Best practices for 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. Companies are increasingly building products that are the technology itself or heavily influenced by technology—every company is becoming a technology company. 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 application development increases the agility of your teams and the reliability, security and scalability of your applications. It automates or abstracts away operational overhead to enable teams to spend more time building business logic. It facilitates an environment in which experimentation thrives because small failures don't lead to system outages. And it also requires a fundamental shift in the way you approach the creation of value.

67%

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

"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

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. However you approach it, you'll want to consider the best practices that we have observed our customers take as they build modern apps.

These best practices for modern application development arose from our experience building applications for Amazon.com, as well as from serving millions of AWS customers around the world. We observed common practices that enabled our customers to increase agility and build better apps that support the success of their businesses. While you can approach these practices from any starting point and in any order, 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:

- Innovation means listening to your customers
- The best practices for modern application development
  1. Enable experimentation by creating a culture of ownership
  2. Componentize applications using microservices
  3. Update applications and infrastructure quickly by automating the release pipeline
  4. Model and provision application resources, using infrastructure as code
  5. Simplify infrastructure management with serverless technologies
  6. Improve application performance by increasing observability
  7. Secure the entire application lifecycle by automating security
- Start with modern application development today

Innovation means listening to your customers

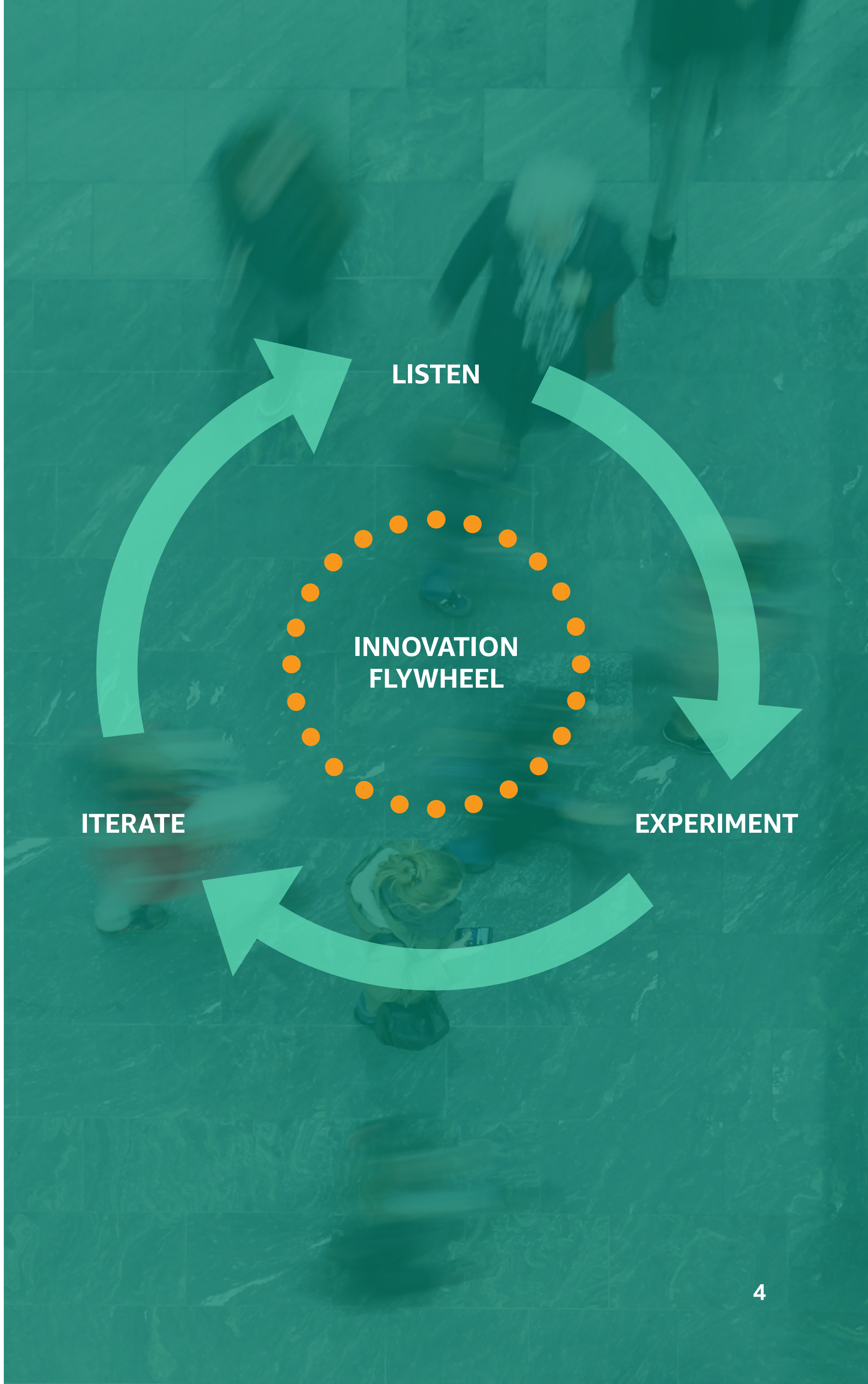
In a recent ***Vision Report***, "Digital Rewrites the Rules of Business," Forrester Research defines the customer-centric mindset of a digital innovator. The core mission of these modern disruptors is to:

