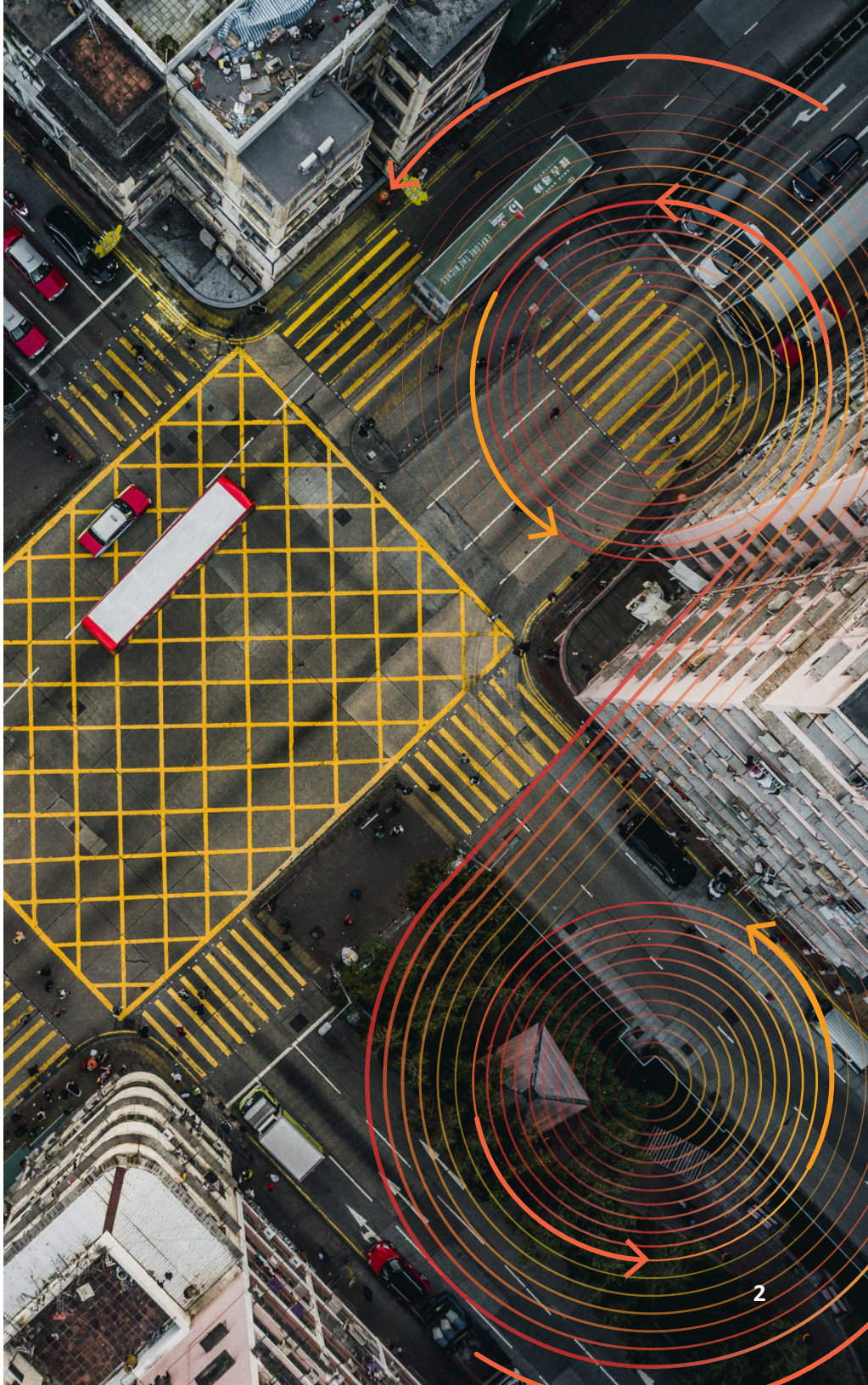

aws modern apps
Build modern applications on AWS
Manage less. Build fast. Innovate more.

Modern applications are changing how you deliver customer value

In the next few years, organizations will build over 500 million new apps, more than the number developed in the previous 40 years combined.[1] Many are building them the hard way as they struggle to find a balance between managing technology and delivering new features.

While the cloud promises agility, that doesn't happen automatically. As organizations look to accelerate innovation, get more out of their data, and build new customer experiences, they need to modernize the way they build and operate applications. Modern applications are built with a combination of modular architecture patterns, serverless operational models, and agile developer processes.

In this eBook, we'll guide you through the three pathways that will help lay the foundation for modern application development in your own organization. We'll also explore how modern application development with AWS can help your organization innovate, reduce costs, accelerate time to market, and improve reliability.



