

# Commerce Cloud Digital Architecture Overview



Student Guide

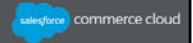


© Copyright 2017 salesforce.com, Inc.  
All rights reserved. Various trademarks held  
by their respective owners.

Welcome to *Commerce Cloud Digital Architecture Overview*.



## Copyright



© Copyright 2000-2017 salesforce.com, inc. All rights reserved. Various trademarks held by their respective owners.

Rights of ALBERT EINSTEIN are used with permission of The Hebrew University of Jerusalem. Represented exclusively by Greenlight.


This document contains proprietary information of salesforce.com, inc., it is provided under a license agreement containing restrictions on use, duplication and disclosure and is also protected by copyright law. Permission is granted to customers of salesforce.com, inc. to use and modify this document for their internal business purposes only. Resale of this document or its contents is prohibited.


The information in this document is subject to change without notice. Should you find any problems or errors, please log a case from the Support link on the Salesforce home page. Salesforce.com, inc. does not warrant that this document is error-free.



# Student Guide


## About the Course






**Audience:**

- Developers
- Administrators



**Duration:**  
45 minutes



**Course Materials:**  
Student Guide

**Course Prerequisites:**

*Commerce Cloud Digital Overview*

© Copyright 2017 salesforce.com, Inc. All rights reserved. Various trademarks held by their respective owners.

*Commerce Cloud Digital Architecture Overview* follows the introductory *Commerce Cloud Digital Overview* course and provides a deeper look at Digital and the process of developing a storefront. It is intended for developers and administrators involved in a Commerce Cloud Digital implementation project.



# Student Guide



## Course Objectives



When you have completed this course, you should be able to answer these questions:



01

What does it mean to work in a software-as-a-service (SaaS) environment?

02

How do I develop a storefront site?

03

How are Commerce Cloud Digital and SiteGenesis related?

04

How can I expand my storefront functionality?

© Copyright 2017 salesforce.com, Inc. All rights reserved. Various trademarks held by their respective owners.



# Student Guide

Course Contents		salesforce commerce cloud	
01	<ul style="list-style-type: none"><li>▪ On-Premise versus SaaS solutions</li><li>▪ Tenants on our cloud</li><li>▪ Traditional versus cloud-based development</li></ul> <p>What does it mean to work in a SaaS environment?</p>	02	<ul style="list-style-type: none"><li>▪ Project team members</li><li>▪ Storefront development process</li></ul> <p>How do I develop a storefront site?</p>
03	<ul style="list-style-type: none"><li>▪ Structure</li><li>▪ Tools</li><li>▪ Development model</li><li>▪ Updates</li></ul> <p>How are Commerce Cloud Digital and SiteGenesis related?</p>	04	<ul style="list-style-type: none"><li>▪ Integrations</li><li>▪ Cartridges</li><li>▪ Performance</li></ul> <p>How can I expand my storefront functionality?</p>

© Copyright 2017 salesforce.com, Inc. All rights reserved. Various trademarks held by their respective owners.

These topics are covered in this course.



What does it mean to work in a SaaS environment?



# Student Guide



01

What does it mean to work in a software-as-a-service (SaaS) environment?



When you have completed this module, you should be able to answer these questions:

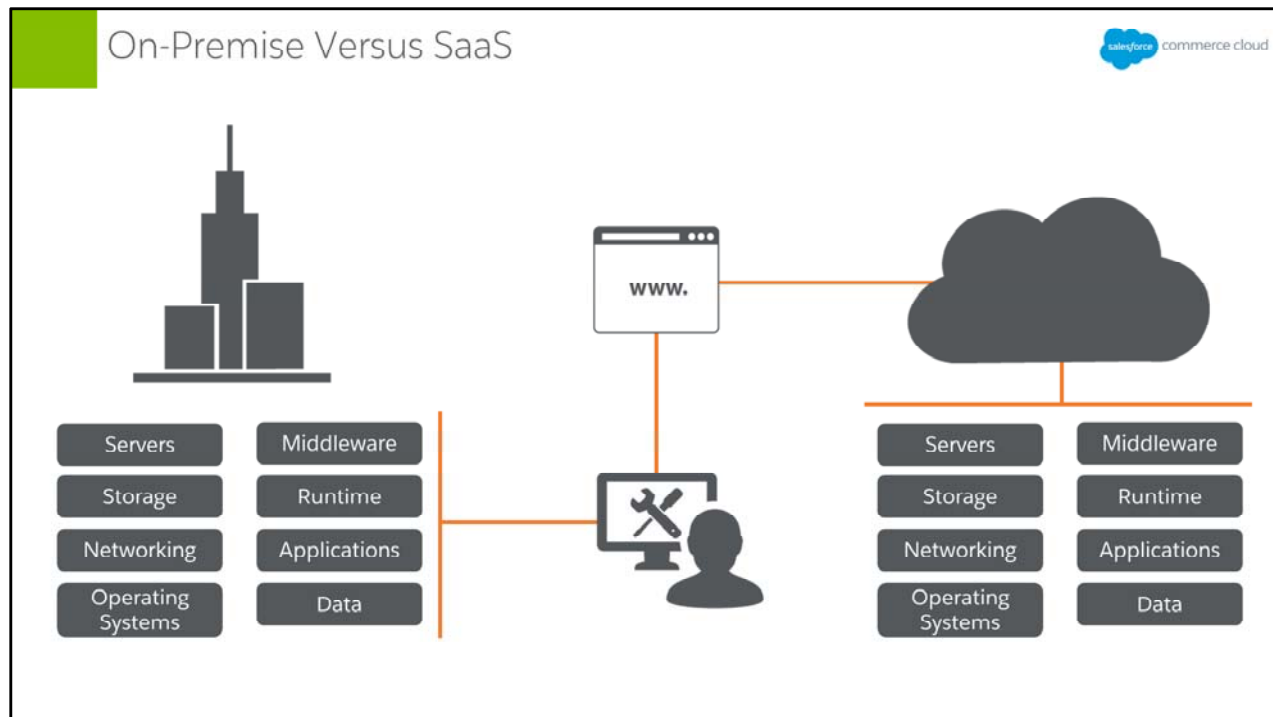
A

What is the difference between an on-premise solution and a SaaS solution?

B

How are tenants managed on Commerce Cloud Digital?





The difference between an on-premise solution and a SaaS solution lies in the location, ownership, management, and deployment of hardware and software.

An on-premise solution follows

the traditional model of software delivery. The organization's IT team is responsible for servers, storage, networking, operating systems, virtualization, middleware, runtime, applications, and data.

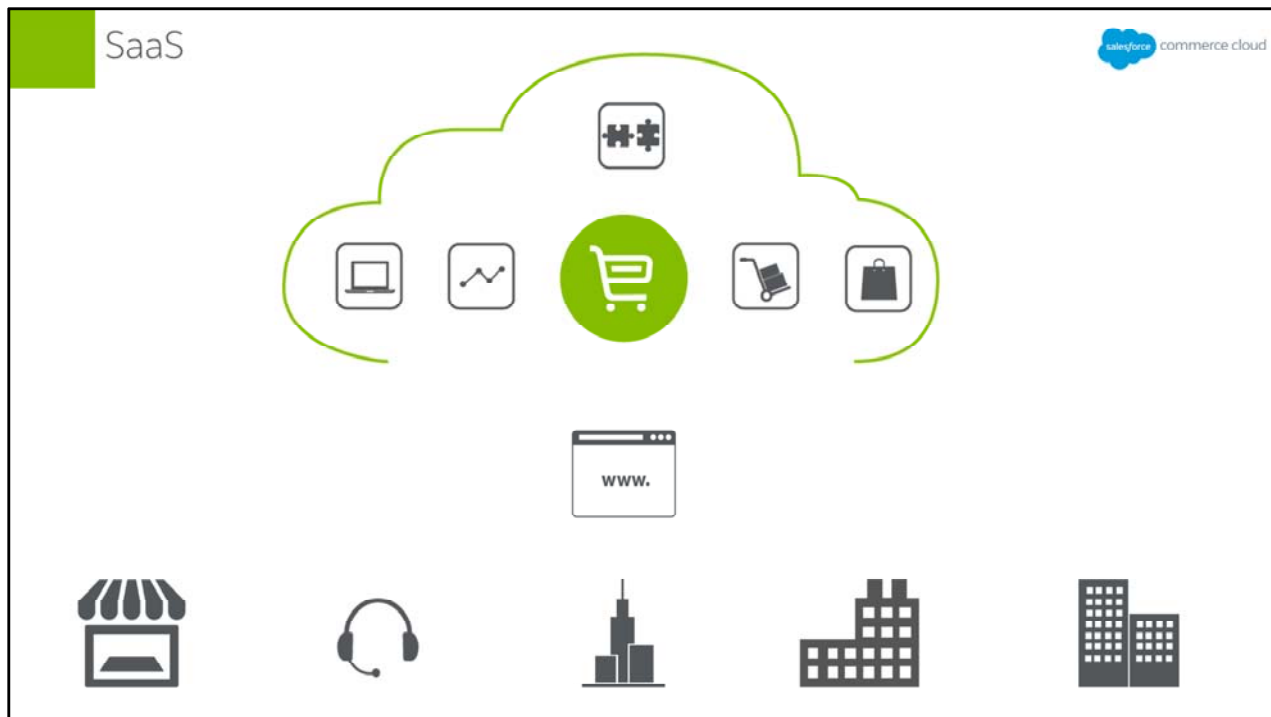
A SaaS solution uses a web delivery model, in which applications reside in the cloud and are accessed on the client's side using a web browser. The organization leases hardware and software resources in the cloud and vendors manage them, freeing the organization to focus on running its business.

While the difference between on-premise and in-the-cloud is significant for the organization and its IT group, for developers the difference is minor. Instead of installing, configuring, and maintaining a development environment on a local computer, each developer connects to an instance and uses an integrated development environment (IDE) to work in the cloud. Developers have no access to the runtime or operating system, but they can access the cloud application through an IDE and the tools that are provided as part of the SaaS solution.



When the commerce infrastructure is on-premise, it is likely to be widely distributed. For example, web servers and web designers for both traditional and mobile interfaces might reside in one location. Inventory could be stored in a warehouse in another location, from which delivery might also be managed. Order management and contact center staff may be housed in yet another location. And of course, there are probably multiple store locations, each provisioned with one or more kiosks and a complete point-of-sale system including registers, scanners, and other devices.

That's a lot of hardware, software, peripherals, and staff spread out, each requiring maintenance and management on-location. It can be challenging to keep it all compatible, up-to-date, and running smoothly with little or no down time.

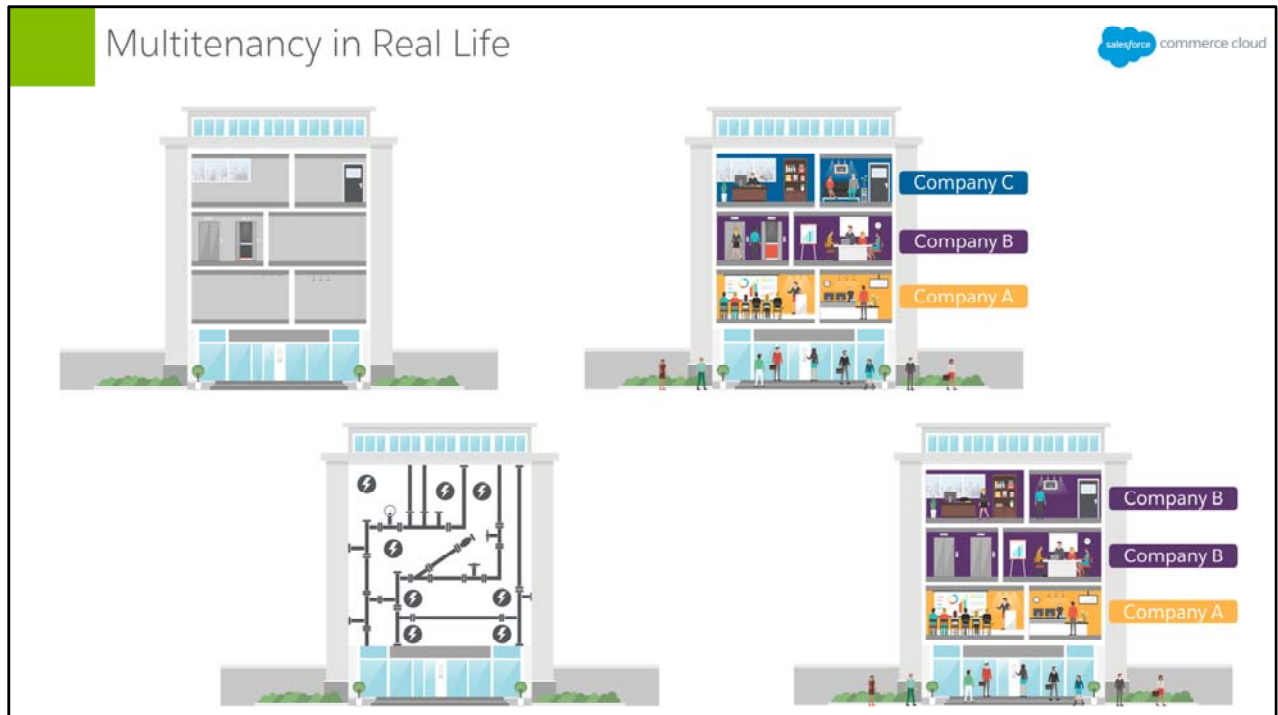


With SaaS, both the physical location and the management activities of a commerce infrastructure are centralized in the cloud. Software in the cloud functions on a single platform, the Internet, with a single type of client interface, a web browser. There is no need to maintain multiple compatible operating systems. Although the store, contact center, order management, inventory, delivery, and web site staff might be widely distributed, they all use web browsers to interact over the Internet with a single comprehensive commerce system.

For developers, working in a SaaS environment means shorter intervals between releases, including upgrades, enhancements, and bug fixes. To keep up with the pace of SaaS development, developers often adopt agile development methodologies and management tools. Because SaaS applications often employ a service oriented architecture, developers work extensively with web services and APIs.



