DevOps Advice and Tips for Beginners
In this e-guide:

If you or your organization is brand new to DevOps it might feel as if you've traveled to a foreign country. Nearly everything you thought you knew about software development, deployment, testing, security and UX is altered when seen through the DevOps lens. DevOps for beginners really isn't for the faint of heart, but with advice, time and patience, you'll get there.

For developers, the move to DevOps requires a mindset shift -- yes, you're going to own some test responsibility -- and new skills, like cutting-edge coding languages such as Groovy or Node.js. Developers headed to that DevOps for beginners land need tolerance, an open mind and a willingness to work with people from a wide variety of backgrounds, including those who are not technical.

Ops pros also face challenges that can feel unsettling. Going down the DevOps for beginners path means letting go of some control while simultaneously taking ownership of new projects neither of which will be easy or feel natural. Testers, who already stare down the barrel of automation even in Agile-only shops, face the biggest reinvention of all. As they set foot on the DevOps for beginners road they'll need to think of themselves as risk managers and scramble to learn the skills to get there.

For user experience and security professionals, DevOps is going to make their already tricky-to-integrate roles even more so. In both cases, being open to working with a much wider group of individuals than before helps to ease the DevOps for beginners transition.

But there is some good news: Start now and you won't be a rank beginner for long.
Your organization wants to jump into DevOps, but it's brand-new to you as a developer. Here are steps both individual coders and aggregated software development teams can take to grease the DevOps slide.

**Learn a scripting language**

Out of the gate, a good first step to enhance the programming team's ability to bring DevOps processes into the software development lifecycle is to learn a powerful scripting language, like Node.js or Groovy. When developers access operational resources, they will be doing it programmatically. The programs that developers will use to access infrastructure-as-a-service resources, AWS instances or even traditional application servers, like WebSphere or WebLogic, are not usually coded in a language like Java or C++. It's important to understand a popular scripting language, know how to write a few lines of code that can authenticate against a resource protected by the operations team and be able to script the control of operational resources. Do all of that while elegantly handling exceptions and error conditions, and your DevOps for developers journey will carry into the operations domain.
In this e-guide

- A comprehensive beginner’s guide to DevOps for developers
- Help ops to begin a DevOps implementation the right way
- A comprehensive beginner’s guide to DevOps for testers
- A DevOps beginner’s guide for user experience professionals
- A complete beginner’s guide to blending DevOps and security

Are you a new DevOps developer? Here’s what you need to know.

1. Do you speak the language? Learn a scripting language like Node or Groovy.
2. Master the CI game—install Jenkins or Concourse locally.
3. Test is what stands between you and disaster. Test is your new BFF.
4. Hit the books. Re-read those Agile/Lean titles on your shelf.

Continuously integrate

In DevOps, it's all about automatic deployments. So, the second step on the DevOps for developers journey is to locally install a CI server, like Jenkins or Concourse. With an installed CI tool on the local workstation, developers can use CI to automate processes such as software packaging, test execution and deployment to local servers. Management, configuration and troubleshooting of a CI server can be the glue that holds the DevOps horse
together, as the tasks a CI server governs often fall on both sides of the DevOps divide. A strong understanding of how to work with a CI server, by both developers and operators, will make the DevOps for developers transition a more seamless one.

Copious and comprehensive test coverage

As the term *DevOps* implies, implementation requires that developers get their hands on systems traditionally under the exclusive purview of operations. But no ops team will easily relinquish control of its resources if it isn't confident in the developers' code. IT ops doesn't want anything to cause irreparable harm to existing systems.

Fortunately, it won't be difficult to reassure the ops side, but it will require devs to do something they don't normally do: properly test software.

All software development teams do testing. That's a given. But DevOps for developers requires more than just paying lip service to the concept. Software must be unit tested, integration tested and regression tested. And unit tests need to be comprehensive. Too often, software developers write a set of lackluster unit tests that serve no other purpose than to appease the audit team when the question of software testing arises. Monotonous tests won't fly in a DevOps shop.
If the operations team is to trust developers to push their buttons and tweak their dials, assurances should be made that every option has been tested and every possible outcome has been evaluated. As development elbows its way deeper into the operations world, there will inevitably be objections that the dev team can't be trusted to take care of the operational side of things. When such complaints arise, the development team must assure all stakeholders that every path has been tested six ways to Sunday, and a history of quality control through applied software testing must exist to lend credence to its case.

