IBM WEBSPHERE COMMERCE – UPGRADING TO V9

A review of the major changes in v9,
and the key considerations
for your upgrade project.

Salmon
SHAPING FUTURE COMMERCE

INTRODUCTION

WebSphere Commerce 9 is here – it's official. In fact, for those not close to the IBM roadmap, this was almost a surprise, given that there was a 6-year gap between version 7 (GA in 2009) and version 8 (GA in 2015). Clearly, for the meantime at least, IBM has accelerated its major release cadence.

Perhaps even more importantly, version 7 (which the majority of customers are running on) has been announced as end of life, meaning it goes out of support April 30th 2019. That said, extended support (separate contract) will be offered through till December 30th 2021.

Customers on v7 therefore need to think about a migration project through 2018, unless they're prepared to enter into extended support. V8 customers are likely to at least have another year or two before the next major release is announced, although this is entirely up to IBM's discretion.

In this article we will cover the major changes in v9, as well as the key considerations for your upgrade project. It shares Salmon's experience of how best to approach ecommerce implementations, to help make sure you get it right.

It is written by Glen Burson, Global CTO at "Salmon", an eCommerce consultancy that has successfully delivered over 100 client implementations over the past 15 years on the IBM WebSphere Commerce platform, and that operates its largest practice in Europe. Clients include Celesio, DFS, Halfords, GAME, Premier Farnell, Sainsbury's and Selfridges.

The views and opinions expressed in this article reflect those of the author and his team at Salmon based on this experience.

ABOUT THE AUTHOR

Glen Burson
Global CTO, Salmon Ltd.

Glen has responsibility for technical strategy across the Salmon Group. He has led the technical architecture team since 2008 with a leadership role in platform delivery, performance, security and best practice implementations across a customer base that comprises many of biggest brands around, including the world's largest online grocery re-platform for Sainsbury's and the world's biggest B2B platform for Premier Farnell.

"Salmon is one of IBM's longest serving commerce partners globally and has countless successful implementations of our WebSphere Commerce platform. We appreciate Salmon's ability to become a trusted partner in turning our clients' commerce vision into differentiated brand experiences and revenue execution."

Russell Scherwin
Director of Marketing, Watson Commerce, IBM

Highlights of v9 – why upgrade?

Given that major upgrades are a significant investment both in terms of time and money, it begs the question – what benefits do you get from an upgrade?

Here, I'll highlight the key advantages of v9 against the currently supported versions as reasons why you may want to accelerate a v9 migration.

It's worth pointing out right upfront; v9 is a largely technical release. Most of the functional business improvements are being delivered as cloud integrations, through products such as IBM Watson Commerce Insights and Watson Content Hub. Version 9 is mostly focused on lowering the cost of installation, running and upgrading your WC infrastructure. In addition, because of the new architecture, upgrades post v9 are likely to be significantly easier.

So, what's new in v9? Here are the key points:

Containerisation

With v9, IBM has taken a radically different approach which should make the cost of installing and upgrading (even major versions) significantly cheaper in the future, through the use of container technology. If you take away nothing else about v9, it's important to grasp this point, and how it affects the way you run and manage WC environments.

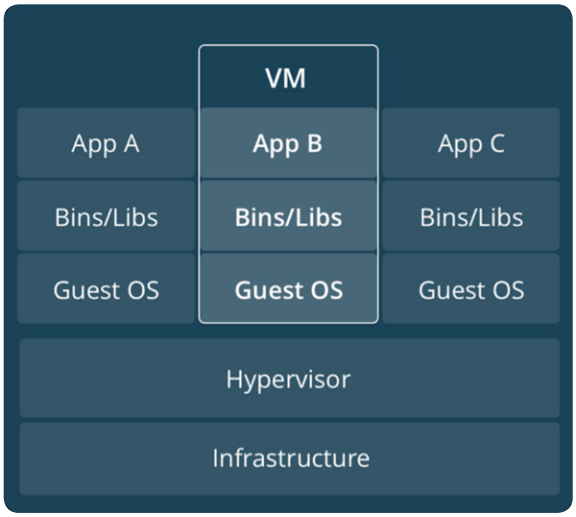
A brief overview of containerisation

Firstly, what is containerisation? Let me start with a simple analogy.