"…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. From there, it means constantly evolving products and services to better deliver the outcomes that customers expect. And, finally, it means listening to what your customers truly care about, so 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 gets repeated 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 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, real-world companies like Dollar Shave Club are focused on meeting customer expectations. Recently acquired by Unilever, the online provider sells razor blades directly to customers through 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 application development helps digital innovation thrive because it reduces the time spent on operational overhead and makes space for experimentation. Through architectures that reduce the risk any failure has on the overall application, teams are also 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 application development 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 the seven best practices for modern application development that digital innovators rely on for success.

Best practices for modern application development

1. Enable experimentation by **creating a culture of ownership**
2. Componentize applications **using microservices**
3. Update applications and infrastructure quickly by **automating the release pipeline**
4. Model and provision application resources **using infrastructure as code**
5. Simplify infrastructure management with **serverless technologies**
6. Improve application performance by **increasing observability**
7. Secure the entire application lifecycle by **automating security**

Enable experimentation by 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 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 over 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.

Capital One recognized the importance of becoming a great digital technology company in order to continue to be a great bank. With that in mind, they adopted a DevOps model for their product teams. DevOps unites development and operations teams and uses automation, monitoring and continuous integration of new code to achieve faster development cycles.

By changing the way they deliver software using the AWS Cloud, Capital One was able to reduce the time it takes to build application infrastructure by 99 percent. The shift has also fostered a more collaborative culture and helped the company attract and retain top developer talent.

"Setting up that strong developer culture is important for attracting and retaining talented people. Moving to DevOps on the cloud is just another way that we can cultivate and support independent, autonomous teams that feel empowered to do their best work every day."

