Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

© 2021 Amazon Web Services, Inc. or its affiliates. All rights reserved.
Contents

Introduction 5

The Four Fundamental Ecommerce Architectures 6
   Custom-built Microservices Architectures 7
   Commercial Off-the-Shelf (COTS) Ecommerce Suites 11
   Custom-built Monolithic Architectures 16
   Custom-built Frontends with Headless COTS Backends 18

How to Setup an Effective Ecommerce Tech Organization 22
   Customer-centric, Lean-agile, and DevOps 22
   Cross-functional Product Development Teams 23
   Centered Around Long-lasting Products, Not Short-term Projects 23
   Delivering Value, Not Features 24
   Loosely Coupled and Highly Aligned 24

The Importance of Analytics, Artificial Intelligence, and Machine Learning 25
   Reference Architecture 26

How to Build, Migrate, and Roll Out 28

Common Mistakes 30
   Not Using a Data-driven Approach 30
   Major Release Philosophy 30
   Not Defining a Strategy First 30
   Focusing on Local Optimization 30

Conclusion 31

Authors and Contributors 32

Sources 33
Abstract

Establishing the right ecommerce architecture is critical to ensuring digital sales channels can support your overall business strategy. It’s important to continuously reevaluate your ecommerce architecture to verify it reflects changes to your business strategy, emerging consumer trends, and advancements in technology. This paper lays out the decision process for ecommerce modernization, with tradeoffs and lessons learned, to help executives conduct a thorough evaluation. It also addresses organizational alignment and how to deliver your chosen approach. Throughout the paper, we intersperse ways in which Amazon Web Services (AWS) can help you achieve these goals.
Introduction

According to Salesforce Industry Insights, global ecommerce revenue increased by an average of 62.5% (YoY) from Q2 2020 to Q1 2021. Globally, mobile ecommerce traffic share is 69%, and the mobile ecommerce order share is now 56%. The Covid-19 pandemic accelerated ecommerce adoption and created a lot of potential opportunities, and it also put pressure on existing and upcoming ecommerce players. With ecommerce accounting for just 18% of total global retail sales there is huge growth potential.

CEOs of nearly all industries ask their CTOs and CIOs to provide or improve ecommerce capabilities, not just in response to demand created by the Covid-19 lockdown but also to support and connect the increasing number of sales channels. The industry started with rudimentary websites, added mobile functionality optimized for smart phones, and progressed to native and hybrid apps for phones and tablets. Now, digital sales channels need to seamlessly integrate with physical sales channels as well as marketplaces. Emerging trends include social commerce and especially live streaming ecommerce. Users have become accustomed to the top-notch digital experience, with ease of use, reliable service, site responsiveness, and competitive pricing, provided by digital market leaders.

CTOs need to be aware of these expectations and emerging trends and should provide an ecommerce architecture and organization that meets the following core requirements:

- Delivers **business value**
- **Adapts quickly** to new industry trends
- **Integrates easily** with all kinds of marketplaces and platforms
- Fulfills the **core non-functional requirements** of ecommerce systems, like performance, availability, scalability, and maintainability

Key differentiators that will separate leaders from laggards are optimization, experimentation, and innovation at scale as well as the seamless integration of online and offline shopping experiences.

“More than 50 percent of companies whose revenue growth is in the top 10 percent are more effective than their industry peers at testing ideas, measuring results, and executing changes to products, services, and ways of working.”

mckinsey.com
The Four Fundamental Ecommerce Architectures

When we talk to Chief Technology Officers (CTOs), they most frequently mention these core capabilities of their ecommerce architectures:

- The **speed of innovation** at the digital and physical frontends to support sales channels.
- The effort to **integrate** the increasing number of other IT systems.

CTOs focused on innovation usually employ in-house development teams. Additionally, they focus on revenue and other metrics, optimize for speed, and experiment quickly. They have a high demand for data and flexibility, and ecommerce is an essential part of their business. Software development is part of their organization, even if they use an external workforce as additional capacity. Because they know how to build software, make-or-buy decisions are easier, with a stronger tendency toward make.

On the other hand, some CTOs focus on integrating different systems and prefer to source best-of-breed solutions from external vendors. They focus on cost reduction and optimization of efforts. They buy ecommerce functionality that can be easily integrated into their backend systems, like supply chain management or enterprise resource planning. They don’t usually experiment. Ecommerce is not their core business. It’s just a web shop, sometimes without an app. They have no substantial software development capabilities and use agencies and consulting partners when necessary. Because they lack builder skills, they have a strong tendency to buy versus make.

