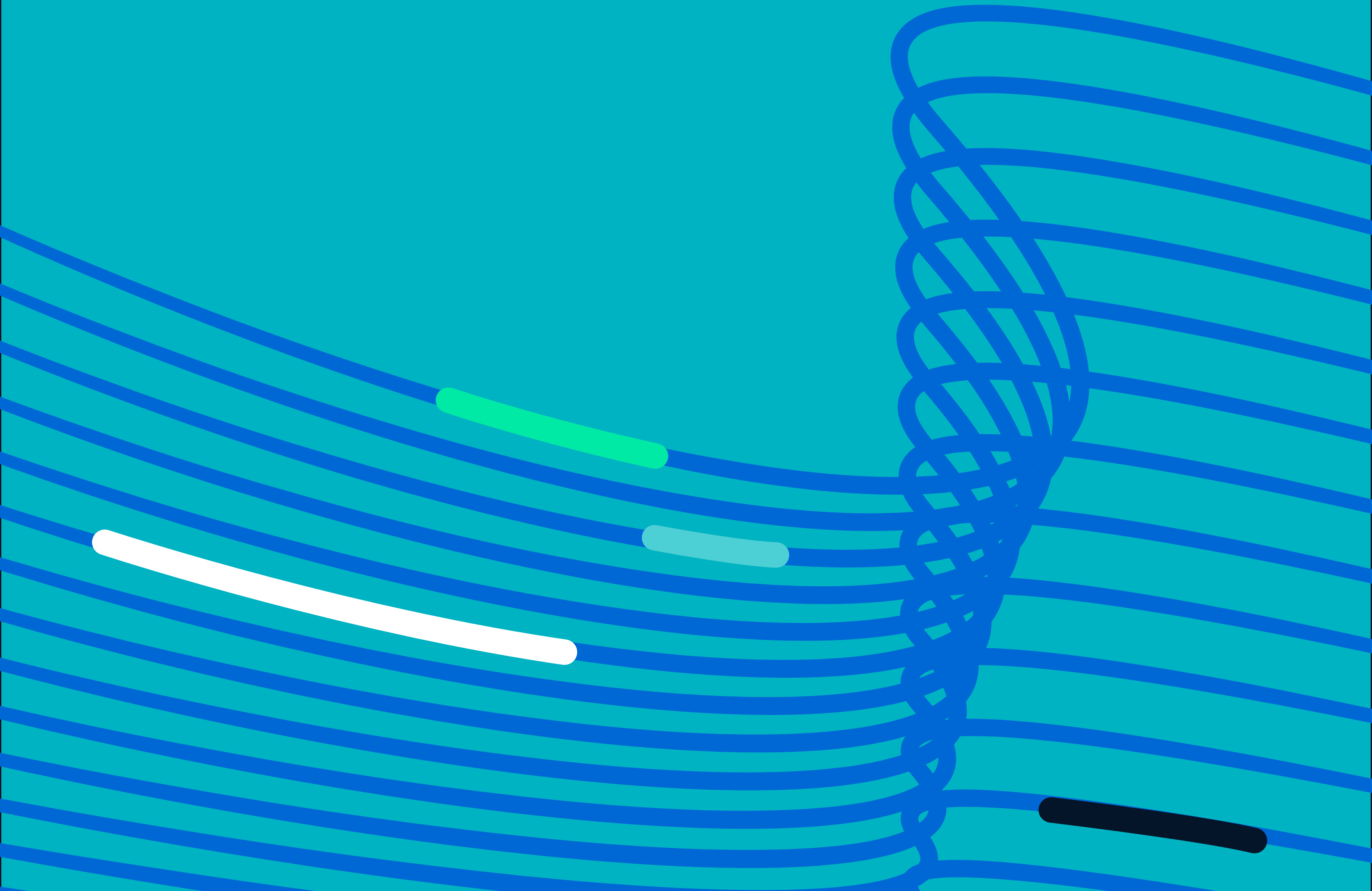
New Relic®

White Paper

Modernizing with AWS:

Getting Started with
‘Observability-Driven’ Replatforming

Part 2 of our 3-part series on legacy app modernization in the cloud



Introduction

Fundamentally, organizations modernize existing applications by moving them to Amazon Web Services (AWS) in order to optimize everything from performance to reliability to customer experience and more. For the business, the goals and measurable outcomes are typically some combination of the following:

- Improve the customer experience with better reliability and performance
- Reduce operational cost and effort
- Accelerate time-to-market for new features
- Develop and enhance technical and business agility

While modernizing existing applications can take several different forms, this white paper focuses on the replatforming approach, also known as component modernization. For an introduction to the six approaches to application modernization on AWS (retire, repurchase, retain, rehost, replatform, and refactor), read our ebook “The Enterprise Guide to Continuous Application Modernization.”

While rehosting involves moving an existing application to AWS without making changes to the business logic, replatforming requires something more—namely, optimizing the application by modernizing one or more of its components. Components are essentially an endpoint, something that the application interacts with via a standardized API, such as a web server, relational database, or messaging system.

Somewhat riskier than rehosting, yet less complex than refactoring, replatforming can be categorized as a modernization “middle road” when it comes to risk and complexity, because modernizing components usually involves some level of changes to the code. Nonetheless, how do you know when the replatforming approach is the right decision? Which applications are best suited for this approach? And, more important, how can you further reduce project risk while streamlining the effort and maximizing all the benefits of a replatformed application on AWS?

Here’s a hint: observability and best practices. This white paper presents the knowledge and experience of New Relic working closely with AWS and explores how you can make informed decisions to optimize your efforts and outcomes.

What Is Replatforming and What Are the Benefits?

While rehosting is a lift-and-shift approach with no code changes whatsoever, moving your application essentially as-is to AWS, replatforming takes this approach a little further. Based on observations about the application, you make an informed decision to change one or more components of the application, and in doing so, achieve additional optimizations.

To be clear, changing a component does not change the business logic. It does, however, require changes to the configuration, set up, tear down, and administrative practices. For instance, coding for a data connection or an API could change, but the change doesn't impact the business logic of the application. Any component change you make as part of a replatforming project should follow this rule. On the other hand, changing an application from using a relational database to an object store requires business logic changes, which makes it a refactoring project, not a replatforming one, because it fundamentally changes how your application behaves.

Let's review a few examples of replatforming on AWS:

- Changing to a managed relational database service such as Amazon Relational Database Service (RDS) or moving to a metered database such as Amazon Aurora
- Using AWS Elastic Beanstalk to deploy your application
- Moving from self-managed Apache Kafka (on-premises, AWS, or some other SaaS Kafka) to Amazon Managed Streaming for Apache Kafka (MSK)
- Leveraging AWS Certificate Manager instead of running your own certificate infrastructure
- Moving from a self-managed Kubernetes environment to Amazon Elastic Kubernetes Service (EKS)

What these examples have in common is that the code is interacting with components through a fairly standardized API. While the API will likely need to be tweaked, the business logic should not be impacted.

Why undertake a replatforming project? The benefits will depend on which component you're changing, but they can and should include outcomes such as:

- Improved scalability to accommodate business growth
- Improved reliability and performance for a better customer experience
- Reduced costs for software licensing and resource usage
- Reduction of the total security surface area of your application
- Reduction in management efforts and associated time and costs
- Improved ability to make informed decisions

What Are the Risks and How Can You Minimize Them?

In any project, understanding the risks involved is the first step in minimizing them. While the risks for this type of modernization are relatively low, there are certain ones you might encounter when you replatform applications. These can include:

- Failure to understand application dependencies and/or overly aggressive and complex project goals
- Lack of cost savings due to automation not being implemented or improved during the replatforming change (manual efforts are still required to operate the application)
- Negative impact on the customer experience due to poor performance or availability caused by unexpected errors
- Project time exceeds original plans because of unexpected errors and/or dependencies
- Failure to identify application incompatibility with the new platform

These risks and others can be avoided or minimized by using best practices that are informed and guided by observability data derived before, during, and after the replatforming project.