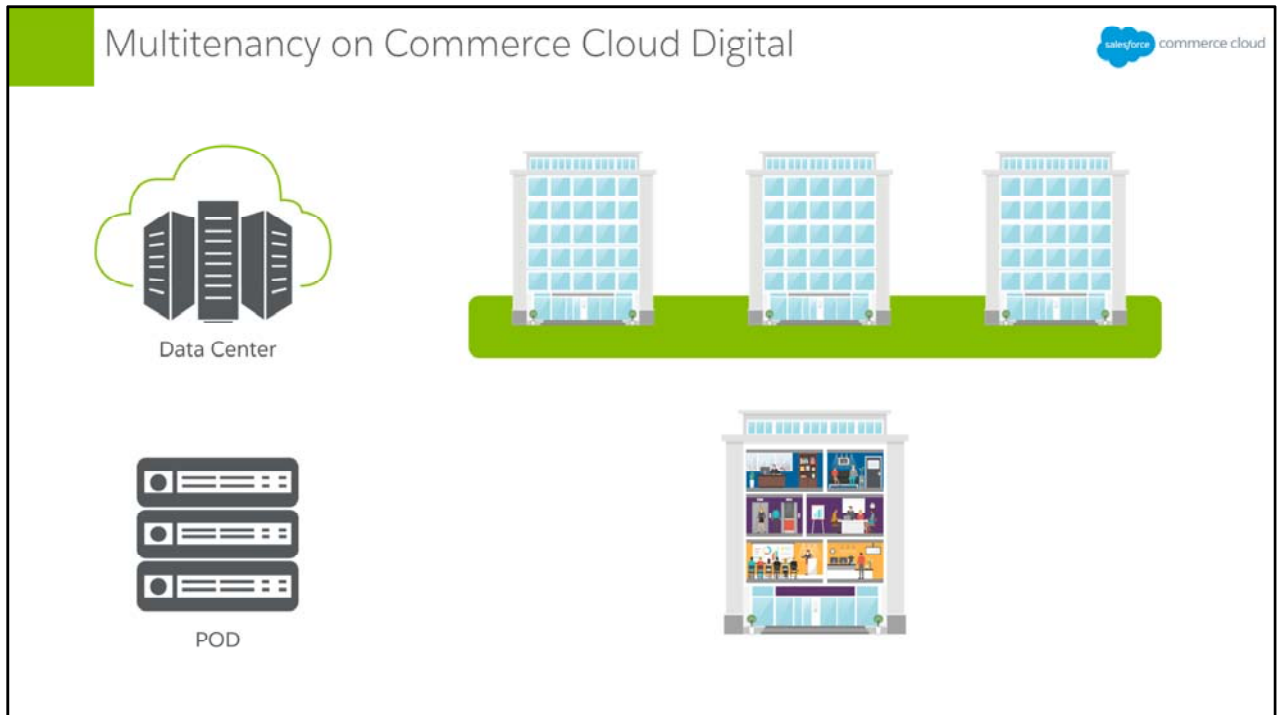
# Student Guide



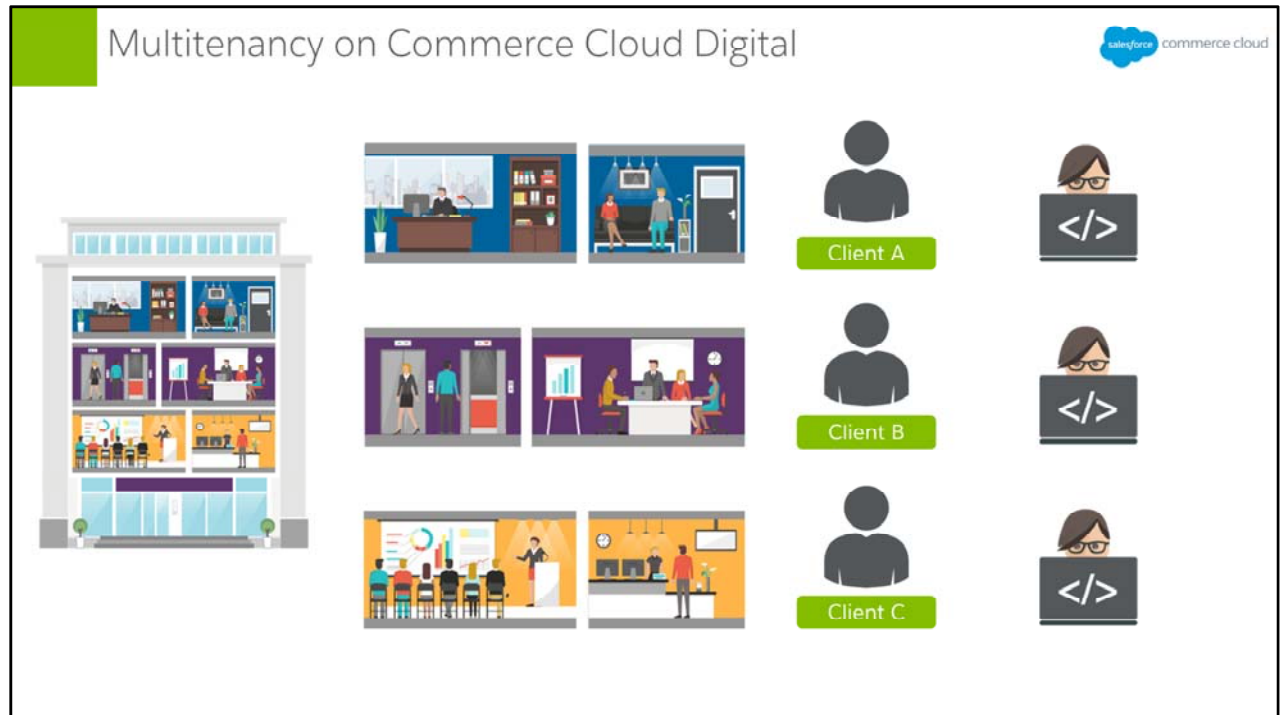
Multitenancy is an architecture that cloud platforms use to share resources in a manner that is cost-effective and provides security. Think of it like leasing space in an office building. You get your own dedicated office space, and walls, doors, and security maintain your privacy.

All tenants share resources like power and water, and a property management group takes care of building maintenance. You can customize your space with color, furniture, and decorations to create an environment that suits your business. As your business grows, you might lease additional space.

Now that you understand this analogy, let's see how Commerce Cloud Digital implements multitenancy.



As you learned in *Commerce Cloud Digital Overview*, the Digital architecture is distributed through numerous data centers on several continents. Each data center is like an office park, in which there are multiple office buildings. Each building corresponds to a point of delivery (POD) in the data center.



Each POD houses multiple tenants sharing resources such as networking, storage, and security. Within a POD, one realm is provisioned for each client or brand. A realm functions like an office space, keeping its tenant's data and resources separated from the other tenants. Developers work within these realms.



## Next Steps

Now you have a general understanding of software-as-a-service and how Digital implements multitenancy. Next we introduce members of the project team and describe the phases of a storefront development project.



How do I develop a storefront site?





## 02 How do I develop a storefront site?



When you have completed this module, you should be able to answer these questions:

A

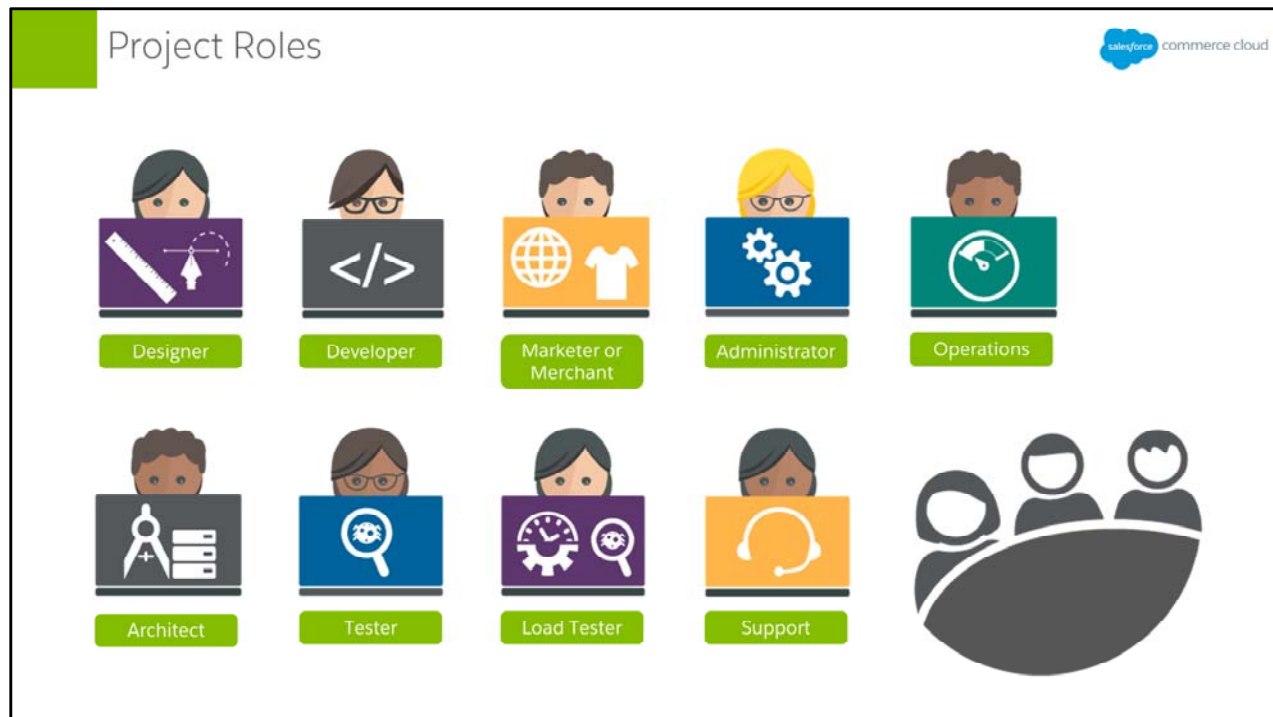
Who do I work with to develop my storefront site?

B

What is the process for creating and launching a storefront site?



# Student Guide



Developing a Commerce Cloud Digital storefront requires a cross-functional group of people who form a project team. To work together effectively, team members should understand one another's roles and responsibilities.

Although each organization and each project is based on the organization's unique needs, the team typically includes designers, developers, marketers and merchants, administrators, operations personnel, support representatives, architects, and testers. Depending on the size of the team and the business needs of the organization, some team members may fill multiple roles.

**Designers** consult with marketers and merchants to understand the brand vision and business needs, design an appropriate look and feel for the storefront, and then work with developers to integrate their design into the development of the site.

**Developers** build the storefront site by creating and customizing templates, adding content slots, implementing custom functionality, and integrating with external systems.

**Marketers** create the strategy for driving customers to the storefront and **merchants** engage customer interest and then present product information and promotions that entice customers to purchase products.

**Administrators** manage, maintain, and troubleshoot problems with instances, integrated systems, data transfers, and users and permissions.

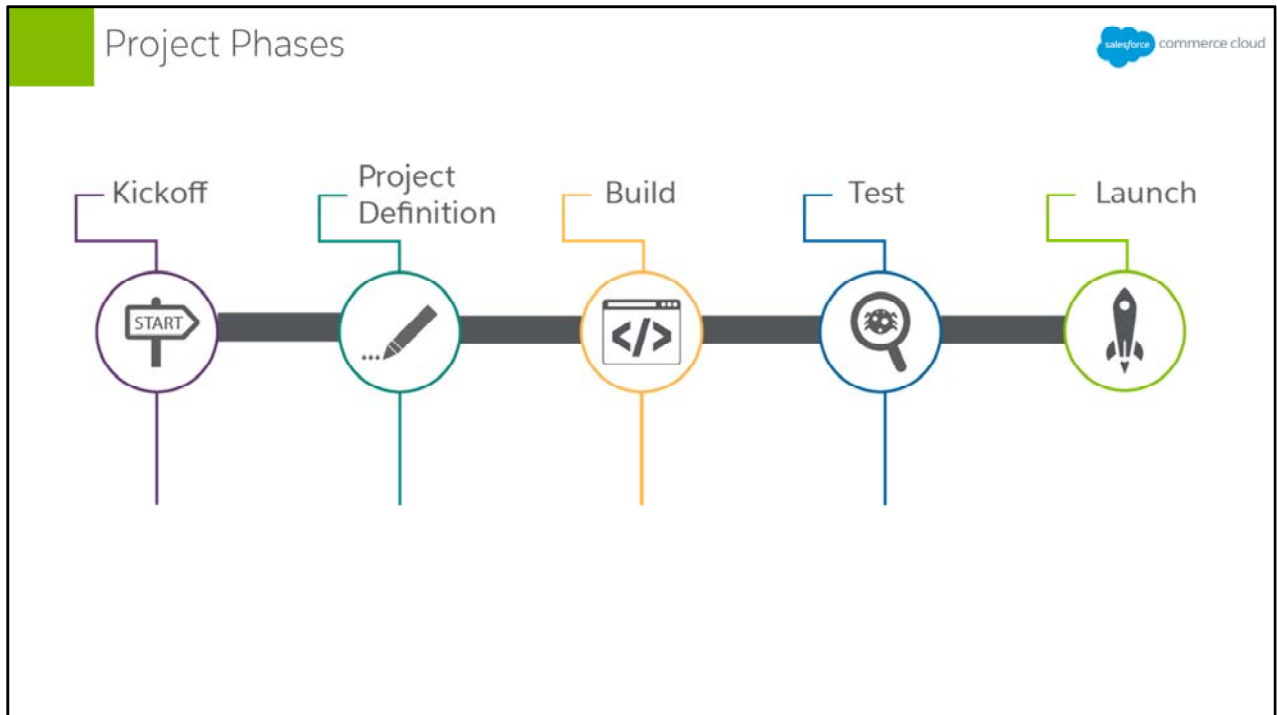
**Operations personnel** determine the anticipated average and maximum volume of shopping on the storefront site, and then identify and configure resources to maintain optimal performance.

**Architects** work with designers to plan and document the design of the site, the required dataflows, and specifications for the system interfaces.

**Testers** explore and evaluate all areas of the storefront site while it is built, providing real-time quality feedback to the project team. They find defects and report them to the development team to fix.

**Load testers** develop automated test scripts that simulate heavy demand on the storefront site and measure how the site performs under that load.

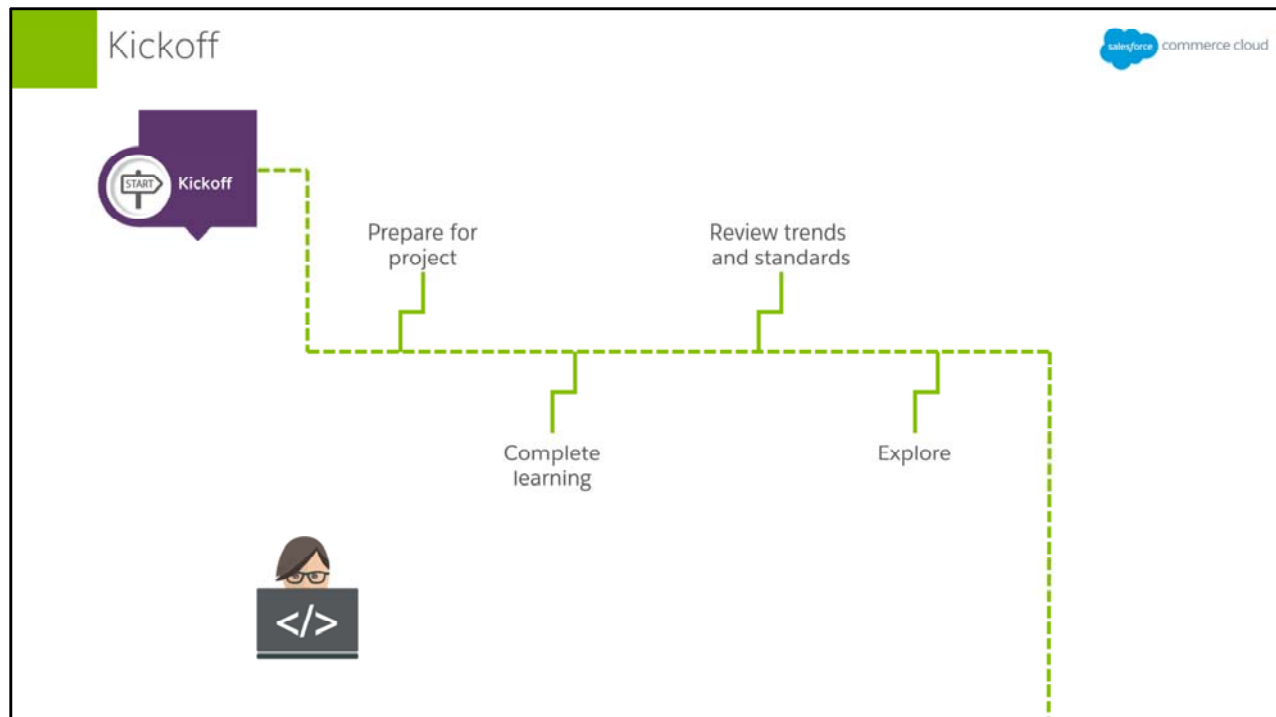
**Support representatives**, while not active members of the project team, are available to assist with initial site setup tasks as needed.



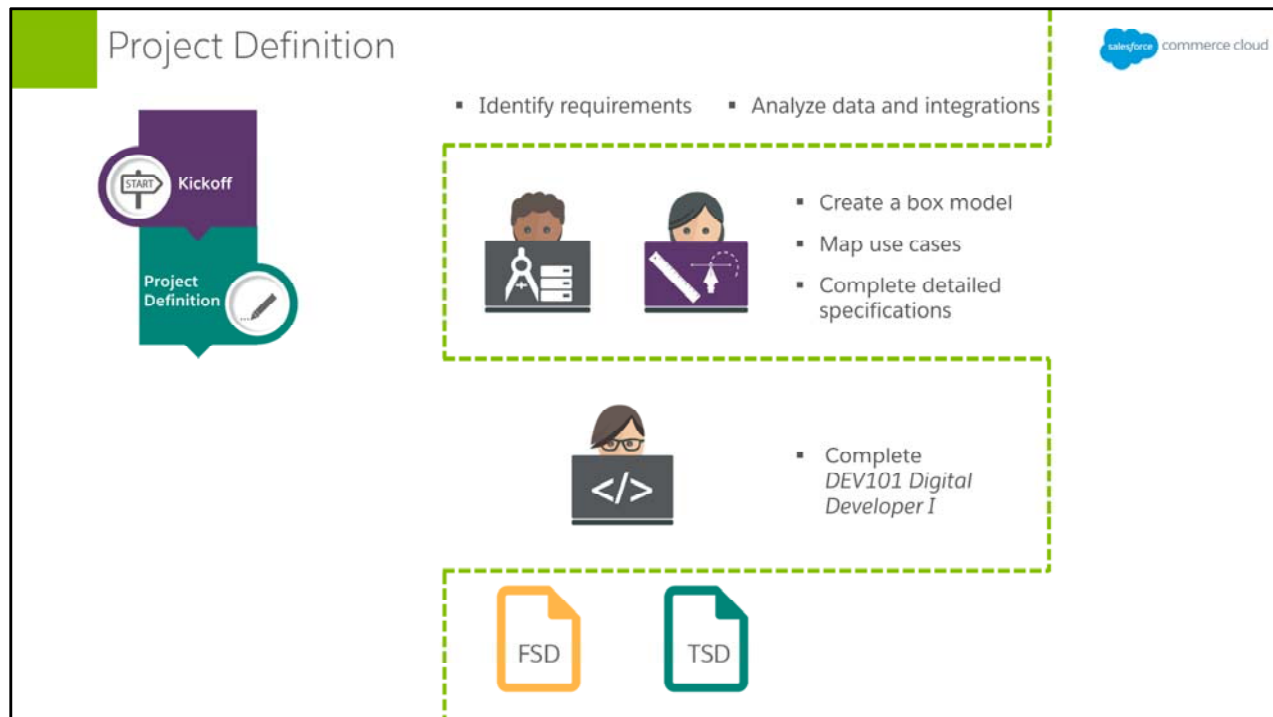
Although every Digital implementation is unique, they all have five main phases: Kickoff, Project Definition, Build, Test, and Launch.



