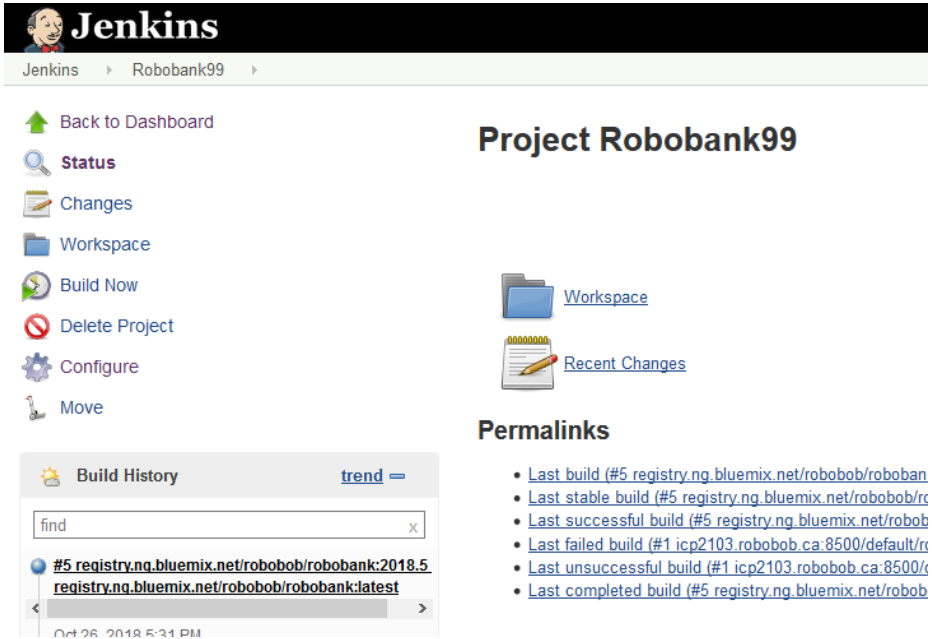
Push Assets to Source Code Control

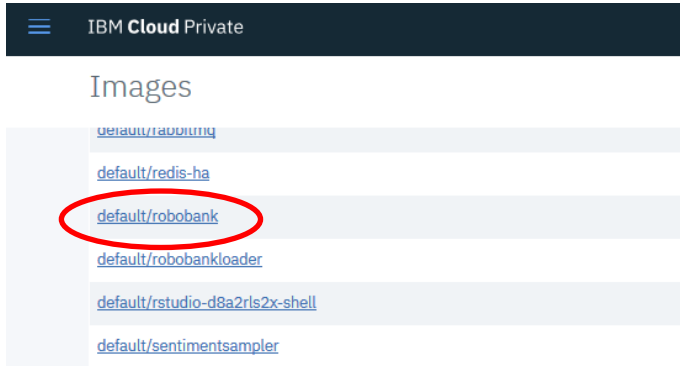
To avoid building the container manually every time a change is made, we can enable continuous integration/continuous delivery (CI/CD) pipelines by pushing all the assets into a source code control system (SCCS).

All of the projects assets, including the Dockerfile, are pushed here.



Use an Image Build Tool…





A tool like Jenkins can have a pipeline that pulls the assets from the SCCS, builds the image…

And then pushes that image into an image repository.

