Modern application development

Reinventing how your business delivers value

The time is right for modern application development

Digital transformation has dramatically impacted the way companies deliver value and the rate at which they make changes to their products and services. Every company is becoming a technology company—companies are increasingly building products that are the technology itself or heavily influenced by technology. In order to compete in this new world, businesses must create better digital products, and they must do it at an increasingly rapid pace.

Many companies are innovating faster by changing the way they design, build, and manage applications, through what we call modern application development. Modern applications increase the agility of your teams and the reliability, security, and scalability of your applications. They automate or abstract away operational overhead to enable teams to spend more time building business logic. They facilitate an environment in which experimentation thrives because small failures don't lead to system outages. And building this way also requires a fundamental shift in the way you approach the creation of value.

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

–Andy Jassy, CEO, Amazon Web Services

67%

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

56%

Say improvements increased profits within the first year

42%

Have adopted a "digital first" business posture

To build modern applications, 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 plenty of others take a step-wise 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 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
  1. Culture of ownership
  2. Architectural patterns: microservices
  3. Computing in modern applications: containers and AWS Lambda
  4. Data management: purpose-built databases
  5. Release pipelines: standardized and automated
  6. Operational model: as serverless as possible
  7. Security: everyone's responsibility
- Start building modern applications today

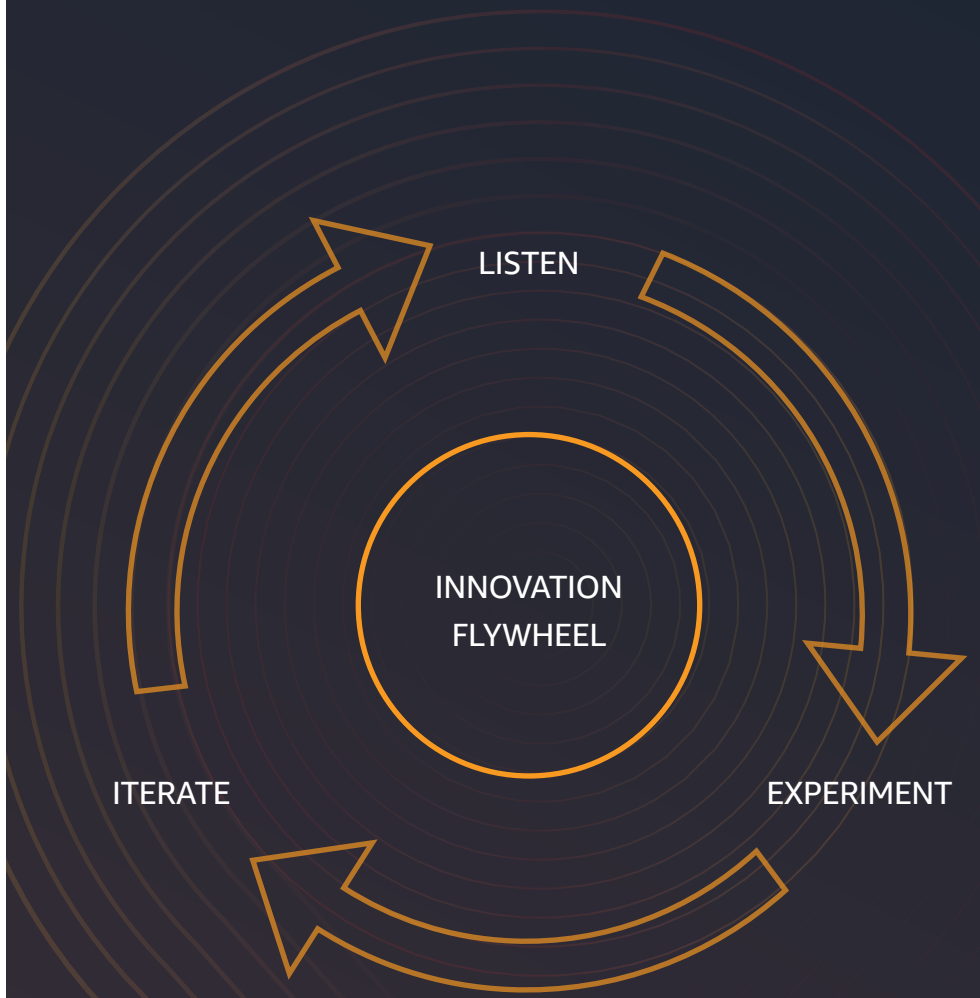
Innovation means listening to your customers

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."[1]

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, 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 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 stronger your differentiation will be in the market and the more you will stand apart from competitors.



