DB2 FOR Z/OS DATABASE CHANGE MANAGEMENT AND DEVOPS

Bringing DevOps to Db2 on the mainframe

By Craig S. Mullins

Mullins Consulting, Inc.
15 Coventry Court
Sugar Land, TX 77479
www.mullinsconsulting.com

Db2 for z/OS Database Change Management and DevOps

TABLE OF CONTENTS

The Promise of DevOps	1
The Adoption of DevOps	4
The Mainframe and DevOps	5
The Database and DevOps	6
What About Db2 for z/OS?	9
Questions to Consider	10
BMC AMI for Db2 for z/OS	11
About the Author	12
About BMC	13

As mainframe development teams begin to rely on DevOps practices more extensively, the need arises to incorporate Db2 for z/OS database changes.

This white paper provides an overview of DevOps, compares application changes practices to database change practices, provides a high-level overview of the types of changes Db2 for z/OS database changes, and discusses the requirements for supporting such changes.

The Promise of DevOps

Organizations of all types are being impacted by the transformation to the modern digital economy. This digital transformation, driven by the 24/7 expectations of users and customers to access data and apps at any time from any device requires the ability to rapidly change technology and software. Indeed, IDC predicts that by 2021, at least 50% of global GDP will be digitized.[1] This means that today's businesses have to deliver and improve the services and software used by their customers faster than they ever have before.

To be able to survive and thrive in the new digital economy has caused organizations to adopt new and faster methods of developing, testing and delivering application software. Moving from a waterfall software development methodology to an agile methodology is one way that organizations are speeding the time-to-delivery of their software development.

An agile development methodology involves the organization of the team and goals such that software development is iterative, incremental, and evolutionary. Using an agile approach, development and testing activities are undertaken concurrently, unlike the traditional Waterfall development model. Large software projects are broken into pieces such that immediate value can be delivered quickly, in smaller pieces, instead of waiting for a monolithic product to be completed. With continuous testing and integration, the smaller software pieces can be integrated into a larger, final deliverable.

[1] Thomson, Jennifer, IDC Technology Spotlight, The Journey to Enterprise-Scale DevOps: Becoming DevOps Determined, March 2019, IDC Corporation

Another key methodology being used to speed up the delivery of software is the DevOps approach, which results in small and frequent code changes. Its name is an amalgamation of Development and Operations. As such, DevOps relies on agile development, coupled with a collaborative approach between development and operations personnel during all stages of the application development lifecycle. Such an approach can significantly reduce the lead time for changes, lower the rate of failure, and reduce the mean time to recovery when errors are encountered. With such benefits that can accrue, it is no wonder that DevOps and continuous delivery are gaining in popularity.

Instead of long software development projects that may not deliver value for months, or perhaps even years (such as are common using the Waterfall development methodology) an agile DevOps approach delivers value quickly, and then incrementally over time. DevOps enables the continuous delivery of new functionality demanded by customers in the digital economy.

Succeeding with DevOps, however, requires a cultural shift in which all groups within IT work in collaboration with one another, and where management endorses and cultivates this cultural change. Because DevOps relies upon incremental development and rapid software delivery, your IT department can only thrive if there is a culture of account-ability, collaboration, and team responsibility for desired business outcomes. Creating such a culture is not easy. The goal of DevOps is a constantly repeating cycle of continuous development, continuous integration and continuous deployment, with all of the activities and disciplines required for each of those activities.

This is typically depicted graphically as the infinity symbol as seen in Figure 1.

