BetterCommerce

We Push Commerce Forward

Architecture Overview

V1.6

www.bettercommerce.io

letstalk@bettercommerce.io

CONTENTS

Executive Summary **4**

Architectural Principles **5**

API First **5**

Omni-channel **5**

Designed to Scale **5**

Managed Services **5**

Your Brand, Your Way **6**

Easy Integrations **6**

Designed for Evolution **6**

Technology Stack **7**

Core Technology **7**

Cloud Platform **7**

StoreFront **7**

Microsoft ASP.Net MVC & C# 7

Next.js Storefront 8

Architecture **9**

Cloud Deployment Architecture **9**

Tier Architecture **10**

Presentation Layer 10

Integration Layer - SDK (Software Development Kit) 10

REST API Layer 11

Services Layer 12

Data Persistence 12

Extensibility / Integration **13**

Data Feeds **13**

Business Applications **15**

Integrations **16**

Affiliate Integration **17**

Development / Staging **17**

Application Security **18**

Basics **18**

OWASP Top 10 ..... **18**

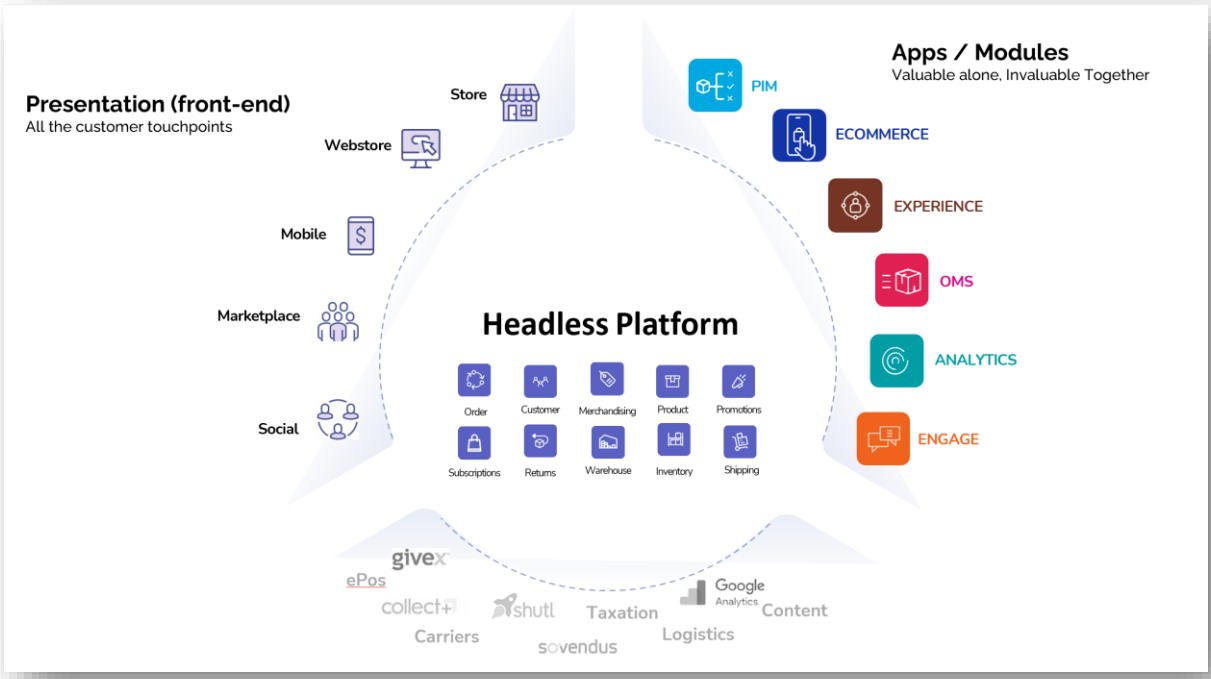
Conclusion **22**

EXECUTIVE SUMMARY

Enterprise computing has shifted from on-premise applications with hefty upfront capital outlay and delivery cycles of years to the era of Cloud Computing with SaaS offerings, faster time to market and flexible pricing options. The problem is, a lot of vendors still use their old on-premise applications and "reimagine" them for the cloud. In other words, they stick their old app on a server in a data centre, add some services, probably from a partner and provide a different pricing model. To expand their proposition, they acquire a company and bolt them together. It's a bit like building something using StickyBricks, Meccano, Knex and Duplo; they're all kids' building bricks, but don't really fit together that well.

