SIMFORM

Strategy for App Modernization on the AWS Cloud:

Driving digital transformation with Containers & Serverless

Companies worldwide are under pressure to modernize and develop systems that produce immediate results, improve service, and meet customers' expectations. For this, they must overcome the limitations of legacy systems and have a tech stack capable of meeting diverse customer demographics.

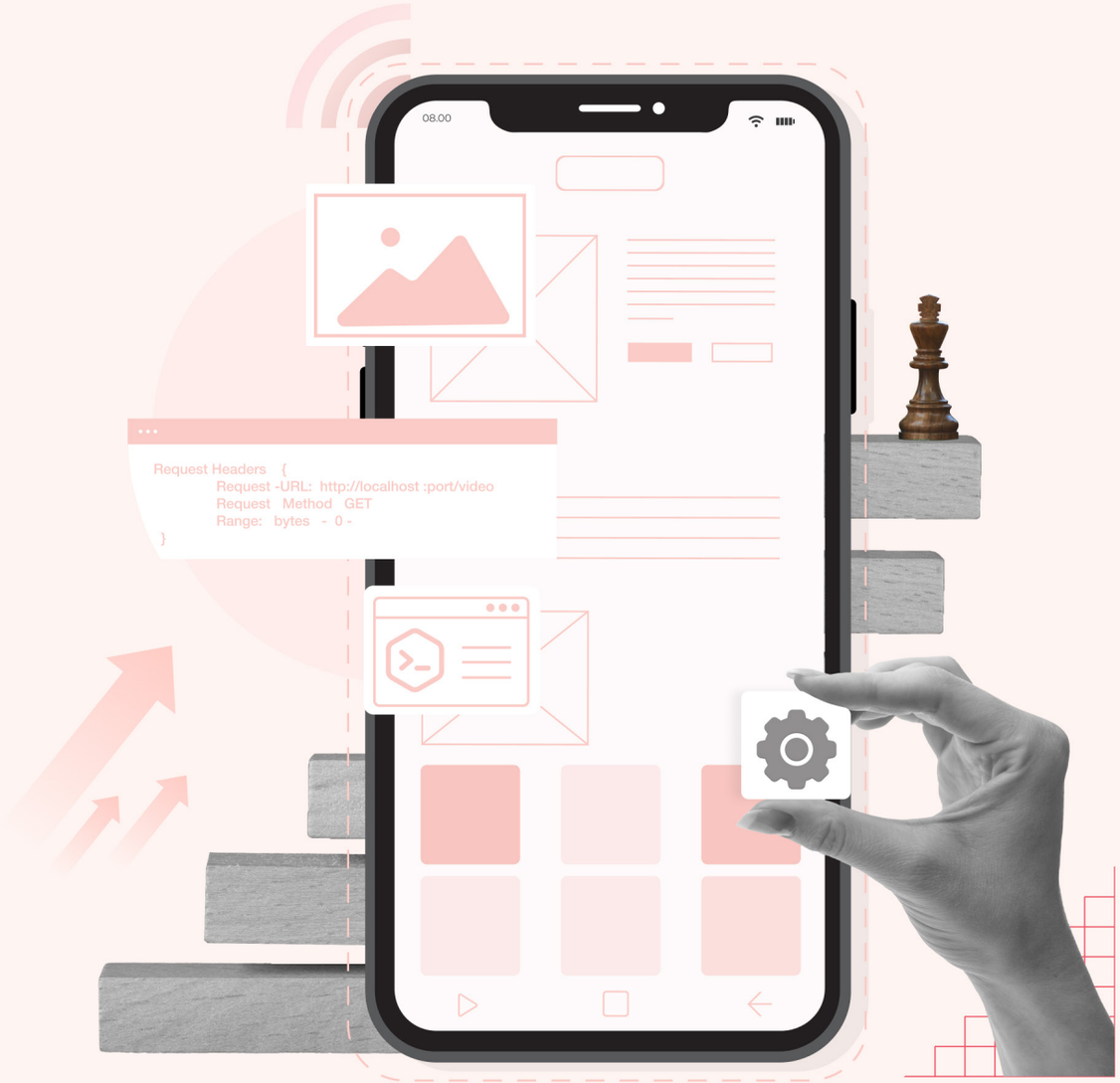
Legacy services and architectures, also known as "monolith" architecture, are still used by many businesses. However, as they race to find better ways and technologies to boost agility and better respond to customer demands, now is the time to focus on individual parts of the application. This will also enable cross-functional teams to better understand different application components and work faster.

Application modernization is a multi-faceted approach to leveraging new technology and staying ahead of the competition. It's all digital these days, and it's not uncommon to see examples of cloud-native businesses and products thriving in the market and gaining customer trust. Modernizing your application portfolio is the first step toward driving digital transformation in your organization through the cloud and automated environments.

AWS, the market leader in on-demand cloud computing platforms, assists businesses in various ways, including containers and serverless computing, which we will discuss in this ebook.

Contents

01



Modernize your application: It’s time to get ahead of your competitors!

The process of transforming existing infrastructure with cloud technology is known as application modernization. Organizations begin modernizing their application portfolios by altering internal architecture/features, replacing legacy technology stacks with modern technologies, and developing a comprehensive plan for moving applications to the cloud. It entails either migrating applications from on-premises to a modern cloud platform or rearchitecting them from the ground up.

Organizations face the following challenges when it comes to application modernization:

- Misalignment of business and technology teams
- Changes in business processes
- The complexity of technology tools
- Cost

The advantages of application modernization, on the other hand, generate more business value and are more meaningful. The advantages outweigh the drawbacks. Instead, it motivates businesses to rethink their business processes.

