E-book:

How to choose the best application modernization journey

A roadmap to a cloud-native transformation

Onica by rackspace technology | aws

Introduction

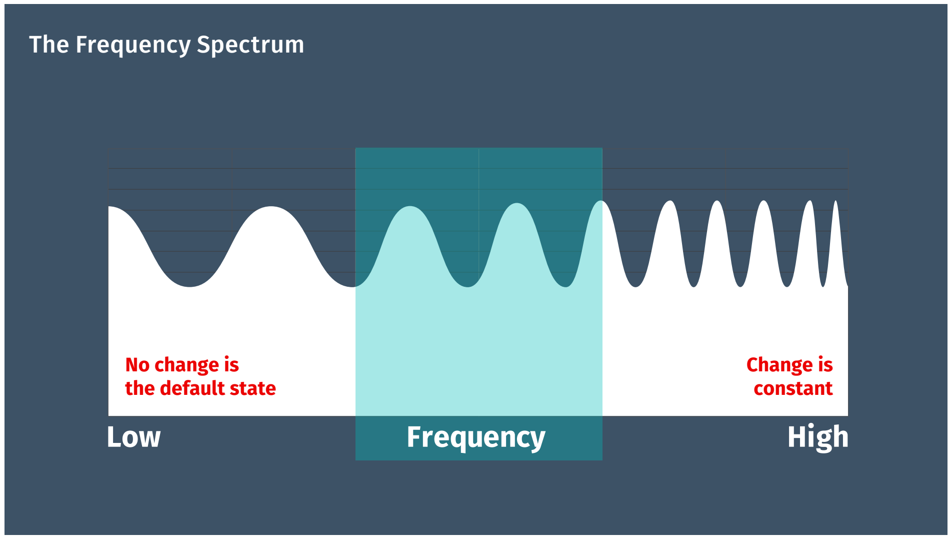
Application modernization is the process of progressively transforming applications, infrastructure and processes to gain the availability, scalability, business agility and cost optimization benefits that the cloud was designed to offer. Modernization is not about meeting the needs of a single point in time, because it's not trying to achieve a static state. Instead, it's ever changing in ways that allow organizations to meet the needs of the present and future — in particular, the frequency required to consistently deliver value to customers.

Why modernize?

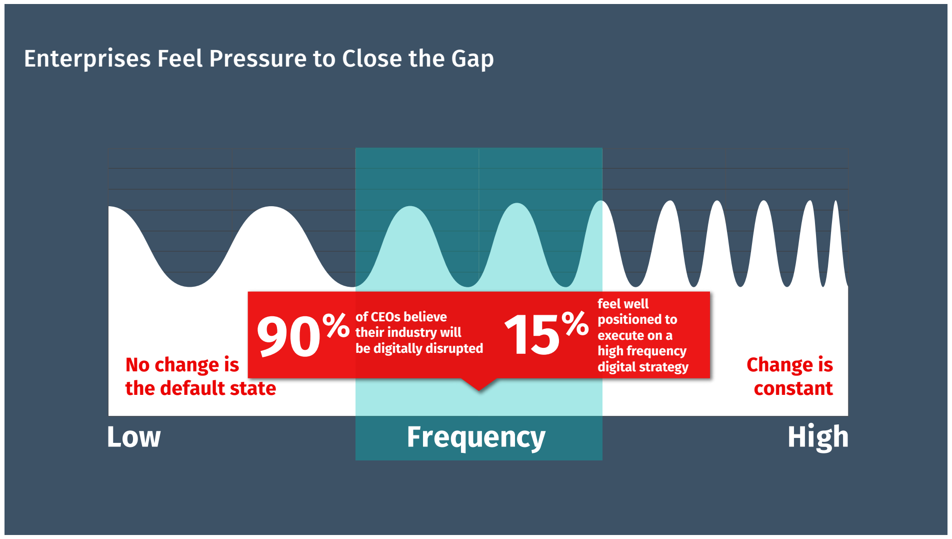
It's no surprise that today 90% of CEOs believe their industry will be disrupted (McKinsey). They believe that the next Uber or Netflix will come along and unseat them. To remain competitive, 67% of business leaders believe they have to move to a high-frequency mode (Gartner) (source: https://aws.amazon.com/blogs/enterprise-strategy/tackling-transformation-you-can-get-there-from-here/). Even if a new startup isn't coming along to unseat them, they still want to deliver new experiences to their customers more frequently, stay relevant and provide more value.