[1] Digital Rewrites The Rules Of Business, The Vision Report In The Digital Business Playbook, Nigel Fenwick and Ted Schadler, February 26, 2018, ©2018 Forrester Research, Inc.

To innovate at scale, put technology at the center of your business

With customers guiding your innovation, you have a clear direction for the evolution of your technology and your business. Consider these examples of AWS customer innovation from the Forrester Research report on how digital transformation "rewrites the rules of business":

- **Digital marketplaces** - Organizations can now build digital platforms that match buyers and sellers in a two-sided market, which enables them to deliver outcomes faster and, in many cases, more cost-effectively. Airbnb, Salesforce, Amazon.com, and many other companies are creating digital marketplaces that connect buyers and sellers—often without owning the underlying assets.
- **Direct-to-customer engagement** - Using digital platforms to reach and serve customers directly helps businesses stay relevant. By reaching and serving their market directly, companies like Dollar Shave Club are focused on meeting customer expectations. Recently acquired by Unilever, the online provider sells razor blades directly to customers on a monthly subscription.
- **Digital products and services** - Companies are generating entirely new revenue streams by creating new digital products and services. For example, Netflix is streaming original content. Innovators are also using APIs and digital platforms to deliver a core competency: Stripe provides payment services that allow its customers to move money quickly and safely over the web.
- **Insight services** - Service-oriented businesses are using data to enhance core competencies and new-business success rates. The Financial Times improved the accuracy of decision-making for everything from editorial to sales by leveraging actionable data collected from numerous sources.

These new-business models can change the economics of your business by creating new revenue streams and building competitive differentiation. This kind of industry disruption is possible when you have the freedom to experiment without living with the collateral damage of failure.

Modern architectures reduce the impact a failure has on the overall application. This means teams are able to try out new ideas more often. When dev teams can try things, they will try things. And it's easier to try things with modern applications because new software delivery processes enable developers to release new ideas faster and more frequently.

"We believe that every company must become a digital innovator. Every CEO must continuously reinvent their business with evolving technology at the core—or watch while their customers defect and their markets are disrupted."[1]

[1] Digital Rewrites The Rules Of Business, The Vision Report In The Digital Business Playbook, Nigel Fenwick and Ted Schadler, February 26, 2018, ©2018 Forrester Research, Inc.

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 common characteristics of modern applications that digital innovators rely on for success.

Characteristics of modern applications

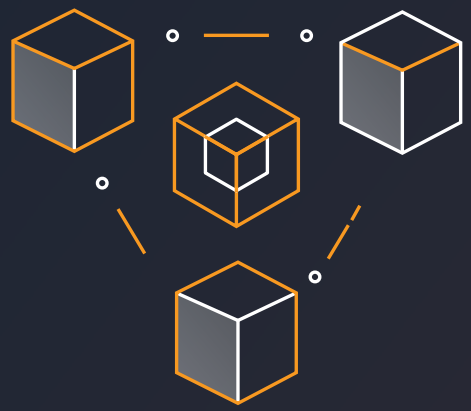
1. Culture of ownership
2. Architectural patterns: microservices
3. Computing in modern applications: containers and AWS Lambda
4. Data management: purpose-built databases
5. Release pipelines: standardized and automated
6. Operational model: as serverless as possible
7. Security: everyone's responsibility

Creating a culture of ownership

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 road map, 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.