**Hit the books, not the exhibition floor**

When Ernest Mueller, director of engineering operations at AlienVault, wanted to make sure his developers were ready for DevOps, he didn't send them to conferences or seminars. He simply reached for the bookshelf.

Mueller, a long-time DevOps devotee, thinks the way to begin the journey is to go back to the beginning of Agile and lean programming. "If you can internalize those things, you have a good beginning," he said.

After that, it's the right choice of books. He feels so strongly about this list he brought copies of these to the company's European headquarters so developers there would have access to the same titles as their counterparts based in Austin, Texas.

Number one on his list is *Continuous Delivery* by David Farley and Jez Humble. "As a developer participant in a build-and-release cycle, you need to understand how everything behaves in order to make applications that are continuously integratable," he said. Then, to get an insight in to the ops side of the world, he suggested *The DevOps Handbook* by Patrick Debois, Humble, Gene Kim and John Willis. And finally, his "hardcore" choice is *Release It!* by Michael Nygard. "This book rigorously enumerates failure modes and the mitigations you can perform. This is like the design patterns books you studied in college, but for production."
Expansive DevOps testing

When it comes to DevOps for developers, teams also need to integrate new forms of testing into their software development lifecycle. Penetration testing, performance testing and security auditing are often tasks performed by operations. But in a world where a developer has the ability to pull the trigger on a deployment, tests must be performed without sending a requisition form to the head of the ops department. Coders not only need to speed up the ferocity of their testing routines, but they must test their systems in ways that may feel alien.

DevOps has the potential to improve software quality, increase the velocity of feature releases and bring great efficiencies to the IT department. Learn a scripting language, familiarize yourself with the nuances of an integration server and bring your test game to a new level of coverage and comprehensiveness. These DevOps for developers steps will make the adoption processes easier and ultimately inevitable.

▼ Next Article
Help ops to begin a DevOps implementation the right way

Stephen Bigelow, Senior Technology Editor

The push is on for organizations to make DevOps implementation, with all of its development and deployment advantages, a priority. But rolling out DevOps for the first time can be a serious challenge, especially for operations groups bogged down by traditional IT silos and practices.

DevOps demands changes on the operations side, and leadership must work to prepare operations for any DevOps transition. It's not just a simple matter of adding a particular tool or system to the existing infrastructure. Success requires a closer partnership between developers and operations staff -- particularly the different groups within operations -- to establish the collaboration, team, tools, practices and infrastructure needed to make a DevOps implementation work.

Prepare to collaborate

Effective communication and collaboration are at the heart of successful DevOps deployments. Developers and operations staff (and even varied groups within IT) must interact well together. Collaboration takes trust, inclusion and empowerment -- attributes that can be difficult and time-consuming to establish. Some organizations may develop interaction by creating interteam focus groups to identify and resolve IT challenges.

For example, 10 development teams are competing for two test environments. One collaborative project might bring developers and operations staff together to better understand
developer needs and allow for access to additional test environments. As another example, developers who need operations to spin up new software components faster can work across silos to identify the time and resource requirements to accelerate response times, while ensuring compliance with company policies and regulatory concerns.

Such initiatives or special project groupings are invaluable to identify silos and bottlenecks within the organization. For a DevOps implementation to be successful, business and IT leaders must make a commitment to mitigate traditional silos and team structures. Even when it's not practical to eradicate IT silos entirely, leadership can and should ease those barriers.

Collaboration can also involve tools such as Microsoft Teams (part of Office 365 Enterprise), Slack, HipChat and Basecamp. The choice depends on the preferences and needs of the particular business.
Prepare the operations team

Team interaction has a critical role to play in DevOps implementation. Traditional IT silos tend to foster a sense of status quo, discouraging the kinds of cross-team support needed to make DevOps work. IT staff -- particularly the leaders -- must commit to learning and mutual support intended to eliminate barriers and solve problems. Finding those flexible and committed professionals early on pays dividends in a DevOps transition process. Consider moving those
individuals to small, cross-disciplinary teams to facilitate pilot DevOps projects. Smaller teams can also work to refine the operations practices and processes used for DevOps and later share that expertise with other IT groups as DevOps implementation expands.

Smaller IT groups will benefit from cross-training. Rather than relying on traditional administration of compute, storage, network and other silos, operations teams benefit if members share roles and tasks. Team members should also cross-train with all the tools that accompany DevOps implementations -- automation, orchestration, code management, configuration management. They should also be well-versed in the back-end systems that bring automation together with developers’ routine work. For example, it’s not uncommon for both developers and operations staff to use a configuration management tool, such as Puppet or Chef.

