New Relic

White Paper

Preparing for the Next Phase of DevOps

3 Ways to Conquer the Complexity of Modern Cloud Environments

Introduction: A Tsunami of Complexity

Here at New Relic, we've been helping organizations do DevOps right for years. During that time, we've shared our own experiences with DevOps, continuously delivered new capabilities that help teams optimize their use of DevOps and cloud native technologies, and led the charge for data-driven approaches and best practices to measure and track DevOps success.

Now that DevOps has gone mainstream and organizations that embrace it are seeing repeated and growing success, it's time to acknowledge the emergence of a serious consequence of DevOps and cloud native technologies and techniques: the growing tsunami of complexity beginning to bear down on organizations as they adopt microservices, serverless, containers, and other approaches and technologies in the cloud.

This is not to say that DevOps and cloud native approaches are bad; on the contrary, the benefits of both far outweigh the cost of complexity. The point is that complexity creates risk that organizations must take steps to mitigate before it negatively impacts IT and business outcomes. DevOps-committed companies that have not yet acquired the skills, best practices, and tools for this next phase of modern application development will find it extremely difficult to continue meeting their application performance and availability goals.

In this white paper, we'll look at where we are with DevOps today, why complexity is increasing, and how your organization can start cutting the complexity sooner rather than later.

Need more background on DevOps?

While this paper assumes a basic understanding of DevOps, here are some additional resources that can be helpful no matter where you are in your DevOps journey:

- *What Is DevOps?*
- *DevOps Done Right: Best Practices to Knock Down Barriers to Success*
- *DevOps Without Measurement Is a Fail*

Long Live DevOps!

DevOps principles, culture, processes, and tools help today's organizations address the challenges they face in a software-driven world—the push to transform digitally, innovate, accelerate time to market, adapt quickly to change, and deliver flawless customer experiences.

In its State of DevOps 2019 report, DevOps Research and Assessment (DORA) says, "Many analysts are reporting the industry has 'crossed the chasm' with regards to DevOps and technology transformation, and our analysis this year confirms these observations. Industry velocity is increasing and speed and stability are both possible, with shifts to cloud technologies fueling this acceleration."

While DORA's report calls out cloud as fueling speed and stability, its analysis shows that the use of cloud computing is also predictive of software delivery performance and availability. The high-

The Role of Roles in DevOps

Originally, roles and responsibilities within DevOps were more distinct:

- ***Dev:** Engineers own their applications end-to-end, including infrastructure, provisioning, capacity, and more.*
- ***Ops:** Engineers use dev-like tools to manage their infrastructure, including monitoring and automation/scripting.*

Today, roles are more amorphous as the DevOps goal has evolved to be "remove all organizational barriers to shipping enhancements to customers; accelerate with tooling and automation."

est performing teams in the DORA report were 24 times more likely than low performers to execute on all five capabilities of cloud computing as defined by the National Institute of Standards and Technology (NIST).

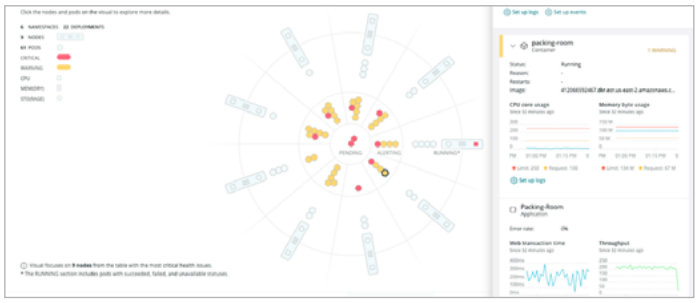
For instance, cloud computing on AWS enables engineers to create self-service methods for provisioning infrastructure through AWS Service Catalog. Developers can quickly try new things, fail fast, and get new products to market faster—without having to wait for services to be provisioned for them. Fully managed services help teams take advantage of AWS resources, while automation in the form of AWS Developer Tools helps them do so more quickly and efficiently.

It can be safely said that DevOps done right delivers resounding success in both IT and business outcomes—but wait, what are those dark clouds building on the horizon? Is a backlash imminent?

The Price of Success

To gain the speed, agility, resiliency, scalability, and other core benefits of modern application development, DevOps teams are starting to pay a price—and that price is complexity. Not that monolithic applications aren't complex (as well as brittle, unwieldy, difficult to scale, and other characteristics that generally slow innovation to a crawl) but modern cloud native environments are creating a different type of complexity for which some teams are largely unprepared.

Teams modernize by breaking up their monolithic applications into smaller microservices that can be independently owned and deployed. Microservices are easier to maintain and can be reused to accelerate development. At the same time, teams also make their applications more portable and scalable by adopting modern technologies such as containers (Docker) and container orchestration (Kubernetes) using Amazon Elastic Container Service (ECS) and Amazon Elastic Kubernetes Service (EKS) or serverless using AWS Lambda.