Building a culture of innovation

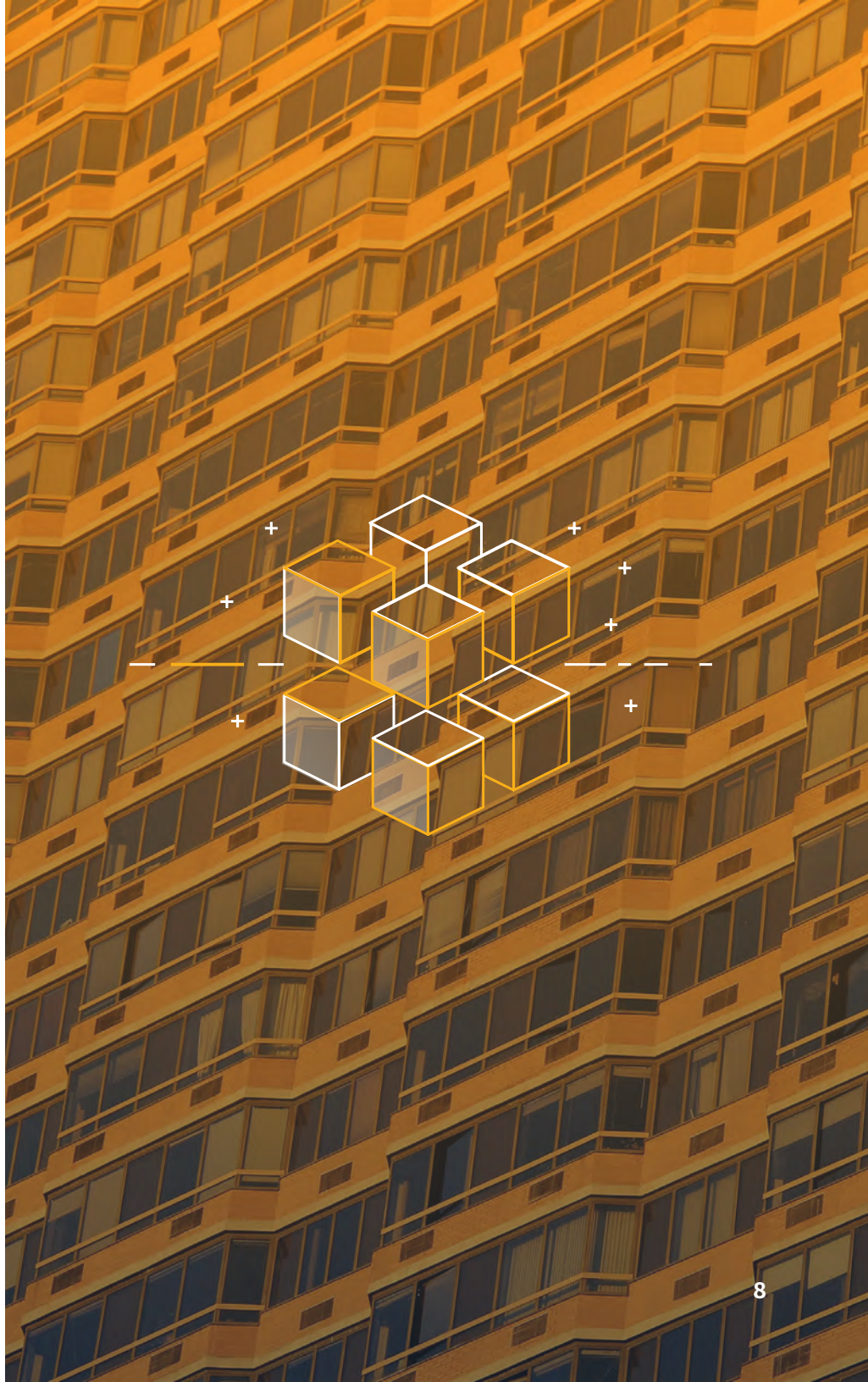
1. **Start with the customers –** Every innovation should start with a customer need and ultimately delight your customers. Prioritize relentlessly to focus on customer demand.
2. **Hire builders and let them build –** Remove any obstacles that slow the process of building and releasing products and 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.

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.