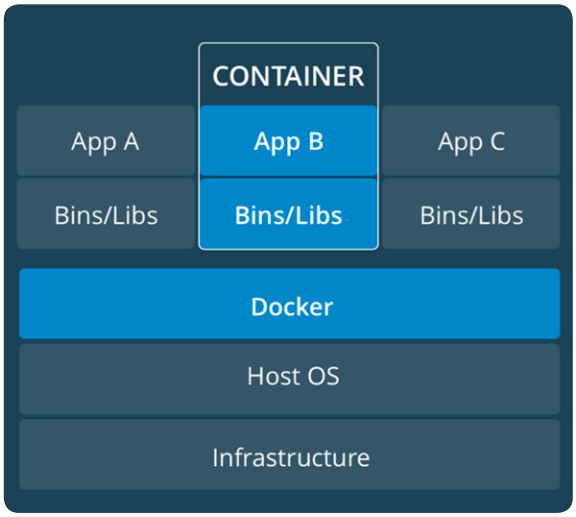
Imagine your WC application lives in a house, in fact multiple houses to support each component part of your application i.e. webserver, application server, search server, etc. Each house has everything; its own bathroom, kitchen, security (locks on the doors, alarm system), heating, plumbing, etc. This is great, but also wasteful.

Imagine instead, that each part of your application lived in an apartment, in an apartment block. Now your application can share resources; communal kitchen, shared plumbing and heating, security (although the apartment can still have its own door lock and additional security if required).

The point is, the apartments are a lot cheaper and efficient on resources that the houses, a concept that is exactly the same between a traditional virtualised hosting approach using Virtual Machines (VMs) vs containers.



Traditional virtualisation



Containerised alternative

In the previous illustration, you can see that in the traditional approach, not only do you have the overhead of the Hypervisor layer between your application and the hardware; you also have the overhead of running multiple operating systems (OS) which consume additional resources (remember the house) in terms of CPU, storage (for multiple copies of identical binaries) and CPU. Using containerisation, each container shares resources from a single operating system (OS), reducing the overheads and therefore hosting costs.

There are other benefits – starting a VM means booting an operating system (e.g. Linux, Windows etc) and then starting the application. This is slow, and doesn't play so nicely in an architecture (e.g. cloud) where starting additional capacity to meet new load might be common practice (e.g. auto-scaling). Containers however, only launch the application – this can be fast, even milliseconds, depending on the application of course.

Containers are also easy to distribute, update, roll out into your environments, and provide portability across different cloud providers.

How does WC v9 use containers?

IBM has opted not only for an (almost) fully containerised deployment strategy in v9, but has also adopted (and contributes to) one of the largest container platforms, Docker.

This is a compete paradigm shift in the way WC is delivered. Specifically:

- IBM no longer provides the software installation packages for WC. Instead, license holders download Docker Images which can be launched as containers. This takes a huge overhead of installing and configuring the WC software packages away from customers and partners completely. End users simply download the latest version as a pre-configured image.
- The maintenance strategy has changed dramatically. Software updates (fixpacks, feature packs, mod packs, interim fixes) which were difficult and time-consuming to deploy, no longer exist. Instead, IBM will provide version updates via new Docker images that can be regularly downloaded. This could vastly improve the uptake of new fix and functional updates, as the implementation effort falls. This forms part of the IBM stated development pipeline using CI/CD techniques.
- The trade-off (I hesitate to say "downside") is that WC integrators need to understand how to build and run WC environments using containers and an ecosystem that support high availability, storage, monitoring, centralised logs etc. This is a big shift in skills and culture compared to traditional hosting approaches.
- Customisations to the core product, must either be layered onto the IBM provided base images, or separated dedicated external customisation servers using the IBM xC programming model. More on the xC model later in this article.
- Developers need to be aware of the new deployment model; for example, containers must be stateless (unless they mount shared storage).

Finally, I should point out that the only component NOT provided by the new containerisation approach is the database (DB2). Instead, WC implementers must install the DB software using a traditional approach. This is because IBM do not recommend running DB2 in a container for a production workload. However, you can create your own Docker image for DB2 without much effort and use this to quickly deploy test environments.

WebSphere Liberty Server

Simply put, WebSphere Liberty is a lightweight, fast application server. It is based upon the open source Open Liberty project.