If you use innovation and integration in a two-by-two matrix, you can easily map the four kinds of ecommerce architectures:

- Custom-built **microservice architectures**
- Commercial **off-the-shelf (COTS) ecommerce suites**
- Custom-built **monolithic architectures**
- Custom-built **frontends** with headless COTS backends
Custom-built Microservices Architectures

A custom-built ecommerce application that follows a microservices, self-contained system, or serverless approach is usually created by a software engineering team and is often found in companies with a strong builder mentality. This architecture is optimized for speed, agility, and innovation. It often has a diverse set of frontends and requires more integration effort with backend systems (CRM, ERP, finance, billing, reporting) compared to COTS that have readily available connectors. Each customer-facing web page or app screen interacts with several small microservices that can be built, maintained, and deployed independently.

These systems are usually highly optimized with intuitive user interfaces to maximize customer conversions. The highly flexible microservices architecture and the organization's agile build-measure-learn mindset encourages high-frequency experiments to improve existing functionality and deploy new features.

COTS software might be used for some specialized business functionality (e.g., product search, payments, or recommendations). Make-or-buy decisions are made on a more granular level. Microservices architectures can easily integrate cloud services to become more flexible, secure, cost-effective, elastic, and globally available. These architectures integrate cloud services for computing, storage, monitoring, and backup as well as more complex services, like analytics, artificial intelligence (AI), machine learning (ML), user management, and recommendations. With flexible architectures, cost optimization is possible at all levels.
Organizations with a builder innovation focus need a workforce with strong software engineering and operations skills as well as business acumen.

Organizations that have adopted a build and run model, who want to harness the benefits of end-to-end ownership, will benefit from this architecture. Teams can be organized around business capabilities, own their destiny, and move at their own pace. To facilitate rapid delivery, the feature teams need platform support for basic cross-cutting concerns, like security, monitoring, and continuous integration.

However, a risk of the custom-built microservices architecture approach is an increase in complexity. Organizations can mitigate this risk by setting a high bar for engineering and operational excellence. The culture should embrace quick changes and continuous learning. Otto, Zalando, Adidas, Nike, and Dunelm have all successfully implemented a custom-built microservices architecture.
A key characteristic of this architecture is the **high degree of decoupling between components**. The frontends are independent from the backend services. These are either single page or mobile applications, or they could be chat and voice interfaces.

Starting with a clean-definition of consumer-friendly APIs is the best way to achieve decoupling. In the backend, services communicate asynchronously using an event-driven approach when possible to improve reliability, availability, and efficiency.

The architecture introduces a clear separation between the different bounded contexts. Each is implemented using a modular microservices architecture. As all concerns are well represented by API contracts, the backend can serve multiple frontends, such as mobile applications, in-store digital touchpoints, or social channels.

Each backend service can be scaled, deployed, and developed independently. A team could own one or more of the services and manage its lifecycle without coordination with other teams as long as the API contract remains stable.
Further downstream, the functionality for each service uses dedicated managed services, fit for purpose. For example, Amazon Elasticsearch Service can be used for search, and Amazon Personalize can be used for product recommendations.

With this approach, it’s easy to create and deploy new experiences and evolve the systems of record and data stores as requirements or consumer transparency needs change, if the API contracts remain unchanged.

Provisioning of cloud services and the deployment of the custom-built microservices are fully automated. This way, updates of any kind are performed quickly and in isolation, reducing the impact and possible blast radius of errors.

Challenges with this approach are inherited from microservice architectures and distributed systems. The overall system complexity grows with the number of independent services, and therefore, managing complexity requires a cohesive approach to automation, logging, monitoring, and distributed tracing. These concerns are well addressed with AWS managed services because they are seamlessly integrated and support a rapid development lifecycle.
Commercial Off-the-Shelf (COTS) Ecommerce Suites

The ecommerce application is licensed from a COTS vendor, like SAP, Adobe, Oracle, or Salesforce. This architecture is designed to offer maximum features out of the box with easy integration to backend systems, like CRM, ERP, and finance. The application suites consist of many modules that are initially easy to implement and may require some customization effort. The advantage of adopting this architecture is speed to market because the customization effort is minimal in the pre-rollout phase. Since there are many features available out of the box, less development staff is required. With a careful product selection process to match the ecommerce application to the organizational requirements, the overall change management impact on both IT and business users is minimal. Finally, with a well-supported enterprise IT ecosystem, integrations to backend and third-party systems will be easier to deploy and maintain.