[1] *IDC PlanScape: Connected Cloud Architectures to Enable the Future of Digital Infrastructure, October 2020*

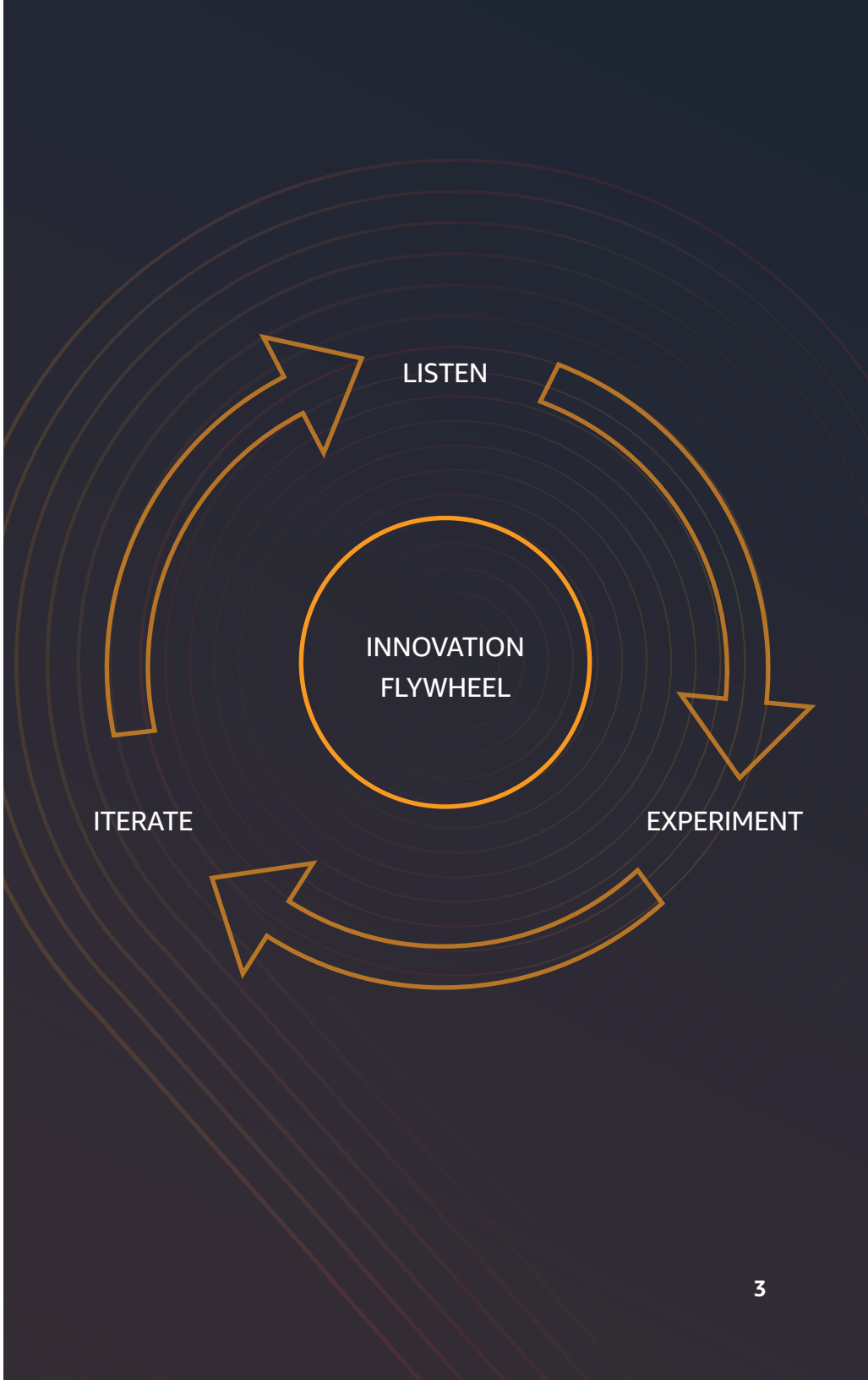
Innovation means listening to your customers

In a recent Vision Report, *Digital Rewrites the Rules of Business*, Forrester Research defines the customer-centric mindset of a digital innovator. The core mission of these modern disruptors is to:

"…Harness digital assets and ecosystems to continually improve customer outcomes and, simultaneously, improve operational excellence…by applying digital thinking to customer experiences, operations, ecosystems, and innovation."

Focusing on your customer means making business decisions by working backward from your customer's point of view. It means constantly evolving products and services to better deliver the outcomes that delight customers. And it means listening to what your customers truly care about so that you can continue inventing and iterating on their behalf. This is called the "innovation flywheel."

