Documenting DevOps:

Agile, Automation, and Continuous Documentation

by Chris Riley

Sponsored by:

DevOps.com

Documenting DevOps: Agile, Automation, and Continuous Documentation

Table of Contents

Executive Summary

Productivity is the key driver in the modern software delivery pipeline –DevOps. But the creation of IT documentation – documents explaining systems and their configuration – is not traditionally associated with productivity. Many have come to view the process of documentation a drag on DevOps and similar methodologies, and some consider it counterproductive and unnecessary.

Still, faster-paced development organizations face increasing complexity in applications and infrastructure. If traditional documentation is not going to provide support, something has to fill the gap. The gap of helping organizations know what they have, how it's configured, and how to plan for the future. To meet those needs, a "modern documentation" is beginning to emerge, built on the same pillars of efficiency and automation as the DevOps methodology it supports. Modern documentation is fundamentally different from traditional documentation in that its creation requires almost no human interaction, but the creation and management of the process that builds it requires a substantial amount of planning.

Key findings include:

- Even the definition of documentation is subjective. Despite several decades of implementation, the methods, drivers, and even language of documentation are nearly limitless. Standards such as ITSM help stabilize the look, feel, and process of documentation, but it does not help an organization understand how to use that documentation and integrate it into development.
- The modern software delivery pipeline introduces new complexities in administration. New tooling in the DevOps space has improved application efficiency and quality, but in some cases, this has also reduced the manageability of the environment. This creates new challenges to knowing what you have, how it functions, and how to extend it into the future.
- As the delivery pipeline moves to a continuous stream of releases, classic documentation processes become impossible. Documentation must happen automatically within the process, or not at all.
- In order for documentation to be useful, it needs to be actionable.

Birth of Agile and DevOps

Ever-increasing customer demand and competition has stressed businesses to the point where traditional, monolithic waterfall methodologies no longer make sense for a great many tasks. Requirements change too fast, and the competitive landscape rarely allows for major releases that can be months or years apart. This need for speed and speed and flexibility continues to drive more businesses toward Agile methodologies and the DevOps mindset.

Agile introduced many new practices, from shorter, standup meeting formats to more specific “user stories” functioning as a spec to a far more iterative, responsive development process. Software developers can add new features quickly, so they are more engaged in product roadmaps, and increased speed and frequency of delivery helps attract and retain satisfied customers.

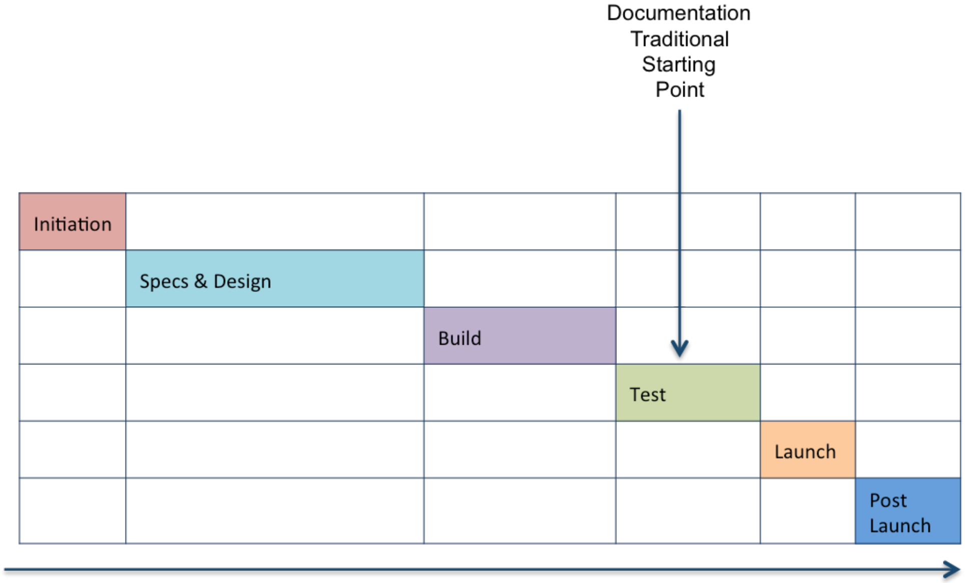
The DevOps framework supports the culture demanded by Agile’s fast-moving environment – a culture that, in theory, unites people, process and tools, creating a continuous stream of application releases. In this world of Continuous Integration and Continuous Delivery, software goes from developer’s hands to production as soon as they press commit.