Innovation on top of these suites is hard. Customers must wait for the vendor to implement a feature, or they have to build it themselves, which can be challenging if the customer doesn’t have the appropriate skills available in-house. Organizations with COTS are often dependent on the software vendor, the vendor’s in-house professional services organization, or a third-party consultant. Upgrades get harder as custom functionality is added to the core system. Some companies never apply upgrades, which puts them in a high-maintenance scenario. Sometimes customers are forced to do major updates for no tangible benefit. A result of these challenges is that experimentation, build-measure-learn-thinking, and optimization are not widely adopted. Companies typically don’t optimize on KPIs or customer needs. Adding or exchanging specialized business functionality is also hard because the vendor’s systems do not support this. The maintenance costs of the solution typically grow over time in proportions larger than the level of customization. Data in a COTS system is often hard to access and not easy to interpret and correlate.
Although the license cost of COTS software is easy to calculate, it’s hard to optimize on an operational level. **COTS architectures can become very expensive at scale.** The most common cost drivers are over provisioning of compute and storage resources to handle seasonal peaks, license costs, and fees for third party consulting services. However, migrating the application to the cloud will likely lead to lower total cost of ownership, better security, more elasticity, and agility. It’s possible to modernize core infrastructure components. The solutions offer good coverage of typical use cases and functionality, but **don’t match the performance of best-of-breed solutions in any category.**

The organization must have employees who are familiar with the COTS suite. Because people with these skills and experience are often hard to recruit, the organization might have to depend on expensive external consultants and recruiting agencies to help find suitable talent. Due to limited capacity and know-how, **the full potential of a COTS solution is often not achieved.**

However, if organizations migrate these systems to the cloud, and as a long term strategy, move their ecommerce architecture into one of the upper quadrants focused on innovation, it will be easier to attract talent. **Modernizing** the technology stack is an important step to introduce a **culture of innovation.**

**Sainsbury’s**, the UK’s second largest supermarket chain, successfully moved to AWS and achieved a 70-80% performance improvement. The company went from five or six releases per year to multiple releases per day and reduced infrastructure utilization by 30%.
COTS platforms are either self-hosted or managed by the platform vendor in a software-as-a-service (SaaS) model. Both cases involve the cloud. First, let’s investigate some of the common denominators.

To react quickly to customer demand, it is beneficial to have a continuous integration (CI) platform to support faster release cycles. More and more COTS vendors are building primitives to support automation. The configuration might be changed by invoking APIs, using CLIs, or uploading new complete system configurations.

The same CI benefits from custom-built software are equally relevant and applicable and include delivering updates faster, finding bugs earlier, and improving developer productivity. AWS provides a combination of services aimed to build, package, and automatically test new versions.
Any COTS platform requires data in the form of product catalogues, product pricing, marketing content, promotions, stock information, and the like. While most of this data can be manually configured via the available interfaces, automating data flows from source systems will eliminate errors and create efficiencies. Whether for real-time or batch data transfers, AWS offers several **application integration services** to simplify communications between decoupled components.

Metrics and analytics from ecommerce operations are a critical enabler of the build-measure-learn cycle. Data drives fundamental business decisions and helps prioritize the deployment of new features. Analytics capabilities can provide a single pane of glass into all aspects of an ecommerce initiative and help compute key performance indicators, such as conversion rates, average order size, revenue per visitor, site traffic, and bounce rate. While COTS platforms might offer built-in data analysis capabilities, it’s beneficial to evaluate additional data segments both internally and externally, like customer information, stock levels, supply chain data, social media sentiment, and third-party marketplace data.

**For self-hosted COTS platforms, there are many advantages of migrating to the cloud.** Even a monolithic ecommerce application will require one or more database engines and several compute resources. On-premise hosted platforms must have the capacity to handle daily and seasonal load spikes, and with the cloud, scalability fundamentally changes with options to automatically scale elastically and on-demand. **Managed cloud-native databases** are a maintenance-free database option with out-of-the-box automated backups and point-in-time restores for disaster recovery. Using these databases depends on the platform provider’s official support parameters, and the databases might need to be certified to be fully supported.

For enterprises operating across multiple countries and continents, AWS offers services that support a globally distributed ecommerce application. Depending on the technology choices, some services will have built-in cross-regional capabilities, further simplifying integrations and maintenance.
High availability can be achieved within and across regions. AWS provides a variety of services to ensure business continuity in case of individual servers, storage devices, or data center outages.

It is usually hard to optimize server-side performance of COTS ecommerce applications. Scaling decisions on the backend requires data insights about infrastructure utilization, whether on compute, storage, or database levels. Meanwhile, client-side performance can be improved with a content delivery network (CDN), like Amazon Cloudfront.