BetterCommerce is the latest generation of eCommerce solution, designed for the era of the cloud enterprise, with an extensive range of APIs providing modern and flexible access, security & compliance, scale and consumption-based costs. A bit like Lego kits, BetterCommerce is designed and built as a suite of modules that included B2C & B2B Commerce, PIM, CMS, OMS, Customer360, Merchandising & Promotions, Customer Service for Instore & Call Centre, a Data and Analytics platform that's GDPR ready and all administered through a single business UI, Commerce Hub.

This document will discuss the design of the BetterCommerce suite of applications and how they have been architected and developed utilizing the Azure infrastructure and services at their core to provide an accessible, resilient and scalable platform in the cloud. It outlines the openness for UX design, integrations and ease to extend & customise without impacting future upgrades. The document concludes with security.



ARCHITECTURAL PRINCIPLES

Born in the cloud, BetterCommerce was designed to be extremely flexible, powerful and highly scalable built on the solid fundamental principles outlined below. The underlying objective behind the design & development of BetterCommerce has been to provide a scalable, resilient, flexible and easy to manage platform for businesses.

API FIRST

The technology landscape has only increased in complexity over the years making it increasingly difficult for businesses to adapt to the speed of the change. The encapsulation of the logic intertwined with models (data-access) and all written along-side the front-end code creates an extremely brittle architecture making it difficult to adapt quickly to change, which also binds a company into a specific technology stack.

An API First approach means that every functionality or feature is developed with a clear API definition document and becomes the guiding principle for any future development. One big advantage of this approach is that the responsibility for business logic is encapsulated within the API and does not needs to be duplicated in any of the front-end implementations.

OMNI-CHANNEL

The principle of omni-channel means providing a consistent content / user experience across all channels without unnecessarily duplicating efforts to maintain the same. In practice, the content is maintained in the *Commerce Hub* and exposed via the API to the web site, instore app, mobile app, mobile friendly website.

DESIGNED TO SCALE

Any application born in the cloud is designed to scale as & when needed without any interruptions / downtime to the service. The BetterCommerce platform is designed to scale horizontally, adding or removing new instances as required without restriction. Advanced planning of busy periods can be handled through auto-scaling which adjusts to increased website traffic as and when required.

MANAGED SERVICES

One of the core BetterCommerce principles is to capitalize on the benefits provided by Azure services and where possible complement the actual features and software development rather than just a managed infrastructure. In the same way, we want our customers to focus on evolving and growing the capabilities to support their business rather than just running their ecommerce platform.

Instead of running BetterCommerce applications on the Virtual Machine in a data-centre or in the cloud, the platform has been designed to run on PaaS (Platform as a Service) across the whole stack from web application to elastic search. The only part of the stack that runs on IaaS (VM) is the SQL database which is on the roadmap to move to PaaS in the future.

YOUR BRAND, YOUR WAY

Being API First, the platform's capabilities are 100% exposed via the API giving a high degree of freedom to developers to build a front end tailored to the channel using the same content from the API. Essentially, it gives 100% control to the business to adapt more quickly and with greater flexibility, thus ensuring that the end customer receives a consistent experience across all of the business touch points.

The business can use the in-built white labelled website template (HTML, CSS, JS) to control the user journey, experience, behaviour and look and feel. No specific structure needs to be followed, no special files to be configured or new syntax to be learnt for the front-end. Plain old, simple & powerful HTML, CSS, JS – all as you want it to be.

Come March 2022, and we shall be releasing our Next.JS based StoreFront as well capitalizing the modern way of development – Static Websites.

EASY INTEGRATIONS

External Integrations have been implemented in a simplistic way – Follow the interface definition, build an endpoint to replace any native feature's endpoint and that's it. The site can replace any native capability with a custom implementation. For example, if the business requires a price and availability check call that needs to be made to an external API rather than the native API, all that's needed is to change the end point in the front end.

DESIGNED FOR EVOLUTION

Change is the only constant especially in the software world where all applications evolve over time, whether to fix bugs, add new features, bring in new technologies, or make existing systems more scalable and resilient. Decoupling of the components is important in order to achieve an extensible architecture which can evolve over time without breaking the existing feature set.

TECHNOLOGY STACK

CORE TECHNOLOGY

BetterCommerce is built on industry standard Microsoft Asp.Net MVC (Model View Controller) using C# as the programming language. The Microsoft technology stack is one of the most trusted enterprise technologies suited for enterprise scale dynamic web applications.