Need of modernizing legacy applications: Improving the quality of customer experiences

Digital transformations have made it imperative for decision-makers to modernize legacy systems. No organization wants its business objectives and processes held back due to the legacy systems. It's not that legacy systems were not making sense before, but the capabilities of modern systems and technologies make customer expectations soar high.

While most businesses have started embracing technology transformations, there is no scope for companies and great products to stay back and wait. But instead, do a technology shift.

The cost of holding onto legacy systems- the technical debt

In the long run, easier and faster solutions do not pay off. There are no shortcuts; you need something concrete and scalable that pays off now and in the future. Technical debt is becoming more of a problem for businesses of all sizes. According to a McKinsey survey, resolving issues related to technical debt consumes 10 to 20% of the technology budget dedicated to new products.

Legacy applications or architectures, such as monolithic and on-premises software, may prevent you from fully exploiting the cloud's high availability and scalability capabilities. Technical debts prevent organizations from reaping the true benefits of technology and creating delightful customer experiences, similar to how financial debts prevent people from achieving long-term goals.

Legacy systems exist but are no longer efficient to stand against the latest security threats and are not strong enough to meet performance benchmarks. What's more, it's a challenge to find people who can manage them with minimum support available. And the biggest factor is the "hidden costs" regardless of its incompetency in scalability and flexibility.

According to one study, the cost of maintaining legacy systems is around $337 million a year. It is also estimated that legacy systems cost approximately a 15% budget increase every year for maintenance to organizations.

Despite these costs, inefficient performance and limited scalability only make for an increased burden on IT departments and decreased business ROI. Meeting business demands and scalability requirements is difficult with monolithic/legacy applications. Monolithic architectures are limited at some point, and it is a time-consuming process to deploy new features quickly on these applications.

To combat these issues, AWS has developed a **well-architected** framework to assist organizations in constructing high-performing and reliable infrastructure that is cost-effective, scalable, and modernized. This is especially useful for teams converting legacy systems to modern cloud platforms for the first time, as it includes a set of best practices that aligns business and cloud goals.

This is a fact that cannot be disputed! The well-architected framework developed by AWS has made it easier for teams to collaborate and identify pain points when developing cloud-based applications on the platform.

Modern cloud technologies provide elasticity, agility, and high availability.

Containers, a cloud computing technology, allow applications to scale automatically and facilitate resource auto-scaling.

In addition, in today's systems, agility is a must. Because of the cloud, it only takes a minute or two to get a Virtual Machine up and running, allowing for quick resource allocation and significant cost savings.

You need applications with the highest availability and zero downtime to handle unpredictable traffic spikes. Thanks to modern technologies like containers, system availability is assured, which provides excellent portability between platforms and clouds.

Containers & Serverless- Two modern technologies at work!

With modern cloud technologies such as containers and serverless, it is possible to update separate modules of your application and deploy features faster. Modernizing applications enables organizations to achieve faster time to market, develop and scale applications more rapidly, and improve overall operational efficiency. Also, the result is enhanced user experiences and organization-wide productivity.
And there will be reduced costs for migrating to cloud technologies and adopting newer practices compared to not migrating to the cloud.

These reduced costs are reduced usage of resources, fewer licensing costs, and fewer expenses related to the support services.

Let's see what the initial migration process looks like when organizations migrate their application portfolios and workloads with AWS.

02

Streamlining the development environment with containers

Containers are lightweight software environments for applications to run and scale anywhere. Containers package the applications with their dependencies, configuration files, and interfaces, removing the unnecessary burden of developers on the heavy lifting of infrastructure management. Instead, they can focus on building the core application—adding new features or adding the latest security instead of solving the compatibility issues of different environments.

It virtualizes your applications as packaged, self-contained, ready-to-deploy parts. It also sometimes contains the business logic to run the applications. Different cloud vendors and services, including AWS, use these virtualization techniques to achieve business goals of rapid deployment of new features, the elasticity of large-scale resources, and scalability that seem unachievable with virtual machines.

While many solutions help businesses modernize their applications, containers have become a go-to solution for developers to package and deploy applications efficiently.

What impact do Containers have on development practices?

Containers not only modernize your apps, but they also redefine your development processes. It replaces the traditional development cycle with ownership of application and code quality control. Previously, developers were solely concerned with “building the application” and had little control over the deployment process. It’s different with containers; now, the emphasis is on “quality,” and they’re given ownership of their work. Containerization is all about fostering high-quality product development and shorter feedback loops, as well as automating deployments and becoming the primary driver of overall improved development agility.

Old enterprise with a bunch of legacy systems? Let's get it sorted!

It's an opportunity for large enterprises to modernize its applications, put themselves forth in the international markets, and develop a brand new image. Old enterprises can transform the processes and achieve agility and faster feedback loops with all-new cloud-based approaches.

Simform has worked with many such organizations to transform legacy systems from concept to prototype and deployment in a matter of weeks with AWS-enabled container services and automated environments.

Containers on AWS

AWS offers a wide range of solutions to run your applications in containers on the AWS platform in the most secure and scalable environment, making it the most favorable platform for customers. As a result, AWS accounts for 80% of the market for running containers in the cloud.

Containers make the transition to microservices easier, allowing developers to work independently on different application modules.

With AWS, you can build secure microservices in your containers using its compliance and governance services and choosing suitable compute and orchestration services.

To make containerization seamless, here is what AWS provides as a solution using provisioning, orchestration, automation, security, and monitoring.

