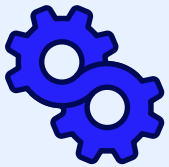
appian

DevOps and Low-Code: Better Together

Best practices for combining the strengths of DevOps and low-code to drive business advantages.



DevOps

A set of principles and practices to meet common objectives of software development (Dev) and IT operations (Ops). DevOps has caused a cultural shift by fostering collaboration between these traditionally separate groups.

Cross-functional DevOps teams work on solving internal development problems and eliminating bottlenecks to the rapid, frequent, and reliable delivery of high-quality software.

+

Low-Code

A set of development tools, environments, and platforms for building software that requires minimal coding. Low-code platforms instead use visual interfaces (drawing tools, wizards, model- and template-driven design, etc.) with automatic code generation.

Low-code is being adopted both by developers (for work acceleration) and business experts (for skill enablement).

Why talk about DevOps and low-code?

DevOps and low-code have complementary goals.

DevOps is about optimizing the software engineering process so you can deliver applications to users faster, compressing time to value. Low-code is about building software faster so it enters DevOps pipelines sooner. It's also about giving organizations the choice of mobilizing a wider portion of their workforce in developing, testing, and updating software applications and tools—accelerating pace and potentially off-loading some routine tasks from DevOps teams.

Their objective is to deliver better results, faster.

Leading-edge DevOps is expanding its focus to include quality engineering as well as speed. Today's goal is to get better software to users, fast. Leading-edge low-code platforms can handle complex challenges with increasingly sophisticated tools for software quality, standardization, and disciplined development. These low-code platforms provide workflow automation and support agile methods and efficient DevOps practices for the entire software life cycle.

They both have a growing presence across enterprises.

DevOps principles and methods have been adopted by 74% of enterprises, according to research reported in a 2021 ZDNet article.[1] Meanwhile, Gartner® forecasts that "by 2025, 70% of new applications developed by enterprises will use low-code or no-code technologies, up from less than 25% in 2020."[2]

Wherever you are with DevOps and low-code, this eBook will help you combine their strengths. Maybe your organization has a dedicated DevOps team, or maybe a senior engineer has taken on DevOps responsibilities. You might have engineers and business analysts already using low-code methods, or you might be researching and evaluating low-code platforms. In any case, leveraging the combined strengths of DevOps and low-code will increase your success and lower your operating costs.

1. ZDNet, "DevOps adoption almost doubles in five years, Covid crisis accelerated adoption," February, 2021.
2. Gartner Risks and Opportunity Index: Low-Code Platforms," Fabrizio Biscotti, Paul Vincent, Jason Wong, Laurie Wurster (June 2021)

How low-code advances DevOps.

As much as DevOps has already improved software engineering processes, it can still make even more of an impact. There are two bottlenecks to smooth-running pipelines that can be cleared by coupling enterprise low-code with DevOps: the coding bottleneck and the testing bottleneck.

1. Eliminate the coding bottleneck.

The efficiency of DevOps relies partly on an efficient stream of high-quality code entering continuous integration/continuous delivery (CI/CD) pipelines.

> **During the pandemic, many organizations embraced low-code platforms to build and deploy new apps fast. These experiences will drive most development shops to adopt low-code tools and more. Expect to see new hybrid teams emerge, with business users and professional developers building apps together with low-code tools built on cloud-native platforms.**

Predictions 2021: Software Developers Face Mounting Pressure, Forrester, October 2020.

Low-code platforms help by doing the following:

Enabling IT developers to accelerate the code stream using, for example, prebuilt connectors that eliminate data silos, acting on data wherever it resides without need for migration. Developers gain speed from reusable objects, such as workflows, business rules, integrations, and APIs. With a best-in-class platform, they can rapidly deliver automation solutions that combine classic business process management and case management with cutting-edge capabilities such as machine learning, process mining, AI-based intelligent document processing, and robotic process automation.

Enabling business experts to contribute to the code stream by working with the same powerful capabilities as developers, except using visual tools (for example, drawing processes instead of coding them), templates, UI frameworks, and guided design flows. Low-code platforms also support strong collaboration, complementing the agile practice of breaking work down into sprint-length tasks performed in parallel by different teams. Any contributor can readily see how an in-progress application's functions are mapped out without having to decipher lines of code.

Enabling developers and collaborators to deliver secure applications with minimal or no effort. Enterprise-grade low-code platforms enable "software, safer, sooner" by baking security into development tools and automating the delivery of secure software—including native rendering on mobile devices—without slowing the process down. At each step in the development cycle, objects and the application are automatically tested for security issues so they can be addressed as soon as they are identified.

