Businesses and governments have accrued millions of lines of custom software code over the last several decades, including for the mission-critical systems that run their businesses and strategic operations. In many cases, that code was built to be reliable and perform the computational task at hand. **What it was not built to do was change and evolve.**

That poses a challenge to these organizations as they face imperatives to:

- Build new customer experiences that integrate with core systems in order to have meaningful business (or mission) impact
- Manage an ever-growing volume of software that is costly to maintain and represents a security risk if unmanaged in the face of growing vulnerabilities
- Take advantage of the flexibility and innovations—such as serverless computing or machine learning and artificial intelligence—offered by the cloud and open source
This combination of imperatives, which represent opportunities for growth, competitive advantage, and operational efficiency, is why in “Modernize Core Applications with Cloud,” a report published in August 2019, Forrester declared modernizing core systems “the most daunting—but important—work for [Application Development & Delivery] leaders over the next decade.” (In this case, “application” refers to custom software. It is not limited to end-user-facing applications, but can include, for example, integration between two backend systems, or even batch processes running on a mainframe.)

There are challenges and risks to app modernization, however. Traditional approaches have often stalled and failed to deliver results. But despite the challenges and failures, recent innovations in application infrastructure have enabled patterns and practices like microservices and continuous delivery that are leading to better modernization outcomes. When you combine these advances with the business pressures from customer expectations and new competitive threats, it would be reckless not to modernize sooner rather than never.

Why modernize applications now?

Modern, well-architected apps can help organizations achieve a wide range of goals and solve a wide range of problems. Here are three big ones that might seem confined to the IT department, but in fact impact the entire business.

Driving business impact from so-called “digital transformation”

Side projects that tick all the buzzword boxes (you know, that mobile-blockchain-AI-IoT initiative) but can’t integrate with core systems might help an executive land their next promotion, but they provide little if any real business impact. In contrast, modernizing something less sexy—say, the integrations with billing systems or between systems of engagements and systems of record—might allow a telco to run new promotions every week to attract customers away from competitors.

As Forrester notes, “Rigid core systems limit your ability to make dramatic changes to business processes that will power the next wave of innovation.” The lost revenue and higher expenses from the inability to change your software as quickly as the business requires represents the opportunity cost of not modernizing.

Paying down crushing technical debt

According to a Gartner report, “Application Modernization Should Be Business-Centric, Continuous and Multiplatform,” (August 2019) “by 2025 technical debt will continue to compound on top of existing technical debt consuming more than 40 percent of the current IT budget.” At a minimum,
many organizations are running on languages, frameworks, and other application-infrastructure layers that are out of support or approaching the end of support life. This carries risks and keeps teams buried in **maintenance work that delivers no business value.**

The most toxic kind of technical debt is code that is difficult to change. From higher fees incurred by extended support agreements to labor-intensive maintenance windows, code that is hard to change gets costlier over time. Add to this the volume of common vulnerabilities and exposures (CVEs) that require patches to remediate, and organizations with expensive and labor-intensive change processes face a risky proposition: managing costs while also maintaining a good security posture. As **Lance Rochelle, technology leader at Wells Fargo, noted at SpringOne Platform 2018**, “There are thousands upon thousands of CVEs that are reported every month. What we don’t want to do is leave those CVEs out on a platform, out in application code.”

Modernizing software so that it is easier to change reduces maintenance costs and makes it easier to follow security best practices, such as regular patching. In a report titled, “**Overcome Barriers to Agility by Reducing Technical Debt in I&O,**” (July 2019) Gartner predicts that “through 2023, I&O [Infrastructure & Operations] leaders that actively manage and reduce technical debt will achieve at least 50% faster service delivery times to the business.”

**Meeting the architectural requirements for cloud innovation**

In the early days of cloud migration, many organizations took a “lift and shift” approach and, as a result, failed to realize the flexibility and innovation promised by cloud computing. As Anne Thomas of Gartner wrote in “**Hype Cycle for Application Architecture and Development, 2019,**” (August 2019) “Organizations that simply ‘lift and shift’ traditional three-tier applications to cloud-native platforms often find that the applications perform poorly. Most traditional applications include cloud anti-patterns, consume excessive resources, and aren’t able to fail and recover gracefully.”