CLOUD PLATFORM

Microsoft Azure

The BetterCommerce platform runs on the Microsoft Azure cloud taking away all of the infrastructure related challenges of setting up servers, operating systems, regular patching, etc. and helps businesses reduce their costs with flexible, on-demand scale for peak periods. Though we provide the storefronts in 2 different programming languages, the API First architecture makes the front-end development technology agnostic giving developers flexibility to use technology of their choice without compromising on scalability, extensibility & flexibility of the platform.

STOREFRONT

The platform provides the storefront in 2 different programming languages out of the box.

MICROSOFT ASP.NET MVC & C#

The white labelled front-end template provided with the platform is developed using Microsoft ASP.Net MVC & C# and packed with the modern & popular client-side technologies: HTML5 (W3C compliant), CSS3, Angular JS 1.7, jQuery. The default site template incorporates all the best practices of development – dependency injection, caching, session management, route management, global exception handling & logging.

Donut Caching: performance optimisation for any web application is crucial, particularly during the peak periods and caching is one of the most effective way to increase performance. Most web applications will typically bring data from one or different data sources for rendering the *output* on the browsers. Output caching essentially allows you to store the *output* of controller actions and return cached result on subsequent calls to the same controller action thus optimising the performance by reducing the number of calls made to the different data sources.

Donut caching is a variant of output caching where the entire page is cached except one or more parts. This is required in scenario where the home page is cached, however the user's login section in the top navigation bar needs to be specific for every visitor – which means it should be populated from the actual data sources, however the rest of the home page content (which is typically the same for all the visitors) can be easily cached and made faster for other visitors.

NEXT.JS STOREFRONT

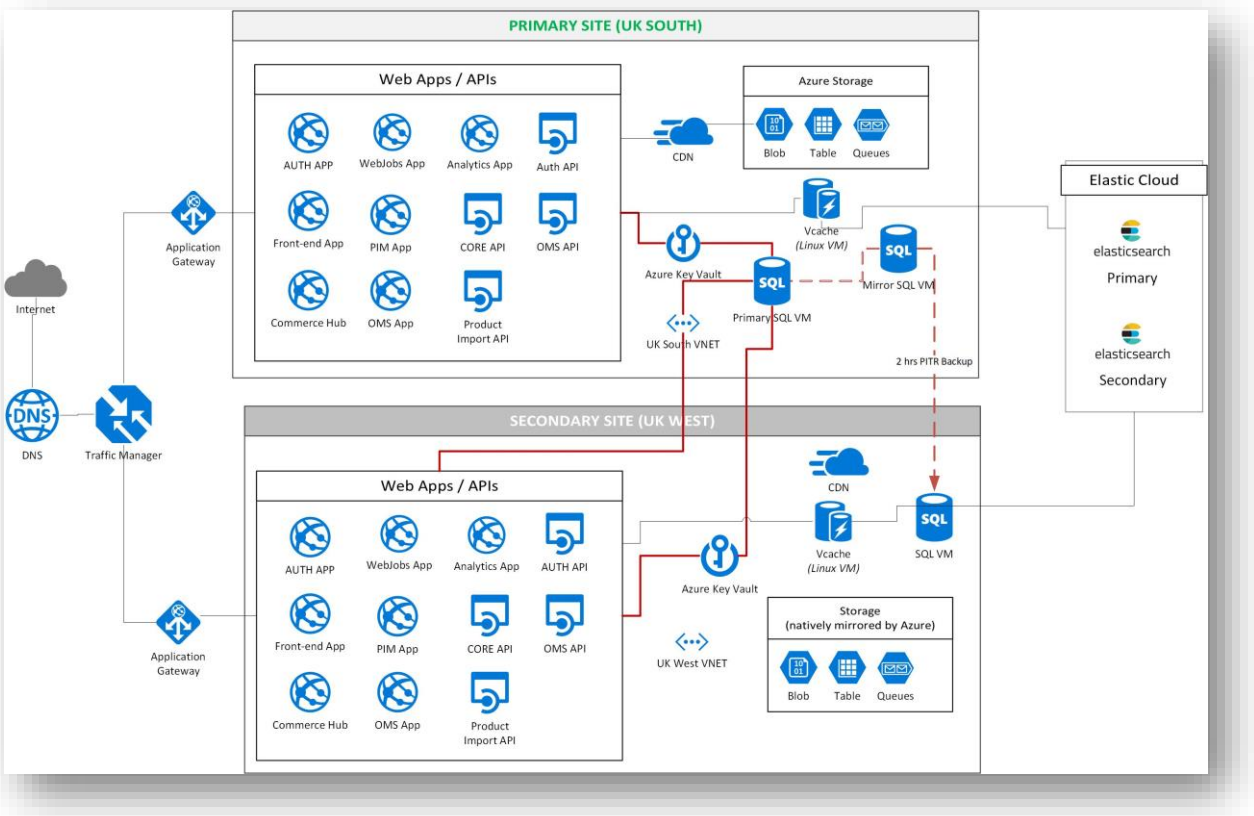
We have developed a storefront based on Next.JS, which is an open-source development framework built on top of Node.js enabling React based web applications functionalities such as server-side rendering and generating static websites. The storefront shall be made open-source hosted on github and is planned to be released by March 2022.