But speed also creates problems. It’s harder to manage due to a lot of moving parts. And unit, functional, integration, and end-to-end testing have to happen faster as well in order to keep up with the higher volume of changes thrown at them in a shorter amount of time, and other, more manual processes, may not be able to keep pace.

Documentation before DevOps

The very nature of IT operations requires documentation to know where assets are and how they are configured. However, the very documentation that provides this information has often been considered an afterthought. Rather than treating documentation as an integral part of building value, it is often an afterthought, managed only at the beginning and end of any deployment project, with periodic updates of only the most critical components.

Documentation tends to be created in waterfall-based sequential chunks, leaving a large window of time before it is updated and creating significant potential synchronization problems. In waterfall-based documentation, recent entries may reflect the current state of software development, but months-old early entries may be entirely wrong. Because classic software development also followed a waterfall model, it mitigated these weaknesses, as application infrastructure could be set up well in advance of any deployment and did not change frequently.



There are few documentation standards. Traditionally, most ISVs and enterprises have viewed documentation as only an insurance policy, designed to resolve disputes, recover from disasters, or assist in troubleshooting. Even that limited take on documentation has been interpreted in many different ways. The same processes of cataloging components and their configurations has been referred to as “Asset Management,” “Change Control,” “Topology Mapping,” “Configuration Management” and countless others. Since many documents are deliverables requested by specific company divisions (e.g., application support or product marketing), there is very little consistency within organizations, and even less consistency in the broader market. This inhibits internal productivity and limits the ability of businesses to use documentation in creative, growth-supportive ways.

Over time, some standards began to emerge. Information Technology Information Library (ITIL) / IT Service Management (ITSM) introduced an assembly line process to documentation, and also started the trend of orchestration – a big step toward increasing the efficiency of Operations teams. The standard has continued to evolve to a more proactive process, incorporating workflow and steps to consume documentation and update architectures to meet new demands. But the new process focus is on change management only, not software delivery.

> *“IT Service Management (ITSM) is the process-based delivery and management of IT services to employees and customers. It aligns the delivery of IT services with business goals.”*
>
> - Margaret Rouse, Tech Target

Regardless of methodology, documentation is also prone to human error, particularly when those responsible for creating the documentation are also coders, operations staff, or others with time-sensitive tasks competing for attention. Partial and error-filled documentation during rush periods is extremely common, since creating that documentation is directly at odds with revenue-generating efforts.

Documentation’s Impact on Modern Teams

Modern software development is substantially different. Demands for new functionality, ongoing product enhancements, and reduced development costs have pushed developers away from traditional waterfall methodologies toward a more flexible, delivery-centered approach. In the new approach, the software delivery pipeline pushes an active, steady stream of code from delivery to release on a much faster schedule. Therefore there is less opportunity to detail assets and their configurations.

As the pipeline inevitably converges on continuous deployment, traditional documentation is not an option. Even in the intermediate move to sprints (which could be compared to a faster waterfall), IT does not have the time to document changes to infrastructure while preparing for new releases and addressing production issues.

Some would question if there is even a need for documentation in a true continuous deployment model. However, it is important to remember that many of the benefits documentation has brought to waterfall development can also be transferred to a fully automated development and deployment process. Among those benefits are:

- **Training and change control**: Onboarding new team has always been a challenge, and businesses have often addressed this by limiting the scope of the operation they expose to new hires and creating a slower, more expensive "ramp up" process to allow employees to learn over time. A robust, interactive documentation could allow new team members to learn on the fly and reduce their time-to-value, while targeting educational resources in a much more logical manner.