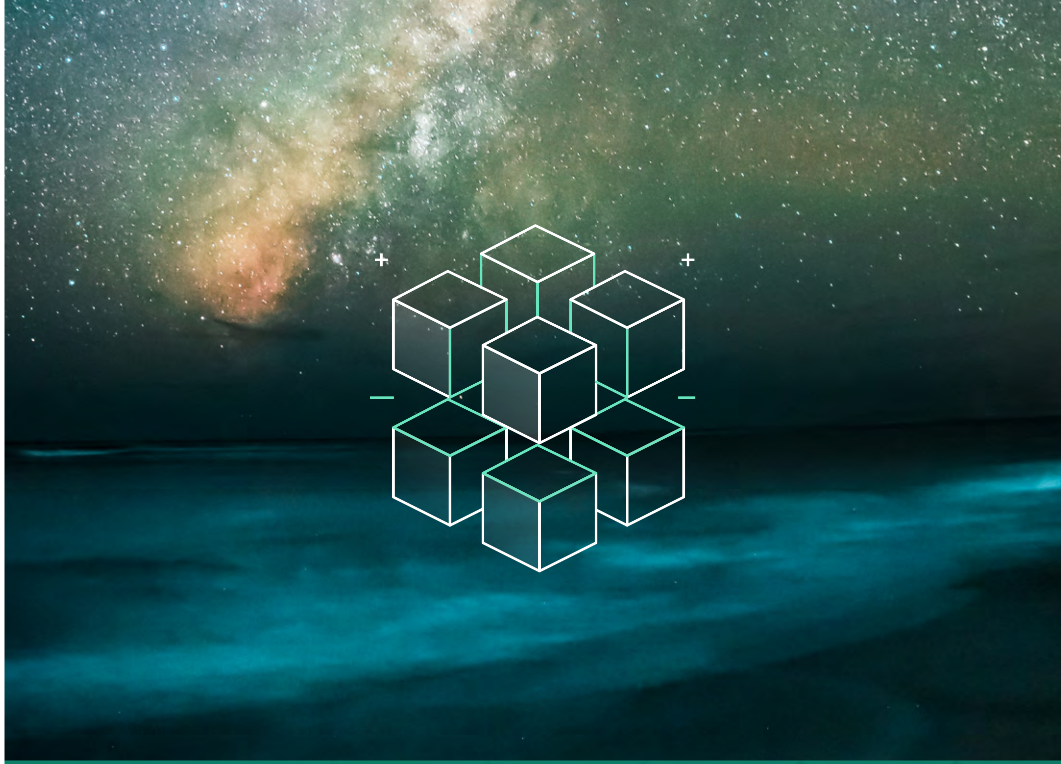
–George Brady, Executive Vice President and Chief Technology Officer, Capital One

Componentize applications using 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 in 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.



8 reasons modern applications are built with microservices

1. Pick the right tool for the job
2. Improve application resiliency
3. Provide granular scaling to control costs
4. Optimize team productivity
5. Simplify rewiring of services into new compositions
6. Enable your teams to experiment with lower risk
7. Support faster adoption of new technology
8. Integrate new features safely and quickly

For a fast-growth company trying to innovate and compete, this overhead and sluggishness is unacceptable. With a move to microservices, you can break down a monolith into minimal function services that can be deployed together to achieve a broader use case. Teams that build each microservice become owners who architect, implement, support in production, fix things and above all, care about quality.

Independence has its perks. By breaking down the monolith, each microservice has its own, fully decentralized datastore. No enterprise service bus, no single database, no top-down anything. Each component can also be changed and updated quickly, without a dramatic impact to the application as a whole. Now that you have reduced the impact of any one change, you can start experimenting with new ideas more frequently and with a lower risk.

EVERYTHING VS. ONE THING: TWO TYPES OF APPLICATIONS

Monolith apps

- Does 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

- Does 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.

"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

Update applications and infrastructure quickly by automating the release pipeline

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 customer faster if your release pipeline is bogged down. Traditional release pipelines get slowed down mainly by manual processes. When we studied our own processes, we found that manual steps throughout the release process—from code changes and build requests to testing and deploying—were the greatest drag on release velocity. Each step that was not automated introduced opportunity for delays and manual errors.

To get the process moving faster, you need a way to automate each of those steps. 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 5 times lower, a commit-to-deploy rate that is 440 times faster and a rate of deployments that is 46 times more frequent. 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 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.

With AWS, Lululemon Athletica can stand up development environments in minutes instead of days, automate its environment 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.

"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."