Provisioning

Seamlessly provision the infrastructure and resources

- AWS Fargate
- Amazon EC2

Orchestration

Scale and manage Dockers or Kubernetes containers effortlessly

- Amazon Elastic Container Service(ECS)
- Amazon Elastic Kubernetes Service(Amazon EKS)

Security

Detect vulnerabilities in containers

- AWS Identity and Access Management (IAM)
- Amazon Virtual Private Cloud (VPC)
- Image scanning solutions

Networking and connectivity

Distribute application traffic across containers

- AWS Elastic load balancer(ELB)
- AWS App Mesh

Automation

Automatically deploy code using CI/CD.

- AWS CodePipeline
- AWS App Runner
- AWS CodeCommit

Observation and monitoring

Ensure seamless communication between containers services and they are healthy

Provisioning

AWS has two options to provide the underlying infrastructure to run your container applications depending on the amount of control and management you need.

AWS Fargate

Using Fargate, you don't need to manage servers or clusters. Instead, you can package the application in containers by specifying the memory allocation, networking and IAM policies, etc.

Use cases:

- When you want to remove the operational overhead of infrastructure,
- Build and deploy applications, microservices, APIs
- Run and scale containerized data processing workloads

AWS EC2

Using EC2(Amazon Elastic Compute Cloud), you get granular level control over the infrastructure to run your container applications.

Use cases:

- Run cloud-native enterprise applications
- For dynamic and scalable environments
- Serve large-scale testing and development environments
- Run on-demand Mac OS workloads

Case Example by Simform

Simform recently implemented a containerized solution for a sports organization to create a fan-centric experience and optimize their application performance and content delivery mechanism to serve thousands of fans during popular events like the Olympics.
We implemented the solution using various technologies, but our main tech stack was around AWS EC2, CodePipeline, and AWS Elastic Load Balancer.

CHALLENGES:

- The main challenges discovered were that their frontend architecture was not scalable and did not follow any standard design pattern, making the code less readable, and some bad architectural decisions needed to be fixed.
- There were no content delivery or caching mechanisms present to serve the global users effectively. Also, there were scopes of improving DevOps workflows and automating build and release cycles to push faster and more frequent updates.

SOLUTION:

- Our team had to refactor the code with standard development practices by replacing large chunks of code.
- We implemented a proper queuing mechanism to handle requests and bind the server responses with the front end, including various scenarios.
- We utilized AWS CodePipeline to manage the DevOps workflows and automate CI/CD pipelines. Using the CI/CD automation tools, we also optimized the mobile app update process from hours to minutes using the CI/CD automation tools.
- Also, AWS EC2 was used to host the mobile application to serve the thousands of requests and scale the resources to handle traffic spikes.
- We used an AWS load balancer to create the EC2 instances automatically; whenever the traffic increases, the number of instances increases.

Orchestration

Once the application is containerized, the next step is to run the containers in a production environment. Amazon ECS (Elastic Container Service) and Amazon ECS Anywhere are the container orchestration services that enable you to deploy containerized workloads on AWS.

To scale your architecture, you will want an orchestration tool. AWS offers orchestration platforms to fit your needs, whether on-premises or in the cloud. With Amazon ECS Anywhere, you can use the same familiar Amazon ECS console and operator tools to manage your on-premises container workloads for a consistent experience across your container-based applications. The AWS Systems Manager (formerly known as SSM) integration automatically
and securely establishes trust between your on-premises hardware and the AWS control plane.

Amazon ECS and Amazon ECS Anywhere:

You can manage enterprise application portfolios using Amazon ECS, which allows us to grow from a single Docker container. It enables enterprises to run and scale container workloads across availability zones. Also, provides a managed and scalable set of services to perform the traffic management functions. This type of service is suitable for enterprises that need control over their infrastructure.

Amazon EKS and Amazon EKS Anywhere:

Kubernetes is a rapidly growing open-source container management platform built to help you run your containers at scale. With Amazon EKS, the Kubernetes management infrastructure is run for you across multiple AWS availability zones. The availability and scalability of Kubernetes control plane nodes are managed automatically by Amazon EKS. These nodes are in charge of scheduling containers, managing application availability, storing cluster data, and performing other important tasks.

You can run your Kubernetes applications on Amazon EC2 or AWS Fargate, which provide serverless compute for containers.

Our cloud engineering teams use Amazon EKS, Amazon EKS Anywhere, and Amazon ECR to automatically manage the availability and scalability of the Kubernetes control plane node for scheduling containers. And to store, manage, and deploy Docker container registry.

Amazon ECR:

To use Docker containers, the first thing you need is a Docker image. This image acts as a blueprint for creating instances of containers.

Amazon ECR is a fully-managed Docker container registry that makes it easy for you to store, manage, and deploy Docker container images. Amazon ECR is integrated with Amazon ECS; it simplifies your development-to-production workflow. Amazon ECR eliminates the need to operate your container repositories or worry about scaling the underlying infrastructure.

Use cases:

Amazon ECS:

- Deploy container-based applications on-premises or in the cloud
- Run and scale applications in multiple availability zones

Amazon EKS:

- Manage Kubernetes clusters and applications in hybrid environments
- Build and run web apps in multiple availability zones in a highly scalable configuration

How did Coursera move to microservices-based architecture?

Coursera, an educational technology company with over 13 million users in 190 countries and over 1000 courses, has switched to a microservices architecture.

CHALLENGES:

- Coursera was facing difficulty running and deploying jobs with a large monolithic architecture. In addition, the legacy architecture was insufficient to handle multiple jobs and handle different amounts of memory and CPU, which was creating inefficiencies.
- The infrastructure team at Coursera attempted to move to microservices-based architecture without using the AWS container services(ECS). Still, they ended up running into problems with managing the Mesos cluster.

AMAZON EC2 CAME UP AS A “SOLUTION.”