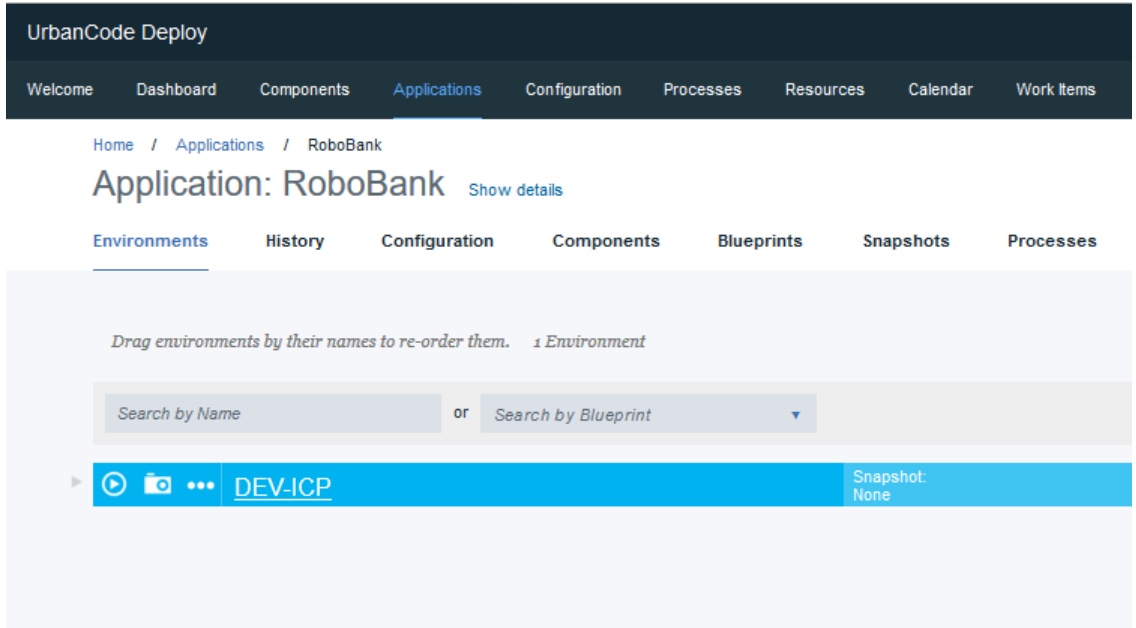
Step 5: Kuberize!

Use an Image Deployment Tool

Once built, we can use a deployment tool to pull the container image from the image repository and deploy into Kubernetes.

Jenkins can be used to deploy an container, but is limited to a single deployment target per pipeline.

A tool like UrbanCode Deploy can use a single pipeline to deploy into multiple Kubernetes targets.



Voila!

IBM **Cloud** Private

Deployments

Search items

20 items per page | 1-10 of 10 items

NAME	NAMESPACE	DESIRED	CURRENT	RE
minecraft-minecraft	default	1	1	1
robobank	default	1	1	1
sentimentsampler	default	1	1	1
robobankloader	default	4	4	4
ldapgui	default	1	1	1
jenkins-ibm-jenkins-dev	default	1	1	1
advisor-ibm-transadv-dev-couchdb	default	1	1	1

IBM Cloud Private | RoboLine Customer Account Sum

robobank.robobob.ca/RoboBank/Account

Most Visited | IBM | handy | ICP | IBM Cloud | IBM Europe

RoboBank 2000

Rob Peeren

Customer Number: **12341234**

Account #	Account Type	Balance	Limit	Balance $CDN
11111-1233	Savings		0.00	1,723.45
12311-4529	Chequing		-1,500.00	2,653.55
13334-6712	US Dollar	1,145.95	0.00	1,604.33
21982-6399	Stock Portfolio #1			62,363.00
22782-3909	RRSP			3,094.50
23922-8212	RESP			3,577.00
			Total:	75,015.83

Transfer $ _____ from 11111-1233 to 11111-1233 TRANSFER

Scroll down to see the details for Savings account 11111-1233

Action	Previous Balance	Amount	Current Balance
Deposit	1,923.45	500.00	2,423.45
Withdrawl	2,423.45	700.00	1,723.45

