PRODUCT DESCRIPTION

2022-01

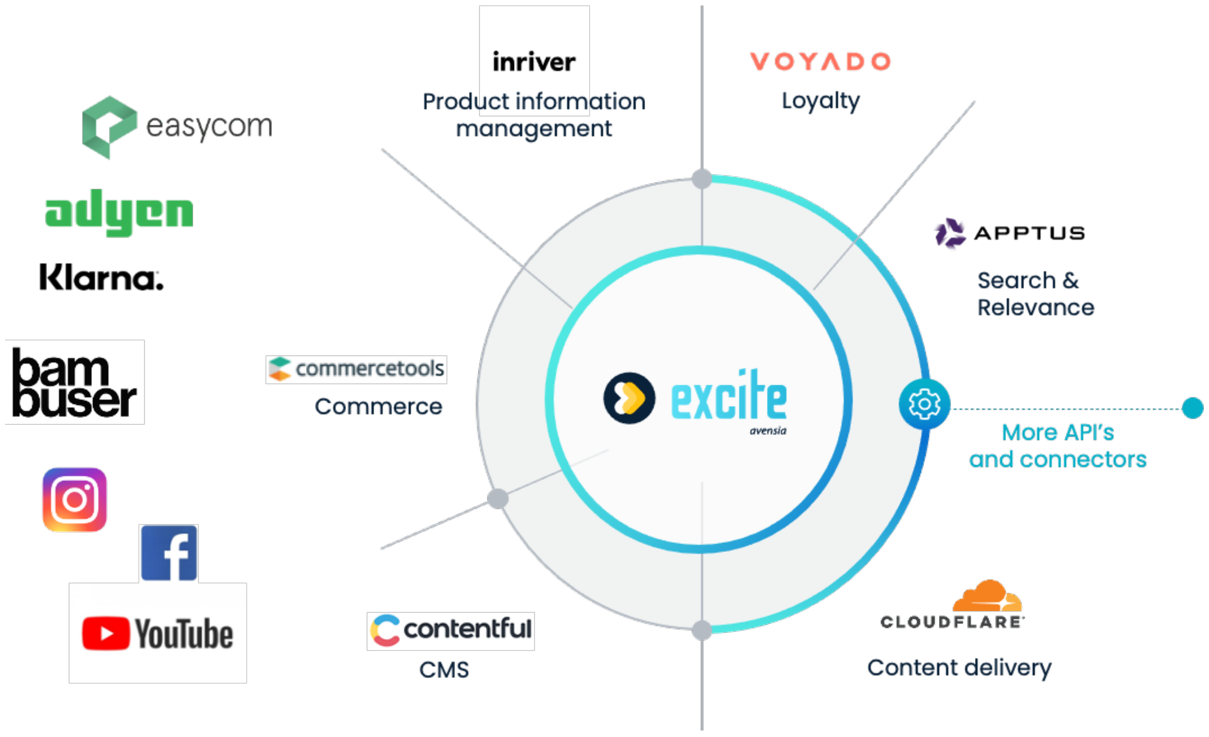
Avensia Excite accelerates the start-up of composable commerce projects, enhances our customers’ business value and the consumers’ experience.

Contents

1 Avensia Excite Solution Framework

Avensia Excite gives you the following key benefits:

- **Solves the complex puzzle of making MACH (Microservices based, API-first, Cloud-native SaaS and Headless) products fit together** to create a high-performance site for modern commerce.
- **Faster time to market** by offering packaged code for basic e-commerce functionality.
- **Smooth customer experience** built on the latest UI technology.
- Full ownership of the infrastructure allows Avensia Excite to **optimize performance to handle high loads and the extreme spikes** that modern medium to large merchants encounter.
- **Battle tested** by market leading merchants, built with the experience from Avensia Nitro that has more than 5 years serving some of the most demanding e-commerce sites in the Nordics.
- **Lower cost of ownership** through shared packages that are continuously tested and updated.
- **Connects to a rich ecosystem of third-party products** for content handling, product information management, search, payment and shipping providers, customer relations and marketing.