IBM has chosen WebSphere Liberty as a replacement for the WebSphere Application Server (WAS), meaning you can run less complex architectures (no need for federating application servers into a cluster), start your application servers faster (auto-scaling, especially with Docker containerisation) and generally run less infrastructure.

However – IBM has only provided support for Liberty for newer parts of the stack i.e. all but the transactional application servers, although we presume the longer-term strategy is to port this to Liberty as well, once the dependencies on WAS have been taken out of the core code.

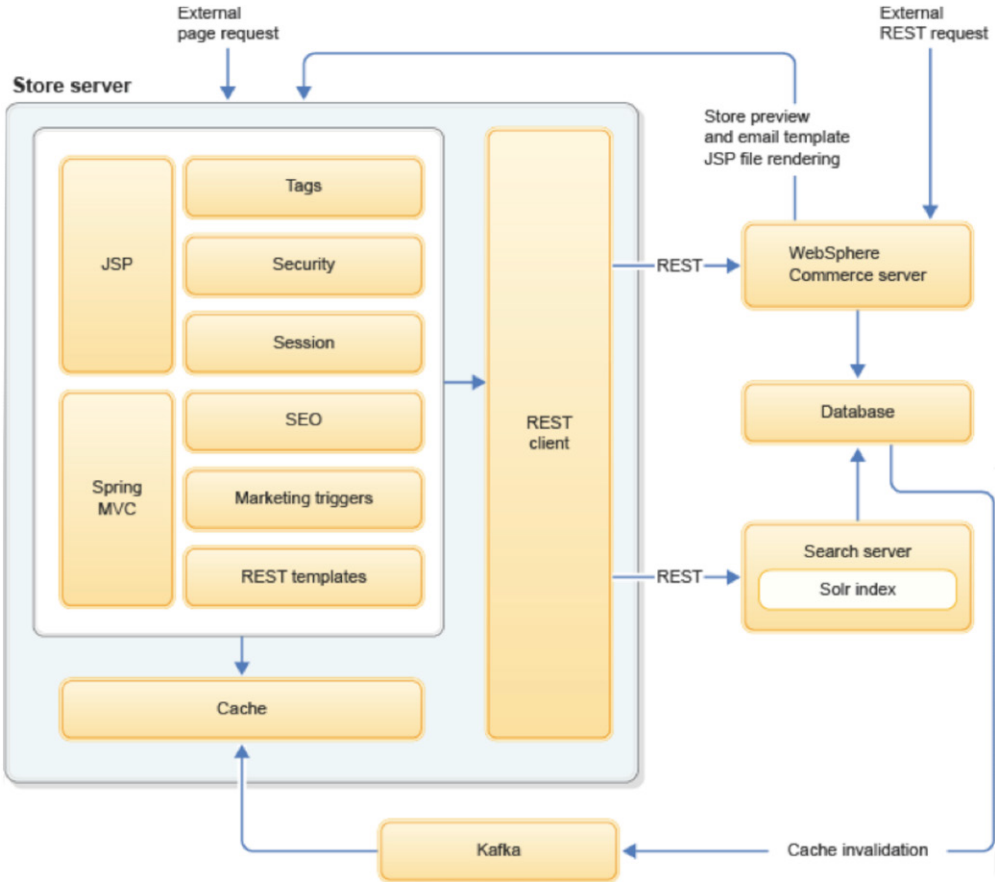
Here is how the application is being delivered:

Component	Application Server	Delivery Approach
Web Server	None – IHS (based on Apache)	Docker image
Search	WebSphere Liberty	Docker image
Store	WebSphere Liberty	Docker image
Transactional Server (tx)	WebSphere Application Server	Docker image
Customisation Server (xc)	WebSphere Liberty	Docker image
Utility Server	None	Docker image
Database	None	Software

Store separation

WebSphere v9 comes with a completely separated store front delivered in a separate Docker image.

This means the storefront component can be deployed, scaled and managed completely separately from the core transactional components (essentially a headless eCommerce architecture). Communication to the transactional server is via the standard WC REST services.