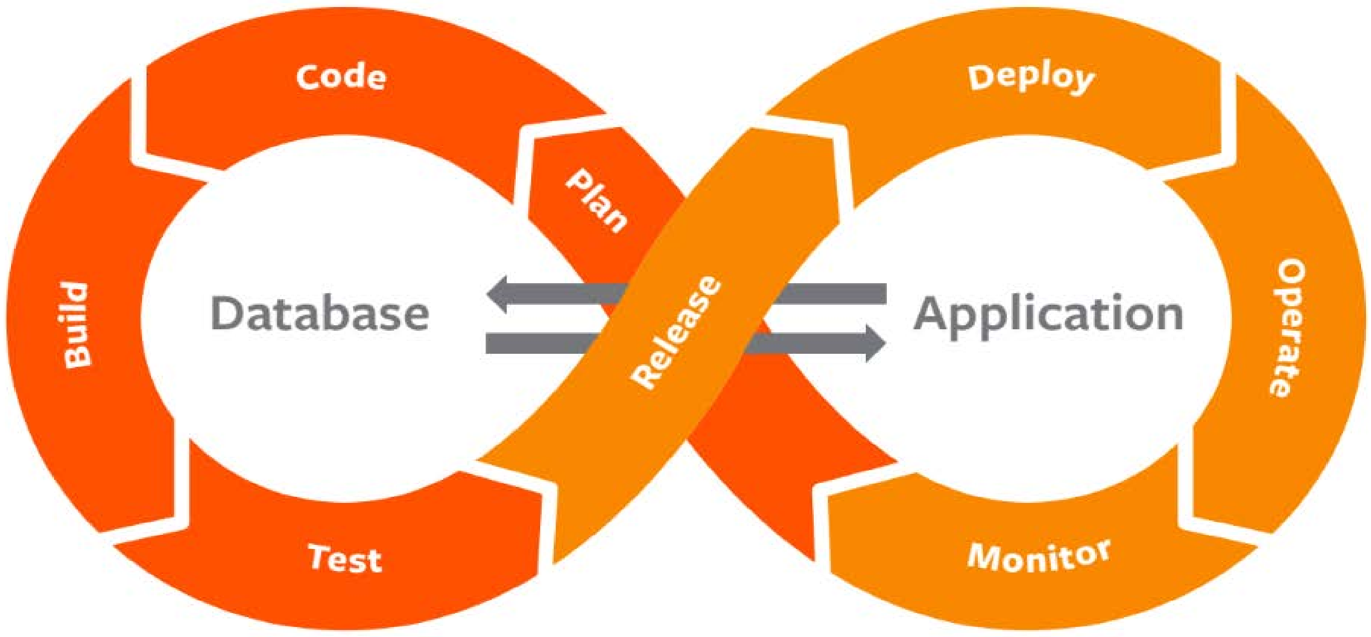


Figure 1. The DevOps Cycle

To achieve such a high level of continuous delivery also requires in-depth automation of the entire development lifecycle. This is often referred to as the DevOps toolchain.

Note, however, that this particular iteration of the DevOps infinity graphic calls out the participation of both the application and the database. This is an important, though often lacking, detail that should be stressed when adopting DevOps practices.

The Adoption of DevOps

Adopting a DevOps culture and process can deliver significant advantages for organizations. According to one study, high-performing IT organizations experience 60 times fewer failures and recover from failure 168 times faster than their lower-performing peers. They also deploy 30 times more frequently with 200 times shorter lead times.[2]

And DevOps adoption is on the rise, with 47 percent using DevOps for at least some of their projects, and only 27 percent not using or planning to use a DevOps approach.[3] And for those that have adopted DevOps, automation is a key success factor, with the highest performers automating significantly more of their configuration management, testing, deployments and change approval processes than other teams.[4]

Another aspect of DevOps is improving the time to deliver software. Traditional IT operations (typically in siloed teams) spend 5 percent more time each week deploying changes than those participating in DevOps teams.[5]

[2] 2015 State of DevOps Report, IT Evolution and Puppet Labs
[3] King, Dr. Elliot, The Current State and Adoption of DevOps, Unisphere Research, September 2016
[4] 2017 State of DevOps Report, DevOps Research & Assessment and Puppet Labs
[5] IT Ops and DevOps Productivity Report 2013: Tools, Methodologies and People, Revel Labs

The Mainframe and DevOps

The adoption of DevOps has been much slower within mainframe development teams than for distributed and cloud application development. There are several reasons for the slower uptake, including:

- The traditional waterfall development methodology has been deployed by most mainframe software developers for multiple decades and DevOps is closely aligned with an agile approach, which differs significantly from waterfall.
- Mainframes are used by large organizations running mission critical workloads and are averse to change of any kind.
- A culture of slow change is integrated into many mainframe shops, with specific change control windows, in-depth sign-off procedures, and 50 years of doing things a *specific way*.
- For z/OS, the deployment processes are highly structured requiring all the changes to follow a common process. In most cases, source code management tools are used that drive the promotion and deployment of mainframe code for all applications.[6]
- The mainframe change control process can be highly custom-ized, perhaps in different ways, at mainframe shops. It can be challenging to integrate the customized process into a new, DevOps approach.
- Specialists with both mainframe and DevOps skills are scarce and often become a bottleneck to deliver the improvements and transformation objectives.[7]

Notwithstanding all of these barriers to acceptance of DevOps on the mainframe, it must be stated that mainframe development can, and in many cases already do successfully utilize a DevOps approach. Technically speaking, the mainframe is just another platform and there is nothing inherent in its design or usage that obviates the ability to participate in a DevOps approach to application development and delivery.