Compatibility Analysis

It's critical that you carefully evaluate your existing application, its structure, and business requirements to determine compatibility with the new target platform. Considerations could include factors such as:

- The new database service does not support the same requirements as the previous one.
- The application is not compatible with the available operating systems on the new platform (in this case, you might need to think about using a container replatform strategy).
- The new platform does not meet the security requirements of the application or the business (HIPAA, FedRAMP, GDPR, etc.)

Ensure you identify and establish compatibility for all relevant application factors that relate to your application and its business requirements. Incompatibilities might indicate that the application is a candidate for refactoring rather than replatforming.

Using Observability to Drive Replatforming Success

Successful modernization projects require deep visibility into your applications, customer experiences, and business outcomes. This means having comprehensive telemetry data at your fingertips so that you can make informed decisions about which applications to replatform and which candidate components to change. Observability and visibility are also essential to help you reduce overall project risk, streamline the project, and measure and prove success.

For complete observability into the applications you are considering replatforming, you need to begin instrumenting them and collecting telemetry data prior to taking any action. You will continue to track data during and after the project. In general, you should gather data about:

- End-user experience
- Application performance
- Application dependencies
- Application issues
- Resource usage
- Existing components

To identify candidates, you will review application endpoints, their versions, and their performance impact on the application. You're looking for applications with clear APIs to specific functionality that is "black boxed." The connections that depend on the component are critical to your success; it's important to identify them all by finding the dependencies using New Relic One.

A great place to start is with a service map of your application. New Relic One service maps are visual, customizable representations of your architecture showing connections and dependencies, including applications, databases, hosts, servers, and out-of-process services.

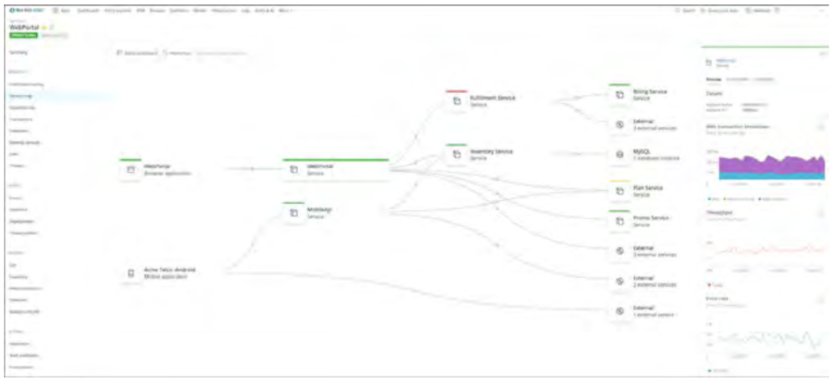


Figure 1: Understand how apps and services in your architecture connect and talk to each other using Service Maps.

Gain a Deep Understanding of Your Components

Develop a deeper understanding of existing componentry using New Relic One host integrations on AWS, which lets you filter, analyze, and query data about your AWS services. For a complete list of AWS integrations and to learn more, visit our documentation page.

With data in hand, you can then begin working through the steps of an informed replatforming project:

1. Identify candidates using service maps, end-user experiences, and application performance data.
2. Analyze the application for known issues.
3. Install the component on host integrations to get deeper visibility into component performance.
4. Choose a limited set of components to change.
5. Apply AWS Well-Architected Framework best practices.
6. Deploy, test, and transition using automation.

Use Observability Before, During, and After Your Project

Observability is not only imperative for the decision making and planning of your project, it's also the best way to gain the confidence and insight to successfully change components in your applications. That's because maintaining visibility into the application, infrastructure, componentry, end-user experience, and business success throughout the project gives you multiple advantages. With the right data, you can accelerate your efforts without increasing risk, anticipate and avoid problems, minimize complexity, and maximize the optimization of the application in the AWS environment.

Here's how to use observability at each phase of the replatforming project:

BEFORE: Begin instrumenting candidate applications in your current environment and collecting telemetry data with New Relic One to understand application performance, end-user experiences, resource consumption, error rates, application uptime, and other critical KPIs. These measurements will serve as baseline metrics for comparison during and after the replatforming. This data also allows you to create a plan for addressing systemic application issues that you've identified. Use the information you gather to understand the existing profile of the component you are looking to replatform and its current issues. Then prioritize the candidates based on the potential positive impact on business goals.