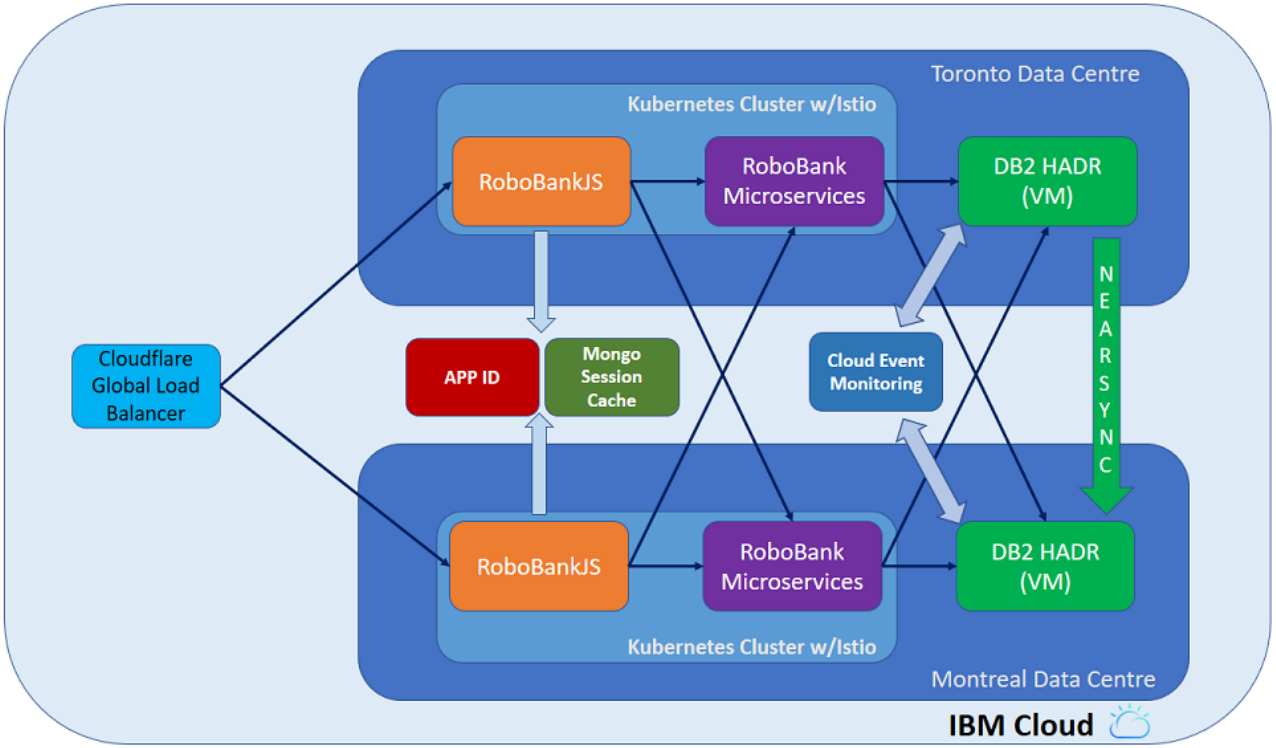
Once deployed, the application is available in the Kubernetes environment, in this case IBM Cloud Private.

Part 2: Transformation

Transforming Applications to Leverage Cloud

In many cases, not only is there a desire to make an application Container-capable, but also to transform that application to make use of modern programming techniques.

Refactoring applications to use microservices and Istio, introducing new programming languages such as NodeJS, and leveraging cloud services to manage complex tasks are all important drivers in cloud transformation.



Step 1: Leverage Cloud Tooling and Templates

Use a Cloud Template

A good place to start transforming your applications is to use code templates based on programming best practices.

IBM **Cloud** Catalog Docs Cookie Prefer

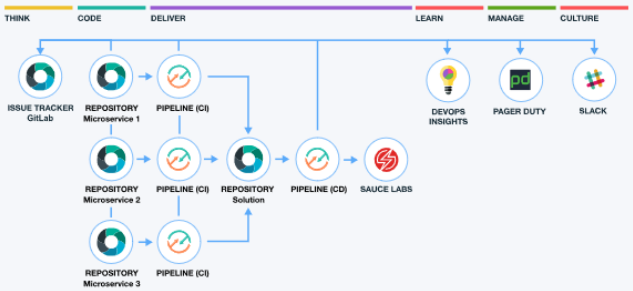
Develop and test microservices with Kubernetes and Helm

With this cloud-native toolchain, you use a sample to create an online store that consists of three microservices: a Catalog API, an Orders API, and a UI that calls both of the APIs. The toolchain is preconfigured for continuous delivery, source control, functional testing, issue tracking, online editing, and alert notification.

