aws FOR RETAIL

Architectural guidance for building composable MACH solutions on AWS

How AWS enables retailers to deliver new features and experiences to consumers faster

A MACH architecture, which stands for Microservices based, API-first, Cloud-native SaaS and Headless (MACH), gives companies a deeper level of agility—allowing them to deliver new features to their customers at speed. This architecture has spread from ecommerce to store systems and is now fundamentally changing the way retailers assemble commerce solutions.

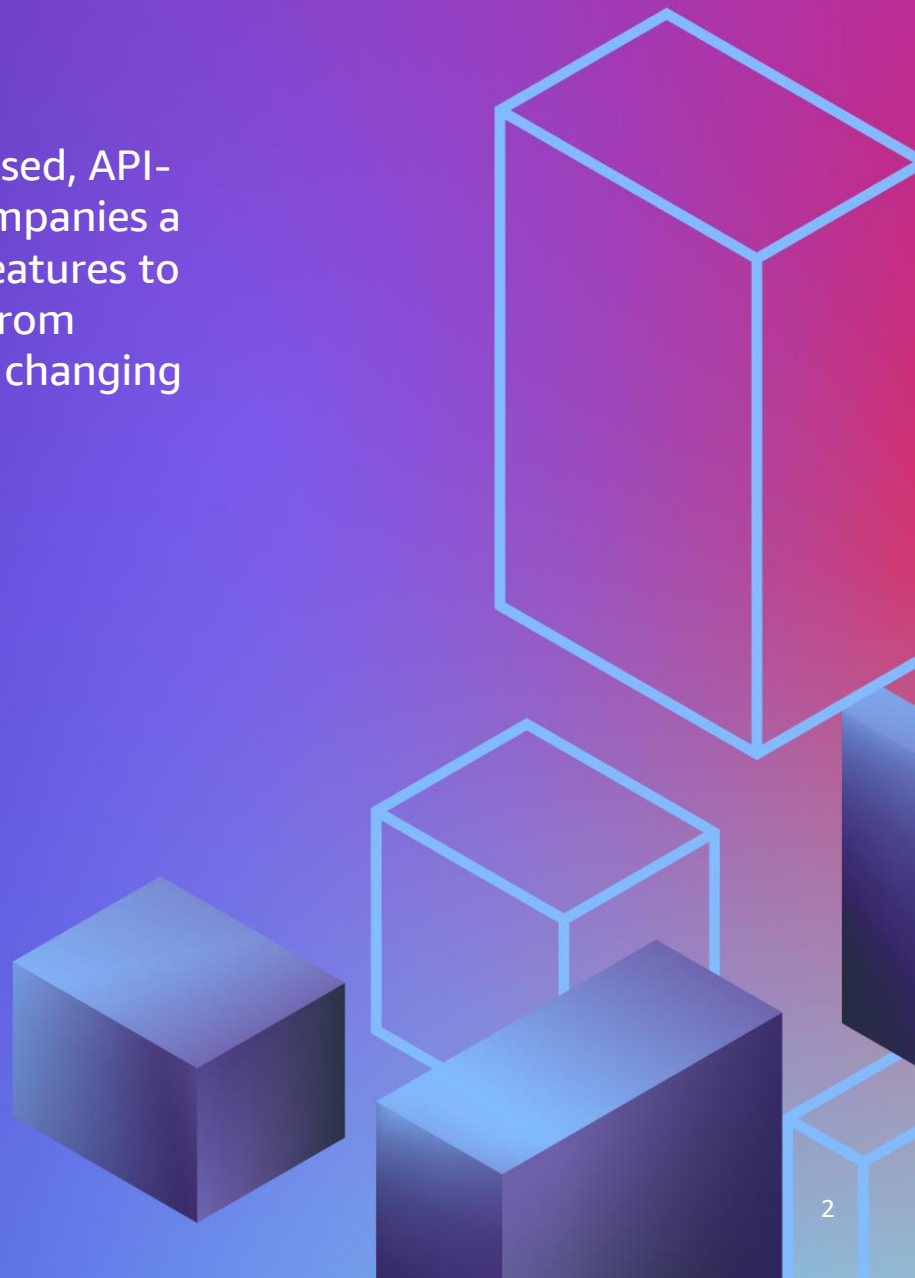


Table of contents

Introduction ……..……..4

MACH Architecture ……..……..5

Microservices……..7

API-first ……18

Cloud-native SaaS ……..24

Headless ……...29

AWS Partners in MACH…….……...34

Introduction

MACH Alliance

The MACH Alliance is a not-for-profit industry body that advocates for open and best-of-breed enterprise technology ecosystems.

The Alliance is a vendor-neutral institution that provides resources, education and guidance through industry experts to support companies on their journey.

The MACH Alliance and the associated approach to software architecture has been recognized by Gartner and Forrester. The terms “composable commerce” and “headless commerce” are sometimes used as synonyms for MACH. MACH technologies support a composable enterprise in which every component is pluggable, scalable, replaceable, and can be continuously improved through agile development to meet evolving business requirements.

Learn more »

MACH certification

To advance the mission and support growth, the MACH Alliance established certification standards that help identify vendors and integrators that embrace MACH philosophies and offer MACH-certified services. To become an Alliance member and carry the MACH Certified seal, a company must be in full compliance.

AWS joined the MACH Alliance in January 2022 as an "Enabler," a category for companies that provide the underlying infrastructure to support MACH products and integrations.

“The MACH Alliance are the bouncers controlling the velvet rope at the entrance to the Coolest Tech in Town Club.”

Joe Cicman, Senior Analyst, Digital Transformation, **Forrester**

Microservices based, API-first, Cloud-native, and Headless (MACH)

Microservices architecture is a software development approach that breaks down a complex applications into smaller, independently deployable services. Building microservices allows you to iterate faster, reduce time to market, and improve resiliency.

APIs allow different software systems to communicate with each other and exchange information. They define a contract of service between two applications. API-first is a design approach that puts APIs at the center of the solution, making it easier to integrate with other systems and services.

Cloud-native SaaS is a software development approach that takes advantage of cloud computing resources and services. Using a cloud-native approach allows you to build highly scalable, flexible, and resilient applications that can be updated quickly to meet customer's demands.

Headless is an architecture that separates the front-end from the back-end of a solution, allowing companies to use different front-end technologies and platforms. Headless architectures allow you to build solutions that are flexible, modular, and scalable, making it easier to deliver personalized and engaging experiences across multiple channels and devices.

Amazon Web Services (AWS) offers a wide range of services that can help developers build MACH solutions. In this ebook we will dive deeper into each principle and discuss how to use AWS to build MACH architectures.

MACH-advanced companies move faster

Four in five decision makers state that volatility in the economy has impacted their organizations' attitude toward MACH. This has been one of the key drivers behind 85 percent of organizations increasing the percentage of their MACH infrastructure in the past 12 months.

Those companies cite increased ability to respond to changes in the market faster, to build and implement new functionality quicker and reduced costs. They're also more likely to say their infrastructure is keeping up with customer demands and that they're ahead of the competition than those with lower MACH adoption rates.

MACH Global Research 2023 »

Great MACH runs on AWS

While MACH architecture offers many benefits, such as increased agility and scalability, it requires a robust cloud infrastructure to function efficiently. That's where AWS comes in.

AWS is the world's most comprehensive and broadly adopted cloud platform that provides retailers with a range of services, including compute, storage, database, and networking. It offers a highly secure and reliable infrastructure that can handle the demands of MACH architecture, allowing businesses to focus on innovating for their customers. Here are some of the reasons why great MACH architecture runs on AWS:

Scalability AWS offers a scalable infrastructure that can grow or shrink on demand. With AWS, retailers can scale their MACH architecture to meet the demands of their customers.