- **Historical data on the performance of the pipeline**: One of the key reasons projects and teams stagnate a lack of a historical understanding. With visibility into where the team has been, where the weak points of an application are and what is going well, teams can resolve stalls more effectively and optimize the development and release process to avoid them in the future.

- **Better understanding of resource spend**: Organizations use multiple private and public clouds at the same time, and DevOps provides the opportunity for a large number of employees to spin up and down environments on their own authority. Resource sprawl is a huge problem that impacts the cost of the operation. Knowing what is being utilized, for what, and for how long is critical to understanding and optimizing infrastructure spending.

- **Building a common language**: The goal of DevOps is to get everyone on the same page. While responsibilities are not heterogeneous, the understanding of the team and its goals must be. Technical teams need to communicate internally, but they must also communicate with business units, as well.

- **Reduced Time to Resolution (TTR)**: Finding the expert who can address a problem is never as easy as it should be. People are out of town, issues come up on off hours, and the experts are already busy with other tasks. Active documentation allows the team to find the answer without needing manual interactions with experts.

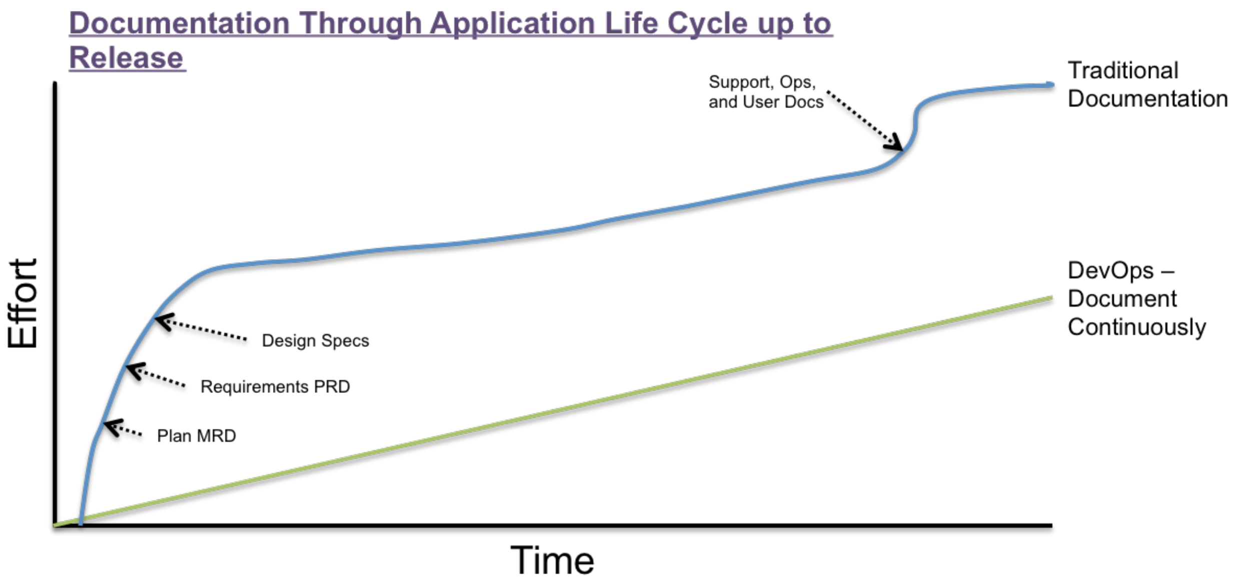
- **Expanded participation in infrastructure**: Relevant, accessible documentation extends knowledge of critical systems throughout the organization, widening the potential number of employees capable of resolving any given issue.

Without making a change, operations will lose control of new systems and their rapid evolution. Developers will not be able to leverage some modern software components that span code and infrastructure. For the team as a whole, the hunt for configuration subject matter experts will become worse as they add new technologies and complexities to the system.