The Catalog and Orders API are backed by a Cloudant store. As part of deploying this application, a no-cost Cloudant service instance is created. For more information and terms, see the IBM Cloud catalog.

This toolchain uses tools that are part of the Continuous Delivery service. If an instance of that service isn't already in the selected organization, when you click **Create**, it is automatically added with the free Lite plan selected.

To get started, click **Create**.



Modify Template Code with your Business Logic

IBM Cloud DevOps templates generate all the stub code necessary to build out a particular style of application.

Developers populate the stub with their own business logic.

Additionally all the scripts needed to build and deploy the application are also generated, greatly reducing the time it takes for developers to transform their code and see it running.

EXPLORER

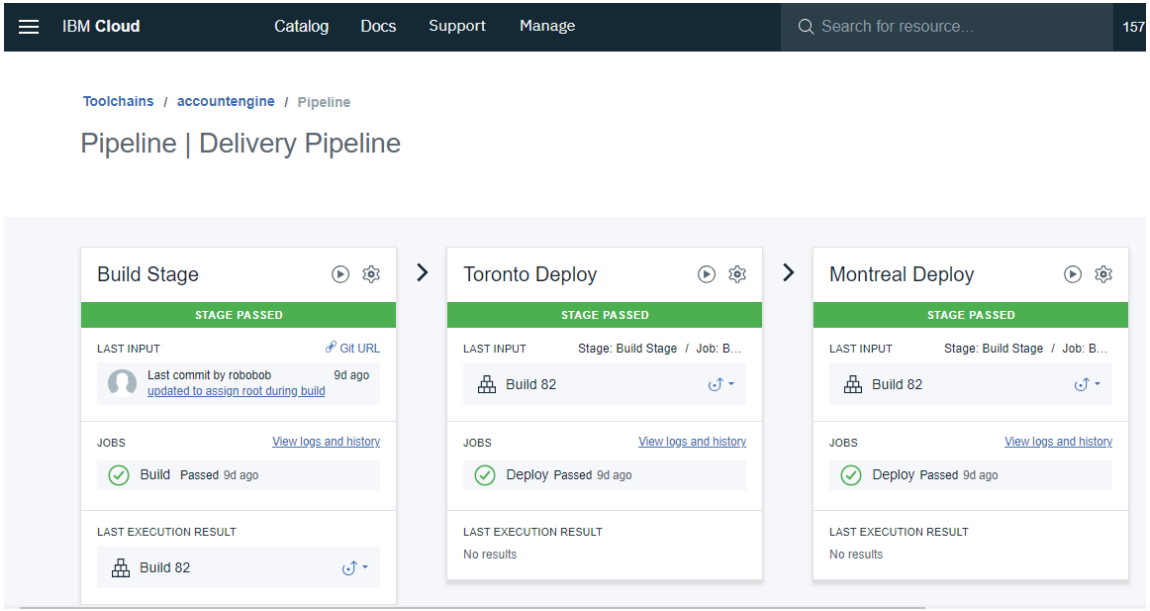
- OPEN EDITORS
- ROBOBANK2020 (WORKSPACE)
  - accountengine
    - .bluemix
    - .settings
    - chart
    - lib
    - manifests
    - src
      - main
        - java
          - application
            - rest
              - v1
                - AccountEngineMicroService.ja...
                - Example.java
              - JaxrsApplication.java
              - RootEndpoint.java
            - HealthEndpoint.java
            - robobank
              - AccountEngine.java
              - AuthEngine.java
              - CustomerBean.java
              - JDBCHelper.java
              - MutualFundBean.java
              - MutualFundEngine.java