- They created each job as a container using Amazon EC2 Container Services and easily moved to a microservices-based architecture.
- With the use of Amazon ECS, it was easy to manage clusters as ECS itself can manage the details of clusters. Coursera also saw the deployment times reduced from hours to minutes, and each team was able to work independently, making the organization more agile and productive.
- There was no additional infrastructure management or engineering time spent as ECS handled everything from cluster management to container orchestration, making infrastructure operationally efficient.

How did Momenta use Amazon EKS to migrate from a self-managed Kubernetes cluster to AWS?

Momenta is a Beijing-based company working on creating the “brain” for autonomous driving. In addition, it is working on enabling autonomous driving at scale.

They have been using the Amazon Web Services ecosystem for a long time, and in 2020, they decided to migrate their Kubernetes clusters to Amazon Elastic Kubernetes Service (EKS).

CHALLENGES:

- Initially, the company relied on self-managed, open-source Kubernetes clusters to run its containers. Still, as the business expanded, they started facing challenges with scaling, availability of systems, and cost.
- They were also facing issues with the performance and ability to scale when it came to increasing the number of nodes for growing the business. Their need for scalability and systems availability put increased operational and maintenance burdens on Momenta.
- So to address all these challenges, they decided to migrate to Amazon EKS for increased security of node deployment and cost-effective, scalable solutions.

SOLUTION:

- Since Amazon EKS is a managed service, they need not have to manage the Kubernetes control plane. Using AWS availability zones, EKS automatically detects and replaces non-working nodes and provides on-demand upgrades and the latest security patches.
- They completed the entire migration process in 2 months and saved significantly on computing storage by optimizing workload performances.

Security

Like quality control, security considerations have also shifted into the earlier stages of the development cycle. With more autonomy, developers can more readily adapt their code to address the latest security threats. However, security must be a priority across the organization. One way to support this cultural change is to make security efforts as transparent as possible and define an architecture upfront that takes security best practices and tools into account.

AWS provides multiple tools to control who has access to your containers. You can use AWS IAM to determine who is authenticated (signed in) and authorized (has permission) to use resources.

Amazon VPC lets you logically isolate container tasks (Docker) or pods (Kubernetes) in a virtual network that you define. You can define security groups to create a virtual firewall between EC2 instances. Using an image scanning solution, you can detect vulnerabilities of container images or image dependencies. Security teams can perform scans on these containers or image dependencies and then publish pre-approved resources that developers can consume confidently.

Networking and Connectivity

Elastic Load Balancing (ELB) helps your users easily access your applications, using sophisticated load balancing algorithms natively integrated with AWS container services.

Amazon Global Accelerator also helps ensure that your application is highly performant and can be accessed by users worldwide and served from AWS Regions closest to them. AWS App Mesh enables fine-grained security and deep visibility to manage communication between container services.

Automation

Next is automation. You need automation tooling and an environment that removes the additional tasks such as manual code deployment. It can be done manually when your application is small, but imagine an application with hundreds or thousands of containers. You'll need to automate CI/CD to scale and optimize the risks of manual errors rapidly.

You can meet compliance, data gravity, and other business requirements by running workloads on your infrastructure with Amazon EKS Anywhere and Amazon ECS Anywhere. And you can still have fun with easy-to-use tools. You can use the same scalability, reliability, and fully managed control plane for your on-premises container workloads instead of installing and operating a local control plane. Run containerized data processing workloads at edge locations on your hardware to stay close to your end customers and reduce latency.

The automated environment removes the task of deploying code manually. AWS App Runner, AWS CodeCommit, AWS CodePipeline, and AWS CodeBuild are some of the go-to solutions to automatically perform tasks like creating a source code repository, configuring a CI/CD pipeline, or deploying and running containerized web applications and APIs at scale.

AWS App Runner is a fully managed application service that allows your developers to quickly deploy and run containerized web applications and APIs at scale. App Runner eliminates the need for infrastructure configuration and management. Instead, all you have to do is provide your source code, container image, or deployment pipeline. App Runner builds and deploys the web application, load balances network traffic, scales capacity up or down based on demand, monitors application health, and encrypts traffic by default. You also do not need prior container experience to use App Runner and take advantage of container portability, efficiency, and cost savings.

AWS Fargate & AWS CodePipeline implementation by Simform

We modernized the recruitment platform, which is one of our AWS customers, by implementing DevOps workflow and automation using AWS CodePipeline. AWS Fargate was also used to manage containers and scalability.

CHALLENGES:

- They needed to modernize the recruitment platform with modern technology and migrate to new systems with more customization and an analytics dashboard to measure the platform's performance and ROI.
- They wanted to implement a decoupled microservices architecture for enhanced performance, reliability, and integration with the existing application tracking system for a modern recruitment management platform.

SOLUTION:

- Our cloud engineering and DevOps team initially understood the existing platform architecture, codebase, and business logic and started building event-driven architecture.
- We used AWS Fargate to manage containers and scalability, which increased by 10 times in platform traffic without affecting the user experience.
- The data was centralized using Athena as the data lake for generating faster analytics and reports. In addition, other tech stacks such as AWS Glue, AWS AI Parser, and Elastic search are used for performance analytics and accurate information detection and search.

Observability and monitoring

To ensure that the microservices you deployed using containers are working fine, you need to observe and monitor your containers' health.

Based on the need and use cases, we leverage services like AWS App Mesh or Amazon CloudWatch Application Insights to get visibility into the compute infrastructure, collect the monitoring data, and set up the monitoring alarms. What's more, Docker container images are used to check the container's health and application, which will let us know if the container is working fine without any issues.