Fortunately, modern processes and tools enable modern documentation. Modern documentation introduces itself somewhat accidently out of new tools, and a metrics driven development team.

Continuous Documentation

While traditional documentation cannot survive the demands of modern development, abandoning documentation altogether equally unviable. An ongoing, automated processes folds modern documentation into the DevOps framework and prevents documentation from becoming a bottleneck to rapid releases. Just as traditional documentation slipstreamed into waterfall software releases, modern documentation does the same for DevOps. It melds to the continuous delivery and deployment of applications as "continuous documentation." Modern documentation becomes an active participant in the software delivery pipeline, which enhances the oversight, metrics, and responsiveness.



In this model, regular processes and systems themselves should be the one source of truth. They will directly provide information about what, and how all systems are operating. And there will no longer be a repository of documents. In fact, other than periodic excerpts from the system, the entire delivery chain is one massive document.

> *"The purpose of a version control repository is to manage changes to source code and other software assets (such as documentation) using a controlled access repository. This provides you with a "single source point" so that all source code is available from one primary location."*
>
> - Paul M. Duval et al, Continuous Integration: Improving Software Quality and Reducing Risk

The contents of continuous documentation are the collection of all data from multiple systems, typically aggregated into a single system for analysis and reporting. The implementation is done with minimal effort, and sometimes on its own, which can pose several challenges. If an organization takes steps into DevOps, it is likely whatever tools they procure will provide some level of analytics. And that alone starts the creation of documentation. But it takes more than accepting the results that are there. As stated above, the documentation is a constant stream of information. And value is obtained by asking specific questions, bringing answer results into reports, and setting up alerts for critical events.

With modern, automated documentation, the burden on the organization is no longer on creating documents, but on planning how to consume them. This will inform the process of setting up the proper style and parameters for generating reports, as well as setting up the responsive systems that will alert the team of changes, deviations and other issues. Documentation will be created continuously with development and infrastructure deployments, rather than by ad-hoc manual efforts.

> Support and documentation are development's customers just as much as end users. Development needs to produce documentation and training that helps customer support understand the applications they're supporting. Support, in turn, needs to provide feedback that helps development understand customers' problems with the applications they're building.
>
> - Jeff Sussna, Continuous Quality

Modern documentation will improve the awareness and abilities of the entire team, expanding everyone's confidence in the system and its setup and allowing more active engagement from everyone. Over time, increased content engagement and accessibility will lead to revisions in the process, which will refine content to better support real-world scenarios in an automated fashion.

Modern documentation will offer fringe benefits, as well, including:

Opportunity and innovation

Development and Operations teams are often focused on pushing the latest clean releases, at the expense of examining the broader chain of software delivery for potential improvements. It is not common for teams to experiment with configuration changes, or cutting edge software components unless necessary to sustain loads or new functionality. Continuous documentation reduces the risk of experimentation. If changes do break a build, continuous allows teams to record and examine the break in almost real-time, while providing precise instructions on how to reverse the changes.

Centralization

Because continuous documentation is to a living thing produced through automation, its potential coverage is dramatically increased. In theory, it can be completely comprehensive, limited only by a lack of planning or misconfigured systems and logging. However, if led by a strong analytics and alerting platform, ongoing tuning and the addition of new data sources is not difficult, leading to a centralized repository that grows in relevance, with all data available to alerting tools and other enterprise applications.

Governance & Security

Because continuous documentation is automated, always on, and not subject to human error, it provides an automatic audit trail. And because it has the ability to correlate all components of the development environment, it can help pinpoint security exploits. Systems acting on the data repository also should permit the creation of policies specifying proper configurations and spawning alerts when configurations are out of compliance

Building in continuous documentation

Continuous documentation is created automatically, and IT's effort is spent on its presentation only. This also means that without deliberate consideration there will be a collection of data from most DevOps tools, and the organization will not know how to leverage it.