AccountEngineMicroService.java

```
    @Produces(MediaType.APPLICATION_JSON)
    public HashMap<String,Float> rates()
    {
        return AccountEngine.getInstance().getExchangeRates();
    }

    @GET
    @Path("provision/{customerNumber}")
    @Produces(MediaType.APPLICATION_JSON)
    public HashMap<String,ArrayList> provision(@PathParam(value = "custom
    {
        CustomerBean customerBean = new CustomerBean();
        customerBean.setCustomerNumber(customerNumber);

        initialize();

        AccountEngine accountEngine = AccountEngine.getInstance();
        accountEngine.createNewCustomer(customerBean);

        return accountinfo(customerNumber);
    }

    @GET
    @Path("info/{customerNumber}")
    @Produces(MediaType.APPLICATION_JSON)
    public HashMap<String,ArrayList> accountinfo(@PathParam(value = "cust
    {
        System.out.println("***************  ROBO  ******************");
        System.out.println("Customer Number: "+customerNumber);
        System.out.println("***************  ROBO  ******************");
```

Optionally use Cloud-Native Toolchains

Existing DevOps toolchains can still be used, but cloud native toolchains can also be leveraged for users who don't want to manage and administer these tools themselves.



Deploy to a Canadian-based Kubernetes Service

IBM Cloud | Catalog | Docs | Support | Manage

Dashboard

RESOURCE GROUP: All Resources | CLOUD FOUNDRY ORG: All Organizations | CLOUD FOUNDRY SPACE: All Spaces | LOCATION: Al | CATEGORY

Clusters

Name
Montreal
Toronto

kubernetes | Search

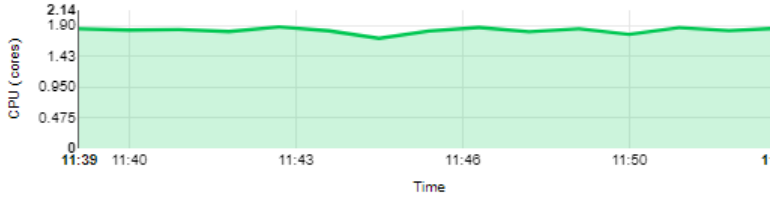
Cluster

- Cluster
- Namespaces
- Nodes
- Persistent Volumes
- Roles
- Storage Classes

Namespace: default

- Overview
- Workloads
- Cron Jobs
- Daemon Sets

CPU usage



Namespaces

Name	Labels
robobank	istio-injection: enabled

IBM Canada has cloud data centres in Toronto and Montreal to provide regional, in-country, HA workloads.

Step 2: Leverage Cloud Services

Global Load Balancing

Through IBM's CloudFlare partnership, IBM offers a global load balancing service providing up to five-9's availability to cloud-based workloads.

IBM Cloud | Catalog | Docs | Support | Manage

- Getting started
- Overview
- Metrics
- Security
- Reliability
  - Global Load Balancers
  - DNS
- Performance
- Plan

Dashboard /

RoboInternetServices

Resource Group: default Location: Global

Global Load Balancers

Protect your service from disruptions and outages with the Global Load Balancing service. You can distribute your traffic across multiple servers with a combination of origin pools, health checks, and a load balancer. Automatically respond to origin failures with active failover.