Companies that migrated their ecommerce COTS platform to the cloud found it easier to evolve their architecture to properly size each layer of the stack with the best mix of cost efficiency and performance by implementing auto-scaling or expanding functionality using cloud managed services. Companies that want to enhance their consumer experience more quickly have built a custom frontend while using the same COTS ecommerce backend. Depending on the API maturity of the COTS platform, some companies implement a lean microservices layer that hides the complexities of the underlying platform. Other customers have migrated end-to-end functionality slices, reimplementing or replacing parts of the backend, as a first step of the modernization journey.
Custom-built Monolithic Architectures

This architecture is no longer a predominant first option for companies. However, many large enterprises are still running such ecommerce systems.

In this architecture, software engineering teams create a custom ecommerce application using traditional layered architecture patterns. The organizations using this architecture are usually early ecommerce adopters. They were originally focused on innovation or preferred to build their own application rather than use a COTS ecommerce suite.

Monolithic architectures have many downsides. They become complex and are hard to maintain. Changes cause downstream side effects, which increases the development and testing effort. Integrating new backend systems or adapting new frontend channels is usually slow and error prone. Because the entire application is custom, companies can experiment and innovate, but to a very limited degree because it’s difficult to iterate at a fast pace with a monolithic architecture.

Despite the inherent complexities of monolithic applications, it is possible to optimize costs. However, it requires a lot of effort. Major cost drivers tend to be expensive COTS databases and on-premises data center infrastructures.

Companies need software engineering and operational skills because monolithic architectures require the most maintenance and management, compared to the other architectures. As a result, the engineering teams have little time to innovate, which often creates conflicts with business stakeholders. Companies with legacy monolithic ecommerce architectures typically find it difficult to recruit business and tech talent.

We recommend modernizing an ecommerce architecture in the cloud in the short term, and then later, move into one of the upper innovation-focused quadrants.
The same technical principles and concerns of the self-hosted COTS ecommerce platforms apply to custom-built monoliths as well. The significant difference is that existing engineering teams can more quickly evolve the functionality of a custom monolithic application to a modern architecture.

If modernizing a monolithic architecture is the goal, AWS offers services to address top priorities. For example, a caching service like Amazon ElastiCache can offload heavy database operations in the backend to improve site performance. New features, such as product recommendations or other personalization technologies, can be deployed without machine learning expertise with Amazon Personalize.

Available AWS application performance monitoring solutions from the AWS Marketplace can provide a deep level of insight and understanding of application usage. Together with the business operation insights and overall strategy, these add-on solutions will provide valuable insight to define a new target architecture. They will also help prioritize pain points and new features.

As a first step, automating deployments and testing can put companies on a path to evolve a monolithic architecture. With stable regression testing in place, companies can replace specific functionality by externalizing it for a smoother transition. Meanwhile, introducing API contracts toward the frontends, will simplify functionality decomposition to facilitate growth around new bounded contexts.
Custom-built Frontends with Headless COTS Backends

This architecture promises the best of both architectures already described. However, there are some downsides. In this architecture, the frontends (web, mobile, social, video, and physical store) are created by software engineering teams while the ecommerce backend is a COTS solution from vendors, like SAP, Spryker, and Elastic Path.

These vendors use a headless architecture approach where the ecommerce business logic is encapsulated in a backend microservices layer. Using an API, this backend microservices layer supports the different frontends with the same business logic (product catalog and search, user management, and inventory). The headless backend consists of many modules and features that are easy to implement and may require some customization.

The solution is optimized for rapid innovation on the different frontends that support the various sales channels with a predefined set of standard functionalities. The solution also offers easy integration out of the box with backend systems, like CRM, ERP, finance, billing, and reporting.

Although innovation is easy on the frontend, it is harder on the backend because typically the software vendor must build new features. However, with some COTS solutions, customers can build new backend functionality by themselves, but they must have access to builder capabilities. Companies can use Amazon AppFlow to create a modern event-driven architecture around a COTS system. AppFlow transfers events from these third-party systems and pushes them into a cloud-based event bus that is the backbone for any custom build service.

Companies can make decisions about whether to make or buy on the component level. To some extent, depending on the restrictions of the COTS vendor, companies can use cloud services too. If the backend consists of multiple systems, some integration and orchestration effort is required.