Moving from low to high frequency

Achieving a high-frequency state means having the ability to reduce innovation delays so that organizations can develop new features and capabilities faster. Large organizations burdened by existing workflows and legacy applications are likely to be low frequency. Younger and more nimble companies often operate at a high frequency. The difference is significant. Companies that have successfully implemented high-frequency practices deliver value 46 times more frequently than traditional, low-frequency enterprises (Puppet Labs State of DevOps Report).



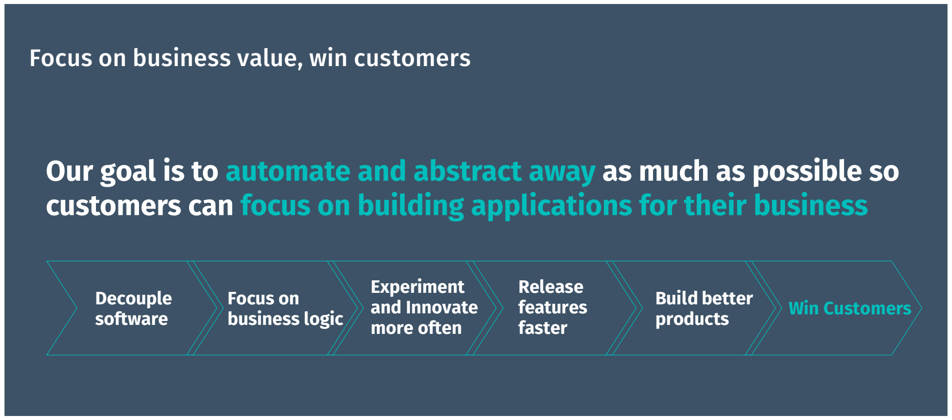
Caption: The Frequency Spectrum: On the left you have low frequency enterprises where no change is the default state. Change is hard, risky, and takes a lot of time and effort. Versus the right side where change is constant. There is a flywheel of frequent value delivery. Ideas are considered hypotheses, and teams are closer to customers and generally more focused on customer value.


Enterprises Feel Pressure to Close the Gap
90% of CEOs believe their industry will be digitally disrupted
15% feel well positioned to execute on a high frequency digital strategy
No change is the default state
Change is constant
Low
Frequency
High

Caption: According to McKinsey, 90% of CEOs believe their industry will be digitally disrupted, but only 15% feel well-positioned to execute on a high frequency digital strategy. (source: https://aws.amazon.com/blogs/enterprise-strategy/tackling-transformation-you-can-get-there-from-here/). Containers versus virtual machines

A new approach to high frequency

As organizations are experiencing this paradigm shift, they're in the process of moving from old-school, single-application monoliths to modern applications. This shift is driven by a process that begins with winning customers, then working backwards.



Focus on business value, win customers
Our goal is to automate and abstract away as much as possible so customers can focus on building applications for their business
Decouple software
Focus on business logic
Experiment and Innovate more often
Release features faster
Build better products
Win Customers

1. To win more customers, you need to build better products
2. To build better products, you need the ability to release features faster
3. To release features faster, you need to experiment and innovate often
4. To experiment and innovate often, you need the ability to focus on differentiating business logic
5. To differentiate business logic, you need to enable better release processes and decouple your software systems
6. To decouple software systems, you need to break up the monolith into smaller microservices so they can be released at their own pace

The hallmark of a modern application centers around shrinking the project scope to solve business challenges faster. This means automating everything to support rapid and continuous innovation and applying the right tools to create purpose-built solutions with less effort and lower costs.

To carry out an application modernization initiative, organizations must apply structure to applications with modern frameworks, such as microservices or domain-driven design. Whenever possible, they must use cloud-native technologies versus trying to shift legacy IT to the cloud. And they must look for ways to use code to model applications and infrastructure versus building them by hand. While the path to modernizing applications can be complex, the benefits are well worth the journey, including:

- Everything is automated, codified and programmatic, so releases are easy and consistent
- Applications are resilient and elastic, automatically adapting to usage spikes and transparently healing from service outages
- Everything is modular and interoperable, which provides the confidence to move quickly
- Security is built into every aspect of the application lifecycle rather than being added at the end of development to reduce the likelihood of risk

Three application modernization journeys

So how do organizations achieve these desired outcomes? There are three primary options: container platforms, serverless and cloud-native replatforming. Each approach offers different opportunities and advantages.

