Rocket® software

Study: The Future of IBM® i+ CI/CD

Contents

Introduction

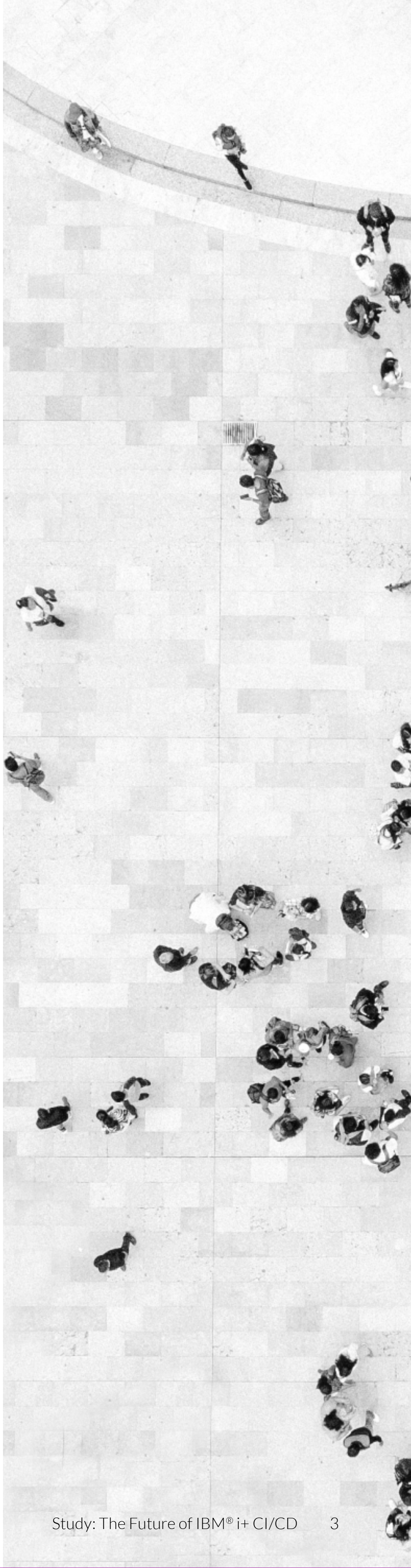
IBM® i teams are in a state of forced change

When organizations were first developing on IBM® i, the work was only RPG, native, green screen-based development in a linear, waterfall method: code, compile, test, deploy, and after months of work, release. Development was still decades away from anyone suggesting agile development, let alone implementing the practice. There were no mobile phones. No Internet. No cloud. Even when the rest of the organization was beginning to adopt agile development in the 2010s, IBM i teams held fast to their waterfall approach. After all, it worked well for them; and while the storm was brewing, the teams were not yet feeling the pressure to change.

That time is long gone.

Over the last five years, IBM i teams have been expected to perform like other DevOps teams. That means:

- Agile development cycles with fast response times to market and customer needs and quality code in a multi-code environment
- DevOps is becoming the standard, which means close collaboration between development and operations/IT for better visibility across the entire development and deployment process, tighter controls of sensitive code and data, and more
- Coordinating with another team—for example, Java developers—across the DevOps process; otherwise, the IBM i developers would also have to learn how to be Java developers
- They have new stakeholders. IBM i no longer lives in silos and is being pulled into new workflows and UI/UX
- Organizations are recruiting younger developers due to development skill gaps on IBM i—and these developers have a new, modern view of what development looks like, what tools they need, what the methodology should be like, and more



← Back to contents

Many IBM i developers are entering retirement age or have already left the labor market. These are the developers who have a deep knowledge of the applications, how they were built, and the business logic behind them. Unfortunately, there is very little documentation left behind. Without documentation, teams are often guessing what they should build—and, by extension, what they should test. For example, let's say you have a customer service representative "enter new customer profile" workflow that is automated with new APIs. What exceptions do you test to make sure the API is working as intended? With modern development and API technology, you can simply go into "add customer profile program" and let it do its thing. The developer would understand the code and see what it was doing. But because developers working on updates today don't know the code, they won't know if the API should be calling other programs or if the code was calling other programs under the covers.