Manage | Health Check Events

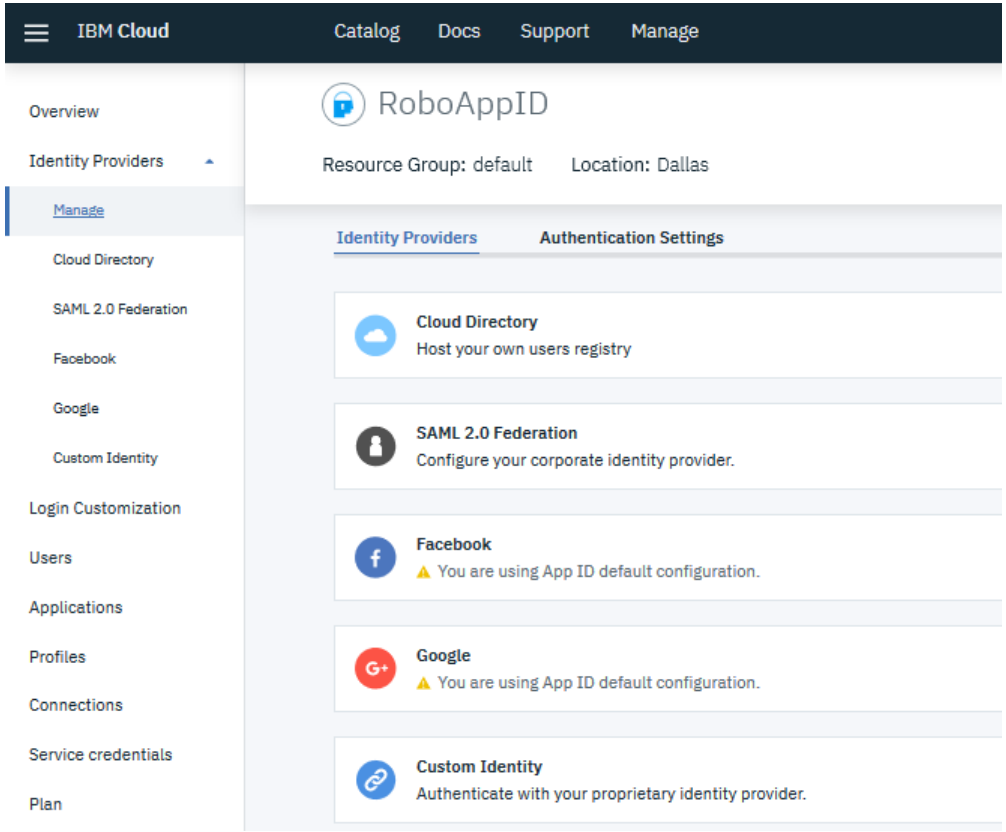
Load Balancers

Global load balancers help distribute your proxied traffic across servers within an origin pool, and based on the pool's health, fail over to a different pool if needed.

	HEALTH	HOSTNAME	AVAILABLE POOLS
>	Healthy	cashback-dc	1 of 1
>	Healthy	cashback	1 of 1
>	Healthy	cpuloader	1 of 1