- **A solution for both B2C and B2B**. The product ships with features for a fully functional ecommerce site, with many capabilities for supporting commerce both to consumers and to businesses such as customer specific assortment and trade agreement prices. The solution can be customized utilizing existing functionality for **D2C and B2G**.

2 Performance

Owning the hosting, infrastructure and site implementation allows a higher degree of performance optimization in Avensia Excite than conventional solutions, from how routing, caching and media is handled to how we work with the search index, to how data is sent to the client and how the client is architected, to how data is loaded from data sources and how we interact with the underlying base systems.

2.1 SPA

The client application is a Single Page Application. It uses the latest browser technologies to ensure the application is resilient against slow and spotty networks. By using server rendering and progressively lazy loaded components, it makes sure to prioritize the resources that are needed for fast page loads. It will then intelligently load resources as they are needed. This makes the web application behave like a native application then a traditional web page with new page loads for each page request.

2.2 Scaling out with asynchronous processes

The solution is divided into 3 different applications, Web App, Content Api and Integration Hub, to allow efficient scaling and process management. Client requests of HTML or JSON data are separated from heavy background processes to enable the solution to scale out to meet the load requirements.

We make heavy use of asynchronous processes to ensure availability and scalability. Instead of sending a product to the search index directly when the product is changed, that change is placed in a queue which a separate background job processes and can therefore process multiple changes at the same time.

2.3 Pre-calculating and pre-processing content

Instead of calculating and processing content on the fly as the user request it, we achieve a higher throughput and faster response times by pre-processing and caching the content. Instead of doing price calculations on the fly we send the list prices to the search index and use event-based triggers to send new prices to the index as they change. For use cases where it’s important to have real-time data we send one version to the index that is good enough to show during the time we asynchronously load the real-time data. The important principle is to not compromise the user experience by making the consumer wait for such operations but still make it clear that something is loading.

2.4 Optimizing the network with Cloudflare CDN

To achieve the best possible network transport, the solution makes heavy use of Cloudflare as CDN. Images are cached with a far future date to ensure that images can be served directly from the CDN. but still makes it possible to change an image and have that propagate over the systems directly.

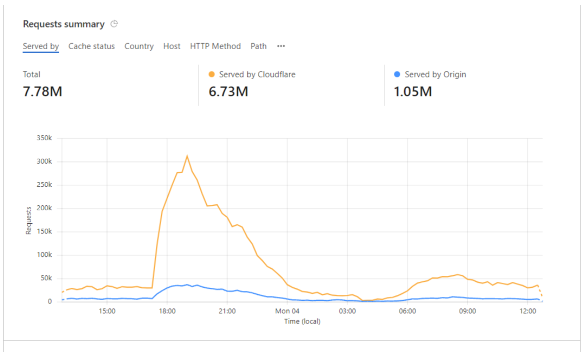
The CDN ensures that the most optimal version of an image is sent to the client. Newer browsers support modern image formats such as WebP, and the CDN sends the image in the best possible format.

The CDN uses HTTP2 when communicating with browsers that supports it, and through that the solution uses HTTP2 Push to push critical assets together with the initial response.

2.5 Cloudflare Edge nodes workers

Utilizing edge node workers in the Cloudflare CDN, Avensia Excite can optimize routing based on client and request type and as such achieve a higher degree of spike management than normal, allowing more of the client request to be served directly from the CDN nodes and as such offload the application layer.

Using Cloudflare edge node workers, Avensia Excite implements advanced caching where the cache is kept up to date by communicating with the origin servers in the background. The graph shows an example of ratio between data served directly by Cloudflare and origin webserver during a sudden traffic spike, e.g. marketing activity.