But the future is bright

As an IBM i owner, you've accepted your fate. You need to be agile. You need to work with modern toolsets. You need to extend a hand outside your immediate team to the rest of the organization. Modern DevOps is here to stay—and that's a good thing. It reduces time to market, increases enterprise agility, and makes businesses more resilient.

One of the current trends is continuous integration/continuous delivery (CI/CD), which enables DevOps best practices and benefits more than other approaches. It allows DevOps teams to work faster without compromising quality.

Continuous integration is the practice of continuously integrating new features, functions, and UI into applications. An example would be a Microsoft developer making a change to Word. They would write code and then use continuous integration to integrate it into Word somewhere, allowing it to be tested while other developers are doing something else in tandem.

Continuous delivery is about sending code when it's ready and needs to be tested as an application at the site/server. The test is to see whether it fails within an environment with a specific combination of servers, OSs, applications, and configurations. The developer pushes the code to the testing team, who then delivers it to the environment for testing. There are versions of the applications that are created so a developer can send them when their code is ready, not when everyone's code is ready.

Continuous deployment is about pushing new changes or codes that pass through to production and to end users who ultimately provide feedback. This happens post-test and is sent based on a release schedule, which can be continuous, every x number of minutes, or some variation of an agile deployment cycle.

← Back to contents

Once that continuous loop is set up, organizations have more freedom to innovate and experiment. There are two important parts to making CI/CD work:

01

Automated testing

Without it, the cycle slows down, and the potential for errors entering production increases. Can you get quality code released to the market quickly so your teams can get the feedback they need and fine-tune their applications to fit what the market wants?

02

The right toolset and the right implementation of that toolset

Do you have the right tools, configured and leveraged in the right way, to support and accelerate your CI/CD approach?

In a 2021 Github survey, 57% of developers said they release code twice as fast as before, up from 25% faster just two years prior. 75% said they use or are planning to use AI or bots for test code review, and nearly 25% of respondents claimed full test automation.

Another interesting and interrelated trend in DevOps right now is in the world of testing. In a 2021 Github survey, 57% of developers said they release code twice as fast as before, up from 25% faster just two years prior. 75% said they use or are planning to use AI or bots for test code review, and nearly 25% of respondents claimed full test automation. While IBM i development trends tend to lag behind the rest of the market, the writing is on the walls. Companies are moving faster and leveraging test automation to do it.

In addition, we are in the early stages of a trend around process discovery. Process discovery is the practice by which organizations understand how users engage an application and dataset, including the specific steps they take in workflows, such as what data are accessed when, how long they spend at certain points in the workflow, steps skipped, shortcuts used, and more. Process discovery takes the guesswork out of what is needed to develop and test, accelerating the execution of automated testing. Testing and process discovery are evolving into more than standard steps in the DevOps process.

← Back to contents

Process discovery is a prerequisite for accurate AI enablement—yet another trend—but one that is fast-moving and likely to heavily influence parts, if not all, of the DevOps practice. To work, however, AI needs data. **With data from process discovery, AI can support:**

Modern UI

In addition to making recommendations on the layout, a chatbot could be the new user interface to an application.

Application integration

AI will integrate modern applications with other systems and services, such as APIs, databases, and third-party applications, improving interoperability and data exchange.

Microservices

AI can identify opportunities to break up monolithic applications into smaller, more manageable microservices, improving scalability and flexibility.

Automation of modernization development

If you are leveraging a process discovery tool, AI could be built to automatically generate the code, whether it's refactoring through API creation, making UI updates, or anything in between. This would mean AI is not just identifying a hotspot of engagement within the application but also building the code with a push of a button. Today, there are even instances of developers leveraging AI to develop the code before pushing it to a Git repository.

Trends like process discovery and AI can sound overwhelming to a team still heavily using linear development and deployment processes. But if an IBM i team knows what's coming and how to prepare, they can set up their approach to DevOps in such a way that the sky is the limit for innovation and experimentation.

← Back to contents

The future of CI/CD for IBM i+

Process discovery is the precursor to AI

There is much conversation about process discovery in the context of modernization projects, for good reasons. Process discovery is a great tool to understand how your business engages applications and data on IBM i, giving you the insight you need to decide how your modernization should look. Process discovery is also key for DevOps processes, e.g., how to modernize. It can provide insights into how to update the code (e.g., move this field, automate this part of the workflow, etc.) and what and how to test the code.

