Modern applications for startups

Reinventing how startups build, iterate, and scale faster with AWS.

Driving customer value faster with modern applications

Digital transformation has dramatically impacted the way startups deliver value and the rate at which they make changes to their products and services. Every company is becoming a technology company—either building products that are the technology itself or building products that are heavily influenced by and dependent on technology. To thrive in this new world, startups must create better experiences—and they must do it at an increasingly rapid pace.

Many startups are innovating faster by changing the way they design, build, and manage applications through the use of modern applications. Modern applications, which are built with microservices, are a shift from traditional monolith (non-modular) architectures. Modern applications increase the agility of small teams as well as the reliability, security, and scalability of your applications. They can be developed faster and scale quickly to potentially millions of users, have global availability, manage petabytes or even exabytes of data, and can respond in milliseconds to customer feedback.

In this eBook, you'll learn how modern applications enable and support a culture of ownership and drive innovation and why developers are increasingly using microservices architecture to store and manage data, as well as to set guardrails for the monitoring, provisioning, deployment, security, and governance of applications—and much more. And through the examples of real customer stories, you'll see how organizations of every kind are building modern applications on Amazon Web Services (AWS) to bring products to market faster, at a lower cost, and at virtually any scale.

"Invention requires two things: 1. The ability to try a lot of experiments, and 2. not having to live with the collateral damage of failed experiments."

Andy Jassy, CEO, Amazon Web Services

95%

of new applications are predicted to be cloud-native by 2025[1]

67%

of executives believe they must pick up the pace to remain competitive[2]

55%

Investment levels for 2022–2024 are expected to be $6.3 trillion and are 55% of all ICT investment by the end of 2024[3]

How modern applications accelerate business success

To build a modern application, you may need to reconsider your application's architectural patterns, operational model, and software delivery process. While these shifts are 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 others take a stepwise approach, one app at a time.

We observed common characteristics of modern applications through our experience building applications for Amazon.com as well as from serving millions of AWS customers. Applications with these characteristics enabled our customers to increase agility, lower costs, and build better apps that support the success of their businesses. While you can modernize applications from any starting point, the outcome is the same: applications that are more secure, reliable, scalable, and quickly available for your customers and partners.

This guide covers the following topics:

- Digital innovators
- Characteristics of modern applications
- Culture of ownership
- Architectural patterns: microservices
- Computing in modern applications: containers and AWS Lambda
- Data management: purpose-built and decoupled
- Developer agility: abstraction, automation, and standardization
- Operational model: as serverless as possible
- Management and governance: leveraging programmatic guardrails
- Start building modern applications today

[1] "Gartner Says Cloud Will Be the Centerpiece of New Digital Experiences," Gartner, November 2021
[2] "Embrace the Urgency of Digital Transformation," Gartner, 2017
[3] "IDC FutureScape: Worldwide Digital Transformation 2022 Predictions," IDC, October 2021

DIGITAL INNOVATORS

It's all about the customer

What's the definition of "innovation"? Today, it's listening and responding 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."[4]

Focusing on your customer means making business decisions by working backward from that customer's point of view. It means constantly evolving products and services to better deliver the outcomes that delight customers. And finally, 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 constantly repeats (and profitably) until the demand changes and the whole cycle begins again. The faster your teams can get your own innovation flywheel spinning, the stronger your differentiation will be in the market and the more you will stand apart from your competitors.

[4] "Digital Rewrites the Rules of Business," Forrester, December 2020

MODERN APPLICATIONS IN PRACTICE

The benefits of modern application development

Modern application development is a powerful approach to designing, building, and managing software in the cloud. As we mentioned, 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 all types of organizations build applications, we've identified seven characteristics of modern applications that digital innovators rely on for success.

Characteristics of modern applications

1. Culture of ownership
2. Architectural patterns: microservices
3. Computing in modern applications: containers and AWS Lambda
4. Data management: purpose-built and decoupled
5. Developer agility: abstraction, automation, and standardization
6. Operational model: as serverless as possible
7. Management and governance: leveraging programmatic guardrails

A CULTURE OF OWNERSHIP

Creating a culture of ownership with product teams

Innovation ultimately comes from people. So it stands to reason that enabling your people to deliver better customer outcomes is at the core of developing modern applications. We use the concept of "products, not projects" to describe how this impacts team structure. It means that the teams that build products are responsible for running and maintaining them. Simply stated, 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 within 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 they are responsible for delivering. As you'll discover, modern applications are the driving force of a culture of ownership—and innovation.

