Mainframe-inclusive DevOps Toolchains

ACHIEVING CONTINUOUS DELIVERY IN LARGE MULTI-PLATFORM ENTERPRISES

Your success increasingly depends on **relentless improvement of your company's digital capabilities**—which, in turn, depends on consistently getting great code into production ASAP.

To do this, companies are building **DevOps toolchains that use automation** to facilitate speed, quality, productivity and good governance across a Continuous Delivery lifecycle.

It is imperative for larger enterprises to **include the mainframe in these toolchains,** since mainframe applications and data are among their most strategic digital assets.

The optimal architecture for mainframe-inclusiveness **harmonizes mainframe platform/task specificity** with good integration into the broader DevOps toolchain environment.

Inclusive toolchains deliver major competitive advantage by **enabling much more frequent drops of quality mainframe code,** while making the platform much more accessible to DevOps teams.

DEVOPS BENEFITS

- 57% increased customer conversion or satisfaction
- 57% reduced IT infrastructure spend
- 49% reduction in application downtime or failure rates
- 46% increase in customer engagement
- 46% increase in sales
- 32% increase in employee engagement

Source: Vanson Bourne on behalf of Rackspace

DEVOPS TOOLCHAINS AND CONTINUOUS DELIVERY

Software plays a central role in the performance of your business. Great mobile and web apps win you customers and keep them loyal. Well-crafted dashboards and BI tools dramatically improve the decisions your people make every day. Software streamlines your supply chain and mitigates your compliance risks. In fact, your business can do just about everything better if it does it with better software.

But the bar for digital excellence keeps rising as customer expectations and market demands continue to escalate. So digital excellence is something you have to relentlessly strive to sustain.

The best companies sustain their digital excellence through a combination of Agile culture, DevOps best practices and smart hiring/coaching of developer talent. And what they achieve to one degree or another is Continuous Delivery—increasingly frequent drops of code that work as required in production and tangibly improve business performance.

These rapid release cycles accelerate the critical learning enterprises must do to quickly discover ideas that matter to the customer—and just as quickly turn them into living digital value.

Continuous Delivery is facilitated by DevOps toolchains. These toolchains link developers, test/QA teams and operations staffs as they perform software lifecycle tasks such as requirements management, coding, regression testing, release packaging and deployment. They include DevOps solutions associated with familiar names such as Atlassian, Chef, Docker, Slack, SonarSource, Splunk and XebiaLabs.

DevOps toolchains can be evaluated by five criteria:

✔ **Functional richness.** Each tool in a toolchain has its own capabilities. Those capabilities include automation that accelerates manual processes and enforces process rules, visualizations that make it easy for DevOps staff to find and fix problems, and management features that enable process governance. The richer those automation, visualization and management capabilities, the better the overall toolchain.

Compuware

✔ **Completeness.** The theory of constraints suggests that a DevOps toolchain is only as useful as its weakest link. Any gap in toolchain coverage—whether that coverage gap is of a task or a platform—is thus an absolute constraint on the end-to-end Continuous Delivery process.

✔ **Integration.** Toolchains don't just facilitate Continuous Delivery by supporting each step in the process. They also facilitate it by effectively linking those steps together. In fact, those inter-step handoffs can be particularly susceptible to inefficiencies and errors. So tool-to-tool integration is a critical factor in toolchain value.

✔ **Ease of use.** DevOps tools aren't of much value if it takes too much time and effort to learn how to use all their features—especially since DevOps staffs often have to quickly become competent on multiple tools. Ease of use is therefore another factor in toolchain effectiveness.

✔ **Adaptability.** DevOps teams have to work on different types of projects involving different types of data, codebases and platforms—each of which is constantly undergoing its own independent evolution. A good toolchain must therefore be capable of adapting to changes in technology, in application architectures and in collaboration partners.

By building and refining toolchains that meet these criteria, companies can deliver more code that more precisely fulfills the needs of the business with fewer defects at a faster cadence and at lower cost with greater reliability.

A well-engineered DevOps toolchain is thus not merely a matter of IT convenience or geek pride. It is a fundamental necessity for any company hoping to successfully compete in an increasingly digital-centric marketplace.

THE MAINFRAME'S SPECIAL ROLE

DevOps and Continuous Delivery best practices have primarily been applied to distributed, web, mobile and cloud platforms. At large enterprises, however, the mainframe is also an extremely important platform. In fact, most development at large enterprises—including development of mobile, web and cloud services—depends on the mainframe as a back-end server.

