



x x x x x x x



MACH-X AND E-COMMERCE: WHY THE FUTURE IS FLEXIBLE?

.....





x x x x x x x



.....



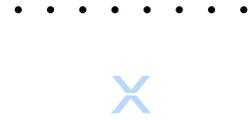
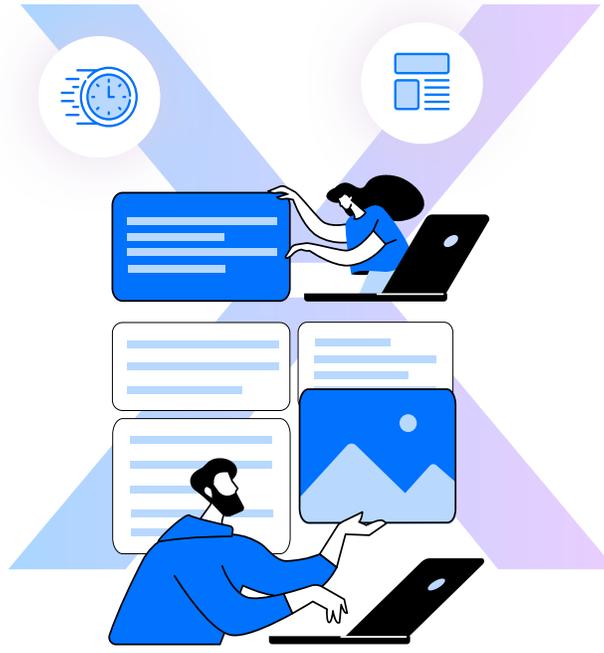
Introduction to MACH

In today's fast-paced and demanding world, digital commerce leaders need to rapidly launch capabilities and support quick geo specific customizations to cater to the evolving business landscape.

However, there are immense challenges involved in keeping [monolithic legacy systems](#) updated in this climate of rapid technological change, evolving buying trends and opportunities for innovation. Monolithic applications are not designed to extend to new touchpoints such as mobile apps, voice, IoT, phygital retail, AR/VR, and chatbots. Legacy code does not support Agile DevOps and CI/CD (continuous integration/continuous delivery), which makes it difficult to roll out required changes efficiently.

This is exactly why IT leaders are exploring MACH commerce solutions (microservices, API-first, cloud-native, and headless) to modernize their tech stacks and adapt quickly to market dynamics. Industry Analysts are increasingly recommending modular design when building new applications and updating legacy platforms.

Today, there are several as-a-service, MACH-first platforms on the market that an enterprise can use to modernize their technology and embrace "headless commerce." MACH is the ideal modular architecture, but what if brands had more flexibility? What is [MACH-X](#), and how can your IT team leverage it for your digital transformation journey?



The MACH-X advantage

What is MACH?

MACH is now an industry-recognized term that stands for Microservices, API-first, Cloud-native, and Headless.



Microservices

Unlike a monolith that contains a collection of tightly coupled services within a common code base, all needing to be implemented together, microservices are loosely coupled, independently deployable, and scalable applications. Microservices contain their own well-defined APIs, data store, and event channels. In the context of e-commerce, each microservice is designed around a single function or business capability such as catalog, payments, pricing, cart and checkout, or address verification.

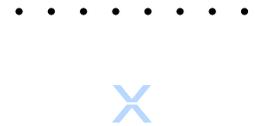
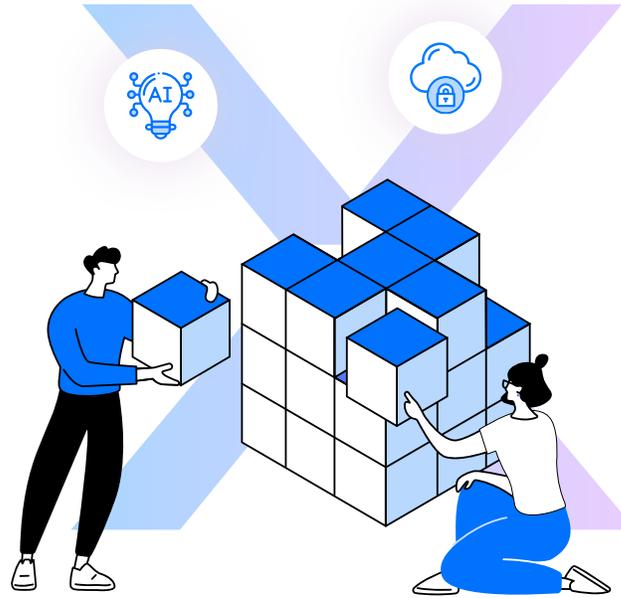
Through an API orchestration layer, microservices can “talk” to each other and other applications. In this layer, business logic is configured through service orchestration, rather than embedded in microservices code. This enables rapid development of one or more microservices independent of the others, or all the microservices together, while reducing technical debt.