Store server architecture

This provides a number of advantages for developers, operations and the business itself in terms of:

- Ability to scale the number of servers independently
- Ability to host different store fronts separately and independently
- Provide more frequent (and independent) software releases to the store front, as opposed to the core transactional services
- Improved security (e.g. developers have no direct DB access from the store servers)

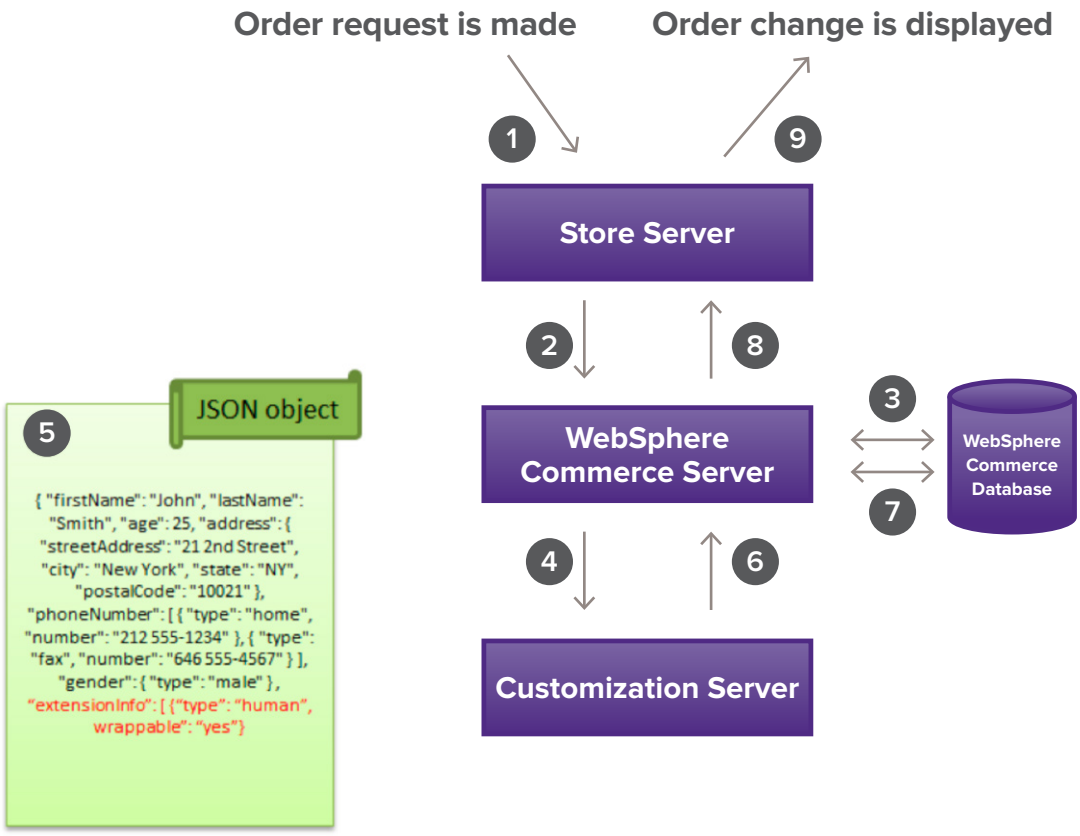
NOTE: *The documentation states that migrated applications (e.g. from v7/8 to v9) cannot take advantage of the separated store model architecture, and the store must be deployed (as it is today) as part of the transactional server. We assume that with additional work a migrated application* ***could*** *take advantage of the new model, but this is not for certain at the moment what this effort entails.*

Support for headless architectures

It goes without saying, that the ability to run WC v9 with a separated storefront paves the way for other "heads" to be integrated. These could be e.g. NodeJS apps, a commercial package such as Adobe AEM or anything else. It seems likely that IBM will deliver a Single Page App (SPA) storefront at some point in the near future, possibly through integration with Watson Content Hub. Most e-commerce vendors are providing sample assets for SPA based storefronts in their roadmap.

xC customisation

The xC (external customisation) extension points customisation model, is a new customisation approach that is designed to avoid customisation of the core product (transactional servers). If customisations can be achieved through the xC model, then you can further simplify your CI/CD pipeline by avoiding extensive re-testing of changes to the transactional server, because (guess what) you don't change it.