The operations team must also radically change the way in which they perceive or value data center resources. IT professionals traditionally see a server as a long-term commitment -- a pet to be looked after and protected along with its unique idiosyncrasies. The move to DevOps abandons this paradigm, treating instances more like cattle, to be identically provisioned, managed and ultimately destroyed in high volumes.

Prepare IT automation and orchestration tools

Although DevOps depends heavily on collaboration and communication, operations also experiences an influx of tools -- Ansible, Puppet, Chef, SaltStack, Ubuntu’s Juju, Jenkins, Vagrant and others -- to support new processes. Tools don't guarantee an effective DevOps implementation, but they're essential components of its success. IT staff must be prepared to use them. Some of the most common tools enable automation and orchestration.

DevOps should accelerate software development and deployment, so automation on the operations side is crucial to keep pace with rapid releases on the development side.
Automation reduces the human errors and unnecessary troubleshooting that often accompany manual provisioning processes. Automation also helps ensure that resources get provisioned and managed in accordance with business policies and regulatory constraints, such as security. Tools typically provide robust management capabilities with communication and notification features that facilitate collaboration.

Developers often use these tools, too. Developers can provision and configure IT resources for testing, and a shared tool set offers a valuable frame of reference for operations staff.

Operations personnel should also learn and use some of the tools that developers rely on. Examples include Docker for running containers, Kubernetes for container orchestration, Hadoop for distributed cluster computing and GitHub for software repository management. For example, developers frequently deliver release candidates to a software repository (or repo) and rely on operations staff to pull, compile and deploy the correct version from the repo.

Prepare processes and procedures

DevOps flexibility and speed should not be mistaken for an abdication of control. Operations staff and the greater enterprise must prepare for a DevOps implementation by embracing the appropriate balance between dynamic change and compliance/governance.

This means business and IT leaders must plan for everyday DevOps interactions -- particularly when operations must manage, maintain and protect production systems. Considerations include, but are not limited to, backup schedules; release testing and deployment processes, such as red-blue testing; and rollback procedures. The goal is to facilitate software development, yet preserve the integrity and reliability of production applications.
Many companies lay the foundation for a DevOps implementation with simple, low-priority projects as test beds. Organizations can use those low-priority projects to identify skill gaps, streamline tool integrations and refine the practices or policies that best suit DevOps and business needs.

It is common practice to attach metrics or analytics to DevOps. Operations teams analyze event logs and log correlation to reduce the wait for root cause analysis and service restoration.

The operations team should share these metrics with the developers to help improve the quality of subsequent patches or releases. For example, operations can share logged errors that help developers understand issues that made the release unstable.

Prepare cloud integrations

DevOps does not technically require the cloud, but initiatives often benefit from the speed and flexibility native to public, private and hybrid clouds.

Rather than provision costly local data center resources for software release testing, operations staff turn to the public cloud as an alternative deployment option. With DevOps, it's easy to simply spin up cloud infrastructure as desired, execute the required testing and then destroy the cloud deployment with no further cost or obligation to the business. Similarly, some workloads are more cost-effective to run as a production workload in the cloud than on premises, freeing more costly local resources for mission-critical workloads.

Private clouds give a company the flexibility and services of the cloud when it requires the close ownership, control and transparency of a local data center. The operations team can also construct hybrid cloud infrastructure to connect private and public clouds, which allows for a transition between and integration of local and provider resources.
Success with any cloud technology requires yet another suite of skills and expertise in diverse cloud services, APIs and API integration with automation tools -- to allow the DevOps automation to extend to the cloud. Operations staff must be familiar with architecting the wealth of cloud services needed to deploy large, complex workloads. DevOps teams should evaluate microservices applications and use cloud monitoring data for event and performance monitoring.

**Stick with DevOps**

The transition to DevOps can be a culture shock, and initial attempts to implement a DevOps culture in operations can stumble or fail outright. But the rewards of operational agility and efficiency are often worth the effort. Introduce DevOps changes in phases with small teams, and focus on easy, low-priority projects where IT leaders can manage roadblocks and failure. Learn from the lessons that failure brings, and make the adjustments needed to get operations ready for a DevOps implementation.
In this e-guide

- A comprehensive beginner’s guide to DevOps for developers
- Help ops to begin a DevOps implementation the right way
- A comprehensive beginner’s guide to DevOps for testers
- A DevOps beginner’s guide for user experience professionals
- A complete beginner’s guide to blending DevOps and security

A comprehensive beginner's guide to DevOps for testers

Matt Heusser, Managing Consultant

People brand-new to DevOps often come in with a negative bias, particularly when it comes to software testing. Automation is a large part of DevOps for testers, but it does not eliminate the need for humans. What a move to DevOps does mean, though, is that testers must prepare to rethink this classic role. Testers must become risk managers. The testing work will not go away, but who does which job and when will shift.