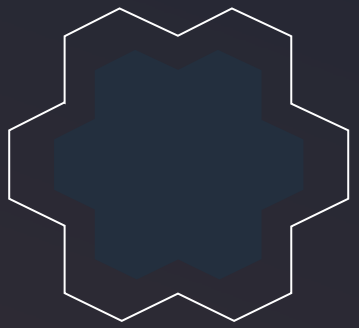
After development, you also face overhead when you're pushing the changes through the delivery pipeline. Even when making a one-line change in a tiny piece of code, engineers need to coordinate their changes ahead of time, merge their code, resolve conflicts within releases, rebuild the entire app, run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

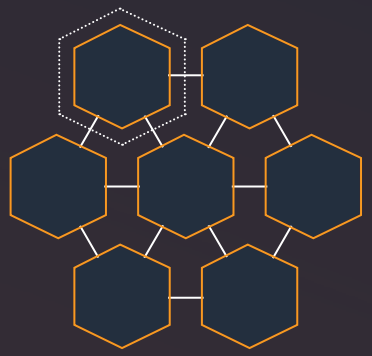
EVERYTHING VS. ONE THING: TWO TYPES OF APPLICATIONS

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 datastore
- Organized around business capabilities
- State is externalized
- Choice of technology for each microservice

Thanks to its unwavering focus on connecting people with local businesses, Yelp has amassed one of the most loyal communities on the internet. The company made the decision to replace its monolithic subscription-billing process, which handles a vitally important part of its operation: recurring purchases across more than 100,000 advertising accounts. Yelp worked with AWS to safely transform the business-critical billing platform to a serverless environment based on modern cloud-based microservices.

The Yelp team utilized AWS Step Functions to re-factor its monolith safely and gradually, carving off small tasks one at a time while adding new steps to the workflow. This provided higher levels of observability for tracking the progress of each step, consistency from built-in error handling, efficiency from parallel frameworks, and flexibility of running tasks regardless of whether they are on-premises, in containers, or in functions.

See the full story >

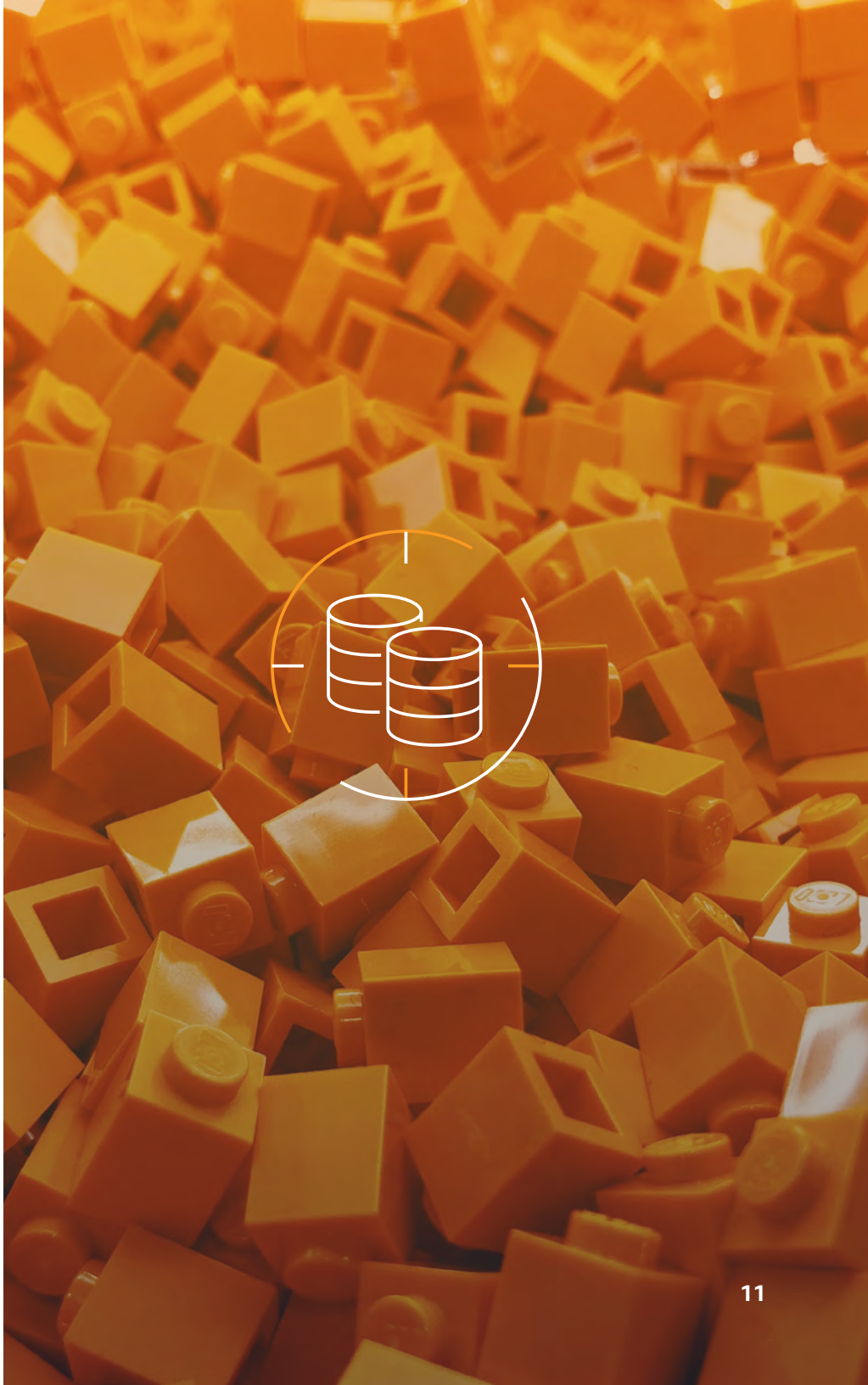
"We can flexibly pick individual tasks and move them over to AWS Lambda, and Step Functions stays with us the whole way from monolith to serverless, helping us quickly build more efficient and resilient systems."