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.

MICROSERVICES

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. Upfront during development, engineers need to coordinate their changes to make sure they're not breaking someone else's code. 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.

Post-development, you also face overhead when 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 the test suites, and redeploy once again.

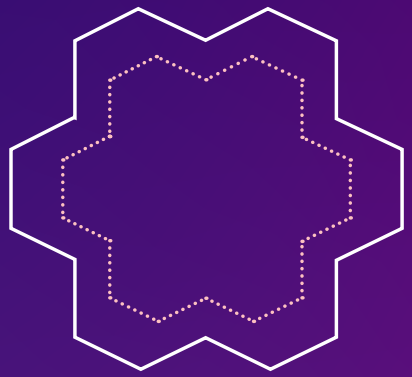
Microservices architecture: Scalability, resiliency, and cost-efficiency

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 demand 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 data store
- Organized around business capabilities
- State is externalized
- Choice of technology for each microservice

CUSTOMER STORY

Bridestory

Bridestory is a web and mobile application developed to inspire brides and help them connect with wedding vendors. The site has up to 700,000 visitors each month and is expanding globally. Two years following its launch, the startup was experiencing growing pains due to its monolithic architecture. Slow iteration and difficulty in deploying and scaling were interfering with time to market—a top priority. So Bridestory migrated to a container-based infrastructure to improve agility and decrease time to market. Doni Hanafi, chief technology officer at Bridestory, worked with his team to split the application into smaller services and set up Docker containers with Amazon Elastic Container Service (Amazon ECS). Prioritizing time to market and reliability, Bridestory aimed to deploy smaller features every few days to lower the rate of failure. Now Bridestory can empower its developers to conduct deployments every day instead of every three weeks, speeding the brand's time to market threefold.

Read the full story ›

bridestory

"Since the end of 2018, we have been hitting our metrics. Previously, we were averaging three weeks to launch one big feature, whereas now our developers are empowered to conduct small releases daily, with less than a one percent failure rate."

Doni Hanafi, CTO, Bridestory

DATA MANAGEMENT

Data management: Purpose-built and decoupled

The shift to microservices architectures has a big impact on how organizations store and manage data. If each microservice is interacting with a monolithic database, the database is still a single point of failure. Modern applications are built with decoupled data stores, each a part of a single microservice. This may seem like a radical departure from a traditional application architecture, but considering that a monolithic application poses scaling and fault tolerance challenges as it grows, we need to apply those same principles to a database.

Databases have evolved significantly over the past few years. Traditional applications such as ERP, CRM, and ecommerce typically used relational databases because these applications required recording transactions and storing structured data in gigabytes and occasionally terabytes in size. But approaches to building applications and application requirements themselves have changed.

Today, applications need to work with terabytes and petabytes of data, support millions of customers globally, and process millions of requests per second with very low latency. To support these needs, developers are increasingly building applications using microservices architectures and choosing relational and nonrelational databases that are purpose-built to meet their application's specific needs, such as storing key-value pairs and documents.

CUSTOMER STORY

Amenity Analytics

Founded in 2015, Amenity Analytics develops cloud-based text analytics solutions using natural language processing (NLP) and machine learning (ML). For its customers—which include some of the world's largest insurance companies, banks, investment firms, and more—the company's software generates top-line trends and scores around the ideas it uncovers, then pinpoints the specific articles and sentences referenced. At heart, the startup is an NLP company. Its algorithms sift through tremendous amounts of data, processing around a million pieces of text formats. The data goes unexploited as a source of insight due to the difficulties of analyzing text meaningfully.

Amenity's customers expect that the company will unearth actionable data points, so they use state-of-the-art linguistic pattern matching techniques. Most companies that use NLP—a branch of machine learning focused on understanding people-based linguistic data versus the well-defined outputs of computers—run commonly known algorithms. However, by devising and creating whatever algorithms Amenity Analytics wants, they can operate them in a way that has an edge on other companies. Predictably, this system of complex NLP classifications means heavy CPU workloads, which are handled by a serverless-first AWS architecture. As a result, Amenity Analytics has built its entire stack on AWS tools.

Read the full story ›

"This gives our data scientists the ability to run many experiments quickly and cheaply. Overall, moving to serverless NLP has reduced our costs of analysis by 90 percent and time of analysis by 95 percent. [Additionally,] maintenance and code complexity are reduced when we use serverless patterns, and that translates to faster development cycles."