High availability AWS provides a highly available infrastructure ensuring that retailers can deliver seamless experiences to their customers, even during peak traffic.

Flexibility AWS provides a flexible infrastructure that can adapt to the changing needs of retailers. It offers a range of over 200 fully featured services, which can be used to build and deploy MACH architecture, allowing retailers to choose the best services that fit their requirements.

Security AWS provides a highly secure infrastructure safeguarding retailer's data and applications. It offers a range of security services that can be used to secure the MACH architecture, so retailers can focus on delivering great experiences to their customers.

Cost-effective AWS provides a cost-effective infrastructure that can help retailers reduce their infrastructure costs. With AWS, retailers pay only for the services they use, ensuring that they can optimize their infrastructure costs.

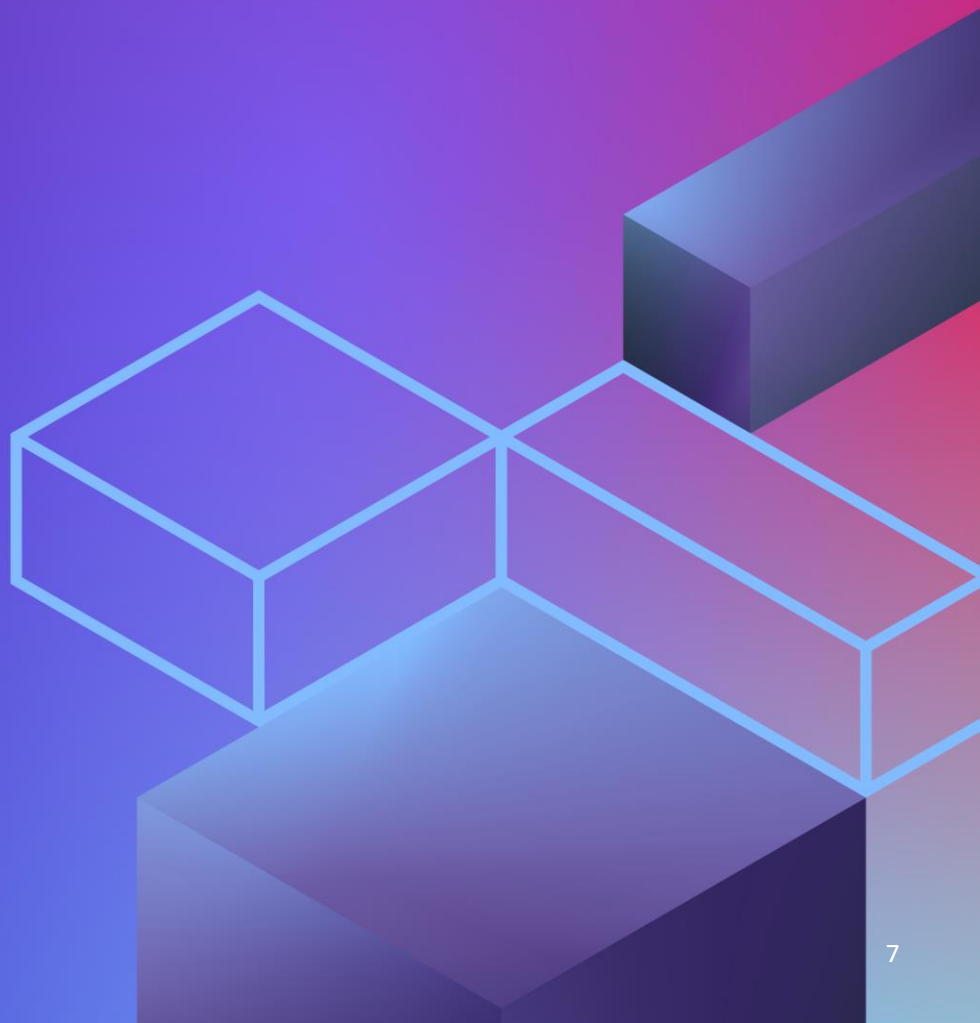
The term **"composable commerce"** was popularized by a Gartner report in 2020.

In the report, Gartner predicted that **"by 2023, organizations that have adopted a composable approach will outpace the commerce approach competition by 80% in the speed of feature implementation,".**

Today, **70% of MACH members run on AWS**—with a large and growing community of MACH Alliance members choosing to build great MACH on AWS.

aws FOR RETAIL

Microservices



Microservices-based

Microservices are an architectural and organizational approach to software development using an agile approach that helps teams work independently. They speed up deployment cycles, foster innovation and ownership, improve maintainability and scalability of software applications, and scale organizations delivering software and services.

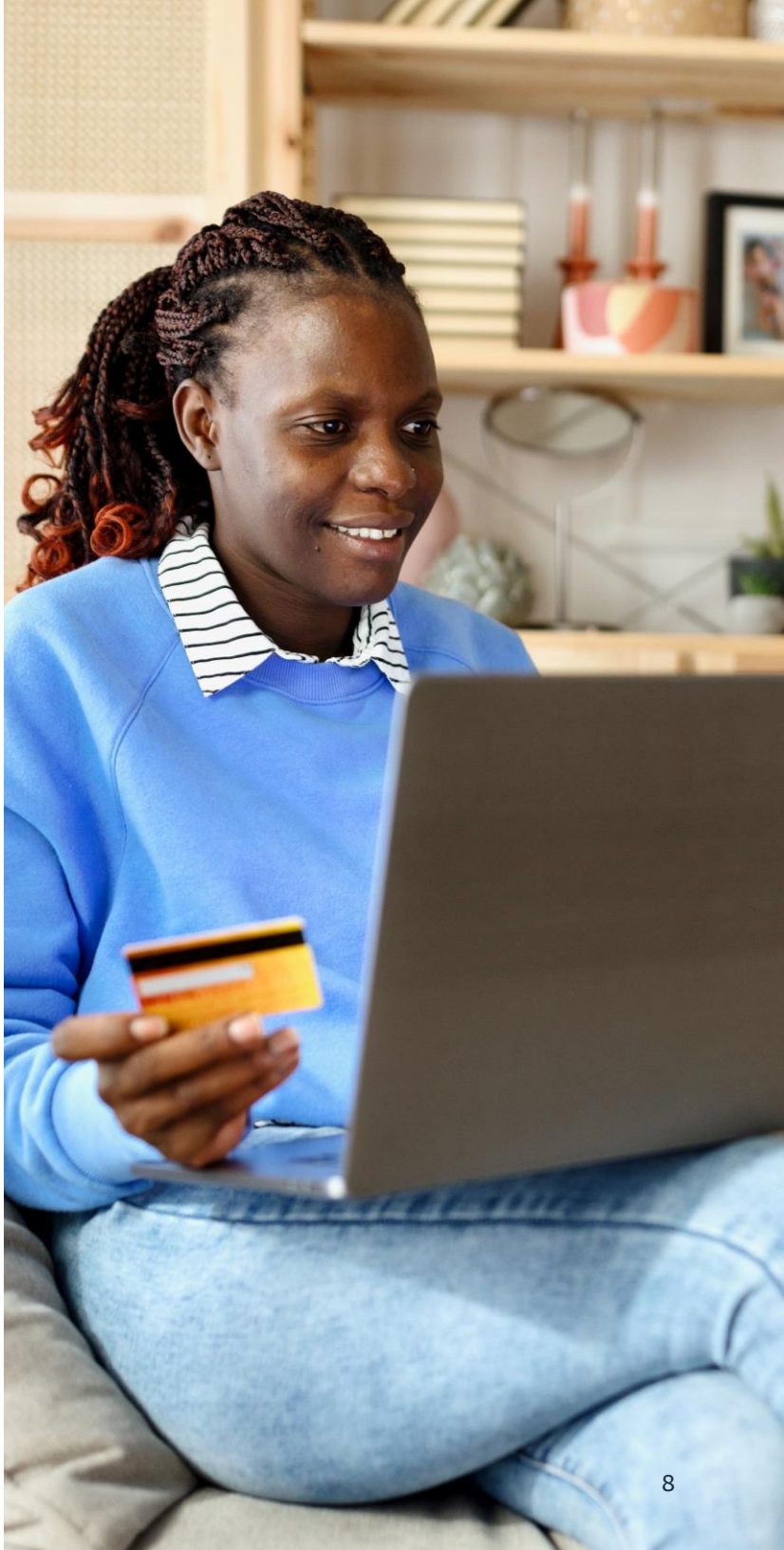
With a microservices approach, software is composed of small services that communicate over well-defined application programming interfaces (APIs) which can be deployed independently. These services are owned by small autonomous teams. This agile approach is key to successfully scaling your organization.

