

Gladwin Rao
Practice Leader
Multi and Hybrid Cloud Architecture

Rob Peeren
Executive IT Specialist
Hybrid Cloud

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?

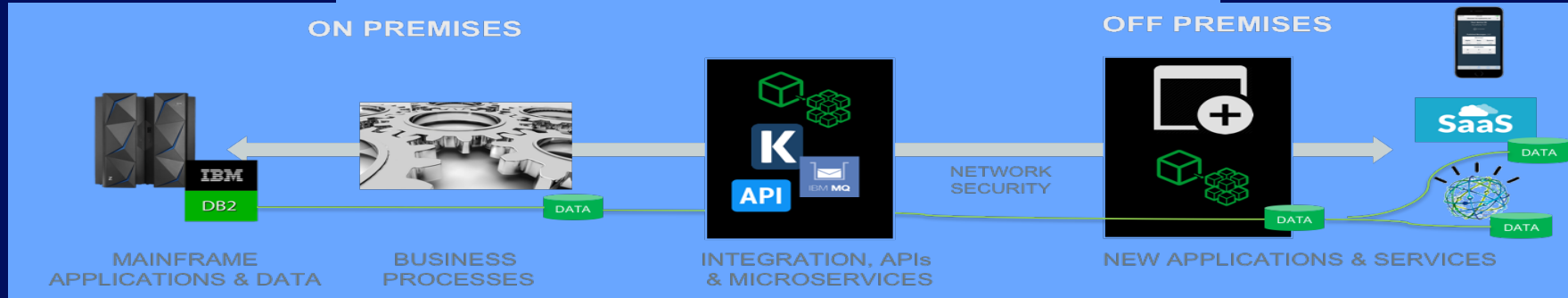
Enabler for Innovation and Agility

Application Modernization

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

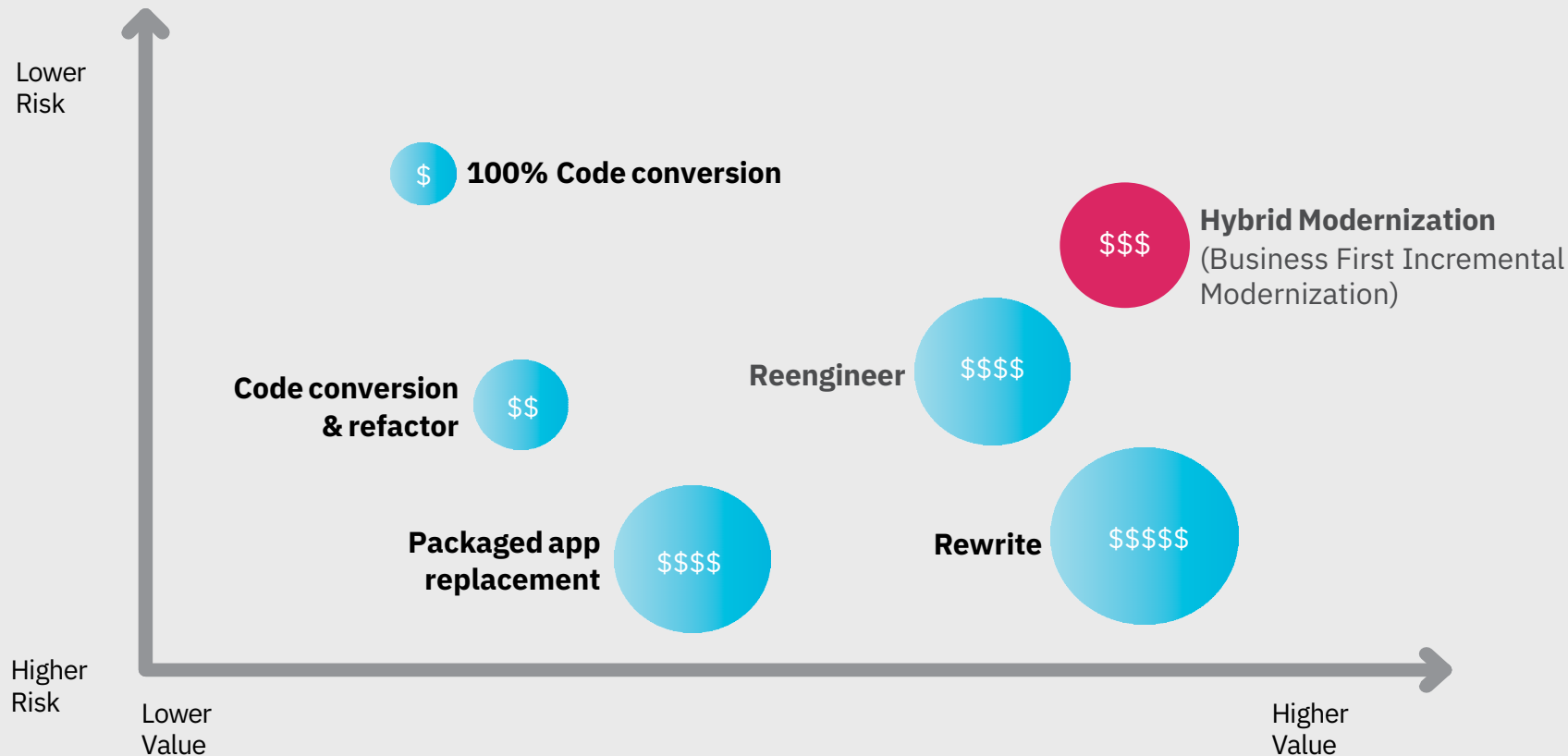
Multi-cloud

Open

Secure

Management

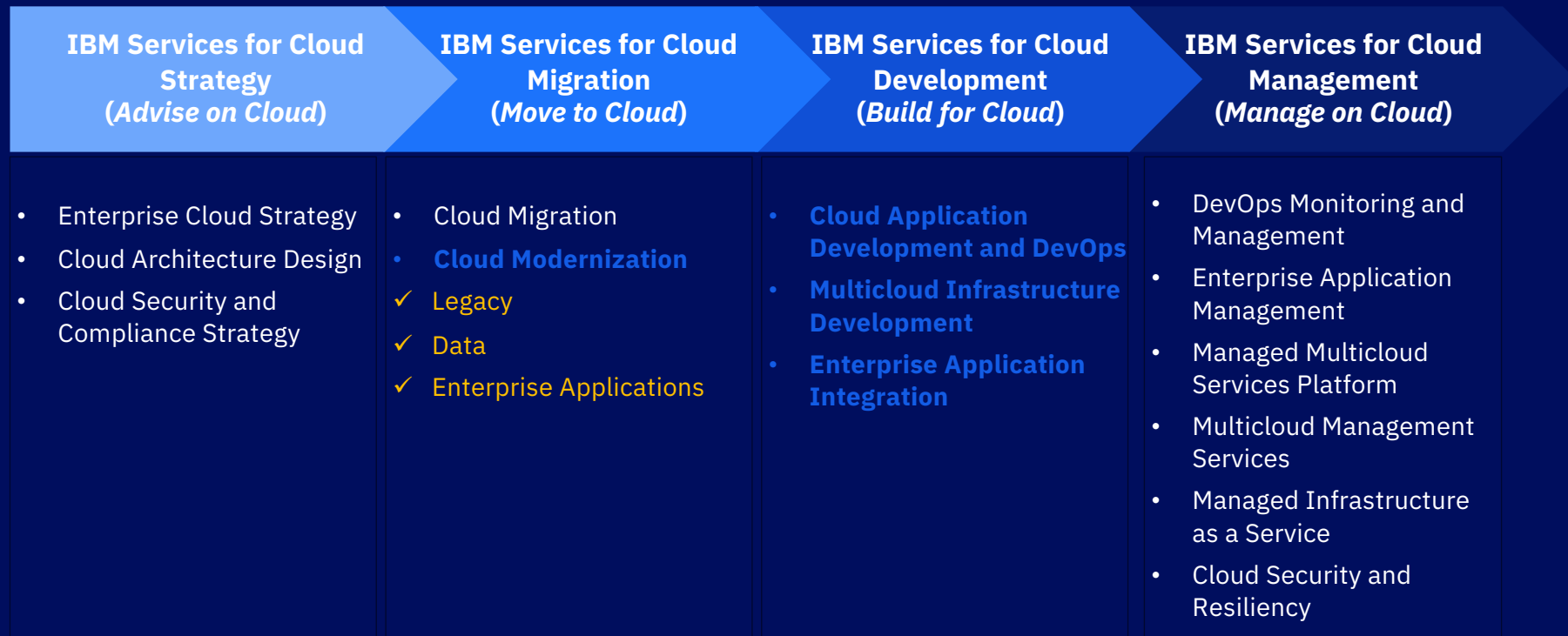
Modernization can take many forms



**Application
Modernization**

**How do we approach
Application Modernization?**

Journey to Cloud



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

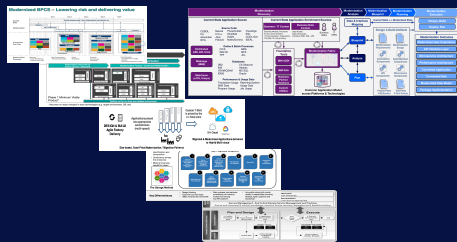
Re-architect

- 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

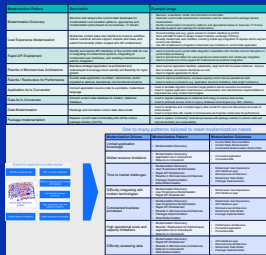
Re-Imagine

- 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

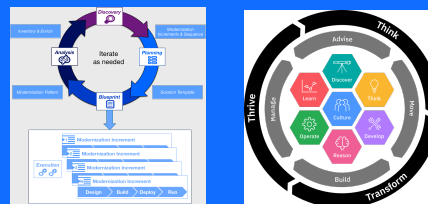
Modernization Toolkit



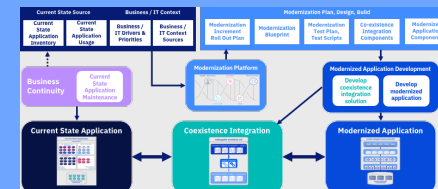
Modernization Patterns



Modernization Method



Coexistence Architecture



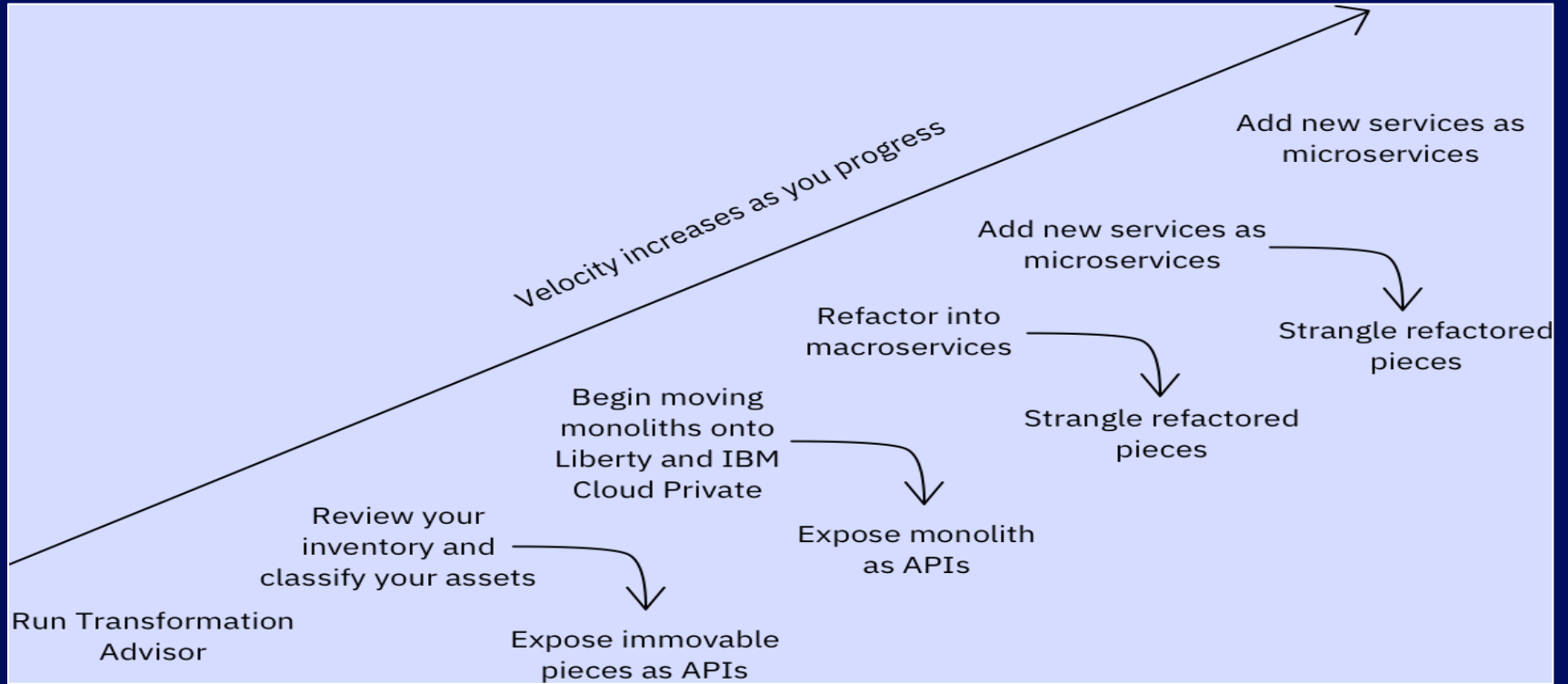
- Tailored integration of discovery and analysis tools for unique combination of current state technologies
- Centralized platform and technology agnostic application model for integrated discovery
- Customizable for addressing unique patterns and technologies
- Data science and analytics based providing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

- 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

- 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 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



World's largest airline: Bottom-up business transformation

Client context

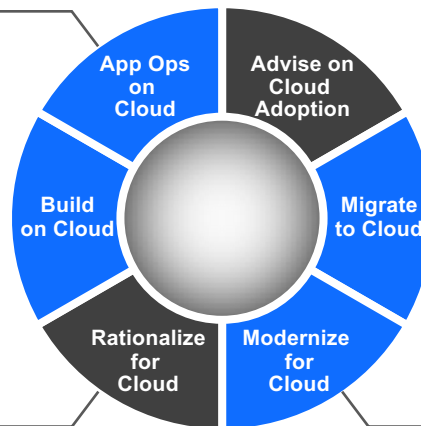
Client had three main asks:

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

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.