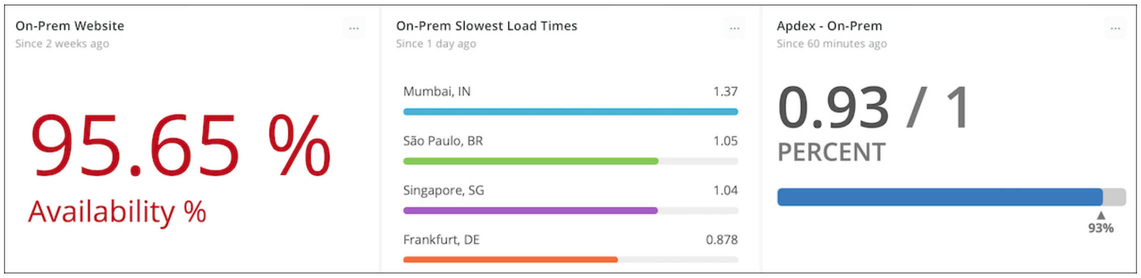
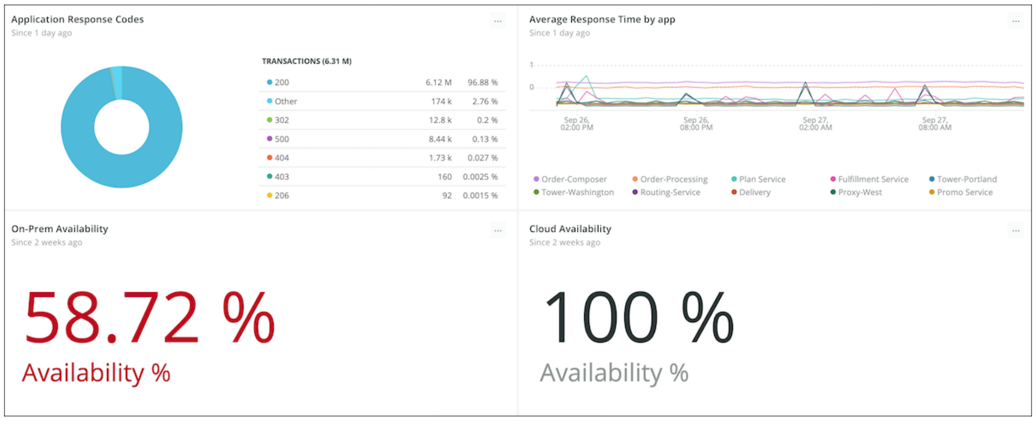


Figure 2: A baseline dashboard will bring all the KPIs together to establish the before state.

DURING: As you begin the replatforming project, it’s important to test your applications to make sure the transition isn’t introducing any problems or slowdowns. During replatforming, use New Relic One to check the performance of application code and external services and compare it against the pre-replatforming baselines established in the “before” phase. This will give you an apples-to-apples comparison.


Application Response Codes
Since 1 day ago

TRANSACTIONS (6.31 M)

● 200	6.12 M	96.88 %
● Other	174 k	2.76 %
● 302	12.8 k	0.2 %
● 500	8.44 k	0.13 %
● 404	1.73 k	0.027 %
● 403	160	0.0025 %
● 206	92	0.0015 %

On-Prem Availability
Since 2 weeks ago
58.72 %
Availability %

Cloud Availability
Since 2 weeks ago
100 %
Availability %

Figure 3: Quickly identify issues and roadblocks during replatforming.

AFTER: Once the replatforming is complete, use New Relic One to prove and measure your success and identify opportunities to further optimize applications for performance gains, efficient use of resources, and ease of operation.