IBM overview of customisation server integration

The xC approach allows you to intercept requests to WC commands using a number of hooks, and perform additional processing on a (xC) separate application, potentially returning a modified response to the command being called. Three hooks are provided: pre-command execution, replacement command execution (do not execute the command at all), and post-command execution.

What we like about xC

- A "hook" based model is a widely understood and adopted software solution to avoid direct customisation. In fact, all major commerce vendors are adopting this approach in varying different ways.
- The v9 development environment provides a good framework for developing xC extensions and deploying them into a Liberty based application server all out of the box.
- It's great for very simple customisation scenarios.

Challenges for the xC model

Before jumping into the xC approach for your project, you need to be aware of some limitations.

- Only a limited number of commands are available for extension – although we expect this list to grow.
- The xC approach only supports a very limited set of customisation scenarios. Many other supported ways of extending WC (such as extensions to the WC data model) are not (or not yet) supported.
- IBM does not recommend mixing both the xC model and traditional customisation approaches – because it adds complexity for no gain.

> ***NOTE:*** *WebSphere Commerce Version 9 supports both the traditional customization model and xC customization model at the same time, but it is not recommended that you use the two customization models in a single WebSphere Commerce deployment. Trying to use both models increases the complexity in architecture and programming and defeats the purpose of xC separation.*
>
> *For more information about developing by using traditional customization model (local model), see* ***Customizing WebSphere Commerce.***

- Therefore, it's important to understand if all of the immediate and future customisations will fit inside the xC model; if not, then a traditional customisation approach would be recommended.
- Access to the WC DB from the xC server is not supported, meaning the extensions must operate within the bounds of its own storage.
- For WebSphere Commerce implementations on earlier versions, which are being migrated to v9, customisations will need to be re-written (if possible) into the new model if you wish to adopt this approach. There is no migration tool that will do this for you.

Licensing changes

We are told that PVU licensing is still supported for v9, although how this will be monitored e.g. via the ILMT tool for a containerised deployment (where all the containers share resource) is not yet clear.

With v9, IBM have included developer licenses in the base entitlement, meaning that customers no longer have to pay for individual named licenses for the development team. This is a significant change in direction and brings IBM in line with other major eCommerce vendors.

Be aware that the licensing for IBM Customer Service changed in v8, and is designated as any use of "forUser" and "forUserId" on command or API invocation. This means that any custom functionality developed on v7 that makes use of these parameters are licensable and deemed IBM Customer Service; even if it was developed before this license restriction came into play.

Other improvements from previous releases

If your current implementation is on late or latest v7, there is not a huge amount of functional change, but there will be other key differences you will need to take account of. We've not provided a full list here, as it really depends what v7 / v8 level you are coming from. But there are a couple of other areas to be aware of:

- Management Centre moved to DHTML with v8 – customisations need to be ported or migrated.
- CSR (IBM Customer Service) functionality was also introduced in v8, but licensed separately.

If you are looking to migrate an earlier customised version of WC v7, then you need to carry out a gap analysis against v9 and your current functionality, and develop a strategy around which customisations are migrated, and which are re-developed, dropped, or ported to new functionality delivered out of the box. More on this in the next section.

Other key v9 technical points of interest

Here is a summary of other key v9 changes to be aware of.

Item	Notes
Development Env	Is still Windows, based on Eclipse. We are desperate for a Linux based environment. Speaking to the IBM product team, this is definitely on the backlog – watch this space.
DB2	Mentioned earlier, this is not Dockerised. Can be done, but there are some special considerations.
Oracle	Not supported yet. Indication is that this is in the backlog, and will be delivered Q2 18 – but this is not a guarantee. If needed, please speak to your IBM rep.
DSL persistence	Data Service Layer – was provided in v7, has now been deprecated in v9 in favour of JPA. This is a great move.
OAGIS messaging	Again, provided in v7, has now been deprecated in favour of RESTful interfaces. And again, this is great news for devs.
RFQ	Request For Quote – deprecated in favour of integration with IBM Configure, Price, Quote (CPQ) tool
ATP inventory	Available to Promise – anyone who has worked in this complicated model will be pleased it's also been deprecated.
Returns processes	Deprecated in favour of, you guessed it, integration with IBM Order Management
massload	Is now discontinued – be aware of this if you have any custom loaders that you want to port to v9.
Derby DB	Sometimes used in development environments – is no longer supported
Managed Files	This business feature has been discontinued. This makes sense in a cloud environment, but may cause some pain for customers who use this functionality and now need an alternative.
Utilities	Some utilities have also been discontinued in favour of REST interfaces, in order to facilitate cloud ops. See here

Top 7 considerations before embarking on a v9 migration

Migrating from v7 or v8 to v9 is a significant undertaking, depending on your current version and business goals. Each project will be approached differently depending on what has been delivered historically, and what the business requirements are including budget and timescales.