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

**Application
Modernization
and Journey to
Cloud**

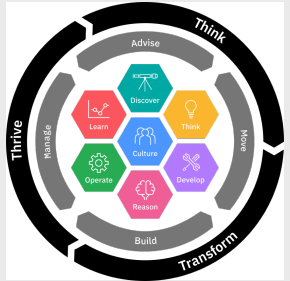
**What are our credentials in
this space and why IBM?**



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



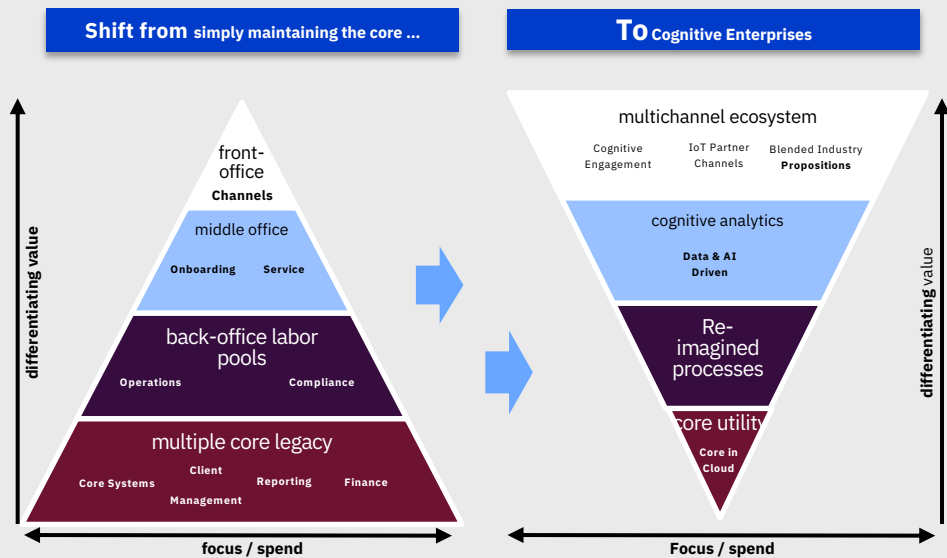
GARAGES DELIVER INNOVATIVE, RAPID TRANSFORMATION AND LASTING CULTURE CHANGE

- RED HAT OPENSIFT
- AZURE
- AMAZON WEB SERVICES
- GOOGLE CLOUD PLATFORM
- IBM CLOUD

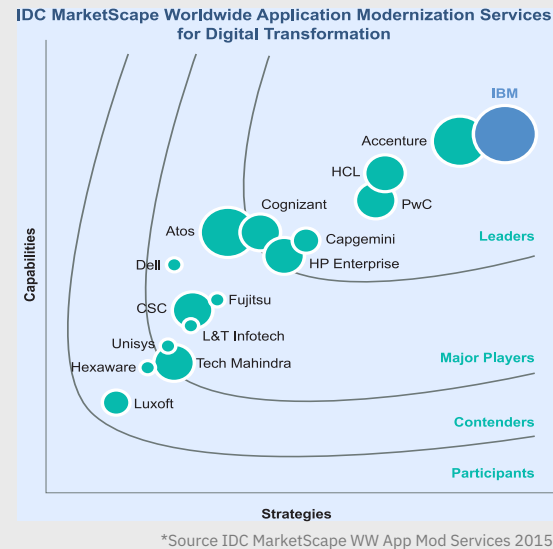
Advise	Move	Build	Manage
E2E advice, design, road mapping, and architectural services to help you navigate the complexity of your multi-cloud journey	Our Garage Method for Cloud employed to assist and accelerate execution; assisted by configurable and automated migration tooling supporting multiple cloud 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

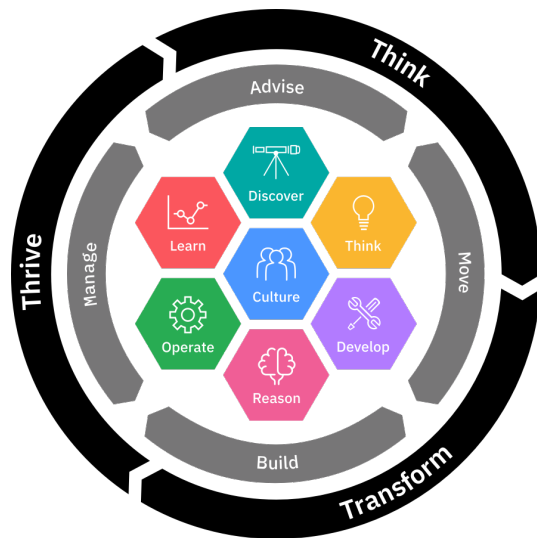
Application Modernization

**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





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

Application Modernization and Transformation

A Case-Study

Rob Peeren, B.Sc.

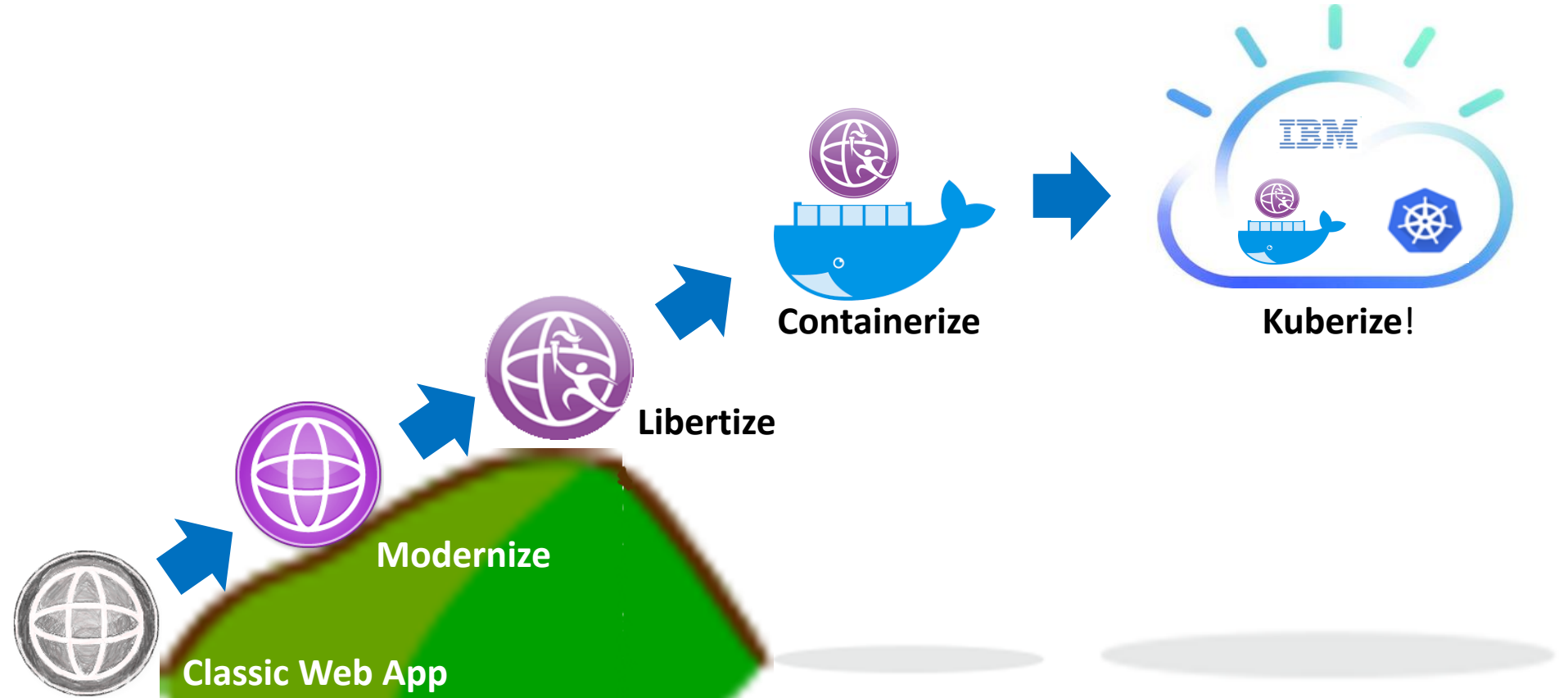
Executive IT Specialist

IBM Canada Ltd.

June 2019

Part 1: Modernization

The Application Modernization Blueprint



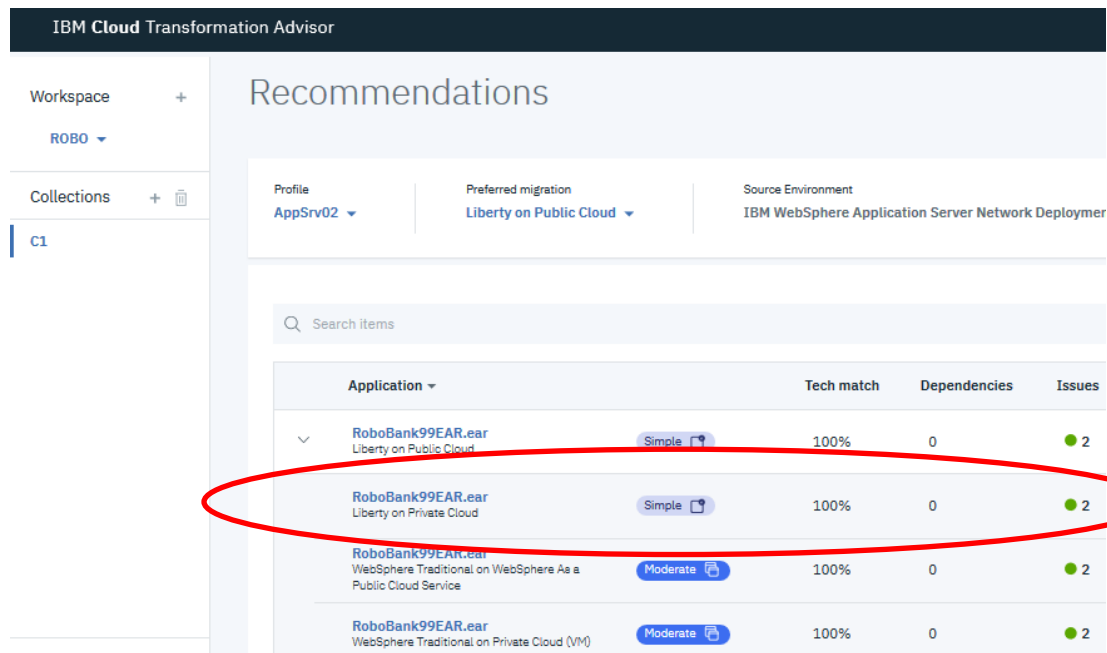
Step 1: Identify the Classic Application to Modernize

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 screenshot shows the IBM Cloud Transformation Advisor interface. The main section is titled "Recommendations". On the left, there is a sidebar with "Workspace" (ROBO) and "Collections" (C1). The main content area shows filters for "Profile" (AppSrv02), "Preferred migration" (Liberty on Public Cloud), and "Source Environment" (IBM WebSphere Application Server Network Deployer). Below the filters is a search bar and a table of recommendations. The table has columns for "Application", "Tech match", "Dependencies", and "Issues". The second row is circled in red, showing a recommendation for "RoboBank99EAR.ear" on "Liberty on Private Cloud" with a "Simple" migration level, 100% tech match, 0 dependencies, and 2 issues.


