How DevOps will change roles

As early as 1999, the advent of extreme programming suggested the end of the testing role in favor of continuous pair programming. At that time, a tester was more of a checklist-follower, and the thought was that automated tests written by a pair could replace that role. Since then, other ideas have evolved. Those include a single tester per developer pair, one test "coach" for a much wider group or a test team that assumes some of the build, merge and CI activities. Another option is to have very few testers but allow a programmer to play the "test" role if there is no tester available. Finally, some software organizations that support a wide variety of platforms (think Android, iOS and web) have turned to crowdsourcing and advanced beta programs to find bugs. In those cases, a few testers organize and oversee the test activities.

In her book, A Practical Guide to Testing in DevOps, Katrina Clokie wrote: "An organization might consider removing a tester if they believe that testing activities can be conducted by
In this e-guide

- **A comprehensive beginner’s guide to DevOps for developers**
- **Help ops to begin a DevOps implementation the right way**
- **A comprehensive beginner’s guide to DevOps for testers**
- **A DevOps beginner’s guide for user experience professionals**
- **A complete beginner’s guide to blending DevOps and security**

other roles within the development team or by the users themselves in the production environment. Though there are no definite rules for determining whether a tester can be removed, I see five areas to consider: context beyond the team, support for quality, team dynamic, measurement and bias."

The common thread in all of these different ideas about software testing is a move toward one-piece flow. Once a story begins, it continues until the work is done, with only breaks for lunch, team meetings and going home at night. To do that, the technical working group needs all skills on deck -- from test to code to the database administrator to a little bit of operations.

When it comes to DevOps for testers, it is all about the value add. Testers can add build responsibilities or be involved in requirements, monitoring and rollback. Where the programmers take an idea from concept to cash, testers can follow along, if they can add value at every step along the way. If testers can contribute to one-piece flow, their role might be needed. As Clokie implied, if not, they might not.

**Before you begin**

To approach the DevOps for testers process for the first time, it’s vital to analyze the current risk, how to manage risk, how threats might change with more frequent delivery and what the ideal end state will look like.

Of course, there is no ideal end state. Not only will the ideal end state change before the team reaches it, but once there, another higher mountain will loom. But an ideal end state gives the
In this e-guide

- A comprehensive beginner’s guide to DevOps for developers
- Help ops to begin a DevOps implementation the right way
- A comprehensive beginner’s guide to DevOps for testers
- A DevOps beginner’s guide for user experience professionals
- A complete beginner’s guide to blending DevOps and security

In this e-guide

- A comprehensive beginner’s guide to DevOps for developers
- Help ops to begin a DevOps implementation the right way
- A comprehensive beginner’s guide to DevOps for testers
- A DevOps beginner’s guide for user experience professionals
- A complete beginner’s guide to blending DevOps and security

If the team wants to ship more often, a regression test or a preflight check might be eliminated or reduced. If you eliminate checks, in turn, it might suggest a change in architecture to enable microreleases rather than working with a massive code base.

Another consideration with DevOps for testers is to determine who will be testing. If the entire team is involved in test, what is the mechanism to hold that together? If programmers will write all of

Page 16 of 27
the automated tooling, what skills will they need? If testers will write tooling, how can they integrate with the rest of the technical staff to enable one-piece flow?

One way to tackle this problem is to make a list of techniques and assign responsibilities appropriately. A better option is to take a further step back: Make a list of possible bugs, including security and performance, and decide how those risks will be managed. Then, address who will perform each activity in the end state.

Finally, consider the tooling and infrastructure that need to be built. Many teams start with the automation, but it is smarter to build the tooling (things like push-button environments) first. The DevOps for testers role at this point is to act as the customer or product owner for the build pipeline and monitoring systems.

Where to begin

At this point in DevOps for testers, the changes may feel overwhelming. Testers can't just switch to DevOps, especially if the build/test/deploy tooling infrastructure is not in place. Instead, identify a single target condition, or stretch goal, and iterate toward that.