2.6 Routing

Routing is a core part of the Avensia Excite solution. By default, it brings together the different headless providers more neutral approaches into a more opinionated and optimized hierarchical solution, whilst still allowing the ability to extend and customise it as needed. Avensia Excite combines slugs from dynamic content such as Contentful and commercetools into full URLs in the background, updating resources as they change and caching them resulting in a more performant routing system whilst also offering breadcrumbs and redirection of historical URLs out of the box.

3 Software packages

The Avensia Excite solution is composed of multiple services. In its simplest form it is five different services that each can be deployed and scaled individually.

Services hosted in Azure are deployed to Azure VM Scale Sets by default and runs on Linux. By using Scale Sets, Azure will automatically scale out the number of instances needed for each service. Each service is scaled individually.

The solution is divided into 3 applications and comes with a set of core compiled software packages, that are continuously updated and maintained. New packages are added to provide additional functionality to the solution and it’s possible to install these packages later in the project if needed.

By using packages, the code is more isolated and therefore easier to test and upgrade. Every solution that has implemented a package can choose to upgrade at their own pace as updates are available.

3.1 Starter Site

Avensia Excite has a pre-built starter site which makes it possible to conduct efficient discovery workshops with a fit/gap approach showing all the out-of-the box features and then build your project on this site.

The starter site is delivered as source code that can be customized to optimize customers' business value. New releases of the starter site give you the opportunity to copy best practices to the implementation of your customer specific e-commerce site.

Our starter package consists of pre-built integrations APIs, starter site, and user stories with test criteria's. The starter site is not a demo site, it is a true project accelerator.

3.1.1 Frontdoor

The Frontdoor is a service that runs on Cloudflare's CDN edge servers to reduce the traffic that will reach the Webapp and Content API.

The Frontdoor manages caching of HTML and JSON responses based on the origins cache-control headers. But instead of just caching based on the URL other factors are included in the cache key such as language, country and type of device. What's included in the cache key can be even more granular than that as the Frontdoor service has access to cookies etc. on the request.

In the case of Cloudflare the Frontdoor service implements Stale While Revalidate (https://web.dev/stale-while-revalidate/) and Stale If Error which means that the cache is kept up to date by sending requests to the origin servers in the background. This improves latency for the requests that comes in after a cached entry has expired and allows you to have fast expirations (30 seconds or less) on content and still have a high cache hit ratio.

3.1.2 Webapp

The Webapp service is a Node.js service handling server-side rendering of the frontend app, which is using React and written in TypeScript. The webapp responds to all requests that expects HTML as the response type, which is typically the first page load of a session, or if the user refreshes the page.

The webapp also utilizes intelligent caching strategies, which allows for e.g. the product page to display the data that existed in the smaller product card, on a product listing page, while it fetches the complete product data in the background. This makes the navigation from product card to product page instantaneous.

The only external service that the Webapp talks directly to is the Content API.

3.1.3 Content API

The Content API is an ASP.NET Core service and its primary purpose is to serve the webapp and other touchpoints with content data in JSON format.

The Content API responds both to API calls such as adding a product to the cart as well as routing for content such as CMS pages, category pages, product pages etc.

The building of routing is processed in an event driven manner in the backend/integration hub service described in section 2.6 Routing.

The pre-processed route data determines which external system should be queries such as Contentful or Apptus eSales, transforms that data into a JSON structure and sends it out to the client.

The Content API serves the Webapp with content but is built to be able to serve other touch points as well such as a mobile app or an in-store display. Customer authentication is done using JWT tokens.

3.1.4 Integration hub

The integration hub (also called "backend") is a service that handles integrations with external systems such as commercetools, Contentful, ERP, WMS, CRM, Apptus eSales, etc.

This service also performs as much data processing as possible to minimize the amount of work that the Webapp and Content API needs to perform. Doing this in the backend service offloads work from the other services which means that they are easier to scale and have better response times.

The integration hub listens to events in the underlying systems, such as webhooks from Contentful and Azure Service Bus messages from commercetools. Such events are often placed in internal queues to be picked up by processing jobs. Processing jobs can either be scheduled using a cron schedule or scheduled to run when new entries are placed in a queue.