ARCHITECTURE

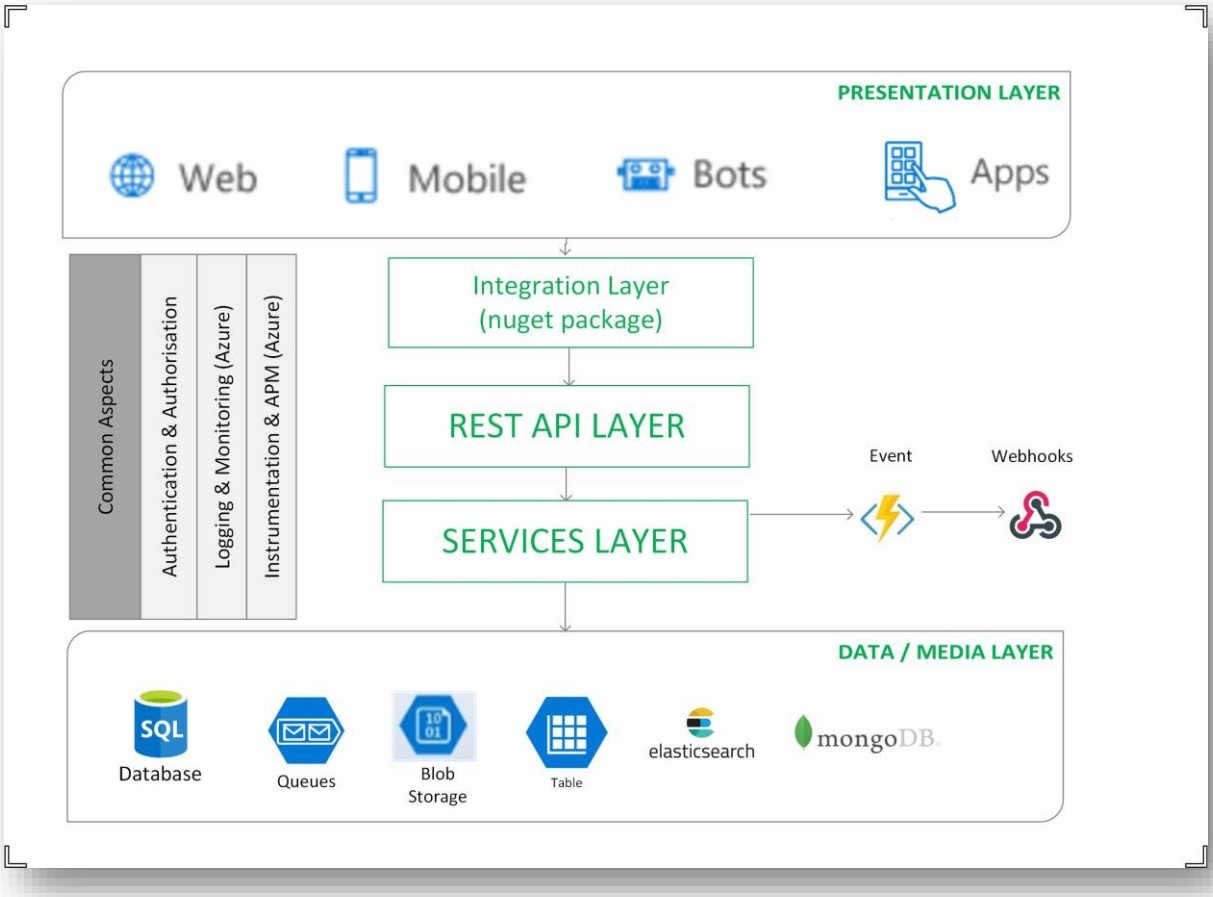
The different architecture diagrams are depicted below to give an understanding of how the platform has been designed, developed & deployed on the cloud.