API-first

API-first means the reversal of the traditional approach to application design. Rather than build backend code first and slap an API on top of it, APIs are the core around which services are built. Designing, testing, and perfecting a consistent, reusable and developer-friendly API comes first, while code comes second.

This API first approach differentiates “headless” platforms built with MACH principles ground up from monoliths that added APIs or decoupled services as an afterthought. The API-first approach accelerates the development process and enables addition of newer capabilities (e.g. Marketplaces, app store etc.) and new channels of engagement (IOT, Chat, Voice, Augmented Reality etc.) as they become market ready.



Cloud-native

An on-premise monolithic application must scale up and down as a whole, even if the load is only on a small piece of it. This can make hosting costly and inefficient.

Deploying an application in the cloud provides the elasticity to scale up or down with demand and protect uptime, especially during peak traffic like during the black Friday weekend or during flash sales.

Cloud-native microservices scale independently keeping costs in line. Because they are independently deployed, a security attack on one component will not take down the entire system. Hence, most commercial microservices are delivered as-a-service in the (vendor's) cloud.

Cloud-agnostic microservices can be deployed in any public cloud, including the popular AWS, Azure, or Google Cloud, or on your own cloud in a custom

deployment. This provides the flexibility to procure one or many microservices from a vendor and deploy them with your existing stack, rather than maintain multiple environments. It also ensures you can move your stack seamlessly should your cloud hosting preferences change. It is important to note that cloud agnosticism is not a core tenet of MACH architecture, and is not available from every vendor.

Headless

Headless means an application ships without a front end, enabling it to extend to any front end or touchpoint, including ones without a user interface. For example, headless commerce can support PWAs and SPAs (progressive web applications and single page applications), bring digital capabilities into physical retail, connect to chatbots and voice assistants, or even integrate with 'smart' appliances for IOT enabled ordering.

However, not all headless applications are modular. First generation headless commerce platforms simply decouple the presentation layer from the backend to enable a CMS or DXP (Content Management System, Digital Experience Platform) to drive the front-end experience through an API. Such non-modular platforms still carry the disadvantages of monolithic architecture (slow delivery, technical debt, and rigidity), and struggle to extend to new and innovative touchpoints and experienced, a baggage that a platform that has been designed as headless from the grounds up does not carry.

Adding the X factor to MACH

In today's market, there are several MACH-ready commerce vendors who can provide the benefits of flexibility, scalability, and technical agility through the API layer. This works as long as you use the solution "as is" and "as delivered" by the vendor.

Yet, many enterprises want greater flexibility, and the ability to address any "unknown variables" during the never-ending process of digital evolution. [MACH-X](#) supports this flexibility with an open-source stack that enables you to extend and override the capabilities of any microservice you may need to resolve.

MACH-X gives you a clear edge over MACH platforms by enabling you to:



Future-proof your development with an open stack



Plug in your own microservices



Streamline your operations

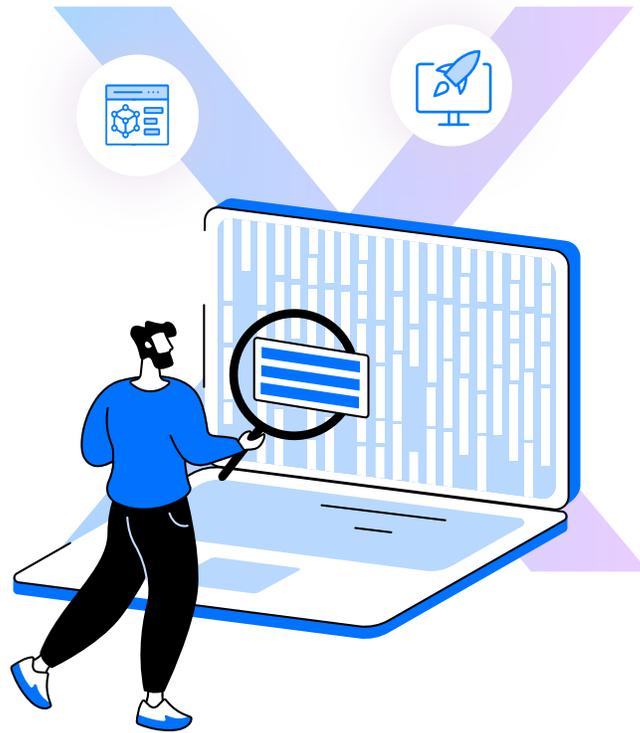


Balance agility and adaptability





x x x x x x x



.....



