IBM i:
JOURNEY TO THE CENTER OF THE CLOUD

Prepared by Matt Shannon, Sr. Solutions Specialist
and Jeffrey Whicker, Sr. Solutions Specialist
Sirius Computer Solutions, Inc.

July 2017

Contents

Executive Summary

The headlines we see in email blasts, newsletters and trade journals rarely tout the latest advances in RPG and COBOL language features. Cloud services and APIs are the buzzwords that are currently grabbing the attention of business and IT leaders. These are also touted as up-and-coming skills that every new programmer should be learning if they want to stay busy for the next decade. But what is the code underlying these APIs and services?

Contrary to what you might have read in the latest trade journal article, cloud computing and APIs do not rely on the latest framework, server refresh or programming language. The goal is to identify reusable packages of business functionality to expose as services, and then to compose these services to better meet the needs of the business. The cloud provides options for where to run the services and where to procure new services, but it does not prescribe the platform or logic that provides the service.

Line-of-business managers are seeing the benefits of available API services, and going outside traditional IT procurement lines to get their own services and answers. Current trends show rapid adoption of third-party REST APIs, microservices, cloud workloads, cloud analytics and storage. How can the IBM® i shop remain an important cog in the wheel of innovation rather than devolving into being solely administrators of a system of records? Over the years, many tools have been made available to the platform to increase agility and connectivity. These include IBM Rational® HATS, HOD, Web services, PHP (Zend®), IceBreak, RPG IV, free-form RPG and others. During the SOA boom a few years back, we showed IBM i clients methods to powerfully leverage the platform in this arena. Are you using those methods to engage developers with both internal and external customers?

As developers who use the IBM i platform, we often see company initiatives for modernization that are not taking advantage of the IBM i platform's capabilities, which match many cloud capabilities. IBM i is ready now to be a part of the emerging world of connected computing. It is capable of breaking out of its insulated past. To attract the next generation of talented programmers, the perception of IBM i as only a legacy platform has to be changed.

To prove this, we connected an IoT device to a Power Systems™ server running IBM i by routing sensor data through a messaging system in the IBM Bluemix® Cloud, and presented the results at IBM InterConnect 2017 in Las Vegas.

History

IBM Power Systems servers running the IBM i operating system has a rich history of compatibility with earlier versions, ever since their predecessor—the AS/400—was architected to maintain full backward compatibility with software originally written on the System/36 and System/38 midrange platforms.

As the years have passed, each major milestone in the operating system and hardware has added new and improved technologies to the platform. These advances in technology have kept IBM Power platforms competitive and completely modern. Though the technological advances have been significant, IBM has maintained a commitment to preserving the investment that companies made in software development for earlier generations. In fact, it is still possible to compile and execute code written in the mid-1970s on a current IBM Power Systems server. Try running an MS-DOS® v2.0 program on a Windows® 10 machine, or compiling a GW-BASIC source file!

This commitment to preserving the investment in software development is impressive, but it has brought with it some unexpected challenges.

Some IT shops have simply failed to embrace the new technologies, and many have limited their efforts to maintaining existing software. For instance, there was not an absolute need to learn the modern Rational Developer for i IDE for coding RPG or COBOL programs, so many shops were (or are) slow to take advantage of that powerful technology.

A failure to embrace IBM Db2® as a full-featured SQL DBMS is also symptomatic. Many shops have continued to use DDS to create the database tables used in their environment. Some shops still have not chosen to take advantage of relational constraints, stored procedures, and triggers.

All of this and more can lead to the perception that the IBM Power Systems platform itself is outdated, and has already led many management teams to declare their intention to replace their IBM i servers with a different platform. This perception also makes it difficult to attract young developers.

Steps to modernization, or “What does a modern IBM i shop look like?”

Modernizing our approach to the IBM i platform is important for the continued viability of the platform.

The first step is to wean ourselves off the Program Development Menu, or PDM. Rational Developer for i is much more robust, provides a better coding and debugging experience, and can significantly reduce development time. Thus, it will aid in improving the perception of the platform.

It is important to use modular programming within RPGLE. Most shops have long since taken this important step, but some shops hold back because of the vast amount of code being maintained that is not modular. Other shops have moved to RPG IV, but have not bothered to consider how to use modular service programs or binding directories. Learning these techniques will not only improve the core business software being maintained; it will also teach concepts needed to add modern languages such as Java, .NET and JavaScript to your skill sets.