Roy Penn, VP of Engineering, Amenity Analytics

Leveraging containers and AWS Lambda

Moving from monolith to microservices affects the way you package and run code—changing the way we compute in modern applications. Instances are no longer the only option. Modern applications now use either containers or serverless event-driven compute service AWS Lambda. Much of this change arose from a desire on the part of developers to manage dependencies within each service through a pipeline, which in turn resulted in new ways of packaging applications.

Choosing between instances, containers, and AWS Lambda boils down to how much your enterprise requires flexibility versus simplicity—both of which make possible the seamless integration with other services and features. There are natural trade-offs the more you move toward simplicity. And one of the more common problems that we continue to try to resolve is how to move toward simplicity without giving up all the flexibility.

Increasingly, our customers choose containers and AWS Lambda for compute. Containers have emerged as the most popular way to package code, making this a great way to modernize legacy applications. Containers offer excellent portability and flexibility over application settings. AWS Lambda offers the most simplicity so that the only code you write is business logic, and it enables you to benefit from the most agility the cloud has to offer.

DEVELOPER AGILITY

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. That's great! However, you won't get new features to your customers faster if your build-and-release process does not keep up with the pace of your team. 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.

To speed the development process, abstract away as much code as possible, particularly the lines of non-business logic code that are 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 in the long term.

Deeper dive...

Continuous integration

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. CI 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).

Learn how Amazon automates safe, hands-off deployments ›

Continuous delivery

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

DEVELOPER AGILITY

Enabling developer agility with CI/CD

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. That's 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 more quickly. In fact, according to the Puppet "2017 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.[5] 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've achieved millions of deployments per year and 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.

[5] "2017 State of DevOps Report," Puppet by Perforce, 2017

Teams that practice CI/CD spend

44%

more of their time creating new features and code instead of managing processes and tools

CUSTOMER STORY

Branch Insurance

Founded in 2018, Branch Insurance offers the ability to buy bundled home and auto insurance online at a price match lower than existing insurance companies can give. Users can enter their name and address to get a price for bundled home and auto insurance.

Managing insurance policies at scale is no easy feat. Each contract is heavily regulated, complicated, and long. Filings frequently contain more than 10,000 pages of content, and governments require a full audit for any changes that are made. To solve the complexity of managing insurance policies and accelerate to market, Branch turned to AWS and its purpose-built services.

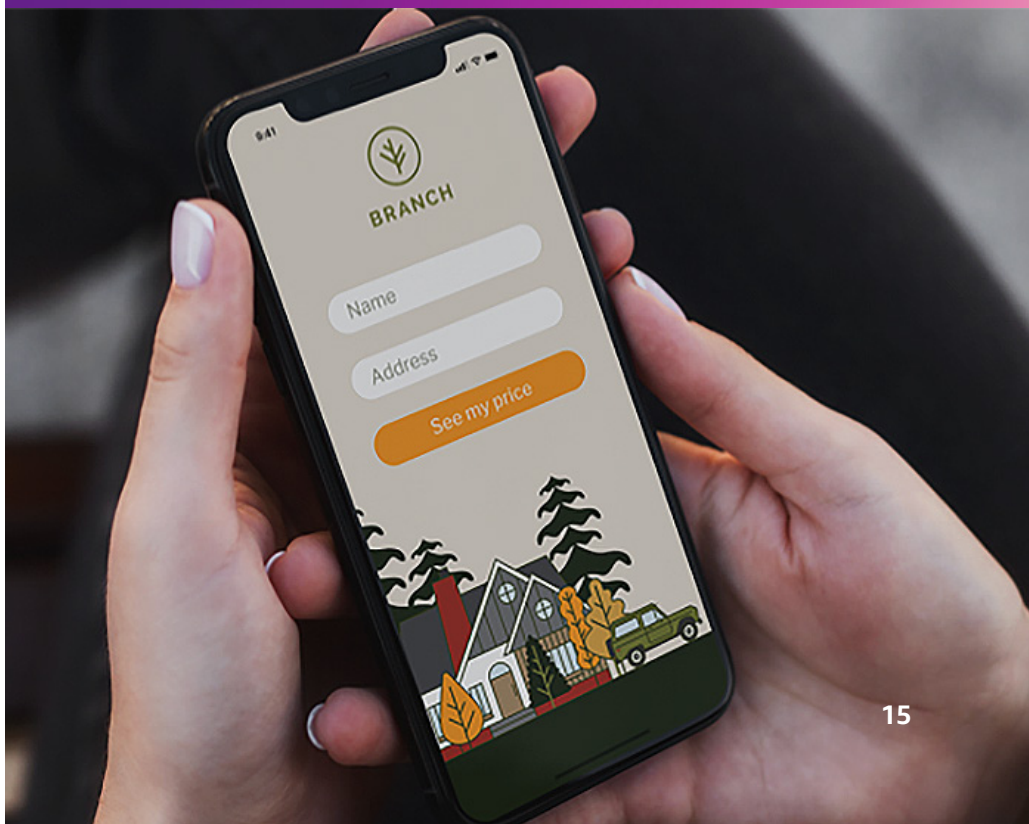
Branch used AWS AppSync, AWS Lambda, and Amazon DynamoDB to increase development velocity. AWS AppSync is used to shuttle traffic that contains authentication information to AWS Lambda functions. This controls what privileges a user should be given. Amazon DynamoDB is used to store records, as you can set triggers within its streams. As a result of using AWS for its policy management functions, Branch Insurance was able to launch its MVP to market months faster than they had originally anticipated.