[6] Radcliffe, Rosalind, “DevOps for the mainframe,” The Invisible Thread: Driving software & systems innovation, https://www.ibm.com/developerworks/community/blogs/invisiblethread/entry/devops_for_the_mainframe
[7] Enabling DevOps on Mainframe, Sandhata, https://www.sandhata.com/mainframe-devops/

There is, however, a significant reason for the mainframe to take advantage of the benefits of DevOps. Mainframes continue to drive a significant portion of mission critical workload. IBM indicates that 80 percent of the world's structured enterprise data resides on the mainframe and that more than 55 percent of all enterprise application transactions run on the mainframe.[8]

Clearly the mainframe continues to be an important platform for hosting and developing critical business applications. Adopting a DevOps approach for your mainframe application and database development can speed time to ROI, reduce cost, and improve quality... all of which are desirable effects!

The Database and DevOps

Integrating database change into the application delivery lifecycle can be a stumbling block on the road to DevOps success. Development teams focus on application code, as they should, and typically view database structure changes as ancillary to their coding efforts. In most application development projects it is not the programmer's responsibility to administer the database and modify database structures. But applications rely on the database being designed, implemented, and changed in accordance with the needs of the business and the code.

So, developers have automated their SDLC tool chain to speed up the delivery of applications. This is the "Dev" portion of DevOps. But the requisite automation and tooling has not been implement-ed to speed up the delivery of database changes. This is part of the "Ops" portion of DevOps. There are several reasons for this dichotomy.

The first consideration to keep in mind is that developers have been migrating to use agile development techniques since The Manifesto for Agile Software Development[9] was created, almost two decades ago. During this timeframe, development processes have been created using an integrated automated toolchain for the SDLC.[10] This toolchain delivers automation, integration, self-service, and collaboration for continuous software development and delivery.

[8] IBM Corporation, Enterprise DevOps, https://www.ibm.com/uk-en/it-infrastructure/z/capabilities/enterprise-devops
[9] The Manifesto for Agile Software Development, https://agilemanifesto.org/
[10] What is a DevOps Toolchain?, BMC DevOps Blog, https://www.bmc.com/blogs/devops-toolchain/

The situation is different for DBAs. Although many rely tools on automated tools for some of their activities, there are not many DBA solutions that integrates into the development toolchain.

The DevOps workflow, which is automated for code changes, frequently is not automated for database changes, at least not in the same way, using the toolchain and workflow that DevOps developers have become accustomed to using. Sure, the DBA may have a tool that helps to automate database change, such as BMC Change Manager for Db2 for z/OS, and such tools will help to speed up the database change process. Such tools can be used to create baselines of the database schema, automate the creation of scripts and JCL to perform all of the changes necessary, and even to backout changes if needed to application or database issues.

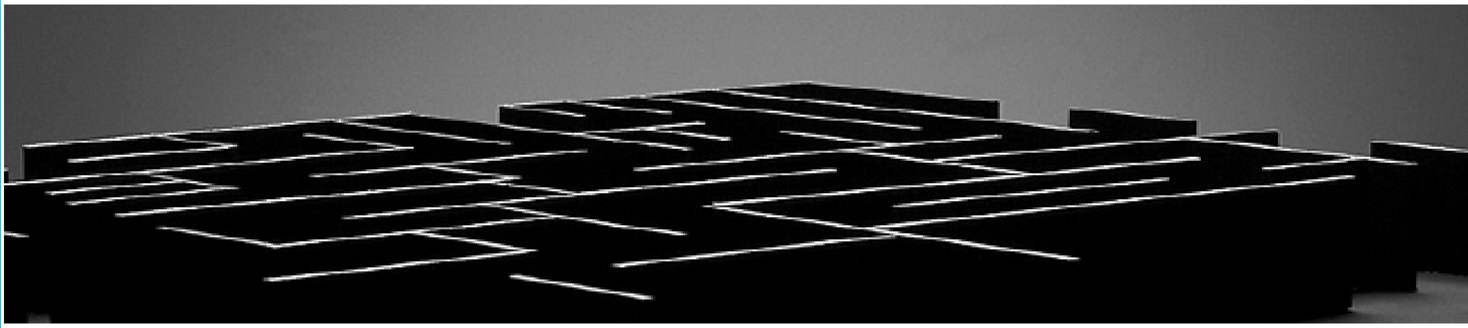
Standalone database tools are great, but database change management is still mostly a separate activity from application change management. The developers have to request the DBA team to make changes using a mechanism not within the DevOps toolchain. And then the DBA evaluates the request and manually processes it through their procedures, methods, and tools. This takes time and slows down the DevOps-enabled development teams. Only once the database change request has been analyzed, approved and implemented can the development team proceed apace with their now standard continuous development, continuous integration, and continuous delivery activities.