The primary challenge of continuous documentation is that it cannot be an afterthought. Unlike traditional documentation, which can be taken up and changed at any point in time, continuous documentation needs to be inserted in monitoring, delivery, and alerting mechanisms when they are first implemented. As a result, documentation readiness must factor into tool choice.

Ideally at the very beginning of designing a modern delivery pipeline, interested parties should ask the following questions.

1) What sorts of questions do we need answered about our system setup and usage?
2) Where is our system of record for infrastructure details?
3) What is our communication standard for discussing system details?

This pre-mortem will help the organization identify expectations of documentation prior to building it into the system. These questions must precede any process adoption, as implementing continuous documentation retroactively is difficult. Once the organization knows what it expects as an output, it can select the system of record for the majority of content. The need for visiting many tools for documentation should also be avoided, so the aggregation of that data to a single reporting and alerting system is ideal.

Once an organization knows the questions, the output, the system, it can standardize on a communication style. The standard might differ depending on the technical depth of the recipient. For example, technical teams may require a query interface, while non-technical workers might require canned reports in PDF format for easy viewing and sharing. Additionally, different audiences might require different sets of language to describe similar phenomena.

After the drivers and value is determined the processes and tools can be implemented to mold to the desired outcome. This will result in a system that automatically builds documentation on the fly with value. The relevant components are:

Continuous Documentation Processes:

- Continuous Integration
- Continuous Delivery (CD)
- Continuous Deployment (CD)
- Production Monitoring

It should be noted that there is a subtle difference between Continuous Delivery and Continuous Deployment, and it is in process only. Delivery takes software to the point just before release, after which a release manager performs a manual deployment following acceptance. Deployment is automatic delivery of code to production. This requires additional measurement, alerting, A/B Testing, and revert mechanisms. Deployment is not for every application.

Continuous Documentation Tools:

- Issue and Project tracking
- Log Analysis
- Orchestration
- Configuration Management
- Deployment Automation
- Testing Automation

Application Performance Monitoring
Alerting

Most often the log analysis platform is the system of record and the alerting tool responsible for making the “documentation” responsive.

Orchestration and configuration management in the modern software delivery pipeline is driven by scripts. Those scripts themselves are documentation. They are versioned in a source repository, and their details contain information about the servers, their network configuration, their installed operating system, the operating systems configuration, and installed packages and their configuration. The scripts themselves can be stored in a log analysis platform or search tool, so the organization has the ability to query configurations at any time. More advanced use cases will adopt anomaly detection, which would allow the examination of script runs, to see if one provision event deployed a configuration that did not match the normal.

> *A set of automated deployment scripts serves as documentation, and it will always be up-to-date and complete, or the deployment will not work.*
>
> - Jez Humble & David Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation

The deployment automation tools manage pipeline states, often containing all of the CI and CD stages and mechanism. The status updates from these systems can be used as triggers in the alerting platform to note timestamps in major stage changes, and their logs can be stored to provide historical data on the number of releases, reverts, or other factors.

Testing automation tools handle the automated testing from Unit, Integration, Functional and End-to-End testing. At all of the stages of Integration and End-to-End testing, documentation is useful, but optional. In other stages, most of the activity in testing is short lived and not relevant to overall process. However, storing data about test runs can help the development and test teams understand quality trends, and making simple calls from test script exceptions can get a failed test back on track automatically, avoiding costly test reruns.

> *A comprehensive automated test suite even provides the most complete and up-to-date form of application documentation, in the form of an executable specification not just of how the system should work, but also of how it actually does work.*
>
> - Jez Humble & David Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation

Application Performance Monitoring (APM) interacts with the application and the alerting layers to measure and report on performance issues. It also serves to store logs in the log analysis platform for historical performance details, which later can be used to correlate performance to infrastructure changes. Real time issues can be spotted when the APM tool takes events and pushes them to an alerting mechanism. This will also allow the team to drill down into the APM system for details, and waste less time tracking down root cause.