Application	Tech match	Dependencies	Issues
RoboBank99EAR.ear Liberty on Public Cloud	100%	0	2
RoboBank99EAR.ear Liberty on Private Cloud	100%	0	2
RoboBank99EAR.ear WebSphere Traditional on WebSphere As a Public Cloud Service	100%	0	2
RoboBank99EAR.ear WebSphere Traditional on Private Cloud (VM)	100%	0	2

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

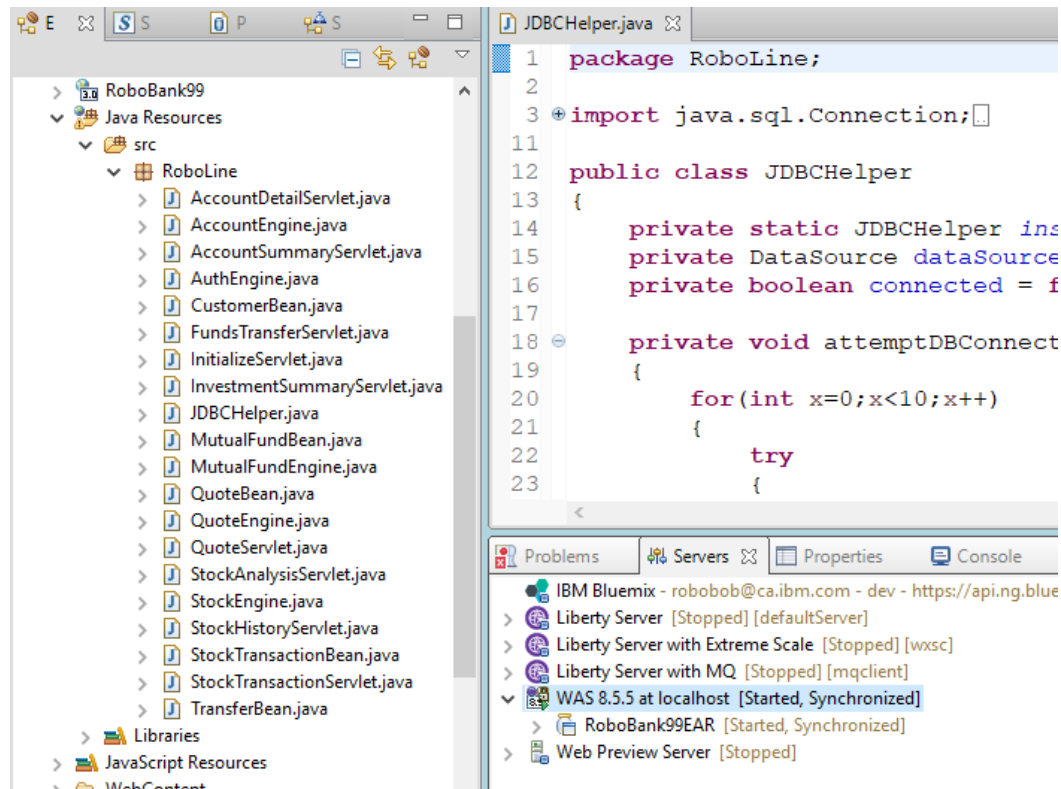
Step 2: Modernize

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.

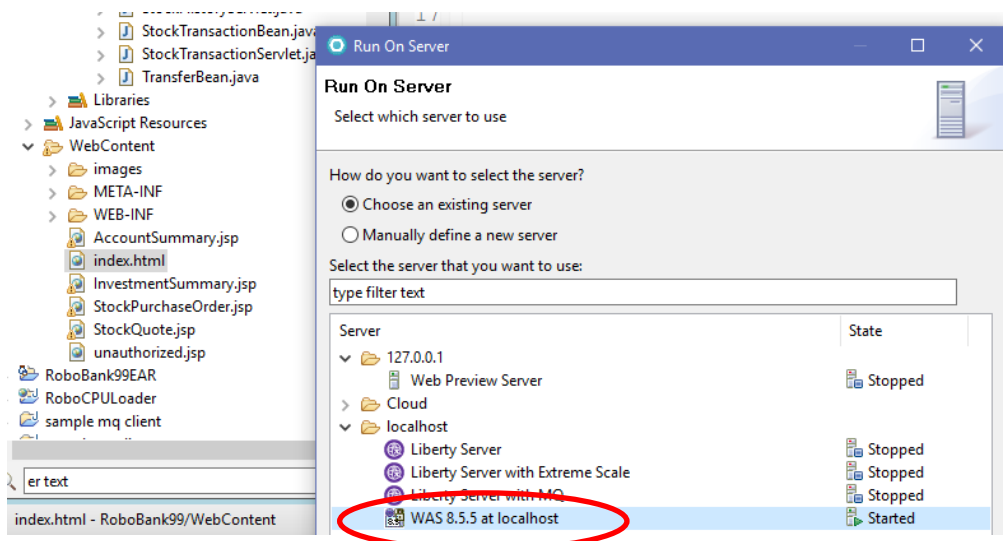


The screenshot displays the Rational Application Developer (RAD) V8.5 interface. On the left, the Project Explorer shows a project named 'RoboBank99' with a 'src' folder containing a 'RoboLine' package. The package contains various Java files, including 'JDBCHelper.java'. On the right, the Code Editor shows the content of 'JDBCHelper.java'.

```
1 package RoboLine;
2
3 import java.sql.Connection;
4
5
6
7
8
9
10
11
12 public class JDBCHelper
13 {
14     private static JDBCHelper ins
15     private DataSource dataSource
16     private boolean connected = f
17
18     private void attemptDBConnect
19     {
20         for(int x=0;x<10;x++)
21         {
22             try
23             {
```

At the bottom of the interface, the Servers view shows several servers, including 'WAS 8.5.5 at localhost [Started, Synchronized]' and 'RoboBank99EAR [Started, Synchronized]'.

Deploy Updated Project to Modern App Server



RoboBank 2000

Rob Peeren

Customer Number: 12341234

Account #	Account Type	Balance	Limit	Balance SCDN
11111-1233	Savings		0.00	2,023.45
12311-4529	Chequing		-1,500.00	2,353.55
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 to TRANSFER

WAS 8.5.5 at localhost (WebSphere Application Server traditional V8.5)

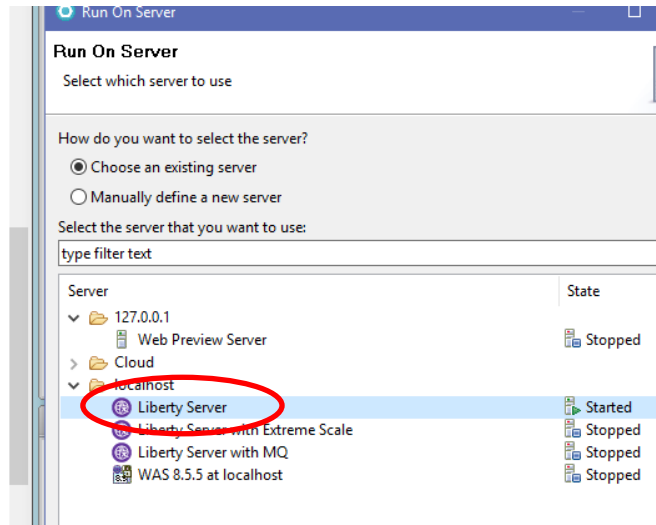
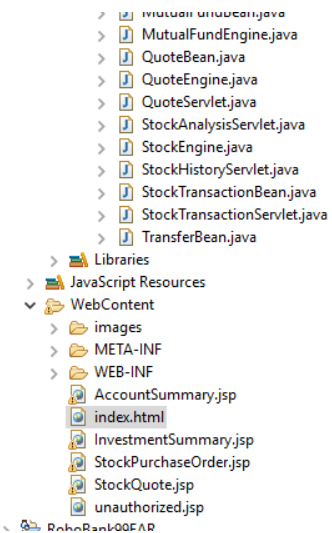
11/1/18 6:48:35:986 EDT1 00000072 ServletWrapper I com.ibm.ws.web

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...



RoboBank 2000

Rob Peeren

Customer Number: 12341234

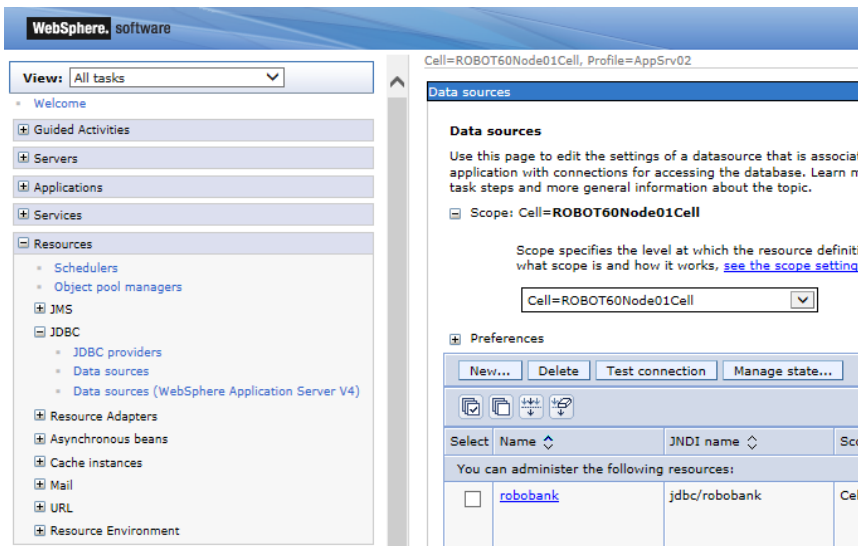
Account #	Account Type	Balance	Limit	Balance \$CDN
11111-1233	Savings		0.00	2,023.45
12311-4529	Chequing		-1,500.00	2,353.55
13334-6712	US Dollar	1,145.95	0.00	1,604.33
21982-6399	Stock Portfolio #1			62,363.00
22782-3909	RRSP			3,094.50
23922-8212	RESP			3,577.00
Total:				75,015.83

Transfer \$ from to TRANSFER

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

```
[AUDIT ] CWWKT0016I: Web application available (def
[AUDIT ] CWWKZ0001I: Application RoboBank99 started
[AUDIT ] CWWKF0012I: The server installed the follow
[AUDIT ] CWWKF0011I: The server defaultServer is re
Attempting connection to RoboBank datasource...
Successfull connection test!
```

What's the Difference?



The screenshot shows the WebSphere Admin Console interface. On the left is a navigation tree with categories like 'Guided Activities', 'Servers', 'Applications', 'Services', and 'Resources'. The 'Resources' section is expanded to show 'Data sources (WebSphere Application Server V4)'. The main content area is titled 'Data sources' and contains instructions on how to edit a data source. It shows a 'Scope' dropdown set to 'Cell=ROBOT60Node01Cell' and a 'Preferences' section with buttons for 'New...', 'Delete', 'Test connection', and 'Manage state...'. Below this is a table of resources:

Select	Name	JNDI name	Cell
<input type="checkbox"/>	robobank	jdbc/robobank	Cell=ROBOT60Node01Cell

```
<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.

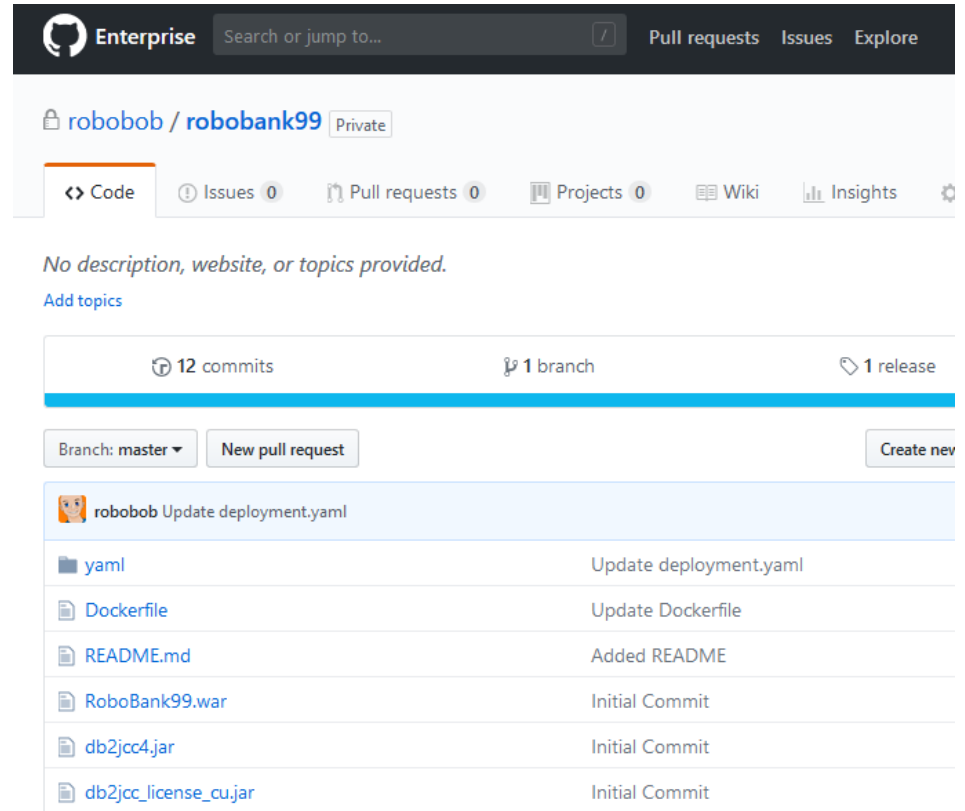
In this case, the web application WAR file, jdbc drivers, and other assets are copied into the container.