IBM has provided IBM Toolbox for Java on the Power System for i platform at
https://www.ibm.com/support/knowledgecenter/ssw_ibm_i_71/rzahh/page1.htm

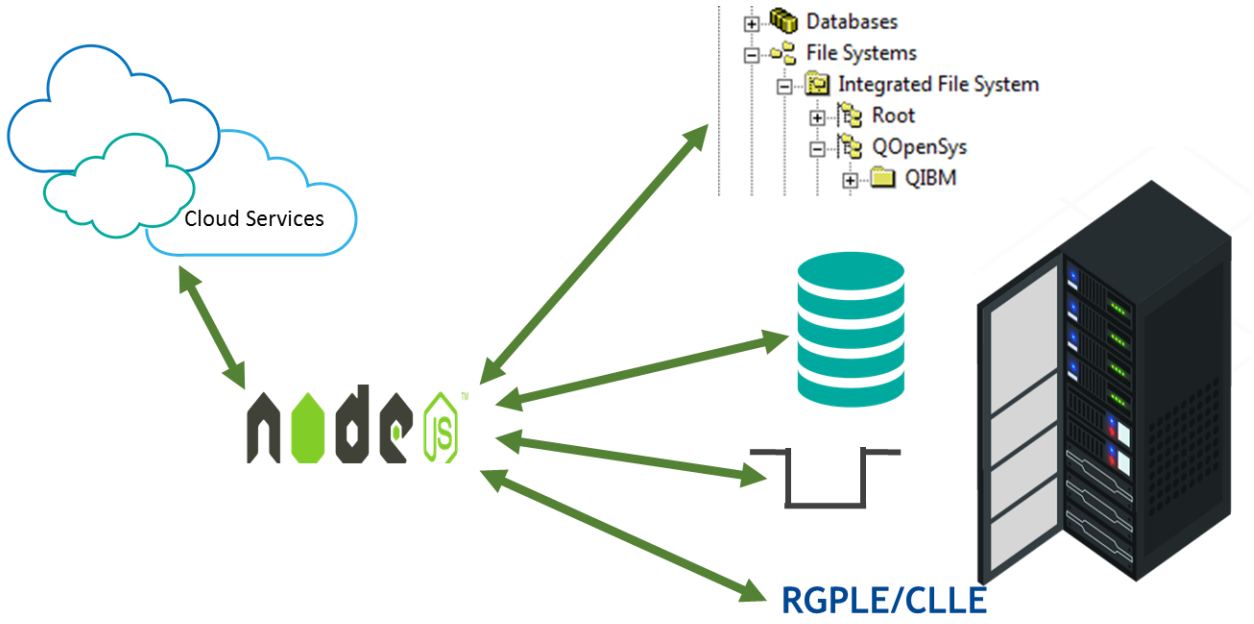
Learning and using this toolkit is one route to the IBM cloud. The strong typing and object orientation of Java make it the most logical language to learn for Power Systems developers, provided they have taken the time to learn, develop and implement programs in the modular Integrated Language Environment (ILE). While ILE is not object-oriented, it is a close enough approximation that the concepts are similar.

JavaScript is available on the Power System for i in the form of Node.js at
https://www.ibm.com/developerworks/ibmi/library/i-native-js-app-ibmi-with-nodejs/index.html

JavaScript is a bit more difficult to learn for Power Systems developers, but the payoff is excellent. The IBM cloud will run Node.js programs natively. Communicating between Node.js applications is relatively easy, so once JavaScript/Node.js is understood there is a relatively low barrier of entry for IBM i developers to integrate with the cloud.

We can turn stored procedures and CL PGM into services. These are the same types of services that get exposed as APIs. Many of the tools are already available if we expand our vision to providing the information to new audiences.

Cloud integration: The new programming paradigm



As part of a proof of concept for a presentation, we connected an IoT device to a Power Systems server running IBM i by routing the sensor data through a messaging system in the IBM Bluemix Cloud. The impetus for this was twofold: first, as a way to throttle a large amount of sensor data to a manageable size; and second, to learn how to integrate IBM i with a Cloud Message provider.

There is a variety of ways to connect to Cloud APIs, but we chose Node.js as it has momentum and community support. We needed to set up Node.js on both the sensor side and on IBM i. Node.js on IBM i is available as a PTF. After doing this, connecting to the Cloud Messaging Application was not difficult. In addition to proving the concept, we were struck by many other capabilities that were now available to us. The same messages we were sending through to the server could be saved to cloud storage for further analytics, decoupling the analytics from the real-time processing, but using the same live data. It would also be possible to have some computational capabilities on the cloud to further filter the data coming in. That way, the processing would be more efficient since it would only need to handle sensor data that was out of an expected range, rather than accepting all data. Many incremental improvement ideas such as this cropped up as we considered the use case, and this was a fairly simple application.

Conclusion

There are many online tutorials and guides regarding how to connect to cloud services, but they won't help you figure out the setup and run the examples. Sirius and IBM have experts who can give you a head start on getting connected to cloud services. Sirius' Cloud Innovation Studio workshops help you create and play with services tailored to your own organization, so you can get a better sense for what will work for your unique needs. Call us today and let us customize a plan to help you jumpstart your development efforts.

To learn more, speak to your Sirius representative, visit the Software Consulting Services page at siriuscom.com or contact authors Matt Shannon at matt.shannon@siriuscom.com or Jeff Whicker at jeff.whicker@siriuscom.com.