2. Eliminate the testing bottleneck.

DevOps' success at increasing the pace of software development, along with added demand for speed coming from accelerated digital transformation, can create a pile-up of work for quality testing.

Low-code platforms help by doing the following:

Supporting the shift-left trend in software testing aimed at finding and fixing code defects earlier in the development process when they're exponentially less complex and less expensive to address.

Putting more hands on deck by enabling business experts, such as product owners and customer support, to play a role in authoring/updating tests and evaluating results.

Expanding the focus of testing beyond code validation to a wider range of software functionality. With business experts involved in testing, it becomes feasible to assess more aspects of user experience and confirm business value earlier on.

> Quality engineering (QE) has traditionally lacked the tools needed to keep pace with digital transformation. Many QE teams struggle with complex, brittle automation frameworks and slow manual testing which create serious bottlenecks to DevOps adoption … we need to shift our focus away from quality assurance as a task you perform late in the software development life cycle in favor of quality engineering practices throughout the entire life cycle.

Low-Code Automated Software Testing Drives DevOps,
Devops.com, April 2021.

How DevOps is changing low-code platforms.

IT organizations are embracing low-code platforms for adding more enterprise-grade features, including support for DevOps practices. Robust security and native integration with CI/CD tools are also raising the profile of low-code—these capabilities increase organizations' confidence that low-code can be a force multiplier in the increasingly competitive race for software-enabled business advantage.

Low-Code Features

- Visual programming tools with real-time feedback.
- Native apps for mobile devices.
- Prebuilt integration connectors.
- Built-in rule and decision capabilities for capturing business logic.
- Powerful and easy-to-use tools for modeling business workflows.
- Collaboration tools that help teams work efficiently.
- Push-button deployments.

Support for DevOps is increasingly woven into the fabric of enterprise-grade low-code platforms.

Robust security at platform, operational, and application levels.

Built-in regulatory compliance.

Automated testing.

Easy integration with external CI/CD tools.

Application- and system-level monitoring.

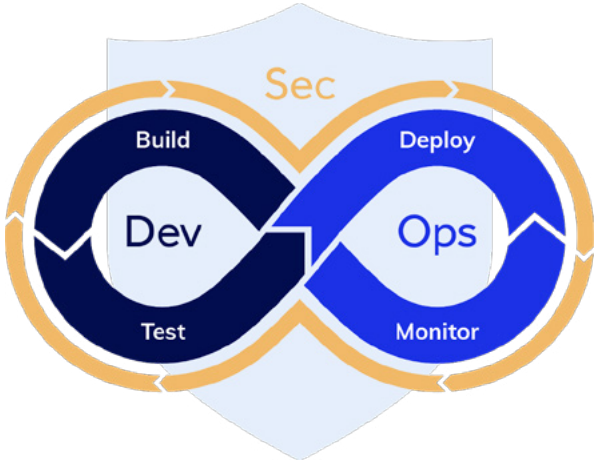
DevOps Principles

- Deliver working software to users sooner to accelerate feedback and time to value.
- Reduce risk by deploying and integrating smaller amounts of functionality more often.
- Plan for operational requirements from the earliest stages of software development.
- Improve software quality by shifting the majority of testing to the build stage.
- Put cross-functional teams to work finding and eliminating process bottlenecks.
- Automate as much of the process as possible.

Low-code across the DevOps cycle.

Enterprise low-code platforms support and enhance DevOps at every stage of software development.

- **Build:** Empower more people to participate in small teams working simultaneously and collaboratively on software development projects.
- **Test:** Discover issues as soon as possible by testing as you build.
- **Deploy:** Accelerate feedback by getting valuable functionality into the hands of users faster.
- **Monitor:** Continuously check the health and quality of your application and environment as you expand functionality, deploy changes, and improve the user experience.



Low-code acceleration.