Future-proof your development with an open stack advantage

MACH-X's developer-friendly approach leverages best-of-breed and highly accessible technologies such as Java Spring Boot, Apache Camel, Mongo/MySQL, and Java extension frameworks. This makes "hacking the stack" accessible to developers without the learning curve of niche or proprietary programming languages.

In addition, MACH-X divides API layers in a way that keeps your customizations separate from the core. This means you can overwrite Infosys Equinox endpoints with your own or add endpoints to existing domains without veering off the upgrade path. This lowers complexity and protects the longevity of your development efforts.

MACH-X is about more than just microservices. It is about providing a holistic environment -- including future-proof infrastructure. Many SaaS providers require you to bring your own middleware

to support your microservices. The ready-to-use Infosys Equinox framework is built with disposable architecture. It can be used as-is, or you can replace any component with your preferred technologies without disrupting your application. For example, you could replace Kubernetes with Amazon ECS for container management or use Apache Kafka over RabbitMQ for messaging.



Plug in your own microservices

Software-as-a-service is a popular delivery model for MACH platforms. It boasts the elastic benefits of the cloud, with releases pushed seamlessly from the vendor to a customer's instance. Multi-tenant SaaS microservices enable customization in the orchestration layer to build unique journeys and experiences.

SaaS microservices suites enable you to work with only those services that you need. Since they are loosely coupled, you can take or leave individual microservices and the system will still work. Also, you can bring your own or third party microservices to the mix via the orchestration layer. However, this adds to the complexity and overhead of your architecture, as you cannot deploy your own services into your SaaS vendor's environment. You need to deploy them separately to make it all work. In the orchestration layer, mix-and-match does not always work optimally. For example, suppose you use your own inventory system. Your vendor's microservices cannot tell if a product is in stock or not unless you are using their native inventory service. In addition, your business team may need to work with different admin interfaces to perform daily activities.

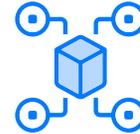
This is where MACH-X technology provides a huge advantage over other SaaS providers. MACH-X enables you to build your own custom microservices using the same open stack as the core or bring third-party microservices. You can deploy everything together including your middleware/orchestration layer and front ends in any cloud or environment resulting in a unified deployment.



Streamline your operations

Unified deployment enables you to streamline monitoring and operations, with end-to-end visibility into how all your microservices are performing across your solution. Many SaaS vendors cannot connect with microservices outside of their own offering requiring you to maintain siloed dashboards.

MACH-X also enables you to chain custom functionality for admin and business tooling. For example, you may want to enable customer service reps to manually issue loyalty points or perform appeasements when resolving customer complaints. You can achieve this in the orchestration layer, even if the loyalty microservice comes from a different vendor than the microservices for account and orders. Similarly, you can ensure that the loyalty microservice can properly reverse points for a partial transaction refund or similar account adjustment.



Balance agility and adaptability

When a business adopts microservices, business users may once again find themselves heavily dependent on IT for administration and content changes they used to manage themselves through the legacy system's native business tooling or CMS. This can result in back-office productivity slow down.

One problem with MACH solutions is the lack of mature business tooling. Until recently, only the most digitally mature and technically savvy organizations could tackle a microservices journey (often building their own tools with in-house developer teams), while commercial vendors focused on the needs of IT. But this is changing, and vendors are investing heavily in admin capabilities.

However, out-of-the-box tooling is often not enough for business teams especially when unique requirements have driven the decision towards microservices in the first place. Backend users include administrators, field sales reps and store associates as well as live help agents and their tooling needs extend beyond the web to include mobile apps and other touchpoints.

The solution lies in building microservice admin interfaces on top of an open-source templating framework. This enables the functionality to be customized and orchestrated for any business tooling use case.

