Rocket® software

CI/CD for IBM® i+ DevOps:

Enable Experimentation for Multi-code Environments

Contents

Introduction

Businesses today face greater levels of uncertainty in global markets than ever before, and this may hold true for years. To ensure resilience to constant change and the ability to continue delivering innovative experiences and offerings, establishing robust DevOps infrastructure—one that powers true CI/CD cycles—is essential.

Effective DevOps infrastructure ideally runs on two main principles: The first is structure, building a repeatable process through automation and controls which drives productivity improvements. Inherent to a structured DevOps environment is a traceable process, key to simplifying compliance adherence and reporting.

The second principle is flexibility, which empowers those involved in your DevOps infrastructure to experiment with new ideas and the freedom to make changes. This experimentation can happen with code, UX, delivery, process, or anywhere the team believes an improvement can be made. But flexibility can only be enabled when there is structure to support it. This includes testing as early in the process as possible, or “shift left” testing. Only with structure from automation and controls—and flexibility through early and often testing—can businesses match the speed of market demand and even pull ahead in today’s ever-changing markets.

Back to contents

5 Steps to Planning CI/CD for IBM i+

Before implementing CI/CD, you need a planning approach. Below are five steps every successful DevOps planning process utilizes before a new project.

01 Standardize tools and processes

For the last 5 years or so, there's been a shift toward fewer vendors and better integration across the development steps: code, build, test, deploy, deliver, and monitor. Development teams still need to be able to select their tools of choice — such as Git for version control. But a vendor that has a more comprehensive DevOps platform can be your main point of visibility, reporting, and collaboration for the process. This standardization approach streamlines communications, minimizes onboarding, and allows teams a common means to rapidly commit, document, and test changes.

02 Strengthen requirements gathering

Sustained improvement and iteration can only be realized effectively with strong requirements gathering processes. Ensure project managers have the tools they need to collect information on evolving customer needs, translate those needs into development requirements, and track resulting iterations. Your requirements gathering and documentation should feed directly into your testing plan and provide clear communication to QA about what to test and how. Changes that flow from detailed requirements are the key to successful DevOps.

03 Due diligence

Always identify, analyze, question, and document your existing processes. Find out what best practices other teams maintain and incorporate those found successful. As your DevOps practices mature, you'll find more efficient and agile ways to operate. Monitor these improvements, blend them into processes, and maximize the impact that DevOps can bring.

04 Elevated testing

Continuous and automated testing squashes bugs before production, thus enabling faster experimentation. Provision multiple test environments and processes like unit testing, integration testing, and regression testing. Standardization allows multi-code testing to be done with greater predictability and by different people, resulting in a more stable production phase.

05 Consistent scoping

Due to the speed and velocity of CI/CD, consistent scoping of requirements is necessary to ensure accurate changes are made and resource usage is optimal. Supply the necessary tools for visibility over the entire development cycle, allowing agile planning and the communication of scope changes to relevant stakeholders.

← Back to contents

Implementing DevOps Testing for Multi-Code Environments

The need to incorporate existing systems infrastructure into DevOps implementation brings unique challenges, especially if those systems are established multi-code, multi-system, and multi-endpoint environments. Also, most DevOps tools don't integrate well with IBM i application or code structures, putting organizations that rely on these core systems at a disadvantage. This is especially challenging when testing IBM i applications. With few testing tools on the market, many companies resort to manual or rudimentary scripted testing.

Establishing CI/CD testing within multi-code environments also means that teams must reconcile multiple, and often conflicting, development workflows and approaches like Agile and Waterfall. CI/CD workflows also face disruption from the different development timelines of various coding languages — especially when managing modern languages like Python and Java and more IBM i-centric languages like RPG. There are also misaligned QA controls, a lack of integrated code testing across the board, and misalignment between development teams to consider.

The edge provided by CI/CD for IBM i, however, is worth overcoming these challenges, which are solvable with the right tools and best practices. Rocket is an organization that lives and breathes multi-code and multi-system development. We've learned key DevOps testing best practices that keep the development and larger operations of today's businesses on the front foot.

Back to contents

5 Testing Best Practices for IBM i DevOps

01 Apply high standards to your IBM i applications.

This means automating as much of the testing process as possible, not relying solely on development for testing, and testing as much of the code and how it interacts with the rest of the application as you can. Also, find an IBM i application testing vendor that can test not only IBM i code but also browser-based code, or other apps that use APIs to link to IBM i applications and data.