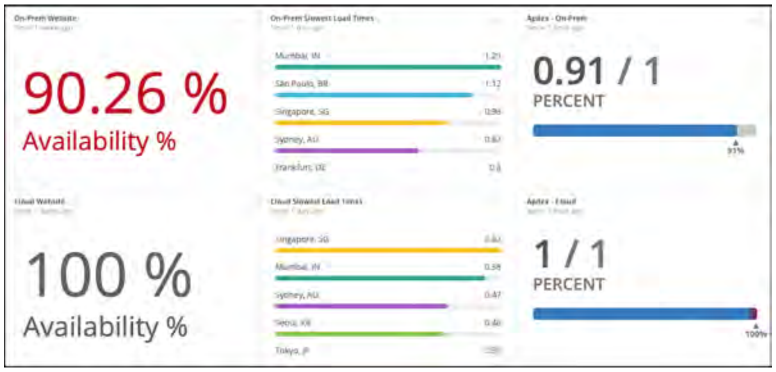


Figure 4: Compare your completed replatforming KPIs against your before KPIs.

Don’t Underestimate the Importance of Infrastructure-as-Code

A strong infrastructure-as-code DevOps practice allows you to experiment quickly. That’s because the ability to quickly set up and tear down environments in isolation lets you rapidly determine whether you will see a benefit in moving to a new component or service. Read AWS CloudFormation to learn more about codifying your infrastructure.

One way to measure and track replatforming success is by using New Relic One and New Relic Cloud Observability Framework combined with the best practices in the AWS Well-Architected Framework. The Well-Architected Framework consists of five pillars: performance efficiency, cost optimization, reliability, operational excellence, and security. It was developed to help companies build secure, high-performing, resilient, and efficient infrastructure for their applications. Since its introduction in 2012, the Well-Architected Framework has been updated eight times.

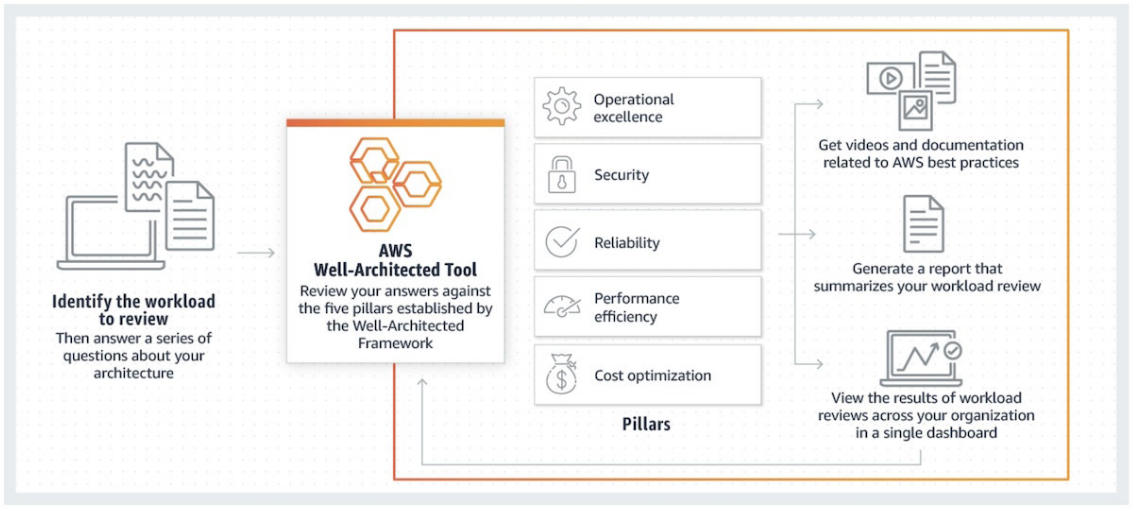
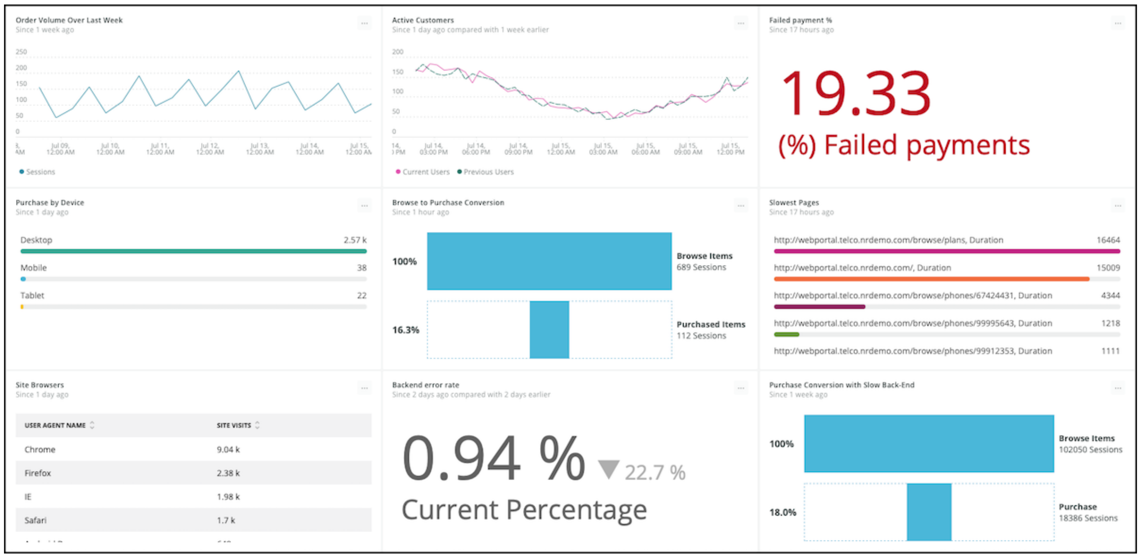