There are two primary reasons for this:

- **Core applications.** Mainframe applications are typically a large enterprise's most crucial and highly evolved systems. Their business logic has been rigorously honed and refined over multiple decades to fulfill the complex requirements of core business processes and mission-critical transaction processing. So any application that extends a core process or enables execution of a transaction must interface with mainframe code.

- **Core databases**. Mainframes typically host a large enterprise's most complete and up-to-date information. The associated mainframe databases have also been rigorously honed and refined over multiple decades to fulfill the complex requirements of the business. Their extended evolution can be particularly important in financial services, insurance, retail, government and other categories where customer relationships and compliance requirements extend over multiple decades.

In addition to hosting the enterprise's core applications and core databases, the mainframe itself is inherently appealing because of its reliability, security, scalability and performance economics. These attributes make it a compelling platform for computing workloads that are mission-critical, require high transaction rates, are especially response-time sensitive, and/or are subject to unexpected demand spikes. The mainframe is also the preferred platform for working with massive mainframe datasets—since it makes sense to operate on those datasets where they are, rather than first performing unacceptably slow and expensive dataset extract, transform and load (ETL) operations to work with the data elsewhere.

Despite the crucial importance of mainframe applications and data, however, the mainframe has almost universally been excluded from DevOps/Continuous Delivery toolchains. This exclusion has severely hampered enterprise agility—especially when IT organizations have found themselves limited to just a small handful of mainframe code drops annually.

There are three main reasons that the mainframe has historically been excluded from the DevOps mainstream:

1. **Lack of appropriate tools.** Most mainframe tools vendors have remained stuck in a very outdated and siloed view of the platform—and have therefore failed to develop and deliver tools that support contemporary concepts such as Agile and DevOps. Nor have they provided large enterprises with mainframe tools that fit well into multi-platform, multi-vendor preferred DevOps toolchains.

2. **The myth of inherent platform non-agility.** Because large enterprises haven't done Agile on the mainframe, it's easy to lapse into the misconception that they can't.

This is false. Code is code. There's nothing about IBM z/OS—or COBOL or PL/I or Assembler for that matter—that makes it inherently impossible to more quickly execute smaller, more frequent drops of code that align tightly with new business requirements. In fact, it can and is being done today.

3. **Cultural/generational issues.** Agile, DevOps and Continuous Delivery are as much about culture as they are about enabling technologies. Mainframe development teams have typically worked under a waterfall model with extremely infrequent (by today's standards) releases for decades. Changing these personal work habits and culture isn't easy. Developers working on other platforms, on the other hand, tend to be "native Agile."

Large enterprises cannot be truly digitally agile if their mainframes are not agile.

The continued exclusion of mainframe applications and data from the enterprise DevOps toolchain, however, is no longer necessary or desirable. Large enterprises cannot be truly digitally agile if their mainframes are not agile. And this failure to become fully agile will ultimately cause any large enterprise to fail—regardless of how invulnerable it may seem to those in denial about the realities of competition in digital markets.

Large enterprises that want to successfully compete in digital markets must therefore craft DevOps toolchains that are mainframe-inclusive.

CRAFTING YOUR MAINFRAME-INCLUSIVE TOOLCHAIN

Enterprise DevOps teams can include the mainframe in their toolchains—and, by extension, into their end-to-end Continuous Delivery processes—using two complementary types of tools: cross-platform and mainframe specific.

1. ***Cross-platform tools* that support the mainframe**
   DevOps teams can benefit from tools that support multiple platforms, including the mainframe. Two areas where cross-platform tools can be particularly useful are:

Release Automation. In many cases, updates to enterprise applications require code to be promoted across several platforms simultaneously. This type of coordinated cross-platform workflow is often best executed using a tool that can orchestrate tasks across multiple platforms—including the mainframe, distributed systems and the cloud. A cross-platform tool also offers DevOps managers advantages when it comes to reporting and troubleshooting.

Quality Management. It's also useful for DevOps managers to use cross-platform tools for quality management requirements such as reporting and dashboards. Again, this allows managers to browse all data and metrics from all platforms relevant to any given release or application in a single place.

Cross-platform tools may, of course, piggyback on platform-specific tools. And it may sometimes be necessary for platform specialists to "drill down" to platform-specific tools as part of a find-and-fix process. But higher-level cross-platform tools are very useful on a day-to-day basis.