–Dave Marin, Data-mining Engineer, Yelp

Data management: purpose-built databases

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. That is why modern applications are built with decoupled datastores, each of which is part of a single microservice. This may seem like a radical departure from a traditional application architecture, but when we acknowledge that a monolithic application poses scaling and fault tolerance challenges as it grows, we need to apply those same principles to a database.

It is also impractical for a single database to meet the specific needs of a set of varied microservices. By decoupling data along with microservices, your teams are free to choose the database that is the best fit for the task at hand. For example, if your application requires data on things that change over time, you could choose a time-series database, like Amazon Timestream, or if you need to maintain a cryptographically verifiable record of transactions, you could choose Amazon Quantum Ledger Database. In general, though, the best database is the database that does exactly what your microservice needs.



Recochoku migrated 10 million sets of user information, covering eight years of operations to Amazon Aurora.

As a music distribution service for mobile phones, Recochoku needed a scalable way to store data from their growing user base while maintaining security standards to keep their customer data safe. As a result of migrating to Amazon Aurora, procurement time was significantly shorter due to the ability to arrange infrastructure easily, and the operational load was lighter because Aurora is a fully managed service. Through Aurora, Recochoku has also lowered costs because of reduced licensing fees.

See the full story >

"Amazon Aurora is a fully managed system, enabling the development team to focus their human resources on service development."

–Shuhei Sakai, Systems Architect Group, Business Systems Promotion Department, Recochoku Co., Ltd.

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 Lambda boils down to how much your enterprise requires flexibility versus simplicity—which we define as seamless integration with other services and features. There are natural tradeoffs 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, we're seeing our customers choose containers and Lambda for compute. Containers have emerged as the most popular way to package code, which makes this a great way to modernize legacy applications. Containers offer excellent portability and flexibility over applications' settings. 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 that the cloud has to offer.

Financial Engines lowered costs by 90%, using AWS Lambda. The company provides financial help to more than nine million people across 743 companies (including 150 of the Fortune 500) and represents $1.18 trillion in retirement assets. As demand for its services grew, Financial Engines wanted to improve the scalability of its core engine component while reducing the need for capacity planning and improving resilience.

Financial Engines chose to migrate to AWS Lambda. By using AWS Lambda instead of Amazon EC2 instances for its workload, Financial Engines is realizing ongoing cost savings and experiencing near-zero downtime and near-zero performance degradation while serving 200–300 million requests.