Three ways to migrate to MACH-X based on your business need

Unlike traditional rip-and-replace e-commerce [re-platforming projects](#), migration to MACH-X architecture can be based on your own schedule. Your legacy systems may have highly customized, critical capabilities that you may not be ready to lose in exchange for faster delivery of new features. MACH-X gives you the best of both worlds -- you can modernize the most critical aspects without disrupting core systems.





Big bang re-platforming

If you have outgrown your legacy platform or it has reached end-of-life, you may have an urgent need for a full re-platform regardless of your extensive customizations.

Since you need to completely replace your current platform, look for a full-suite microservices platform that covers all core commerce components such as catalog, pricing, promotions, payments, inventory, and cart and checkout as well as capabilities that serve the entire commerce lifecycle – hyper-personalization and marketing optimization, for example.

In addition to commerce services, you will need middleware to orchestrate your APIs and configure your business logic. A vendor that provides a ready-to-use framework including an API gateway, orchestration layer, service discovery, container management, messaging, monitoring, and DevSecOps can help you launch years faster than it would take to build your own deployment infrastructure.



Using the Strangler Pattern

An alternative to a big bang is the incremental approach taken by leading e-commerce microservices adopters. This method has been dubbed the Strangler Pattern by Martin Fowler, who likened the process to the Australian strangler figs that attach themselves to the upper branches of a host tree and slowly and systematically eat away at the tree from the top down, until the host is fully replaced by a new structure of vines.

Although this process is slower than a big bang for a full overhaul, it enables an organization to quickly take advantage of the capabilities of modular applications without duplicated maintenance between old and new systems. It also allows the business to get the most out of its legacy investment without causing disruption to operations and customers, while getting the benefits of the newer capabilities being implemented through the microservices

MACH-X is ideal for the strangler pattern migration because it combines ready-to-use microservices with the ability to refactor existing unique and mission-critical components to rapidly provide new capabilities. Custom-built microservices can share the same stack and documentation as core microservices, eliminating the need to reinvent the wheel while simplifying development and maintenance.

When independent teams build microservices with their own programming languages and frameworks, there can be inconsistencies across the application. If developers leave the organization, it is not always a seamless transition for a new developer to take over. Standardizing on MACH-X supports a future-ready application, one that can be easily enhanced and maintained by your team or vendor.



Deploying satellite services

Another scenario is where you have no immediate plans to fully replace your legacy monolith but want to launch greenfield innovation quickly, or simply supplement your capabilities with new microservices without adding new code to your backend.

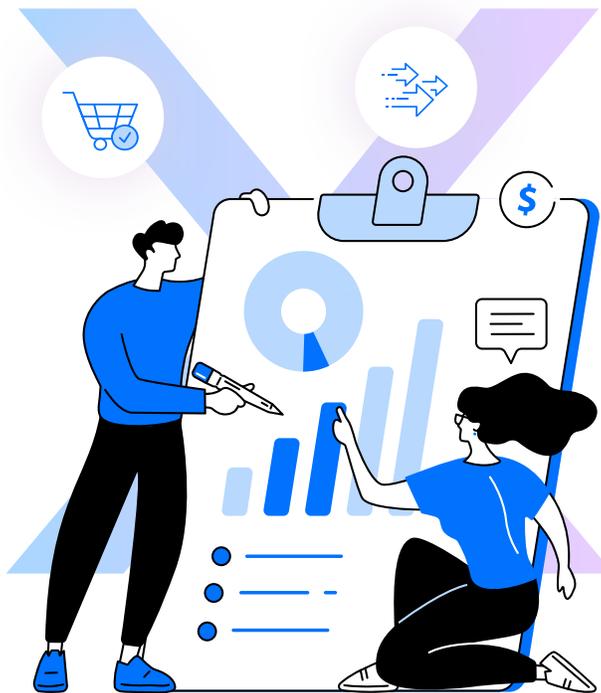
Individual a-la-carte microservices and prebuilt experiences can serve as point-solutions around your core platform. For example, a voice-enabled chatbot experience can hook into your legacy system through APIs. If the legacy system is replaced or a new front-end is added, the same point solution can be unhooked and re-hooked easily.

Satellite solutions can be composed from individual APIs to deliver unique functionality. Often, several standard microservices are bundled with custom-built APIs to achieve this. MACH-X provides the flexibility to build with the same, open stack across the bundle, and the ability to deploy them together in the same environment.