Microservices Implementation

AWS has integrated building blocks that support the development of microservices. Two popular approaches to implement microservices are using AWS Lambda and using containers.

AWS Lambda is a serverless compute service that runs your code in response to events and automatically manages the underlying compute resources for you. Lambda runs your code on high availability compute infrastructure and performs all the administration of your compute resources. This includes server and operating system maintenance, capacity provisioning and automatic scaling, code and security patch deployment, and code monitoring and logging. All you need to do is supply the code.

Lambda is a good choice for services that require quick response times and low traffic. Since Lambda functions are only charged for the time they execute, they are a cost-effective option for services with low to medium traffic. Lambda also provides automatic scaling and high availability, making it easy to handle sudden spikes in traffic.



Containers

Container technologies like Docker are a popular choice to build your microservices due to benefits like portability, productivity, and efficiency.

On AWS, you can select from the broadest choice of services to run your containers.

Choose AWS Fargate for scalable, serverless compute for containers where AWS will manage your infrastructure provisioning.

For full control over your compute environment, choose to run your containers on Amazon Elastic Compute Cloud (Amazon EC2).

For container orchestrators, you can choose either Amazon Elastic Container Service (Amazon ECS) or Amazon Elastic Kubernetes Service (Amazon EKS).

Both Amazon ECS and Amazon EKS automatically manage the availability and scalability of the control plane responsible for scheduling containers, managing application availability, storing cluster data, and other key tasks. This allows you to take advantage of all the performance, scale, reliability, and availability of an AWS infrastructure.

With capacity providers (Amazon ECS) and Kubernetes Cluster Autoscaler (Amazon EKS) you can automatically scale the infrastructure for tasks and pods to meet changing demands.

Containers are a better choice for services that require more control over the environment or need to run for longer periods.

Containers are also more suitable for applications that need to run continuously or require complex dependencies.

Learn more about Containers at AWS for help choosing the right container options for your services.

Containers at AWS

Run your containers in the most secure, reliable and scalable environment.

Containers at AWS »

Data Stores

A microservices design principle is Decentralized Data Management, meaning each microservice should manage its own data, without relying on other microservices, to ensure scalability and reliability.

AWS provides several purpose-built data store options to use with your microservices, including in-memory databases, relational databases, and NoSQL databases.

In-memory databases such as Amazon MemoryDB for Redis and Amazon ElastiCache are designed to enable minimal response times by eliminating the need to access disks. In-memory databases are ideal for applications requiring microsecond response times or have large spikes in traffic such as gaming leaderboards, session stores, and real-time analytics.

However, since all data is stored and managed exclusively in the main memory, in-memory databases risk losing data upon a process or server failure.

Relational databases are still very popular to store structured data and business objects. AWS offers seven database engines (Amazon Aurora with MySQL compatibility, Amazon Aurora with PostgreSQL compatibility, MySQL, MariaDB, PostgreSQL, Oracle, and SQL Server) as managed services through Amazon Relational Database Service (Amazon RDS).

This means the code, applications, and tools you already use today with your existing databases should work seamlessly with Amazon RDS. With Amazon RDS you benefit from the flexibility of being able to scale the compute resources or storage capacity associated with your relational database instance. In addition, Amazon RDS makes it easy to use replication to enhance database availability, improve data durability, or scale beyond the capacity constraints of a single database instance for read-heavy database workloads.

NoSQL databases have been designed to favor scalability, performance, and availability over the consistency of relational databases. One important element of NoSQL databases is that they typically don't enforce a strict schema. NoSQL databases use a variety of data models for accessing and managing data.

These types of databases are optimized specifically for applications that require large data volume, low latency, and flexible data models, which are achieved by relaxing some of the data consistency restrictions of other databases. AWS offers a wide range of NoSQL databases, including key-value (Amazon DynamoDB), document (Amazon DocumentDB (with MongoDB compatibility)), graph (Amazon Neptune), and search (Amazon OpenSearch Service).

Networking

One of the primary challenges with microservices architectures is allowing services to discover and interact with each other. The distributed characteristics of microservices architectures not only make it harder for services to communicate, but also presents other challenges, such as checking the health of those systems and announcing when new applications become available.

AWS can help address these challenges, both from a service discovery and service mesh point-of-view.

Service discovery is the process of locating and connecting to services in a distributed system.

AWS Cloud Map is a fully managed resource discovery service that allows you to discover and connect to services by name.

AWS Cloud Map extends the capabilities of Amazon Route 53's Auto Naming APIs by providing a service registry for your cloud resources. This includes Internet Protocols (IPs), Uniform Resource Locators (URLs), and Amazon Resource Names (ARNs). It offers an API-based service discovery mechanism with a faster change propagation and the ability to use attributes to narrow down the set of discovered resources.

A service mesh is a dedicated infrastructure layer for managing service-to-service communication within a microservices architecture. A service mesh can help to simplify service discovery, load balancing, and traffic management, and can provide visibility and control over network traffic.

AWS App Mesh is a fully managed service mesh that provides traffic management, observability, and security for microservices. App Mesh can be integrated with AWS Fargate, Amazon ECS, Amazon EKS, Amazon EC2, and Kubernetes on EC2, making it easy to manage and monitor microservices running on these platforms.

Networking

Amazon VPC Lattice is a networking service that consistently connects, monitors, and secures communications between your services, helping to improve productivity so that your developers can focus on building features that matter to your business. You can define policies for network access, traffic management, and monitoring to connect compute services in a consistent way across instances, containers, and serverless applications.

As your microservices infrastructure scales Elastic Load Balancing (ELB) allows you to distribute network traffic to improve application scalability and to deliver applications with high availability.

For your microservices applications, AWS offers the Application Load Balancer (ALB) and the Network Load Balancer (NLB):

Application Load Balancer operates at the request level (Layer 7), routing traffic to EC2 instances, containers, IP addresses, and Lambda functions. ALB provides advanced request routing based on the content of the request. Application Load Balancer simplifies and improves the security of your application, by facilitating that the latest SSL/TLS ciphers and protocols are used at all times.

Network Load Balancer operates at the connection level (Layer 4), routing connections to EC2 instances, containers, and IP addresses within Amazon VPC. NLB is ideal for load balancing of both TCP and UDP traffic, capable of handling millions of requests per second while maintaining ultra-low latencies. Network Load Balancer is optimized to handle sudden and volatile traffic patterns.