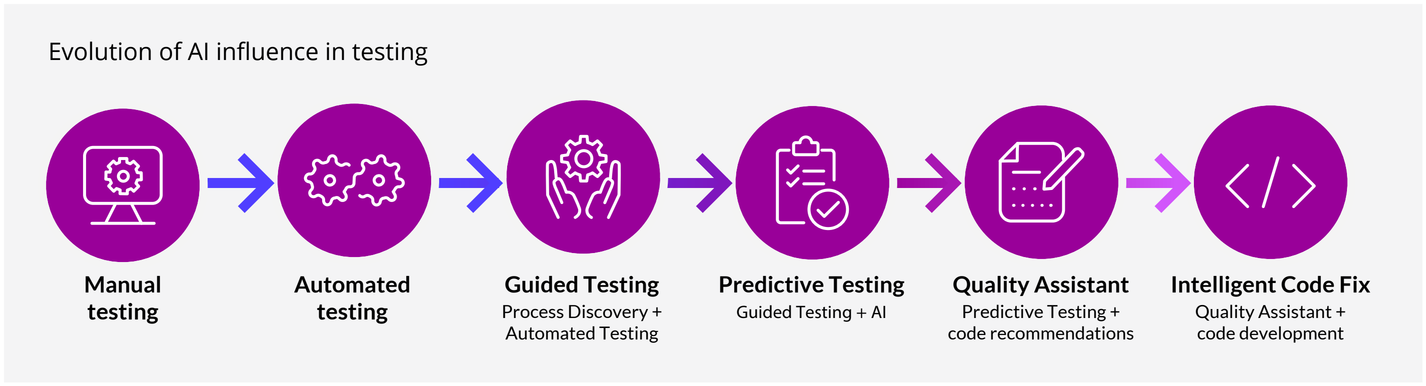
In addition, there is a strong push for automated or “shift left testing” to enable agility, speed, and quality, essentially achieving true CI/CD. The beauty of automated testing is the ability to free up resources and shift timing, enabling faster time to market and more innovative development. With automation, the options for where, when, and what DevOps teams can test are expanded. Before, most testing happened after most of the development was done, right before it was pushed to production; now, developers can test during development. Automated testing also makes doing specific tests like integration and regression easier. And, if you are truly agile, your release to users is in and of itself a test of feature/function/UX.

The future of DevOps is the combination of new and established technologies to accelerate innovation even more. This is not just for IBM i, but it is critical to IBM i teams.

With technology as transformative as AI, it is difficult to accurately predict the evolution of how AI influences DevOps more than a few years out. However, we believe in the prediction of AI’s role in testing below with high confidence.

← Back to contents

The Influence of AI on Test


Evolution of AI influence in testing
Manual testing
Automated testing
Guided Testing
Process Discovery + Automated Testing
Predictive Testing
Guided Testing + AI
Quality Assistant
Predictive Testing + code recommendations
Intelligent Code Fix
Quality Assistant + code development

Below is an explanation of each step in the evolutionary journey of testing. Manual and automated testing are self-explanatory; let's dive into the rest.

Guided testing

While understanding how an application is developed is important, what's key is how it is used. An application can be developed for CRM, for example, with standard database fields and typical workflows for a CRM application like "create new customer profile," "update profile," and so on. But what businesses really need to understand is:

- Who uses the application? Is it just the contact center? What about accounting? Sales? Legal?
- What are they doing in the application? Are they jumping to screen 12 and updating one field 90% of the time? Do they start at the first screen, but the CRM team goes through workflow 1 while accounting goes through workflow 2?
- How do these teams use it? Are they in the application 8 hours a day? How long do they spend creating a new profile or making an update? Perhaps accounting is only in the applications 10% of the time compared to the CRM team, who is always in there, but the workflow's accounting walk-through is critical to managing the business's finances.