The basic idea is that the driver for any innovation begins with customer demand, improves with customer feedback, and repeats constantly (and profitably) until the demand changes and the whole cycle begins again. The faster your teams can get your own innovation flywheel spinning, the better you'll be able to build modern applications and the more you will stand apart from competitors.



Building modern applications on AWS will get you to market faster. By speeding up the build-and-release cycle and offloading operational overhead, developers can quickly build new features. You'll increase innovation with a modular architecture that lets teams experiment with individual application components without risking the entire application. By automating test procedures and monitoring at every stage of the development lifecycle, you'll improve reliability. And you'll improve total cost of ownership (TCO) with a pay-for-value pricing model that reduces the cost of over-provisioning or paying for idle resources.

To build modern applications, you may need to reconsider the foundation on which they are built. While shifting this architecture may be dramatic at an organizational level, the process doesn't need to be brutal. Many organizations take an inspired leap to build new modern apps in the cloud, but plenty of others take a hybrid approach, often taking a team-by-team and workload-by-workload journey, moving opportunistically one step at a time.

50%

of information and communications technology is predicted to be directly allocated for digital transformation by 2023

67%

of executives believe they must pick up the pace to remain competitive

90%

of new applications are predicted to be cloud-native by 2025

Through our experience building applications for Amazon.com and from serving millions of AWS customers, we've observed three pathways that organizations can take for translating their vision of application modernization into a reality, generating value for business in the process.

1. Re-platform to managed container services. Organizations already running containers on-premises or thinking about moving applications to containers can re-platform those workloads to container services on AWS to simplify operations and reduce management overhead costs, such as orchestration and infrastructure provisioning.

2. Build new apps on serverless architecture. As organizations build new applications or features, we recommend that they use serverless technologies and purpose-built databases to maximize agility, as well as advanced development tools to accelerate development.

3. Transform to a Modern Dev+Ops model. To create a cultural shift to build modern applications at scale, organizations can leverage DevOps services and tools while maintaining a high bar on security and governance.

We'll explore each pathway in more detail, demonstrating how each can help lead to increased agility, lower costs, and building better apps that support business success. While you can modernize applications from any starting point, the outcome needs to be the same: applications that are secure, reliable, scalable, and quickly available for your customers and partners at the onset.

Three pathways to modern application development

Modern application development is a powerful approach to designing, building, and managing software in the cloud. This proven approach increases the agility of your development teams and the reliability and security of your applications, allowing you to build and release better products faster. From our experience helping organizations of every kind build applications, we've identified three solution pillars of modern application development to help you on your journey toward modernization.

Pathways to modern applications

1. **Re-platform to managed container services**
2. **Build new, secure apps on serverless architecture**
3. **Transform to a Modern Dev+Ops model**

Re-platform to managed container services

Containers are a lightweight and portable way to run and deploy applications. Containerizing existing applications is often a first step in an organization's modernization journey. If you are considering moving your applications to containers, you would benefit from re-platforming those workloads to AWS managed services like Amazon Elastic Kubernetes Service (Amazon EKS) or Amazon Elastic Container Service (Amazon ECS) with AWS Fargate. A managed container service helps to reduce operational burden while improving scalability, reliability, security, and availability. With managed container services on AWS, you no longer have to worry about managing containers. Instead, you can focus your resources on education for the upskilling you need to develop modern applications with serverless computing.

CUSTOMER SNAPSHOT

Vanguard®

> *"AWS Fargate Spot has reduced our unit costs and reinforced our business case to migrate to AWS. We're delivering more value for our dollars each month with this optimization. Returning value to our shareholders through increased efficiency is core to our company-wide mission."*
>
> – Tim Treston, Senior Manager, Cloud Business Office, Vanguard

Since beginning its migration to AWS in 2015, The Vanguard Group has seen significant benefits. For example, Vanguard's use of AWS services removed the need for its IT department to manage servers. As a result, developers have more time to build innovative, new microservices and enhance current applications, increasing Vanguard's speed to market from three months to 24 hours.

But speed to market is not the only benefit the company has seen. Vanguard opted to use Amazon Elastic Container Service (Amazon ECS), a fully managed container orchestration service, alongside AWS Fargate. AWS Fargate is a serverless compute service that removes the need to provision and manage Amazon Elastic Compute Cloud (Amazon EC2), a web service that provides secure, resizable compute capacity in the cloud. And by taking advantage of a new purchase option for AWS Fargate, the financial services firm reduced its unit costs by 50 percent.

See the full story »

Modernizing app development is the adoption of services, practices, and strategies that enable developers to build more agile applications. And with the speed and reliability of modern infrastructure, developers can deliver secure apps that scale from prototype to millions of users automatically, so they can innovate and respond to change faster.