Containers — self-contained platforms

A container is a lightweight, stand-alone, executable package of software that includes all of the elements needed to run applications, including code, runtime, system tools, system libraries and settings. It has long roots in other technologies, but Docker changed the game when it simplified the creation, management and operation of containers, making them more approachable for nearly any type of application.

Containers deliver a wide range of benefits, including:

- Bridges the gap from a simple lift-and-shift to full cloud-native serverless
- Delivers a portable application environment that can run anywhere, providing support for hybrid strategies
- Packages applications and dependencies into a single artifact that simplifies deployment and promotion across environments
- Isolates software from its surroundings in self-contained packages
- Reduces conflicts between teams running different software on the same infrastructure
- Speeds up development and deployment cycles
- Provides better resource utilization

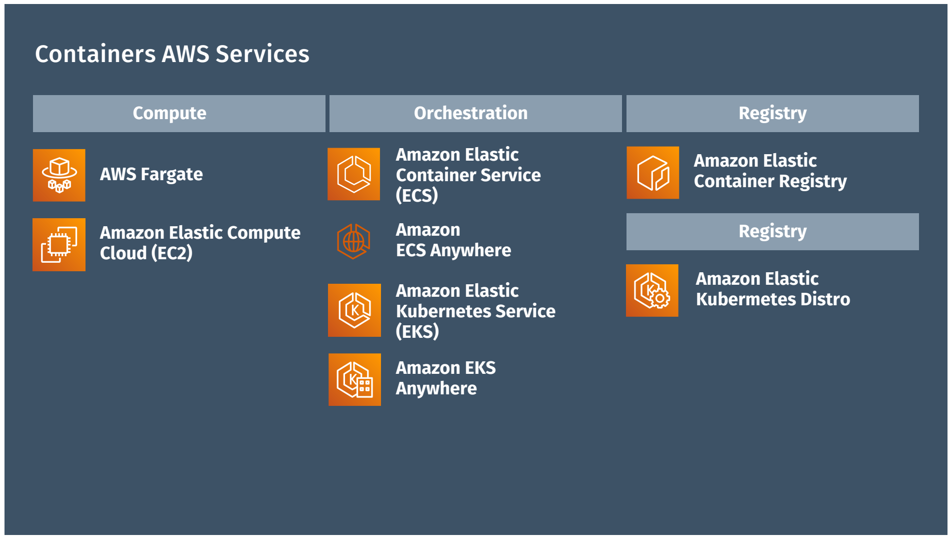
Containers versus virtual machines

When comparing containers to virtual machines (VMs), organizations gain many advantages, including reducing application size, becoming more agile, scaling faster and deploying rapidly.

AWS delivers services for containers

We no longer live in a world with a single database or architectural option. For example, Amazon Web Services (AWS) is providing ways for organizations to shrink the scope of their initiatives, offload the undifferentiated parts, choose the right tool for the job and automate everything. In this way, organizations can build modern applications that win customers and retain their competitive
advantage in the marketplace.

AWS offers a range of compute, orchestration and registry services to help run containers, including those depicted in this image:


Containers AWS Services
Compute
Orchestration
Registry
AWS Fargate
Amazon Elastic Compute Cloud (EC2)
Amazon Elastic Container Service (ECS)
Amazon ECS Anywhere
Amazon Elastic Kubernetes Service (EKS)
Amazon EKS Anywhere
Amazon Elastic Container Registry
Registry
Amazon Elastic Kubermetes Distro

Serverless supports self-healing and autoscaling applications

Serverless computing introduced a unique capability into the IT world. It provided a place where applications run without much management. This has allowed them to focus on application development without worrying about infrastructure constraints. It has enabled organizations to break free from the constraints of maintaining the place where their applications run, so they can focus on the application itself and what it needs to do. This freedom supports several advantages, including:

- Reduces operational complexity and increases developer productivity
- Reduces most concerns around provisioning and utilization, availability, fault tolerance, scaling, operations and management
- Supports innovation by freeing internal teams to focus on business challenges
- Frees organizations from dealing with legacy issues that predate it
- Speeds up time-to-market
- Reduces infrastructure costs by eliminating the need to pay for idle resources

AWS delivers services for serverless

When most people think of serverless, they think of compute and AWS Lambda. While Lambda epitomizes the serverless revolution, it's just the tip of the spear. AWS provides a serverless ecosystem that consists of a number of services that, when combined, help build fully serverless applications, for example:

Serverless operation best practices

There are three key ways to make the most of serverless environments, including strategic code organization, serverless-specific tooling to enable automated deployment, and streamlined operations and governance. Together, they make serverless an important consideration when organizations want to develop modern applications.