These systems are often highly optimized from a user interface and usability perspective because of the highly flexible frontend architecture. Experimentation and optimization that touch backend services are more difficult and time consuming, which leads to a limited optimization of the overall conversion rate. There are specialized digital agencies that can help companies implement the COTS solution and also develop and deploy modern frontends if in-house resources are not available.
Cost optimization is easy on the frontend but harder at the COTS backend. Comparable to ecommerce suites, **this architecture might become expensive at scale**. Cost optimization is usually only possible through contract negotiations with the COTS vendor and by using cloud infrastructure. Data in the COTS system is often hard to access and not easy to interpret and correlate. Sometimes customers are forced to do major updates from which there is no tangible benefit to the actual use cases. **Companies will need software engineering and operations skills at the frontend and COTS admins at the backend.**

Organizations with this kind of architecture need strong software development skills to manage the frontend and the overall system. They also need employees who are familiar with the COTS suite. Because people with these skills and experience are often hard to recruit, companies might become dependent on expensive third-party consultants. **Due to limited capacity and know-how, the full potential of a COTS solution is often not achieved.**
By developing frontends from the backend, companies have maximum flexibility and agility to choose different technologies. Additionally, a micro-frontend approach with decoupled modern web functionality gives companies even more autonomy and speed.

Developers can connect the frontends to the backend through the API layer available with the COTS headless backend. However, COTS vendor APIs may be too verbose, very complex, or follow a different API approach than your organization. Therefore, it is beneficial to introduce a lean microservices layer as proxy or façade to the commercial backend. Consumers of the API platform will have a consistent experience to speed the learning curve and adoption. Each microservice should be organized around a business entity or functionality, such as products, customers, or search. This approach also hides unnecessary backend complexity so companies can aggregate and orchestrate multiple backend endpoints to simplify the API contract, providing a central layer for monitoring with additional functionality, such as caching, fallbacks, or rate throttling.
There is a wide range of commercial headless ecommerce products ranging from turnkey packages with the most common features to point solutions for specific ecommerce functionality. Companies that choose point ecommerce solutions will need additional products to provide an end-to-end cohesive experience. In this case, companies should use a custom-built API layer to ensure a consistent API contract.

In this architecture, cross-cutting concerns become very important. Because many systems are involved, companies will have additional challenges to ensure a high degree of security, availability, and performance. Continuous integration practices, including automated testing across the different systems, will likely raise challenges as well. These are key considerations when selecting and adopting multiple products to create a comprehensive ecommerce solution.

Data consistency is another challenge when combining platforms that share the same business entities (e.g., article or customer). For example, the search engine and the distinct product catalogue share the same product data. When data is changed in a system of record, companies can use a pub/sub system to send messages to both the catalogue and search applications. If they are two distinct systems with different integrations, one near real-time and the other one not, this will result in inconsistent data which will require several workarounds in the stack to compensate for inconsistencies, which could negatively impact the customer experience.

Ideally, the distinct applications should use the company’s centralized identity and access management (IAM) solution to manage security. At the same time, each platform should use identity federation for central IAM control.

From an operations perspective, the more systems involved, the more challenging it will be to manage the overall ecommerce solution. Dealing with incidents or problems will require well defined interfaces between systems with meticulous monitoring. A central logging and monitoring system will simplify operational processes. Log shipping to a central repository will provide a single pane of glass with dashboards and alerts.
How to Setup an Effective Ecommerce Tech Organization

Without the right organization, every ecommerce architecture described will fail to achieve its objectives. As a CTO, you have to build a working system out of culture, people, process, and technologies that delivers business value. An effective ecommerce organization includes people from marketing, product, and technology who work together with highly aligned common objectives and principles.

**Customer-centric, Lean-agile, and DevOps**

The key concepts of this organization are customer centricity, lean-agile, and DevOps. Customer centricity means that every feature you build delivers value to the customer. Methodologies at hand are Design Thinking and Lean Startup, especially the Lean Startup concepts of minimum viable products and the build-measure-learn cycle. At Amazon, we use an approach called “working backward,” where we write a press release and FAQs, add visuals to describe the customer, his or her needs and benefits, and any other important aspects of the problem and the solution. AWS shares this methodology with its customers.

Lean and agile methodologies, like Extreme Programming, Scrum, and Kanban, help ensure your organization can build features rapidly and securely with predictable, reliable processes.

DevOps includes all practices to deliver features to customers with predictable, reliable, repeatable, secure, and fast processes. You are good when every team can release features into production at any time, and the duration of a release from source control to production is measured in minutes instead of days.
Cross-functional Product Development Teams

Cross-functional product development teams build the core of this organization. “Cross-functional” means that these teams consist of people from different functional areas who work together to define, build, ship, and run the product, regardless of the functional reporting line of each individual contributor. Typical roles on these teams are product managers, software engineers, usability engineers, designers, and testers.