AWS App Mesh makes it easy to run services by providing consistent visibility and network traffic controls for services built across multiple types of compute infrastructure. AWS App Mesh removes the need to update application code to change how monitoring data is collected, or traffic is routed between services. It then configures each service to export monitoring data and implements consistent communications control logic across your application. This makes it easy to quickly pinpoint the exact location of errors and automatically reroute network traffic when there are failures or when code changes need to be deployed. AWS App Mesh uses the open-source Envoy Proxy, making it compatible with a wide range of AWS partners and open-source tools.

Docker container images allow you to verify if the container and application are healthy. With a simple health check command, a Docker file will check a container to confirm that it is still working. This process can detect when a web server is stuck in an infinite loop and unable to handle new connections, even though the server process is still running.

Amazon CloudWatch Application Insights supports container monitoring: You can now easily set up monitoring, alarms, and dashboards for your applications deployed in Amazon ECS, Amazon, EKS, and Kubernetes on EC2 containers running on AWS. Monitoring tier options are now available for capturing the metrics, telemetry, and logs for monitoring the health and wellness of applications running in containers on AWS.

Containerization—increasing delivery speed and changing customer experiences.

Portability:

Container packages applications with all their dependencies, and hence, you can run the application anywhere, on any platform.

Reduced costs:

With containers, you don't have to worry about infrastructure operations as multiple containers generally run on a single virtual machine.

Agility:

You get the maximum benefit of an agile environment as it works as an essential tool for integrating DevOps workflows. Using Container orchestration, you can create a container anytime and automatically shut it down when it's no longer needed.

Faster delivery & deployments:

Containerization divides your applications into microservices(small parts) and makes it much easier to deploy new features. It enables developers to work on segmented parts of the application. So developers can work faster on the issues or changes on the app module without affecting the whole application, hence the faster delivery.

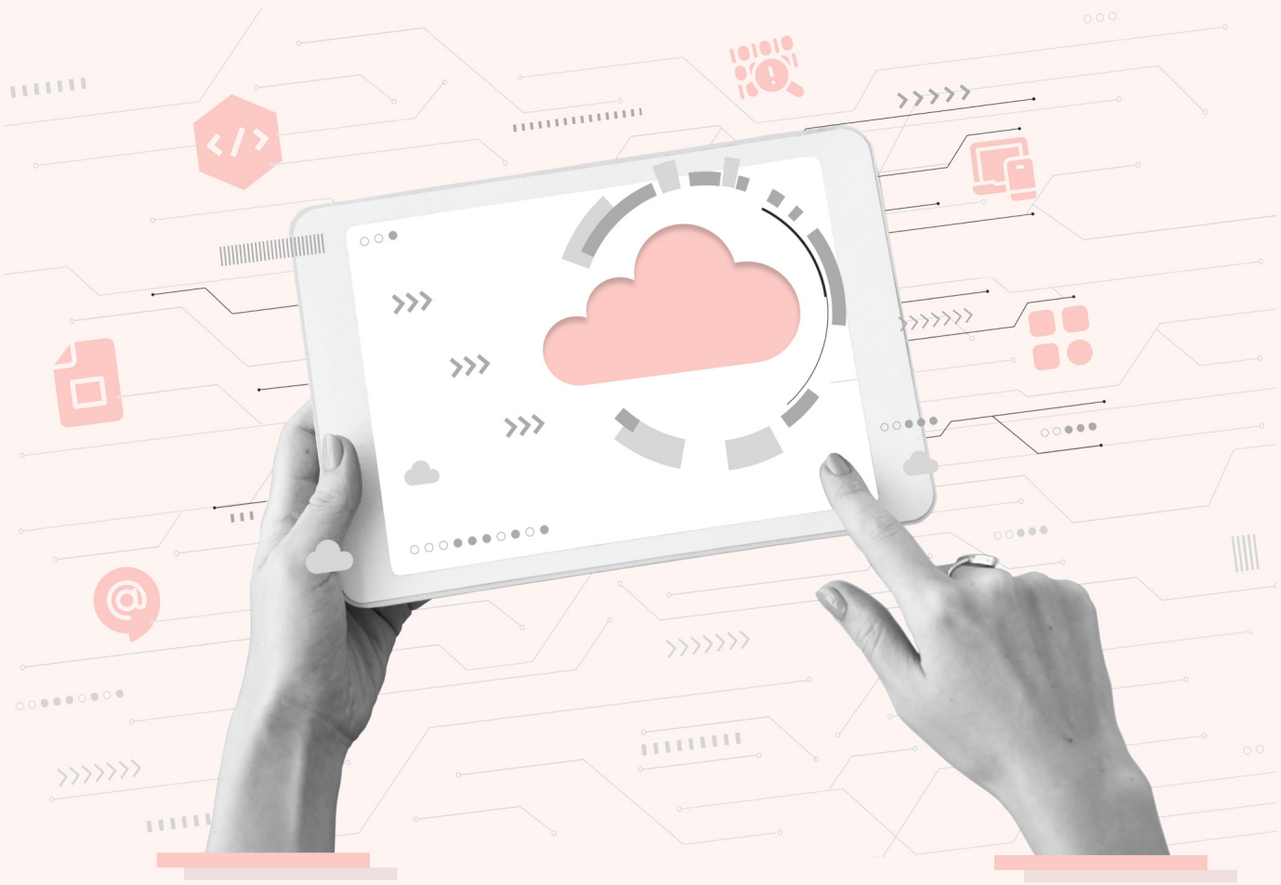
Enhanced security:

Containers put application parts in isolated packages, which puts an additional layer of security and runs applications in a self-contained environment.