02 Apps are more commonly designed to interconnect across systems.

Multi-code applications or UIs mean longer testing times on account of multiple testing tools. Ultimately, code takes longer to reach production and QA remains stuck working on the same code instead of moving on to the next task. Instead, find a testing tool that is as close to code-agnostic as possible.how. Changes that flow from detailed requirements are the key to successful DevOps.

03 Manual testing can play an important role in QA but most testing is best done through automation.

Manual testing is slow and more prone to missing bugs. Simplified testing is similarly problematic. If most of your testing is done via very specific developer scripts, you're likely not capturing all the testing data you need, such as how the code is changing the database. You also can increase productivity within the testing tasks, by building repositories of tests that can be triggered when new code is introduced. Without automated testing, done early and often, agile development and CI/CD are nearly impossible.

04 Separation of Duties (SOD) is critical to testing.

Many regulatory mandates that touch on sensitive data or security specify that those who write the code should not be the ones to fully test it. The right DevOps tool has access controls and permissions built into the platform so you can easily stay compliant—and prove that you are offering a secure way to develop and test code.

05 Testing should be an integrated and managed part of your DevOps environment.

As application functionality grows and is extended to new platforms and UIs, a typical release might involve dozens of developers. Eventually that code comes together to be tested. To properly coordinate the timing of the code being released to QA, and the availability of QA engineers, can be complicated if testing is not a natural part of your DevOps environment. With an integrated platform, there's enough visibility across functions and teams to simplify coordination, saving teams precious time.

Back to contents

Bringing it all Together

What do these best practices look like in the day-to-day life of your DevOps teams? How exactly do they facilitate more fluid and continuous workflows, from development all the way to delivery? Here's how a successful CI/CD-driven cycle plays out from end to end.

Develop Phase

This is where the tone and cadence for any development cycle is set, and where the structure and alignment of DevOps hold tremendous value:

- Ensure everyone on the project understands their roles and scope, what tools and technologies are standard issue, and the appropriate processes and workflows.
- Establish the cadence and frequency of daily standups, meetings, or town halls, based on what works for your teams. These meetings are for progress updates, task allocation, and resource management. Align everyone on a set of control repositories, especially if your applications use traditional IBM i code, such as RPG and COBOL, as well as non-traditional code, such as Python. This minimizes discrepancies and variables as development shifts between traditional and modern environments.
- It's recommended to break down development into two-week sprints—at least to start—to allow for more flexible iteration from user feedback. Testing can be slotted where sensible for easy management and execution. As your teams adjust to faster sprints, you can experiment with shorter sprints to enable faster testing and experimentation.

Build Phase

In this phase, continuous integration and continuous development practices leverage real-time user feedback loops, automated testing, and agile development:

- Project managers must establish strong lines of communication with all stakeholders to provide user feedback, scope requirements, and kickstart development.
- Leverage a DevOps platform that allows teams to track code changes and access a shared resources, such as code repositories, test cases and results, and dashboards. Choose platforms capable of automated testing for detecting bugs or errors before code is deployed onto a live production environment.
- If you don't have good test data, you might as well skip testing all together. Generate test data from live systems in real-time and ensure that sensitive data is anonymized.