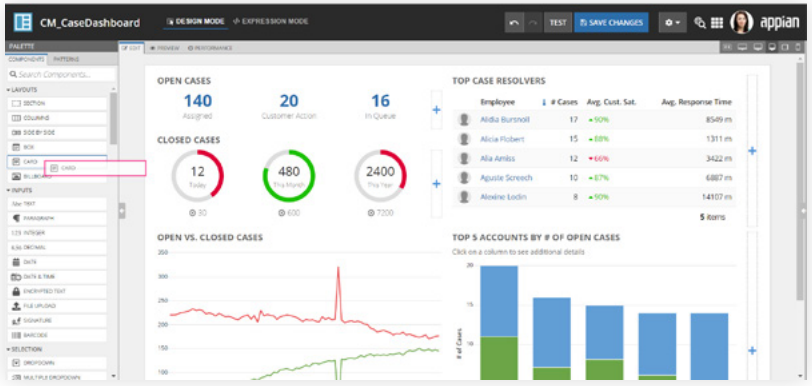
20x faster development of native or mobile web apps.

10x faster development of fully integrated artificial intelligence (AI) apps.

Just 8 weeks for nonprogrammer to build first app (Appian Guarantee).

Low-code advantages for DevOps.

Low-code platforms support agile and DevOps methods by empowering more people to participate in small teams working simultaneously and collaboratively on software development projects. They accelerate development with ready-made integrations that overcome data and system silos. Speed also comes from integrated design-time testing and automating numerous routine development functions. And since low-code platforms make it easy to reuse objects or logic and guarantee backward compatibility on platform upgrade, developers can spend less time maintaining existing applications and more time adding new functionality.



An example of visual programming in Appian. Developers can quickly build out their interfaces with drag-and-drop capabilities.

Case study

Bankhaus von der Heydt

Challenge: One of Europe's oldest banks, Bankhaus von der Heydt was shifting its focus from wealth management and private banking to institutional asset servicing, including blockchain and cryptocurrency. A major impediment to this plan: traditional software development platforms and methods were taking too long to create the software applications needed for new business requirements and dynamic fintech markets. Existing software interfaces were neither user friendly, modern, nor mobile.

Solution and results: With the Appian Low-Code Platform, Bankhaus von der Heydt has linked brokerage and crypto services with its core banking system.

Highlights:

- One day to customize partner solutions.
- Full partner visibility into the integrated Know Your Customer process.
- Ability to offer data APIs and other new digital services to professional customers.

Key features.

- **Visual programming tools** for designing interfaces, queries, process flows, and decisions (e.g., modeling processes by drawing them, using drag-and-drop to assemble fully interactive interface elements).
- **Simplified automation of complex business processes** with low-code business process management that unifies different types of people, data, systems, and technologies (e.g., decision rules, AI-based document processing, robotic process automation, case management).
- **Built-in templates and patterns,** reusable business logic and workflows, and guided object generation that enable fast starts.
- **Prebuilt connectors** that make it simple to integrate with a broad range of data sources, systems, and web services (e.g., Salesforce, SAP, AWS, Microsoft Dynamics, Amazon Alexa, Google Assistant).
- **Guided design recommendations, including access and data security,** and best practices for easier future maintenance that are fit for purpose, model driven, and provided in real time.
- **Automatic generation** of build-once/use-anywhere code that runs natively on the web and virtually any mobile device; seamless processes can span and freely wander across multiple end-user devices while maintaining compliance and accessibility standards.
- **Simplified incorporation of identity authentication and access management services** (e.g., support for all identity providers, including Kerberos; multi-IdP selection; device keychain passwords; adherence to AppConfig Community guidelines and standards for enterprise mobility management control of mobile security, policy, and provisioning; Department of Defense information network access).

Best practices.

- Get key participants and other stakeholders involved early. For example, using low-code interface design tools, business analysts can mock up forms with the exact content, sequence, and look and feel they want. There's complete clarity when they then hand them off to another team member for refinement.

- Use/adapt the low-code platform's application delivery governance model (standards, measurements, process alignment, etc.) or extend your organization's existing governance to low-code development. Organizations should look for a lightweight, effective way to govern their delivery teams. This model provides delivery teams with the standards, measurements, and process alignment needed to develop quality low-code applications quickly and efficiently.

- Agree on Definition of Ready and Definition of Done. All development teams and stakeholders need to align on a Definition of Ready (criteria user stories must meet before development teams start working on them) and a Definition of Done (criteria development work must meet before it is released to production).

- Decide what can be reused to accelerate delivery and boost productivity. At sprint 0, teams should identify business objects that could be reused within the same project or by other projects. Follow good naming conventions that are clearly documented so they can be understood by all teams. Share with other apps on the low-code platform as utilities, with external apps as web APIs.