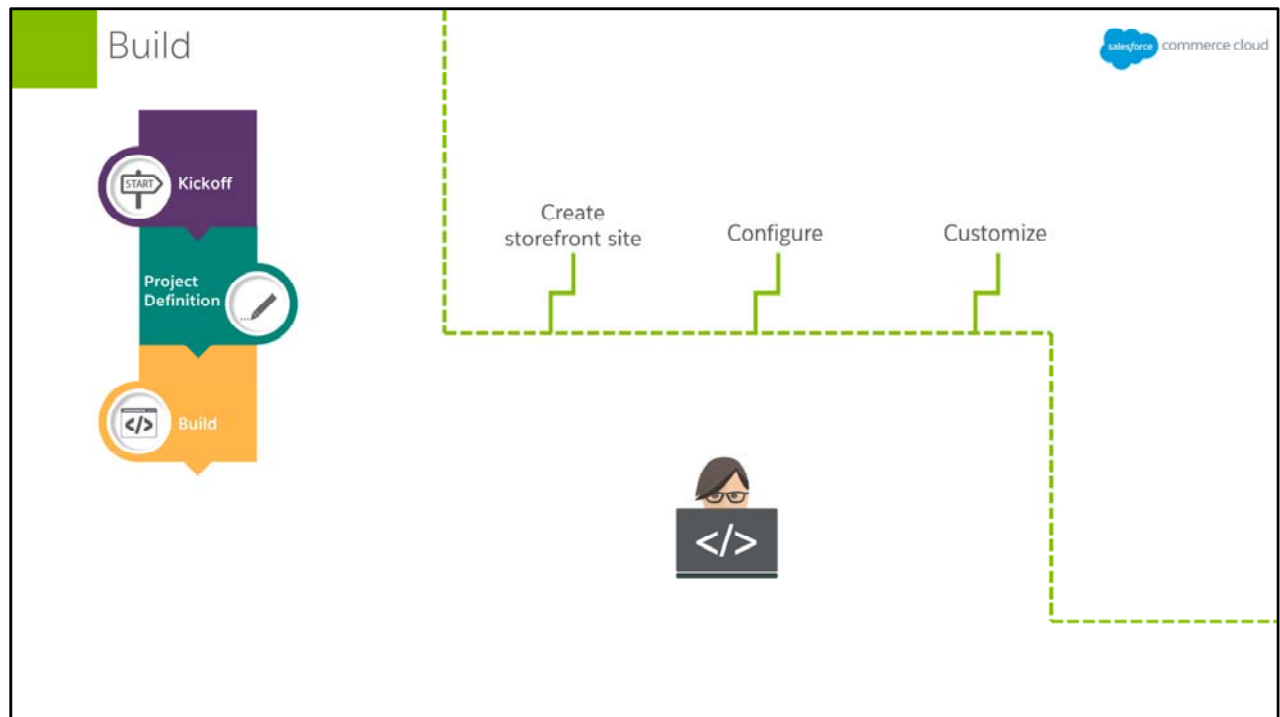
# Student Guide



The goal of the kickoff phase is to establish a shared high-level understanding of the project. During the kickoff phase, developers prepare for the project. They complete the *Commerce Cloud Digital Overview* and *Commerce Cloud Digital Architecture Overview* elearning courses. If developers are new to ecommerce development, they learn about ecommerce trends and standards, such as the Payment Card Industry Data Security Standards (PCI DSS). And they explore Commerce Cloud resources, such as the developer community, documentation, SiteGenesis repository, and Center of Excellence.

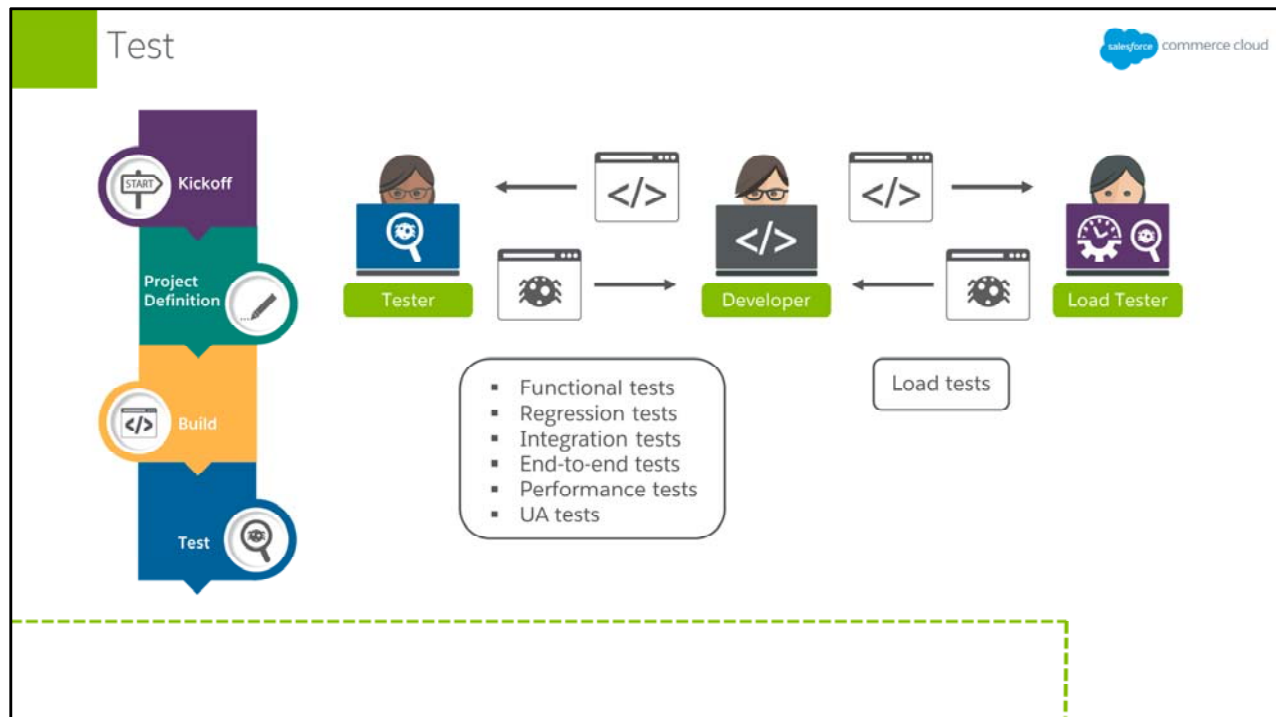


The project definition phase is characterized by incremental, iterative design activities. During this phase, developers identify functional and non-functional requirements, and analyze required data flows and integrations. They work with designers and architects to create a box model of the system landscape, map use cases to system interfaces, and complete a detailed specification for each interface. Developers also complete the *DEV101 Digital Developer I* course. Finally, they may also be involved in finalizing the functional specification document (FSD) and the technical specification document (TSD).



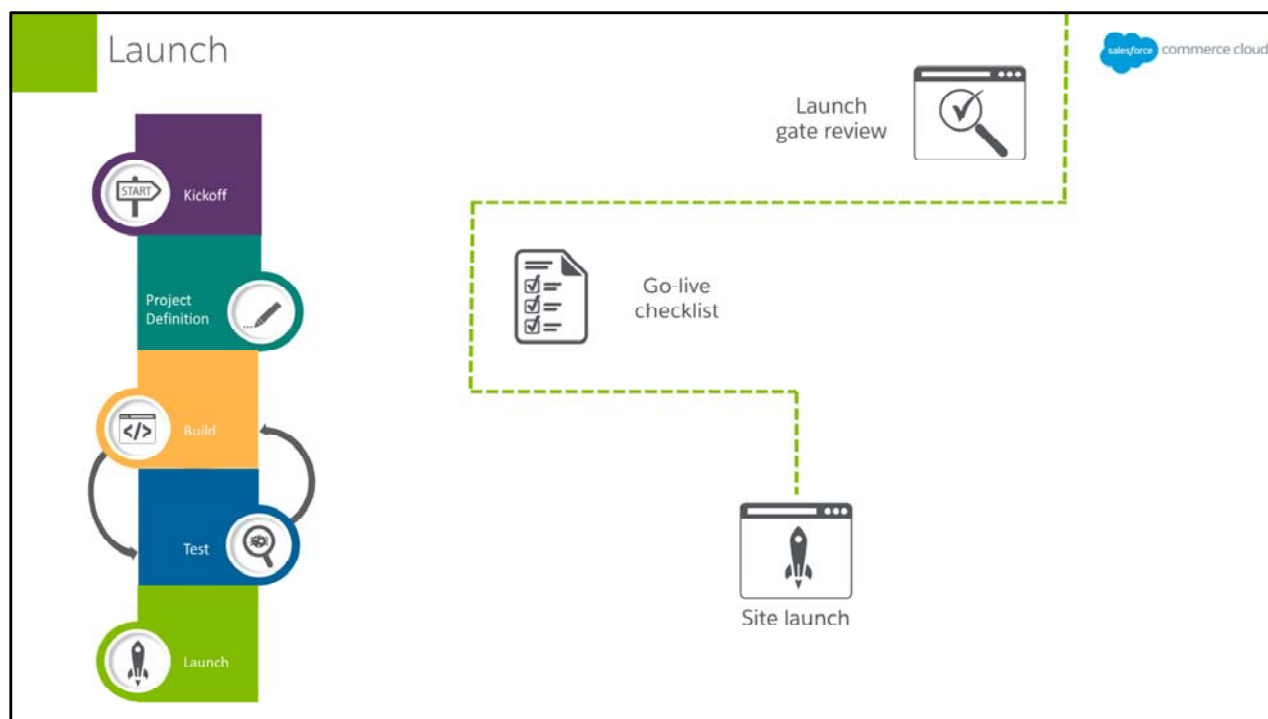
During the build phase, developers create storefront sites, configure integrations, and customize functionality to satisfy the requirements in the FSD and TSD.

The duration of the build phase depends on business needs and project size. The scope of build activities increases as the number of storefront sites, brands, and geographies increases. The volume of custom code needed, and the number and complexity of integrations also affect the duration of the build phase.



During the test phase, testers conduct testing on the Development instance. Testing can include functional tests, regression tests, system integration tests, end-to-end tests, performance tests, and user acceptance tests. Testers submit defect reports and developers analyze and resolve defects. Developers and testers iterate through cycles of build and test until each site is ready for launch.

Developers may also participate in some testing activities. For example, they might help to develop a load profile and work with a load test partner to create, monitor, and revise load test scripts.



In preparation for launching new storefront sites, the project team completes a launch gate review and go-live checklist. When all go-live preparations and configurations are complete, the storefront sites are launched.





## Next Steps

Now you are familiar with the players and phases involved in a storefront development project. Next we explore the Digital architecture and the developer's development environment. We also examine the relationship between Digital and its SiteGenesis reference application.



How are Commerce Cloud Digital and SiteGenesis related?

© Copyright 2017 salesforce.com, Inc. All rights reserved. Various trademarks held by their respective owners.



## 03 How are Commerce Cloud Digital and SiteGenesis related?



When you have completed this module, you should be able to answer these questions:

A

How is Commerce Cloud Digital structured?

B

What tools can I use to work with Digital?

C

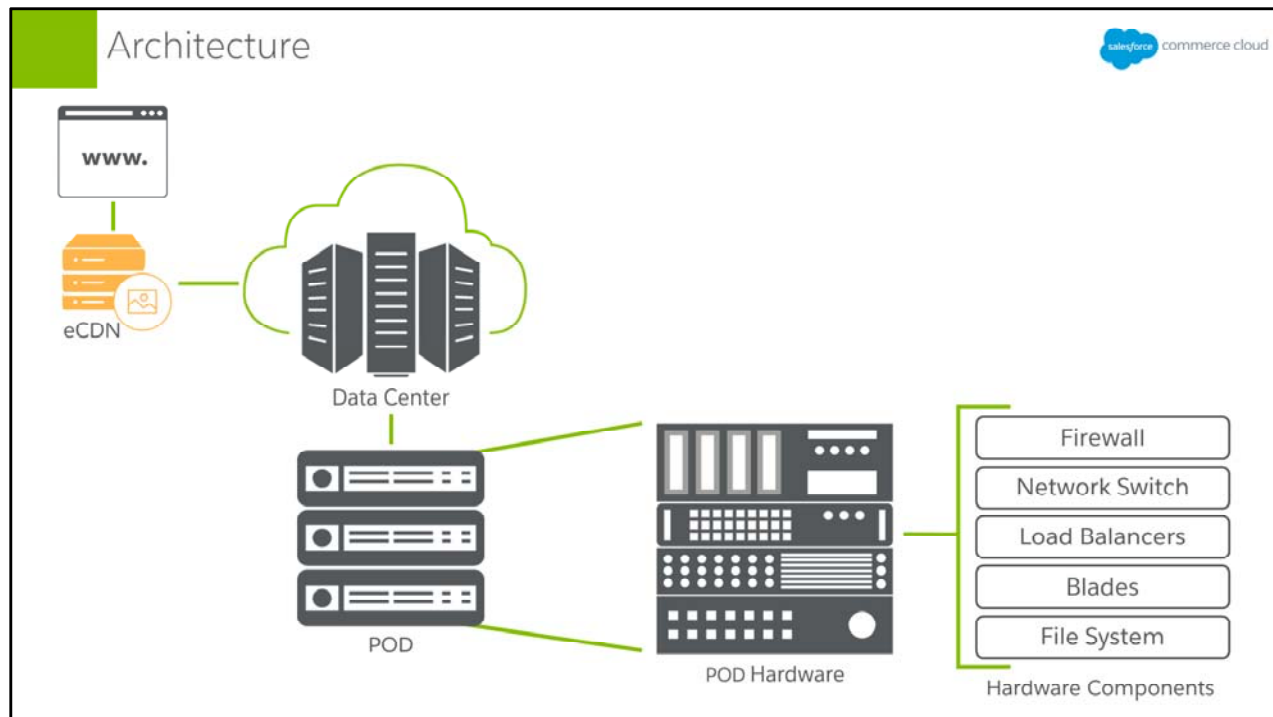
What is the development model for Digital?

D

What happens when Commerce Cloud releases an update to Digital?



# Student Guide

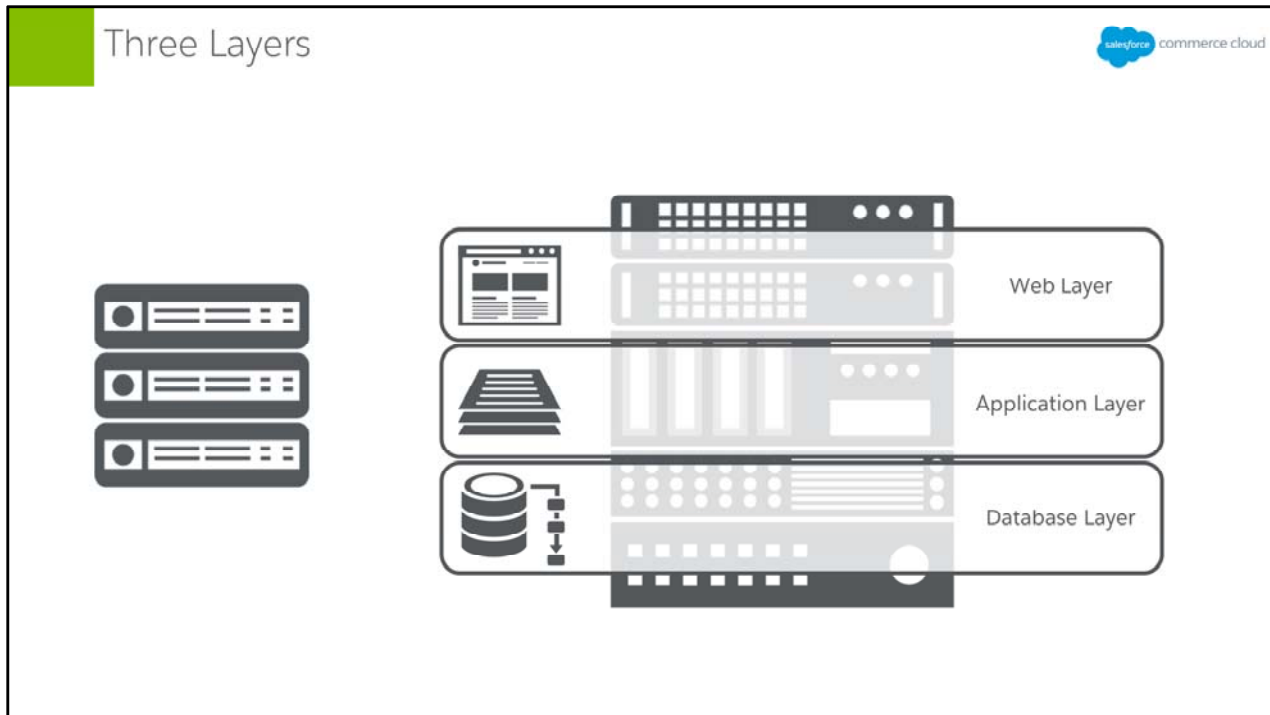


Commerce Cloud Digital resides in many geographically distributed data centers that provide processing, networking, and storage. Each data center supplies the power, network connectivity, and cooling required to keep its hardware running smoothly.