Many modern applications are built serverless-first, a strategy that prioritizes the adoption of serverless services so customers can increase agility throughout the application stack. With serverless technologies, you no longer have to manage physical servers and you'll benefit from automatic scaling, built-in high availability, and a pay-for-value billing model. Instead of worrying about managing and operating servers or runtimes, you can focus on product innovation while enjoying faster time to market.

In addition, front-end web and mobile tools and services can be built on top of AWS. The reliability of this infrastructure helps brands deliver secure, highly available apps that can scale automatically across the globe.

Key considerations for building scalable modern apps

Architectural patterns: microservices

Although your monolithic app might be easy to manage today, challenges often arise as you grow, including how to distribute ownership of the app across your teams. You can build a strong culture of ownership but still struggle to scale up if your application architecture includes hard dependencies that prevent teams from taking ownership of the final product. This is why we recommend building microservices architectures for apps that grow and change rapidly. Microservices are the architectural expression of a culture of ownership—they neatly divide complex applications into components that a single team can own and run independently.

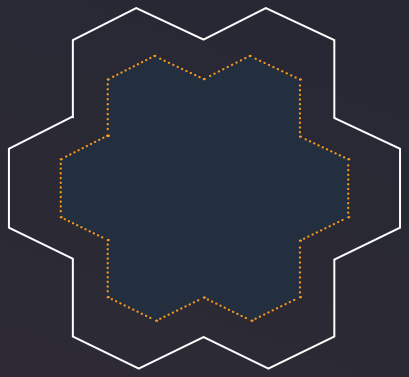
With a monolith, you have many developers all pushing changes through a shared release pipeline, which causes friction at many points of the lifecycle. During development, engineers need to coordinate their changes to make sure they're not breaking someone else's code. In order to upgrade a shared library to take advantage of a new feature, you need to convince everyone else to upgrade at the same time—a tough ask! And if you want to quickly push an important fix for your feature, you still need to merge it with changes in progress.

After development, you also face overhead when you're pushing the changes through the delivery pipeline. Even when making a one-line change in a tiny piece of code, engineers need to coordinate their changes ahead of time, merge their code, resolve conflicts within releases, rebuild the entire app, run all of the test suites, and redeploy once again.

With a microservices architecture, an application is made up of independent components that run each application process as a service. Services are built for business capabilities, and each service performs a single function. Because it runs independently and is managed by a single development team, each service can be updated, deployed, and scaled to meet the demands for specific functions of an application. For example, an online shopping cart can be used by many more users during a sale. Microservices communicate data with each other via well-defined interfaces, using lightweight APIs, events, or streams. Our customers are increasingly relying on event-driven architectures—those in which actions are triggered in response to changes in data—to improve application scalability and resiliency while also reducing costs.

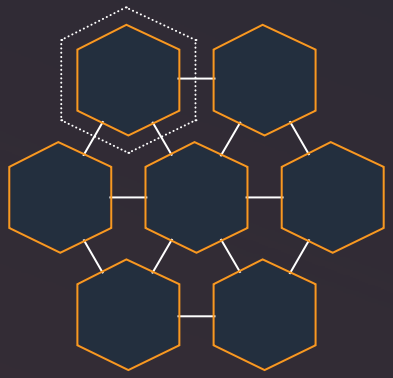
EVERYTHING VS. ONE THING: TWO TYPES OF APPLICATIONS

Monolith apps



- Do everything
- Single app
- Must deploy entire app
- One database
- Organized around technology layers
- State in each runtime instance
- One technology stack for entire app

Microservices



- Do one thing
- Minimal function services
- Deployed separately, interact together
- Each has its own datastore
- Organized around business capabilities
- State is externalized
- Choice of technology for each microservice

CUSTOMER SNAPSHOT

Centrica is a British multinational energy and services company that was looking to decrease its costs and increase its agility by changing the way its application was architected. The company opted to refactor to a microservices architecture and adopt a serverless strategy to achieve those goals.

In order to change the organization, Centrica set up a serverless working group with representative teams and started building a pilot together.

Building on that success, other teams in the organization have now also adopted the approach—serverless is now common across the organization.

With serverless, Centrica is able to see and respond to customer issues in real time, something it had not been able to do before.

Watch the video »

Serverless operational model

As serverless as possible

As your architectural patterns and software delivery processes change, you will probably want to adopt an operational model that enables you to offload any activity that isn't a core competency of your business. To gain agility that can enable rapid innovation, we recommend building microservices architecture, operating and deploying software using automation for things like monitoring, provisioning, cost management, deployment, and security and governance of applications. Choosing a serverless-first strategy—opting for serverless technologies wherever possible—enables you to maximize the operational benefits of AWS.