Here are the key points you need to consider when scoping your migration project:

1. What is your starting position?

There are two major strategies taken for a major version upgrade.

Assuming your WC implementation has customisations (almost all implementations will fall into this category), you have to decide whether to simply migrate your customisations directly to v9, or re-write your customisations on the newer architecture.

Often the solution is a combination of the two, requiring some analysis to identify which developed features are still fit for purpose, and which have either become technical debt, been disused by the business, or superseded by out of the box functionality.

Here are some key points to consider:

- **If you are on the latest feature pack / fix pack version of v7,** or on v8, you are most likely to be in a position to migrate customisations directly to v9.
- However, you need to **be aware of architectural changes** and factor those into your effort. For example:
  - Changes to Management Centre (DHTML from v8 vs Flash in v7)
  - Changes to SOLR architecture between v7 feature packs
  - Changes to Commerce Composer Widget implementations between v7 feature packs.
- **If you are on earlier versions of v7,** you will need to complete a gap-fit analysis to understand if any developed features have now been provided by IBM. This is a good chance to get back to out of the box.
- The IBM **implementation of some features may have changed,** and you may need to migrate your customisations, data or both to take advantage of new functionality. For example, for newer versions of the storefront enable Commerce Composer functionality, the new attribute dictionary data model is required for best use of Management Centre tooling.
- As mentioned earlier – **JPA has replaced EJBs and DSL.** If your customisations include extensions to the data model, you will have an additional task to migrate your persistence layer customisations. IBM provide a tool for this purpose.
- **Consider your technical debt** – if you have areas of functionality suffering from persistent high levels of bugs because of an accrual of technical debt, this is your chance to rewrite them.
- **Think about your store-front strategy.** Storefronts developed on Aurora cannot be directly migrated if you wish to take advantage of the Store Separate feature discussed earlier in this article. In addition, the Aurora store changed substantially through v7 feature packs & v8 to become completely REST service driven, incorporate responsive designs etc. A v9 upgrade may be the catalyst to take a different approach (e.g. headless) considering all of your different channels.
- **Check for the use of deprecated and removed features.** Features such as returns functionality, ATP inventory and more have been deprecated. See here for a full list. If you use those features, now may be a time to look for an alternative strategy. In addition, a number of operating systems, out of the box integrations and utilities have been discontinued and can no longer be used. See here for a full list of discontinued features.

2. Rethink and future-proof your target architecture

A major version upgrade is a great time to check that your target architecture is future-proofed and supports the business needs. Often legacy implementations of an earlier version of WC require investment to take advantage of new development and deployment technology advances. The cost of making these changes as part of a migration project can sometimes be smaller when taken as part of a migration, associated testing, management etc.

Here are some key architecture considerations when moving to v9:

- **APIs.** Earlier versions of WC were weaker on APIs and custom integrations may have been developed. Late WC v7 and v8 have a broad set of APIs that will support many use cases and are much easier to implement than earlier versions. Reverting to out-of-the-box APIs should be considered. Think about an API first development approach.
- **Micro Services.** More commonly, businesses are scaling down functionality delivered through their eCommerce platform, rather than increasing it. This can be done gradually over time by introducing microservices alongside your eCommerce platform (see Martin Fowler's Strangler Pattern). A re-platform can be a good opportunity to revisit functionality and evaluate what functions could be delivered as a standalone service.
- **API Gateway.** To support an API first strategy, a major version upgrade and focus on APIs is an excellent chance to put in place an API gateway, that can add a layer in front of your application, manage payload transformations / orchestration, facilitate future changes in the architecture and deal with non-functional requirements (security, logging etc).
- **Headless.** As mentioned earlier, headless is a common architecture to separate the UI from business logic. Whilst WC fully supports headless, and indeed the Aurora storefront out-of-the-box can be deployed separately in headless fashion, migrated stores do not. Consider whether a headless architecture is right for the business and whether it should be incorporated into the migration effort or not.
- **SPA.** An alternative to Headless is a Single Page App (SPA) that works a bit more like a (JavaScript) application downloaded onto your browser, and uses the v9 APIs. It's generally faster (for the customer) and a lighter weight integration. I suspect this type of integration will be commonplace in the next 12 months, with all the major vendors planning a SPA enabled storefront.
- **Search.** If you are not using IBM Commerce Search (SOLR) – (customers who implemented base version 7 will be in this bracket), you really should be, and this should almost automatically be included in your scope. Many new IBM features rely on the deep IBM integration into SOLR for customer and tooling features.
- **CDN.** Implementing a CDN can save you license costs (if a PVU based license is in use) but also dramatically improve your end user experience. If you are revamping your UI as part of a migration project, removing personalised content in favour of AJAX driven content is definitely worth considering.