This subset of examples of how tool data, their configuration, and their integration into alerting systems allows the continuous documentation to push interesting information to the team. And allows the team to run simple queries to learn about any component of the system.

Use Cases and Examples

While often not named as such, continuous documentation has been leveraged in organizations for several years. Adopters tend to be large enterprises with a mix of both traditional and modern development teams, or software companies whose web application has been in production for 5 or more years.

The users of continuous documentation have streamlined and unified their data collection. They have built analytics, reporting and alerting layers. And they have benefited the entire team with faster TTR, better communication, and greater awareness.

A Cloud Provider:

A niche cloud provider founded in 2009 has identified a need to change. The company offers virtual machines for organizations to use as sandboxed testing environments. Its value is the rapid provisioning of multi-tier environments, which organizations will use for short times, and then let them de-provision automatically.

To offer this, the company owns and manages its own datacenter, with more than 2,000 VMs provisioning at any moment in time. IT is comprised of a traditional networks operation center, and support, which is closely tied with Development and Product Management.

As the application and adoption grew, the provider ran into large challenges. Their application development team needed to understand how their infrastructure was working. Because their code interacted directly with the hypervisor layer, and because their offering was worldwide their infrastructure and VMs were prone to hacking, Development had to manage security policies across the entire inconsistent collection of VMs, regardless of whether they were truly being used.

The provider knew it needed to put all its system data and events in one place to provide better communication and visibility. It built a reporting engine from commercial DevOps tooling, and an alerting platform for on-call support staff. And the entire development team was able to investigate issues that cross both code and infrastructure without interacting with operations directly.

Top 5 US Bank:

Large banks are engines built on top of documentation and governance, but they must also respond decisively to their ever demanding customer base. They have to combine traditional waterfall development teams with consumer facing mobile and web application development teams. The technologies need to interact, but the speed of the traditional team cannot impact the release cycles of the modern one. Mobile banking is highly competitive and fast-moving.

Because of their security and governance requirements, one large bank found it very hard to balance the old with the new. Its approach was to start with strategy, through a new IT unit called Shared Services. This unit then eventually owned all documentation. It reported to operations, but its customers were development. The unit was a facilitator and conduit for Development to IT, and members were able to use continuous documentation to provide active feedback to development teams.

Consumer High-Tech:

One of the largest High tech consumer device manufacturers in the world has no choice but to balance the rapid development with an integration backbone that is slow and immutable.

They want and need to embrace DevOps but are faced with challenges brought about by their size. Their answer to the adoption of DevOps was to build separate DevOps units for each application team. The units consisted of a DevOps manager (responsible for automation and releases) and a small, very focused development team. These units were built for both public and internal applications.

They did have access to some tooling from IT services, but IT services mostly served to regulate the groups.

This structure helped developers move quickly, but sacrificed cross-team communication to do so. The company's biggest challenge was tool overlap. Each team had some amount of autonomy, and each selected and used tools in its own way. Standardization seemed futile, thus the original answer was heavy management layer to police it. That helped reduce the problem but did not stop it, and overhead started leaking back to old release processes.

Eventually the company deployed a massive analytics system, in which there was not only data on tools being used and their configuration, but also a collaboration system that had documentation of systems being used and the problems they solved. Allowing teams to identify an existing tool to their problem. With internal subject matter expertise to help if needed.

eCommerce Retailer:

A top retailer of athletic shoes has recently shifted a good deal of focus to its ecommerce platform. In order to be competitive, they needed to grow their eCommerce site rapidly, while still providing pricing, sizing and availability data to other large eCommerce sites.

The business was a victim of its own successful strategy. It grew too fast, and operations staff found that their tools sprawled out of control. As they did not have a single window into the entire operation, visibility suffered, and their eCommerce call center was the identifier of 90% of all bugs and issues.

Development and Operations decided to implement an “APM plus alerting” system. This allowed them to find and respond to bugs before customers did, and finally brought every employee together “on the same page” in their understanding of systems and processes. The retailer expects to expand to include a full analytics platform so that they can look at the development processes, as well as production. This will further increase awareness and help the business release faster.