The target condition is generally a combination of the largest gap. It is the gap with the most opportunity to add value to the process, but it is also the easiest to do. In other words, start with something that will give the most bang for the buck. Teams rolling out DevOps for testers generally start with infrastructure and tooling, which is fine as long as waiting for builds, deploys or environments is the single largest issue stopping the delivery organization.

Automation and tooling

Many organizations target the GUI for automated tests early in the DevOps process. The team needs to create a large number of tests, like a comprehensive web, to catch every bug.
But if the problem is too many bugs, that’s not a test issue; it’s an engineering problem. A better plan is to focus on tools to accelerate the build and deploy of environments. The build tools make it possible to go from commit to test in five or 10 minutes.

The opportunities for DevOps testers

Exactly what DevOps for testers means and how it will work will be up to the organization. The first quick win is often the ability to create an environment that contains only the changes for a very small piece of work that can be tested. That step represents the desired one-piece flow from dev to test. Testers may also help with requirements at the beginning and act as a sort of junior product owner during the story. These new roles may be sufficient to keep their jobs relevant. Others move into the ancillary roles mentioned above. In any event, a DevOps for testers effort is a chance to get all technical staff to stop thinking of test as a boring checklist activity best done by someone else and instead to think of it as a continuous set of activities designed to reduce risk. It must touch every role.
A comprehensive beginner’s guide to DevOps for developers

Help ops to begin a DevOps implementation the right way

A comprehensive beginner’s guide to DevOps for testers

A DevOps beginner’s guide for user experience professionals

A complete beginner’s guide to blending DevOps and security

In this e-guide

A DevOps beginner's guide for user experience professionals

Jason Grant, Lead Analyst UX/UI

DevOps increases an organization’s ability to deliver applications and services at a greater pace. Customers now expect this continuous delivery of bug fixes and new features, but they also expect seamless experiences. Disappoint them and they can abandon a transaction, rethink the app altogether or trash it on social media. Customer experience has never been more important, but it can be tough for design leaders who are DevOps beginners. Here are seven steps to make it easier to navigate your journey into the DevOps world.

1. Look to the organization

Since this is a cultural shift that includes people, processes and technology, the organization needs to drive the change and encourage collaboration at every level. DevOps beginners should prepare to work right next to their colleagues at every stage in the process.

2. Understand the strategy and success criteria

A user experience (UX) pro must understand what the goals of the application are and what success means. Designers need -- and like -- to move quickly, but in a DevOps world, it’s vital that everyone moves toward the same goal. Do the research necessary to understand the needs and desires of the customer. Then, create a beginning-to-end workflow for each service that can be a reference for designers, developers and operations.
3. Reduce the need to reinvent the wheel

Create a design system. It will accelerate the design process and unite designers and developers around a common visual language. Invest upfront, and build a collection of repeatable components and a guide to describe the design rules and patterns. This will allow designers who are DevOps beginners to be more nimble and developers to integrate components faster.

How to bring UX into DevOps: Follow these six steps

1. Define success
2. Agree on repeatable components everyone can use
3. Work side by side
4. Prioritize features and fixes
5. Build in UX testing at the beginning
6. Create a tight feedback loop, then use it
4. Collaboration is key

Designing alongside application developers allows UX pros to understand technical limitations, while the devs can better understand why decisions were made. This results in less need for thorough documentation, produces functionality that aligns with user needs and requires fewer sacrifices from both designers and developers.

5. Prioritization requires team input

The choice of the next bug, function or service to work on is important, and each member of the team must understand the different constraints and dependencies. Sometimes, new features need to be delayed in order to fix existing features, and other times, the backlog items are good enough and something new should be developed. Having UX provide the voice of the user in this decision process adds another important viewpoint to consider. This is a place even DevOps beginners can really add value.

6. Don't release without testing

Detailed success criteria captured in strategy or design enable integrated UX test monitoring in a DevOps model. Any team member -- even a DevOps beginner -- should be able to read the success criteria and determine if the user’s needs will be addressed by the functionality.
Additionally, heuristic evaluations must become part of the solution validation process to determine if accessibility, general UX rules of thumb and performance metrics are addressed.

7. Always seek feedback

Embrace the opportunity afforded by DevOps to get rapid and reliable feedback from customers, whether through embedded tools, logs and/or customer outreach. A smaller feedback loop allows for constant validation and can help find new bugs, decide on improvements or amend future requirements. Incorporate UX into DevOps, and you will have the unique opportunity to experience the benefits of continuous measurement.