When it comes to testing, if you know what you must test, you can test more often and build more comprehensive testing scripts that can be reused. Integration, unit, and regression testing can all benefit from test scripts built from process discovery insights. If you build a repository of tests, you can theoretically turn 2 months of testing into 2 days or even 2 hours. The repository becomes a knowledge base for the team that is continuously updated, getting better over time.

← Back to contents

Predictive testing

Once you have the data from process discovery and you're heading towards "shift left" testing, you can add AI into your testing process. Predictive testing knows what to test when code starts changing, and the AI can trigger pass/fail testing before QA knows it needs to be tested, lessening the burden on the testing team.

The AI will need the rules by which it tests your code. While the AI vendor will likely have a standard set of rules preprogrammed into the model, you will need to customize the model based on what your processes look like.

Often, there is already AI in other parts of the DevOps process—for instance, there may be a model that is trained to compile code in the same way it was compiled before. However, you could build nuances into the model or change it directly to compile the code differently. For example, when code is ready to test, you could tell the AI model to pull a particular library from the test repository, depending on the workflow that is being updated.

AI can also be built to establish relationships between tests already run and changes that are made. This means functional testing can be focused on specific areas with minimal human intervention.

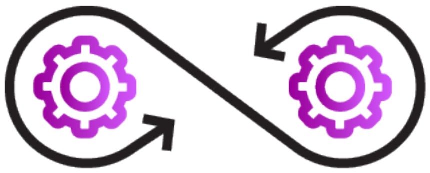
Quality assistant

Think of this stage as an Amazon recommendation that suggests other things to buy based on what is currently in your cart, and what other customers bought in addition to that thing you're about to buy. As quality assistance, AI can make recommendations about how to update code based on a failed test the AI model ran. For example, the AI model could raise a flag that says, "You changed this part over here, but you either haven't tested it or haven't tested it as comprehensively as you did before—are you sure you don't want to?" Or it could prompt you to make sure you want to test code from a workflow that is not recorded in the process discovery, which could mean that the process is not fully understood.

Intelligent code fix

At this stage in the evolution of testing, we expect machine learning to step in, taking the intelligence built into the AI model and learning to finetune the rules to reality—perhaps even improving on the intelligence itself. For example, sometimes you want to change how something is compiled, but only in particular situations. The AI can understand when that is the case or not and updates the rules based on that information. There are three ways this could be incorporated into testing:

- The machine learning model can notice when you omit a particular test you typically run, telling you it will run it because the scenario is similar to when you ran it in the past.
- There may be rules to test something in a certain way, but based on the process discovery data, users seem to engage that workflow in a different way. The machine learning model can recommend a different approach to testing based on the process discovery data. The machine learning can also review code that's been moved forward from development to make sure that testing is appropriate and comprehensive, adding a test to the queue if it's already available or pushing a request to either QA or the developer to create a new test to add.
- In a fully advanced scenario, the machine learning model could create the test, run it, and do a code fix without any human interaction.



Intelligent testing is the way forward

Organizations will likely see an explosion of innovation enabled by QA when moving toward intelligent testing. The QA team can build more comprehensive test cases based on process discovery and then share the test they built with development, which could include scripts for unit and integration testing. This empowers development to fix errors more quickly and build higher-quality code from the beginning. While there will always be some level of functional fixes that come from user feedback, pushing higher quality code to production will increase the share of the user experience and in situ feedback, thus speeding up the CI/CD cycle. This sets up the organization to be more competitive, uncover more opportunities for improving productivity, and align more quickly and closely with what the market is looking for.

In addition, intelligent code fixing brings up the question of "how does AI change an organization's approach to RPG?" It's possible for AI to take over both the development and testing of RPG code, minimizing the resource challenges for IBM i systems. In fact, it's possible for an AI to be the IT admin, developer, and QA team for RPG, all in one.

We're likely years away from that last scenario becoming commonplace. In fact, it might never happen.

← Back to contents

There already have been situations where the limitations of AI are concerning. In fact, there was a letter published in Spring of 2023 and signed by hundreds of leading engineers and scientists, including Steve Wozniak—founder of Apple, pushing for a halt of AI model development for six months. The goal is to give the industry time to review and provide recommendations on policies and procedures that address the serious ethical challenges AI introduces to the world economy, not the least of which is human displacement from the workforce. In addition, governments are starting to ban the use of ChatGPT. Italy announced a temporary halt of the "…processing [of] Italian users' data amid a probe into a suspected breach of Europe's strict privacy regulations."