Identify the workload to review
Then answer a series of questions about your architecture
AWS Well-Architected Tool
Review your answers against the five pillars established by the Well-Architected Framework
Operational excellence
Security
Reliability
Performance efficiency
Cost optimization
Pillars
Get videos and documentation related to AWS best practices
Generate a report that summarizes your workload review
View the results of workload reviews across your organization in a single dashboard

Figure 5: The AWS Well-Architected Framework helps cloud architects build cloud infrastructures based on a proven track record.

AWS Well-Architected Framework: Performance Efficiency

A best practice related to the first pillar of the Well-Architected Framework is to monitor and observe the efficiency of resource usage before, during, and after your replatforming project. You can use New Relic One to review how resource usage has changed. How much efficiency have you gained?

While your replatforming project might not be triggered by performance improvements, you still need to understand the impact on your end users. Are you making things better? Did the change have any impact on the outcomes being driven by the application, such as more click-throughs or conversions because pages load more quickly? Gaining visibility into this information is critical to assess the impact of your efforts on revenue-generating systems.



Purchase by Device
Since 1 day ago

Desktop	2.57 k
Mobile	38
Tablet	22

Slowest Pages
Since 17 hours ago

http://webportal.telco.nrdemo.com/browse/plans, Duration	16464
http://webportal.telco.nrdemo.com/, Duration	15009
http://webportal.telco.nrdemo.com/browse/phones/67424431, Duration	4344
http://webportal.telco.nrdemo.com/browse/phones/99995643, Duration	1218
http://webportal.telco.nrdemo.com/browse/phones/99912353, Duration	1111

Site Browsers
Since 1 day ago

USER AGENT NAME	SITE VISITS
Chrome	9.04 k
Firefox	2.38 k
IE	1.98 k
Safari	1.7 k

Figure 6: Bring business, customer, and operations KPIs together.

AWS Well-Architected Framework: Cost Optimization

Apdex is an industry standard for measuring user satisfaction with the response time of web applications and services, which helps you see how satisfied users are with your application. Use New Relic One to compare your Apdex to the costs incurred by the application. Is this an appropriate level of investment for the role the application plays? What should your targets be for the end-user experience? Your goal is to balance these with the costs, meeting your end-user experience objectives without breaking the bank.