Best Practices

The concept of continuous documentation is new, but its practice is not. It has naturally evolved out of the implementation and move to DevOps. That natural evolution, without deliberate oversight has created a number of issues. Steps to avoid these complications include:

1) **Be aware of a tool’s reporting functionality:** Often, organizations do not realize the availability or power of reporting that comes with tools.
2) **Pushing for tool integration:** In order to get all the value of continuous documentation, all variables/assets need to be able to relate to each other. This can be done with manual integration and log design. But pre-built integrations help tremendously. The alerting system is the most critical point of focus, with log analysis and delivery systems providing key inputs.
3) **Involve the entire technical team in alerting and log analysis**: Siloed continuous documentation inhibits the realization of benefits.
4) Build use cases for your DevOps environment: Detail the answers you want from the documentation in advance.
5) **Avoid data memorization**: Data can be paralyzing to teams that have not set boundaries. Paralysis can be avoided by designing queries, reports, and alerting in advance so that manual queries are not necessary. Otherwise, teams will waste time searching and designing ad-hoc reports for small, non-recurring concerns.
6) **Prepare for business’ involvement**: Once the power of continuous integration is unleashed, other teams may become interested. Carve out a place for Product Management or Marketing to do reporting, or a way to regularly

deliver standard reports in advance. This can bolster Operations’ reputation, and also mitigate the distraction requests can cause.

The surprising element of continuous documentation is that it cannot require manual effort, abut this places a larger burden on the planning stage To fully realize the benefits of CD, the team needs to be aware that documentation does exist, and that it’s an entirely different creature than it was before.

Key takeaways

Not only is the modern delivery pipeline moving faster, but the complexity of infrastructure and applications is also increasing with it.

- Traditional documentation was successfully implemented, but only periodically used. It was subjective and prone to human error, and required waterfall development.
- Traditional documentation will remain for some time, as waterfall development remains. This is particularly true for business applications like Enterprise Content Management (ECM), and Enterprise Resource Planning (ERP). The nature of this type of development and its reliance on third party systems out of their control means traditional documentation may remain the only option.
- DevOps brings processes and tools that introduce new complexities and complicate administration. But it also provides more data and new ways to look at documentation.
- For all other web and mobile applications, documentation has to follow the shift into faster development, greater software quality and a stream of releases.
- Traditional documentation is simply not possible as delivery pipelines move to DevOps.
- Documentation in modern systems has to be responsive and interactive compared to the reactive approach employed in waterfall development.
- The need for documentation has not gone away, but its value has changed.

About Chris Riley

Chris Riley is a coder turned market analyst. He is co-founder of Fixate IO a market analysis firm. Riley has spent 15 years helping organizations transition from traditional development practices to a modern set of culture, processes and tooling. He is an O'Reilly author, regular speaker, and subject matter expert in the areas of DevOps strategy and culture and Enterprise Content Management. Chris believes the biggest challenges faced in the tech market is not tools, but rather people and planning.

Throughout Chris's career he has crossed the roles of marketing, product management, and engineering to gain a unique perspective of how the deeply technical is used to solve real-world problems. By working with both early adopters and late, he has watched technologies mature from rough solutions to essential and transparent. In addition to spending his time understanding the market he helps ISVs selling B2D and practitioner of DevOps Strategy. He is interested in machine-learning, and the intersection of BigData and Information Management.

About DevOps.com

Launched in 2014, DevOps.com has quickly established itself as an indispensable resource for DevOps education and community building. We make it our mission to cover all aspects of DevOps—philosophy, tools, business impact, best practices and more.

Our site is the largest collection of original content related to DevOps on the web and one of the top result for DevOps-related search terms. Our content includes in-depth features, bylined articles, blog posts and breaking news about the topics that resonate with IT readers interested in DevOps: teamwork through improved IT culture, continuous integration, automated deployment, agile development and infrastructure-as-code among them.