So, what happens? The automated application development workflow halts as manual processes kick in when database changes are required to be made in conjunction with application code migration.

An additional consideration that is not often factored into the overall change management equation is that application changes and database changes are inherently different and need to be managed using different techniques. Consider what must happen when moving an application change from the test environment to production. Furthermore, let’s assume that the application relies on a series of database changes.

When an application program changes, the code is compiled and the load module is migrated from test to production. The old load module is saved for posterity in case the change needs to be backed out, but the change is a wholesale replacement of the executable code.

Database changes are different. The database is an entire configuration in each environment and changes get migrated. There is no wholesale replacement of the database structures. DDL commands are issued to ALTER, DROP, and CREATE the changes to the database structures as needed.



Some database changes can be quite complex. Many developers fail to understand the intricacies involved in database change management. Oftentimes their exposure to DDL is simply adding a column to a table or building a quick index, if they have seen any DDL at all. And yes, these types of changes can be implemented very quickly. Whether or not they always should be implemented that quickly is another discussion because adding a column might not be to the correct table or might break down normalization. Furthermore, building another index might not be the best approach if an existing index could be used by making a simple adjustment to the SQL

Additionally, some database changes are very invasive and must be implemented by dropping and re-created database objects. One quick example is trying to add a column to the middle of an existing row. To do so, the table must be dropped and recreated with the new column in the middle. But what about the data? When the table is dropped the data is deleted, unless the DBA was wise enough to first unload the data. What about the indexes on the table? Well, they were dropped when the table was dropped, so unless the DBA knows this and recreated the indexes too, performance will suffer. The same is true for database security – when the table was dropped all security for the table was also dropped. And this is but one simple example. Other types of database change can be just as invasive.

This is why tools like BMC Change Manager exist: to automate the entire database change process and make sure that nothing is lost when anything is changed. But heretofore there has been no integration into the DevOps toolchain.

Another way that the database is different from the application in terms of change management is that that each database environment (test vs. prod, for example) might be different, however so slightly. Differences can "drift" in, such as optimization changes, emergency changes, and other differences. And there are things that are planned differences, for example, test tables contain less data, production tables may have a different owner than test, or maybe table space names are different.

It should be obvious then that the same exact procedures cannot be used to implement application code changes and database structural changes. Nonetheless, it is just as clear that both types of changes must be coordinated and conducted in sync for applications to function properly.

What About Db2 for z/OS?

There is nothing inherent to Db2 for z/OS that makes it different than other DBMS products in terms of its ability to participate as a partner in a DevOps environment. Db2 for z/OS is a relational, SQL database management system, albeit one designed for and integrated into the mainframe infrastructure. But as we discussed before, the mainframe is well-suited to benefit from a DevOps approach, and the same is true for Db2 for z/OS. All that is needed is tooling that understands both Db2 for z/OS and the DevOps methodology and toolchain.

From the perspective of database changes on Db2 for z/OS, DBAs need the ability to modify all the database objects supported by Db2 for z/OS. There are two dozen database objects and structures that can be created and modified by DDL, as shown in the table, below.

Database Structure Related Objects	Program, Security and Other Objects
• DATABASE	• FUNCTION (UDF)
• TABLESPACE	• PROCEDURE
• STOGROUP (Storage Group)	• PACKAGE
• TABLE …	• PLAN
• MQT	• TRUSTED CONTEXT
• GLOBAL TEMPORARY TABLE	• ROLE
• INDEX	• MASK
• ALIAS	• PERMISSION
• SYNONYM	• TRANSFER OWNERSHIP
• VIEW	• LABEL
• SEQUENCE	
• TRIGGER	
• TYPE (UDT)	
• VARIABLE	

In addition to supporting changes for all of the supported types of database structures, DBAs need to ensure that changes are made in the most efficient and effective manner. This is important because changes can be made to Db2 for z/OS objects three different ways.

1. Simple changes can be implemented immediately without requiring intervening actions.
2. Pending changes require a bit more work to implement by running a REORG.
3. Complex changes require dropping and re-creating the object to implement (as well as a series of intervening actions to make sure nothing is lost)

Of course, there are prerequisites, requirements, and nuances to each of these three mechanisms that are beyond the scope of this white paper. Suffice to say, DBAs work with tools to minimize the amount of effort required to choose the correct method, as well as to implement that method as simply, and with as little downtime as possible.