2. ***Mainframe-specific* tools that integrate into the DevOps toolchain**
   For other steps in the end-to-end DevOps lifecycle, mainframe-specific tools are more appropriate. That's because it is typically counterproductive to move thousands of mainframe application source code modules—or large volumes of mainframe test data—back and forth between platforms over the network. It is also essential to test application components in their target runtime environment, in order to ensure production readiness—especially given the prohibitive cost and time required to set up a simulated mainframe environment. Native mainframe environments also greatly simplify and streamline disaster recovery and DR testing for mission-critical applications and data.

Mainframe SCM tools that "hold code hostage" are unacceptable in today's Agile workplace.

Three areas where mainframe-specific tools work best are:

Source code management (SCM). As noted above, it's not economically or operationally practical to shuttle massive amounts of code between platforms. Also the speed of recovery required for mission-critical mainframe applications is best achieved by keeping source code on the mainframe. This way, that source code is subject to the same disaster recovery procedures as all other mainframe objects—and appropriate version control/linkage is maintained between mainframe production source and production objects.

Mainframe application environments are also characterized by highly complex (and often poorly documented) dependencies between application modules. Platform-specific intelligence is extremely useful for accurately navigating these dependencies to ensure code quality and developer productivity.

That said, DevOps teams need to exercise caution when selecting a mainframe SCM tool for their toolchain. Many traditional mainframe SCM tools require laborious coding for each reconfiguration or upgrade. That's unacceptable given the pressure on DevOps teams to be adaptable and efficient. Most mainframe SCM tools also require coding for build automation. This coding adds cost and slows development. A template-based approach that enables DevOps teams to quickly and easily define and modify processes is a much better fit with mainstream DevOps.

Also critical is the ability of developers to work in parallel. Traditional mainframe SCM tools that "hold code hostage" because someone checked it out—or, worse yet, forgot to check it back in—are unacceptable in today's Agile workplace (see chart).

Choosing the Right Mainframe-specific SCM Tool for Your Cross-platform DevOps Toolchain

This chart contrasts traditional siloed mainframe SCM tools with more innovative mainframe-specific SCM solutions that work well in the context of today's DevOps toolchains. Git is included for comparison purposes.

SCM Capability	Traditional Mainframe SCM	Distributed SCM	DevOps-enabled Mainframe SCM
Allows developers to work in parallel	✗	✓	✓
Customizable build processes	✗ requires custom coding	✓	✓ template-enabled
Visualization of code promotion	✗	✓	✓
Integration with deployment tools	✗ limited	✓	✓
Mobile-enabled approvals	✗	✓	✓
Early code collision detection	✗	✗	✓
Code promotion workflow for development, test and production	✗	✗ limited	✓

Build automation. The creation of production-ready z/OS application executables from mainframe source code is highly platform specific. Compile, bind and packaging operations must be precisely performed to ensure that code performs properly. Above and beyond the platform's precise (and often unforgiving) technical requirements, builds for mainframe applications that have evolved over the span of multiple decades may entail management of thousands of interrelated components with tens of millions of lines of code. That combination of scale, complexity and precision demands a specialized tool.

Ease of automation is particularly important when it comes to mainframe builds. If it takes too much time, effort and expertise to put builds together, the end-to-end software delivery process will always be delayed by compile, bind and packaging tasks. The process will also be highly vulnerable to errors that require full stop, potentially extended troubleshooting and re-start. The mainframe build automation tool in any enterprise DevOps toolchain will therefore ideally facilitate automation with some kind of configurable, template-based build prep capability with built-in platform intelligence—rather than requiring extensive custom coding by increasingly scarce mainframe SMEs.

Deployment automation. Tools that automate and manage deployment of mainframe applications in enterprise DevOps should also be platform-specific. Enterprise mainframe environments typically include thousands of software components with complex, hard-coded interdependencies. Various versions of these components may be spread across development, test and production areas. The right deployment tool ensures that all of these components are fully and accurately prepared to run in their target environments. It will also monitor those components throughout the deployment process to provide clear, ongoing visibility into that process—as well as fallback and re-start capabilities that the DevOps team can use if it encounters any issues mid-process.

In the case of mainframe SCM, DevOps teams must typically replace an existing, outdated tool that is inherently engineered for waterfall development with a new tool designed for DevOps workflow. In the case of mainframe deployment, DevOps teams are typically replacing existing homegrown scripts. Mainframe teams have often written many such scripts over the years. But an ad hoc collection of individual task scripts can't support Continuous Delivery. They don't provide complete coverage of all software components across all tasks at all times. They don't interface well with other tools in the toolchain. And they constantly have to be re-written, which slows DevOps considerably—and can be a real problem if the original script writer isn't around anymore.