With a serverless operational model, you can build and run applications and services without provisioning and managing servers. This eliminates server management, provides flexible scaling, enables you to pay only for value, and automates high availability. This model lets you build and manage the aspects of your application that deliver customer value without having to worry about the underlying detail.

Whether you are building net-new applications or migrating legacy, building with serverless primitives for compute, data, and integration will enable you to benefit from the most agility the cloud has to offer.

How do we define serverless at AWS?

When we say serverless, we mean it's the removal of the undifferentiated heavy lifting that is server operations. This is an important distinction because it allows you to focus on the building of the application rather than the management and scaling of the infrastructure to support the application. The four tenets of a serverless operational model are:

1. **No server management** – There is no need to provision or maintain any servers. There is no software or runtime to install, maintain, or administer.
2. **Flexible scaling** – Your application can be scaled automatically or by adjusting its capacity through toggling the units of consumption (e.g., throughput, memory) rather than units of individual servers.
3. **Pay for value** – Instead of paying for server units, pay for what you value—consistent throughput or execution duration.
4. **Automated high availability** – Serverless provides built-in availability and fault tolerance. You don't need to architect for these capabilities since the services running the application provide them by default.

A serverless operational model is ideal for high-growth companies that want to innovate quickly. Serverless enables teams to move even faster and keep a laser focus on the activities that differentiate your business so you can speed up your innovation flywheel.

Leveraging AWS Lambda and managed container services on AWS

With the rise of containers and serverless computing, instances are no longer your only cloud computing option. Choosing the optimal compute for your modern application starts with exploring several questions. Does self-managing infrastructure improve your business results? Do you have the expertise to do it? And will the extra effort ultimately drive value?

Increasingly, customers are choosing to offload server management by adopting container services like Amazon ECS and Amazon EKS or event-driven serverless compute services like AWS Lambda.

In reality, most customers use a combination of both. About 80 percent of AWS containers customers have also adopted AWS Lambda.[2] Leveraging both options has its benefits, including fully managed services that have deep integration with AWS infrastructure, support for a wide range of use cases, abstraction from complexity, and a broad ecosystem of partners.

[2] *Datadog State of Serverless, 2020*

So how do you frame the decision?

Customers choose AWS Lambda when they have teams focused primarily on writing code and no limitations on existing instances or container platforms. AWS Lambda offers the maximum abstraction from infrastructure, and so it enables customers to release fastest, which is why new applications are a great fit for AWS Lambda.

Customers often choose containers when they have existing containers investments, open-source preferences for Kubernetes, or specific requirements for managing or configuring infrastructure. Containers are the most popular way to package code and are a great choice for modernizing legacy applications.

	AWS Manages	Customer Manages	
AWS Lambda Serverless functions	Data source integrations Physical hardware, software, networking, facilities Provisioning	Application code	
AWS Fargate Serverless containers	Container orchestration, provisioning Cluster scaling Physical hardware, host OS/kernel, networking, facilities	Application code Data source integrations Security config and updates, network config	Management tasks
Amazon ECS/ Amazon EKS Container management as a service	Container orchestration control plane Physical hardware, networking, facilities	Application code Data source integrations Security config and updates, network config, firewall	Work clusters Management tasks
Amazon EC2 Infrastructure as a service	Physical hardware, networking, facilities	Application code Data source integrations Scaling Management tasks	Security config and updates, network config Provisionining, managing scaling and patching of servers

CUSTOMER SNAPSHOT

AbiBird®

> *"We believe our technology running on AWS will allow the growing number of elderly people to live independently at home—where they are happier."*
>
> – Robin Mysell, CEO, ATF Services and AbiBird

AbiBird is a wholly owned division of ATF Services, an Australian group of companies with 350 employees and contractors and 60 branches in Australia and New Zealand. AbiBird offers a service consisting of infrared sensors for homes, which help monitor the activities of elderly residents through a smartphone-based app.

AbiBird was running on Microsoft Azure and found it was raising far too many support tickets with its cloud provider to keep its service running. The company needed more stability and scalability. AbiBird moved to AWS and now uses a combination of compute services based on its needs. It uses AWS Lambda on the backend due to its ease of use and scalability, as well as for the managed nature of the service.

AbiBird also uses container services on AWS, hosting a public API on Amazon ECS and leveraging AWS Fargate to run its containers without the hassle of administering its own fleet of virtual machines.

Since launching this system on AWS in 2019, AbiBird has not had to raise a single support ticket, which enables the company to run efficiently with a minimal amount of overhead support.