```
1 FROM websphere-liberty:webProfile7
2
3 USER root
4 RUN mkdir /opt/ibm/wlp/usr/shared/resources/db2
5 COPY runBank.sh /usr/local/bin
6 RUN chmod +x /usr/local/bin/runBank.sh
7 COPY db2jcc4.jar /opt/ibm/wlp/usr/shared/resources/db2
8 COPY db2jcc_license_cu.jar /opt/ibm/wlp/usr/shared/resources/db2
9 COPY server.xml /opt/ibm/wlp/usr/servers/defaultServer
10 COPY RoboBank99.war /opt/ibm/wlp/usr/servers/defaultServer/apps
11 RUN /opt/ibm/wlp/bin/installUtility install defaultServer
12 USER 1001
13
14
15 EXPOSE 9080 9443
16
17 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.



The screenshot shows a GitHub repository page for 'robobob / robobank99' (Private). The repository has 12 commits, 1 branch, and 1 release. The current branch is 'master'. A 'New pull request' button is visible. The commit history shows the following files and their commit types:

File	Commit Type
yaml	Update deployment.yaml
Dockerfile	Update Dockerfile
README.md	Added README
RoboBank99.war	Initial Commit
db2jcc4.jar	Initial Commit
db2jcc_license_cu.jar	Initial Commit

Use an Image Build Tool...

The screenshot shows the Jenkins web interface for 'Project Robobank99'. The top navigation bar includes 'Jenkins' and 'Robobank99'. A left sidebar contains navigation links: 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', 'Configure', and 'Move'. The main content area is titled 'Project Robobank99' and includes links for 'Workspace' and 'Recent Changes'. Below this is a 'Permalinks' section with a list of build links. At the bottom left, there is a 'Build History' widget with a search bar and a list of build entries, including '#5 registry.ng.bluemix.net/robobob/robobank:2018.5' and 'registry.ng.bluemix.net/robobob/robobank:latest'.



The screenshot shows the 'IBM Cloud Private' interface for 'Images'. A list of image names is displayed: 'default/robobank', 'default/redis-ha', 'default/robobankloader', 'default/rstudio-d8a2rls2x-shell', and 'default/sentimentsampler'. The 'default/robobank' entry is circled in red, indicating it is the image built by Jenkins.

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 a 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.

The screenshot displays the UrbanCode Deploy web interface. At the top, the navigation bar includes 'UrbanCode Deploy' and several menu items: 'Welcome', 'Dashboard', 'Components', 'Applications' (which is highlighted), 'Configuration', 'Processes', 'Resources', 'Calendar', and 'Work Items'. Below the navigation bar, the breadcrumb trail reads 'Home / Applications / RoboBank'. The main heading is 'Application: RoboBank' with a 'Show details' link. A secondary navigation bar contains 'Environments' (highlighted), 'History', 'Configuration', 'Components', 'Blueprints', 'Snapshots', and 'Processes'. The main content area features a message: 'Drag environments by their names to re-order them. 1 Environment'. Below this is a search bar with 'Search by Name' and 'Search by Blueprint' options. A table lists the environment 'DEV-ICP' with a 'Snapshot: None' status.

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 x RoboLine Customer Account Summ x +

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 to 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
Withdrawal	2,423.45	700.00	1,723.45

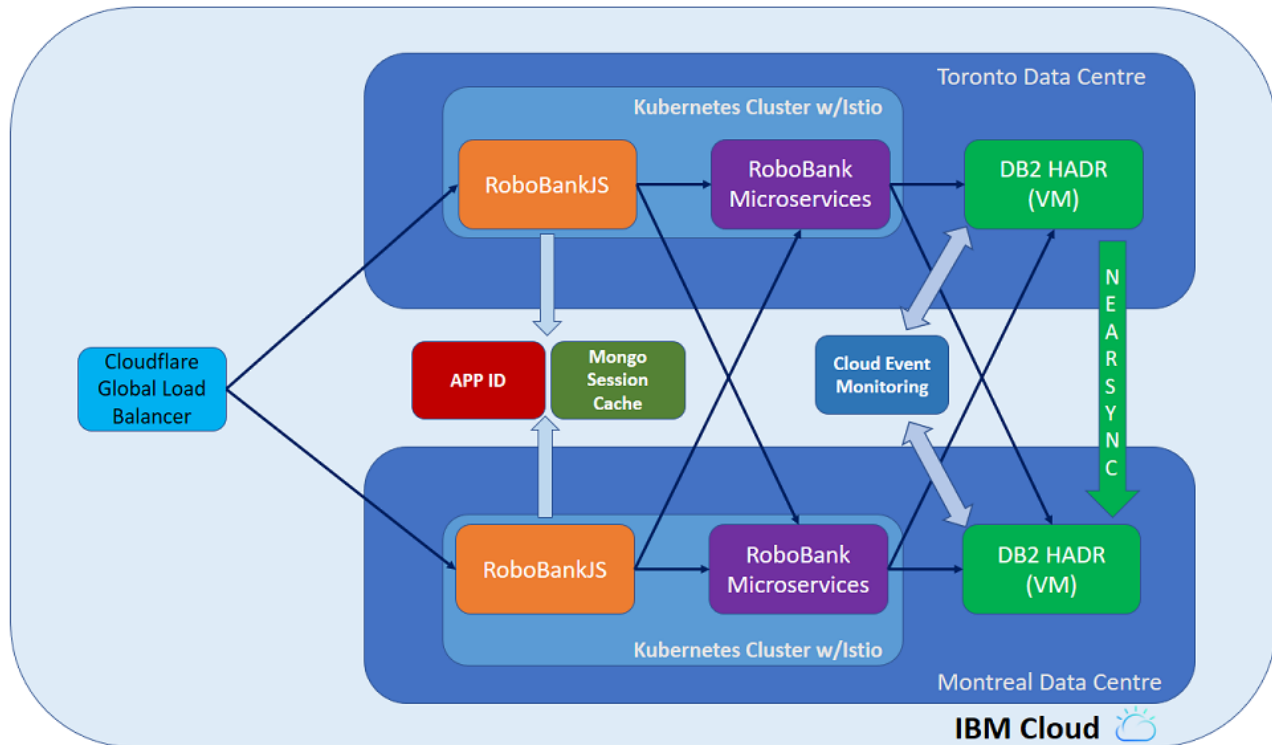
Once deployed, the application is available in the Kubernetes environment, in this case IBM Cloud Private.

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.

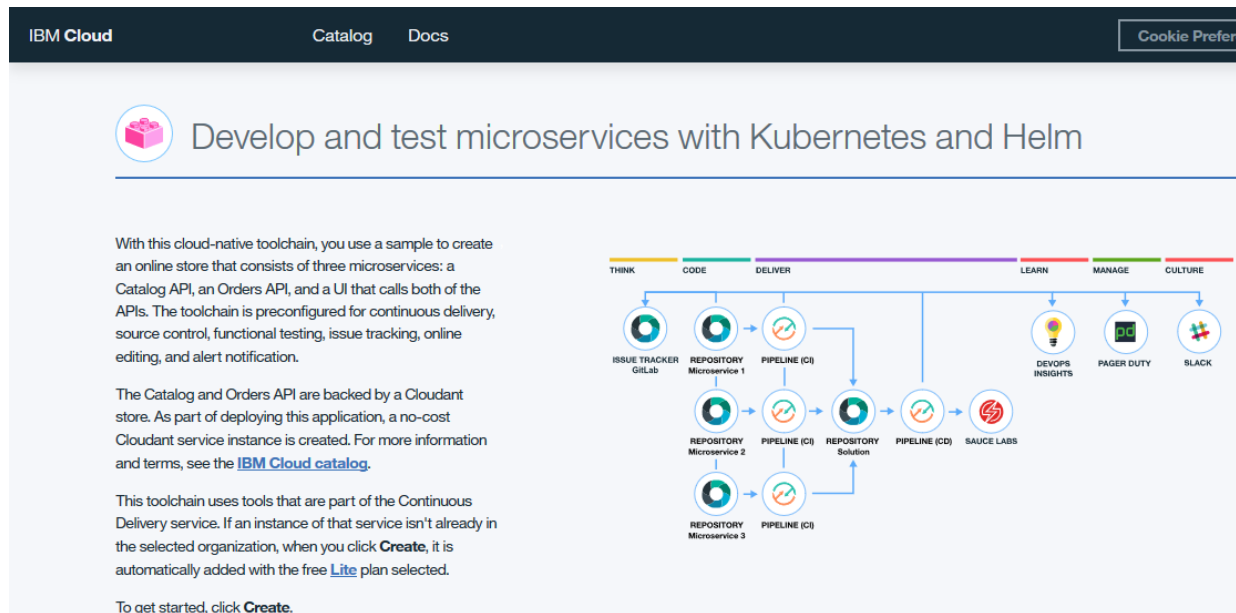
Refactoring applications to use microservices and Istio, introducing new programming languages such as NodeJS, and leveraging cloud services to manage complex tasks are all important drivers in cloud transformation.



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.



The screenshot shows the IBM Cloud Catalog interface. At the top, there are navigation links for 'Catalog' and 'Docs', and a 'Cookie Preferences' button. The main heading is 'Develop and test microservices with Kubernetes and Helm'. Below the heading, there are three paragraphs of text describing the toolchain and how to use it. To the right of the text is a diagram illustrating the CI/CD pipeline and DevOps tools.

IBM Cloud Catalog Docs Cookie Preferences

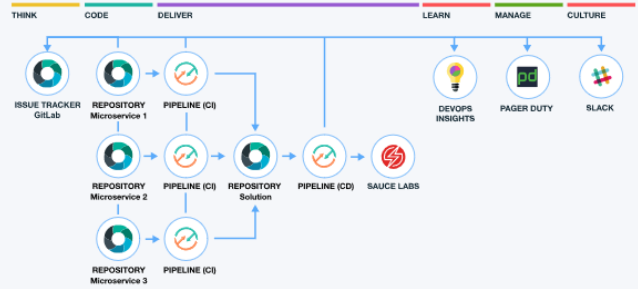
Develop and test microservices with Kubernetes and Helm

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**.



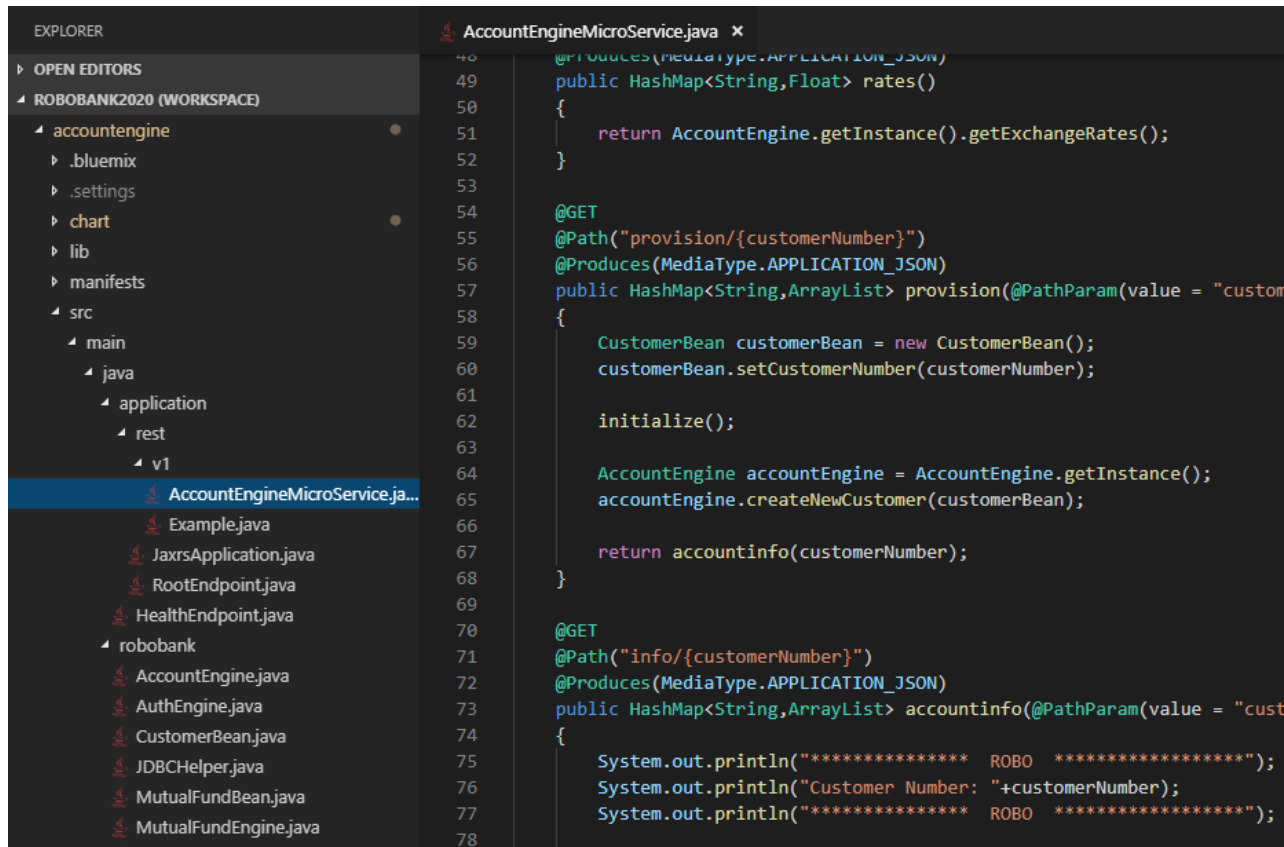
The diagram illustrates a CI/CD pipeline and DevOps tools. It is organized into six stages: THINK, CODE, DELIVER, LEARN, MANAGE, and CULTURE. The pipeline starts with 'ISSUE TRACKER GitHub' in the THINK stage. In the CODE stage, there are three 'REPOSITORY Microservice' instances (1, 2, and 3). In the DELIVER stage, there are three 'PIPELINE (CI)' instances and one 'REPOSITORY Solution'. In the LEARN stage, there is 'DEVOPS INSIGHTS'. In the MANAGE stage, there is 'PAGER DUTY'. In the CULTURE stage, there is 'SLACK'. The diagram shows arrows indicating the flow of the pipeline and the integration of various tools.