For applications serving global users, AWS helps you improve the availability and performance of those applications with AWS Global Accelerator. Global Accelerator provides two global static public IPs that act as a fixed entry point to your application endpoints, such as ALB's, NLB's, Amazon EC2 instances, and elastic IPs. Global Accelerator always routes user traffic to the optimal endpoint based on performance, reacting instantly to changes in application health, your user's location, and policies you've configured.

Monitoring and Observability

A microservices architecture consists of many different services that have to be monitored. Monitoring and tracing in a distributed environment can be complex, but AWS has the tools to help you address these.

You can use Amazon CloudWatch to gain system-wide visibility into resource utilization, application performance, and operational health. CloudWatch provides a reliable, scalable, and flexible monitoring solution that you can start using within minutes.

In addition to Amazon CloudWatch, you can also use CloudWatch Container Insights and CloudWatch Lambda Insights to collect, aggregate, and summarize metrics and logs from your containerized and serverless applications and microservices. These features automatically collect metrics for many resources, such as CPU, memory, disk, and network. They are aggregated as CloudWatch metrics at the cluster, node, pod, task, and service level for containerized applications and at the function level for serverless applications.

Another popular option, especially for Amazon EKS, is to use Prometheus. Prometheus is an open-source monitoring and alerting toolkit that is often used in combination with Grafana to visualize the collected metrics. Amazon Managed Service for Prometheus is a Prometheus-compatible monitoring service that enables you to monitor containerized applications at scale.

With Amazon Managed Service for Prometheus, you can use the open-source Prometheus query language (PromQL) to monitor the performance of containerized workloads without having to manage the underlying infrastructure required to manage the ingestion, storage, and querying of operational metrics. You can collect Prometheus metrics from Amazon EKS and Amazon ECS environments using AWS Distro for OpenTelemetry or Prometheus servers as collection agents.

Monitoring and Observability

Amazon Managed Service for Prometheus is often used in combination with Amazon Managed Grafana. Amazon Managed Grafana makes it easy to query, visualize, alert on and understand your metrics no matter where they are stored. With Amazon Managed Grafana, you can analyze your metrics, logs, and traces without having to provision servers, configure and update software, or do the heavy lifting involved in securing and scaling Grafana in production.

Microservices enable you to ship more often and encourage engineering teams to run experiments on new features, consistent logging becomes critical for troubleshooting and identifying issues. On AWS, you can use Amazon CloudWatch Logs to monitor, store, and access your log files. For applications running on Amazon EC2 instances, a daemon is available to send log files to CloudWatch Logs. Lambda functions natively send their log outputs to CloudWatch Logs and Amazon ECS includes support for the *awslogs* log driver that enables the centralization of container logs to CloudWatch Logs.

For Amazon EKS, either Fluent Bit or Fluentd can forward logs to CloudWatch Logs. Because of its smaller footprint and performance advantages, Fluent Bit is recommended instead of Fluentd.

It's often the case that multiple microservices will work together to handle a single request. Even if every microservice is logging properly and logs are consolidated in a central system, when an error occurs in one of the services in the call chain it can be difficult to find all relevant log messages. AWS X-Ray helps developers analyze and debug production and distributed applications.

The idea of AWS X-Ray is the use of correlation IDs, which are unique identifiers attached to all requests and messages related to a specific event chain. The trace ID is added to HTTP requests in specific tracing headers and included in the response. You can use X-Ray with applications running on EC2, ECS, Lambda, and AWS Elastic Beanstalk. In addition, the X-Ray SDK automatically captures metadata for API calls made to AWS services using the AWS SDK and provides add-ons for MySQL and PostgreSQL drivers.

Release Process

AWS provides a set of flexible services designed to enable companies to more rapidly and reliably build and deliver applications. These services simplify provisioning and managing infrastructure, deploying application code, and automating software release processes.

To provision and manage your infrastructure we recommend the AWS Cloud Development Kit (CDK) and the AWS Serverless Application Model (SAM):

AWS Cloud Development Kit

The AWS Cloud Development Kit (AWS CDK) lets you build reliable, scalable, cost-effective applications in the cloud with the considerable expressive power of a programming language. The AWS CDK supports TypeScript, JavaScript, Python, Java, C#/.Net, and Go.

Developers can use one of these supported programming languages to define reusable cloud components known as Constructs. You can then compose these together into Stacks and Apps. You can learn more about the AWS CDK concepts in the AWS Documentation.

AWS Serverless Application Model

The AWS Serverless Application Model (AWS SAM) is a convenient way to define serverless applications. AWS SAM is natively supported by AWS CloudFormation and defines a simplified syntax for expressing serverless resources.

To deploy your application, specify the resources you need as part of your application, along with their associated permissions policies in an AWS CloudFormation template, package your deployment artifacts, and deploy the template.

AWS developer tools

The AWS Developer Tools help you securely store and version your application's source code, and automatically build, test, and deploy your application.

AWS CodeCommit is a fully managed source control service that enables secure and highly scalable private Git repositories. You can use CodeCommit to securely store anything from source code to binaries, and it works seamlessly with your existing Git tools.

AWS CodeBuild is a fully managed build service that compiles source code, runs tests, and produces software packages that are ready to deploy. CodeBuild scales continuously and processes multiple builds concurrently, so your builds are not left waiting in a queue.

AWS CodeDeploy is a fully managed deployment service that automates software deployments to various compute services, such as Amazon EC2, Amazon ECS, and AWS Lambda. CodeDeploy allows you to rapidly release new features, helps avoid downtime during application deployment, and handles the complexity of updating your applications. You can use CodeDeploy to automate software deployments, eliminating the need for error-prone manual operations.

AWS CodePipeline is a fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates. CodePipeline automates the build, test, and deploy phases of your release process every time there is a code change, based on the release model you define. This enables you to rapidly and reliably deliver features and updates. You can also integrate CodePipeline with third-party services such as GitHub or with your own custom plugin. Microservices systems can grow to become complex systems. Understanding how the system behaves under stress and identifying potential issues before they impact business operations become the key value for continuous business operation.

AWS Fault Injection Simulator (FIS) is a managed service that enables you to perform fault injection experiments on your AWS workloads. Fault injection is based on the principles of chaos engineering. These experiments stress an application by creating disruptive events so you can observe how your application responds. You can then use this information to improve the performance and resiliency of your applications.

Learn more about how to repeatedly test the impact of fault actions as part of your software delivery process.

Amazon Builders' Library

Learn more about how Amazon builds and operates software at scale.

Visit Amazon Builders' Library »