Internal queues are needed, instead of always processing the event directly when the external system calls Avensia Excite, because some computation needs synchronization. An entity in Contentful might be changed at the same time as an entity in commercetools and if the computation is performed in the event listeners, the same work will be performed twice if those entries are dependent on each other.

Depending on the needs of the Content API the backend will perform content processing, which is a core concept in Avensia Excite. Both Contentful and commercetools are represented as content providers and the system has one or more content processing jobs which take data from the providers, processes it and sends it to another system such as Apptus eSales. The content processing infrastructure automatically handles dependencies between entities in different providers. If a category in commercetools has a dependency on an entity in Contentful the content processing for the commercetools category will start whenever the entity in Contentful changes. The infrastructure will also start incremental processing by loading all changed entities and their dependencies in as few API calls as possible to make content processing fast.

3.1.5 commercetools Merchant Center App

The Merchant Center in commercetools can be extended with custom apps. Avensia Excite contains an extendable app for displaying status of running background jobs in the integration hub, the status of internal queues. It also includes an extension for order management (OMS) with features currently missing in the Merchant Center, to enable customer to make changes in post-sales support.

The Merchant Center from commercetools will never contain everything you need. commercetools instead focus on expanding their API and have building blocks to extend the backoffice functionality. Avensia Excite helps you in doing so by making it easier to build such apps.

A Merchant Center app is a React.js based webapp that can be hosted on any static file server and by default Azure Static website in an Azure Storage account is used. The app can talk directly to the commercetools API or any of the Avensia Excite APIs.

3.1.6 Adding more services

As it's expected that more services might be needed, depending on the requirements in a project, Avensia Excite has the process of building and deploying it already automated, making adding a new service very simple.

3.2 Payment Connectors

Avensia Excite contains a set of fully featured connectors against common payment providers such as Klarna and Adyen (see Functional Specification for full list). When a new payment provider is implemented in an Avensia Excite project, that is also packaged and made available to our customers.

3.3 Checkout Framework

Our experience is that the checkout flow is the most complex part of an e-commerce solution and it’s a challenge to make it robust. The possibility to combine different payment and delivery options, gift cards, promotion codes, service fees and discounts create a plethora of scenarios that must be covered. If we do not get this right, we risk losing not only conversion but also customer trust.

To cope with this, we have created a shared package that abstracts checkout complexity. It enables the development team to focus on customer experience rather than technical edge cases.

The checkout package is agnostic in presentation so it’s possible to implement any design or checkout flow.

Another example is getting a fast and consistent experience when the user rapidly clicks Add to cart or quickly switches between payment or shipping methods.

3.4 Typescript Type Generator – Ensures Quality

This package takes a model defined in C# and generates a TypeScript definition for it. This means that any JSON sent out from the API is expressed with a type-safe contract and that contract is automatically updated when a C# model is updated.

Having this in place gives confidence in making solution wide changes since the type checker gives compile time errors if the client code does not follow the generated contract.

After we introduced this in our solutions, we have seen that a whole category of bugs we were used to seeing in all projects simply disappeared.

3.5 URL Redirector – Accelerates SEO

Keeping track of redirects is crucial to not losing business from users coming to the website from old URLs. This package monitors renaming of content and keeps track of old URLs and ensures that we redirect the user to the new URL.

It also enables you to add custom rules that resembles regular expressions. You can set up a rule to redirect “/en-us/*” to “/en/$0” which would redirect a request from “/en-us/about” to “/en/about”. It also contains the same powerful matching for query parameters and makes it possible to pass through query parameters when we redirect.

3.6 Search & Relevance

A fundamental part of any site is to serve relevant content to the visitor and that the conversion of the usage is directly related to the performance of the site. In Avensia Excite almost all content (category listings, brand pages, campaign pages, blog articles, editorial content etc.) is served by a powerful search engine.

3.6.1 Apptus eSales

As part of the standard implementation, Avensia Excite offers **Apptus eSales** as the internal search and relevance engine.