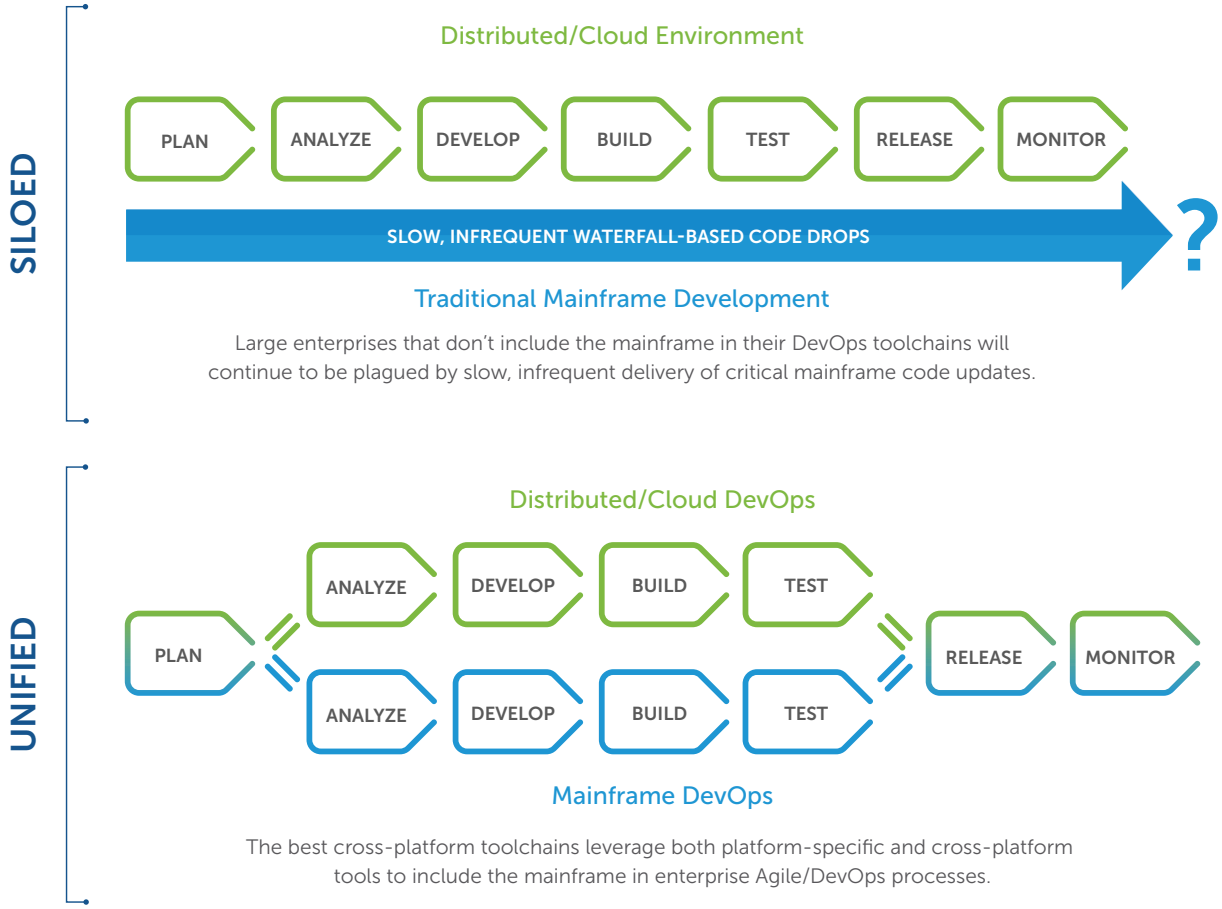
Mainframe DevOps tools should be as intuitive and graphical as non-mainframe tools.

Cross-platform DevOps toolchains work best when their mainframe tools provide interfaces that are as intuitive and graphical as those of non-mainframe tools. Ideally, mainframe DevOps tools should also offer built-in platform intelligence that makes it easier for IT staff without much hands-on mainframe experience to successfully perform common mainframe-related tasks.

Because DevOps toolchains engage both development and operations staff, mainframe tools in those toolchains should meet the needs of each. In the case of operations teams, that typically means providing secure web interfaces that allow them to quickly complete tasks from anywhere, any time. In the case of developers, that can often mean providing a plugin to an Eclipse-based IDE.