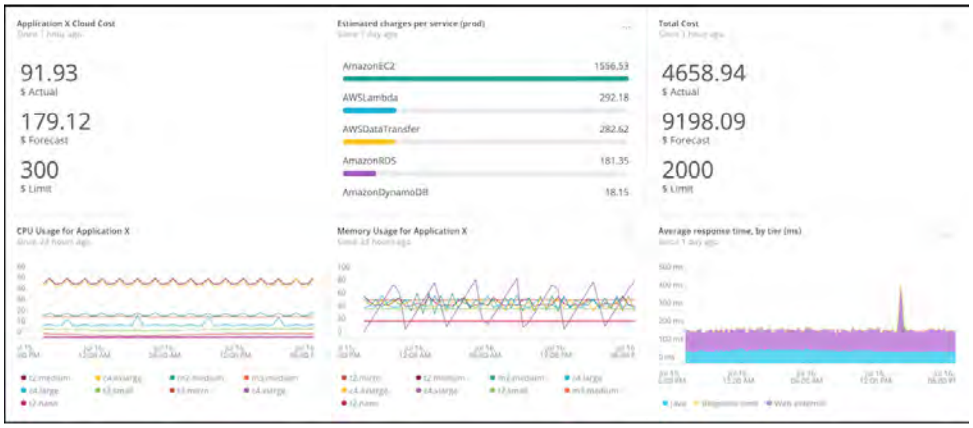


Figure 7: Track AWS budgets side by side with infrastructure and application KPIs.

Keep in mind that cost optimization as part of a replatforming exercise is about more than just license fees. It is also related to the cost of labor required to self-manage a component as well as the costs of any production issues or outages related to that component. Therefore, the costs to consider should include:

- The number of components (including component versions) that are being managed
- License cost
- Infrastructure cost (focusing on the balance between performance, reliability, and scalability)
- Operational cost
- Impact on the security footprint
- Error management and reduction of rework and issues related to the component

AWS Well-Architected Framework: Reliability

Today's digital businesses have stringent uptime requirements for critical applications. Start with a clear understanding of your service level agreement (SLA) and uptime requirements. Then compare reliability metrics to see how the application is performing against the SLA. What is your rate of application uptime? How often are your users impacted by errors?

Also consider that your application needs to scale and be reliable for your company’s busiest day, including expected and unexpected spikes. You need both scalability and reliability, working in concert, to achieve your SLA goals and keep your customers happy.

To that end, how well is your scaling performing? For most applications, scaling requirements are difficult to anticipate, even when predicting only six months into the future. With a small number of users, you don’t need to worry much about how well the application scales, but at higher numbers, scaling must happen automatically. You shouldn’t be spending time tuning your autoscaling rules. Based on the number of users, are you using the right technologies? How ready are you for an increase or decrease in usage?

You can get in-depth scaling information on your components using New Relic One host integrations. You can look at metrics such as refused connections, queue lengths, and shards relocating—all of which are indicators of an undersized system. On the other hand, the component may be overprovisioned, using too many resources to accommodate the peaks, but wasteful during the valleys. When you take advantage of an AWS managed service, much of this work is handled for you behind the scenes.