3. Rethink your development pipeline

I haven't ever worked with a website owner who wanted to make changes slower; at Salmon we are promoting development pipelines that can deliver incremental changes into production much more frequently than has historically been achieved with the major platform vendors.

Much of this is about culture between your organisation and any partners, but there is also a tech aspect to it as well. The change in architecture in v9 helps move towards a continuous integration or even continuous delivery pipeline, but you still need investment and expertise to deliver this.

A major version migration is an excellent opportunity to revisit your development processes and build in improved pipelines that will benefit the business.

4. What is your skillset and partner landscape?

It's clear that v9 provides a unique set of opportunities to improve eCommerce delivery, but also contains a unique set of challenges.

You must consider whether you have the capabilities in-house to deliver environments using the new Dockerised approach, but also if you or your partners have the skills and experience to capitalise on the opportunities around new architectures, better delivery approaches, and cloud hosting.

5. Hosting

WC v9 is architected for the cloud. If your current hosting is in-house or through a managed service, there could be significant reductions in hosting costs with an effective cloud deployment, if coupled with a high degree of automation.

This version of WC could deliver on the promise of:

- Quickly deploying new test environments based on need
- Auto-scaling product workloads
- Performance testing environments that can be created identically to production environments for tests and destroyed afterward to minimise costs

6. Licensing

Traditional customers will own PVU licenses. This model licenses the transactional server tier based on the number of CPU cores that are allocated to it.

Unfortunately, this model does not fit well either with cloud deployments, where the amount of deployed capacity could (if well architected) vary depending on load. Furthermore, although we understand IBM has committed to honour PVU licensing in v9, it is hard to see how IBM can apply a PVU based license in a containerised architecture*.

IBM does support an RVU model, where the license cost is based upon the number of order lines sold rather than the amount of deployed CPU cores.

As mentioned earlier in this article, customers using "forUser" or "forUserId" parameters on REST APIs or in calling commands, are subject to CSR seat based licensing from v8. If in doubt, speak to your IBM representative.

7. Taking advantage of IBM Cloud Offerings

Finally, as mentioned at the start of this article, most functional / business improvements are being incorporated into other cloud hosted products that come with pre-built integrations that may be of interest to your business.

If you are in the position of re-inventing some of your capabilities, especially around business tooling and content, it's worth looking into the features offered by products such as IBM Watson Commerce Insights and Watson Content Hub.

*We have asked IBM how PVUs will be measured when running in containers – it's not clear at present, but we will include an update here when we have an answer.

HOW SALMON CAN HELP

As we explore the integrations delivered with v9, we will post more information here in the future.

As consultative guide, partner and architect with over 100 client implementations on IBM WebSphere Commerce to our name, Salmon is already working with clients to help upgrade them to the new v9 platform. Let's discuss how we can help you.