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.java
                  └─ Example.java
                  └─ JaxrsApplication.java
                  └─ RootEndpoint.java
                  └─ HealthEndpoint.java
            └─ robobank
              └─ AccountEngine.java
              └─ AuthEngine.java
              └─ CustomerBean.java
              └─ JDBCHelper.java
              └─ MutualFundBean.java
              └─ MutualFundEngine.java

AccountEngineMicroService.java
48 @Produces(MediaType.APPLICATION_JSON)
49 public HashMap<String,Float> rates()
50 {
51     return AccountEngine.getInstance().getExchangeRates();
52 }
53
54 @GET
55 @Path("provision/{customerNumber}")
56 @Produces(MediaType.APPLICATION_JSON)
57 public HashMap<String,ArrayList> provision(@PathParam(value = "customerNumber") String customerNumber)
58 {
59     CustomerBean customerBean = new CustomerBean();
60     customerBean.setCustomerNumber(customerNumber);
61
62     initialize();
63
64     AccountEngine accountEngine = AccountEngine.getInstance();
65     accountEngine.createNewCustomer(customerBean);
66
67     return accountinfo(customerNumber);
68 }
69
70 @GET
71 @Path("info/{customerNumber}")
72 @Produces(MediaType.APPLICATION_JSON)
73 public HashMap<String,ArrayList> accountinfo(@PathParam(value = "customerNumber") String customerNumber)
74 {
75     System.out.println("***** ROBO *****");
76     System.out.println("Customer Number: "+customerNumber);
77     System.out.println("***** ROBO *****");
78 }
```

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.

The screenshot displays the IBM Cloud DevOps interface for a 'Delivery Pipeline'. The top navigation bar includes 'IBM Cloud', 'Catalog', 'Docs', 'Support', 'Manage', and a search bar. The breadcrumb trail shows 'Toolchains / accountengine / Pipeline'. The main title is 'Pipeline | Delivery Pipeline'. The pipeline consists of three stages:

- Build Stage:** STAGE PASSED. Last input: Git URL. Last commit by robobob updated to assign root during build (9d ago). Job: Build Passed 9d ago.
- Toronto Deploy:** STAGE PASSED. Last input: Stage: Build Stage / Job: B... Build 82. Job: Deploy Passed 9d ago.
- Montreal Deploy:** STAGE PASSED. Last input: Stage: Build Stage / Job: B... Build 82. Job: Deploy Passed 9d ago.

Each stage card includes a 'View logs and history' link and a 'LAST EXECUTION RESULT' section.

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 All CATEGORY

kubernetes

Search

Clusters

Name ▾
Montreal
Toronto

Cluster

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

Namespace default ▾

Overview

Workloads

Cron Jobs

Daemon Sets

CPU usage

Time	CPU (cores)
11:39	1.6
11:40	1.6
11:43	1.7
11:46	1.6
11:50	1.7

Namespaces

Name ▾	Labels
✔ robank	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

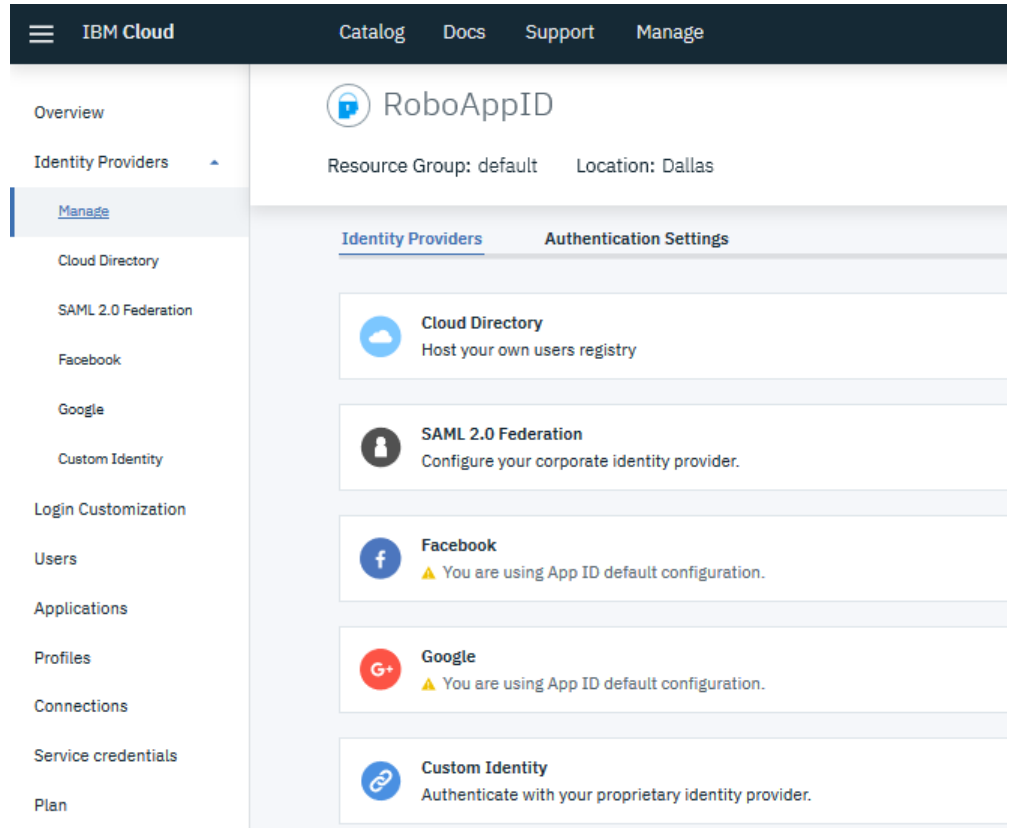
Through IBM's CloudFlare partnership, IBM offers a global load balancing service providing up to five-9's availability to cloud-based workloads.

The screenshot shows the IBM Cloud console interface. The top navigation bar includes 'IBM Cloud', 'Catalog', 'Docs', 'Support', and 'Manage'. The left sidebar contains a menu with items: 'Getting started', 'Overview', 'Metrics', 'Security', 'Reliability', 'Global Load Balancers' (highlighted), 'DNS', 'Performance', and 'Plan'. The main content area is titled 'Dashboard / RoboInternetServices' and shows 'Resource Group: default' and 'Location: Global'. The 'Global Load Balancers' section includes a description: '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.' Below this, there are tabs for 'Manage' and 'Health Check Events'. The 'Load Balancers' section contains a table with the following data:

	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.

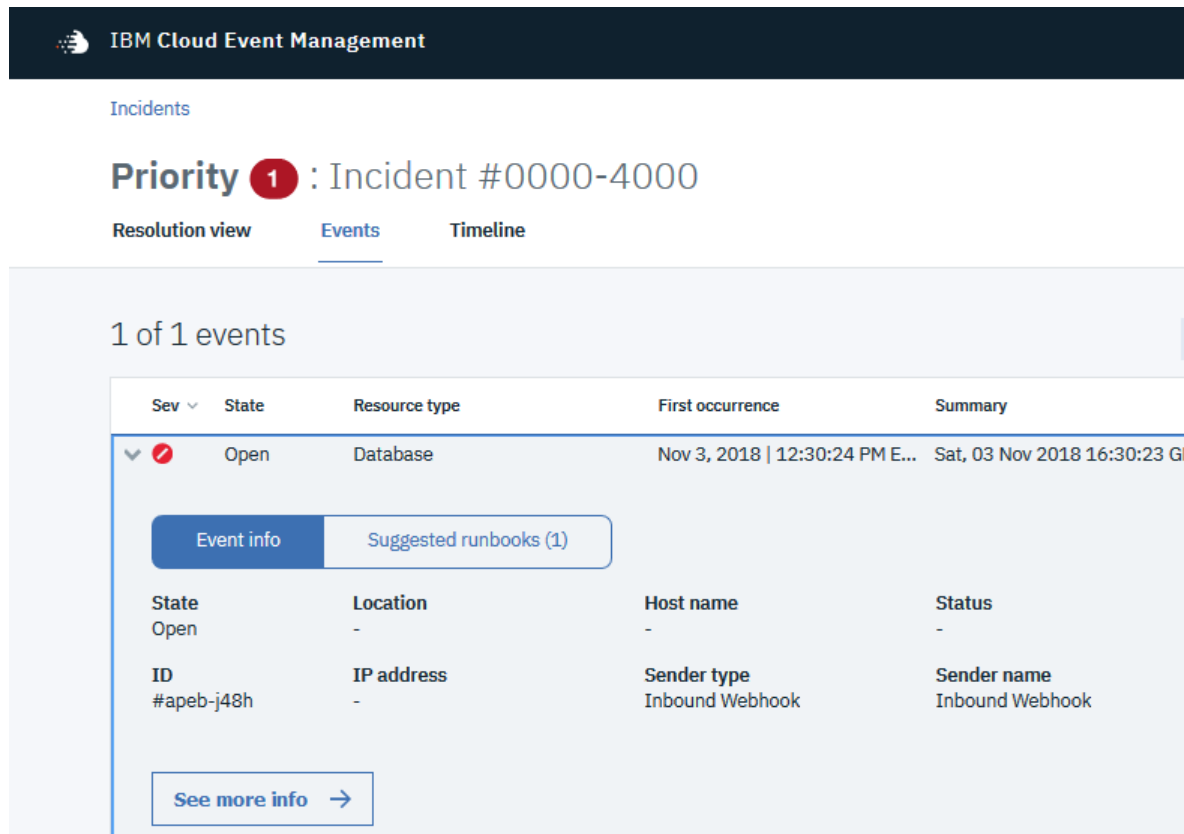


The screenshot displays the IBM Cloud management interface for the RoboAppID service. The top navigation bar includes 'IBM Cloud', 'Catalog', 'Docs', 'Support', and 'Manage'. The left sidebar lists various management options, with 'Manage' selected under 'Identity Providers'. The main content area shows the 'RoboAppID' configuration for 'Resource Group: default' and 'Location: Dallas'. It features two tabs: 'Identity Providers' (active) and 'Authentication Settings'. Under 'Identity Providers', four providers are listed:

- Cloud Directory**: Host your own users registry.
- SAML 2.0 Federation**: Configure your corporate identity provider.
- Facebook**: You are using App ID default configuration.
- Google**: You are using App ID default configuration.
- Custom Identity**: Authenticate with your proprietary identity provider.

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.



IBM Cloud Event Management

Incidents

Priority 1 : Incident #0000-4000

Resolution view **Events** Timeline

1 of 1 events

Sev	State	Resource type	First occurrence	Summary
✓	Open	Database	Nov 3, 2018 12:30:24 PM E...	Sat, 03 Nov 2018 16:30:23 GM

Event info | Suggested runbooks (1)

State	Location	Host name	Status
Open	-	-	-

ID	IP address	Sender type	Sender name
#apeb-j48h	-	Inbound Webhook	Inbound Webhook

See more info →

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	▼	> icp31-1.robobob.ca	Virtual Server	Toronto 1	
Storage	▼	> icp31-w1.robobob.ca	Virtual Server	Toronto 1	
Network	▼	> icp31-w2.robobob.ca	Virtual Server	Toronto 1	
Security	▼	> icp31-w3.robobob.ca	Virtual Server	Toronto 1	
Services	▼	> icp31-w4.robobob.ca	Virtual Server	Toronto 1	
		> icp4.robobob.ca	Virtual Server	Toronto 1	
		> icp5.robobob.ca	Virtual Server	Toronto 1	
		> kube-mon01-cr751ef7ab97c74631944355e6f	Virtual Server	Montreal 1	
		> kube-mon01-cr751ef7ab97c74631944355e6f	Virtual Server	Montreal 1	
		> kube-mon01-cr751ef7ab97c74631944355e6f	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.

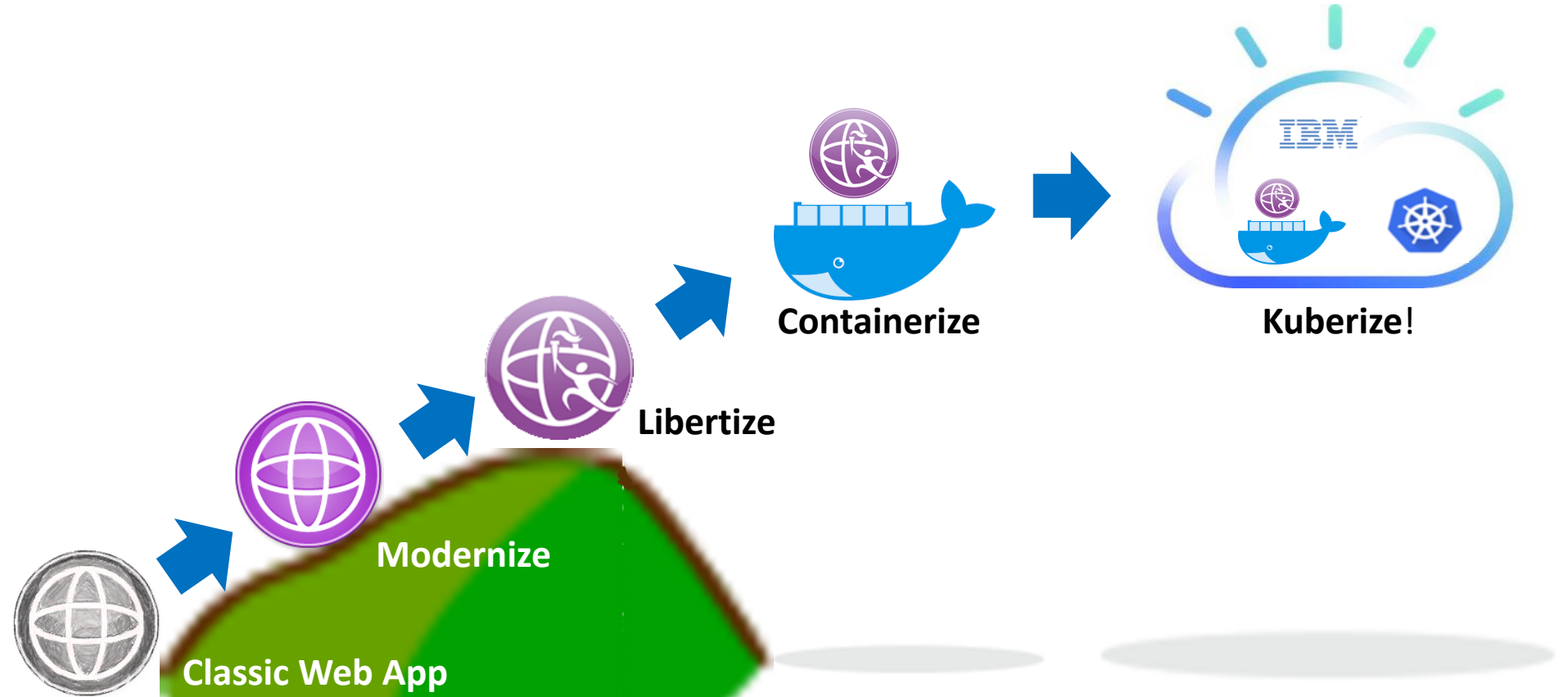
Executive IT Specialist

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 screenshot displays the IBM Cloud Transformation Advisor interface. The main section is titled "Recommendations". On the left, there is a sidebar with "Workspace" (ROBO) and "Collections" (C1). The main content area shows filters for "Profile" (AppSrv02), "Preferred migration" (Liberty on Public Cloud), and "Source Environment" (IBM WebSphere Application Server Network Deployer). Below the filters is a search bar and a table of recommendations.

Application	Tech match	Dependencies	Issues
RoboBank99EAR.ear Liberty on Public Cloud Simple	100%	0	2
RoboBank99EAR.ear Liberty on Private Cloud Simple	100%	0	2
RoboBank99EAR.ear WebSphere Traditional on WebSphere As a Public Cloud Service Moderate	100%	0	2
RoboBank99EAR.ear WebSphere Traditional on Private Cloud (VM) Moderate	100%	0	2


















A red oval highlights the second row of the table, which represents the "Liberty on Private Cloud" migration option.

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

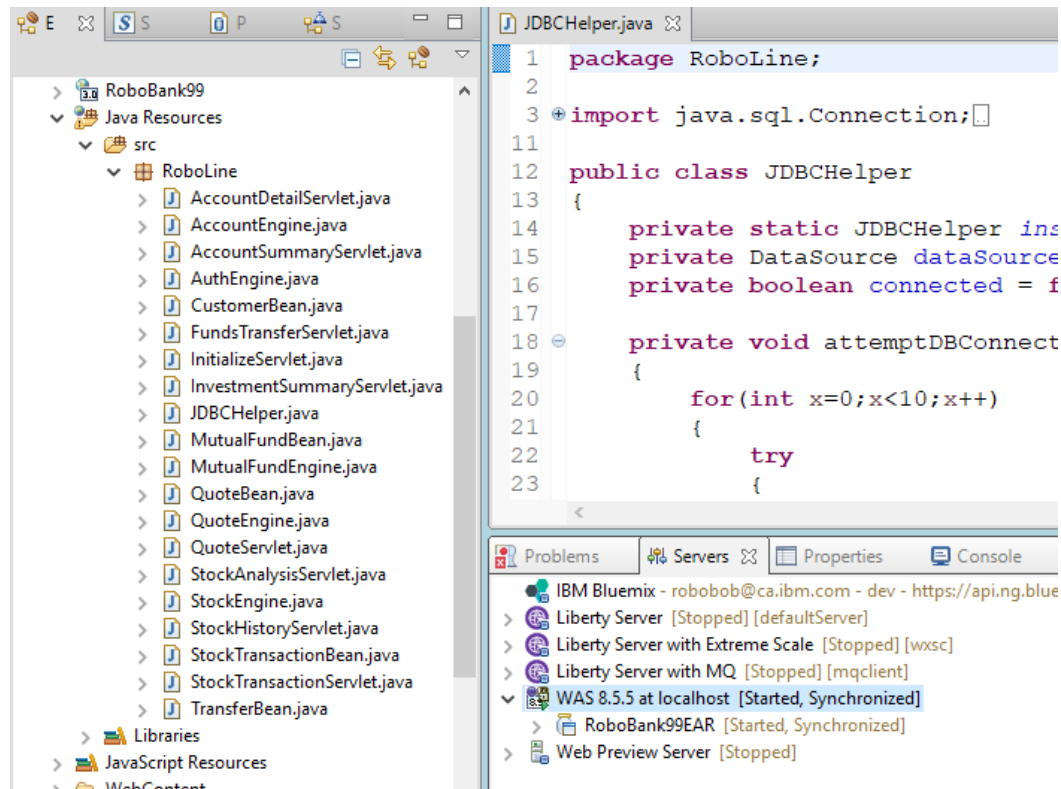
Step 2: Modernize

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.

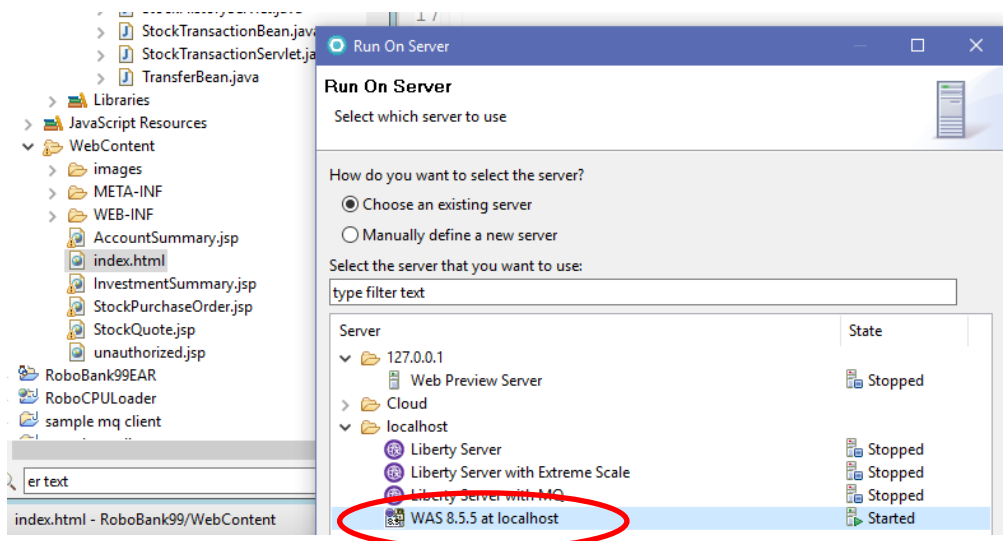


The screenshot displays the Rational Application Developer (RAD) V8.5 interface. On the left, the Project Explorer shows a project named 'RoboBank99' with a 'src' folder containing a 'RoboLine' package. The package contains various Java files, including 'JDBCHelper.java'. On the right, the Editor window shows the code for 'JDBCHelper.java'. The code includes a package declaration, an import statement for 'java.sql.Connection', and a public class 'JDBCHelper' with several private static fields and a private void method 'attemptDBConnect'.