CLOUD DEPLOYMENT ARCHITECTURE

The platform is designed & developed for the cloud with scalability, flexibility and high availability incorporated into every element of the stack. All the components used have been natively provided by Microsoft Azure as a PaaS offering and the whole idea of designing for PaaS was to focus on the core application features rather than maintaining & optimising the underlying infrastructure.



TIER ARCHITECTURE



PRESENTATION LAYER

In terms of presentation layer, the platform provides 2 variants of storefront out of the box as outlined above.

INTEGRATION LAYER - SDK (SOFTWARE DEVELOPMENT KIT)

The current integration layer is built & released as a Nuget Package which is basically a SDK encapsulating all the ground level tasks and API calls – token-based authentication, logging, REST calls, converting responses into simple objects. It enables the business to focus on customising the front end rather than spending effort on these plumbing but essential capabilities. As of now, only .Net based SDK is available to be downloaded from the *Nuget* package library. Going forward, SDKs for every programming language shall be published in the relevant package libraries as well.

REST API LAYER

The REST API layer is exposed to the business for building their presentation layer on top of it. The APIs are organized primarily in 4 groups:

1. Catalog – product, categories, brand related data.
2. Commerce – customer, basket, promotions, shipping services.
3. Content – localisation, email, FAQ, Views, Navigation services
4. Infrastructure – configuration, cache services.

Catalog Services	
Product Service	Product details, search products.
Brand Service	Brand details, search brands
Category Service	The category details including the products, brands & sub-brands
Collection Service	The collection list, collection items. A collection is basically a group of different entities - products, brands, categories, images, blogs.
Lookbook Service	Lookbooks in the system along with details

Commerce Services	
Customer Service	Authenticate, customer detail, create customer, Wishlist, manage address book, subscriptions, forgot / reset password.
Basket Service	Manage the basket – add, remove, update items, manage promotions, apply shipping methods, update basket.
Shipping Service	Provide shipping methods based on region, nominated deliveries and click n collect options based on the post code provided.
Order Service	Create an order, update status, add / update product qty, apply promotion
Return Service	Create & update a return request.

Content Services	
Site View Service	Site views (page content) based on the slug (url identifier) provided
Content Service	Mega menu navigation tree, FAQ, Redirection rules, product feeds and any master data setup.
Store Service	The Physical Stores defined in the Commerce Hub.
Microsite Service	Details for the microsites if any have been setup

Infrastructure Services	
Config Service	The site configuration grouped by different params.
Cache Service	Clear the cache explicitly

SERVICES LAYER

Domain services layer encapsulates and implements all the business logic required for order processing, managing product information, attributes, customers, taxation, etc. all the modules within the system – customer, order, basket, PIM (product information management) interact using the services-based architecture thus ensuring the principle of isolation of responsibility.

Services are used across different modules using dependency Injection which can be made configurable on-demand to interchange the native rules with some custom rules as long as interface definitions have been followed.

DATA PERSISTENCE

Data persistence in BetterCommerce is conducted across different storage technologies:

Microsoft SQL Server 2019	Industry standard enterprise scale Microsoft SQL Server 2019 in a cluster configuration providing high availability & scalability. All the structured & relational data is stored in MS SQL.

	Application databases have been split up to cater to specific functionalities of the platform and enable further scalability: 1. Order DB – stores the Order related information and supports the checkout 2. Master DB – primary database storing all the master information – products, attributes, customer data, etc. 3. Transient DB – stores the basket data which is transient in nature to isolate it from the rest of the system. 4. Audit DB – stores the complete audit trail along with extra logging information.
Azure Storage	Azure Blob storage is used for maintaining the images, PDFs, documents. The content can be uploaded using the in-built web-based file manager or by using the desktop-based utility provided by Microsoft.
Azure Queues	Azure queues are used for temporary persistence to support the asynchronous architecture and provide higher level of scalability to the platform. All the webhooks are pushed to the azure queues before being processed and fired to the external resources.
Elastic Search	Elastic Search is used for the fast searching of product and the main product indexing system.
MongoDB	Used for storing website event stream to provide AI driven product recommendations.