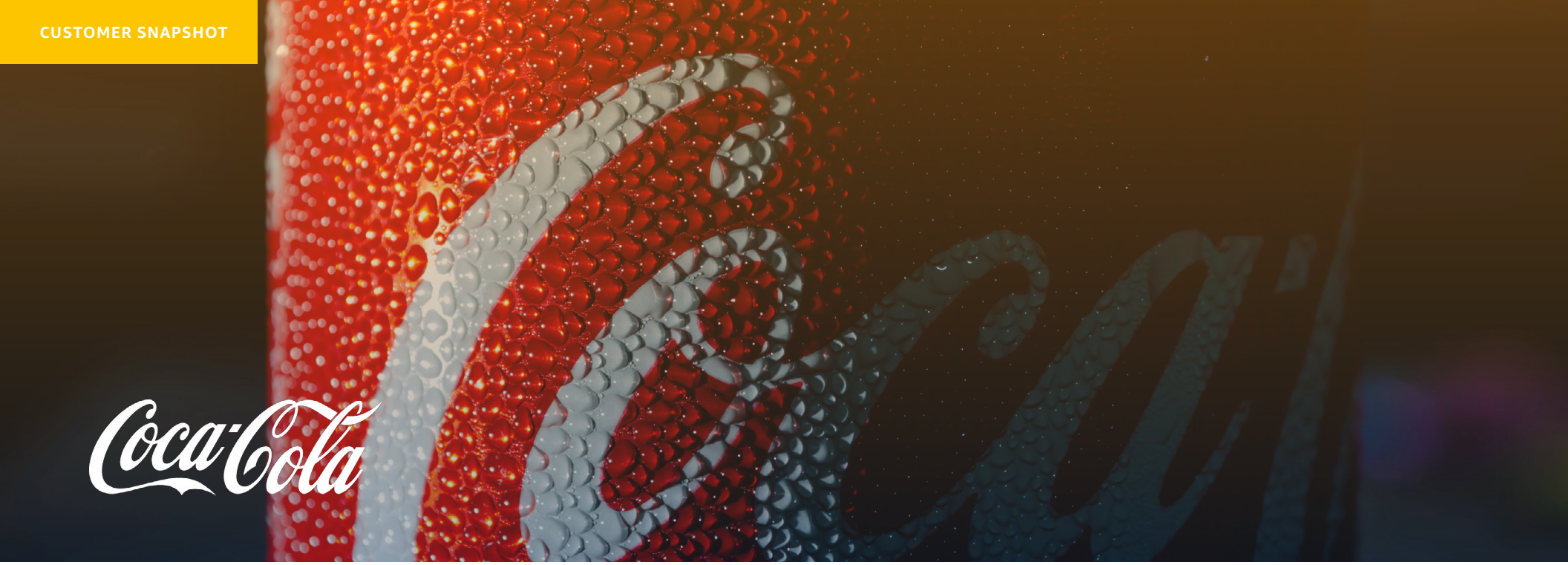
See the full story »

CUSTOMER SNAPSHOT

One of America's most iconic food brands, Taco Bell has over 7,000 restaurants in the U.S. During the COVID-19 pandemic, Taco Bell needed to rapidly shift to meet consumer demand for delivery. According to Vadim Parizher, vice president of engineering and analytics, Taco Bell runs nearly all of its infrastructure on Amazon Web Services (AWS) and uses serverless on AWS to focus less on managing servers and more on building business logic and data transformations to deliver real-time menu and restaurant information to its delivery partners. “We have a menu that is very complex and has to be shared across multiple digital channels. Serverless fits that model really well,” says Parizher. By using serverless services, Taco Bell is able to save on upfront costs, start small and only pay for what is needed, and automatically scale as it consumes more services.

Watch the video »

CUSTOMER SNAPSHOT



When COVID-19 hit, consumer habits literally changed overnight. Coca-Cola responded rapidly with a no-touch drink dispensing experience to go along with its innovative Freestyle drink dispensers. Coca-Cola opted to build with AWS Lambda, and as a result, its team was able to focus on the application rather than security, latency, or scalability. Because with AWS Lambda, that's all built in. The new application launched in just 100 days, and now more than 52,000 machines have the touchless capability.

Watch the full story »

"Low latency is essential to the user experience, which is why we're committed to a serverless solution on AWS."

– Michael Connor, Chief Architect, Coca-Cola Freestyle Equipment Innovation Center

Modern Dev+Ops is the combination of cultural philosophies, practices, and tools that enables an organization to quickly and safely develop software, release it to production, and maintain its target availability and performance.

AWS has identified a set of common, broadly accepted practices that, when adopted, provide a mechanism for building a high-performing DevOps organization. This approach takes a simple idea—continuous improvement—and applies it to everything in the DevOps lifecycle, from planning and code writing to deployment and monitoring.