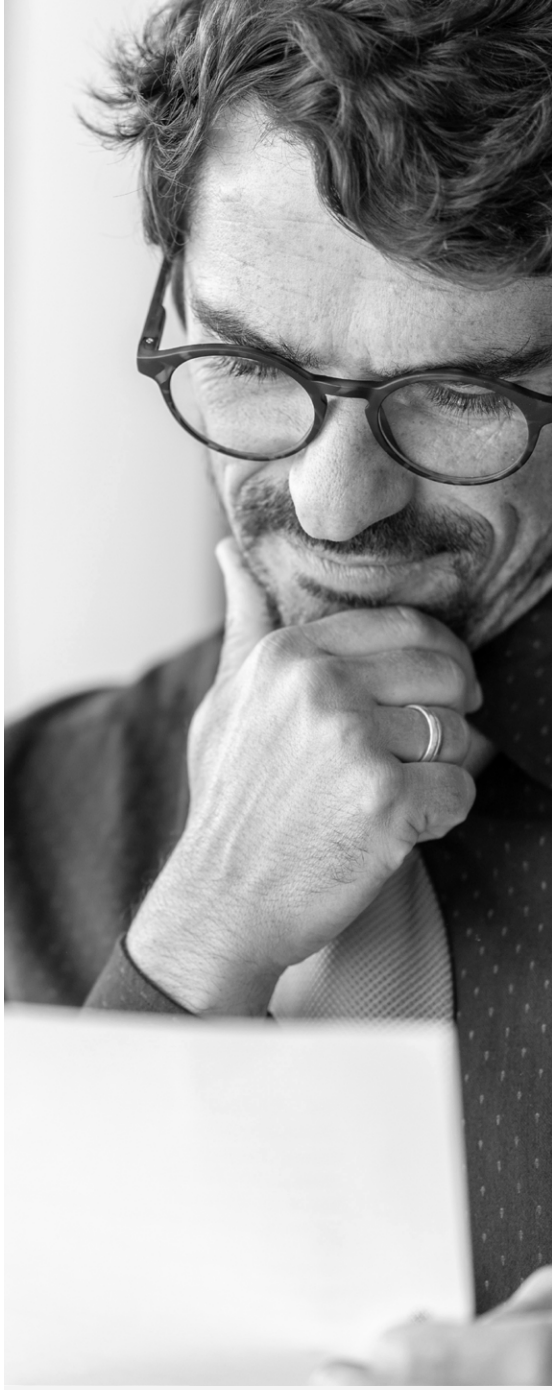
Back to contents

Delivery and Deployment Phase

Usually the most time and resource-intensive stage, and one where the benefits of CI/CD can truly shine. Heavy use of automation here allows teams to focus on more high-value tasks:

- Use a standard set of tools and processes that allow code to be committed to a repository with minimal integration and deviation, speeding up the delivery process.
- Leverage automation for continuous delivery and deployment, especially when multiple environments or production servers are involved. This frees development teams for higher value tasks. Ensure automation is set for rollbacks or backups in the event of a mishap, ensuring teams a stable system when they return to work.

Implemented well, CI/CD has the potential to instill greater levels of innovation, resilience, and competitiveness in today's uncertain markets. Leading DevOps platforms even provide a standardized suite of productivity and collaboration tools, out-of-the-box automation workflows for testing and deployment, and the ability to integrate a large variety of third-party applications.



Learn more about how you can enable CI/CD for IBM i today!

Request a demo

Back to contents

Rocket® DevOps

Secure, end-to-end CI/CD for IBM i

Rocket® DevOps is a modernization platform specifically designed to enable end-to-end Continuous Integration and Continuous Delivery (CI/CD) practices for your multi-code environments that include IBM® i. Rocket DevOps enables you to build the structure and flexibility you need to extend holistic DevSecOps best practices to the IBM i, pursue innovative experimentation, respond easily to compliance audits, and adapt to ever-changing expectations be it via process, technology, or experience.

Product Benefits

- Leverage an end-to-end DevSecOps solution built for IBM i powered businesses
- Standardize DevSecOps best practices across the organization
- Deliver holistic and simplified reporting
- Build structure to speed up time to market and ensure compliance
- Make truly flexible CI/CD processes a reality
- Enable non-RPG talent to engage and support the IBM i DevOps process

The unmatched experience of the Rocket DevOps' services team quickly positions you for success with a customized implementation that works best for your business and with the power to take ownership over any future changes.

Rocket® DevOps solutions provide the tools and infrastructure you need to automate everything from the moment a change request comes in until a solution is delivered to production. Keep your software running while using it in new ways to increase business value.

Request a demo

About Rocket Software

Rocket Software is the global technology leader in modernization and partner of choice that empowers the world's leading businesses on their modernization journeys, spanning core systems to the cloud. Trusted by over 12,500 customers and 750 partners, and with more than 3,000 global employees, Rocket Software enables customers to maximize their data, applications, and infrastructure to deliver critical services that power our modern world. Rocket Software is a privately held U.S. corporation headquartered in the Boston area with centers of excellence strategically located around the world. Rocket Software is a portfolio company of Bain Capital Private Equity. Follow Rocket Software on **LinkedIn** and X (formerly Twitter) or visit **RocketSoftware.com** to learn more.

Modernization. Without Disruption.™

Visit RocketSoftware.com ›

Rocket software

© Rocket Software, Inc. or its affiliates 2024. All rights reserved. Rocket and the Rocket Software logos are registered trademarks of Rocket Software, Inc. Other product and service names might be trademarks of Rocket Software or its affiliates.

IBM and IBM i are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide.

MAR-10713_CI-CDforIBMiDevOps_Whitepaper_Rebrand_v5