```
1 package RoboLine;
2
3 import java.sql.Connection;
4
5
6
7
8
9
10
11
12 public class JDBCHelper
13 {
14     private static JDBCHelper ins
15     private DataSource dataSource
16     private boolean connected = f
17
18     private void attemptDBConnect
19     {
20         for(int x=0;x<10;x++)
21         {
22             try
23             {
```

At the bottom of the interface, the Servers view shows several servers, including 'WAS 8.5.5 at localhost [Started, Synchronized]' and 'RoboBank99EAR [Started, Synchronized]'.

Deploy Updated Project to Modern App Server



RoboBank 2000

Rob Peeren

Customer Number: 12341234

Account #	Account Type	Balance	Limit	Balance SCDN
11111-1233	Savings		0.00	2,023.45
12311-4529	Chequing		-1,500.00	2,353.55
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

WAS 8.5.5 at localhost (WebSphere Application Server traditional V8.5)

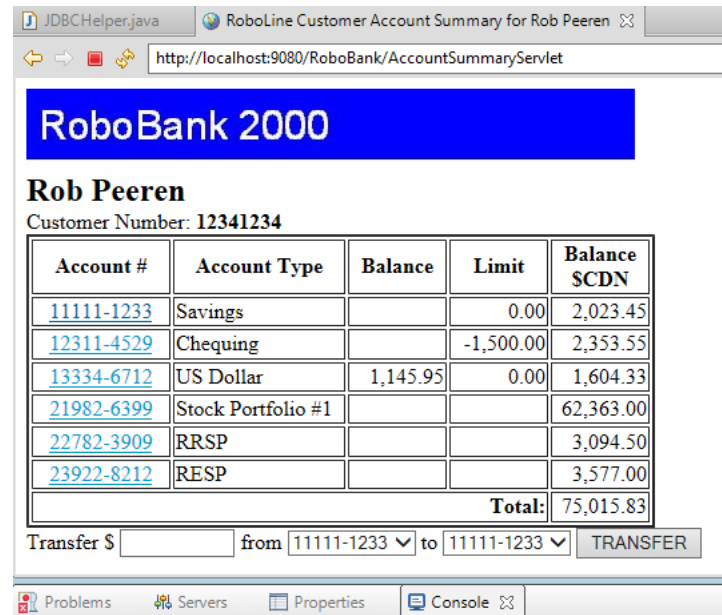
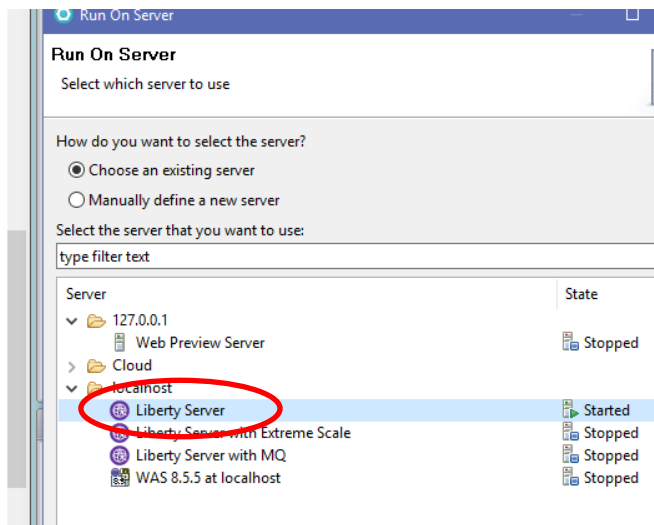
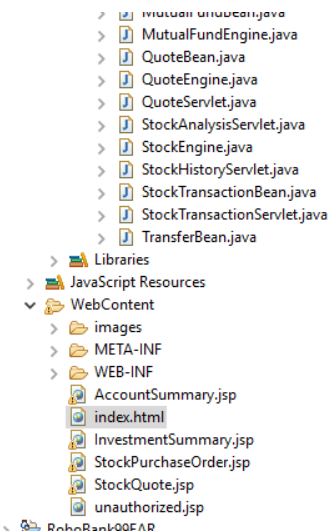
11/1/18 6:48:35:986 EDT1 00000072 ServletWrapper I com.ibm.ws.web

WAS 8.5.5 is made the run target of the application...

And it runs!

Step 3: Libertize

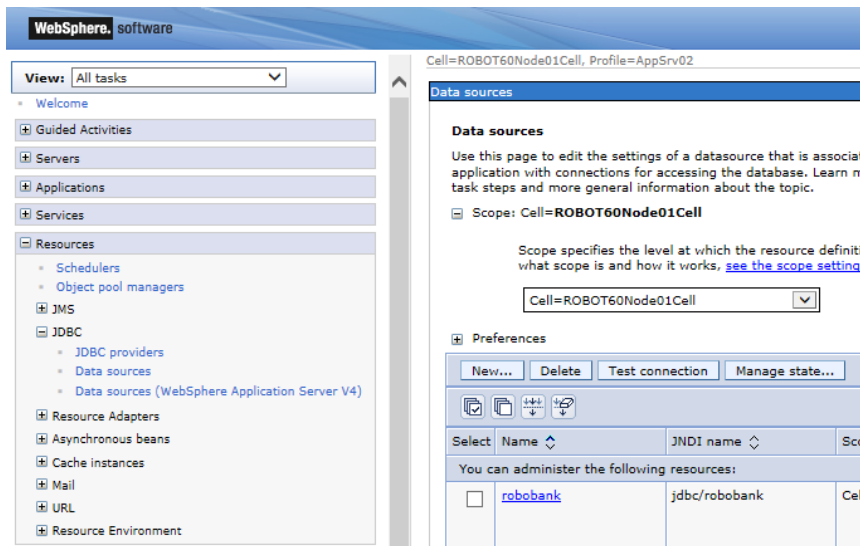
Repeat the Deploy to Liberty Application Server...



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