We call this approach Modern Dev+Ops, and it's centered around bringing developers and operations closer by sharing operational tasks like compliance, observability, resilience, and infrastructure earlier into the development process and enhancing it with artificial intelligence and machine learning (AI/ML).

Developer agility: abstraction, automation, and standardization

Microservices architectures make teams agile and enable them to move faster, which means you're building more things that need to get released—great! However, you won't get new features to your customers faster if your build-and-release process does not keep up with your team's pace. Traditional development processes and release pipelines are slowed mainly by manual processes and custom code. Custom code is ultimately a long-term liability because it introduces the possibility for errors and long-term maintenance. Manual steps—from code changes and build requests to testing and deploying—are the greatest drag on release velocity. The solution involves abstraction, automation, and standardization.

In order to speed the development process, abstract away as much code as possible, particularly the lines of non-business logic code, required to develop and deliver production-ready apps. One way to do this is to employ frameworks and tooling that reduce the complexity of provisioning and configuring resources. This gives developers an ability to move quickly while also enforcing best practices for security, privacy, reliability, performance, observability, and extensibility throughout the development process. Development frameworks give you confidence that your architecture will support your business growth long term.

By defining your software delivery process through the use of best-practice templates, you can provide a standard for modeling and provisioning all infrastructure resources in a cloud environment. These "infrastructure as code" templates help teams get started on the right foot because the template provisions the entire technology stack for an application through code rather than using a manual process.

Through automation, you can create a repeatable motion that speeds up your software delivery lifecycle. Automating the release pipeline through continuous integration and continuous delivery (CI/CD) helps teams release high-quality code faster and more often. Teams that practice CI/CD ship more code, do it faster, and respond to issues quicker. In fact, according to the Puppet 2020 State of DevOps Report, teams that employ these practices have a failure rate that is five times lower, a commit-to-deploy rate that is 440 times faster, and a rate of deployment that is 46 times more frequent.[3] Most notably, teams that practice CI/CD spend 44 percent more of their time creating new features and code instead of managing processes and tools.

CI/CD pipelines have become the new factory floor for building modern applications. At Amazon, we started using CI/CD to increase release velocity, and the results were dramatic—we have achieved millions of deployments a year and we grow faster every year. To help companies benefit from our experience, we built a suite of developer tools based on the tools we use internally so our customers can deliver code faster.

A bit more detail:

Continuous integration - (CI) is a software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. Continuous integration most often refers to the build or integration stage of the software release process and entails both an automation component (e.g., a CI or build service) and a cultural component (e.g., learning to integrate frequently).

Continuous delivery - (CD) is a software development practice where code changes are automatically prepared for a release to production. Continuous delivery expands on continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage.

Learn how Amazon automates safe, hands-off deployments »

[3] *Puppet 2020 State of Devops Report, 2020*

CUSTOMER SNAPSHOT

With AWS, Lululemon Athletica can stand up development environments in minutes instead of days, automate its environments, and enable continuous integration and deployment. The Canadian company sells yoga-inspired apparel and other clothing at more than 350 locations throughout the world. Lululemon runs its dev and test environments—as well as an upcoming mobile app—in the AWS Cloud.

Lululemon decreased its time to build new production accounts from two days to a few minutes, using AWS CloudFormation templates and AWS CodePipeline. With that increased agility, Lululemon dev teams can now experiment and get to the best solutions rather than having to settle for what they have resources committed to.

See the full story »

"Any continuous integration and deployment pipeline should be automated, easy to manage, and discoverable, and that's exactly what we get using AWS. We get a level of simplicity and transparency we simply couldn't have in our previous on-premises environment."

– Sam Keen, Director of Product Architecture, Lululemon

HyperTrack is a self-serve cloud platform for live location tracking through apps. When it launched in late 2015, HyperTrack needed to build a platform that could scale automatically to meet their anticipated growth without reducing the time their developers spent on building new features.

HyperTrack opted to use AWS Amplify for a mobile development framework and a serverless architecture in order to scale up and down automatically without engineering intervention.

As a result, the company has realized a 30 percent cost savings compared to the architecture they were using before we switched to serverless. A big part of that savings comes from not needing operational resources to focus on server management. HyperTrack saves 40 hours of work, every single week, while managing millions of events.