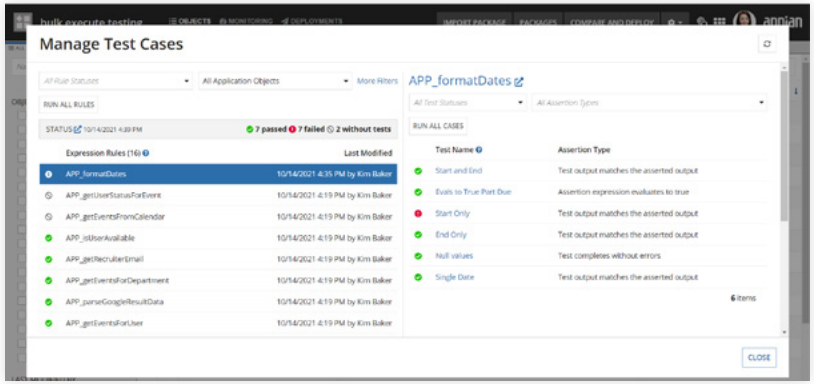
- Clean up as you build. Use comments to describe and explain logic and decisions. Delete unnecessary configurations, rules, variables, and objects that might complicate software maintenance.

- Embed testing into development. The majority of testing should be performed during the build stage as part of the development workflow for each story.

- Implement a peer review process. Encourage developers to get feedback ahead of formal application reviews and other internal governance checkpoints. It's a great way to scale and avoid over-tapping governance resources, and it can help less experienced developers up-level their skills faster. Low-code platform features like collaborative packages facilitate early, frequent peer review.

Low-code advantages for DevOps.

Low-code platforms support the DevOps trend toward earlier, more frequent testing to discover issues as soon as possible. They provide live test-as-you-build tools at the unit level while also automating a full range of tests (unit, UI, performance) for code deployed to test environments and production environments.



Testing in Appian helps you verify changes throughout the development life cycle to catch issues earlier and fix them faster.

Case study

Telus

Challenge: Preparing to launch 5G to more than 10 million subscribers, Canadian telecom company TELUS conducted an internal audit of all of its digital systems and tools. The company's expectation was that 5G would bring an exponential increase in the volume of application development activity. Here's what the audit found: to support rapid growth, TELUS needed a platform that would increase efficiency, scalability, connectivity, and automation.

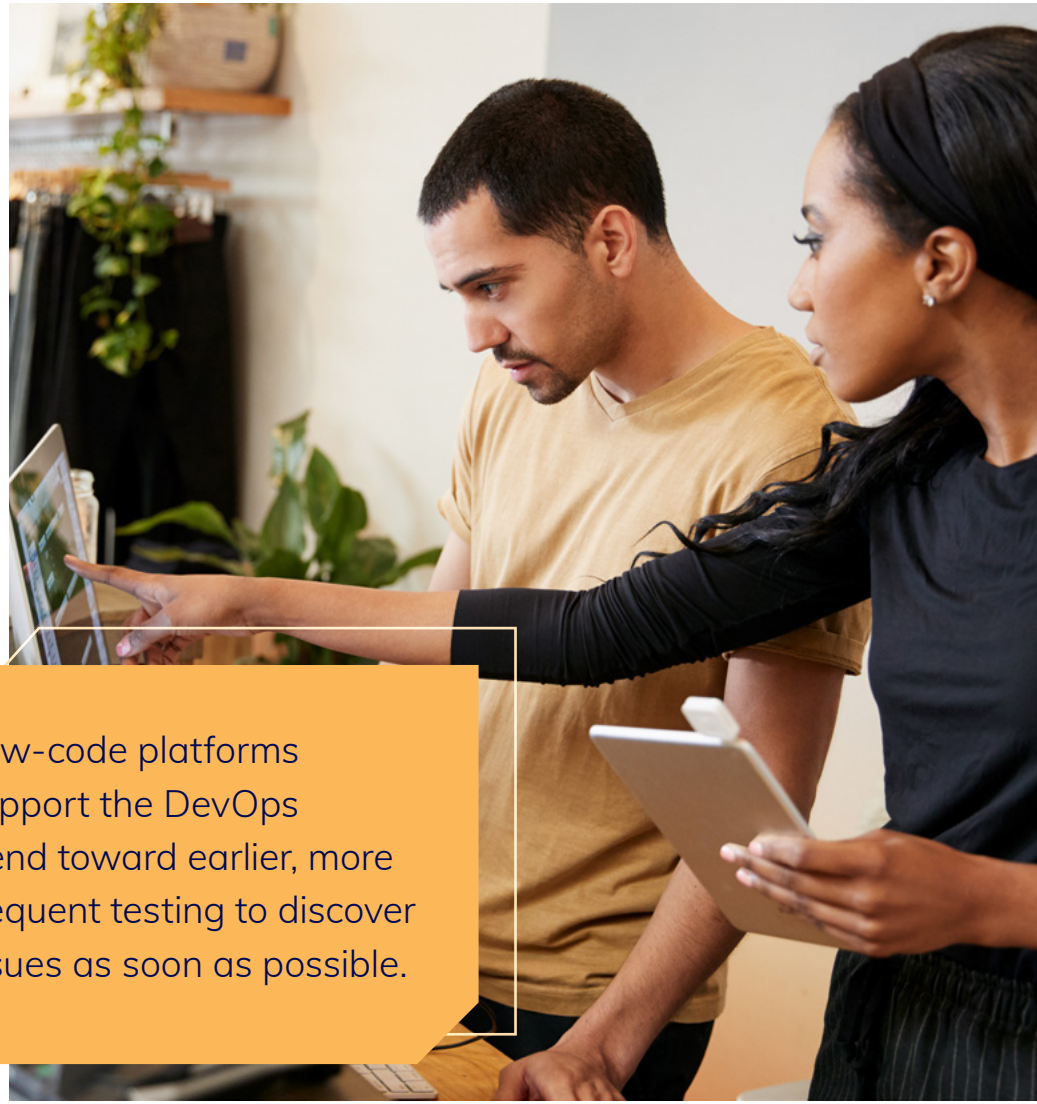
Solution and results: In just 12 weeks, using the Appian Low-Code Platform, TELUS developed an end-to-end workflow management tool to automate and maintain all build activities surrounding the 5G network. Ten thousand business activities now flow through the platform, which is accessible from any device.