Flexibility:

Containerized applications are always ready to switch from the bare-metal environment to virtual and vice versa depending on the needs, providing the needed flexibility to enterprises to move some services to the cloud and some resources to the bare-metal environment.

03



Modernizing your application portfolio with Serverless

Serverless translates to "without servers," though there are servers. But these servers are going to be managed by cloud service providers. Companies that aim to modernize their applications should simplify the application architectures and reimagine the workload implementations on serverless cloud services.

Serverless is one of the highest ROI-generating technologies among the many options available to modernize your application portfolio.

With serverless, developers focus more on building the product/ application than scaling the infrastructure, OS patching, maintenance, etc. Here, developers write the code and create functions that are executed with the suitable infrastructure by cloud providers. For modernizing applications with serverless, you need to consider rewriting and revising strategies as essential migration strategies.

Out of 7 Rs of AWS migration strategies, replatform, refactor, rewrite, and revise are crucial for modernizing applications using serverless.

While replatform and refactor calls for updating certain application parts with new code and updating some infrastructure components, a rewrite and revise approach requires writing completely new architecture and revisiting workload requirements. We at Simform practice combining **rewrite + revise** strategies to get the best results from serverless architecture and cloud services.

Rewriting applications from scratch reap most benefits from the cloud. It would reduce the burden of managing old code in a new environment, reduce configuration efforts, and reduce the overall monthly cloud costs.

What's more, serverless technology is a union of two concepts: Backend as a Service and Function as a Service.

Let's understand both concepts in brief:

i. **Function as a service:**

This concept runs on the idea of running backend code without managing your own servers; for example, AWS Lambda is an AWS's function as a service product.

With AWS Lambda, you can run code without provisioning or managing infrastructure. In addition, it can handle code execution requests, from dozens of events per day to hundreds of thousands per second.

This serverless, event-driven compute service lets you run code for any type of application and lets you trigger Lambda over 200 AWS services and software as a service.

Use cases of AWS Lambda

- Processes data as you need and scales automatically as per the requirements.
- Creates interactive mobile and web app experiences
- Establishes communication between decoupled services by serving peak demands by appropriate utilization of resources
- AWS Lambda handles such events in web or mobile apps that produce events. For example, files saved on the S3 bucket must be automatically processed, like creating a thumbnail version. Other examples include notifying SNS or sending out an email whenever new books are added to the library. The AWS Lambda function will be triggered to notify SNS whenever the new entries are added to the database.

Case Example by Simform

Simform recently implemented a data-retrieval mechanism for its AWS customer to retrieve millions of tweets from thousands of influencers.

CHALLENGES:

- The client wanted to design a microservices-based architecture and convert the millions of tweets data into an appropriate structure to enable strategic decision-making.

SOLUTION:

- We designed a system that captures the historical data in batches of 500 tweets every 2 hours, keeping the data up to date.
- AWS Lambda was utilized to automatically filter the original tweets from the retweets and reply data to derive the sentiments of the tweets.
- We also optimized the PostgreSQL database hosted on AWS RDS to reduce query response times as part of the application performance and display influencer's feed data.

iRobot case example

A consumer robots-making company.

iRobot, a consumer robots-making company, migrated its mission-critical application platform to Amazon Web Services Cloud, using about 25 AWS services. iRobot provides a wide range of robotic vacuum cleaners, with around 15 million iRobot models now being used globally.

The iRobot team leveraged serverless architecture and technology to provide additional connectivity and scalability in their products. They saw IoT and serverless as having a lot of synergy and alignment between them. To make their product iRobot Roomba 900 series truly an Internet of Things product, they integrated some AWS services and Alexa.

At the core of the platform, there's AWS Lambda and the AWS IoT platform. AWS Lambda runs code in response to events and provides function-based compute services for the serverless application. iRobot revealed that they could keep the costs of the cloud platform and were able to manage the cloud infrastructure with less than 10 people.

"To achieve a seamless smart-home experience in which the home, and the smart devices within it, responds to our daily needs autonomously, two things must happen," says Kehoe. "First, the burden of programming devices must be removed from the consumer. And second, the home needs to understand itself: what the layout of the home is, the location and purposes of each room, and where the home's various smart devices are located." - Ben Kehoe, a cloud robotics research scientist at iRobot.

ii. Backend-as-a-service

This serverless, cloud-service model outsources all the backend work that developers do. With this cloud execution model, developers only write the front-end code. Then, they need to integrate backend functionalities using APIs and SDKs provided by the cloud service providers. Examples of Backend as services include Firebase, ClouKit, AWS Amplify, etc.

Serverless-the transition that enables you to scale without limits!

No need to manage the servers:

Although serverless doesn't mean there aren't any servers, since the cloud vendors manage them, it lowers the organization's expenses and efforts to control the servers.

Scalability has no limit:

You're free to scale according to the traffic spikes on your web or mobile application, as serverless applications can automatically scale the resources according to the usage.

Faster deployments:

Releasing new features and application modules are a relief in the serverless cloud computing model. Developers don't need to hesitate to release new builds and versions as they no longer manage the backend configuration and are allowed to upload parts of the application without affecting the entire application or user experience.

Reduced costs:

The biggest benefit is the reduced architecture costs as the enterprises will be outsourcing the server and database management from the cloud vendors. So you don't need to invest in the DevOps team or any costs related to internal architecture.

More focus on building core products:

Since architecture is handled by third-party, developers have more time to focus on building great user experiences and meeting end users' expectations.

Speed:

Developers don't need to worry about allocating and estimating resources. Cloud vendors will manage the resources allocation, accelerating the core development work.