Standard with all Digital implementations, the Commerce Cloud embedded content delivery network (eCDN) stores and serves all static content, such as images, css, and javascript, improving performance and scalability. All Digital implementations are positioned behind the eCDN, enhancing security.

For practical purposes, the structure of Commerce Cloud Digital starts with the POD. Each POD contains a combination of hardware and software, including web servers, a web adapter, a firewall, a network switch, load balancers, blades, and an appliance that provides a file system shared by all blades.

Redundancy and failover are built into the POD design to ensure that system or power failures do not cause service interruptions. Most maintenance tasks are performed without affecting storefront sites.

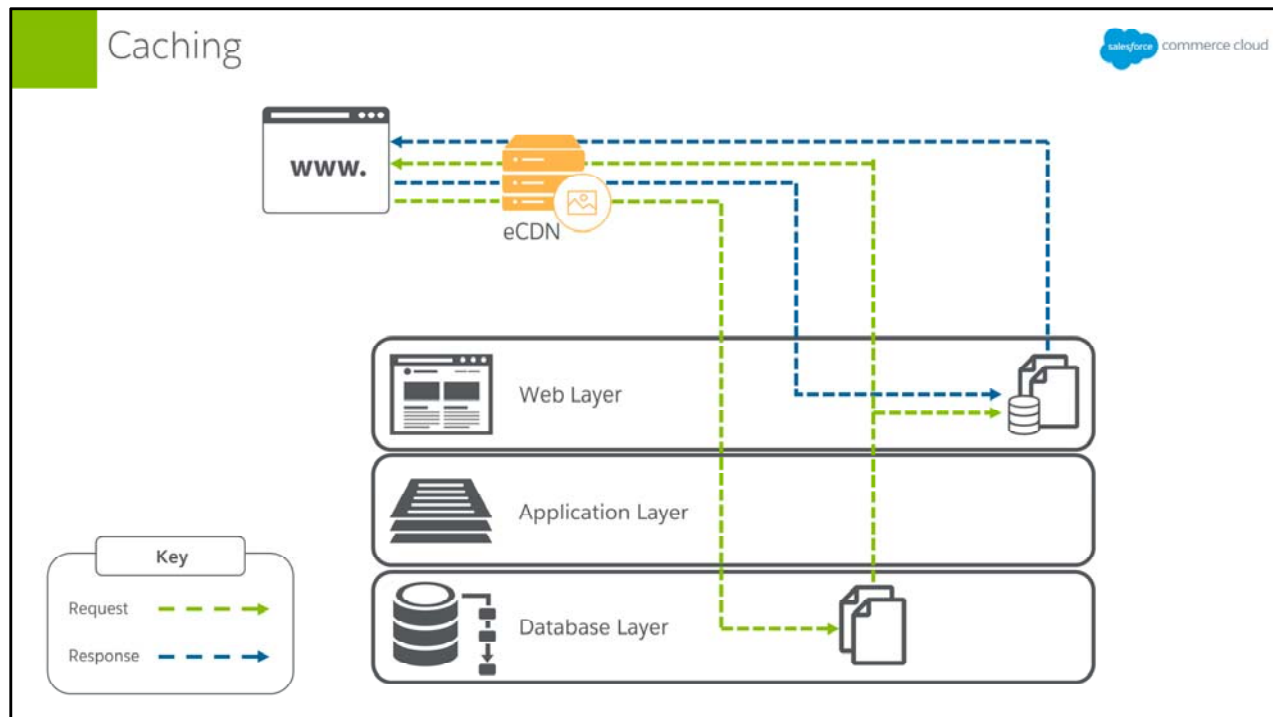


Each POD has a typical three-layer structure that consists of a web layer, an application layer, and a database layer.

The **web layer** is responsible for presentation. It determines how information is presented in a web browser. The web layer includes a web server, web adapter, and page cache.

The **application layer** represents the business logic and workflows. It processes commands, makes decisions, evaluates, calculates, and passes data between the web layer and the database layer.

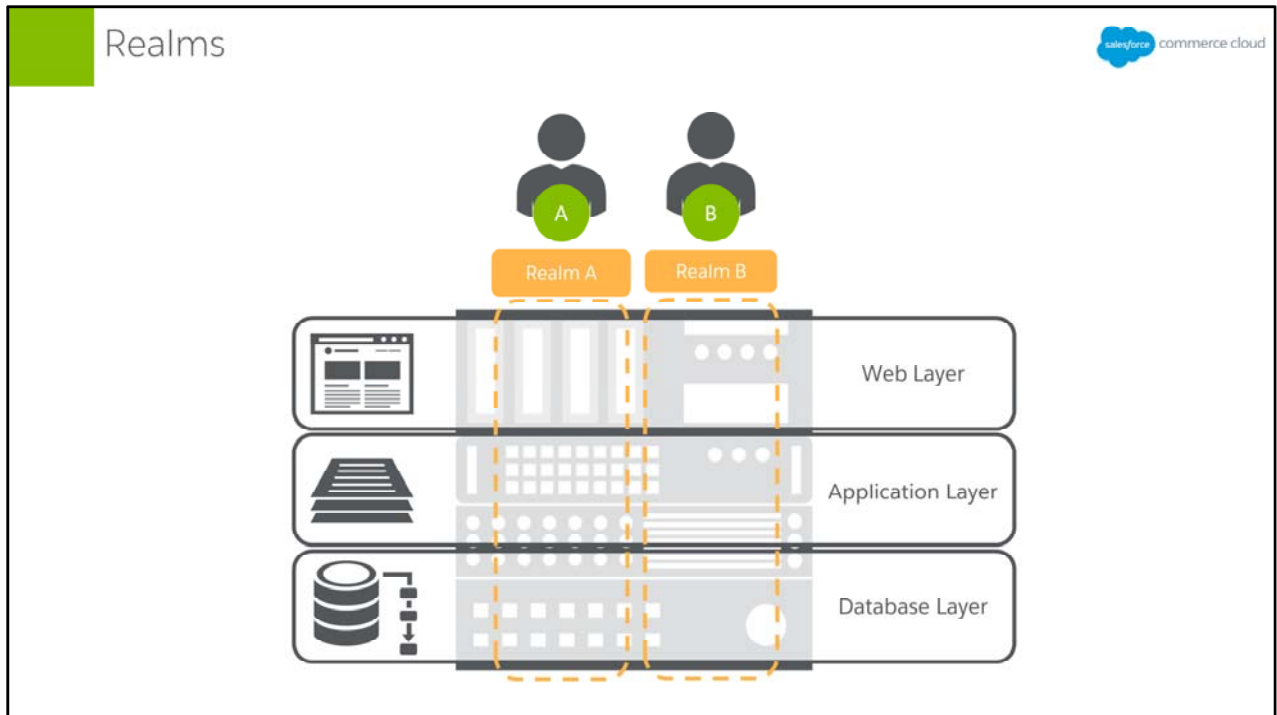
The **database layer** stores information. It receives requests for data, queries the data source, retrieves data, and then returns requested data to the application layer. The database layer contains the database and a file storage system.



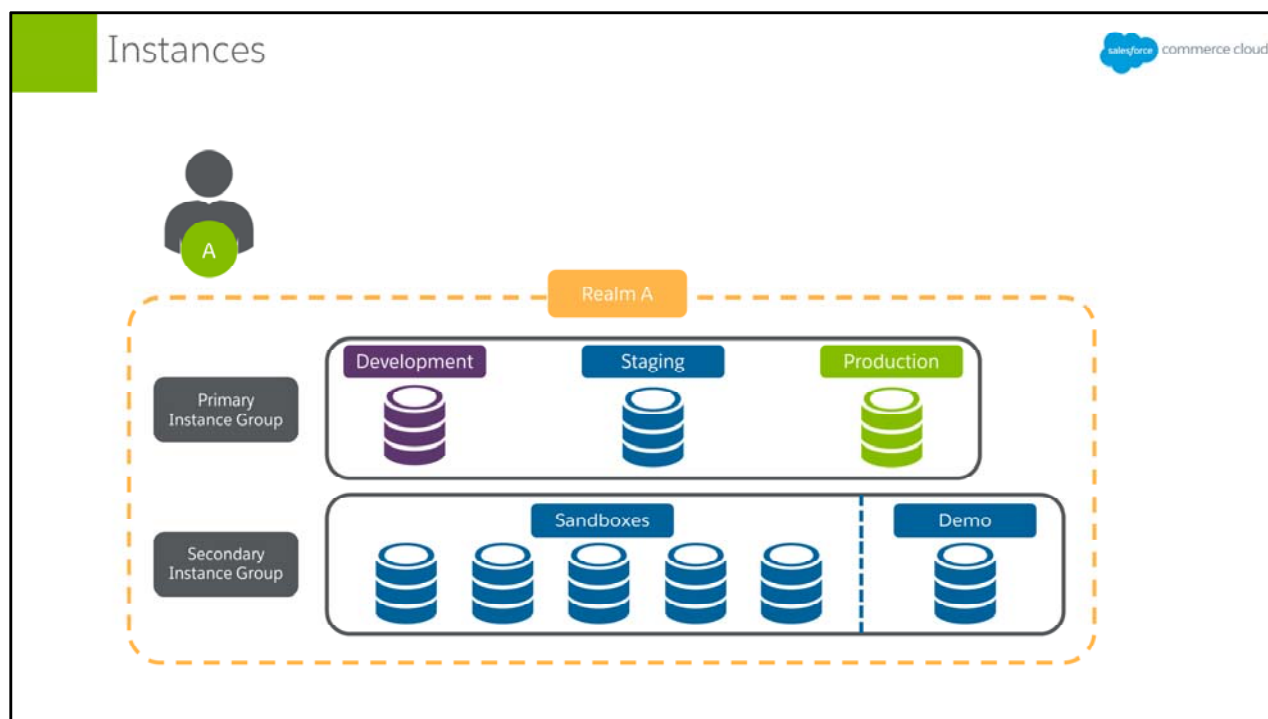
Developers can improve performance by caching storefront pages at the web layer. Let's look at how a page request is handled when caching is enabled for a storefront site.

A user browses the storefront and clicks a link. The browser sends a page request. The request passes through the eCDN to the web layer. The web layer forwards the request to the application layer. The application layer processes the request, querying the database layer if necessary, and creates the page result. The application layer then passes the page result to the web layer, which does two things: It stores a copy of the page in the page cache, and it passes the page back to the web client that initiated the request.

The next time a web browser requests this page, the web layer returns the page from the cache, reducing the number of calls to the database layer, until the cache expires or is deleted.



PODs are divided into multiple realms. A realm is a portion of a POD's hardware and software resources that is allocated to a client or brand.



Realms contain instances. An instance is a copy of a client's sites, configurations, and customizations. Instances in a realm are separated into two instance groups:

The Primary Instance Group (PIG) contains these instances:

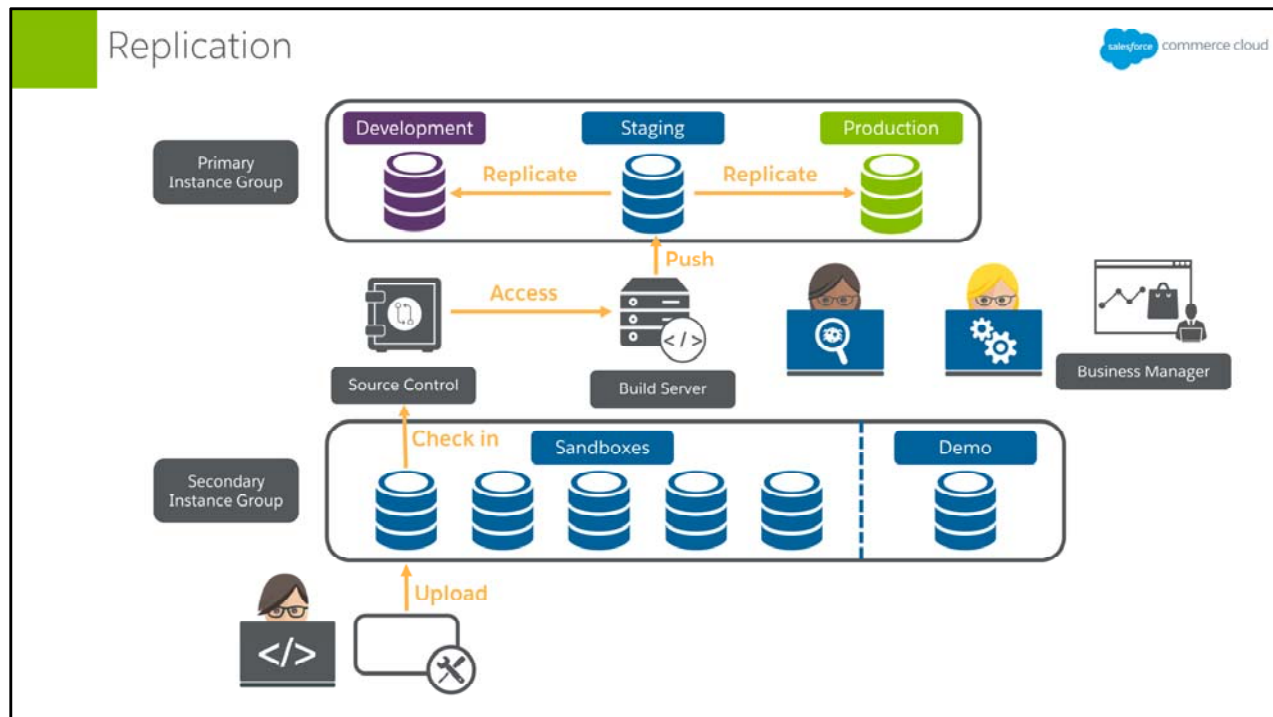
- The **Staging instance** is an environment where **system administrators** import data and upload code to prepare it for testing. **Merchants** create and maintain storefront data in the staging instance.
- The **Development instance** simulates the production environment and is where **testers** evaluate features and processes before they are deployed from staging to production.
- The **Production instance** is a live storefront.

The Secondary Instance Group (SIG) contains Digital sandboxes in which developers create and test code. The demo instance is a dedicated sandbox that is reinitialized with each release so that it always has the latest version of SiteGenesis for reference. Because the demo instance is frequently overwritten with new releases, it is not intended to store data or active development work.





# Student Guide

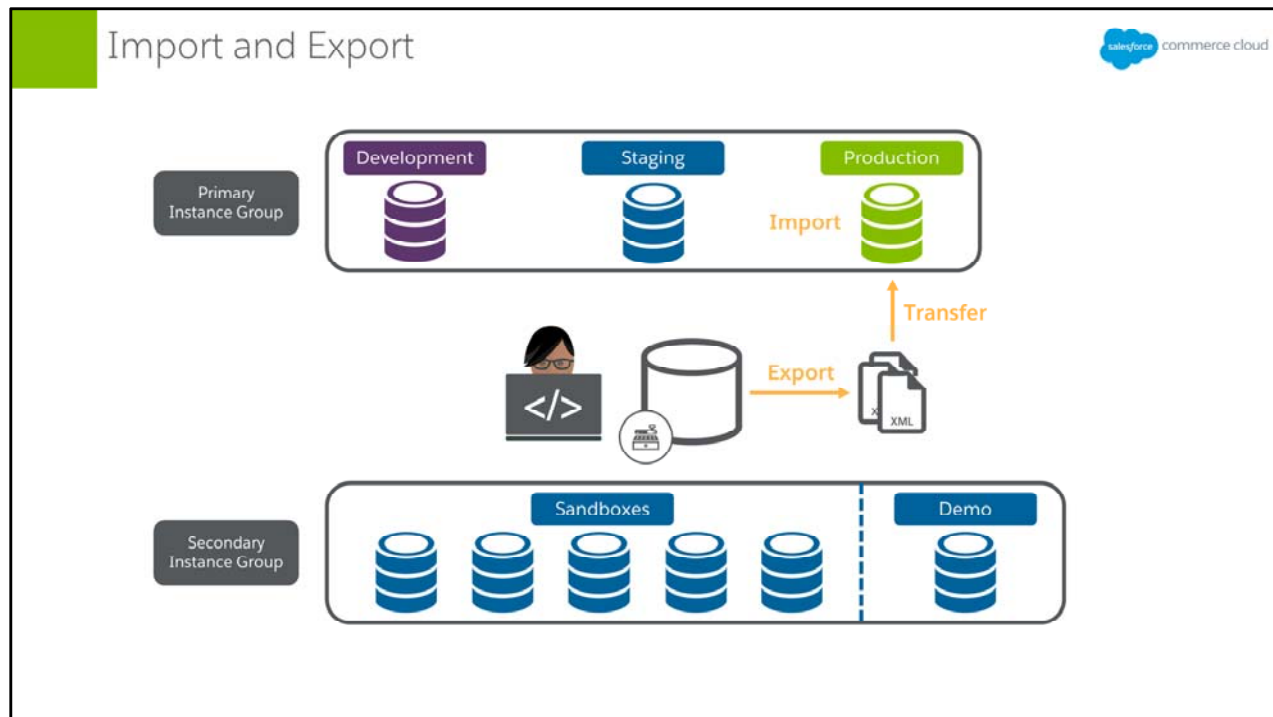


How data flows through Commerce Cloud Digital depends on the data source and target. Within the primary instance group, code and data are replicated from the staging instance to the development or production instance. All other movement of code, data, and configuration settings is performed by import and export processes. Let's look at replication first.