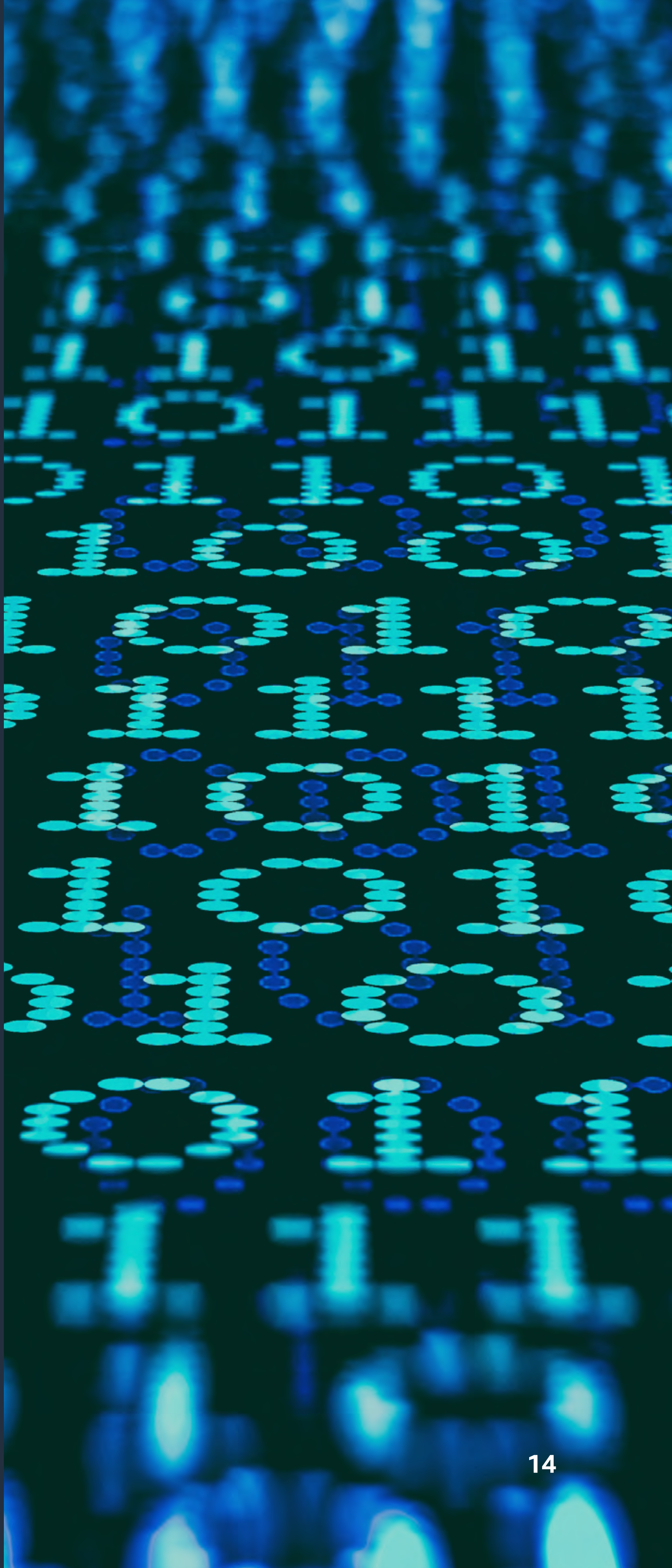
–Sam Keen, Director of Product Architecture, Lululemon

Model and provision application resources using infrastructure as code

When your application architecture is modular and you're releasing things really quickly, there are a lot of moving parts—and they're moving really fast. When you make changes to your environment manually, it's easy to introduce errors that break your application. Additionally, as your application scales, it can become challenging to scale your infrastructure seamlessly. To keep all of your resources working as expected you need a consistent environment, which you can easily scale or modify based on application needs. We address this challenge by modeling all application resources and infrastructure as code.

Infrastructure as code enables you to programmatically provision and change your resources using code. It acts as your single source of truth, so you can standardize infrastructure components, maintain configuration compliance and troubleshoot issues faster. Traditional infrastructure was hardware, and traditional source code was websites, apps, business logic, back-end services, transactions and the like. Today, however, anything can be source code—servers, firewalls, routers, load balancers, identity permissions, monitoring alerts, all of it. In the cloud, everything can be accessed by an API and provisioned on demand and elastically—you can use code to automate everything.