commercetools
Next generation commerce

AWS Partner MACH spotlight

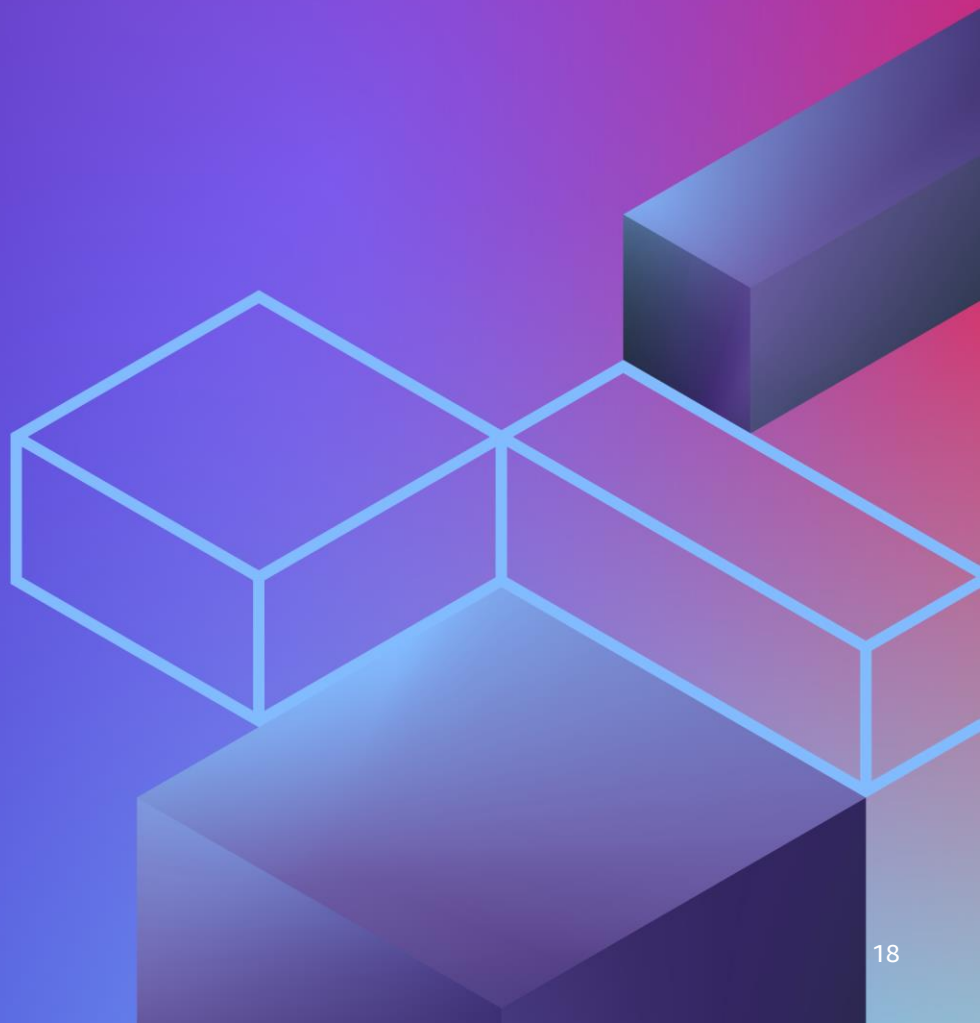
The story of The Very Group's commerce modernization

Learn about The Very Group's journey to modern commerce from the company's previous monolithic architecture to the commercetools and AWS solution, all of the strengths of MACH (Microservices-based, API-first, Cloud-native, Headless) and composable commerce architecture with the benefits this foundation has brought to the business.

Read the story »

aws FOR RETAIL

API-first



API-first

APIs are mechanisms that enable two software components to communicate with each other using a set of definitions and protocols. APIs define a contract of service between two applications. This contract characterizes how the two communicate with each other using requests and responses. Their API documentation contains information on how developers are to structure those requests and responses.

AWS provides several tools and services to build and deploy APIs quickly and efficiently.

REST vs. GraphQL APIs

REST (Representational State Transfer) and GraphQL are two popular API styles used for building web applications. The choice of REST or GraphQL depends on the requirements of your application. REST is a good choice for simple applications that require fast performance, while GraphQL is suitable for complex applications that require flexibility and scalability.

REST is a well-established API style that uses HTTP methods like GET, POST, PUT, and DELETE to access resources. It follows a stateless client-server architecture. The client sends a request to the server, which sends back a response. RESTful APIs are easy to understand, cacheable, and scalable.

For building REST-based APIs, we recommend Amazon API Gateway, a fully managed service enables developers to create, publish, maintain, monitor, and secure APIs at any scale. API Gateway handles all the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls, including traffic management, authorization and access control, throttling, monitoring, and API version management.

For simplifying REST API integration, API Gateway can generate client SDKs for a number of platforms to quickly test new APIs from your applications and distribute to third-party developers.

REST vs. GraphQL APIs

GraphQL is a newer API style that uses a single endpoint to access data using a query language. GraphQL allows clients to specify the data they need, avoiding over-fetching or under-fetching of data. It can improve performance and reduce network traffic.

For building GraphQL-based APIs, we recommend AWS AppSync, which is a managed service that uses GraphQL to make it easy for applications to get exactly the data they need.

With AWS AppSync, you can build scalable applications, including those requiring near real-time updates, on a range of data sources such as NoSQL data stores, relational databases, HTTP APIs, and your custom data sources with AWS Lambda.

For mobile and web apps, AWS AppSync additionally provides local data access when devices go offline, and data synchronization when they are back online.

Securing API calls

It's important to secure your APIs in order to identify the user making the API call. There are different types of authentication methods, such as basic authentication, token-based authentication, and secure keys.

Basic Authentication

Basic authentication is a simple authentication method that involves sending a username and password in the request headers. The server verifies the credentials and sends back a response with the requested data. Basic authentication is not considered a secure method as the credentials are sent in clear text, making them vulnerable to interception. It should only be used for low-risk applications or in conjunction with other authentication methods.

Token-based Authentication

Token-based authentication is a widely used authentication method for APIs. It involves generating a token after successful authentication and sending it to the client. The client includes the token in subsequent API calls to authenticate itself.

Two popular token types are **JSON Web Tokens (JWT)** and **OAuth** access token.

JSON Web Tokens

JSON Web Tokens is a standard for securely transmitting information between parties as a JSON object. JWT tokens are digitally signed to ensure their authenticity and can include information about the user, such as user ID and user roles.

To authenticate a user using JWT, the server generates a token after successful authentication and sends it to the client. The client includes the token in subsequent API calls to authenticate itself. The server verifies the token's signature and extracts the user information from the token.

OAuth

OAuth is an authentication and authorization protocol widely used in modern web applications. OAuth allows third-party applications to access resources on behalf of a user without exposing the user's credentials. OAuth involves three entities: the user, the client (the third-party application), and the authorization server (the server that hosts the protected resources).

The client requests authorization from the user to access the protected resources. The user grants authorization, and the client receives an access token from the authorization server. The client includes the access token in subsequent API calls to authenticate itself.

Secure Keys

Secure keys are another mechanism to secure API calls. Secure keys are generated by the server and sent to the client after successful authentication. The client includes the secure key in subsequent API calls to authenticate itself.

Secure keys can be of two types: API keys and HMAC keys.

API keys are simple, randomly generated keys used to authenticate clients to the API. The server generates an API key after successful authentication and sends it to the client. The client includes the API key in subsequent API calls to authenticate itself. API keys can be used for rate limiting or access control. However, they are not considered a secure method as the API key can be intercepted or stolen, allowing unauthorized access to the API.

HMAC (Hash-based Message Authentication Code) keys are more secure keys used to authenticate clients to the API. HMAC keys involve a shared secret key between the server and the client. The client generates a signature using the secret key and includes it in the request headers. The server verifies the signature using the secret key. HMAC keys provide message integrity, authenticity, and non-repudiation. They are widely used by popular APIs such as Amazon Simple Storage Service (Amazon S3) and Twitter.

How to Build a Public Facing API

Explore technical content series crafted by AWS Startup Solutions Architects to help guide you in setting the foundations needed to start building quickly and easily.