In Apptus eSales you get a market leading relevance engine that not only returns the search or list result with high performance, but also sorts the result to prioritize the most relevant result for the current visitor. Behavior by the current and other visitors are collected and used to determine whet products are relevant at what time and context.

3.6.2 Filter facets

The search engine enables the user to filter the product listing on product attributes and with the Apptus eSales implementation, the relevance of the filters and filter options are automatically displayed and sorted to minimize the need for manual administration while constantly ensuring optimal conversion.

3.6.3 Search

All editorial content such as blog articles, how-to articles, stores etc. and the product catalogue such as categories, product and variants are indexed by the search engine and can be retrieved by the visitor by normal search, auto-complete and supports fuzzy search, synonyms and search suggestions.

3.6.4 Recommendations

Apptus eSales also adds recommendation panels to Avensia Excite. This could be products lists added to the product detail page or the cart or checkout and can be configured in Apptus eSales management app to show e.g. people who view item A also purchased B..E, top-sellers for one or more categories etc. The recommendation panels also use the same automatically optimized relevance algorithm as the search results to ensure optimal conversion.

3.7 PIM Connector

Avensia Excite has a connector between commercetools and inRiver PIM with an event-based architecture which ensures that we only send the data needed to make data flow from inRiver to commercetools as fast as possible.

Our experience has taught us that you don't want a one-to-one mapping of the model in the PIM to the model in commercetools. You want to exclude some fields that aren't relevant to commercetools and ensure that changes to those fields doesn't trigger data to be sent. It might make sense to express the product hierarchy as two levels in PIM but three levels in commercetools.

This experience made us design the connectors between these systems in a way that makes it possible to re-map the model in a layer between these systems.

4 Technology stack

The framework is built on battle tested components from companies such as Microsoft and Facebook. The value proposition of Avensia Excite is that we've taken these components and glued them together so that they work as one coherent solution.

Microsoft .NET Core / C# / ASP.NET Core. The industry standard web framework from Microsoft. These are the tools that the Avensia Excite backend (with business logic) and all integration components are built with.

React. Developed and maintained by Meta, React is the world's most popular JavaScript library for building user interfaces. Not only does React improve performance with its virtual DOM, but it also allows us to structure the frontend into components with a clear interfaces and responsibility, making the solution more maintainable.

Node.js Node is a widely popular JavaScript runtime built on Chrome's V8 JavaScript engine designed with real-time, push-based architectures in mind.

TypeScript. TypeScript is central to the way we handle data models in Avensia Excite: all data that is returned from the server is a typed model in .NET, and corresponding models are generated automatically for the frontend to use. This streamlines the work by making the data model available through intellisense in the development environment. If a data contract is broken, a compile time error will be issued.

Azure Service Bus is a fully managed enterprise PaaS message broker used to decouple applications, load balance work, and provide high reliability of communication. It offers queues for point-to-point communication and storing messages durably in triple-redundant storage, as well as detecting duplicate messages and allowing consumers to pull messages when they need. It also provided topics to enable pub/sub communication to multiple subscribers.

PostgreSQL is a powerful, open-source object-relational database system with over 30 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance. PostgreSQL is used as the primary data store or data warehouse for many web, mobile, geospatial, and analytics applications.

Azure Storage. The Azure Storage platform is Microsoft's cloud storage solution for modern data storage scenarios. Core storage services offer a massively scalable object store for data objects, disk storage for Azure virtual machines (VMs), a file system service for the cloud, a messaging store for reliable messaging, and a NoSQL store.

Webpack. Webpack bundles JavaScript, CSS and assets into smaller files which allows the loading of browser resources to be as efficient as possible.

NuGet / npm. These are package managers (for backend and frontend respectively) that are used to enrich the functionality by reusing existing components/implementations. Avensia Excite uses a mix of publicly available packages (like commercetools) and several internal packages.