Questions to Consider

So how do you manage change? Do you integrate database schema changes with application changes? If so, how? If not, how do you coordinate things to keep the application synchronized with the databases?

Have you adopted a DevOps methodology for the continuous delivery of application changes in your organizations? If so, how do you manage database changes in conjunction with your application development lifecycle?

If you want to bring your DBA practices into the modern era, you need to automate and integrate database application testing and change management with application development testing and change management as part of an integrated DevOps toolchain.

If you do not have the tools in place to automate DBA procedures, implementing DevOps for database applications will be difficult to impossible.

BMC AMI DevOps for Db2 for z/OS

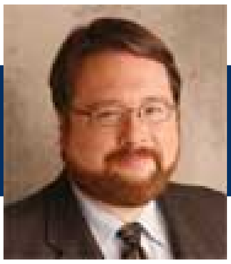
If you are looking to implement DevOps for Db2 for z/OS, BMC has a new solution integrated with a long-time DBA stalwart: BMC AMI DevOps for Db2 integrates change management into your DevOps toolchain.

Because BMC AMI DevOps for Db2 integrates with your application development orchestration tools your developers can seamlessly utilize it to automatically capture database changes and communicate them to the DBA, while enforcing DevOps best practices.

With BMC AMI for DevOps you can support business agility while maintaining database continuity, delivering the benefits of DevOps for your mainframe application development teams. You can improve the quality and efficiency of Db2 schema changes because it is built on the field-tested, reliable capabilities of BMC Change Manager. All while enabling the speed and self-service capabilities required by modern development practices.

Instead of experiencing slowdowns and bottlenecks when it comes to implementing database changes, you can rely on BMC AMI DevOps to integrate Db2 schema changes into your agile development process, enable application developers to submit schema changes to the database without waiting on the DBA, and thereby speed up the process to deploy schema changes on Db2 for z/OS.

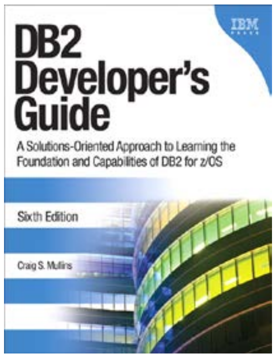
More details available at:
https://www.bmc.com/it-solutions/ami-devops.html

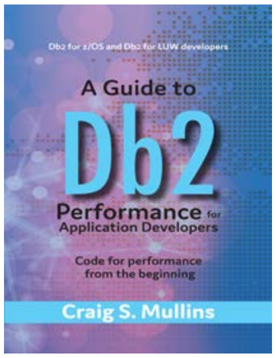


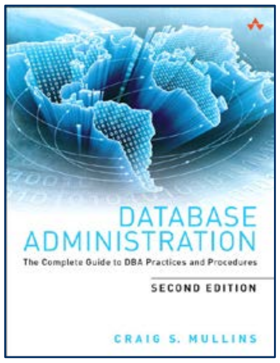
About the Author

Craig S. Mullins is a data management strategist, researcher, and consultant. He is president and principal consultant of Mullins Consulting, Inc. Craig has been named by IBM as a Db2 Gold Consultant and an IBM Champion for Analytics. He was also named as one of the Top 200 Thought Leaders in Big Data & Analytics by Analytics Week magazine.

Craig has over three decades of experience in all facets of database systems management and development. He has worked with Db2 since V1. You may know Craig's popular books including:


DB2 Developer's Guide
A Solutions-Oriented Approach to Learning the Foundation and Capabilities of DB2 for z/OS
Sixth Edition
Craig S. Mullins


Db2 for z/OS and Db2 for LUW developers
A Guide to Db2 Performance for Application Developers
Code for performance from the beginning
Craig S. Mullins



Mullins Consulting, Inc.
15 Coventry Court
Sugar Land, TX 77479
http://www.mullinsconsulting.com

This white paper was produced by Mullins Consulting, Inc. for BMC Software, Inc.

About BMC

BMC helps customers run and reinvent their businesses with open, scalable, and modular solutions to complex IT problems. Bringing both unmatched experience in optimization and limitless passion for innovation to technologies from mainframe to mobile to cloud and beyond, BMC helps more than 10,000 customers worldwide reinvent, grow, and build for the future success of their enterprises.

http://www.bmc.com

Copyright and Trademark