Read the full story ›

BRANCH

"It's been great to have services like AppSync, Lambda, and DynamoDB that work together to take a lot of the development time off our plates. We love these architectural models and didn't see them in other offerings in the market."

Steve Lekas, Co-Founder & CEO, Branch Insurance



SERVERLESS OPERATIONAL MODEL

Operational model: Be as serverless as possible

As your architectural patterns and software delivery processes change, you will likely 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 and operating and deploying software using automation for operations such as monitoring, provisioning, cost management, deployment, and application security and governance. 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 applications, 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?

We think of serverless as the removal of the undifferentiated heavy lifting that is server operations. This is an important distinction, as 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 and 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.

CUSTOMER STORY

Stedi: Achieving “zero touch operations” with serverless on AWS

Stedi scales to meet unpredictable usage volume and operates without traditional QA and infrastructure teams. The startup—which has raised $71 million in funding—built its modern, fully self-service global EDI platform for B2B transactions on a serverless architecture using more than 30 AWS services, including AWS Lambda, AWS Cloud Development Kit (AWS CDK), and AWS Amplify. Now, Stedi can maximize long-term development velocity to deliver a pioneering global network for businesses, one that can automate trillions of B2B transactions and ultimately achieve its mission of processing every B2B transaction worldwide.

The startup practices what it calls “zero touch operations,” or designing for operational efficiency at any scale, with no buttons to press or knobs to turn. With serverless architecture, Stedi can remove as much code from its technical balance sheet as possible and take advantage of a pay-per-use pricing model on AWS.

Read the full story ›

stedi

“Stedi helps businesses offload and automate as much of the B2B transaction process as possible, and we’ve been able to find similar benefits by leveraging a serverless-first approach for our architecture. Our goal internally is zero touch operations with no knobs to turn or buttons to press. Using AWS’s offerings, we’ve built a system that is designed to scale ad infinitum with minimal toil.”

Zack Kanter, Founder & CEO, Stedi

MANAGEMENT & GOVERNANCE

Management and governance: Leveraging programmatic guardrails

Managing your organization securely, legally, and safely is first priority for nearly every company. But often, strong governance becomes a series of checkpoints that slow down innovation. When it comes to operations, organizations have two options: Move slowly and safely or move at lightning speed without limitations but introduce serious risks to the application and the business.

The good news is that you can have it both ways by implementing guardrails. Companies are increasingly adopting the concept of guardrails, which enables teams to operate quickly without becoming a risk to the business.

So, what are guardrails? Guardrails are mechanisms, such as processes or practices, that reduce both the occurrence and blast radius of undesirable application behavior. Usually expressed as code, guardrails are established and standardized through a central team and delivered and updated to teams of builders in automated and programmatic ways. This is where operators, or those who establish, manage, and maintain the multiple flexible pieces of distributed architectures, are key.

There are a few different areas where we tend to see operators working. In a modern application world, operators set guardrails for the monitoring, provisioning, deployment, cost management, and security and governance of applications. Operators have a role in not only establishing the boundaries, rules, and best practices for each of these areas but also in building automated solutions and packaging and deploying these guardrails throughout the organization.