Each developer creates and edits code in an IDE and then uploads to a personal development sandbox. Typically, code is checked in to source control and then accessed by a build server. When the build process runs, it pushes a copy of the new code version to the Staging instance.

In the Staging instance, the code is combined with storefront data to form a fully-functioning ecommerce site. That site is then replicated to the Development instance, where an administrator verifies that the replication was successfully completed and testers evaluate the site to validate that it behaves as expected.

This cycle repeats until the Development instance is considered ready for release. Then the administrator uses the Business Manager to replicate from staging to production. To ensure that a site change is seamless and instantaneous, replication creates a copy, then the administrator activates the new version.



Now let's look at import and export. If you want to move data between the primary and secondary instance groups, or between an external system and a Commerce Cloud Digital instance, you must import and export. There are three steps to the import and export process. First, on the source instance, you export data to XML files. Second, you transfer the XML files to the target instance. Third, you import the XML files into the target instance database.

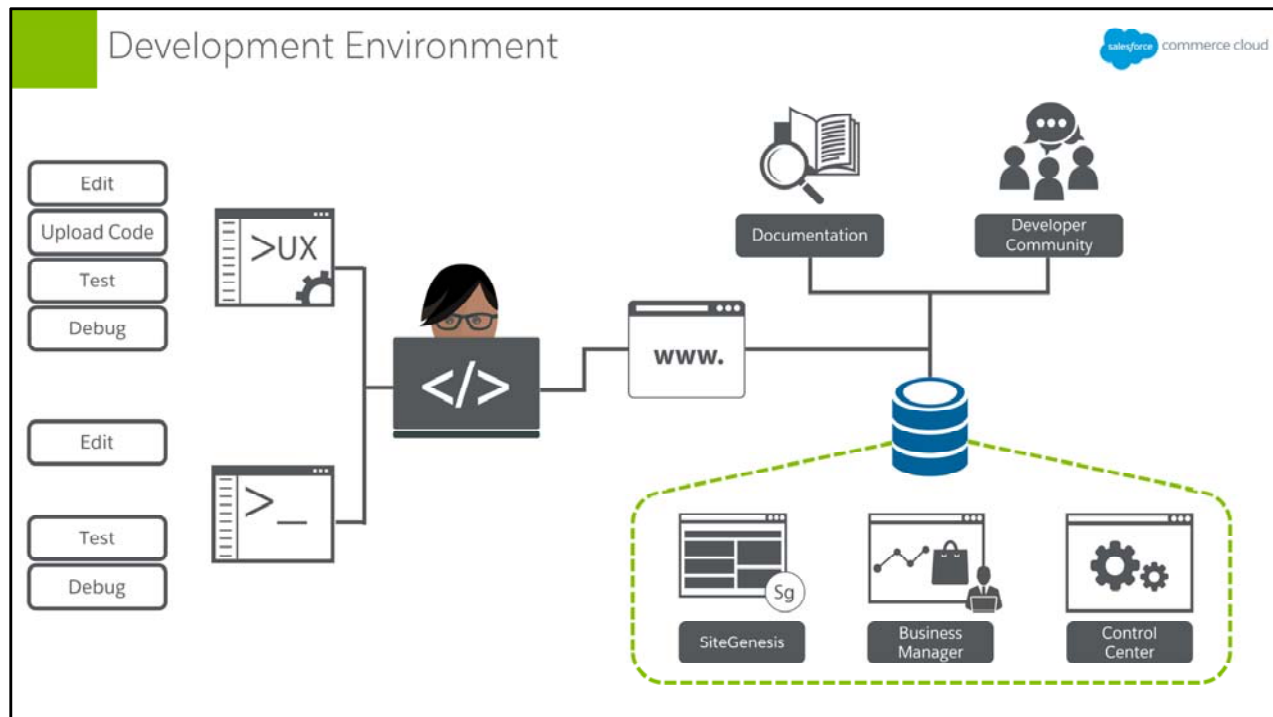
Some reasons to import and export include:

- Loading data from a product inventory management (PIM) system into the Staging or Production instance.
- Setting up multiple developer sandboxes with the code, data, and configuration settings of the development instance.
- Reproducing the configuration settings of a Production instance in a sandbox to facilitate troubleshooting a production issue.

When you export from a Digital instance, you choose which elements to export. For example, you might choose to export only the customer list, price books, or cache settings. For a full list of objects you can export, see the Digital documentation.



# Student Guide



Developers work in UX Studio, which is a storefront development environment that is provided as a plugin to the Eclipse IDE. With UX Studio, developers edit their code locally, upload to a sandbox, and then test and debug their code in the sandbox.

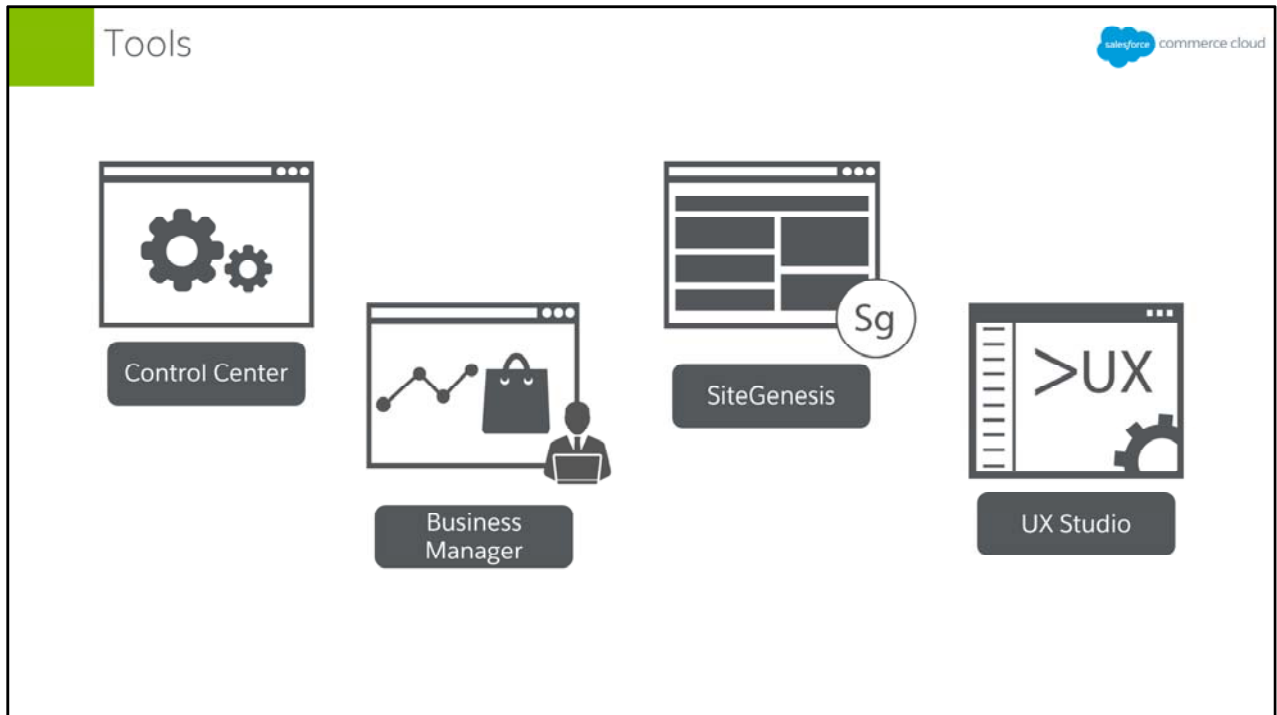
Since the introduction of JavaScript controllers in Digital 15.8, developers can edit, debug, and test controller code using other IDEs with plugins developed by members of the Commerce Cloud Digital community. However, UX Studio is the only IDE that uploads code to a sandbox instance.

From a web browser connected to a sandbox, the developer has access to other Digital tools used to develop the storefront site, such as the Business Manager, the Control Center, and a development instance of the SiteGenesis reference application.

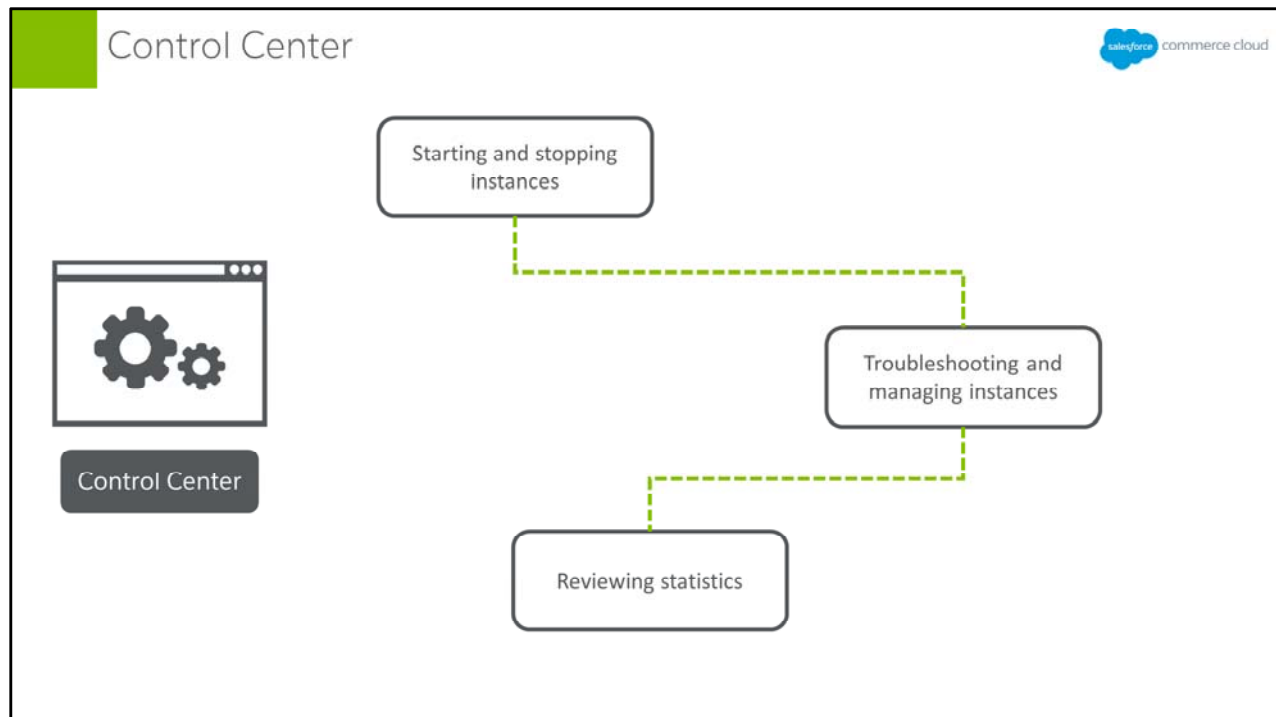
In addition, developers can refer to the Digital documentation and the developer community and Center of Excellence for information, support, and best practices.



## Student Guide

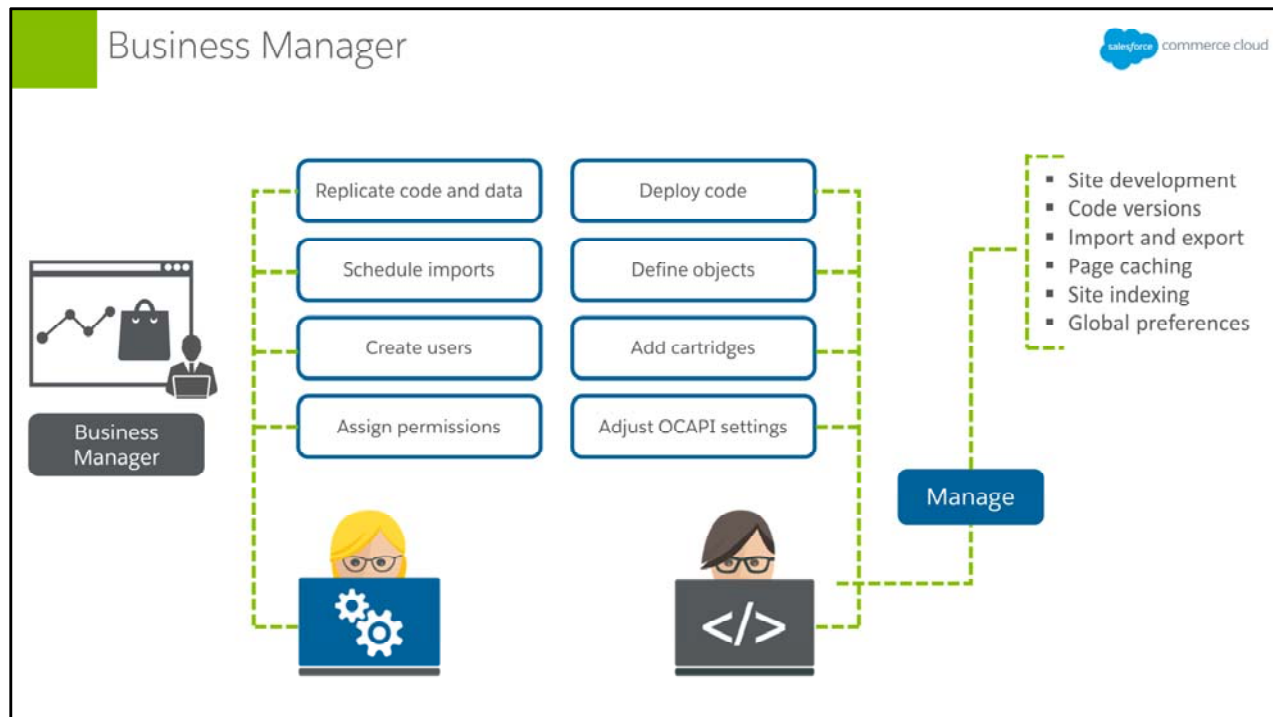


As you learned in *Commerce Cloud Digital Overview*, realms provide tools that help you to manage instances and to create, customize, configure, test, and deploy your storefront. Developers use each of these tools while creating and maintaining storefront sites.



The Control Center is an operations management application that system administrators and developers use to initialize, start, and stop instances, and to manage and troubleshoot system integrations and data transfers. The Control Center also provides detailed information about network utilization, data usage, and file system usage. Developers use this information while troubleshooting performance issues.

Logging into Control Center requires a user account. Within Control Center, each user's role, and the permissions assigned to that role, determine which actions the user can perform.

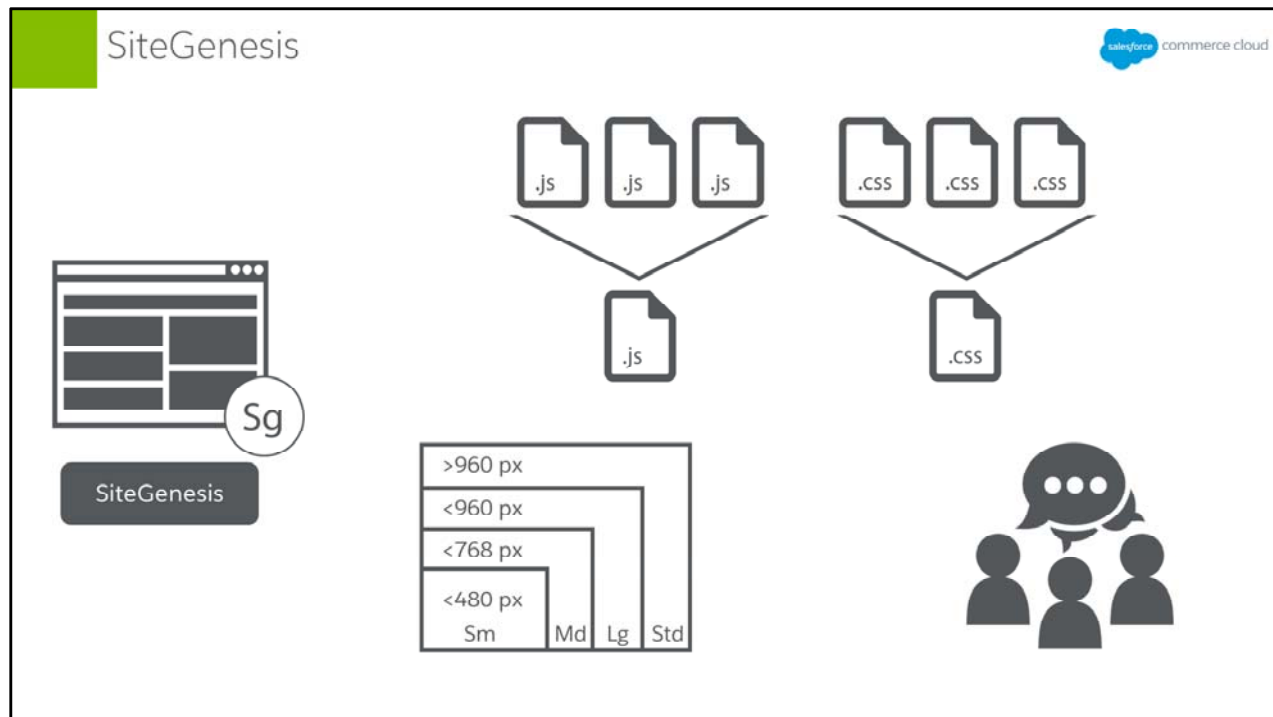


Although the Business Manager is used most frequently by merchants to manage storefront data, it is also used by administrators and developers.