Strategic code organization: AWS Lambda functions should contain the minimal code required to handle an event — from an API, IoT message, or other trigger — before handing off processing into code that's abstracted from the runtime. By following this organization pattern, developers can continue to leverage best practice code organization techniques, including those that enable automated unit and integration testing.

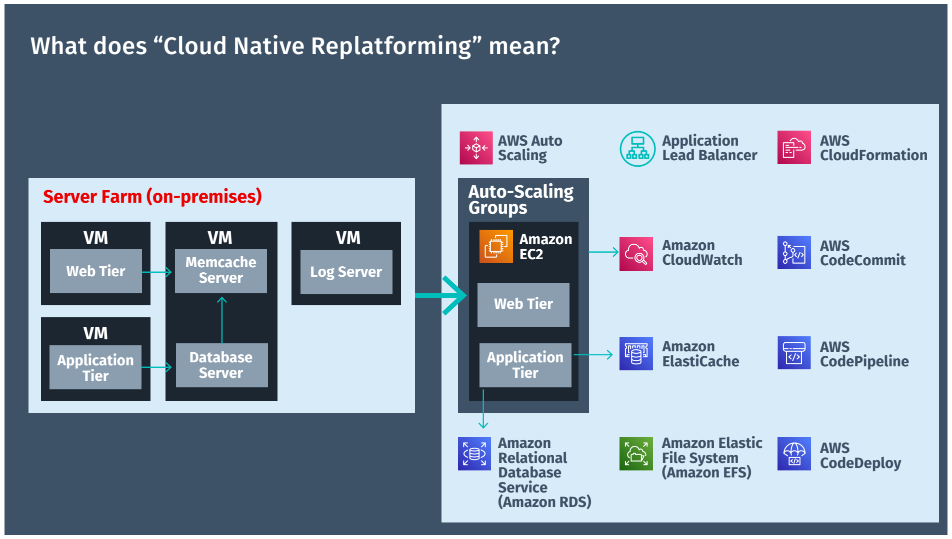
Serverless-specific tooling: Serverless apps have dozens or even hundreds of moving parts, which means deploying by hand is not only a waste of time, but nearly impossible. Moreover, to effectively test apps, developers must constantly deploy them to AWS, overwhelming traditional infrastructure as code (IaC) approaches. Serverless-specific IaC tools provide the right level of abstraction to ensure simple and quick deployments for each developer, as well as promotion paths to quality assurance and production.

Automated operations and governance: The number of pieces involved in a serverless app means operations and security teams struggle to identify problems and security issues. Rather than requiring pre-approval and manual audits for every developer decision, automated monitoring techniques and security guardrails provide operations and security teams with the visibility and governance required, without stifling developer innovation or reducing overall project velocity.

Cloud-native replatforming — lift, tinker and shift

This is a leading-edge option. By incorporating cloud managed services, organizations gain elasticity and responsiveness without rebuilding their applications.

To understand cloud native replatforming, think of it as a process of "lifting, tinkering and shifting" to a cloud-native environment, while minimizing intrusive code change. By adding tinkering to the traditional lift-and-shift paradigm, application environments can be simplified. For example, instead of managing a database cluster, leveraging Amazon Relational Database Service (RDS), offloads complex but undifferentiated work around software patching, data durability, availability, scaling and backup management to AWS. Though the applications don't change, the simplified environment improves the overall application experience and allows teams to focus on what matters.



Managed services for cloud-native replatforming

Cloud-native replatforming is moving applications to the cloud in a way that enables you to take advantage of the benefits that the cloud was designed to deliver. Managed services play a significant role in this process.

To embrace managed services for cloud-native replatforming, organizations need to identify the nonproprietary services they're running and managing with their applications. Common dependencies include databases, queues, cache layers, load balancers, API gateways, email-sending services and container orchestration layers. Finally, they need to identify AWS managed solutions that can replace the dependency to simplify overall management responsibilities.

The goal is not to replicate what the organization had on-premises, but rather, to take what they had and pull off the pieces that aren't unique to it, such as relational databases, and then switch those to a managed solution, such as Amazon RDS. In this way, without really changing the applications, organizations can improve their reliability, reduce their management overhead and save money.