EXTENSIBILITY / INTEGRATION

DATA FEEDS

Import Feeds: The platform supports the following data imports out of the box. A default FTP location is defined for the feeds to be uploaded. These feeds can potentially be exported by merchant's ERP system or an external PIM or a mix of other systems.

Master Data	Once a day	Contains the master product data including the list price, sell price (incl / excl. tax), name, description, stock code, category ID, etc. This is a global data feed at the org level and no site identifier is needed in this feed.

Item Category	Once a day	Master list of item categories, including the sub categories. Basically, provides a complete category tree. This is a global data feed at the org level and no site identifier is needed in this feed.
Attribute Definition	Once a day	Master list of attributes & attribute sets. Attributes could vary from one category / product to another. This is a global data feed at the org level and no site identifier is needed in this feed.
Product Attribute Feed	Once a day	Product codes along with their attribute key / values pairs. This is a global data feed at the org level and no site identifier is needed in this feed.
Brand	Once a day	Master list of brands being sold by the merchant. This is a global data feed at the org level and no site identifier is needed in this feed.
Sub Brand	Once a day	Master list of sub-brands being sold by the merchant. This is a global data feed at the org level and no site identifier is needed in this feed.
Inventory	Every 10 minutes (configurable)	The platform maintains its own inventory data as the orders are placed and the counter is reset whenever the updated inventory feed is updated in the system. This is a site-specific feed; hence the site identifier is needed in the feed.
Pricelist	Once a day	The pricelist can be defined on a combination of region (site), currency code. i.e. for one website – multiple pricelist can be defined specific to currency. This is a site-specific feed; hence the site identifier is needed in the feed. In fact, if a product is in the pricelist for a specific site, then that product is made available in the website.
Other Channel Sales	On-demand	The platform also provides capability to import the sales data from other channels – for eg. Physical store

	sales. These are then collated & reflected in customer 360 view as long as the customer identifier (or some points to link with customer record) is available in the feed.

In absence of any or all feeds available from existing systems, the platform provides functionality to create the data manually either via bulk upload tools or simply creating them one by one in the Commerce Hub.

BUSINESS APPLICATIONS

The platform is in-built with integration interfaces for standard business application integrations.

ERP	The inbound master data integration is done using the data feeds defined above. The outbound integration for order transactions, customer data, refund is easily configurable using the webhooks which push out the data on REST.
OMS	The inbound status updates sent by OMS are managed by the specific web-services. The outbound integration covers – stock availability checks across physical stores, checking real time order status (if provided by OMS), posting the orders once approved for processing. All the outbound integration is easily configurable using the webhooks.
CRM	The customer master data can be imported using a data feed in csv format. Specific inbound integrations for different CRMs can be developed as an *app* and integrated into the platform. The outbound integration covers – pushing the customer data update, order history, session history as the business demands.

INTEGRATIONS

The platform provides an extensible architecture to integrate with 3rd party apps as needed. Some of the integrations offered out of the box are listed below. For more exhaustive list, please visit: https://www.bettercommerce.io/integrations

Identity Management	Besides the built-in login provider, the platform provides login integration with social apps
Payment Gateway	The platform provides support for the following payment gateways.
Shipping Methods	The platform provides **Standard Shipping** and in addition provides integration with 3rd party providers as well
Email	
Geo Tools	

The above-mentioned apps are provided out of the box; however, all of these are ***easily replaceable*** by any other 3rd party provider.

AFFILIATE INTEGRATION

The majority of affiliate integrations are driven by including a JavaScript snippet in the front-end code. The platform provides “content snippet” functionality which enables the merchandisers to define the snippets in the Commerce Hub and select the entity for which the data is to be populated dynamically.

This enables merchandisers to on-board affiliates at a faster speed and without any dependency on the development team. The platform comes pre-integrated with google tag manager as well which enables the merchandisers to include 3rd party JavaScript code much more efficiently.

DEVELOPMENT / STAGING

During development lifecycle of the store, multiple people would be working on different tasks. There is a standard release process from designing, developing, integration testing, UAT to finally go-live. Having separate environment to be able to develop, test and make bug fixes is quite crucial to ensure a high-quality end product.