```
[AUDIT ] CWWKT0016I: Web application available (def
[AUDIT ] CWWKZ0001I: Application RoboBank99 started
[AUDIT ] CWWKF0012I: The server installed the follow
[AUDIT ] CWWKF0011I: The server defaultServer is re
Attempting connection to RoboBank datasource...
Successfull connection test!
```

What's the Difference?



The screenshot shows the WebSphere Admin Console interface. On the left is a navigation tree with categories like 'Guided Activities', 'Servers', 'Applications', 'Services', and 'Resources'. Under 'Resources', 'JDBC' is expanded to show 'Data sources (WebSphere Application Server V4)'. The main content area is titled 'Data sources' and contains instructions on how to edit a data source. It shows a 'Scope' dropdown set to 'Cell=ROBOT60Node01Cell' and a 'Preferences' section with buttons for 'New...', 'Delete', 'Test connection', and 'Manage state...'. Below this is a table of resources:

Select	Name	JNDI name	Cell
<input type="checkbox"/>	robobank	jdbc/robobank	Cell=ROBOT60Node01Cell

```
<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.

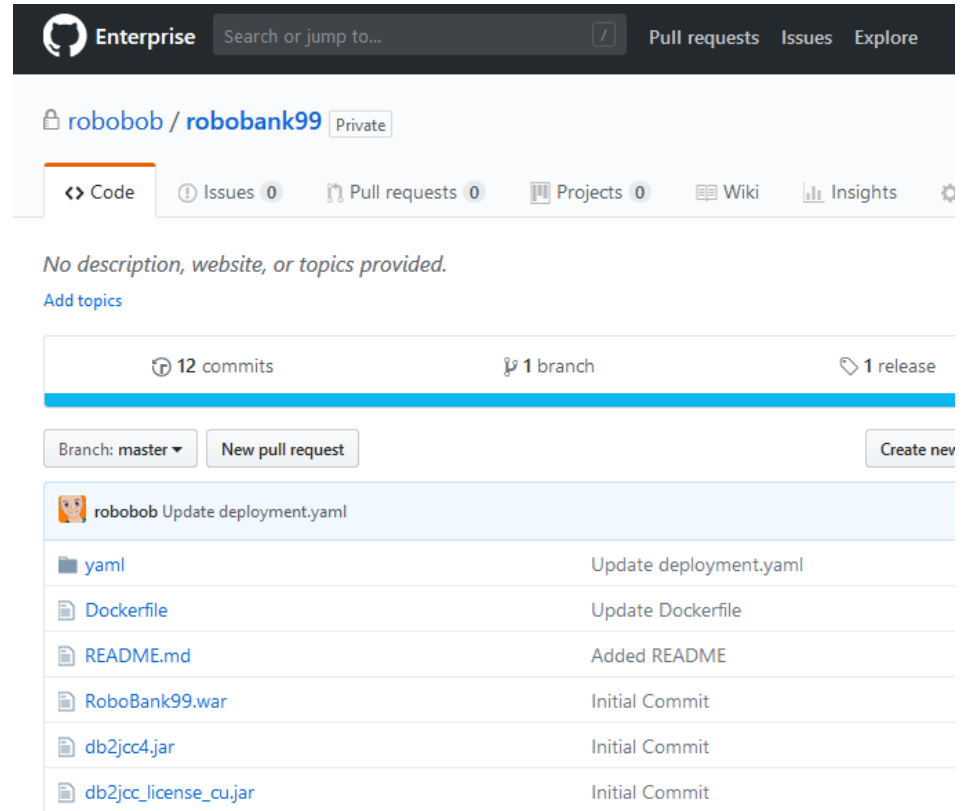
In this case, the web application WAR file, jdbc drivers, and other assets are copied into the container.

```
1 FROM websphere-liberty:webProfile7
2
3 USER root
4 RUN mkdir /opt/ibm/wlp/usr/shared/resources/db2
5 COPY runBank.sh /usr/local/bin
6 RUN chmod +x /usr/local/bin/runBank.sh
7 COPY db2jcc4.jar /opt/ibm/wlp/usr/shared/resources/db2
8 COPY db2jcc_license_cu.jar /opt/ibm/wlp/usr/shared/resources/db2
9 COPY server.xml /opt/ibm/wlp/usr/servers/defaultServer
10 COPY RoboBank99.war /opt/ibm/wlp/usr/servers/defaultServer/apps
11 RUN /opt/ibm/wlp/bin/installUtility install defaultServer
12 USER 1001
13
14
15 EXPOSE 9080 9443
16
17 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.



The screenshot shows a GitHub repository page for 'robobob / robobank99' (Private). The repository has 12 commits, 1 branch, and 1 release. The current branch is 'master'. A 'New pull request' button is visible. The commit history is shown below, with the most recent commit by 'robobob' updating 'deployment.yaml'. The commit list includes:

File	Commit Message
yaml	Update deployment.yaml
Dockerfile	Update Dockerfile
README.md	Added README
RoboBank99.war	Initial Commit
db2jcc4.jar	Initial Commit
db2jcc_license_cu.jar	Initial Commit

Use an Image Build Tool...

The screenshot shows the Jenkins web interface for 'Project Robobank99'. The top navigation bar includes 'Jenkins' and 'Robobank99'. A left sidebar contains navigation links: 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', 'Configure', and 'Move'. The main content area is titled 'Project Robobank99' and includes links for 'Workspace' and 'Recent Changes'. Below this is a 'Permalinks' section with a list of build links. At the bottom left, there is a 'Build History' section with a search box and a list of build entries, including '#5 registry.ng.bluemix.net/robobob/robobank:2018.5' and 'registry.ng.bluemix.net/robobob/robobank:latest'.



The screenshot shows the 'IBM Cloud Private' interface, specifically the 'Images' section. A list of images is displayed, including 'default/robobank', 'default/redis-ha', 'default/robobankloader', 'default/rstudio-d8a2rls2x-shell', and 'default/sentimentsampler'. The 'default/robobank' entry is circled in red.

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 a 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.

The screenshot displays the UrbanCode Deploy web interface. At the top, the navigation bar includes 'UrbanCode Deploy' and several menu items: 'Welcome', 'Dashboard', 'Components', 'Applications' (which is highlighted), 'Configuration', 'Processes', 'Resources', 'Calendar', and 'Work Items'. Below the navigation bar, the breadcrumb trail reads 'Home / Applications / RoboBank'. The main heading is 'Application: RoboBank' with a 'Show details' link. A secondary navigation bar contains 'Environments' (highlighted), 'History', 'Configuration', 'Components', 'Blueprints', 'Snapshots', and 'Processes'. The main content area features a message: 'Drag environments by their names to re-order them. 1 Environment'. Below this is a search bar with 'Search by Name' and 'Search by Blueprint' options. A table lists the environment 'DEV-ICP' with a 'Snapshot: None' status.

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 x RoboLine Customer Account Summ x +

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 to 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
Withdrawal	2,423.45	700.00	1,723.45

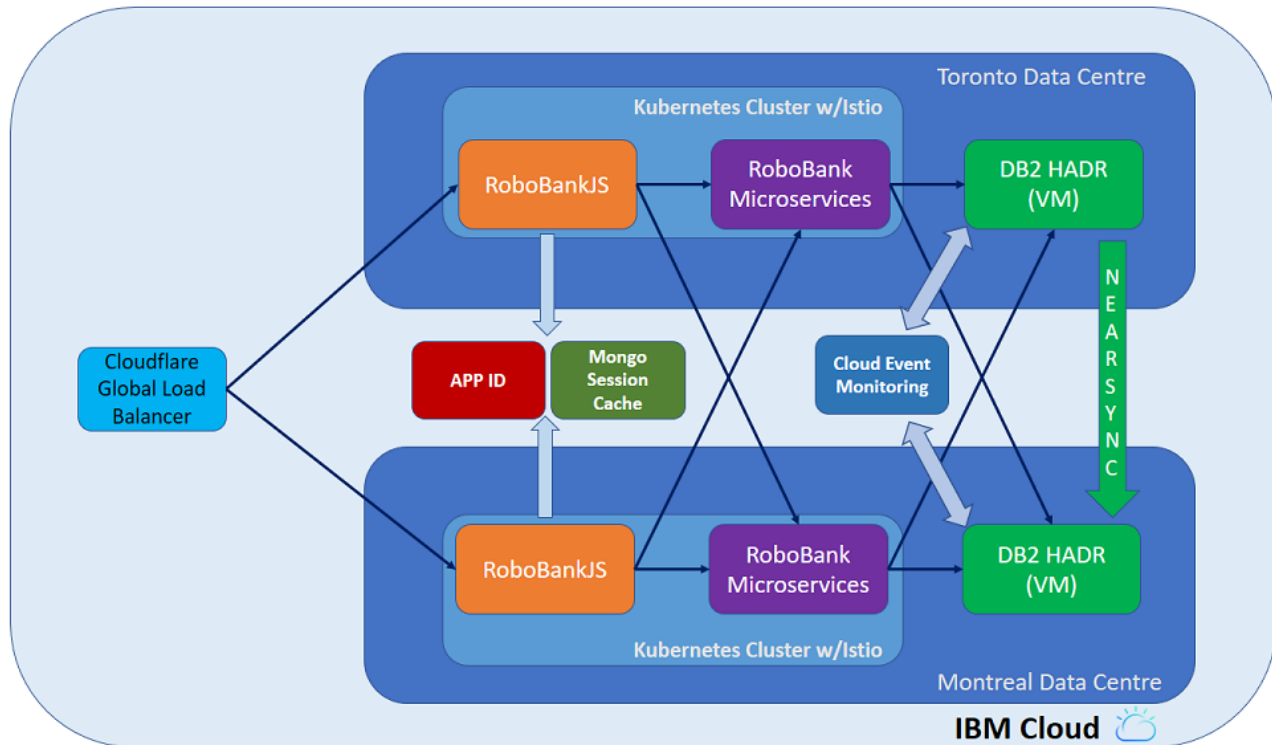
Once deployed, the application is available in the Kubernetes environment, in this case IBM Cloud Private.

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 Preferences

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**.

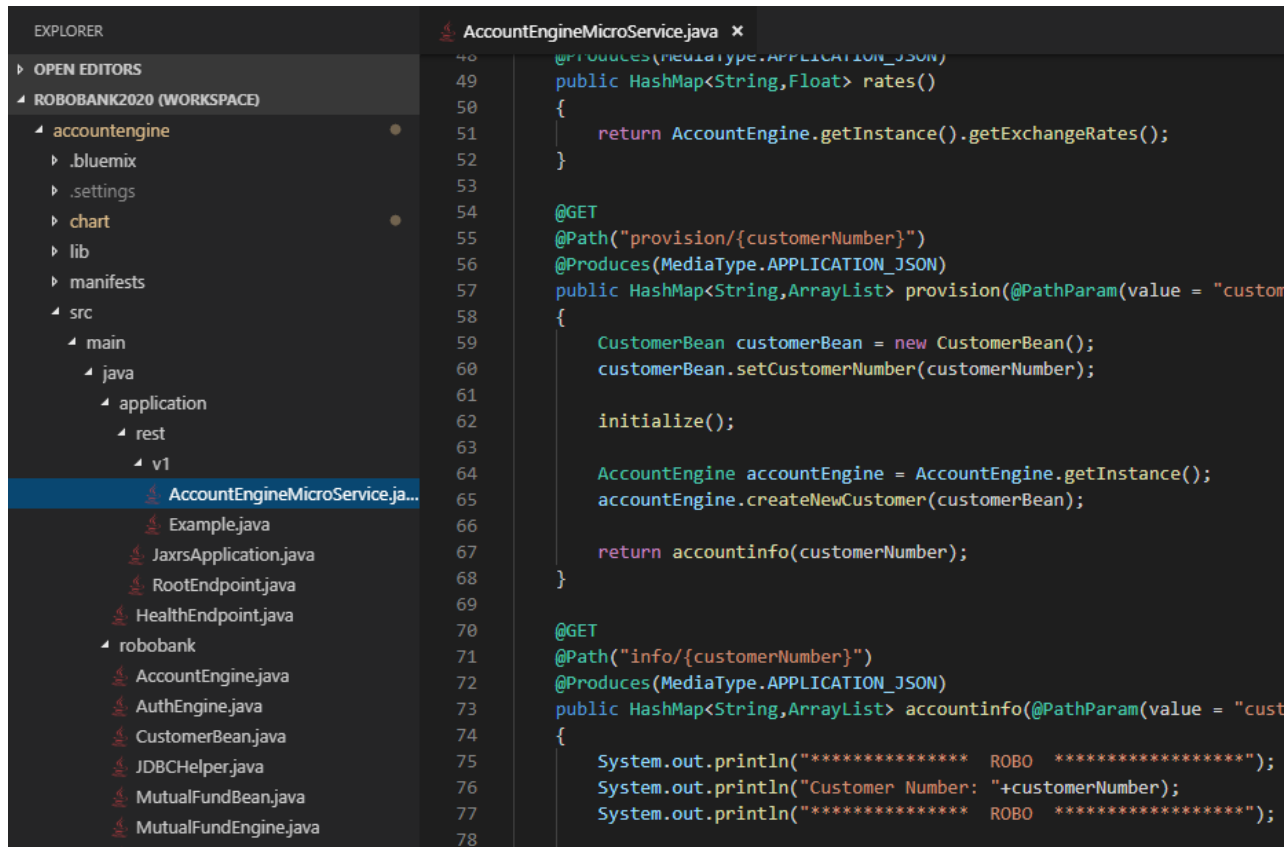
```
graph TD; subgraph THINK; IT[ISSUE TRACKER GitHub]; end; subgraph CODE; R1[REPOSITORY Microservice 1]; R2[REPOSITORY Microservice 2]; R3[REPOSITORY Microservice 3]; end; subgraph DELIVER; P1[PIPELINE CI]; P2[PIPELINE CI]; P3[PIPELINE CI]; RS[REPOSITORY Solution]; end; subgraph LEARN; DI[DEVOPS INSIGHTS]; end; subgraph MANAGE; PD[PAGER DUTY]; end; subgraph CULTURE; S[SLACK]; end; R1 --> P1; R2 --> P2; R3 --> P3; P1 --> RS; P2 --> RS; P3 --> RS; RS --> P4[PIPELINE CD]; P4 --> SL[SAUCE LABS]; IT --> P1; IT --> P2; IT --> P3; IT --> DI; IT --> PD; IT --> S;
```

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.java
                  └─ Example.java
                  └─ JaxrsApplication.java
                  └─ RootEndpoint.java
                  └─ HealthEndpoint.java
            └─ robobank
              └─ AccountEngine.java
              └─ AuthEngine.java
              └─ CustomerBean.java
              └─ JDBCHelper.java
              └─ MutualFundBean.java
              └─ MutualFundEngine.java

AccountEngineMicroService.java
48 @Produces(MediaType.APPLICATION_JSON)
49 public HashMap<String,Float> rates()
50 {
51     return AccountEngine.getInstance().getExchangeRates();
52 }
53
54 @GET
55 @Path("provision/{customerNumber}")
56 @Produces(MediaType.APPLICATION_JSON)
57 public HashMap<String,ArrayList> provision(@PathParam(value = "customerNumber") String customerNumber)
58 {
59     CustomerBean customerBean = new CustomerBean();
60     customerBean.setCustomerNumber(customerNumber);
61
62     initialize();
63
64     AccountEngine accountEngine = AccountEngine.getInstance();
65     accountEngine.createNewCustomer(customerBean);
66
67     return accountinfo(customerNumber);
68 }
69
70 @GET
71 @Path("info/{customerNumber}")
72 @Produces(MediaType.APPLICATION_JSON)
73 public HashMap<String,ArrayList> accountinfo(@PathParam(value = "customerNumber") String customerNumber)
74 {
75     System.out.println("***** ROBO *****");
76     System.out.println("Customer Number: "+customerNumber);
77     System.out.println("***** ROBO *****");
78 }
```