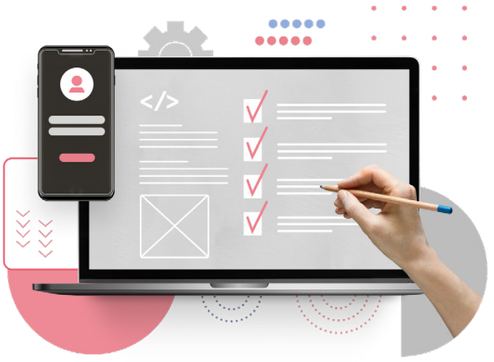
04

Migrating applications & workloads with AWS

The migration process starts with the application assessment. In AWS, the application migration process is divided into three phases: Assess, Mobilize, and Migrate & Modernize.

Assessment phase

The first phase focuses on evaluating the existing infrastructure, inventorying what you have, understanding the dependencies of applications, and listing the tools and technologies you'll need during the migration.



The assessment phase assesses your organization's readiness to adopt the cloud technology or operate within a cloud platform like AWS.

With the help of a Migration evaluator, AWS migration hub, and AWS cloud economics center tools, our teams at Simform help companies evaluate their on-premise resources and build a roadmap for running their applications in AWS. Therefore, it's important to have a fair idea about the total cost of ownership(TCO) based on the actual utilization of resources.

Mobilize phase

The second phase of cloud migration determines the migration costs and estimates how much you can save by choosing the AWS platform.

This is also a phase for developing and customizing a strategy for migrating your applications to the cloud. From this stage, organizations start analyzing their migration data and applications against the migration strategies: Rehost, replatform, rearchitect, repurchase, relocate, retire, and retain.

The mobilization phase includes developing a list of priority migration groups, defining the infrastructure data elements, creating a portfolio plan, and mapping out a migration plan.

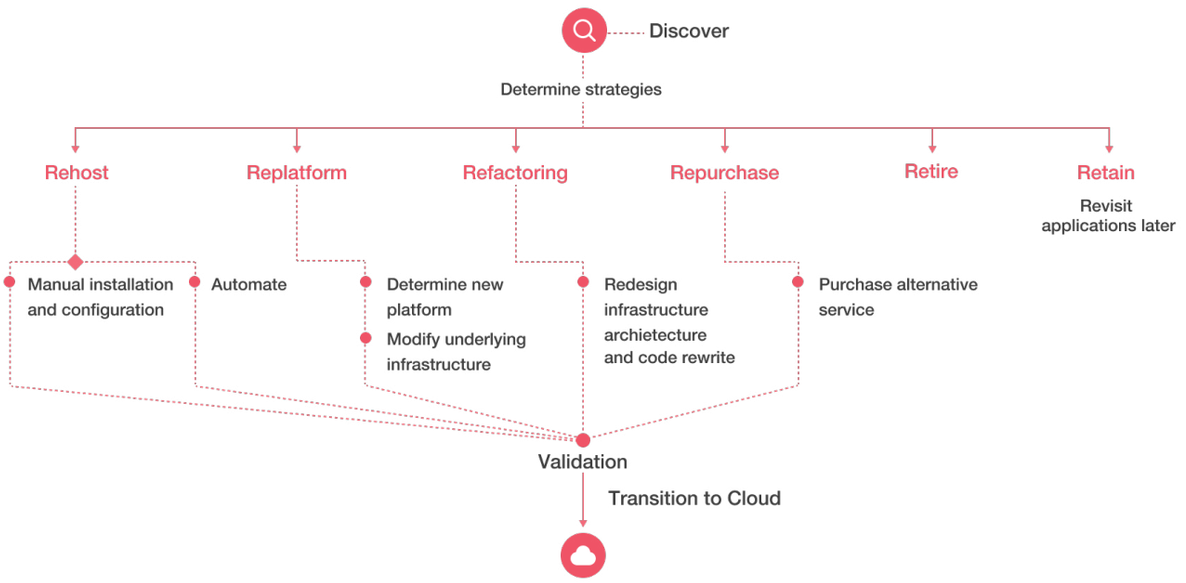
Migration phase

The migration phase decides the team structure and which teams will be working on certain tasks. For example, some organizations with a business case will rehost their applications or simply replatform the applications and workloads. AWS refers to these team formations as “migration factory”, a blueprint for implementing and scaling operations.

Generally, larger migrations with complex applications choose to refactor/rearchitect migration patterns.

The 7R methodology

Here we’ll choose routes for the application migration. Each application will have a different path depending on the application. Then, the actual dependency mapping will take place, and compare migration time and value provided by the types of methodology.



1 REHOST

Rehost or “lift and shift” is one of the most popular cloud migration strategies that deal with lifting your data assets and moving them to cloud services. Especially for large-scale enterprise migrations, rehost is the best option.

2 REPLATFORM

Replatform is also popularly known as the “lift, tinker, and shift” strategy. As opposed to rehosting, where you lift and shift the entire legacy application unchanged, re-platforming deals with app optimization to match the target configuration. It saves time and resources on managing database instances post-migration, especially when migrating to DaaS(Databases as a Service).

3 REFACTORING

Refactoring or Rearchitect strategy is the right choice for organizations looking to revamp the core architecture of monolithic applications. Especially if you plan to add more features that can help scale better, rearchitecturing of the application core services becomes vital. Prioritizing the services that you want to refactor helps with decision-making on the strategy’s inclusion early on and later during the migration process.

4 REPURCHASE

The repurchase strategy is also termed the “Drop and Shop.” It’s about dropping legacy applications and simply switching over to a cloud app without any migration. This strategy includes decommissioning the monolithic application and replacing it with a cloud-based version. Primarily, it entails a licensing change that allows you to use cloud-based services instead of on-premise licenses.