Because satellite services have been designed with the properties of MACH, they can be independently scaled and updated without impacting any other services. These services can be deployed in weeks by a small team or even a single developer.



x x x x x x x



.....



MACH-X is the future of e-commerce

Key advantages of MACH-X over MACH

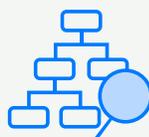
MACH-X leverages the flexibility and agility of MACH, enabling developers to build and extend microservices to satisfy unique requirements serving both customers and business users.



Accessible open-source Java technologies help lower development and maintenance costs



Extensible admin tooling facilitates agility for both IT and business



Layered design supports the extension and override of microservice code without falling off the upgrade path



Custom microservices can be built with the same stack and documentation as the core leveraging Infosys Equinox foundational services



The stack and framework can be used as a foundation to build and leverage microservices outside of digital commerce



Unified deployment simplifies orchestration and enables streamlined operations and monitoring within common dashboards



Cloud-agnostic microservices can be deployed in any cloud or environment, together with third-party microservices and applications



x x x x x x x



.....

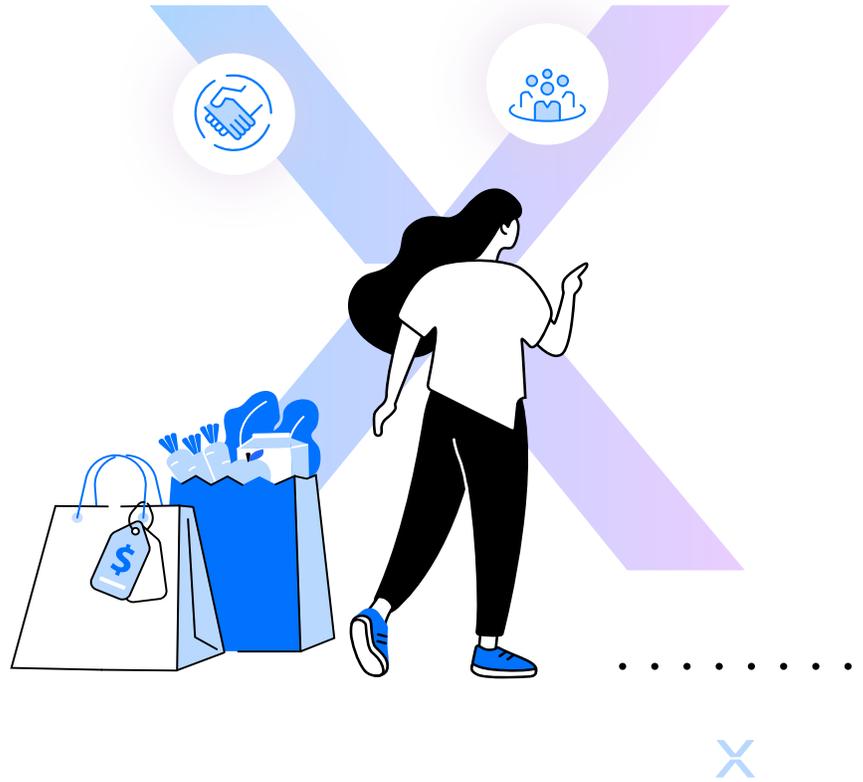




x x x x x x x

Looking to digitalize your e-commerce with the power and flexibility offered by MACH-X architecture? Infosys Equinox has got you covered.

Drop us a note at contactus@infosysequinox.com and together we can arrive at the best migration method for your unique requirement.



About Infosys Equinox

Infosys Equinox is a future-ready digital commerce and marketing platform built using Microservices-based, API-first, Cloud-native, Headless and eXtensible (MACH-X) architecture that enables businesses to drive human-centric experiences for their customers across touch points.

For more information and a product demo, please reach out to us at contactus@infosysequinox.com



For more information, contact askus@infosys.com



© 2023 Infosys Limited, Bengaluru, India. All Rights Reserved. Infosys believes the information in this document is accurate as of its publication date; such information is subject to change without notice. Infosys acknowledges the proprietary rights of other companies to the trademarks, product names and such other intellectual property rights mentioned in this document. Except as expressly permitted, neither this documentation nor any part of it may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, printing, photocopying, recording or otherwise, without the prior permission of Infosys Limited and/ or any named intellectual property rights holders under this document.