4.1 Single Page Application

The website(s) built with Avensia Excite becomes a Single Page Application, where all internal site navigation is done by loading only the content needed for that click. Since this is driven from the client application it enables us to efficiently use cached data that we have loaded before.

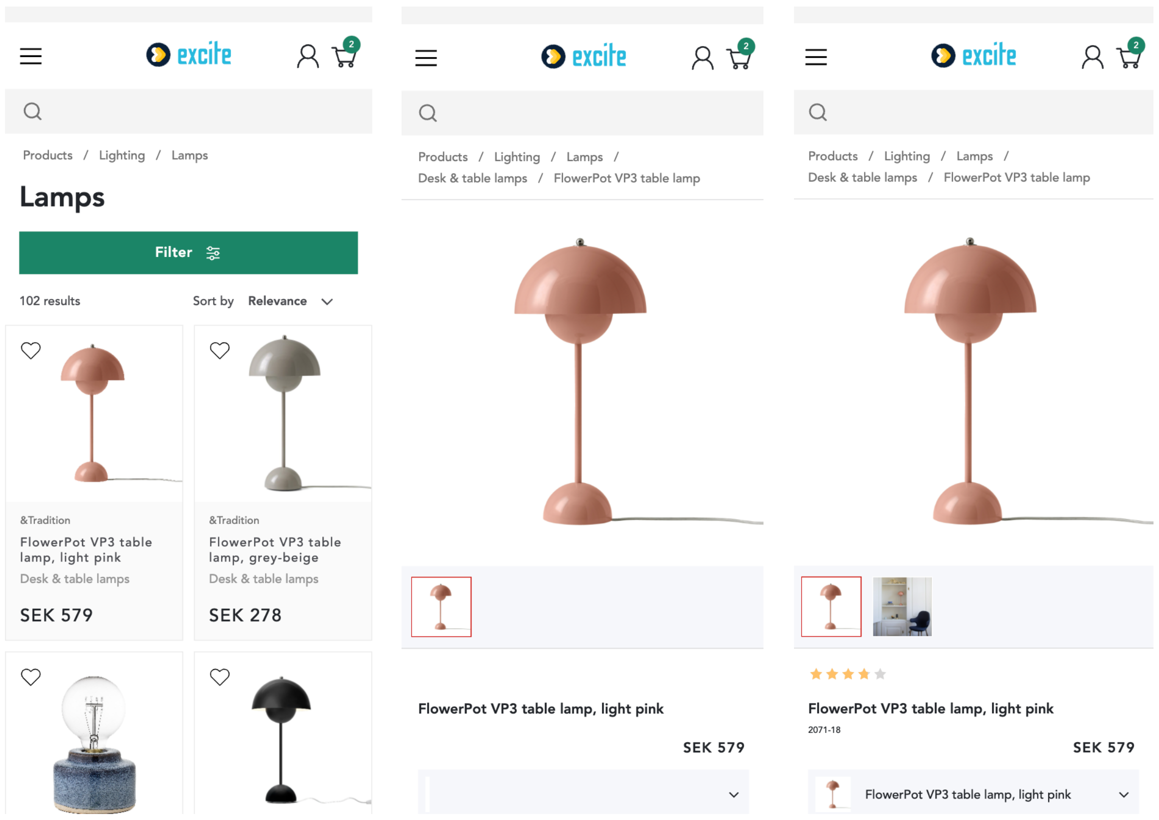
From an operational excellence perspective, the most interesting aspect of having a website as a Single Page Application is that it becomes much easier to scale and to further enhance the UI interactions and UI complexity. Since the client becomes a real application, we can use industry standard patterns and practices for managing complexity.

4.2 API first

All data that is visible on the website – such as texts and product data – are available through a JSON API that the client application uses. This means that any data presented on any page can be utilized by any client and not just the website.