Kubernetes cluster explorer displaying KPIs for a specific pod

All of this leads to a growing number of microservices, running across hundreds of containers or operating as serverless functions. Multiply that by daily or hourly deployments for many different services at a time. Add in fluid infrastructure and automation such as containers, load balancing, and autoscaling.

The result is a swarm of moving parts that grows and changes moment by moment. Where should you look to find the root cause of an issue? It may be a problem that occurs in a container that only exists for a short amount of time under certain circumstances.

It's not just the moving parts that create complexity, but the amount of operational noise that's generated by each of the parts. Every piece of infrastructure and every microservice can create data about metrics, events, logs, and traces (we call this collection of telemetry data M.E.L.T. and you can read more about it in this introduction to the topic). Where should you look to understand the true origin of a performance issue?

Taming the complexity and tuning out the noise without losing actionable insight should be the next step in your DevOps journey.

Key Principles for the Next Stage of DevOps

Up until now, teams have often focused the majority of their efforts on the Dev part of DevOps—honing their abilities to deliver software faster, more frequently, with greater reliability and availability, and with fewer errors.

Now, it's time to focus on the Ops part of DevOps, where two key principles will serve you well as you move to overcome the complexity of modern cloud native applications and environments:

1. Focus on getting the culture right

For DevOps success, culture has been, and will continue to be, critical. Culture is about how teams are formed, how team members work together collaboratively, and how they share responsibility for the software that's being built. Transparency and communication within and across teams helps develop shared understanding in support of DevOps' goals.

> *"The new microservices way of building software optimizes for rapid and large-scale change, but there is a cost, and that cost is complexity. With powerful technology like Kubernetes, real risks can result if organizations aren't ready for the complexity that comes with distributed systems. I implore customers to take serious steps to get ready for modern systems and use modern tools to work with them."*
>
> **Kelly Looney, SI Practice Lead, DevOps, Amazon Web Services**

However, one core characteristic—trust—is sometimes overlooked. Trust is perhaps the most critical cultural component for DevOps teams to accept and master. Leaders must trust their teams, teams must trust other teams, and team members must trust each other.

Without trust, the other essential cultural characteristics of autonomy and responsibility can't function. When you give teams the autonomy and responsibility to own what they ship, and the tools they need to do so correctly (such as Amazon Elastic Beanstalk, AWS Lambda, and AWS CloudFormation), you have to trust that they will do the right thing.

2. Instrument and measure everything

Measurement is one of the five pillars of the CALMS framework (Culture, Automation, Lean, Measurement, Sharing) coined by DevOps expert Jez Humble. Comprehensive and contextual data is critical to the successful functioning of DevOps as complexity grows. Getting a complete view of your architecture, no matter how ephemeral, is key to coping with this complexity. Having the right data will tell you what's working well, what's not, and where to focus your team.

Data reduces finger pointing by removing emotion from the decision-making process and instead fosters a culture of collaboration and empathy. It gives everyone within DevOps teams a common language across skills, experience, and roles.

Netflix Adopts an Operate-What-You-Build Model

In a post on its Tech Blog, Netflix tells the story of how and why it decided to move to a shared ownership approach, where the team that builds the software is also responsible for operating and supporting it.

3 Ways to Conquer Complexity and Continuously Improve Outcomes

As you continue to mature and evolve your DevOps efforts and prepare for the next stage of complexity, there are three competencies your team should strive to master as it hones its Ops performance:

1. Observability

The antidote to complexity in modern cloud native environments is observability. When you instrument everything and use that telemetry data to form a fundamental working knowledge of the relationships and dependencies within your system as well as its performance and health, you're practicing observability.

Observability gives teams the ability to see a connected, contextual view of all of their performance data in one place, in real time, to pinpoint issues faster, understand what caused an issue, and ultimately deliver excellent customer experiences. It helps teams make informed decisions about changes being made. Does the service being updated by the team communicate with another service that the change might break?

To learn more about observability, read The Age of Observability: Why the Future Is Open, Connected, and Programmable

2. Fast feedback loops

Succeeding at failing, in the form of "failing fast," is a desired capability for DevOps. Author, researcher, and DevOps expert Gene Kim names amplified feedback loops as one of the Three Ways that frame the processes, procedures, and practices of DevOps. He cites the importance of creating right-to-left feedback loops, with the goal of shortening and amplifying them so corrections can be made continually.