Meanwhile, many of the innovative cloud offerings require modern architecture to participate. Serverless computing, for example, is a fundamentally event-driven architecture that is difficult to graft onto a monolithic piece of software. Similarly, powerful machine learning services assume an API-centric architecture to pass data between different services. Leveraging these new patterns and services requires modernizing the software that supports the business capability that benefits from these innovations.

Not all software needs to be modernized, but apps should be modernized if long, manual, costly process to change that software hamstring an organization’s ability to:

- Deliver revenue-generating features to customers
- Update any part of the software stack to remove critical vulnerabilities or out-of-support components

- Move the stack to less costly infrastructure

**No 'easy button' for modernization**

In light of all the benefits that modernization can bring, it’s fair to ask why successful modernization is so elusive. One big reason is that the sheer volume of code at most large organizations makes declaring modernization victory nearly impossible. For example, **Air France-KLM is modernizing more than 2,000 applications in its portfolio**, including mission-critical software used for buying tickets and handling baggage. Such systems can’t tolerate availability issues or bugs as they are modernized. It’s a risky proposition, and one that may seem untenable to do at scale. But it simply requires careful planning and an approach that catches issues early.

Gartner claims* that “one key reason for past failures is the IT-centric nature of this approach to modernization,” referring to projects attempting to modernize the entire application estate with a focus on reducing costs. Yet organizations continue to spend billions in **application modernization and migration services**. Unfortunately, simply throwing money at modernization will not achieve any desired outcomes, nor will it fix the problems with legacy approaches.

To succeed, organizations have to take a pragmatic approach to modernization. In particular, they must determine how practices affecting **people, process, and technology** need to change along with the broader modernization effort.

**People: Organizing teams and aligning goals**

Expecting different outcomes from your teams while keeping them in the same organizational structure, with the same skills and the same goals, is ludicrous. At the same time, trying to “boil the ocean” and modernize **everything, all at once**, is doomed to fail. What does work: a willingness to rethink how your people are organized and the goals they’re trying to meet, but with a sharp focus on the work that needs to get done first.

Here are a couple of suggestions to get started.

**Focus on specific business outcomes**
Gartner recommends* that organizations “modernize valuable business capabilities through an end-to-end product life cycle, across all applications, platforms and ecosystems that comprise the complete capability. Avoid focusing modernization efforts on a single application, platform or technology.” This aligns with event-storming methodologies used in domain-driven design (DDD), a process that helped Discover Financial modernize its customer acquisition process to get new features to market 6x faster and achieve higher credit-card application rates.

Dedicate a team to modernization

As the adage goes: If you want to get something done, make it someone’s job. Gartner finds* that a “continuous approach enables application leaders to stop treating modernization as an occasional exception that requires a big project, and instead more proactively manage technical debt throughout the product life cycle.” For example, a large European bank has modernized dozens of apps in its payments division with a modernization team that blends resources from the bank and external partners. Key to its success was creating an identity for the modernization team so that app teams could come to expect a certain experience from its members rather than being distracted by individuals or vendor associations.

Image source: Gartner; “Application Modernization Should Be Business-Centric, Continuous and Multiplatform;” Thomas Klinect and Stefan Van Der Zijden; August 2019
Process: Understand and optimize value streams in the software 'assembly line'

Code isn’t the only thing that needs to get updated in an app modernization effort. The process to get code into production can be as “bloated” as the code itself. Over the years, many organizations have added checks and controls to minimize a perceived risk of going live in production. Doing so has led to processes so cumbersome and arcane that code changes struggle to get out the door.

Here are a few things you can do to prevent your organization’s process of getting code into production from becoming bloated.

Embrace the mindset shift of going fast in order to stay stable

Outages, failures, and breaches continue to occur, despite all the layers of process designed to avoid risk in software delivery. The false assumption here is that checks and controls prevent such things from happening when, in fact, they are inevitable. However, architecting and operating based on the assumption that those events will occur, and optimizing for speed of change, make it easier to recover when they do.

As the annual DORA report has found: “For six years in a row [...] our research has consistently shown that speed and stability are outcomes that enable each other.” DORA has also found a number of practices that contribute to achieving both higher speed and stability. Many are related to continuous delivery practices, as well as psychological safety, clear change processes, and the use of cloud computing in a way that meets NIST’s essential characteristics.