See the full story >

"Using AWS Lambda will enable us to scale to serve millions more who need financial help."

–Paul Gibson, Principle System Architect

Samsung eliminated the need for an administrator, reduced monthly costs by approximately 44.5%, and improved developer efficiency by migrating to AWS Fargate. Samsung was running developer portals on Amazon ECS and looking for a way to improve overall service reliability, security, and performance.

By opting for a serverless compute engine for containers, AWS Fargate, Samsung was able to move faster while being more economical with resources.

See the full story >

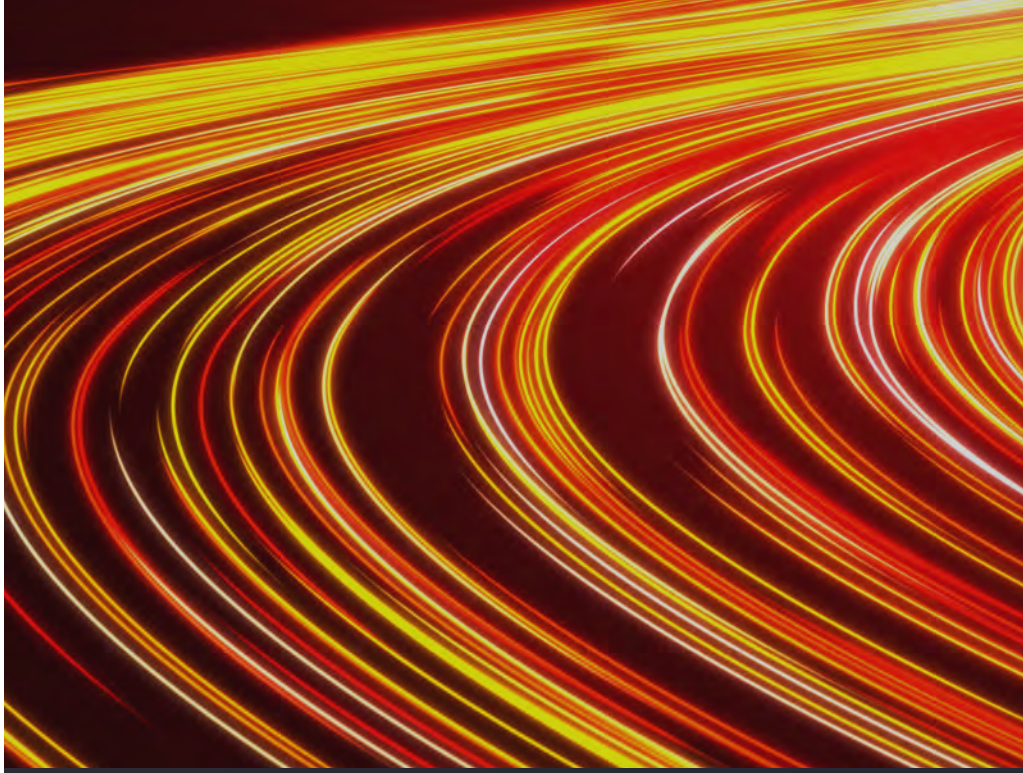
Release pipelines: standardized and automated

Microservices architectures enable teams to move faster, which means that you've got more stuff to release—great! However, you won't get new features to your customers faster if your release pipeline is bogged down. Traditional release pipelines are slowed mainly by manual processes. Manual steps—from code changes and build requests to testing and deploying—are the greatest drag on release velocity. The solution to this is two-pronged: standardization and automation.

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 flywheel. These processes are called continuous integration and continuous delivery (CI/CD). Automating the release pipeline through CI/CD helps teams release high-quality code faster and more often. Teams that practice CI/CD ship more code and do it faster. In fact, according to the "Puppet 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. Most notably, teams that practice CI/CD spend 44% 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 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 - Continuous integration 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 - Continuous delivery is a software development practice where code changes are automatically prepared for a release to production. Continuous delivery expands upon continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage.

lululemon

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—on 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

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 architectures, using serverless services wherever possible.