Healthy team size is around six to ten people. At Amazon, we follow a two-pizza rule that means teams can’t be larger than you can feed with two (American-size) pizzas. We strongly recommend that software engineers on these teams are generalists who can work on every part of the application stack, with a specialized technical expertise. These so-called t-shape or full-stack engineers can typically manage the most important elements of your ecommerce solution. Organizations using a COTS solution should also leverage cross-functional teams, even if there is less software development necessary. The software developers are then experts in the configuration of the respective COTS solution.

Centered Around Long-lasting Products, Not Short-term Projects

High performing cross functionality teams are responsible for a business process, such as checkout, product search, recommendations, and payments, across all technological layers (frontend, backend, and database). This is the best way for teams to fully understand customer needs so they can design the software quality for long-term maintainability.

This contrasts with project-based approaches where teams are formed again and again to deliver individual parts of a project. The team’s responsibility for the result ends with the delivery. Maintenance and maintainability are not the priority. Teams are merely groups of individuals working together.
Delivering Value, Not Features

All members of the ecommerce organization must share the fundamental belief that features do not necessarily deliver value. Too often, we see organizations that can release and deliver changes several times a day, but the KPIs of the organization do not change. The organization delivers output, but no outcome. They build features with the best intentions, but the features do not necessarily meet the needs of the customer.

To overcome this feature-driven mindset, companies need to adopt the build-measure-learn mindset first described in detail by Eric Ries in his book The Lean Startup: How Constant Innovation Creates Radically Successful Businesses. The hypothesis-driven thinking treats ideas as experiments that need to be tested. The organization should run several A/B tests in parallel to determine how a new idea performs against the current implementation or other possible solutions.

The success of these experiments must be judged based on data. For this purpose, the organization needs strong business intelligence and data analytics skills. Analytics, AI, and ML are essential capabilities of a successful ecommerce organization.

A simple, easy, and stress-free user experience is a key performance driver of every ecommerce site. In addition to product management, engineering, and data skills, user experience design is the fourth discipline the organization needs to master. User experience design includes the appealing look and feel of the frontends and also includes streamlined process flows. Especially around user experience, decisions must be made based on data and user feedback. Every change to the user experience must be A/B tested.

Loosely Coupled and Highly Aligned

Speed matters in business, with decisions, experiments, deploying new features, and releasing updates. No matter which architecture you choose, set up an organization that is loosely coupled and highly aligned to maximize the speed in your organization.

Loosely coupled means teams can function as independently as possible from each other and management leaders. To accomplish this, you need to reduce communications and system architecture dependencies in your organization.

Companies should increase autonomy without ending up in anarchy. Managers should decrease involvement in tactical day-to-day decisions and instead focus on the strategic alignment within the organization. To create alignment, use several instruments, like principles or tenets, strategy documents, and technology radars. On the other hand, OKRs and Kanban Flight Levels are good tools for the tactical orchestration of teams.
The Importance of Analytics, Artificial Intelligence, and Machine Learning

We cannot emphasize enough the importance of analytics capabilities throughout the whole organization—not just in IT and controlling, but especially in product management, user experience, marketing, supply chain, and logistics. IT provides the infrastructure to store, process, and visualize the data. However, **companies need to turn this data into insights, decisions, and actions.** For example, companies need data insights to manage performance-based marketing spends, optimize SEO and SEM activities, improve the conversion funnel, decide product assortment and pricing, and prioritize the product development roadmap. We call this a data-driven company.

Today, some decision-making can be automated with AI and ML services that provide ready-made intelligence for applications and workflows to help improve business outcomes. Amazon uses these same technologies to power our own businesses. You can build AI-powered applications without any ML expertise, such as using Amazon Personalize to make recommendations or Amazon Connect for contact center intelligence. Companies can use their own models and algorithms with Amazon SageMaker to help data scientists and developers prepare, build, train, and deploy high-quality ML models quickly.

The technical basis for this is data lakes. A data lake is a centralized repository that stores all structured and unstructured data at any scale. Data can be stored as-is, without having to structure the data, to process different types of analytics—from dashboards and visualizations to big data processing, real-time analytics, and ML to guide better decisions. The decentralized approach for data lakes is the data mesh architecture.

From an organizational perspective, analytics, AI, and ML infrastructure should be built and managed by a dedicated data platform team that’s responsible for providing the data platform itself as well as user support and training. Your product teams act as data producers and consumers. They are responsible for publishing data and metadata to the data lake as well as the definition and visualization of metrics and the creation of data-driven products and insights.
The proposed solution is a component-oriented architecture. The component orientation ensures separation of concerns, decoupling of tasks, and therefore, autonomy across different teams interacting with the platform, whether as producers or consumers.