Highlights:

- Robotic process automation interfaces with legacy systems and speeds development by 10x.
- Five legacy applications collapsed into a cohesive process, eliminating 20,000 user email notifications per month.
- Eleven legacy and external systems will be integrated upon project completion.

Key features.

- **Unit testing** of the smallest testable pieces of the application, such as object expression rules, to ensure each piece performs as expected under various conditions.
- **UI testing** to see if the application is performing as it should from a user's perspective (responsiveness, navigation, etc.).
- **Performance testing** to see if the application will scale and behave as expected in various production scenarios, including peak usage.
- **Support for third-party integration testing** (e.g., Selenium, Cucumber, FitNesse) and load testing (e.g., Locust).
- **Comprehensive health checks** covering design patterns, performance risks, user experience, and configuration issues.



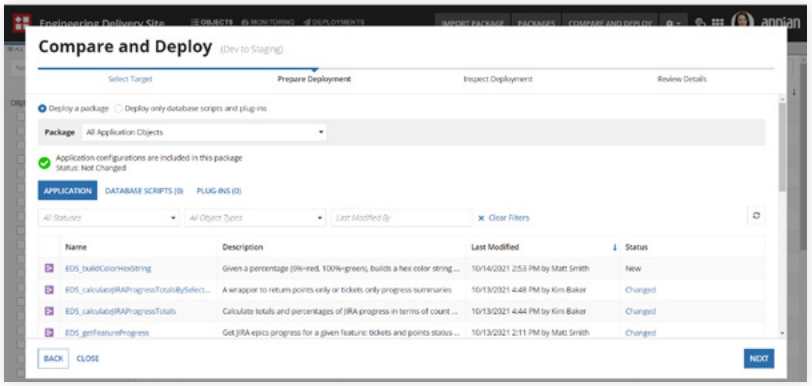
Low-code platforms support the DevOps trend toward earlier, more frequent testing to discover issues as soon as possible.

Best practices.

- Develop a testing strategy. Decide which types of testing you'll need for your project and when/where it's worthwhile to set up automated testing.
- Test in small increments as you build. For example, use live testing inside design objects, such as integrations or decisions. Make sure to test realistic use cases to cover the various ways a rule, decision, or integration could be used.
- Automate common/valuable regression tests. Save test cases as you build to simplify automated regression testing later on. Perform regression testing after deploying changes to the test environment to uncover potentially unintended side effects of changes to reused rules. These regression test cases are also useful as inline documentation to describe expected behavior and results, helping peer reviewers and other developers understand what a rule should do.
- Track quality metrics. Measure progress as you iterate toward higher performance to make sure you're investing in the right areas.
- Invest in root cause analysis. Analyzing the causes of high-priority bugs will help identify potential improvements to your development and quality processes.
- Perform health checks in a production-like environment. Baseline testing should mirror the data volume and concurrent user activity volume you expect in production. Stress testing pushes volume beyond expectations (1.5x, 2x, etc.).
- Give end users the opportunity to perform hands-on testing. Include all relevant user groups (by personae, geographic regions, etc.).