A serverless operational model allows you to 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.

Checkout

Checkout processes payments faster and meets compliance standards with a serverless infrastructure that takes advantage of AWS Lambda and AWS Fargate. As a provider of payment and other services to online retailers, the organization needed a way to easily scale up during busy periods while maintaining continuous uptime.

Not only is Checkout now able to scale to handle those heavy periods of demand but it also has a better customer service-level agreement and can meet compliance standards more rapidly. Rather than spending weeks collecting access and log details from subcontractors, the company can use the logging and audit capabilities integrated with its serverless infrastructure to easily provide regulators with the details required.

See the full story >

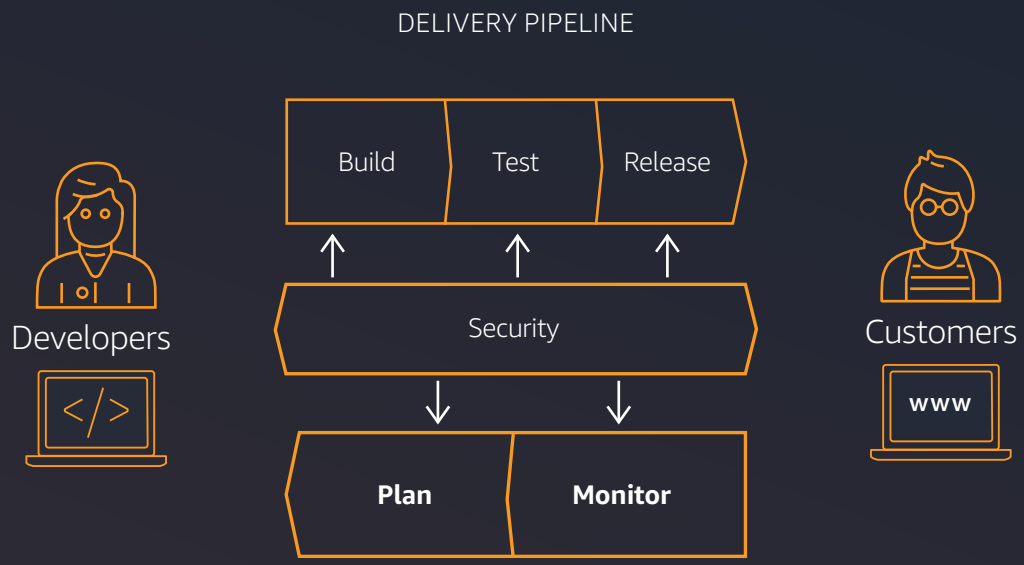
"The auto-scaling capabilities of our AWS serverless infrastructure means that we don't worry about scaling when traffic goes up. It just works."

–Mika Turunen, Chief Technology Officer,

Security: everyone's responsibility

We arrive at security last, but make no mistake—application security is priority number one. Security and compliance should be included at every stage of the app development lifecycle and in every component of the app itself. You want to ensure that your applications work as intended—and only as intended—so you can protect your business and your customers.

In the past, many companies treated security as if it were magic dust—something to sprinkle on after an application was ready for release. This doesn't work well in a continuous release cycle, so later, many took a new approach to security, building firewalls around the entire application. But that also introduced challenges, as every piece of the application had the same security settings applied. This is problematic if an application is built with independent microservices.



Secure configuration and automation are needed:

- While resources are being built, tested, and released
- After resources have been deployed (to ensure they remain configured as designed)

In modern applications, security features are built into every component of the application and automatically tested and deployed with each release. This means that security is no longer the sole responsibility of the security team; it is deeply integrated into every stage of the development lifecycle and engineering, operations, and compliance teams all have a role to play.

Security is integrated within tooling (like code repositories, build-management programs, and deployment tools) and is applied to both the release pipeline itself as well as to the software being released through the pipeline. With serverless services, security posture is even easier to maintain because the underlying infrastructure security—such as operating-system version updates, software patching, and monitoring—is built into each service.

A bit more detail:

- **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.