Using infrastructure as code, you can provision your resources in a safe, repeatable manner, allowing you to build and rebuild your infrastructure and applications without having to perform manual actions or write custom scripts. Instead, your teams can make changes in source control and let the pipeline test and deploy the updates. This not only reduces error rates but also dramatically speeds up the process of deploying infrastructure.



The Washington Post

The Washington Post reshaped its business model by creating Arc Publishing, a SaaS platform that enables any media company to take advantage of the scalable, flexible publishing platform The Post develops for its own purposes. Designed for constant innovation, Arc supports more than 100 microservices composed of more than 3,000 containers running on more than 150 Amazon EC2 instances.

The Post took an infrastructure-as-code (IaC) approach to building infrastructure templates, which enables users to rapidly build their own apps on the platform using the code from these templates. With a commitment to maintaining both quality and velocity, The Post empowers its distributed teams to innovate autonomously with minimal oversight. That freedom, and the paper's ability to keep its teams small and agile, has helped make Arc Publishing a success beyond the walls of its own newsroom. Utilizing a custom development platform, also built on AWS, Arc completes more than 50 deployments daily.

"The capability of being able to deploy rapidly is so critical. In those moments when you need to iterate quickly on a system that capability is enormously beneficial. That's what we get using AWS."

–Patrick Cullen, Principal Architect, The Washington Post

Simplify infrastructure management with serverless technologies

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 technologies wherever possible.

Serverless technologies allow you to build and run applications and services without provisioning and managing servers. Serverless technologies give you an operational model that 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.

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.

Serverless technologies are ideal for high-growth companies that want to innovate quickly. These technologies support teams of owners who build modern applications and eliminate the need for these teams to think about servers—with serverless, it's all automated. 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.

Improve application performance by increasing observability

Observing a monolithic app built on-premises was relatively straightforward—you used monitoring software on your main server. The world of monitoring is different and more challenging with modern applications. Microservices add complexity when it comes to monitoring in production because there are many things to monitor, and those things might have dependencies on each other. Microservices probably also live in many places because your application is distributed. Plus, your microservices may be running on ephemeral "servers," some of which only run for a few seconds. So how do you get an accurate, real-time view of your application?

You need a systemwide view that can fully trace and correlate all the metrics to provide a single pane of truth for what is going on in the system. You need all the metrics and data to get app-level visibility and then to correlate that to customer behavior to optimize the customer experience. This means collecting as much data as possible and monitoring the small things, from logs to application metrics. With more visibility into the health of your services and applications, you can solve problems faster.

Full observability helps you get to answers quickly and it requires metrics, logs and traces. To improve observability, you can start by having your services expose basic metrics themselves: response rates, error rates, etc., so that you can collect and aggregate as much as possible. You can monitor the small things and use aggregation to see the bigger picture—monitor the microservices to monitor the application. It's also critical to know what healthy looks like, and standardization is important: You need to view the system in a holistic way. When you have a unified view of your resources—and you can get that view in real time—you can rapidly respond to application and customer issues to improve overall performance.