Next Article
A complete beginner's guide to blending DevOps and security

Kevin Beaver,

DevOps can be daunting enough for those brand-new to it. And it becomes even trickier when you begin to add security into the mix. If you want to build out your DevOps and security programs in order to improve your application security initiatives, vulnerability testing and verification have to be baked into your day-to-day processes. The DevOps/DevSecOps approach allows for security to be introduced earlier in the software development lifecycle. Instead of performing security checks once code is fully developed and has gone into production, the process must begin much earlier so both development and security professionals can test (and provide resolution) for software flaws along the way -- even if it's in small code snippets here and there. Modern development environments, static source code analyzers and dynamic application security testing tools make this process much easier. Interactive application security testing (IAST) is an evolving approach that introduces automation and related efficiencies in security testing to ensure both static and dynamic approaches are taken.

The long-term goals of DevOps and security are to prevent vulnerabilities from being introduced upfront and then to catch and resolve them quickly when they do exist. It is critical to reduce the overall attack surface and maintain proper visibility and control at the network and application layers. Visibility and control are the essence of what DevSecOps is all about. You'll get out of application security exactly what you put into it. It has to be treated as a core component of an overall security program, not just something developers, QA professionals or IT operations staff is responsible for. Don't implement DevOps or DevSecOps just for the sake
In this e-guide

- A comprehensive beginner’s guide to DevOps for developers
- Help ops to begin a DevOps implementation the right way
- A comprehensive beginner’s guide to DevOps for testers
- A DevOps beginner’s guide for user experience professionals
- A complete beginner’s guide to blending DevOps and security

of doing it. Collaboration, speed and agility mean nothing if security is not integrated properly and low-quality software is still being produced.

The overall best approach to addressing DevOps and security today requires just a handful of things:

- **Have the right people on board.** The team should include not only development, QA and IT professionals but also those responsible for application security oversight,
internal auditing and compliance. Even higher-level business executives should be a part of these discussions and the decisions that are being made.

- **Outline specific security requirements.** This involves more than just application security basics, like passwords, encryption and so on. Detailed security requirements must include operations-specific issues at layers below the application, including database setups, cloud versus on-premises configurations and integration with existing network security controls, to ensure proper oversight.

- **Know what you have.** Many organizations aren't aware of the extent of the existing code and applications. All it takes to lead to gaps in your application security program is to overlook critical applications and data or, just as bad, acknowledge them and assume they don't need attention. In certain situations, this can be a simple exercise that developers and/or security operations team members can perform quickly. Other times, it might require a more in-depth -- i.e., cross-departmental -- review/audit of applications. Be sure to look at internal, external and third-party code. It all counts when it comes to DevOps and security.

- **Understand how it's at risk.** Manual code reviews and simple vulnerability scans are not enough, especially when they're done after the fact. Proper security assessments that include vulnerability and penetration testing, static source code analysis and perhaps even IAST are essential for finding the flaws so they can be fixed. Keep in mind that, just because security vulnerabilities have been uncovered by the tools or even by manual analysis, it does not mean they
necessarily matter to the business. This is why it's important to get the right people on board and establish reasonable security standards so you'll have something to aim for.

- **Create a plan to properly address the risks, and see it through.** Sometimes, one specific vulnerability that's uncovered can help all DevOps and security team members produce higher-quality applications moving forward. One of the biggest gaps I see in application security programs is people who go through the motions to find and report on security flaws and then do nothing about them. Perhaps, in some cases, it's justified because there is no real risk or there is simply no budget to properly address an issue. Still, you need to see things through -- at least to the point where risks are reasonably mitigated. Otherwise, you've got a liability on your hands that's only going to get worse, especially once an incident or breach occurs.

DevOps and security, or DevSecOps, are not the be-all, end-all solution to all of your challenges. With culture, skill sets, budget and the growing need to define the outcomes of application development and security, you're going to have to do the best that you can with what you've got. I have seen plenty of situations where DevOps and security were well-integrated into the business, but still, numerous high- and critical-rated vulnerabilities were uncovered when the software was given a fresh look. An unbiased perspective and the use of different security tools and known hacking approaches are often what's needed to initiate the cleaning up of risky code bases and get the applications back on track for a smoother DevOps experience.
In this e-guide

- A comprehensive beginner’s guide to DevOps for developers
- Help ops to begin a DevOps implementation the right way
- A comprehensive beginner’s guide to DevOps for testers
- A DevOps beginner’s guide for user experience professionals
- A complete beginner’s guide to blending DevOps and security