Other challenges of AI that have already been experienced stem from unconscious bias being built into the AI models. For example, because AI execution is directly related to the data you use to teach the model, there have been instances where biased data influenced its behavior in such a way that it was labeled racist and sexist. Governance around what data we use to train AI and how we manage that data is paramount if organizations don't want a PR disaster on their hands.

Many of the concerns of AI are driven by the limitations of the technology, specifically:

- **AI has no common sense:** It cannot understand context and meaning, which can lead to errors and misinterpretations, as evidenced by the example above.
- **The high cost of training and managing AIs:** Not only do you need a large amount of accurate, high-quality historical data for AI to work well, but you also need to continue to feed it quality data so the models can be finetuned to work better over time. The probability of biasing the AI is high if one isn't careful.
- **Code ownership:** AI doesn't create anything new; it generates answers based on what it's seen in the data it's given. Who owns the code if a development team decides to use AI to build code, and the AI's data input includes third-party code? There's a potential liability for businesses here that could create an issue if that AI-built code is used externally.
- **Transparency:** AI algorithms can be complex and difficult to understand. We know the inputs and the outputs, but sometimes there is a "black box" of decision-making happening within the AI. This complexity will only grow over time, and the risk this black box poses will grow, too, as AI evolves to not just decision-making but execution, as well, without manual review.

Every organization will need to decide what their risk tolerance is for AI. DevOps teams might decide to implement permissions across AI models in a similar way they do with human workers. How big of a "black box" that organizations might want to build around their critical data and applications will also be a factor. Still, one can see the potential of this technology to revolutionize not just testing, but also DevOps overall, especially within IBM i environments.

What are your next steps?

As an IBM i owner, you're likely trying to envision the future for your IBM i as well as what steps you need to take to make it a reality. Below are the key considerations we recommend you review when determining your future DevOps strategy within an IBM i environment.

What type of IBM i company are you?

There are three different types of IBM i organizations:

01

Your IT environment is IBM i only, now and forever. You don't have other systems where you host applications and data; if you do, the other systems are incidental at best.

02

IBM i is the central system within your IT network, but you do have peripheral systems and places where you need to engage the applications and data on IBM i.

03

IBM i is an important but not central system within your IT network. Other systems also host critical applications and data, and the organization is trending towards continued increasing complexity.

How would you answer that question today? How about five years from now? If you are the first type of organization and don't see that changing anytime soon, perhaps you don't expect to take advantage of modern tools, technology, and trends. Most IBM i organizations tend to look more like the second option, where IBM i is central in the network, but there are other systems that IBM i needs to work with. Often, organizations in the last type are large and have had mergers and acquisitions in the past. The last two types of IBM i organizations benefit the most from adopting modern IT best practices, of which intelligent testing is one (or will be soon). Organizations that are more like option three often are already on the path of planning how to take advantage of new technology or are planning to start down that path soon.

← Back to contents

Move towards agile development

If you're like most other IBM i teams today, you're already on the path to agile product development; slowly, perhaps, but on the path nonetheless. The need for agile processes is greater than ever before if you want to take advantage of modern tech and best practices. AI is fast and iterative by design, as new data and outcomes must be continuously fed into algorithms. AI will quickly break down into a waterfall development and process improvement approach.

If the IBM i team speeds up the DevOps and modernization processes, the amount of data available from process discovery to either the IBM i team and/or to AI increases significantly. It also means feedback to the models is fast, allowing for faster tuning of the model to its intended goal than if development was deploying code only once or twice a year.

An agile approach could also make it easier for businesses to quickly test new tools and adjust based on what works best for them. For example, maybe you adopt Git for your source code repository but find it doesn't really provide the functionality you had in your previous tool; with an agile approach to DevOps, you can make adjustments quickly.