At the center of the architecture, the storage layer provides durable, scalable, secure, and cost-efficient components to store vast quantities of data. Amazon Simple Storage Service (Amazon S3) matches all these requirements. Whether it’s structured or unstructured, data can be stored as Amazon S3 objects without requiring a predefined schema. Based on the data quality or the different use cases, the storage layer might further be organized into zones such as:

- **Raw** for storing data in its original form
- **Cleaned** after data has been validated and normalized
- **Curated** for hosting a consumption-ready state respecting organizational standards and data models. The data in the curated zone is typically partitioned and catalogued.
The ingestion layer uses a set of purpose-built AWS services to import data from a variety of sources, such as operational data sources, whether relational or NoSQL, using services such as AWS Data Migration Service, Amazon Kinesis Data Firehose for streaming data from internal or external sources like clickstreams or monitoring metrics. Organizations using SaaS applications, like Google Analytics, Salesforce, or Marketo, can leverage the fully managed integration service, Amazon AppFlow.

The purpose of the processing layer is to create or orchestrate multi-step data processing pipelines. AWS Glue and AWS Step Functions provide serverless components to build, orchestrate, and run pipelines that can easily scale to process large volumes of data. Companies can catalogue, validate, clean, and transform datasets that advance through the different zones in the storage layer.

The cataloguing and governance layer protects data in the storage layer and processing resources in all other layers. It provides mechanisms for access control, encryption, usage monitoring, cataloguing, and auditing. AWS Lake Formation provides these capabilities and more.

For data science, AWS provides a broad set of ML services. In ecommerce systems, ML is used for personalizing the user interface, making product recommendations, predicting demand and segmentation, as well as for contact center chat bots and post-call analytics. Predictive analytics, such as sentiment analysis and customer segments, can be persisted in the cleaned storage zone for analysis and consumption.

Companies can create an analytics consumption layer by using fully managed, purpose-built analytics services for interactive SQL, business intelligence dashboards, and batch processing. Amazon Athena is an interactive query service that uses standard SQL to analyze data in Amazon S3. When running queries against data using Lake Formation, the appropriate permissions are validated. Amazon Redshift is a fully managed data warehouse service that can host and process petabytes of data and run thousands of highly performant queries in parallel.

With Amazon Redshift Spectrum, companies can perform queries on datasets stored in Amazon S3 without loading data into the cluster. Lastly, Amazon QuickSight is a serverless business intelligence service to create rich, interactive dashboards for data visualization. Amazon QuickSight supports automatically generated ML insights for use cases, like forecasting, anomaly detection, and narrative highlights. Large enterprises that plan to scale analytics and data science capabilities should consider the data mesh architecture approach to data management.
How to Build, Migrate, and Roll Out

Our general advice on how to build, migrate, and roll out a new ecommerce system is to use an incremental, iterative approach that’s data driven.

With incremental development, the product is divided into fully functional, vertical slices. Each increment must deliver value to the customer. Start with a specific small piece of functionality, build it, and then add more and more functionality. Increments can be differentiated based on geography, customer types, product segments, data attributes, and non-functional requirements. For example, the very first increment of a product detail page could be a simple web page with a few attributes, such as the product image, name, and description. The next increments would add more pictures, product attributes, a checkout button, web tracking, monitoring, and so on.

Avoid building your system horizontally. For example, by creating the database with attributes and relationships, building business logic, and then adding frontends. Usually, this approach leads to a lot of speculative design, false assumptions, and complexity.

In iterative software development, the entire process, from idea to release, is run through several times in succession. This repetition hardens and improves your development process and often leads to better quality and predictability. At the end of an iteration, all finished increments are delivered to customers. In microservice architectures with continuous delivery and deployment, this iteration might happen several times a day.

In data-driven software development, a feature is not done when it is released to the production environment and approved by QA. Done means customer needs have been met, and you only know needs are met when you measure it. Developers need to gather data and anecdotes to show that the feature really works. For example, measure how many customers find the new feature, how many people use it, and how many customers successfully complete the task behind the feature. Build, measure, learn, and optimize until goals are achieved or efforts no longer bring improvement. If the success rate of the feature is low, consider killing the feature to reduce the overall complexity of your ecommerce system. From a system readiness perspective, this approach focuses on “delivering business value” rather than “feature completion.”
These three strategies are valid for all four architectures described in this paper. However, specific tactics and implementations processes may differ.

If you introduce ecommerce to your business, we recommend starting with a minimal feature set and learning about the needs of your users, instead of building a feature-complete shop that takes a long time to deploy. From the very beginning, focus on how to measure success with customer feedback to drive your delivery roadmap.