Call: **+44 (0)20 3858 0061**

Email: **info@salmon.com**

Visit: **www.salmon.com**

MORE HELP

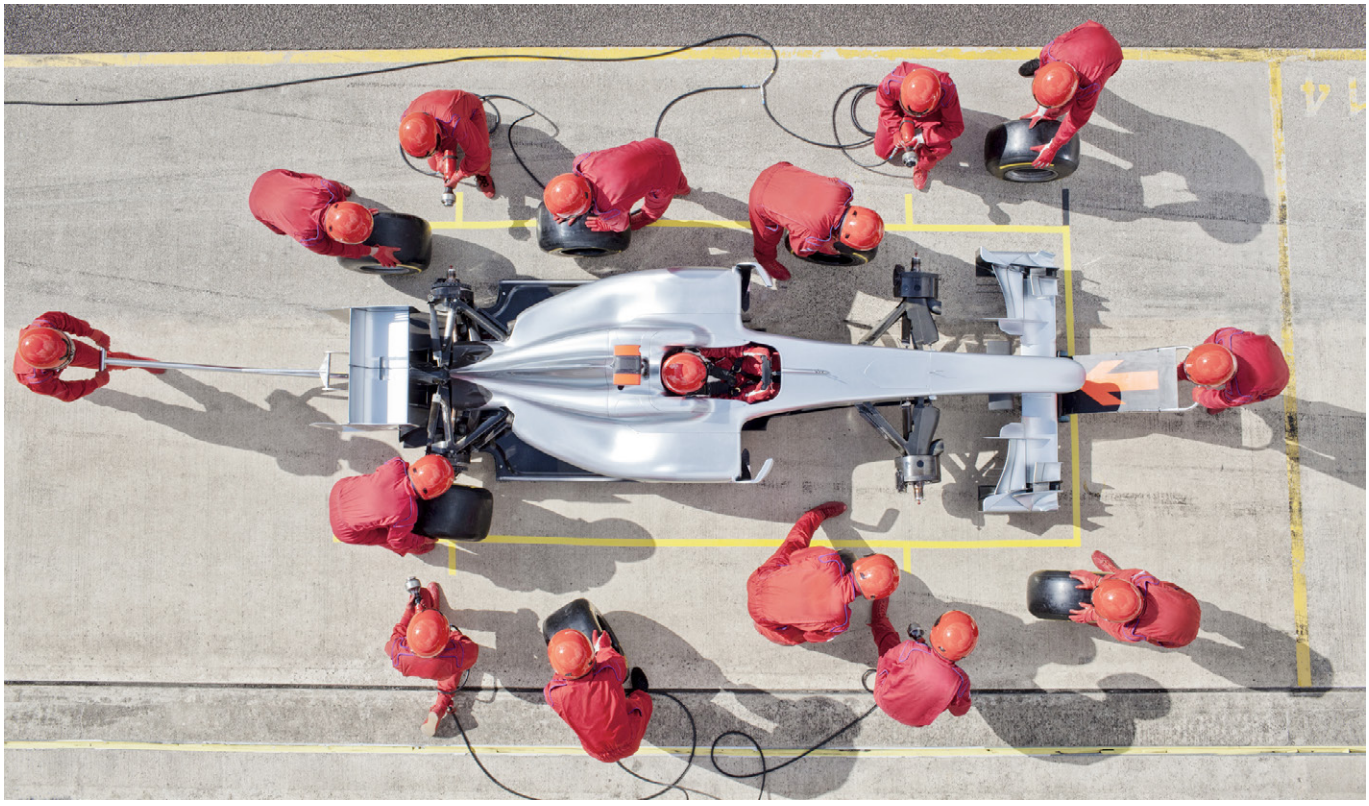
Here's some great content from IBM that's specific to WebSphere Commerce v9:

IBM WebSphere Commerce v9 Marketing Site

IBM WebSphere Commerce v9 Knowledge Center

You'll find Salmon's version 9 resources (including updates) here.

We are currently offering a **free, bespoke expert consultation** on migrating businesses to version 9. Details with sign-up can be found here.



Salmon
SHAPING FUTURE COMMERCE

ABOUT SALMON

Salmon is a global digital commerce consultancy – the biggest in WPP's network of companies – that defines and delivers market-changing solutions and customer journeys for the world's leading brands.

Established in 1989, with operations in London, Amsterdam, Sofia, New Delhi, Seattle, Beijing and Melbourne, Salmon clients include Argos, Asian Paints, Audi UK, DFS, Halfords, Jumbo, LloydsPharmacy, Premier Farnell, Sainsbury's, Selfridges and Sligro Food Group.

For more information,
call: **+44 (0)20 3858 0061**
email: **info@salmon.com**
visit: **www.salmon.com**