5 RETIRE

There are many applications in your organization that become obsolete over time. The retirement strategy is to decommission these applications and start from scratch to create cloud-based versions. It’s a strategy related to software and applications with monolithic architectures that are not compatible with modern APIs(Application Programming Interface).

6 RETAIN

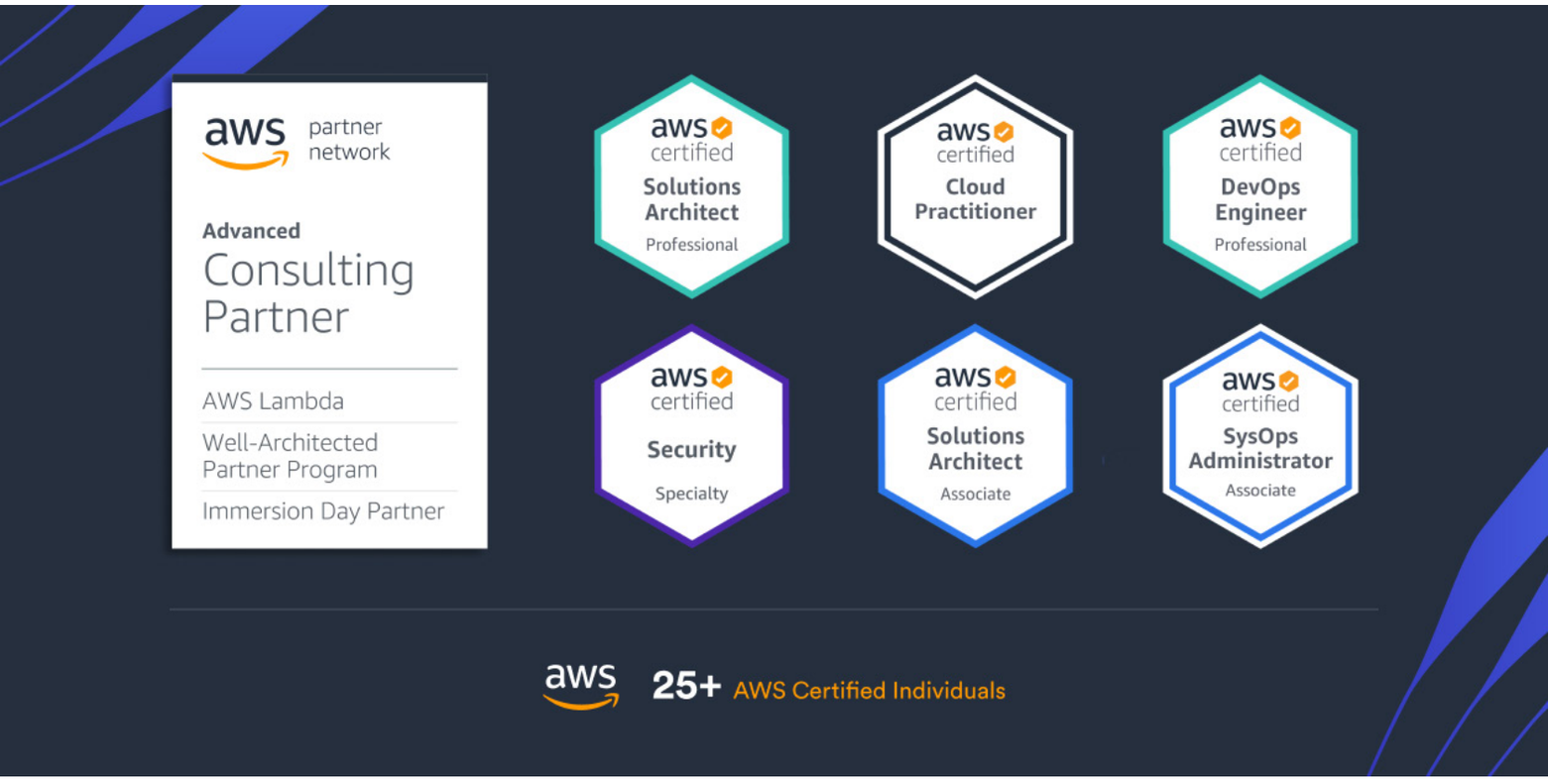
This cloud-migration strategy is all about retiring some obsolete apps and migrating the rest to the cloud. It also relates to a hybrid cloud model where you can migrate assets in a phased manner. With this approach, you can move some cloud assets while still retaining others at the on-premise infrastructure.

Modernizing applications with AWS and Simform

Simform and AWS help you drive top-tier performance in cloud infrastructure. We help you embrace the cloud ecosystem to achieve new levels of success. Simform's pre-vetted team of cloud experts and AWS certified professionals know how to modernize legacy applications on AWS by optimizing workloads and choosing technologies that lead the charge of modern application development. We're committed to leveraging cutting-edge technologies and utilizing the AWS best practices to use the full potential of the AWS platform.

Simform, as an AWS Advanced Consulting Partner, has AWS certified Lambda competency and partner programs, including well-architected framework review and immersion days to help our customers make the most out of the AWS platform.

Simform works with enterprises to modernize application portfolios by refactoring, rearchitecting, or revising applications aligned with the business objectives. We help our customers modernize applications for scalability, greater operational efficiency, enhanced performance, and overall reduced costs.



We are Simform!

With over 10+ years of experience under our belt, we are more than ready to supercharge your project with extraordinary code. 10 years ago, Simform was one person. Today, we're over 600+ people strong and growing.

Simform is a custom software development powerhouse. Let's get in touch to discuss your next project!

Contact Us

- Managed software engineering teams
- Quality Engineering and Testing
- Cloud native development and modernization
- DevOps