See the full story »

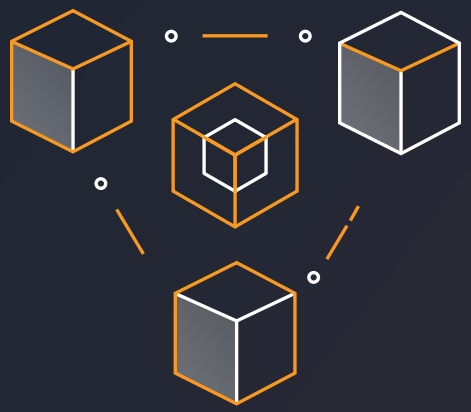
Creating a culture of ownership: manage less, innovate more with Modern Dev+Ops

Innovation ultimately comes from people, and so enabling your people to deliver better customer outcomes is where modern application development starts. We use the concept of "products, not projects" to describe how this impacts team structure. Simply stated, it means that the teams that build products are responsible for running and maintaining them. It makes product teams accountable for the development of the whole product, not just a piece of it.

After more than a decade of building and running the highly scalable web application Amazon.com, we've learned firsthand the importance of giving autonomy to our teams. When we gave our teams ownership of the complete application lifecycle, including taking customer input, planning the roadmap, and developing and operating the application, they became owners and felt empowered to develop and deliver new customer outcomes. Autonomy creates motivation, opens the door for creativity, and develops a risk-taking culture in an environment of trust.

While embracing a culture of ownership is not inherently technical, it remains one of the most challenging aspects of modern application development. Empowering teams to become product owners involves changing the mindset of your organization, the structure of your teams, and the work for which they are responsible.

For most organizations, IT falls into one of two camps. It is viewed as a strategic competitive weapon or, more commonly, as a cost center necessary to support business growth.



Building a culture of innovation

1. **Start with the customers –** Every innovation should start with a customer need and ultimately delight your customers. Prioritize relentlessly to focus on customer demand.

2. **Hire builders and let them build –** Remove any obstacles that slow the process of building and releasing products and features for customers. The faster you iterate, the faster your flywheel spins.

3. **Support builders with a belief system –** Don't pay lip service to innovation—live and breathe innovation in all areas of the business, from leadership to sales to support.

Manage less, innovate more

Modern applications create competitive differentiation by enabling rapid innovation. By adopting services, practices, and strategies that underscore speed and agility, you can shift resources from business as usual to differentiating activities with deep customer value. You can experiment more and turn ideas into releases faster. You can foster an environment where builders spend more time building and less time managing. Modern applications are how organizations, including Amazon, innovate with speed and agility.

Why build modern applications on AWS?

Faster to Market	By speeding up the build-and-release cycle and offloading operational overhead, developers can quickly build new features. Automated test and release processes reduce error rates, so products are market-ready faster.	**See the proof:** **Urbanbase launches services 20x faster with AWS**
Increase Innovation	With a modular architecture, changes to any individual application component can be made quickly and with a lower risk to the whole application, so teams can experiment with new ideas more often.	**See the proof:** **iRobot uses AWS Lambda and the AWS IoT platform to manage its Roomba robotic vacuum cleaners**
Improve Reliability	By automating test procedures and monitoring at every stage of the development lifecycle, modern applications are reliable at deployment. Any issues can be evaluated and addressed in real time.	**See the proof:** **Siemens decreases customer control system alerts by 90 percent and reduces infrastructure costs by 85 percent**
Improve TCO	With a pay-for-value pricing model, modern applications reduce the cost of over-provisioning or paying for idle resources. By offloading infrastructure management, maintenance costs are also lower.	**See the proof:** **Save up to 80 percent on app maintenance with AWS Lambda**

Start your application modernization journey

Replatform to Managed Container Services

80% of all containerized applications running in the cloud run on AWS* 84% of all Kubernetes workloads in the cloud run on AWS*

Resources
Amazon ECS workshop
Amazon EKS workshop
AWS AppRunner workshop

Recommended Training (Classroom)
Running Containers on Amazon Elastic Kubernetes Service (Amazon EKS)

Recommended Training (Online)
Amazon Elastic Container Service (ECS) Primer

Build New Modern Applications with Serverless Technologies and Tools

Save up to 80% time on maintenance and approximately 70% on development when adopting a serverless-first strategy for building modern applications**

Resources
Innovator Island - Serverless web application development workshop. Build a serverless web app video tutorial.

Recommended Training (Classroom)
Advanced Developing on AWS

Recommended Training (Online)
Architecting Serverless Solutions

Transform to a Modern Dev + Ops Model

60% of teams with higher evolved DevOps practices fully remediate security vulnerabilities in less than a day#

Resources
The Amazon Builder's Library
AWS DevOps services

Recommended Training (Classroom)
DevOps Engineering on AWS

Recommended Training (Online)
Getting started with DevOps on AWS

Learn more about building modern applications on AWS →

Talk to an expert to implement the best practices of modern application development →

Connect with an AWS partner to accelerate your modernization projects →

aws modern apps

* Source: Nucleus Research
** Source: Deloitte
# Source: 2020 DORA state of DevOps Report