Administrators use the Business Manager to replicate code and data, schedule imports from external systems, and create users and assign user permissions.

Developers use the Business Manager to deploy code, define system and custom objects, add cartridges to a site, and adjust Open Commerce API (OCAPI) settings.

They also manage site development, code versions, data import and export, page caching, site indexing, and global preferences for all of the sites in a single instance.



The SiteGenesis ecommerce site is a customizable reference application that developers and merchants use to explore Commerce Cloud Digital and learn about its features and capabilities. For merchants, it provides sample storefront data. For developers, it provides sample code in the form of JavaScript controllers, pipelines, scripts, and templates.

To optimize performance on the site and prevent namespace collisions, the client-side JavaScript and CSS for SiteGenesis are modularized. Developers work on individual .js and .css files which are then combined in one app.js file and one style.css file.

Because storefront sites are viewed on devices with varying screen sizes, SiteGenesis facilitates responsive web design with predefined styles and media breakpoints, one for each of four viewports:

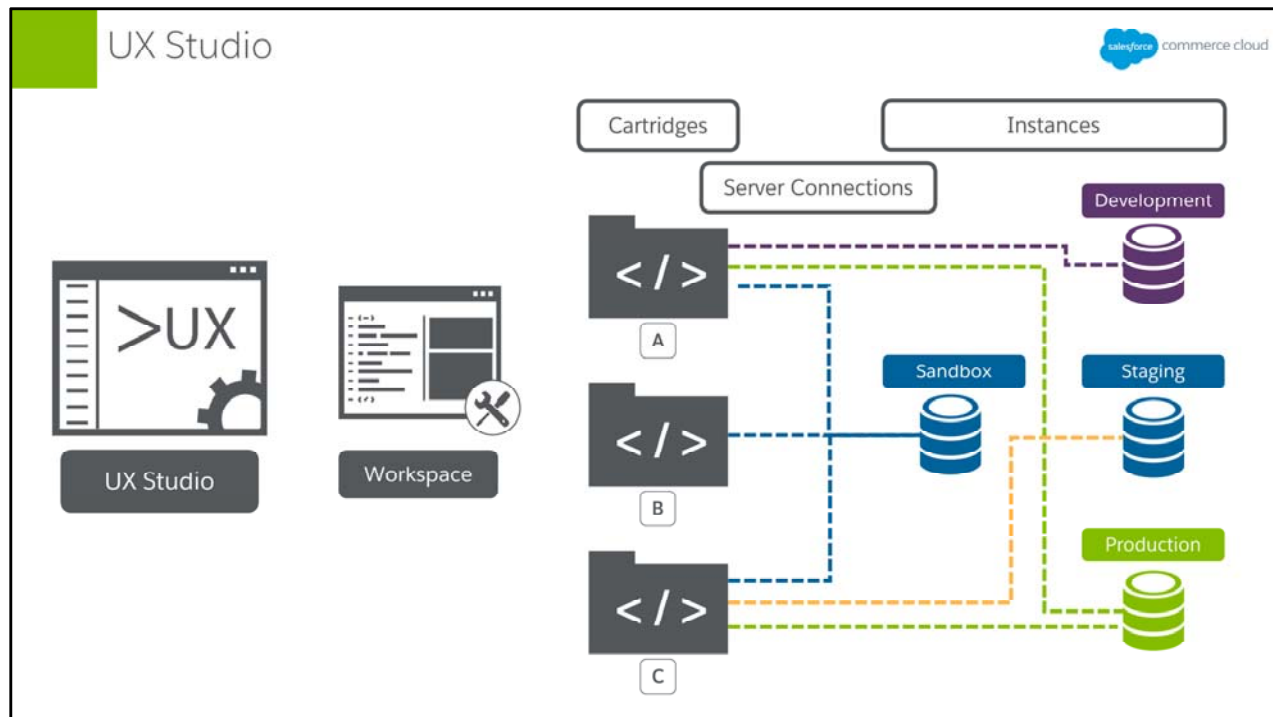
- Small: less than 480 pixels wide
- Medium: less than 768 pixels wide
- Large: less than 960 pixels wide
- Standard: more than 960 pixels wide

In the `_responsive.scss` file, each breakpoint specifies the layout-specific style attributes that define how content is displayed in one viewport, identified by its `min-width` value.

You can download the SiteGenesis source code, functional specifications, wireframes, and release notes from the [SiteGenesis](#) space in the Commerce Cloud developer community.



# Student Guide



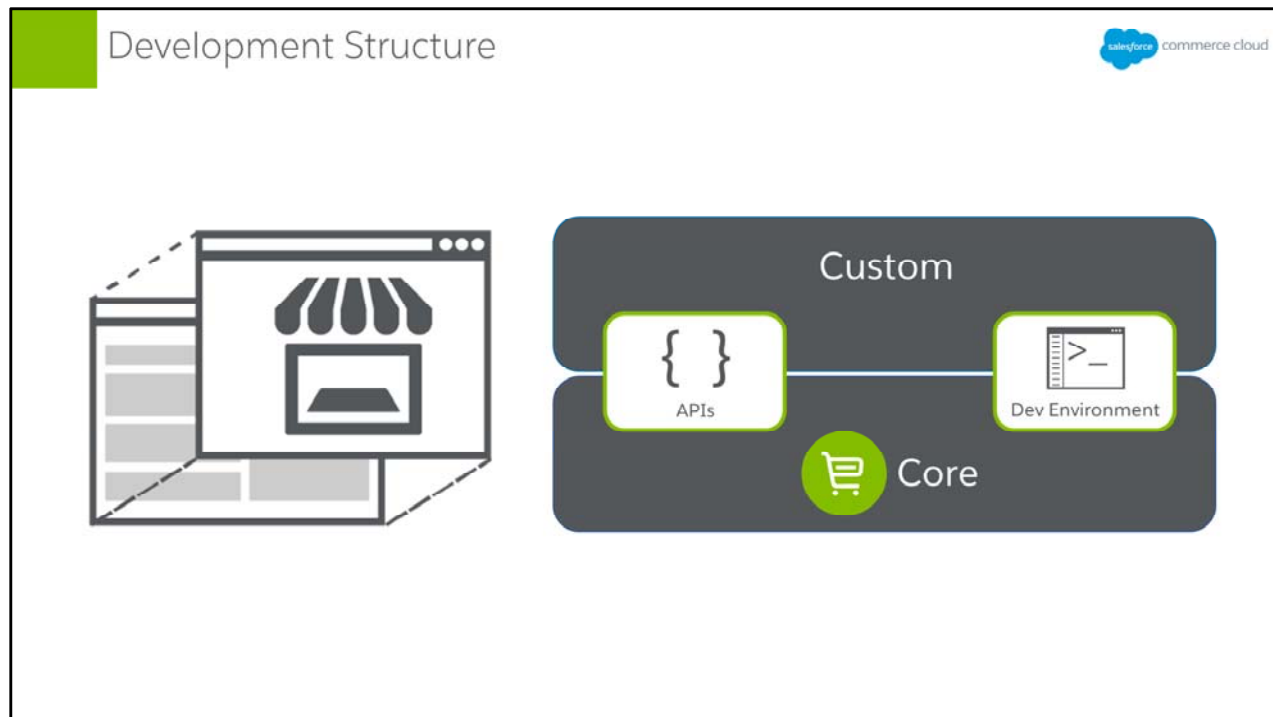
UX Studio is a storefront development environment that is provided as a plugin to the Eclipse IDE. The Eclipse workspace defines Eclipse-specific settings, project file paths, and server connection details.

Within the UX Studio perspective, developers access Digital cartridges, which are represented as projects. Cartridges are mapped to instances by server connections.

Each cartridge has one or more server connections, each connecting the cartridge to a different instance. Multiple cartridges can share a single server connection.

Developers use UX Studio to build and customize the storefront by creating and editing the application presentation and implementing business logic. Within UX Studio, developers work with template layouts, payment processors, shopping cart details, shipping options, internationalization, and integration with other applications. Developers can use UX Studio to examine sample code from SiteGenesis and incorporate it into their own storefronts.

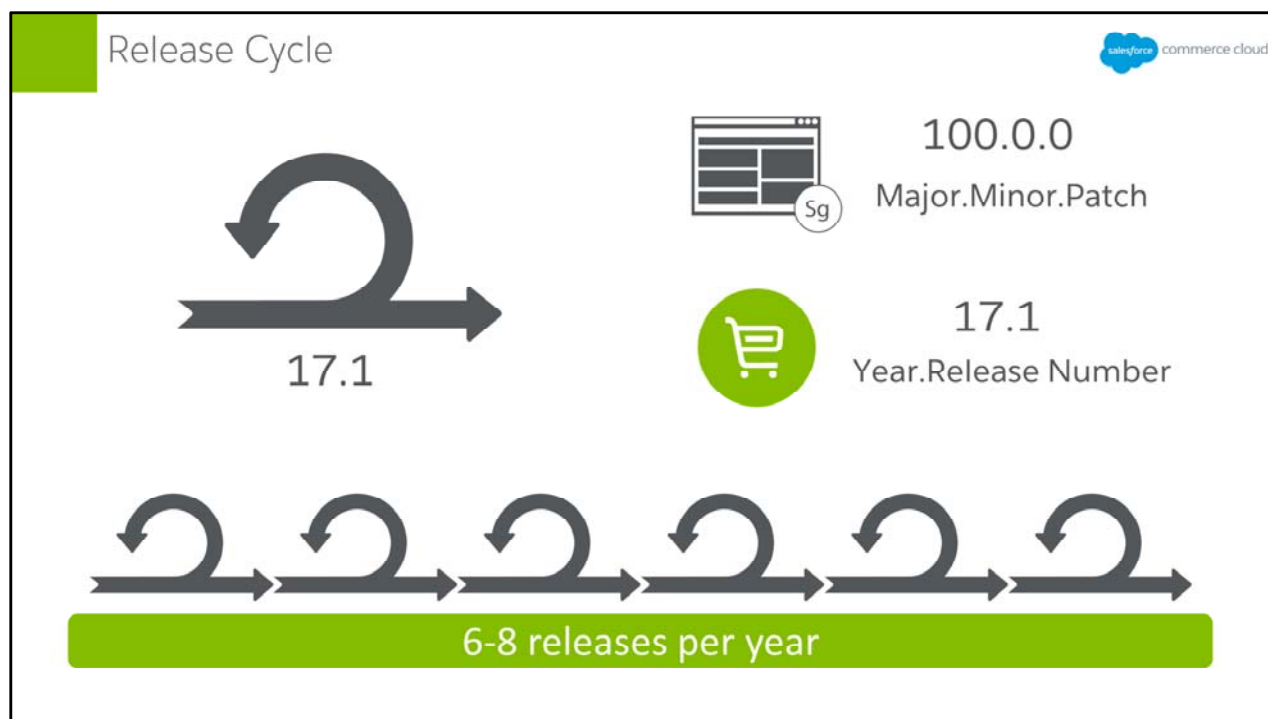




Commerce Cloud Digital has two layers:

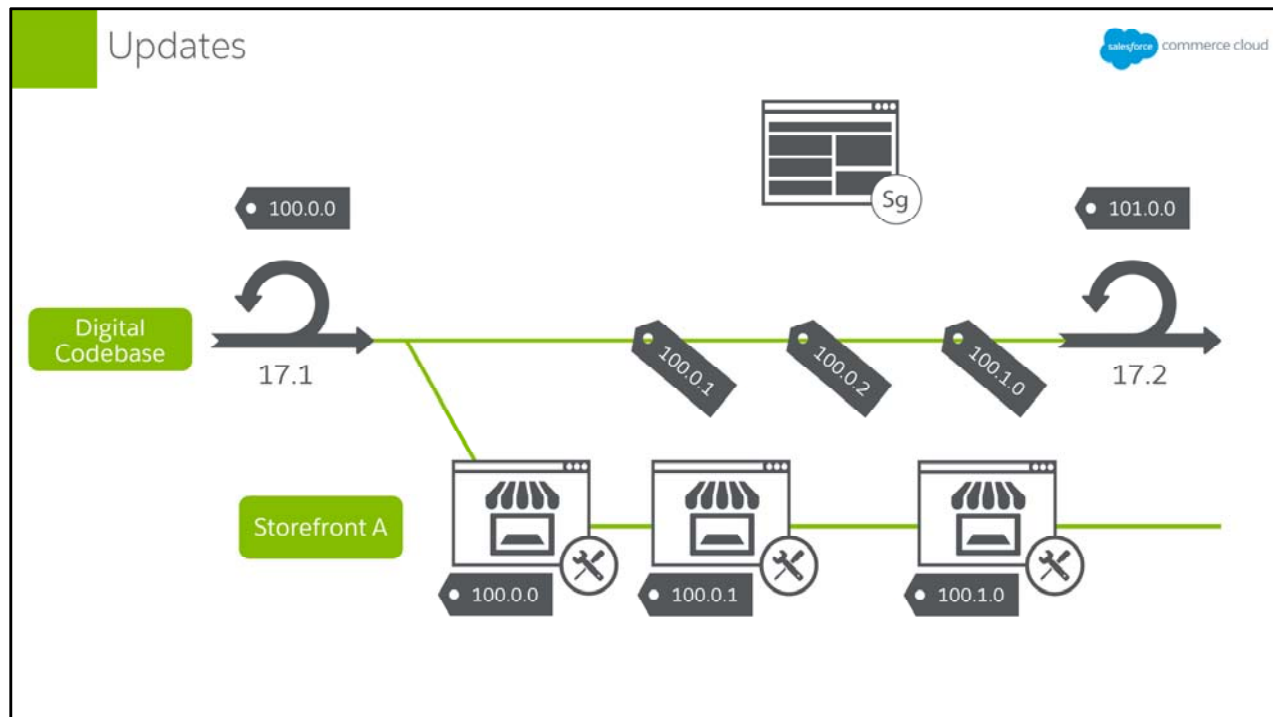
- The **core layer** contains the codebase that is the foundation of Commerce Cloud Digital. Commerce Cloud maintains the core layer and developers have no visibility inside this black box. All tenants operate on the same version, which is updated by Commerce Cloud six to eight times per year.
- The **custom layer** contains custom code that lays on top of the core layer. By selectively overriding core functionality, developers customize a storefront to provide a unique brand experience.

Working in an IDE, developers use APIs to access the core layer. To access core layer functionality from **within** a Digital instance, developers use the Commerce Cloud API. To access core layer functionality from **outside** a Digital instance, they use OCAPI.



Versions of Commerce Cloud Digital are identified by the two-digit year and the release number. For example, 17.1 represents the first release of 2017.

Versions of SiteGenesis are identified by semantic numbering, which is better suited to more frequent updates. SiteGenesis numbering follows this pattern: major release number, minor release number, patch number. In this scheme, a **major version** contains API changes that are incompatible with previous versions of SiteGenesis, a **minor version** contains new functionality that is backward-compatible, and a **patch** contains one or more bug fixes that are backward-compatible. SiteGenesis 100.0.0 was the first version that used this numbering scheme.



What happens when Commerce Cloud updates Digital? For each release of Digital, one version of SiteGenesis is tagged as the latest compatible version.

Let's suppose that Commerce Cloud Digital 17.1 is the current release. You use the corresponding SiteGenesis version 100.0.0 to create Storefront A. Over time, two SiteGenesis patches and one minor release become available. You review the release notes and decide to apply the first patch (SiteGenesis 100.0.1) but not the second patch (SiteGenesis 100.0.2). Later, you choose to implement the minor release (SiteGenesis 100.1.0). Finally, SiteGenesis undergoes a major update and version 101.0.0 is attached to the Digital 17.2 release.