Low-code advantages for DevOps.

Low-code platforms support the DevOps principle of frequently deploying small sets of software changes to test environments and ultimately production and other target environments. This strategy accelerates feedback for development teams by getting valuable functionality into the hands of users faster. It also reduces the risk of introducing complicated object dependencies that can be difficult to resolve late in the cycle.



The Appian platform makes it easy to improve your applications with incremental releases.

Case study

Aviva

Challenge: After 350 years of growth and M&A activity, Aviva, the UK's largest insurance provider, was suffering from application sprawl. Front-line staff had to log in to as many as 22 different systems to resolve just one customer service request. The situation was not only an impediment to the company's efforts to deliver an outstanding customer experience, it was also a constant source of productivity loss and dissatisfaction for employees.

Solution and results: Aviva has sped up its customer service response by 9x. Using the Appian Low-Code Platform, the company unified 22 systems into a single solution for its call center.

Highlights:

- A single screen gives agents a 360-degree view of each customer, including every policy.
- Robotic process automation bots now handle repetitive work, allowing agents to focus on delivering outstanding customer experiences.
- 25% increase in same-day settlements.
- 530% increase in three-day settlements.

Key features.

- **Collaborative tools,** visible to all users with privileges, make it easy to share work in progress, see each others' changes, and obtain assistance or feedback from teammates or members of other teams.
- **Comparison tools** highlight differences between new and previously deployed code and provide insights for making efficient choices about what to deploy.
- **Validation steps packaged with security summary warnings** (e.g., objects with missing precedents, untested rules, inappropriate role permissions) and tools for making on-the-spot fixes.
- **Push-button deployment** to test and production environments, with real-time status and successful completion feedback, per-environment administration controls, and the ability to trigger custom workflows.
- **APIs** for initiating deployments programmatically to an external CI/CD orchestration tool, such as Jenkins; set up once and run the same way for future deployments, saving time and avoiding manual errors.
- **Easy-to-review deployment history** with drill-down detail about the changes made and people involved (e.g., requestor, approver).

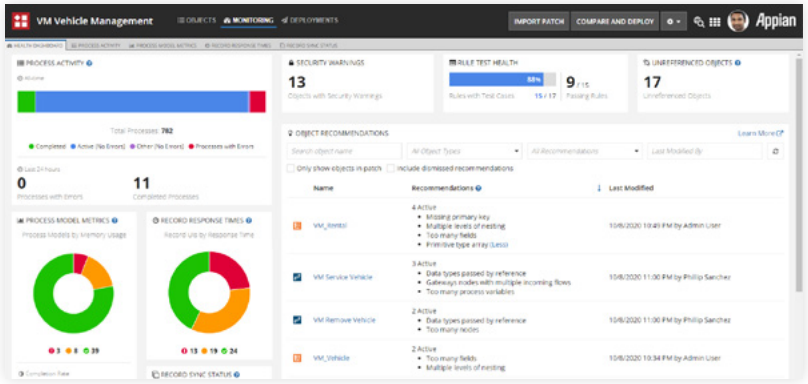
Low-code platforms support the DevOps principle of frequently deploying small sets of software changes to test environments.

Best practices.

- ☑ Decide if you're going to use built-in deployment tools or third-party CI/CD. If you don't already have in-house DevOps expertise, consider starting with the low-code platform's deployment capabilities. You can always switch to a different CI/CD tool later.
- ☑ Determine the people/roles that need to participate in the deployment process. Early on, identify and resolve dependencies, such as required change request approvals, environment-specific integration credentials, and user provisioning.
- ☑ Run a comprehensive health check as part of your pre-deployment checklist. Find any issues with design patterns, performance risks, user experience, and configuration before pushing software to production environments.
- ☑ Establish recovery processes. It's rare for a deployment to fail or have major issues, but you need to be ready just in case. Because low-code development is so fast, it's often better to apply an application hotfix (a new code delivery aimed only at fixing the problem) or "rollback" by rolling forward than to actually revert to a previous delivery.

Low-code advantages for DevOps.

Low-code platforms provide user-friendly ways of continuously checking the health and quality of your application and environment as you expand functionality, deploy changes, and improve the user experience. They provide built-in monitoring tools and also integrate with standards-based third-party tools.