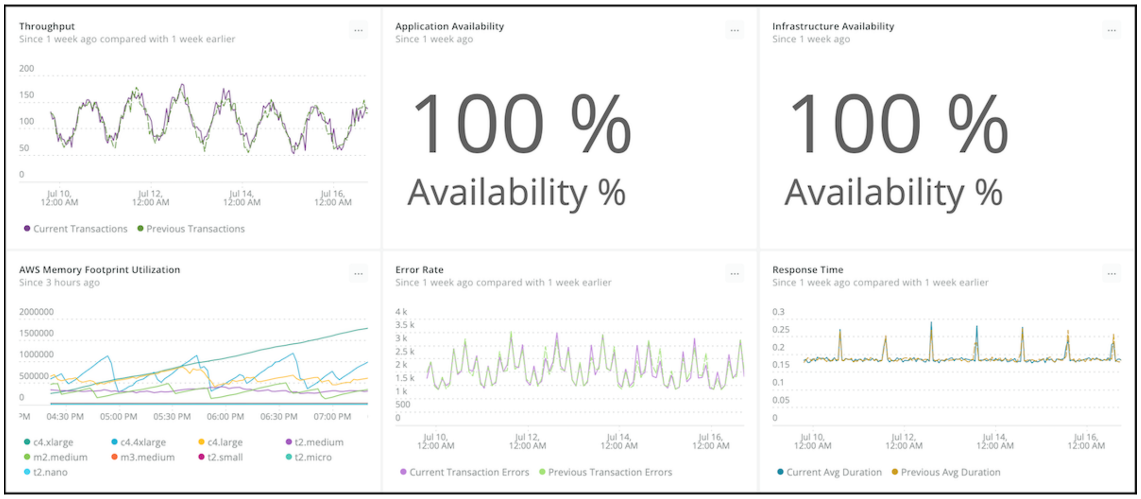


Figure 8: High-level dashboard displaying availability KPIs as well as backend metrics.

Keep in mind that this pillar of the Well-Architected Framework isn’t just about measuring your current reliability, but assessing whether you’re using the right technologies to be scalable and reliable tomorrow. Compare your current user base and expected user base to how much of your application is using seamlessly scalable technologies. While not every application will be entirely serverless, with the right data you can make better choices about what you are managing yourself.

Deploy Web Applications at Scale Using AWS

View this presentation from AWS Online Tech Talks to learn about how to make your web applications more scalable.

AWS Well-Architected Framework: Operational Excellence

As you go through the replatforming exercise, you should be looking for ways to increase the amount of automation. For instance, adopting managed services can free up developer time and make it easier for developers to experiment, because they can easily spin up a new environment to try out a new idea. Over time, how is development improving? How much more quickly can you deliver features? You should see that you have more time to spend on the tasks that will make a real difference to the business.

If you have code sprawl in relation to a particular component, you should identify all the areas that need to change. Then start monitoring things such as:

- Number of lines of code
- How well data is componentized
- Issues related to data in production
- How quickly you can provision an environment

As you componentize the data access layer, you reduce the number of lines of code you need to maintain. Less code means fewer errors, typically more rigorously tested code, and fewer areas that could be potential security risks. To achieve this, you'll need to start leveraging strong DevOps practices.

Deploying self-managed components such as Apache Kafka will not usually be as easily scriptable as using a CloudFormation template on AWS. Streamlining deployment and maintenance processes for a component results in fewer moving parts that you have to manage.

AWS Well-Architected Framework: Security

For security, an important metric to understand is how much your organization is reducing the security surface area and effort by using managed services. How many databases do you no longer need to track, manage, and patch from a security perspective? In addition to the number of databases, how many types of databases (such as MySQL, Oracle, Microsoft SQL Server), and how many different versions (1.2, 2.7, 5.4.2, etc.) do you no longer need to track? The more database types and versions you need to secure, the more your security challenge increases.

For instance, with Amazon Aurora, many security tasks are handled for you, allowing you to focus on other areas. You still need to provide proper rules to protect the data and control who has access, but the lower-level security concerns, such as operating system patches and updates, are managed by AWS. Because cybersecurity is both a technical and a legal issue, it's important to understand your organization's security and governance requirements and how they map to what is contractually provided by AWS.

Next Steps

Application modernization is an iterative process—you're never really finished. You should be continuously looking across your applications for additional components that could benefit from replatforming. And remember that replatforming isn't necessarily the next step in modernizing an application. Sometimes the best next step will be to refactor the application (make code-level changes to take advantage of modern cloud services, architectures, and technologies) to achieve new levels of quality, performance, scalability, reliability, and flexibility.

Regardless of where you are in your modernization efforts, New Relic One can help you optimize your applications faster, more cost-effectively, and with less risk. As an AWS Advanced Technology Partner, New Relic has been granted AWS Competency in six solution areas (Migration, DevOps, Containers, Mobile, Retail, and Government) based on our technical proficiency and proven customer success.

To learn how New Relic One can help you in replatforming or any phase of application and infrastructure modernization, visit newrelic.com/aws.

© Copyright 2020, New Relic, Inc. All rights reserved. All trademarks, trade names, service marks and logos referenced herein belong to their respective companies. 12.2020