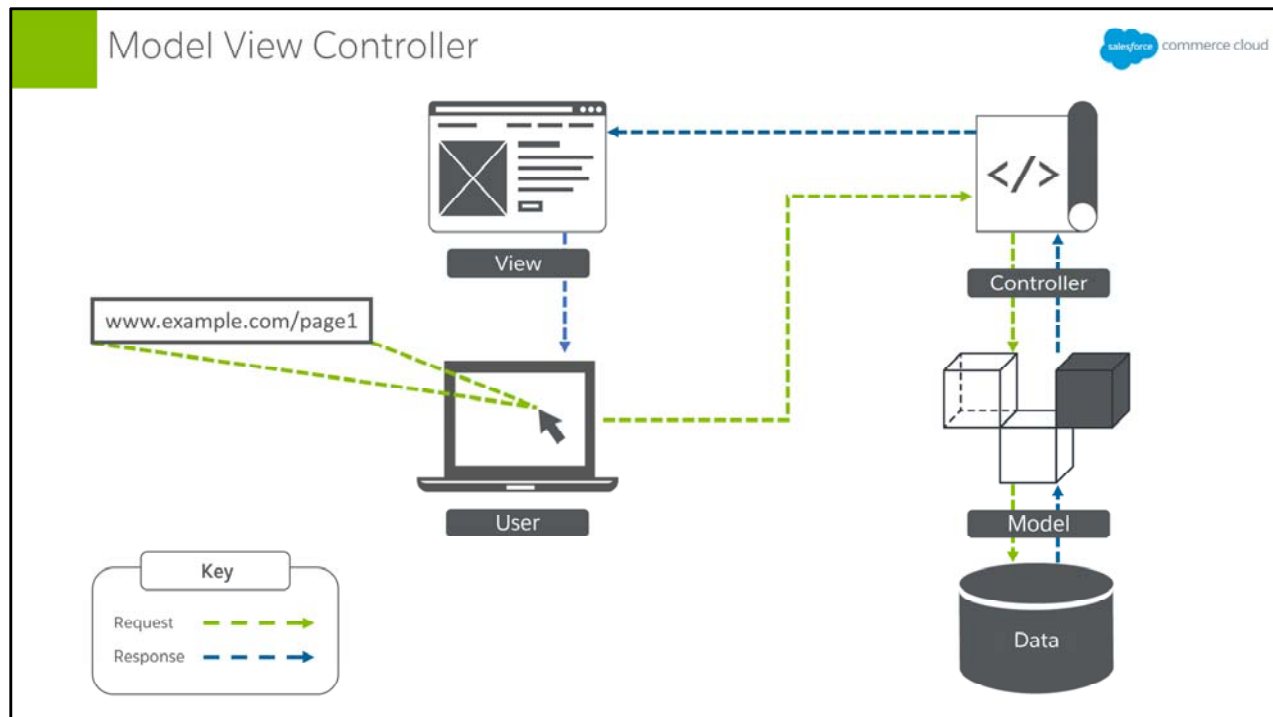
The 17.2 update to the core layer is pushed to all instances automatically, providing all clients the same backward-compatible new version. SiteGenesis version 101.0.0 is included with release 17.2, but it is not automatically applied. New SiteGenesis features are not reflected in Storefront A because Storefront A is still based on SiteGenesis 100.1.0. To accommodate the new features, you must modify the Storefront A code, including any affected custom code.



Within the custom layer, developers work with two types of storefront building blocks: pipelines and JavaScript controllers.

A **pipeline** is an XML file that defines a logical model of a business process. In the past, developers used pipelines to enact business processes for storefront sites. UX Studio provides a graphical interface for building, modifying, and visualizing pipelines as workflows.

In new storefront development projects, developers use **JavaScript controllers** instead of pipelines. A JavaScript controller is a server-side script that implements a storefront business process. Unlike pipelines, which are proprietary to Commerce Cloud, JavaScript controllers employ a common scripting language, allowing developers to work in an IDE other than UX Studio (except for code uploads).



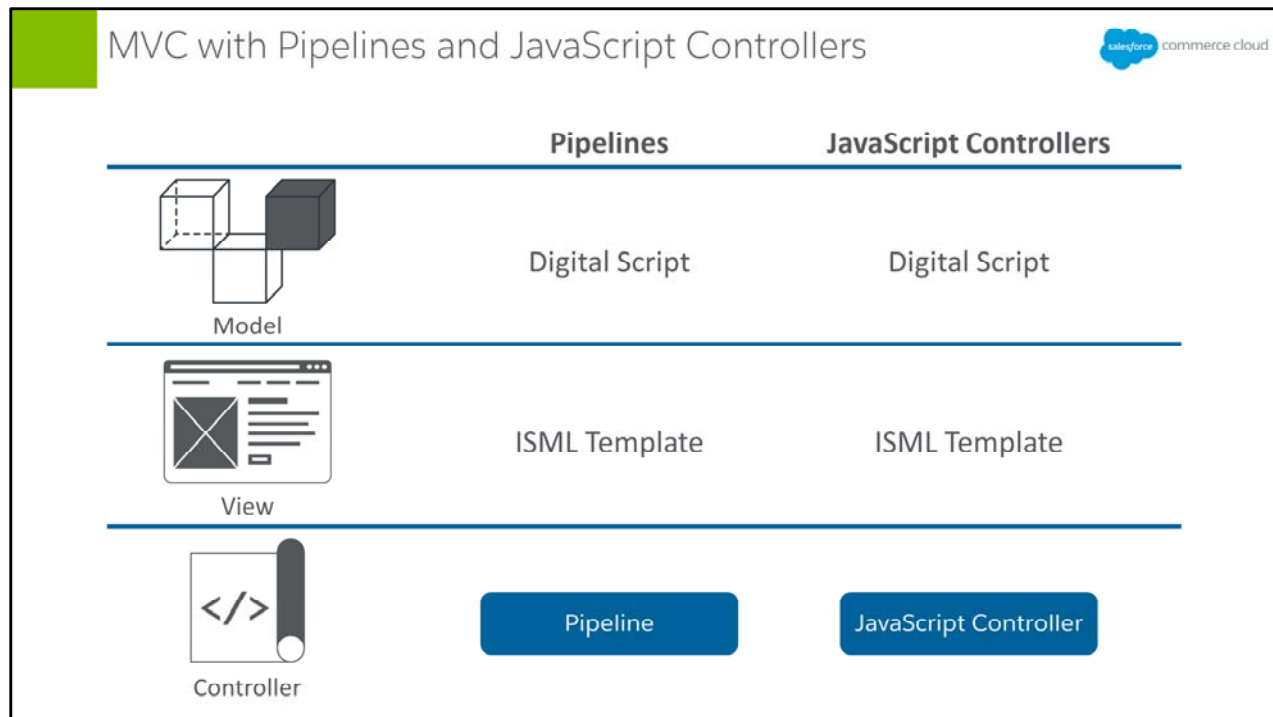
The widely-used Model View Controller (MVC) architectural pattern embodies a best practice for software development: separating business logic from the user interface. Each piece of the MVC pattern has unique responsibilities:

- The **model** manages the data, logic and rules of the application.
- The **view** displays output to the user based on the model data.
- The **controller** manages communication between the view and the model by receiving input, converting input to commands, and passing commands between the model and the view.

The MVC flow generally follows these steps:

1. A user requests a page by submitting its URL in a web browser. The browser sends the URL to the controller.
2. The controller commands the model to retrieve necessary data from the database.
3. The model retrieves and processes the data and then returns it to the controller.
4. The controller commands the view to update the web page with the new data.
5. The view renders the data in the requested page, which is then presented in the user's browser.

By isolating the business logic from the user interface, the modularity imposed by the MVC pattern enables parallel code development, supports code reuse, and simplifies code maintenance.



Let's look at how MVC applies to Commerce Cloud Digital.

Each piece of the MVC pattern corresponds to an element of the Digital storefront site:

- The **model** is represented by Digital Script. Digital Script is a proprietary, server-side variation of JavaScript.
- A **view** is represented by an ISML template. Internet Store Markup Language is an proprietary ecommerce-related HTML extension.
- A **controller** is represented by a pipeline or a JavaScript Controller, depending on the development model of the storefront site.

Strict adherence to the MVC pattern improves the performance and scalability of a storefront site.



## Next Steps

Now you know a bit about Digital architecture, the development environment, and how SiteGenesis relates to Digital. Next we discuss ways that you can expand a Digital storefront with integrations and customization. We also recommend strategies for developing high-performance code.



How can I expand my storefront functionality?

© Copyright 2017 salesforce.com, Inc. All rights reserved. Various trademarks held by their respective owners.





04

How can I expand my storefront functionality?



When you have completed this module, you should be able to answer these questions:

A

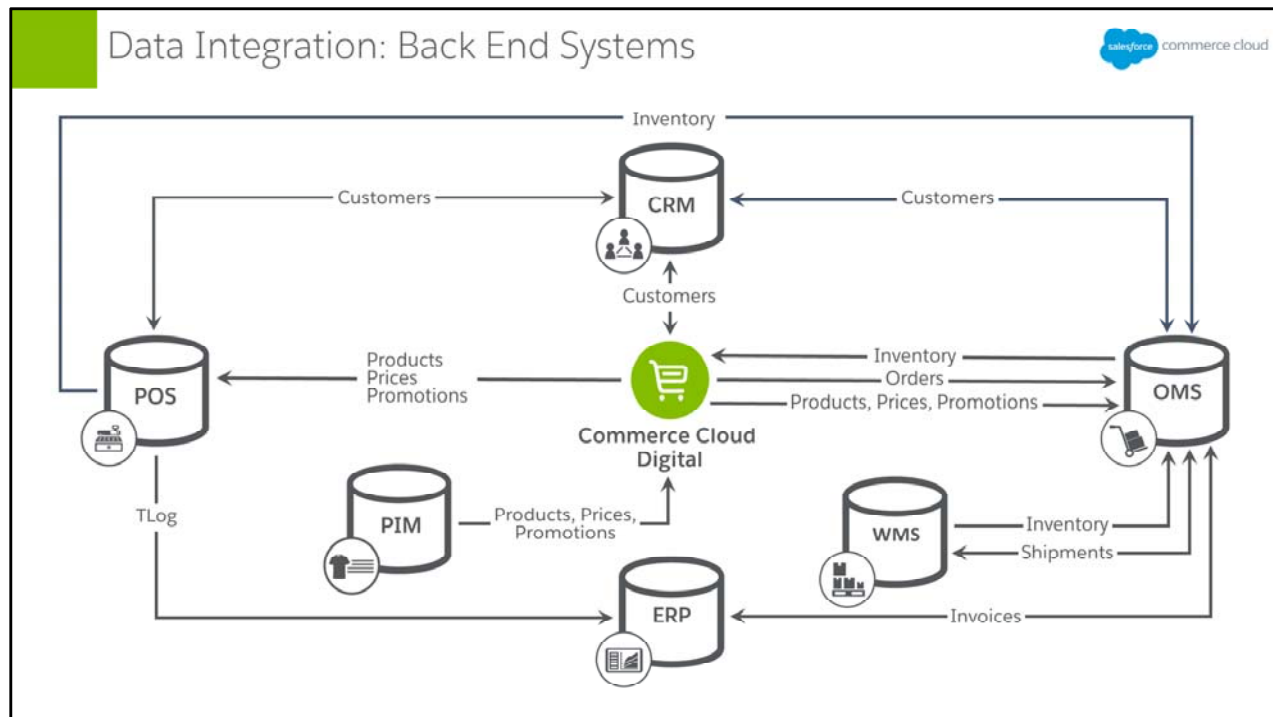
How are external systems and data integrated with my storefront?

B

How is storefront functionality organized?

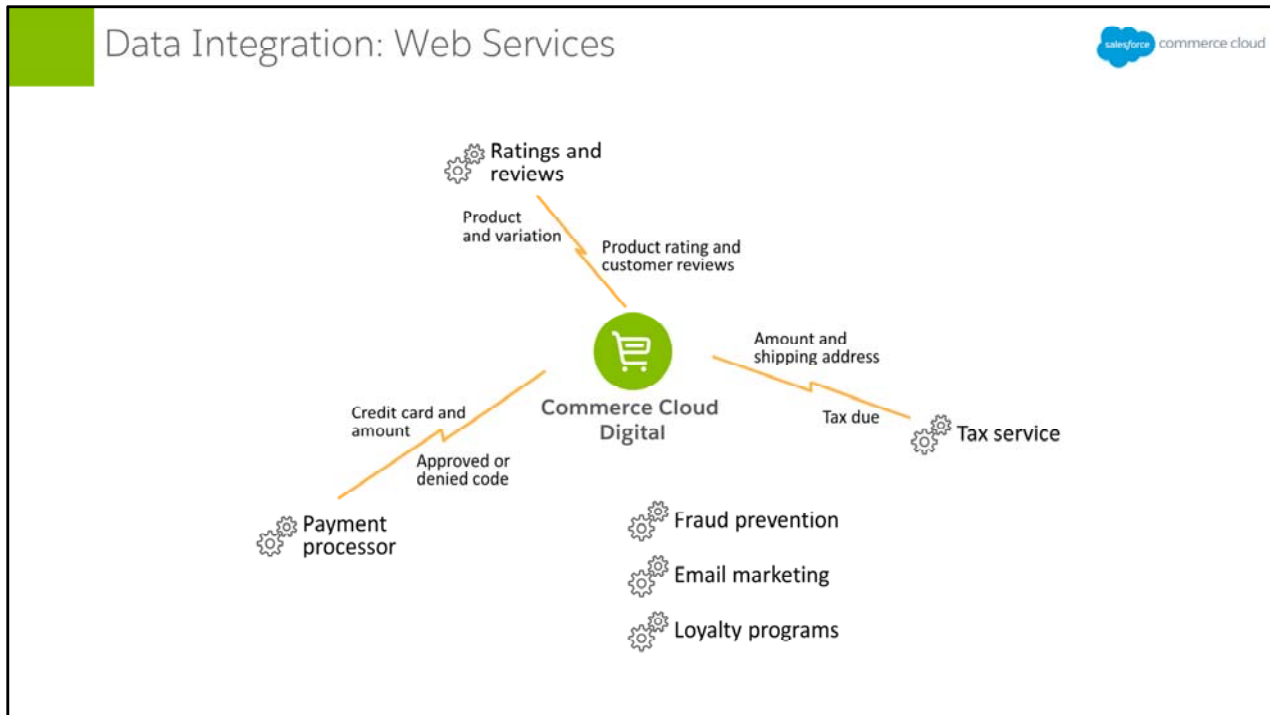
C

How can I develop code that performs well and supports increased storefront traffic as needed?



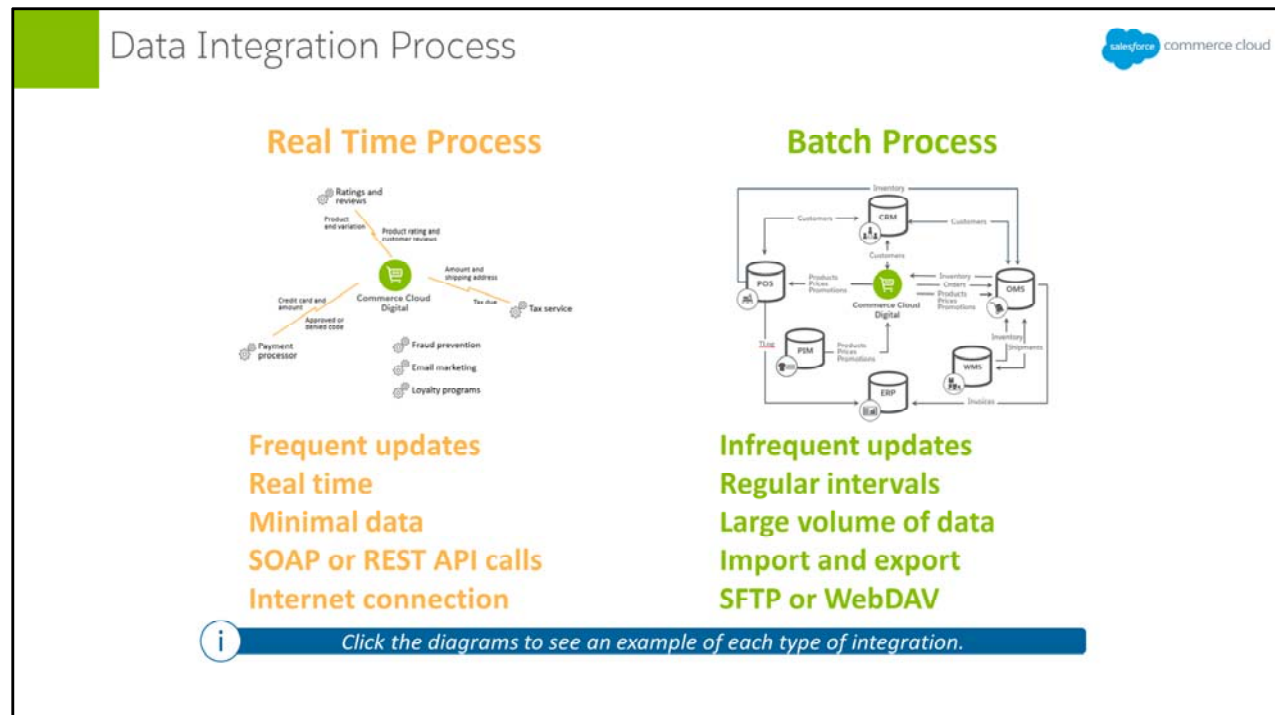
Commerce Cloud Digital is not designed to stand alone. There is undoubtedly data in other systems that must interact with your Digital storefront site. For example, most product and marketing information is stored in a product information management (PIM) system, and customer data usually resides in a customer relationship management (CRM) system.

Other common back end systems include point of sale (POS), order management (OMS), warehouse management (WMS), and enterprise resource planning (ERP) systems.