Fast feedback loops combine the principle of trust with tools and processes that enable your teams to:

- **Encourage experimentation:** Promote a culture of innovation and bold ideas (along with failures) to help accelerate time to market and adoption of new technologies while improving business outcomes.
- **Deploy frequently:** Encourage small, frequent deployments of micro features and bug fixes, and measure the success of each deployment. Learn more about instrumenting your pipeline in this webinar.
- **Accept risk:** Small, low-impact failures are okay when you recognize that they are the cost of shipping faster. Accept that there will be more frequent, but smaller, production issues, with the understanding that consistency is more important than perfection. The production issues that do occur will be, in aggregate, vastly less impactful than large outages.
- **Learn from every mistake:** After resolving an incident, accurate and thorough documentation helps your team learn from the incident and take preventive measures to keep it from occurring again. The preferable way to accomplish this involves holding a blameless retrospective that focuses on constructive learning and improvement, not punishment or blame.
- **Use release dashboards:** For every feature release, create a "feature dashboard" that tracks key performance indicators specific to that feature. Your dashboard should track feature deployment as well as usage of AWS services by those features to understand the impact on the application.

KPIs related to specific feature deployment

3. Continuous improvement

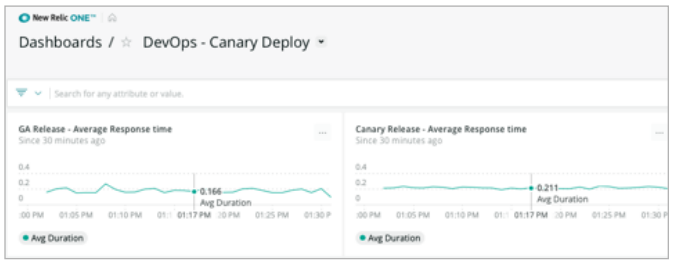
To progress from DevOps adoption to becoming a high-performing DevOps organization that mitigates complexity and optimizes outcomes requires a culture and process dedicated to continuous improvement. That culture of continuous improvement is clearly a key reason the number of elite performers in the DORA State of DevOps report tripled compared to the previous year.

Start by correlating performance to business outcomes. This will help you show how infrastructure and application changes ultimately impact the customer experience and business metrics. If the results can't be measurably seen by the business, it's hard to argue the deployment was successful.

Other ways of continuing to improve DevOps performance despite complexity include using advanced release strategies to help you:

- **Make side effect-free sandboxes available:** Part of encouraging experimentation is creating environments in which there are no consequences of taking risks and trying new approaches and technologies.
- **Turn on feature flags:** When a release happens, the feature flag is flipped. It can be unflipped if things are not going as expected.
- **Communicate with consumers of your services:** Each business unit should behave as production-level consumers of each other. This means a period of maintaining new and previous versions simultaneously. While this results in more overhead, the benefit is low-risk, headache-free releases.
- **Take a rollout approach:** For riskier features, use a rollout across your user base. The feature flag can be a meter instead of a toggle (e.g., 0% of users → 5% → 10% → 25% → 50% → 70% → 100%).

- **Use canary testing:** Canary testing refers to the practice of releasing a code change to a subset of users and then looking at how that code performs relative to the old code that the majority of users are still running. Canary servers or containers run the new code and when new users arrive, a subset of them are diverted to the canary hosts. New Relic can help you monitor and measure whether the new code is working correctly.



Canary deploy KPIs

- **Adopt blue/green deployments:** To avoid downtime at go-live and to accelerate rollback if something goes seriously wrong, consider using a blue-green deployment approach. With blue-green deploys, you have two nearly identical production environments: one green and one blue. For instance, the blue environment is your current production environment. The green environment is where you conduct final testing, and, based on that success, you start routing users to that environment as you go live. The blue environment is then idle but gives you a rapid way to roll back if needed.
- **Take advantage of dark launching:** Dark launching is similar to canary testing but rather than launching a new feature to assess how that code performs versus the old, you instead release it to a small set of users to assess the response of users to the new feature. Usually, these users aren't aware of the new feature and you don't highlight it to them, hence the term. For example, to achieve a dark launch of a new UI on your website, launch it on an alternate domain or URL or include it on a hidden IFrame on your homepage.

No Finger Pointing

The purpose of a blameless retrospective is for all parties to reach an understanding of the situation that led to an incident, the resulting incident response, and gaps or points for process improvement—all with the goal of avoiding, or at least mitigating, a recurrence of the problem.

Learn how and why your teams should hold blameless retrospectives in this blog post.

Conclusion

As more and more organizations "cross the chasm" with DevOps, taming complexity and focusing on improving the Ops part of DevOps will become essential. By building observability, using fast feedback loops, and continuously improving, you can elevate your team to the high-performing or elite-performer category.

New Relic can help you cut through the complexity of modern application environments. Our tight integrations with AWS help you manage the complexities of your AWS environment, including EC2, Lambda, and Kubernetes deployments.

To learn more, visit newrelic.com/devops.