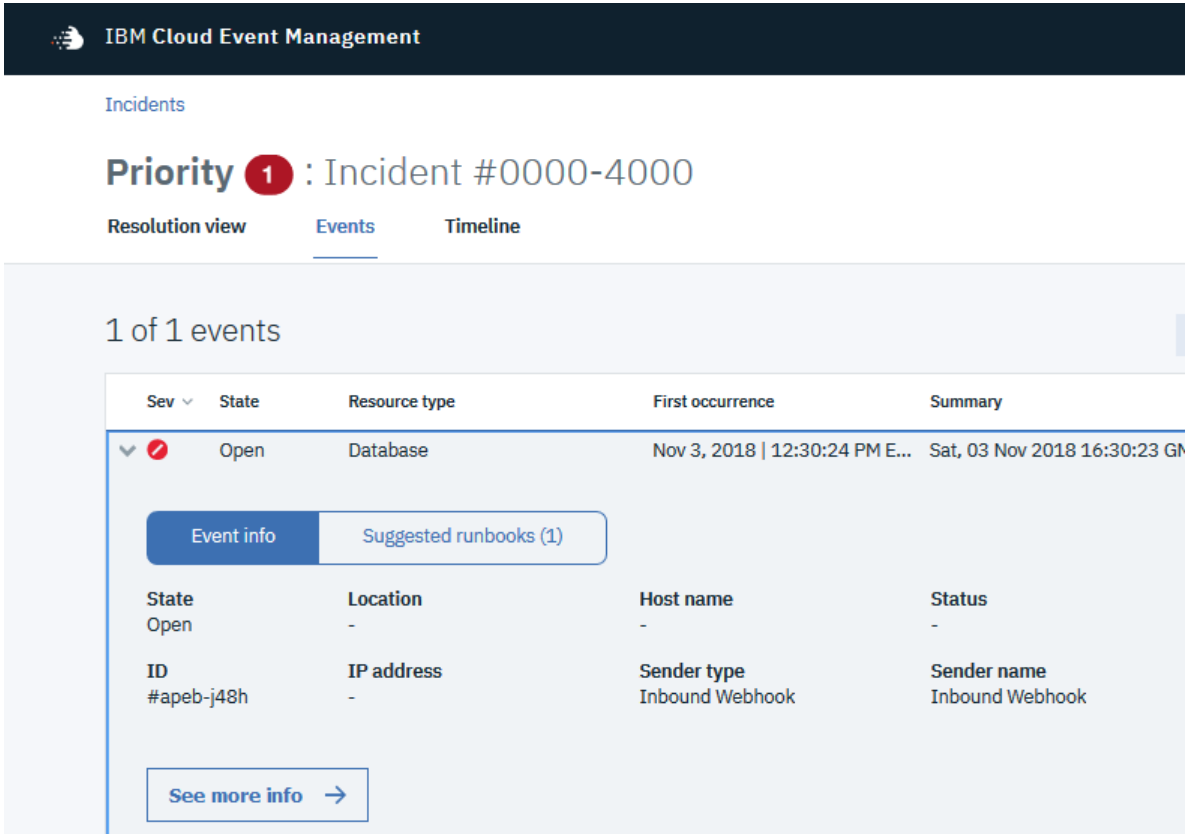
Cloud-based Authentication Service

IBM's AppID service enables the ability to farm out authentication to an external service.



Cloud-based Event Management

IBM's Cloud Event Management service enables the ability to monitor cloud applications and respond to situations either automatically or through human-oriented run books.



Canadian-based Infrastructure

Data centres in Toronto and Montreal ensure cloud based applications and associated data can reside in Canada and be highly available.

IBM **Cloud** Catalog Support Account Help

Devices
Storage
Network
Security
Services

icp31-1.robobob.ca	Virtual Server	Toronto 1
icp31-w1.robobob.ca	Virtual Server	Toronto 1
icp31-w2.robobob.ca	Virtual Server	Toronto 1
icp31-w3.robobob.ca	Virtual Server	Toronto 1
icp31-w4.robobob.ca	Virtual Server	Toronto 1
icp4.robobob.ca	Virtual Server	Toronto 1
icp5.robobob.ca	Virtual Server	Toronto 1
kube-mon01-cr751ef7ab97c74631944355e6(	Virtual Server	Montreal 1
kube-mon01-cr751ef7ab97c74631944355e6(	Virtual Server	Montreal 1
kube-mon01-cr751ef7ab97c74631944355e6(	Virtual Server	Montreal 1
kube-tor01-cr1e8c530c24e745ef91ca564518	Virtual Server	Toronto 1
kube-tor01-cr1e8c530c24e745ef91ca564518	Virtual Server	Toronto 1
kube-tor01-cr1e8c530c24e745ef91ca564518	Virtual Server	Toronto 1
montreal.db2.ibmcloud	Virtual Server	Montreal 1
toronto.db2.ibmcloud	Virtual Server	Toronto 1

Shameless Self-Promotion

Watch my Videos!

Application Modernization Part 1: Moving Old J2EE Apps into Docker and IBM Cloud Private
https://www.youtube.com/watch?v=XJ014-YowV8

Application Modernization Part 2: Java Workloads using IBM Middleware like MQ and IIB in IBM Cloud Private
https://www.youtube.com/watch?v=yn-j6-7KydA

Continuing the Modernization Journey: Part 1, Dev Practices in a Cloudified World
https://www.youtube.com/watch?v=BJ_rYuroQgU

Continuing the Modernization Journey: Part 2, Ops Practices in a Cloudified World
https://www.youtube.com/watch?v=dpo0RDJ2wqA


Спасибо
Gracias
شكرا
Obrigado
Спасибо
Dank U
Grazie
Ευχαριστώ
Danke
Ευχαριστώ
Obrigado
Merci
Thank You
Diolch
Ngiyabonga
Thank You
Ngiyabonga
Dank U
Merci
Tack
Ευχαριστώ
Dziękі
Danke
Grazie
תודה
Dank U
Diolch
Terima Kasih
Grazie
Merci
Tack
Tack
Dzięki
Diolch
Gracias
תודה