Unwind the layers of controls to understand what is required

Once an organization embraces the idea of reducing friction in order to frequently deploy code changes, the next step is reassessing concerns around risk and compliance. As Len Hardy, senior vice president and chief architect at Northern Trust, has explained, each existing process is an interpretation of a compliance requirement, but people become attached to the process rather than its objective. He describes “unwinding the folklore” to create new interpretations of how to comply with regulations that better support modernization efforts.
Similarly, the U.S. Air Force couldn’t ignore the more than 800 controls in the Department of Defense’s risk-management framework, but it was able to build a platform that addressed 90 percent of them. The result was a reduction in the time from “dev complete” to production from 10 months to one week.

Get modernization into production

Modernization projects that are permanently in proof-of-concept mode don’t deliver actual business results. One reason that projects get stuck in the POC phase is that organizations try to do too much right off the bat. Another reason is that POCs and pet projects rarely include backend integration in their scope. The solution is to focus instead on modernizing systems that turn the cash register and can actually get to production quickly.

This often requires effort and automation to understand every manual step and interaction, from idea to production. In a report titled “Designing and Operating DevOps Workflows to Deploy Containerized Applications With Kubernetes,” (January 2020), Gartner recommends that I&O technical professionals responsible for infrastructure, operations and cloud management “collaborate with application development teams to automate as many parts of the application delivery pipeline as possible in order to deliver business value quickly. At minimum, automate the process for pushing updated containers in production.” By focusing on that path to production The Home Depot has streamlined from 70 interactions per change to more than 500 production deployments per week.

Technology: Microservices, containers, and the Kubernetes ecosystem

While technology is often turned to as a “magical cure,” it is but one part of the app modernization solution. You can’t simply “rub some Kubernetes” on an application and magically modernize it in a way that delivers any sustained benefit. In fact, introducing a new technology like Kubernetes while only having a limited skill set available to manage it can introduce new complexities and challenges.

All that said, the rise of containers and elastic infrastructure-as-a-service have made new application patterns possible where they previously would have been cost-prohibitive. These patterns—largely microservices and, increasingly, serverless—support an application operating model that makes modernization efforts worthwhile. The adoption of these patterns, in turn, reinforces the need for the very technologies that enabled them.

Microservices as a modernization destination
While the real goal of app modernization is (and should be) the ability to quickly change code for business and security purposes, modernization often manifests as going from monoliths to microservices. Microservices are useful for app modernization because microservices, or any way of decoupling code, allow it to change—and fail—with less impact to the rest of the code base. According to 451 Research, microservices are the most critical cloud-native technology or methodology for DevOps implementations.

Now, microservices aren’t a technology; they’re an architectural pattern that isolates services that do one thing, and do it well. But they do favor certain technologies. Microservices introduce the need for API management, because they mostly communicate with REST interfaces. They require software for service discovery, configuration management, and support for the circuit breaker pattern. And they favor containers, which are lightweight, start up quickly, and allow for a consistent runtime across infrastructure. Given the complexity that a microservices architecture can introduce, automation of container orchestration is also very useful.

**Which cloud-native technologies and methodologies are most critical to your organization’s DevOps implementation? (Select all that apply)**

<table>
<thead>
<tr>
<th>Technology</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microservices</td>
<td>49.4%</td>
</tr>
<tr>
<td>Containers</td>
<td>41.2%</td>
</tr>
<tr>
<td>Serverless</td>
<td>34.1%</td>
</tr>
<tr>
<td>Kubernetes</td>
<td>28.9%</td>
</tr>
<tr>
<td>Service mesh</td>
<td>19.1%</td>
</tr>
<tr>
<td>Other (please specify)</td>
<td>0.2%</td>
</tr>
<tr>
<td>None of the above</td>
<td>4.2%</td>
</tr>
</tbody>
</table>

Source: 451 Research Voice of the Enterprise: DevOps H2 2019; Sample size - 425

*Microservices top the list of critical technologies and methodologies*

**Containerize for quick IT ops gains and a consistent microservice runtime**
The spectrum of modernization ranges from the most basic replatforming, to refactoring, to fully re-writing software. Containerizing an application is a form of replatforming, which can accompany heavier refactoring or not. While acknowledging that merely “lifting and shifting” onto the cloud does not yield the desired improvements and cost savings, Forrester has found that containerizing apps can yield some quick wins around portability, security, and operational responsiveness.