Mainframe SCM and deployment automation tools should also be tightly integrated with each other. This integration further streamlines DevOps processes and better enables IT to execute more mainframe code drops more often.

Of course, DevOps toolchains include other functions as well—including unit testing and the monitoring of application behaviors in production (see graphic). The functions discussed above simply serve as primary examples of how DevOps toolchains can be best expanded to include the mainframe's essential application and database resources.



IS IT URGENT?

Mainframe developers and operations staffs have done things the same way for decades. And DevOps teams focused on the distributed/cloud environment have been content to ignore the mainframe. So it is natural to ask if there is any truly urgent reason for enterprises to now bring both groups together via a common mainframe-inclusive DevOps toolchain.

The answer is most certainly a resounding "Yes!" Large enterprises must move quickly to build mainframe-inclusive DevOps toolchains because:

The ability of large enterprises to compete in digital-centric markets fundamentally depends on mainframe-inclusive agility. In market after market, large enterprises are seeing their marketshare and brand power being eroded by smaller, more digitally nimble market entrants. These new competitors almost universally have "green-field" IT that is cloud-based and native Agile. Large enterprises cannot compete with these companies if they can only manage a few code drops a year on their core mainframe systems. They must optimize agility across all platforms—the mainframe chief among them.

The mainframe cannot be wished away. Empirical studies make it clear that the mainframe will remain the platform-of-choice for core systems of record at large enterprises for the next ten years or more. So it doesn't make strategic sense to just hope that mainframe applications and data will somehow be magically re-platformed in the next 18 months. Nor does it make sense to adopt a strategy, as some have misguidedly suggested, of relegating the mainframe to some kind of "maintenance mode" while other platforms move towards Continuous Delivery. No business can be agile if its core systems are not agile. So large enterprises must invest in mainframe agility now.

Mainframe demographics are inexorably shifting. The relentless march of time is forcing IT organizations to confront the impending retirement of their most senior and experienced mainframe SMEs. This mainframe "brain drain" cannot be remedied by simply outsourcing development—for the same reasons that Facebook and Google don't outsource their newsfeed or search algorithms. Continued aggressive evolution of mainframe code is essential to an enterprise's competitiveness in digital markets, so internal mainframe mastery and vision remains essential. Enterprise IT leaders playing to win in the digital economy must therefore accomplish two objectives—and accomplish them quickly:

1. They must empower a declining number of senior mainframe SMEs to become vastly more productive.

2. They must empower their growing next-generation DevOps SMEs to master mainframe tasks.

These two objectives can best be accomplished by better automating mainframe development and operations tasks within the context of cross-platform enterprise toolchains.

The bottom line: Large enterprises that invest in mainframe re-tooling and DevOps toolchain integration will:

- **More successfully compete in today's fast-moving, customer-centric digital marketplace**
- **Drive down mainframe development and operations costs**
- **Better navigate the unavoidable generational shift in mainframe staffing**
- **Improve collaboration across IT disciplines**
- **Avoid operational and regulatory risks associated with mainframe errors**
- **Dramatically increase the value returned by their massive mainframe investments**

For these reasons and others, mainframe-inclusive DevOps toolchains are a must-have for every large enterprise.

> "Successful digital businesses unleash the data and business processes encoded in their mainframe-based applications."
>
> - "Digital Transformation Needs Mainframe DevOps," by Kurt Bittner and Rob Stroud Forrester Research, Inc., June 20, 2016

A GROWING CONSENSUS

The inclusion of the mainframe in DevOps toolchains is strongly endorsed by independent market observers. In a recent research note entitled "Digital Transformation Needs Mainframe DevOps," Forrester Research asserted that:

- Mainframe applications benefit from DevOps practices that boost speed and lower risk
- Years of neglect mean the barriers may be higher, but they are not insurmountable
- Improving mainframe application delivery speed is a survive-and-thrive imperative

And in a recent InformationWeek article entitled, "How Bimodal IT Can Kill Your Company," Forrester Research VP and Senior Analyst John C. McCarthy stated that:

> "... bimodal continues the cozy complacency between CIOs who don't want to transform and vendors who don't want to change. I don't think the strategy is going to work for a lot of these companies and a lot of these vendors."

A recent McKinsey research note entitled "An Operating Model for Company-wide Agile Development" further asserted that enterprises deploying Agile at scale accelerate their innovation by up to 80%.