Read AWS guidance »

Constructor

AWS Partner MACH spotlight

Home living ecommerce company home24 drives double-digit growth in search conversion rates after implementing Constructor.io

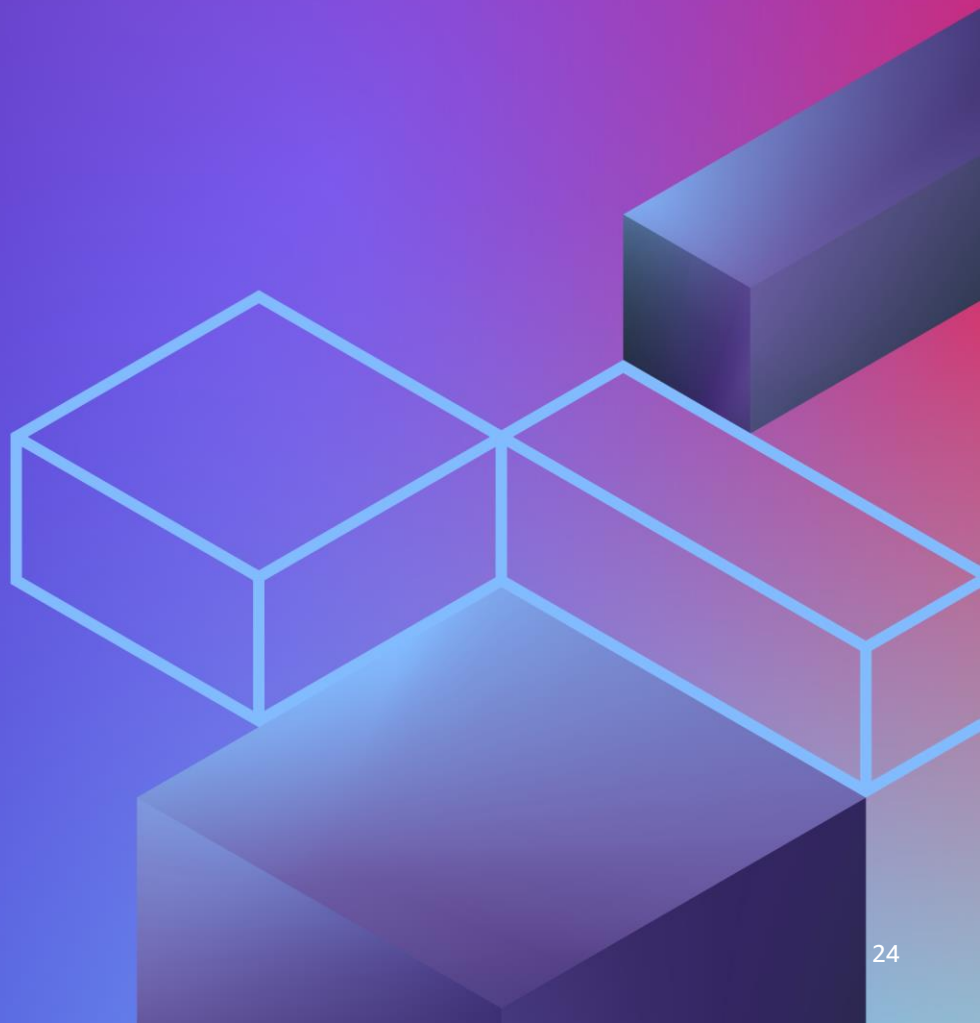
home24 achieves significant growth with search using Constructor.io on AWS.

Read the story »

Constructor Product Discovery and Search in AWS Marketplace »

aws FOR RETAIL

Cloud-native SaaS



Cloud-native SaaS

Cloud-native is the software approach of building, deploying, and managing modern applications in cloud computing environments. Modern companies want to build highly scalable, flexible, and resilient applications that they can update quickly to meet customer demands. To do so, they use modern tools and techniques that inherently support application development on cloud infrastructure.

These cloud-native technologies support fast and frequent changes to applications without impacting service delivery—providing adopters with an innovative, competitive advantage.

The Cloud Native Computing Foundation (CNCF)—an open-source foundation that helps organizations kick-start their cloud-native journey—defines the technological blocks of cloud-native architecture as immutable infrastructure, microservices, declarative APIs, containers, and service meshes.

A common pattern for cloud-native applications is event-driven architecture. Event-driven architecture is a modern architecture pattern built from small, decoupled services that publish, consume, or route events. An e*vent* represents a change in state, or an update, (for example, an item placed in a shopping cart or an order becoming ready to ship).

Event-driven architecture promotes loose coupling between producer and consumer services, which makes this architecture approach particularly suitable for MACH solutions. AWS services commonly produce or consume events, making it easy to build solutions with an event-driven architecture.

Cloud-native SaaS

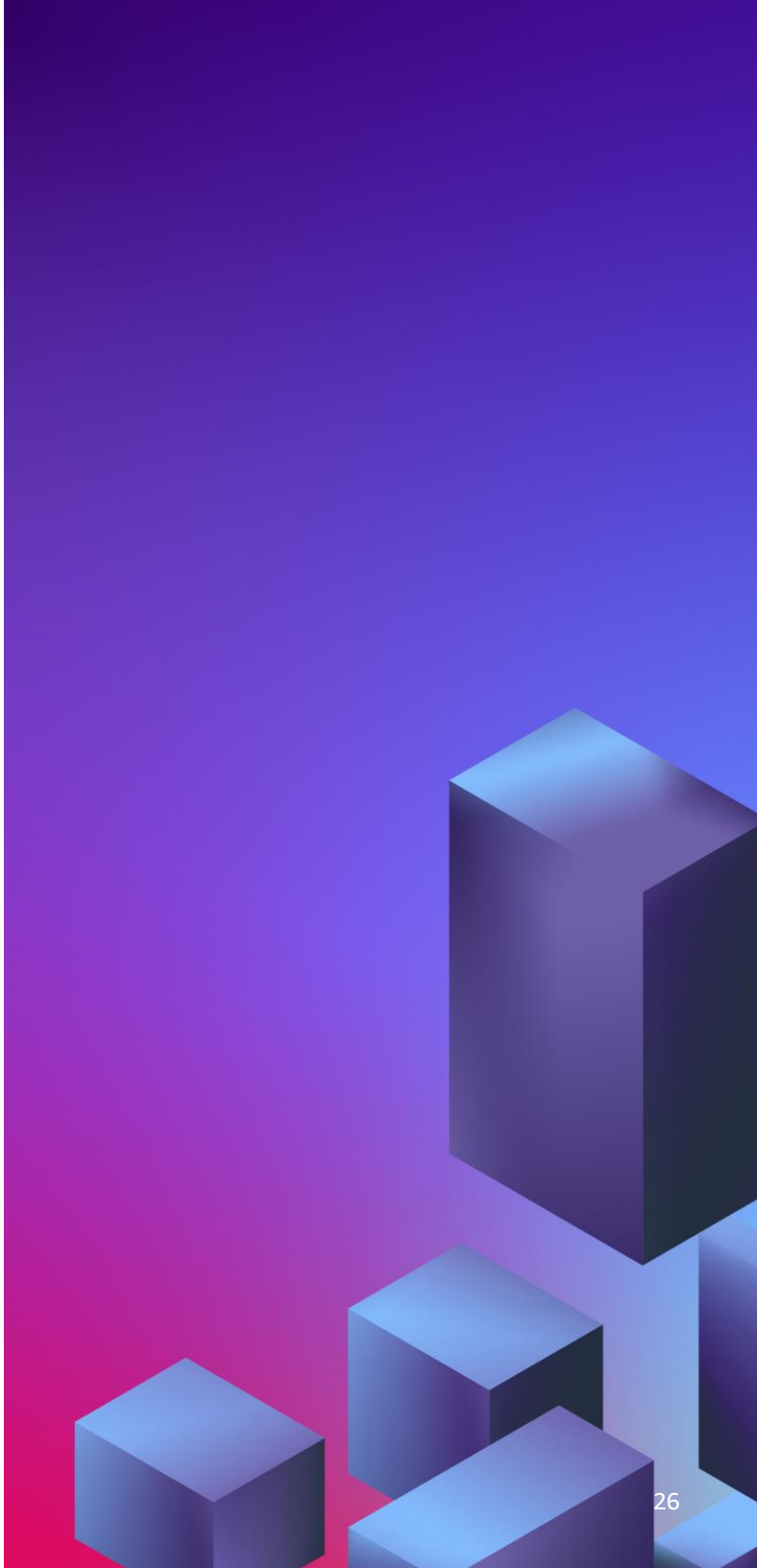
AWS services such as Amazon EventBridge, Amazon Simple Notification Service (Amazon SNS), Amazon Simple Queue Service (Amazon SQS), and AWS Step Functions include features that help customers write less boilerplate code and build event-driven architecture faster:

You can use Amazon EventBridge to build event buses for event-driven applications at scale using events from SaaS applications, other AWS services, or custom applications. EventBridge applies rules to route events from event sources to different targets. Targets can include AWS services such as AWS Lambda, Step Functions, and Amazon Kinesis, or any HTTP endpoint through EventBridge API destinations. AWS Partners can enhance and extend their platform by publishing their events to EventBridge. They can rely on EventBridge to manage the routing of those events to customers in a reliable and secure manner. Learn more in the EventBridge Partner Onboarding Guide.

A popular integration for event-driven architecture use cases is Step Functions, in which events trigger specific workflows. AWS Step Functions includes Workflow Studio, a low-code visual workflow designer that builders use to orchestrate different AWS services. You can use Workflow Studio to build distributed applications, automate IT and business processes, and build data and machine learning pipelines using AWS services.

We recommend using Amazon SNS to build applications that react to high throughput and low latency events published by other applications, microservices, or AWS services. You can also use Amazon SNS for applications that need very high fanout to thousands or even millions of endpoints.

Amazon SQS offers a secure, durable, and available hosted queue service that you can use to integrate and decouple distributed software systems and components. Amazon SQS offers common constructs such as dead-letter queues and cost allocation tags.



Cloud-native SaaS

Modern cloud-native applications are often delivered through a Software-as-a-Service (SaaS) model. Software-as-a-Service is a cloud-based software model that delivers applications to end-users through an internet browser. SaaS vendors host services and applications for customers to access on-demand.

The SaaS model gives businesses access to powerful software that would previously have been too expensive or resource-intensive to run on their own.

Cloud-native SaaS solutions are composed of several independent services, following the microservices approach described earlier. The cloud-native SaaS model allows software providers to respond to customers' demands quickly and safely without disruptions.

AWS SaaS Factory Program

For AWS Partners building SaaS application, AWS offers the AWS SaaS Factory Program. This program is designed to help AWS Partners at any stage of the SaaS journey, from building new products to migrate existing applications, or optimize SaaS solutions on AWS.

The AWS SaaS Factory Program give AWS Partners direct access to technical and business content, best practices, solution architects, and SaaS experts that can guide and help you accelerate delivery of SaaS solutions on AWS.

"The data tells us that too many companies are stuck keeping the upgrade wheel turning. If you add the cost of time and business stand still, the absolute cost is much higher. At the same time, fewer companies see themselves as agile early adopters, and fewer also see themselves as being ahead of the competition compared to our research a year ago. The pace of transformation is relentless but the cost of not innovating is much higher.

While transitioning to MACH is no small undertaking, continuing the status quo is an ongoing, big undertaking, especially for larger organizations, which needs addressing. We're not going to wipe legacy out of the picture overnight. However, those moving toward MACH are better equipped to mitigate future obstacles."

Casper Rasmussen, MACH Alliance President

mindcurv

AWS Partner MACH spotlight

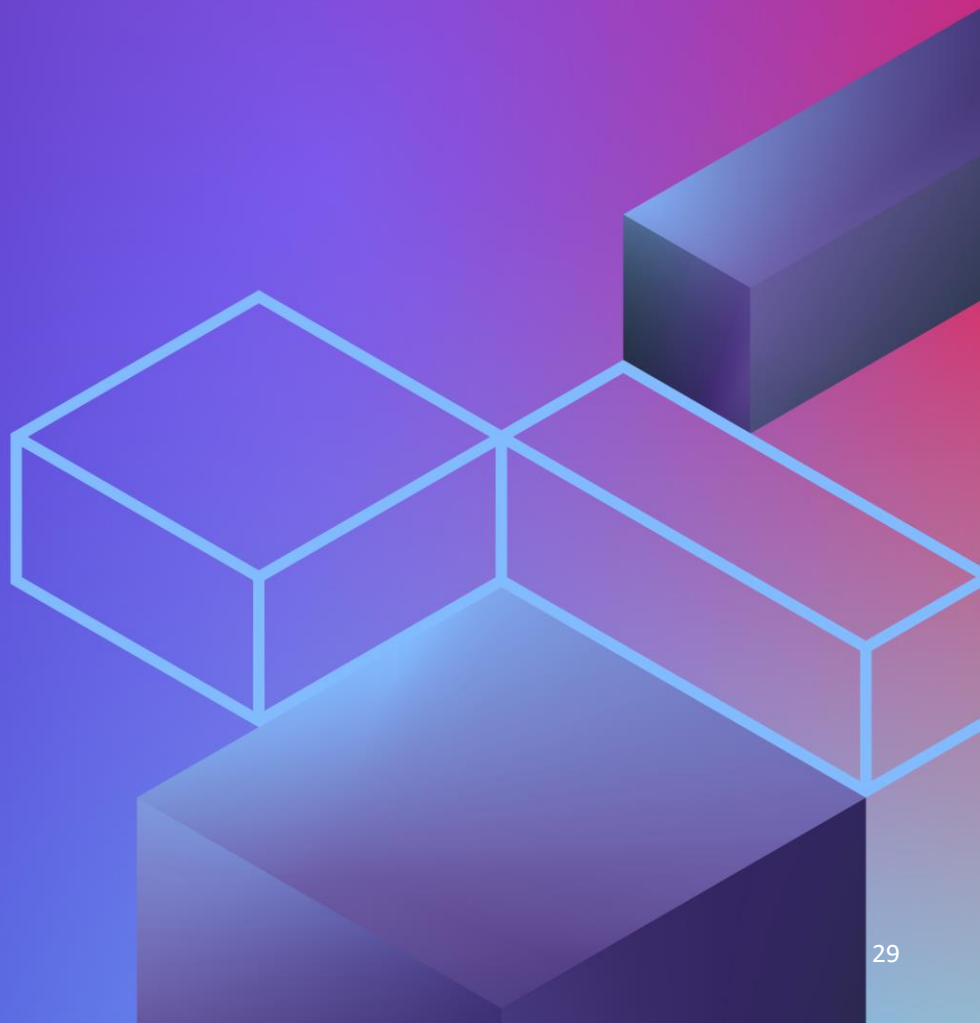
Innovating Foodl, Netherland's first B2B food services marketplace

Mindcurv partnered with Foodl to launch the first open food marketplace in the Netherlands. Read more to learn how Foodl enabled a composable digital shop, where each component was plug-and-play, scalable, and replaceable.

Read the story »

aws FOR RETAIL

Headless



Headless

Headless architecture is an architecture pattern in which the front-end (for example, the user interface) and the back-end (for example, the business logic and data storage) are decoupled from each other. This means that the front-end and back-end are developed and maintained independently, and communicate with each other through APIs.