Reimagine QA in the world of intelligent testing

No matter how much automated testing you have, you will still need manual testing. But QA won't need to look for superficial bugs like color issues, spelling errors, or the location of fields—because test cases can be easily built with intelligent testing and shared with development early in the CI/CD process. Instead, QA can focus on higher-level issues that could affect the functionality and usability of the application, such as user, regression, or exploratory testing. QA teams can also spend more time figuring out what needs to be tested and building more comprehensive test cases based on process discovery. The question is: what makes sense for both your DevOps process and the larger organization?

Build a data strategy... and stick to it

At its core, AI is a data synthesizer and juggler. It relies on high-quality historical data to feed the model and uses the data it receives to create uniquely aggregated outputs. Continuous data inputs then feed into the AI model, so adjustments are made to fine-tune its performance.

At their baseline, organizations need a data strategy to ensure the availability, quality, and security of their data, especially if that data are Personal Identifiable Information (PII). The strategy should include:

- Data governance policies—how you handle PII and ensure that biases are not built into models
- Investing in data infrastructure
- Building an AI team to build and monitor models
- Training employees on best practices for data management

Start collecting data now

Since AI will rely heavily on good historical data to get started, organizations should start collecting data now. The types of data you'll want to collect are:

- Copies of databases
- Data on workflows: how users engaged the database and the applications
- Information on how things work: for example, if you want AI to help process insurance claims for the United States, make sure it understands how insurance is regulated in the U.S.
- Language models

One of the easiest ways to get started would be to keep your audit logs each year. However, you will need a recorder that understands the ins and outs of the IBM i system to collect all the relevant data. It should be able to track engagement with an application, such as libraries accessed, the application's business logic, and how it engages other applications, as well as the steps users take through an application.

Envision how DevOps teams coordinate with data scientist teams

Typically, data collection and R&D happen in different departments of a company, which makes for a potential disconnect when considering time to market. To take advantage of machine learning within DevOps, the ML code should be promoted throughout the development cycle. DevOps teams should coordinate closely with MLOps or ModelOps teams when deploying new software packages. DevOps can also adopt AI engineering best practices to better align with MLOps/ModelOps teams.

← Back to contents

Rocket Software

Rocket® DevOps is a platform designed specifically to enable end-to-end CI/CD for IBM i+ environments. Businesses can build the structure and flexibility they need to extend holistic DevOps best practices to IBM i, while enabling their teams to easily adapt to any change in process or technology.

With Rocket DevOps:

- Enable end-to-end DevOps with a solution that delivers everything from deployment management to "shift left" testing, your IBM i development and delivery teams will have everything they need to develop and deploy high-quality code quickly, efficiently, and securely
- Build structure through automation and controls. Remove as much manual, error-prone work as possible, enforce compliance mandates through separation of duties, and drive faster time to market
- Empower flexibility for true CI/CD and enable teams to quickly experiment, make changes, and adapt as needed to drive innovation
- Deliver holistic, simplified reporting to quickly respond to regulatory audits and SLAs
- Standardize DevOps across the company with integrations to popular third-party and open-source DevOps tools
- Enable non-RPG talent to engage and support the IBM i DevOps process, alleviating some of the bottlenecks with RPG developers while getting code out of the door

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

Rocket DevOps is part of the Rocket Solutions portfolio for IBM i. Meet digital age demands while maximizing your IT investment. The world is evolving; so should your applications.

Learn more about Rocket® DevOps and how to enable CI/CD for IBM i+ environments

Book a demo

About Rocket Software

Rocket Software partners with the largest Fortune 1000 organizations to solve their most complex IT challenges across Applications, Data and Infrastructure. Rocket Software brings customers from where they are in their modernization journey to where they want to be by architecting innovative solutions that deliver next-generation experiences. Over 10 million global IT and business professionals trust Rocket Software to deliver solutions that improve responsiveness to change and optimize workloads. Rocket Software enables organizations to modernize in place with a hybrid cloud strategy to protect investment, decrease risk and reduce time to value. Rocket Software is a privately held U.S. corporation headquartered in the Boston area with centers of excellence strategically located throughout North America, Europe, Asia and Australia. Rocket Software is a portfolio company of Bain Capital Private Equity. Follow Rocket Software on LinkedIn and Twitter.

The future won't wait—modernize today.

Visit RocketSoftware.com ›

Book a demo

Rocket software