Governance philosophies

Free for all	Guardrails	Central control
Fast dev time, but risk to legal and app reliability ↓ Chaos	Fast time and low risk to the business ↓ Win-win	Low risk but very slow to release Dependencies and time lags ↓ Delays

Common uses of guardrails

Monitoring
- CPU utilization
- Database throughput
- Business processes

Security & compliance
- CPU utilization
- Database throughput
- Business processes

Deployment
- Time window
- Toolsets available
- Size or timing of test releases

Provisioning
- Access permissions
- Resource availability
- Configuration

Cost management
- Resource costs
- Resource utilization
- Spend run rates

Boosting application security posture with guardrails

The concept of guardrails is essential to application security as well. It isn't surprising that the "2017 State of DevOps Report" by Puppet states that guardrails, or integrating security deeply into the software delivery lifecycle, double team confidence in their security posture. The report also suggests that firms at the highest level of security integration can deploy to production on demand at a significantly higher rate than firms at all other levels of integration.

Deeper dive...

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes.

DevSecOps is the philosophy of integrating security practices within the DevOps process. DevSecOps involves creating a "security-as-code" culture with ongoing, flexible collaboration between release engineers and security teams.

CUSTOMER STORY

Coinbase

Coinbase is a digital currency wallet and platform company based in San Francisco, California. To protect its customers from attacks, Coinbase engineers must be able to deploy updates and new features quickly, reliably, and securely for all company systems, with some software deployments needing 20 hours or longer to finish. Adopting a serverless architecture anchored by AWS Lambda and AWS Step Functions enables the company to increase the rate of successful mission-critical deployments from 90 percent to 97 percent.

With this approach, Coinbase was able to substantially reduce the complexity of its architecture. This, in turn, improves visibility for the team to monitor and troubleshoot, ultimately reducing the number of trouble tickets about failed deploys. Serverless architecture also simplifies the process of auditing, enabling a single audit trail for all deployments.

Read the full story ›

coinbase

"With deployers built on AWS Step Functions and AWS Lambda, our engineers can move code into production safely. The upshot is that we can release new features more often, respond quicker to security threats, and more easily achieve our SLAs. This adds up to an even better, more secure, customer experience."

Graham Jenson, Sr. Infrastructure Engineer, Coinbase

CLOSING SUMMARY

Final notes: Build modern applications today

Modern applications create competitive differentiation by enabling rapid innovation. By changing your architectural pattern, operational model, and software delivery process, you can shift resources from standard operations to differentiating activities. You can experiment more and turn ideas into releases faster. You can foster an environment where builders spend more time building. Simply put, modern applications are how organizations, including Amazon, innovate with speed and agility.

Modern applications on AWS can help to:

- Speed up time to market
- Increase innovation
- Improve reliability
- Reduce costs

AWS is trusted by millions of customers around the world—including the fastest-growing startups, largest enterprises, and leading government agencies—to power their infrastructure, increase their agility, and lower costs. AWS offers a comprehensive portfolio of services to support your business as you develop modern applications.

Any starting point. Any application. Any innovation. AWS is how modern application development happens. Start your modernization journey today with AWS.

Get started ›

AWS services for modern application development

Developer tools

Mobile/front-end development
AWS Amplify

Source code management
AWS CodeCommit

Pipeline automation
AWS CodePipeline

Infrastructure as code: TypeScript, Java, Python, C#
AWS CDK

Build and test automation
AWS CodeBuild

Artifact repository
AWS CodeArtifact

Infrastructure as code: YAML/JSON
AWS CloudFormation

Deployment
AWS CodeDeploy

Compute

Serverless Event-driven compute
AWS Lambda

Serverless containers
AWS Fargate

Kubernetes container orchestration
Amazon Elastic Kubernetes Services (Amazon EKS)

Container orchestration
Amazon ECS

Integration

GraphQL API
AWS AppSync

Message queues
Amazon Simple Queue Services (Amazon SQS)

Pub/sub and push notifications
Amazon Simple Notification Service (Amazon SNS)

Visual workflows
AWS Step Functions

REST API
Amazon API Gateway

Service mesh
AWS App Mesh

Data streaming
Amazon Kinesis

Event bus
Amazon EventBridge

Data stores

Object storage
Amazon Simple Storage Service (Amazon S3)

Key-value database
Amazon DynamoDB

Relational database
Amazon Aurora

File system
Amazon Elastic File System (Amazon EFS)

Document database
Amazon DocumentDB

In-memory database
Amazon ElastiCache