AWS delivers services for cloud-native replatforming

AWS provides a range of services to support cloud-native replatforming operations, including:

AWS native services:

- AWS CodeCommit: source control
- Amazon CloudWatch: monitoring
- AWS CodeDeploy: deployment

Reducing operational pain:

- AWS Auto Scaling: elasticity and resiliency
- AWS CloudFormation: automated and consistent infrastructure management
- Amazon Elastic File System: automated backups

Lowering cost:

- Amazon CloudWatch: accountability
- AWS Lambda: usage-based billing

Iterative DevSecOps:

- AWS CodePipeline: multi-environment
- AWS Config: governance and guardrails

Which modernization path should you choose?

There are considerations and tradeoffs for containers, serverless and cloud-native replatforming. The right approach for you will depend on your specific workloads, processes, and team’s level of understanding and maturity. As your capabilities evolve, you can start to incorporate other approaches to the mix. The goal is to match your specific workloads to the right approach at the right time.

The following insight provides guidance for making the best decision for your organization.

When to consider containers

The common lingua franca of all cloud providers, offers portability, scalability and immutability:

- When consistency between stages, clouds and environments is a priority
- When you want to develop microservices applications using familiar languages and frameworks
- If you’re just beginning to break apart monolithic applications and you need a good steppingstone
- In situations that require rapid scaling, potentially across platforms
- When you need to build stateful or long-running applications

When to consider serverless

Next-level abstraction, moving applications further away from the infrastructure, with usage-based billing:

- Public cloud as the primary deployment target is a good place to start
- Event-driven workloads, such as API backends or stream-processing applications
- Short-lived, responsive applications that typically complete a task in seconds
- Net-new or rebuilt applications or services, where prior architecture and tools don’t constrain technical decisions
- In stateless applications, where data can be stored externally versus applications that require in-memory
- When hybrid deployments and cloud-to-cloud portability are not requirements

When to consider cloud-native replatforming

Leverage a “rising tides lifts all boats” approach:

- Applications that are being actively developed or have recently evolved can use cloud-native replatforming to recoup that investment and continue to grow
- When you’re using off-the-shelf commercial software that you want to make more cloud native
- When you have integrated software platforms, such as ERP, that are highly complex within your cloud-native platforms
- If you want to improve pieces of an application that are creating dependencies or technical debt, but you don’t want to reinvent the entire application
- If you’re replacing single-function VMs and self-managed dependent services with PaaS
- If you’re looking to enable automation and auto-scaling without investing in an entirely event-driven approach
- If you’re looking to embrace DevOps and need to modernize legacy workloads as a steppingstone toward improving some of your larger operational concerns

Questions to consider

As you make your transition, you want to do more than just select tools. You want to ensure your team is structured in a way that supports the new development process. The right approach will find the right balance. To choose the best path forward, consider your answers to these questions:

- How do we approach this from an infrastructure-as-code perspective, in other words, automate the provisioning of infrastructure and applications?
- How do we build applications and structure for our team to test as early in the process as possible, and automate testing before it reaches users?
- How do we build and monitor safety every step of the way to reduce organizational risk while becoming more high frequency?
- What capabilities do we need to innovate quickly to meet the needs of our customers in ways that help us continue to stay relevant in the marketplace?

Choose your modernization journey

Choosing between the three main pathways to application modernization is largely based on the age of your application, its need to change and your ability to change it. By following one of these pathways, you will be able to leverage AWS services to modernize an existing application system, solution or workload.

To achieve the best of all worlds on your journey to application modernization, you’ll want to consider both the tools and the way you structure your teams to support the development processes, and to take advantage of best practices throughout the entire application lifecycle management. In this way, you’ll optimize every advantage of application modernization — and set your organization up for maximum success.

To start down that path, request your complimentary Application Modernization Strategy Session today.

About Onica by Rackspace Technology

Onica by Rackspace Technology is the dedicated Amazon Web Services (AWS) business unit at Rackspace Technology serving North America. Onica helps customers drive innovation, agility, cost savings and operational efficiency in the AWS cloud. Onica delivers professional services and engagement practices focused on bringing the most cutting-edge AWS capabilities to every customer through deep expertise in AWS strategy, cloud native development, containers, application modernization, AI and machine learning, and IoT. With in-depth advisory services built on 15 competencies and experience with 1,000+ customer launches, Onica by Rackspace Technology is here to help you accelerate cloud native transformation on AWS.

Learn more at rackspace.com/onica-by-rackspace-technology or call 1-800-961-2888.

Rackspace-Ebook-App-Modernization-on-AWS-TSK-8146-V2 :: March 8, 2023