4.3 Caching

A typical example of smart client caching is going back and forth between pages such as a product listing page and a product detail page. When navigating back from a detail page to a listing page, the data needed to render the listing page is fetched from local client storge, so that rendering can happen without having to load anything from the server. This makes the experience very fast regardless of network speed or the current load on the server. The images below show an example of going from a listing page to a detail page. The detail page is rendered immediately from the data already existing on the listing page and, in parallel background requests, the full details are fetched from the server. The consumer isn't hindered by this and can quickly and safely navigate back even if the full details aren't loaded yet.


excite
Products / Lighting / Lamps
Lamps
Filter
102 results
Sort by Relevance
&Tradition
FlowerPot VP3 table lamp, light pink
Desk & table lamps
SEK 579
&Tradition
FlowerPot VP3 table lamp, grey-beige
Desk & table lamps
SEK 278
excite
Products / Lighting / Lamps / Desk & table lamps / FlowerPot VP3 table lamp
FlowerPot VP3 table lamp, light pink
SEK 579
excite
Products / Lighting / Lamps / Desk & table lamps / FlowerPot VP3 table lamp
FlowerPot VP3 table lamp, light pink
2071-18
SEK 579
FlowerPot VP3 table lamp, light pink

5 Content

Avensia Excite contains a variety of features to create and display content on a site, ranging from plain article pages (with media and text) to specific landing pages and start page with a more banner- focused layout. Avensia Excite comes bundled with a whole range of different content-blocks that, when combined, provides great flexibility when creating pages and content. And thanks to the flexibility that the Contentful provides. Existing blocks can be customized, and new blocks can be added to fit any need.

Blocks are used to build up the structure of a page, and blocks are shared between pages. The Avensia Excite layout blocks automatically adapt the web page according to the users' devices - mobile devices or desktop.

6 Operations

Avensia hosts the solution in Microsoft Azure, but the solution could be hosted in any cloud env such as Amazon Web Services (AWS) or Google Cloud Platform. It utilizes a combination of Azure Web Apps, VM's and data storage to ensure availability and scalability.

6.1 Environments

The solution is usually deployed to three different environments. The **development environment** is the first environment and should be seen as a pure testing environment where all changes are continuously deployed.

The second environment is the **stage environment** which is used as a testing and verification environment for changes before they are deployed to production. This environment is both used but the development team and but the customer to ensure functionality and acceptance test the release.

The last environment is the **production environment** which is the env that manages the actual transactions from the web site visitors. This environment has auto-scaling set up to scale op and/or out as more traffic comes in, to ensure availability and performance.

External systems such as ERP, PIM, WMS, PSP, etc. need to have at least one test environment as well as the production environment.

6.2 Deployment

Deployment is automated which reduces the risk of manual errors and limits the deployment time. By using a blue-green deployment strategy an inactive deployment slot is published to whilst the site still runs and quick verification can be done on the deployment slot before going live. When going live we can use Cloudflares internal DNS to swap to the deployed slot to achieve zero-downtime deployment.

To achieve zero downtime deployment, the concept of blue/green deployment (https://en.wikipedia.org/wiki/Blue-green_deployment) is used. During deployment you have two almost identical environments and when you've verified that the new environment role is working you swap them. Once the swap is complete, the other environment role is taken down.

In some cases, deployment needs a service window to run migrations or other major system changes and in those cases a maintenance page is displayed for the end users until the deployment is done. The goal is always to have as few planned downtime deploys as possible.

6.3 Monitoring and logging

The solution is monitored using Azure Application Insights, Pingdom and Raygun.

Application Insight gives insights into application performance and utilization and provides triggers for auto-scaling new instances. It is possible to profile a live application through Application Insights to fine-tune the code for better performance, and to locate failure points to accelerate debugging, and problem solving.

Pingdom is used to monitor that the website is accessible and pings the site from different geographical regions to ensure that all parts of the world can access it. Alerts from Pingdom are sent both to the customers and Avensias incident team as well as the project development team.

Raygun is used to capture application errors. Such errors and all details about them are sent to Raygun which notifies the development team, and allows them to track, prioritize and resolve issues within releases.