If you migrate to a new system from an old system, we recommend an incremental approach within a hybrid setup, going page by page and function by function. Always measuring and comparing the KPIs of the old and the new system with A/B testing and performance monitoring. For a while, both systems will operate in parallel, which adds complexity to your operations in the short term, but this approach mitigates risks on a very granular level.

If you cannot follow the incremental-hybrid approach and you have to build a new system in parallel, bring the new system into production very early with just a small amount of your overall traffic. Then, build and optimize every new increment until you achieve the overall business performance of the old system. As features are added and performance improves, you can increase traffic on the new system. You don’t have to be feature-complete with the old system. You need to match or surpass the performance of the system.
Common Mistakes

Not Using a Data-driven Approach

The most common mistake is not using a data-driven approach to ecommerce. An example would be making changes in a frontend user interface without measuring the impact and monitoring KPIs. Data insights should drive the roadmap and priorities for all business and technical decisions. Companies that capture insights but do not properly use the insights to drive decisions will not succeed. You should always prioritize value, experiment often, and validate hypotheses with data.

Major Release Philosophy

The ecommerce space typically has a constant flow of customers. Sometimes testing new hypotheses is possible with just parts of a larger feature. Companies should avoid waiting months or years to release a fully-fledged ecommerce solution. Small, incremental releases while constantly monitoring is a much better option.

Not Defining a Strategy First

Companies that have traditionally ignored the ecommerce channel for too long and run on antiquated technology, come to eventually realize the missed potential. When that happens, there is a temptation to set audacious targets and fund massive overhaul projects. Instead, a modern ecommerce platform requires a fundamentally different mindset, similar to adopting cloud technologies, as explained by AWS Enterprise Strategist Gregor Hohpe in his book Cloud Strategy. As a first step, companies should seek expertise to help define the right ecommerce strategy.

Focusing on Local Optimization

From a technical and organizational perspective, especially with decoupled, distributed architectures, there is a natural tendency to focus on local optimization. Lead architects should avoid ending up with a distributed ball of mud, even when it’s made with best of breed components. Instead, focus on the customer and optimize across journeys. In the realm of distributed architectures, another trap is increasing the number of systems or technical components beyond what the organization can manage. This applies equally to individual teams as well. Properly managing the cognitive load at all levels will pay back in the long run.
Conclusion

We discussed three types of architectures:

- **Custom-built microservices** – ideal for companies that focus on innovation and differentiation with an ambitious omnichannel program across web, mobile, social commerce, retail kiosks / endless aisle, and loyalty programs with many third-party partners.
- **COTS suites** – best suited for companies that want standard ecommerce functionality that’s not highly innovative or differentiated, with little to no omnichannel presence other than web and mobile.
- **Custom-built frontends with a commercial headless backend** – ideal for companies that want standard ecommerce processes with a highly differentiated user interfaces for different sales channels.

However, your **organization is more important than the ecommerce architecture and the tooling**. This includes the talent you attract as well as your mindset and processes. Fulfilling customer needs and delivering business value should be the focus rather than building features. Instead, build independent, cross-functional teams that are responsible for products and not projects. Data and customer feedback should drive your efforts.

The foundation for a data-driven ecommerce solution is a strong data platform that provides real-time insights, forecasts, and recommendations that are readily available to all appropriate people in your organization.
Authors and Contributors

Matthias Patzak is a Principal Advisor at AWS. Before joining AWS, Matthias was Vice President of IT at AutoScout24 and Chief Digital Officer at Home Shopping Europe. In both companies, he introduced lean-agile operational models at scale and lead successful cloud transformations resulting in shorter delivery times and increased business value.

Paul Vassu is a Manager Solutions Architecture at AWS, working with Retail Enterprises. Paul has 20 years of experience in Software Engineering and was Director of Software Engineering at adidas before joining AWS. There, he contributed to the modernization of the ecommerce landscape while driving adoption of agile and DevOps practices.

Many thanks to our team of contributors at AWS:

- Michael Needham, SA Readiness Specialist
- Till Hohenberger, Senior Solutions Architect
- Jordi Fernandez Moledo, SA Senior Specialist
- Frank Blessing, Solutions Architect
- David Dorf, Principal Retail GTMS
- Alejandro Mondragon, Principal BDM
- Bastien Leblanc, Senior Solutions Architect
Sources

1. https://public.tableau.com/profile/salesforcEcommercecloud#!/vizhome/SalesforceShoppingIndex/SalesforceShoppingIndex

2. Global Ecommerce Update 2021, Inside Intelligence, eMarketer, Jan 2021, Ethan Cramer-Flood