Observability =

Comprehensive visibility across an entire system

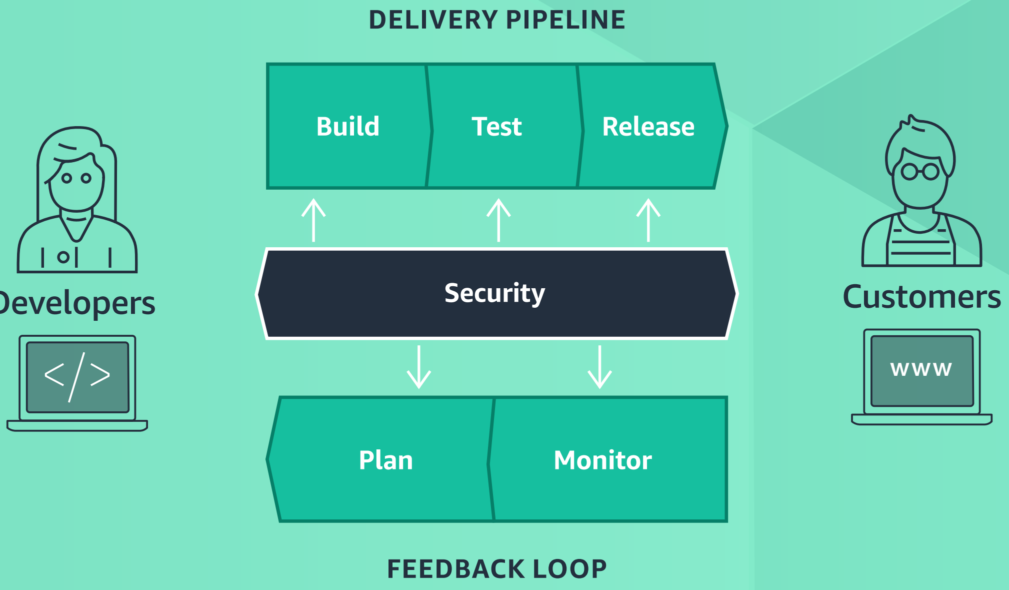
Observability requires:

Metrics, Logs, Traces

Secure the entire application lifecycle by automating security

This best practice is listed 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.

Microservices architectural patterns can improve the security of your applications by giving you fine-grained control over the security policies of each individual microservice. However, a microservices architecture also introduces challenges because of the increased surface area you need to protect. The way we can address that challenge is by automating security. Securing apps in the cloud means moving security capabilities deeper into the software engineering space and addressing security throughout the entire application lifecycle, not just at checkpoints. Automating the process of testing that security features are operating as intended helps keep applications secure without slowing down the development cycle.



Secure configuration and automation are needed...