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.

The screenshot displays the IBM Cloud DevOps interface for a 'Delivery Pipeline'. The navigation bar at the top includes 'IBM Cloud', 'Catalog', 'Docs', 'Support', 'Manage', and a search bar. The breadcrumb trail shows 'Toolchains / accountengine / Pipeline'. The main title is 'Pipeline | Delivery Pipeline'.

The pipeline consists of three stages, each with a 'STAGE PASSED' status:

- Build Stage:** Shows 'LAST INPUT' as a Git URL with a commit by 'robobob' updated to assign root during build, 9 days ago. The 'JOBS' section shows a 'Build' job that passed 9 days ago. The 'LAST EXECUTION RESULT' is 'Build 82'.
- Toronto Deploy:** Shows 'LAST INPUT' as 'Stage: Build Stage / Job: B...'. The 'JOBS' section shows a 'Deploy' job that passed 9 days ago. The 'LAST EXECUTION RESULT' is 'No results'.
- Montreal Deploy:** Shows 'LAST INPUT' as 'Stage: Build Stage / Job: B...'. The 'JOBS' section shows a 'Deploy' job that passed 9 days ago. The 'LAST EXECUTION RESULT' is 'No results'.

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 All CATEGORY

kubernetes

Search

Clusters

Name ▾
Montreal
Toronto

Cluster

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

Namespace default ▾

Overview

Workloads

Cron Jobs

Daemon Sets

CPU usage

Time	CPU (cores)
11:39	1.6
11:40	1.6
11:43	1.7
11:46	1.6
11:50	1.6
11:51	1.6

Namespaces

Name ▾	Labels
✔ robank	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

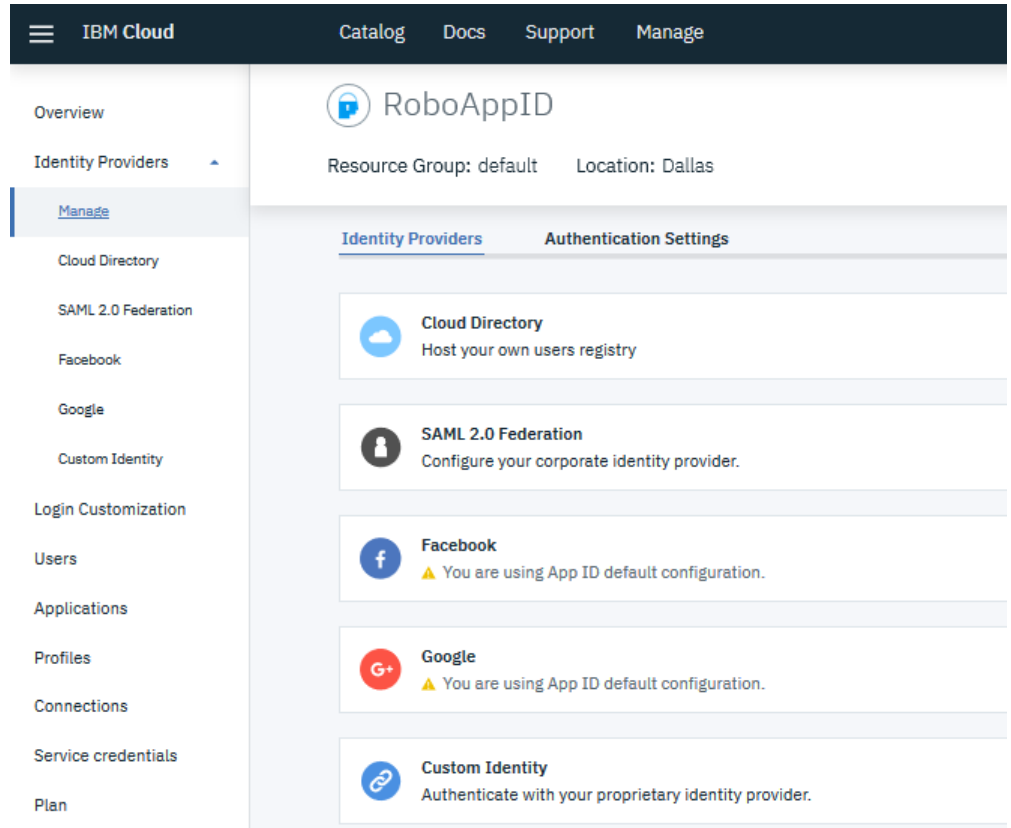
Through IBM's CloudFlare partnership, IBM offers a global load balancing service providing up to five-9's availability to cloud-based workloads.

The screenshot shows the IBM Cloud console interface. The top navigation bar includes 'IBM Cloud', 'Catalog', 'Docs', 'Support', and 'Manage'. The left sidebar contains a menu with items: 'Getting started', 'Overview', 'Metrics', 'Security', 'Reliability', 'Global Load Balancers' (highlighted), 'DNS', 'Performance', and 'Plan'. The main content area is titled 'Dashboard / RoboInternetServices' and shows 'Resource Group: default' and 'Location: Global'. The 'Global Load Balancers' section includes a description: '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.' Below this, there are tabs for 'Manage' and 'Health Check Events'. The 'Load Balancers' section contains a table with the following data:

	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.



The screenshot displays the IBM Cloud management interface for the RoboAppID service. The top navigation bar includes 'IBM Cloud', 'Catalog', 'Docs', 'Support', and 'Manage'. The left sidebar lists navigation options: Overview, Identity Providers (with a dropdown arrow), Manage (highlighted), Cloud Directory, SAML 2.0 Federation, Facebook, Google, Custom Identity, Login Customization, Users, Applications, Profiles, Connections, Service credentials, and Plan. The main content area shows the 'RoboAppID' configuration page for 'Resource Group: default' and 'Location: Dallas'. It features two tabs: 'Identity Providers' (selected) and 'Authentication Settings'. Under 'Identity Providers', four providers are listed:

- Cloud Directory**: Host your own users registry.
- SAML 2.0 Federation**: Configure your corporate identity provider.
- Facebook**: You are using App ID default configuration.
- Google**: You are using App ID default configuration.
- Custom Identity**: Authenticate with your proprietary identity provider.

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.

The screenshot displays the IBM Cloud Event Management interface. At the top, the header reads "IBM Cloud Event Management". Below this, the word "Incidents" is visible. The main heading is "Priority 1 : Incident #0000-4000". There are three tabs: "Resolution view", "Events" (which is selected), and "Timeline". Below the tabs, it says "1 of 1 events". A table lists the event details:

Sev	State	Resource type	First occurrence	Summary
1	Open	Database	Nov 3, 2018 12:30:24 PM E...	Sat, 03 Nov 2018 16:30:23 GM

Below the table, there are two tabs: "Event info" (selected) and "Suggested runbooks (1)". Under "Event info", there are four columns of details:

State	Location	Host name	Status
Open	-	-	-
ID	IP address	Sender type	Sender name
#apeb-j48h	-	Inbound Webhook	Inbound Webhook

At the bottom of the event details, there is a button labeled "See more info" with a right-pointing arrow.

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	▼	> icp31-1.robobob.ca	Virtual Server	Toronto 1	
Storage	▼	> icp31-w1.robobob.ca	Virtual Server	Toronto 1	
Network	▼	> icp31-w2.robobob.ca	Virtual Server	Toronto 1	
Security	▼	> icp31-w3.robobob.ca	Virtual Server	Toronto 1	
Services	▼	> icp31-w4.robobob.ca	Virtual Server	Toronto 1	
		> icp4.robobob.ca	Virtual Server	Toronto 1	
		> icp5.robobob.ca	Virtual Server	Toronto 1	
		> kube-mon01-cr751ef7ab97c74631944355e6f	Virtual Server	Montreal 1	
		> kube-mon01-cr751ef7ab97c74631944355e6f	Virtual Server	Montreal 1	
		> kube-mon01-cr751ef7ab97c74631944355e6f	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>

A word cloud featuring the phrase 'Thank You' in multiple languages and various fonts and colors. The words are arranged in a circular pattern. The languages include: English (Thank You, Thank U, Danke, Tack, Merci, Diolch, Grazie, Obrigado, Terima Kasih), Greek (Ευχαριστώ), Hebrew (תודה), Spanish (Gracias), Russian (Спасибо), and Arabic (شكر). The colors used are primarily red, blue, and purple. The fonts vary in size and style, with some being bold and others more elegant.