The health dashboard in Appian is a central place to review adherence to best practices and application performance.

Case study

US Air Force

Challenge: Acquisition contract writing was taking too long. To speed it up, the Air Force needed to standardize and unify disparate systems, including contract management for all offices and seven legacy contract writing applications. The solution had to meet high expectations: senior leadership set a goal to shave a collective 100 years from the service's acquisition schedules.

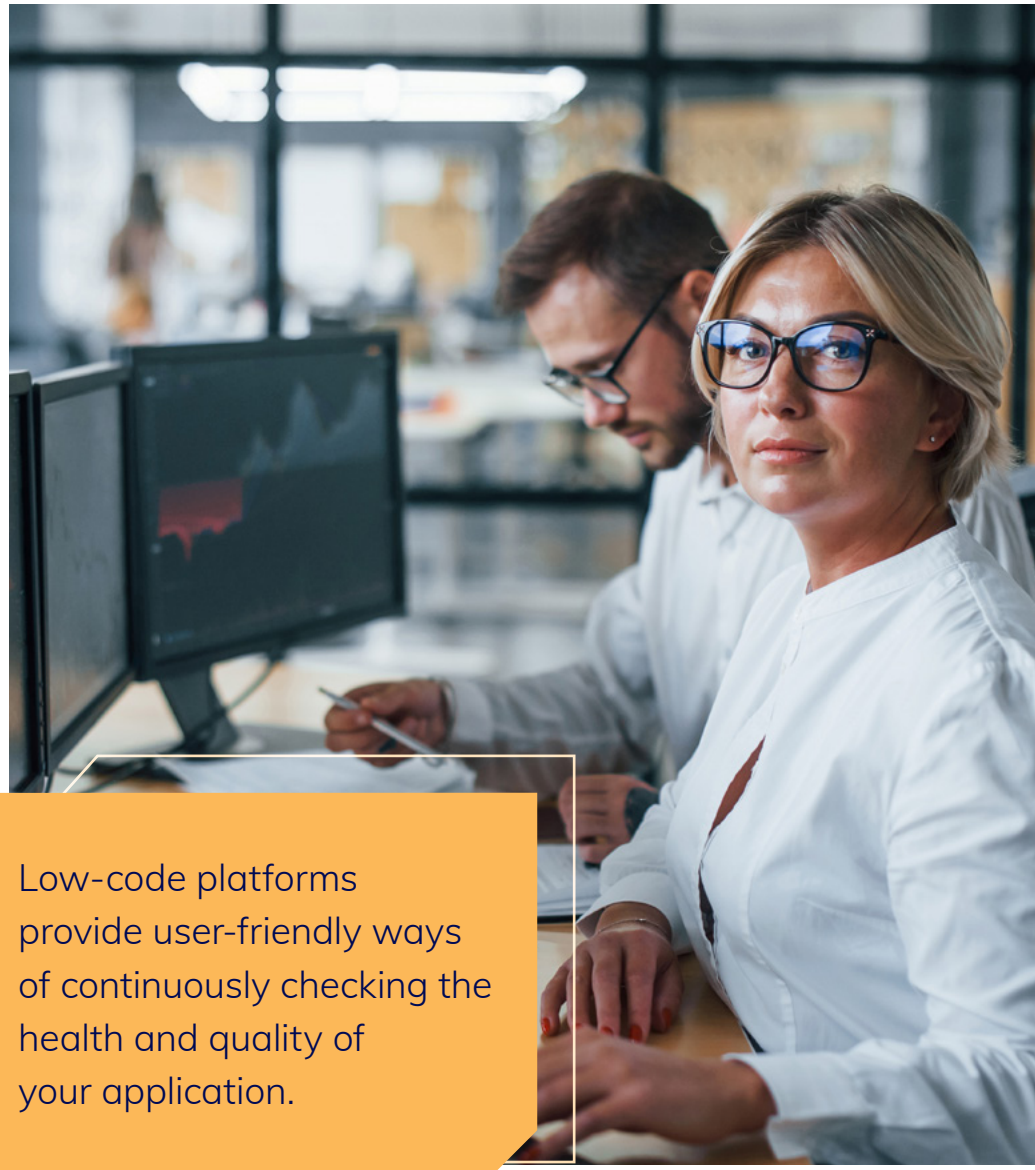
Solution and results: In less than nine months, working with Appian business process management technology and low-code development, the Air Force designed, developed, and deployed its new cloud-based unified Contracting-Information Technology (CON-IT) application.