This is because packaging applications into containers benefits from many modernization advantages, such as the ability to move a workload between infrastructure and have it run in a consistent fashion. Containerization also allows for packing applications more densely onto machines (virtual or bare metal)—which can lead to cost savings—and simplifying security practices like patching, because instead of trying to patch a running application, a new container image is built and deployed.

Luckily, containerizing applications and doing light re-factoring to replatform an application doesn’t have to take a long time. For example, Liberty Mutual replatformed 10 apps in 10 weeks, which quickly delivered operational gains, as well as providing the institutional knowledge to continue modernizing more of its portfolio.

Homogenize software lifecycle and configuration management with Kubernetes

Kubernetes has emerged as the most popular container-orchestration system. It takes a declarative approach to infrastructure, allowing a user to declare the desired state of a container environment and then rely on Kubernetes to continuously reconcile to that desired state. As part of this declarative approach, Kubernetes offers a consistent taxonomy for configuration beyond the container (e.g., pods, ReplicaSets, etc.). These constructs simplify operations by preventing variation in how applications are packaged and configured. When managing a large number of containerized applications and microservices, this declarative approach provides a useful degree of automation and standardizes lifecycle management across any infrastructure platform.

Final recommendations

Home in on the business capability you’re trying to build

When it comes to app modernization, remember that you aren’t trying to boil the ocean. Some applications are just fine as well-formed monoliths that don’t change very often. Get clarity on the business capability you want, regardless of whether that capability is served by a single application or
spread across a dozen of them. Consider everything that touches the ability to deliver that capability in scope.

**Map the value streams of getting applications to production**

Look at the applications that currently power the business capability you’re looking to improve. Interview those developers and try to understand any of their touchpoints along the path to making changes in production. Your goal is a granular, high-definition picture of what it currently takes to make changes to these applications.

**Start thinking about modernization as an ongoing affair**

You will likely hire someone—possibly a trusted vendor or consulting firm—to help with your modernization efforts. In doing so, be sure to approach the engagement less like an oil change appointment and more like a driving class.

And regardless of whether you engage an external team or not, start thinking about how you would make room in your organization for a team dedicated to ongoing modernization efforts. Such a team must be able to work with software teams to apply modernization best practices and help navigate modernization challenges. Once the modernization team has finished working with a software team, measure the success of their efforts by how much they reduce the **amount of friction needed to change the software** and/or the **cost needed to operate the software**.

**Further reading (or watching or listening)**

- *Modernize Core Applications With Cloud*, Forrester Research, August 2019
- *Application Modernization Should Be Business-Centric, Continuous and Multiplatform*, by Thomas Klinect and Stefan Van Der Zijden of Gartner, Published August 2019
- *Securing Pivotal Cloud Foundry by Regularly Rebuilding*, by Lance Rochelle of Wells Fargo
- *Overcome Barriers to Agility by Reducing Technical Debt in I&O*, by Nathan Hill and Milind Govekar of Gartner, Published July 2019
- *Understanding the Risk Profile of Your Technical Debt*, by Dormain Drewitz of VMware
- *Hype Cycle for Application Architecture and Development, 2019*, by Aashish Gupta and Keith Mann of Gartner, Published August 2019
- *How AirFrance KLM is Modernizing 2,000 Legacy Applications*
Accelerate State of DevOps 2019, by Nicole Forsgren, PhD; Dustin Smith, PhD; and Jez Humble of Google, and Jessie Frazelle

Designing and Operating DevOps Workflows to Deploy Containerized Applications With Kubernetes, by Tony Iams of Gartner, Published January 2020

Solving the Kubernetes Puzzle for Microservices

Modernizing from Mainframe to .NET Core at Travelers Insurance

Crossing the Value Stream: Improving Development with Pivotal and Cloud Foundry

About Intersect

Intersect is a publication designed to guide executives and other enterprise decision-makers through the ever-evolving nexus of business and technology. Through articles, reports, newsletters and more, Intersect provides the insights and information you need to keep up with what’s happening in enterprise IT and how it affects your business.

If you’d like to contribute to Intersect, please email intersect@vmware.com.

View all posts