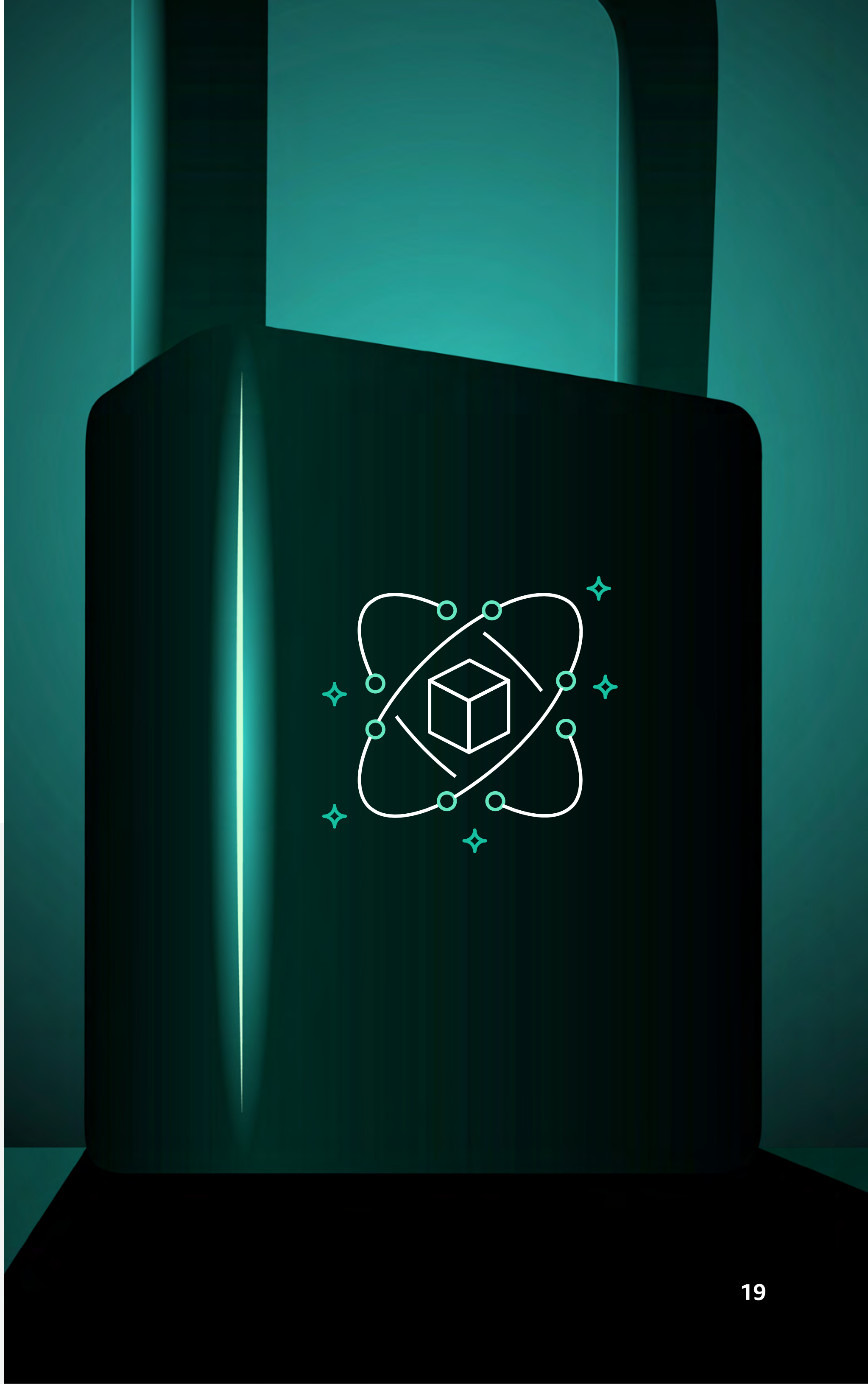
- **While resources are being built, tested and released**
- **After resources have been deployed (to ensure they remain configured as designed)**

We recommend building security into the development workflow (referred to as DevSecOps) during the predeployment phase. Following deployment, your team can ensure that the configuration you approved is still intact; and because it's cloud-based, you can capture automated events/notifications of changes to your infrastructure. You can also capture and act on all of those events/notifications to take actions, such as detection, alerts, remediation, countermeasures and/or forensics. With security policies automated from start to finish, you will gain greater insight into any changes in your environment.

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 percent of the largest U.S. 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 still 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. To set guardrails around what code goes into production, FICO uses a managed software development lifecycle to ensure their customers' financial data is in compliance, using their own security tools along with AWS services to fully meet security and regulatory requirements.

Get started with modern application development today

Modern application development creates competitive differentiation by enabling rapid innovation. By changing your architectural pattern, operational model and software delivery process using the best practices of modern application development, 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. The practices of modern application development are how organizations, including Amazon, innovate with speed and agility.

Modern application development with AWS helps to...

> Maintain performance

> Improve reliability

> Protect the business and its users

> Increase developer productivity

> Shorten time to market

> Increase revenue

> Lower operating 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
- Message Queue Service — Amazon SQS
- Publish/Subscribe Messaging — Amazon SNS
- Orchestration — AWS Step
- MYSQL And POSTGRESQL Database — Amazon Aurora Serverless
- Serverless Containers — AWS Fargate
- API Management — Amazon API Gateway
- Object Storage — Amazon S3
- No SQL Database — Amazon DynamoDB

DEVELOPMENT, DELIVERY AND OBSERVABILITY

- Infrastructure Provisioning — AWS CloudFormation
- Continuous Delivery — AWS CodePipeline
- Cloud-Based IDE — AWS Cloud9
- User Auditing — AWS CloudTrail
- Monitoring — Amazon CloudWatch
- Debugging — AWS X-Ray

SECURITY

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