BetterCommerce provides following types of sandbox environments:

Developer	Provides a working environment to do the development and test locally. Includes a very small data set.
Staging	Environment where the releases are made and integration testing & UAT can be conducted – one after the other. Includes optimum size (not same as production) of data & compute capacity to be able to run the end-to-end test cycles. *This environment has been decommissioned in 2021 Jul, but we shall soon be re-commissioning it.*

APPLICATION SECURITY

Capitalizing on the built-in security features of Azure, the platform is continuously scanned for any new / known vulnerabilities to ensure that we deliver a secured platform for our customers.

BASICS

1. Standard Authentication & Authorisation mechanism extended with Google / Microsoft MFA (Multifactor Authentication)
2. SSL has been configured across the whole website, API, Commerce Hub.

OWASP TOP 10

OWASP (Open Web Application Security Project) Top 10 is a list of ten most critical Web Application Security Risks. The following section provides a brief about the risks and their preventive measures built in the platform.

Risk	Definition
Injection	Injection flaws, such as SQL, OS, XXE, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query.
Preventive Measures	*All Database commands are executed using parameterized interfaces such as stored procedures.* *White list input validation is implemented in front end and at the server side as well.*

Broken Authentication and session Management	Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities (temporarily or permanently).
Preventive Measures	*OWASP recommended guidelines for a single set of authentication and session management controls have been baked into the platform, for example – unique session id, session id never exposed in url, strong password policies, passwords / session ids are always passed over HTTPS and other best practices.*

Cross-site Scripting (XSS)	XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping or updates an existing web page with user supplied data using a browser API that can create JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
Preventive Measures	*Auto Sanitisation library like Microsoft AntiXSS has been baked into the platform for all input form validations to escape untrusted data.*

Broken Access Control	Restrictions on what authenticated users are allowed to do are not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.
Preventive Measures	*Access controls are implemented for indirect object references along with additional checks to ensure that data is not accessed purely on the input received from the browser. Authorisation controls based on permissions are integral part of the solution.*

Security Misconfiguration	Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, platform, etc. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.
Preventive Measures	Utilising Azure PaaS environment takes away the risk of mis-configuration at the operating system, application server level. The application level configuration is strictly controlled and made only on need basis. Isolation of Development & Staging environments along with their configurations makes the process more efficient.

Sensitive Data Exposure	Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.

Preventive Measures	*The sensitive data elements are stored in an encrypted format. The SHA algorithm used is used to implement one-way encryption for the password. Any other sensitive data for customer can be configured to be stored in an encrypted format using CNG (Cryptography API: Next Generation) library. This is the latest cryptography engine with support for HSM devices as well, if needed.* *In addition, the whole site and Commerce Hub runs on SSL to ensure that no data is exposed over plain HTTP.* *The platform does not store any credit card information and stores the PSP specific tokens and that are also stored in a encrypted format.*

Insufficient Attack Protection	The majority of applications and APIs lack the basic ability to detect, prevent, and respond to both manual and automated attacks. Attack protection goes far beyond basic input validation and involves automatically detecting, logging, responding, and even blocking exploit attempts.
Preventive Measures	*Azure operations management portal enables us to configure automated alerts which can be triggered on detection of any unusual activity. All the events are logged in Application insights giving us the right level of information to act on any attack in progress and thus enabling us to review / patch the system, if a situation arises.*

Cross-site Request Forgery (CSRF)	A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application
Preventive Measures	*.Net (server side framework) and Angular JS (client side framework) used in platform, provide built-in support for CSRF defences. A unique token is generated with every HTTP request in a hidden form field and submitted when the form is submitted.*

Using components with known vulnerabilities	Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defences and enable various attacks and impacts.
Preventive Measures	*As part of ALM process, every release is scanned automatically for any security vulnerabilities across the whole platform including the dependencies. All the dependencies (frameworks, libraries, etc.) are installed using nuget package manager which provides a built-in mechanism to alert for any updates. All package updates are tested extensively in an isolated environment before being incorporated & released on the environment.*

Underprotected APIs	Modern applications often involve rich client applications and APIs, such as JavaScript in the browser and mobile apps, that connect to an API of some kind (SOAP/XML, REST/JSON, RPC, GWT, etc.). These APIs are often unprotected and contain numerous vulnerabilities.
Preventive Measures	*The APIs are available only over HTTPS and it follows a token based authentication. The tokens can be configured to expire after every 5 minutes (configurable at the key level) to maximum of 24 hours. The API keys, shared secret and tokens are stored in an encrypted format.*

CONCLUSION

Nothing reimagined, it’s all built for the cloud!