Highlights:

- Hosted in an Impact Level 4 (IL4) data center, which covers mission-critical data, including Controlled Unclassified Information (CUI), which under law or policy requires protection from unauthorized disclosure.
- Increased efficiency, reduced costs, and enhanced functionality while enabling the service to make statutory changes faster.

Key features.

- **A centralized view of application status** (e.g., process activity and completion, errors, warnings, test coverage).
- **Streamlined auditing**, tracking errors and performance issues, and troubleshooting via easy access to process history.
- **Application performance monitoring** to validate an application is running as expected (e.g., efficient, functional, not producing errors) and providing business value and a good user experience.
- **System monitoring** to track system performance and resource usage.



Low-code platforms provide user-friendly ways of continuously checking the health and quality of your application.

Best practices.

- Set up agreements, processes, and tooling to proactively monitor key functionality in your applications. For instance, you can use built-in business process management and workflow tools to initiate actions, reviews, and follow up.
- Familiarize yourself with key log files. For instance, design error logs capture every expression error that directly impacts a user. These types of errors can be showstoppers, preventing users from completing their work, so it's important to identify and resolve them quickly.
- Evaluate whether to integrate one or more low-code platform logs into your existing security information and event management (SIEM) tools. For example, a good candidate might be streaming performance logs to a tool like Splunk that can aggregate information across various environments and platforms.
- Collect metrics to establish an expectation for how the system should perform under different scenarios. Periodically audit your metrics to ensure thresholds are still useful.
- Configure alerting to push notifications of important events to the appropriate people. Such events would likely include a process failure or exceeding a time/resource usage threshold.
- Schedule health checks on a regular basis. Low-code platforms can provide health dashboards to efficiently surface runtime information, including metrics and key performance indicators, as well as recommendations for continuous improvement. You can also view a history of all health check results on a health dashboard.

The bigger picture: low-code infrastructure and DevOps.

Application development is not the only area where low-code can support DevOps. A best-in-class low-code platform should handle 90% of the infrastructure work, such as platform installation, upgrades, and security patches, so your team can spend time on more important issues. And over time, the platform will continue improving in the background, bringing automatic benefits to your existing applications.

Powerful low-code platforms keep your applications future-proof:

- **Backward compatibility** for platform upgrades gives you confidence that everything you've already built will continue to work, so you can skip regression testing.
- **Automatic performance improvements** are built into the platform—so as the platform gets faster and more efficient, your applications automatically get better.
- **Your preferred hosting environment** is supported, so you continue to see its benefits, whether that's the ease of a managed SaaS offering or the control and flexibility of self-management.
- **Forward compatibility** is included for web browsers and native mobile operating systems, even those that haven't been released yet.
- **Elastic scale** allows you to respond automatically to rapid growth and spikes in demand.
- **Different deployment options are available**, such as a traditional web application, an offline mobile application, or a serverless web application.
- **Built-in security and compliance with standards**, including GxP, PCI, and ISO 27017 and 27018, make it easy to follow changing global and local regulations.

From dysfunctional separation to efficient collaboration.

DevOps first emerged when some software companies' employees realized that the traditional separation between development and IT operations didn't make sense for 21st century businesses. They started to shift the software development culture toward cross-functional methods and teams. Today, it's clear this approach is exponentially more effective.

A similar unification of no-longer-advantageous boundaries is happening now between DevOps and low-code. With the rising popularity of enterprise-grade low-code platforms, we no longer need to keep developers and business experts in separate boxes. More accessible, collaborative development tools mean they can join forces to make greater operational impacts faster.

And for IT developers who still have qualms about collaborating with business experts using low-code tools that until recently were seen as "toys," here's another historical precedent to keep in mind: when the first microcomputers arrived, they too were dismissed as "toys," but now there's not an organization of any size in any sector of the economy that doesn't run on them. Savvy DevOps folks know that low-code is one of the keys to continued progress and success.

To learn more about how low-code and DevOps can accelerate time to value for your organization, take Appian for a test drive with Appian Community Edition, a free personal development environment.

appian

Appian helps organizations build apps and workflows rapidly, with a low-code platform. Combining people, technologies, and data in a single workflow, Appian can help companies maximize their resources and improve business results. Many of the world's largest organizations use Appian applications to improve customer experience, achieve operational excellence, and simplify global risk management and compliance. For more information, visit **appian.com**.

EB-775757534-V4