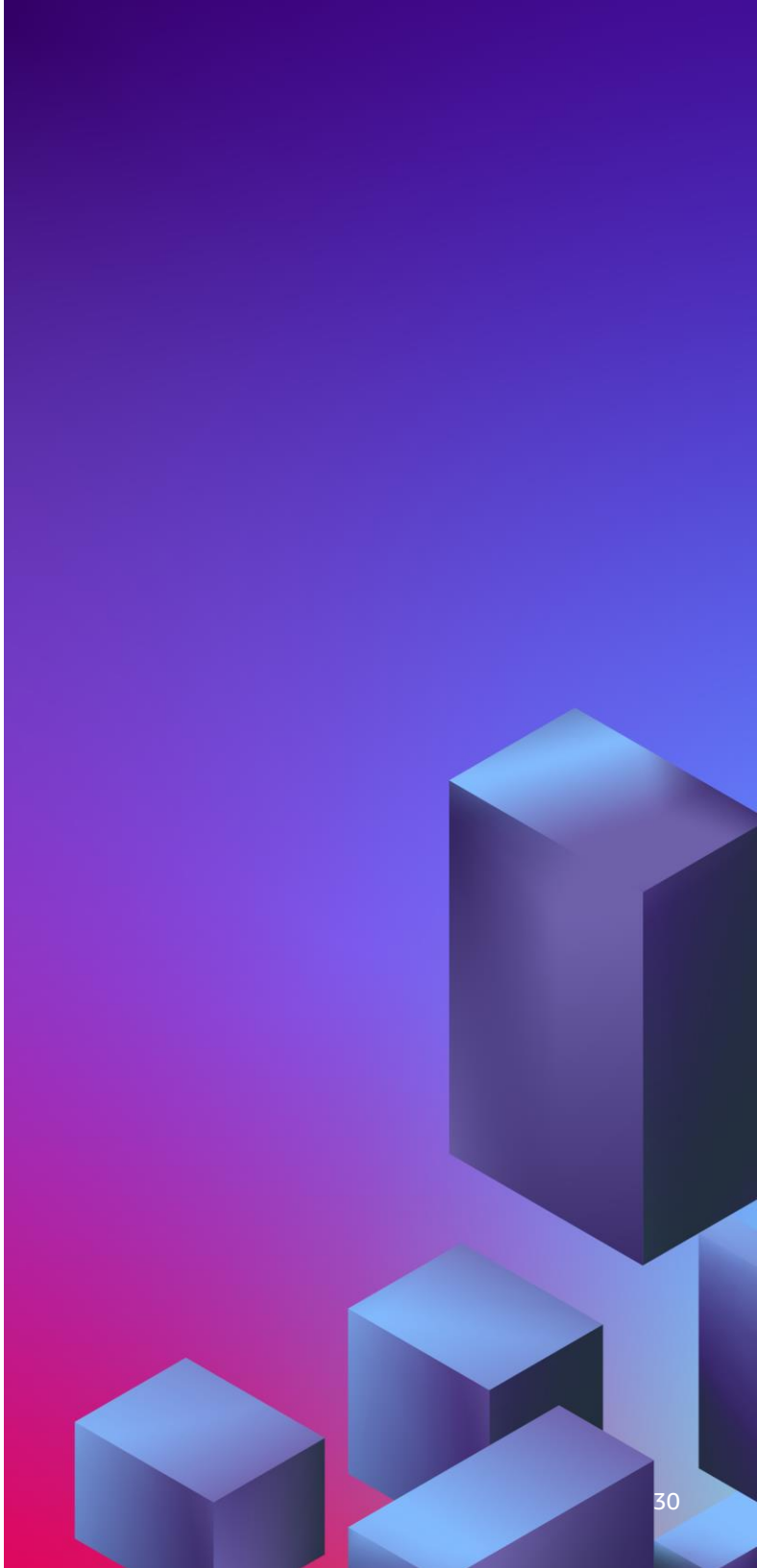
In a headless architecture, the front-end can be any device or application (such as web browsers, mobile apps, or IoT devices) and the back-end is typically implemented by microservices that provide data and functionality through APIs.

Headless architectures allow for fast innovation at the front-end and sales channels, while providing the biggest feature set and smooth integration in the back-end systems (CRM, ERP, finance, billing, reporting) out of the box.

One common approach that has gained popularity is micro-frontend architectures. The micro-frontend architecture introduces microservice development principles to front-end applications. In a micro-frontend architecture, development teams independently build and deploy “child” front-end applications.

These applications are combined by a “parent” front-end application that acts as a container to retrieve, display, and integrate the various child applications. In this parent/child model, the user interacts with what appears to be a single application.

In reality, they are interacting with several independent applications, published by different teams. Refer to this reference architecture to learn how to implement micro-frontends using a server-side rendering approach with AWS services.



Headless

AWS makes it easy to design, build, and host your headless front-ends. AWS Amplify is a complete solution that lets you build, ship, and host full-stack applications on AWS:

Amplify Studio

AWS Amplify Studio is a visual interface that allows developers to easily build and ship complete web and mobile apps in hours. With Amplify Studio, you can quickly create rich user interface (UI) React components, and connect a UI to your back-end in clicks. Amplify Studio exports all UI and infrastructure artifacts as code that you would write yourself, so you can maintain full control over your app design and behavior.

Amplify Hosting

AWS Amplify Hosting is a fully managed CI/CD and hosting service for fast, secure, and reliable static and server-side rendered apps that scale with your business. Amplify Hosting supports modern web frameworks such as React, Angular, Vue, Next.js, Gatsby, Hugo, Jekyll, and more.

KIN+CARTA

AWS Partner MACH spotlight

Kin + Carta builds an end-to-end platform for Cazoo using MACH architectural principles on AWS

Learn about how Kin + Carta designed, built, and operationalized an end-to-end platform for Cazoo—UK's used car sales unicorn startup—in just 4 months. With Cazoo, customers can now choose from thousands of top quality cars, pay with a credit card, choose part-exchange or other financing options, and get their chosen vehicle to their doorstep in as little as 72 hours. Kin + Carta provides both, the technical expertise, as well as the strategic change adoption, to ensure a smooth and successful switchover to modern MACH architecture.

Read the story »

Contentful Headless CMS by Kin + Carta in AWS Marketplace »

About the authors

Daniele Stroppa

AWS Retail & CPG Partners and Solutions Architecture Leader, AWS

Daniele leads solutions architecture and technical strategy for Retail & CPG partners on AWS. Daniele is passionate about modern software development methodologies and helping developers delivering the best results. Over his 15-years career, Daniele helped organizations across the Telco, Travel & Hospitality, and CPG industries build scalable, resilient, and modern applications. He holds degrees from the University of Milan and the University of Bradford.

Krithika Ganesamoorthi

Global Partner Solutions Architecture Leader, Consumer Verticals, AWS
MACH Alliance Technology Council Co-Chair

Krithika Ganesamoorthi is a Global AWS Solutions Architecture Leader for AWS Partners in consumer verticals. Leveraging her experience, she is responsible for solutions architecture and the technical strategy for technology and consulting partners on AWS. She recruits, nurtures, and helps partners transform their solutions on AWS. She is an advocate for partners within Amazon Retail, AWS Sales, and AWS Product teams.

Getting started with MACH on AWS

Since the launch of AWS in 2006, we've supported retailers and partners to build modern architectures to facilitate rapid change and innovation with their websites and software applications. This includes the use of microservices, APIs, and a build-for-the-cloud design.

AWS offers numerous benefits that make it the ideal platform for building MACH solutions. Its comprehensive suite of services, API-first approach, cloud-native design, and headless solution provider capabilities provide businesses with a reliable and robust cloud infrastructure to build, deploy, and manage their MACH applications with ease.

Find AWS Partners in MACH

Download AWS Partners in MACH infographic »