FICO™

FICO provides credit scoring and other services to 95% of the largest US financial institutions. Rapid innovation is key to remaining a leader in the industry. For many years, however, FICO lacked the agility to quickly develop and deploy its solutions, such as its flagship FICO Decision Management Suite, or DMS. To enable developers to focus more on creating features and functionality, FICO wanted to move DMS to the cloud. At the same time, they had to offer strong security and remain compliant with PCI, GDPR, and other regulations.

After evaluating their options, FICO moved DMS to AWS, enabling the organization to bring more innovation to the marketplace, faster. What used to take weeks now can be done in a day. They are also meeting stringent security and compliance requirements by running DMS on AWS Cloud.

See the full story >

"We have to be very careful about what code goes into production, so we use a managed software development lifecycle to ensure our financial data is in compliance. We can do that by using our own security tools along with AWS services to fully meet our regulatory requirements."

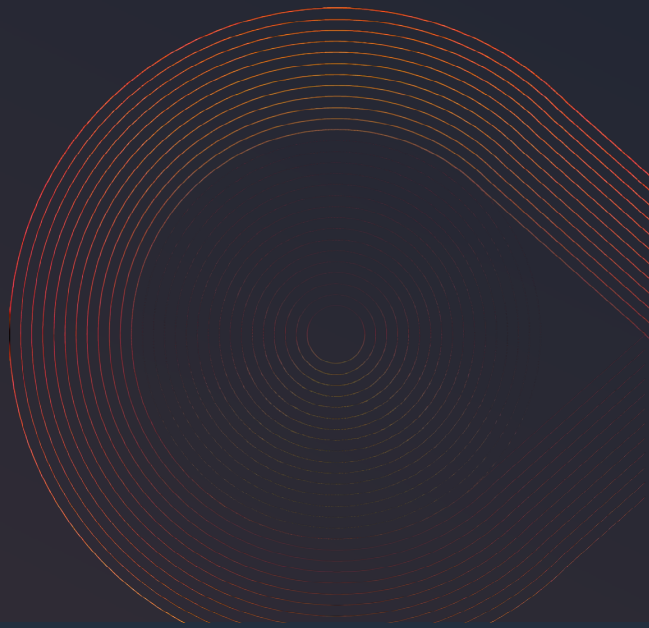
–Joshua Prismon, Vice President of Product Development, FICO

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. Modern application development is how organizations, including Amazon, innovate with speed and agility.

Modern applications on AWS help to:

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



Any starting point. Any application. Any innovation. AWS is how modern application development happens.

Modern application development on AWS

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.

Learn more about implementing the best practices of modern application development →

AWS services for modern application development

SERVERLESS MICROSERVICES

- Serverless Event-Driven Compute: AWS Lambda
- Serverless Containers: AWS Fargate
- API Management: Amazon API Gateway
- Message Queue Service: Amazon SQS
- Publish/Subscribe Messaging: Amazon SNS
- Orchestration: AWS Step Functions
- Serverless Event Bus: Amazon EventBridge
- Service Mesh: AWS App Mesh
- GraphQL Serverless API: AWS AppSync
- MYSQL And POSTGRESQL Database: Amazon Aurora Serverless
- No SQL Database: Amazon DynamoDB
- Object Storage: Amazon S3

DEVELOPMENT, DELIVERY, AND OBSERVABILITY

- Infrastructure Provisioning: AWS CloudFormation
- Cloud-Based IDE: AWS Cloud9
- User Auditing: AWS CloudTrail
- Continuous Delivery: AWS CodePipeline
- Monitoring: Amazon CloudWatch
- Software Development Framework: AWS Cloud Development Kit

SECURITY

- Access Control: AWS Identity and Access Management (IAM)
- Single Sign-On: AWS Single Sign-On (SSO)
- Account Management: AWS Organizations
- Threat Detection: Amazon GuardDuty
- Security Assessment: Amazon Inspector
- Virtual Private Cloud: Amazon Virtual Private Cloud (VPC)
- WAF Management: AWS Firewall Manager
- DDOS Protection: AWS Shield
- Encryption Keys: AWS Key Management Service (KMS)
- SSL/TLS Certificate Management: AWS Certificate Manager