Most storefront sites also use an external payment processor, product review provider, and tax service.

They might also use a variety of other external services, such as fraud prevention, email marketing, and loyalty programs.

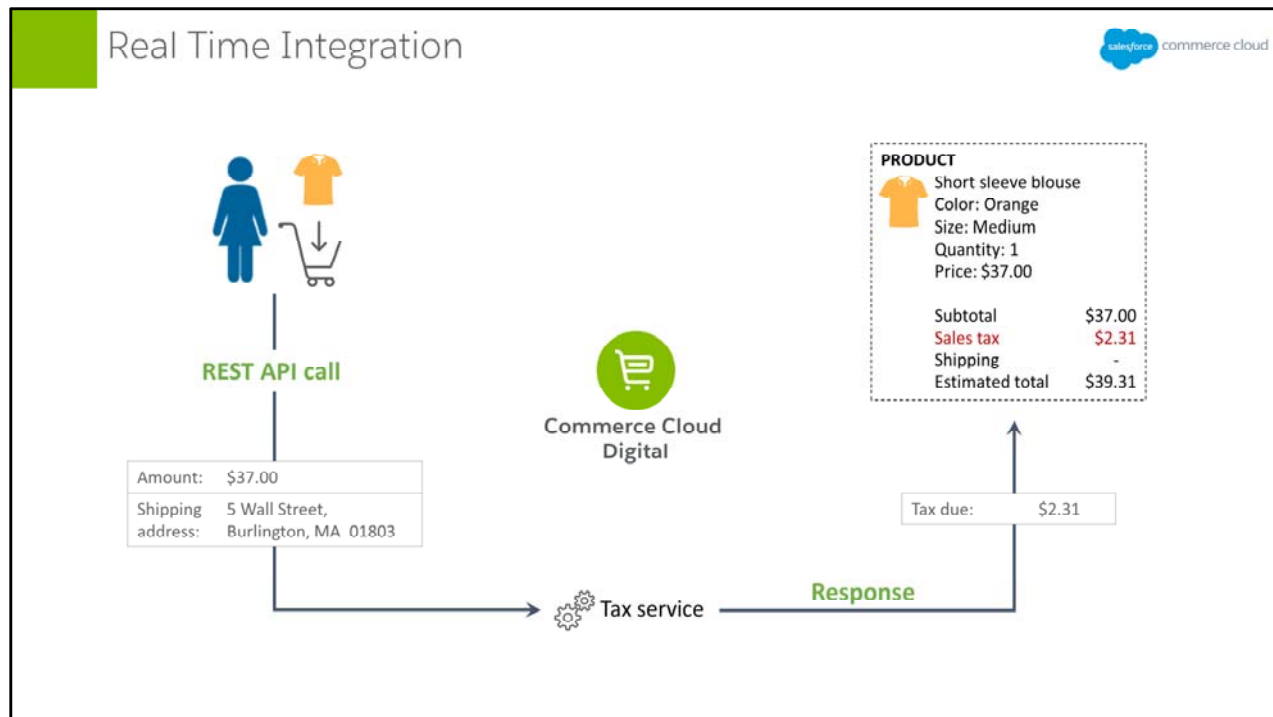


Data is integrated with a Digital storefront site either in real time or as a batch process.

Real time integration is employed when small amounts of data must be updated immediately and frequently. It is implemented with either SOAP or REST API calls to a web service.

Batch process integration is employed when an external system provides a large volume of data to a storefront infrequently. It is implemented with (Business Manager) import and export jobs and a data transfer protocol (FTP, SFTP, HTTP, HTTPS, or WebDAV). After the initial mass data transfer, integration is accomplished by regular delta feeds. The batch process is typically automated with the Job Schedules module in the Business Manager.

Click the diagrams to see an example of each type of integration.



An example of **real time integration** is a web service providing tax calculations.

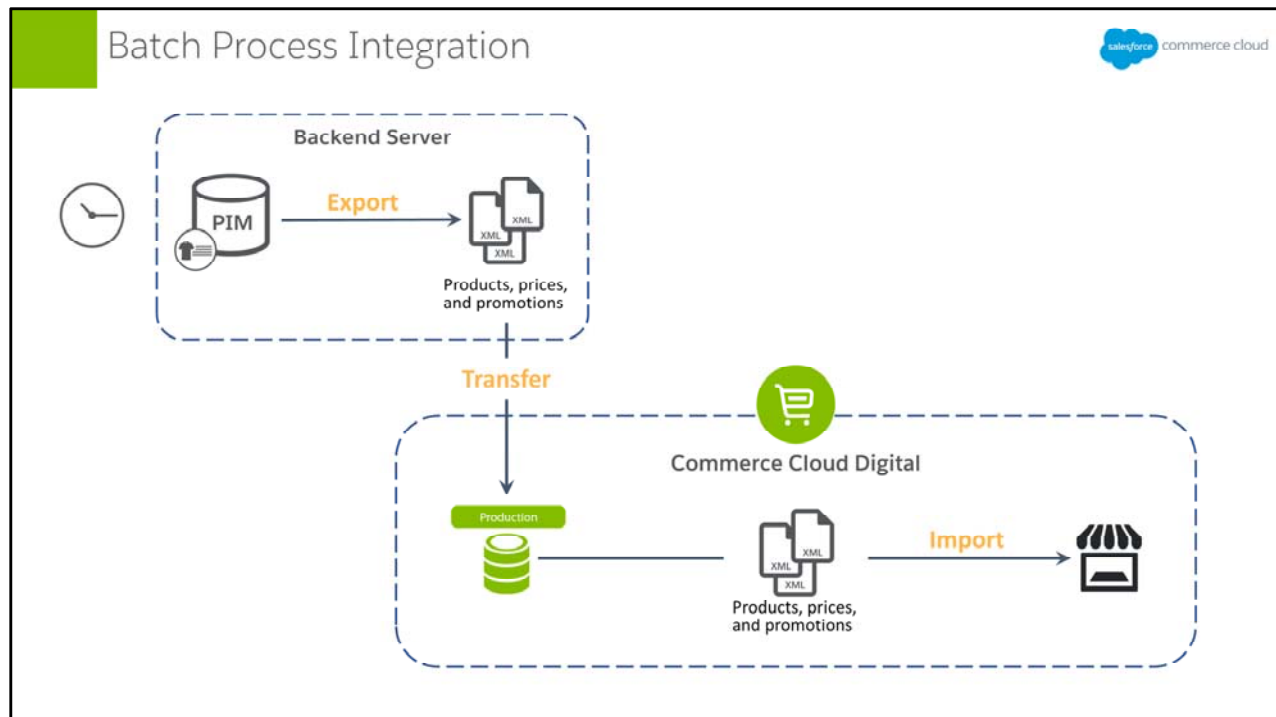
When a customer purchases a product, the storefront sends transaction data to a tax service provider. The service immediately calculates and returns the amount of tax due. The sales tax is then displayed in the storefront.

This example illustrates the general flow of real time integration with a web service. Understanding the specific integration requirements for each external service involves some research.

What type of API is in use?

Which API calls are supported?

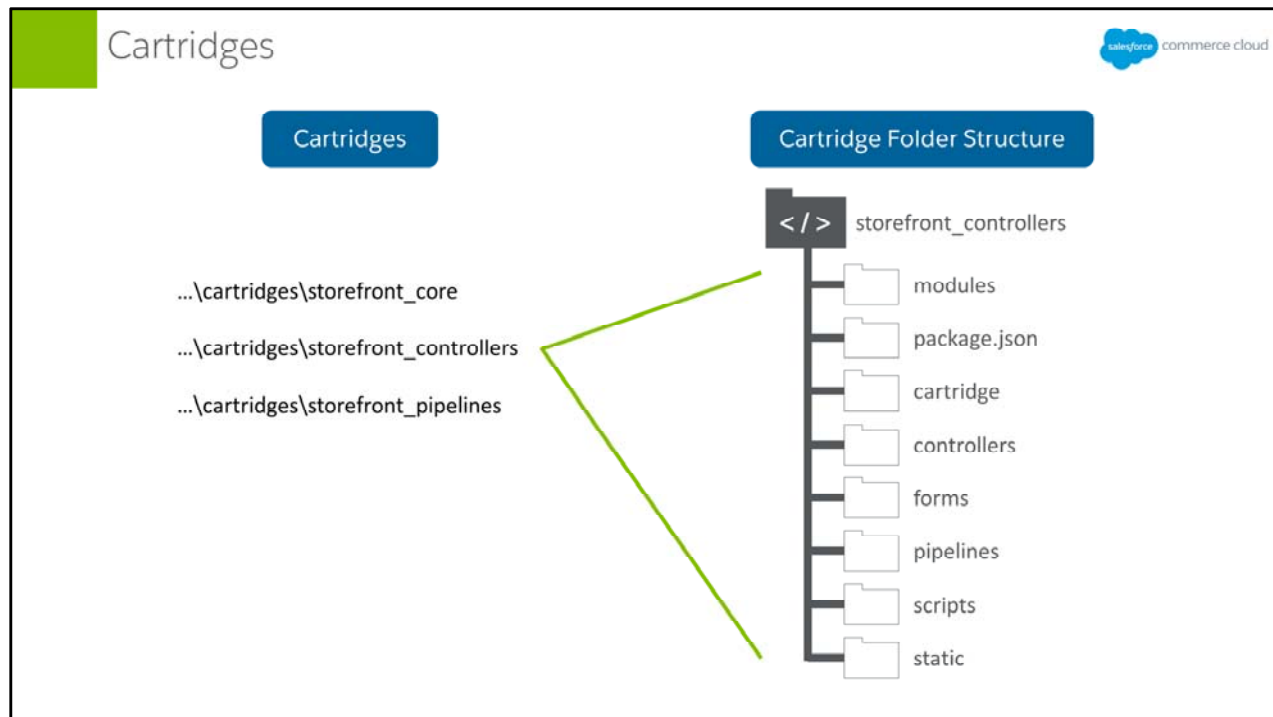
What parameters are required in a request?



An example of **batch process integration** is a nightly update to the storefront catalog with data from a PIM.

At the scheduled time, data is exported from the PIM, transferred to a Digital instance, and then imported into the storefront site.

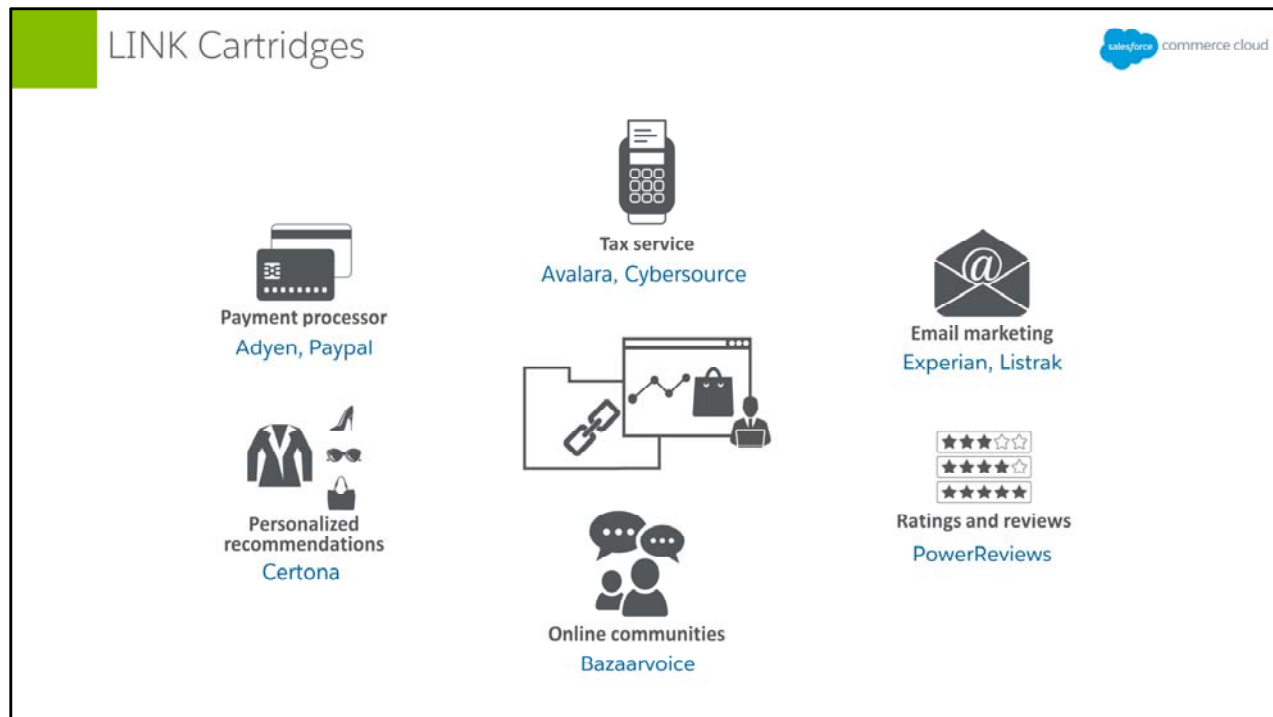
Now that we've examined how external services augment the data in a Digital instance, let's look at how additional code expands the capabilities of a Digital storefront.



Digital storefront functionality is organized by cartridges. The storefront\_core, storefront\_controllers, and storefront\_pipelines cartridges define core functionality, functionality implemented with controllers, and functionality implemented with pipelines, respectively.

A cartridge is a folder containing a set of resources that provide specific storefront features or an external integration. Every cartridge is fully contained within one folder. Each cartridge folder has specific subfolders for different types of resources, including pipelines, templates, scripts, forms, and static content, such as image files, CSS files, and client-side JavaScript files.

You can introduce new functionality to a storefront by adding new cartridges. You might add new functionality to customize the storefront experience or to integrate with an external service.

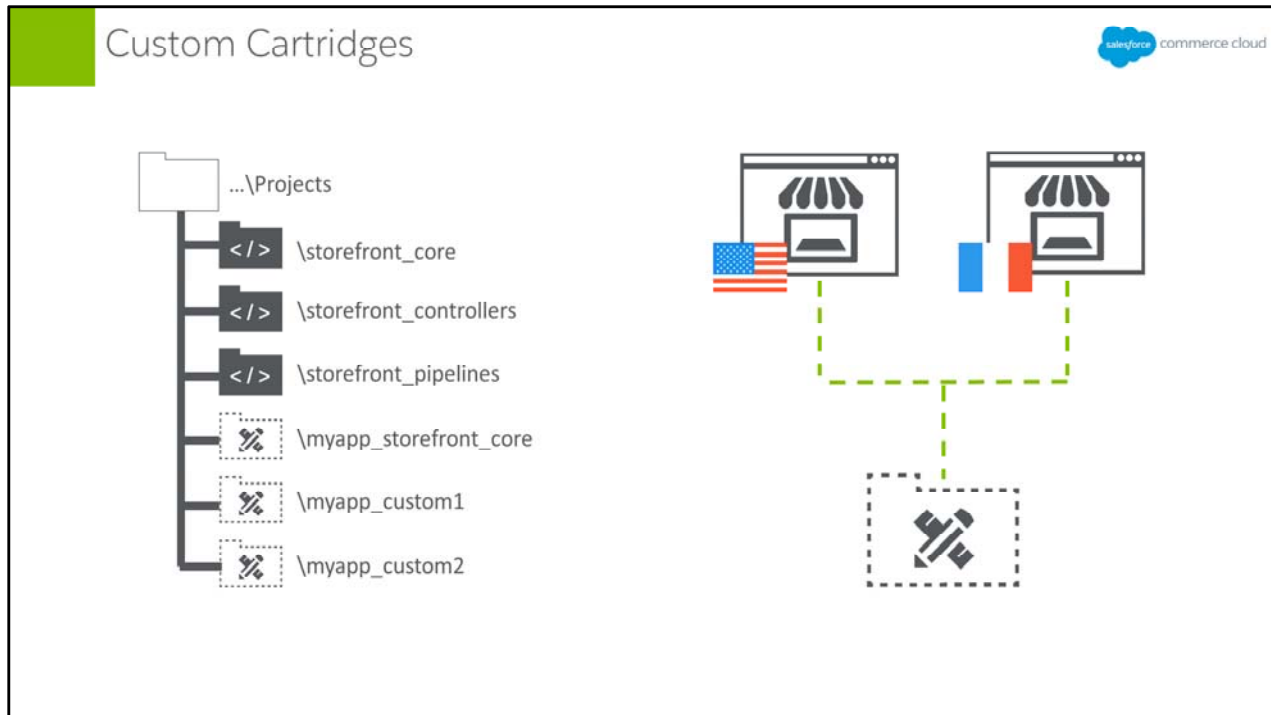


Developed by Commerce Cloud certified LINK partners, LINK cartridges are predefined Digital cartridges that integrate external web services with a storefront. Many LINK cartridges are available in the Commerce Cloud LINK Marketplace. Examples include:

- Payment processors, such as Adyen and PayPal
- Tax service, such as Avalara and Cybersource
- Email marketing, such as Experian and Listrak
- Ratings and reviews, such as PowerReviews
- Online communities, such as Bazaarvoice
- Personalized recommendations, such as Certona

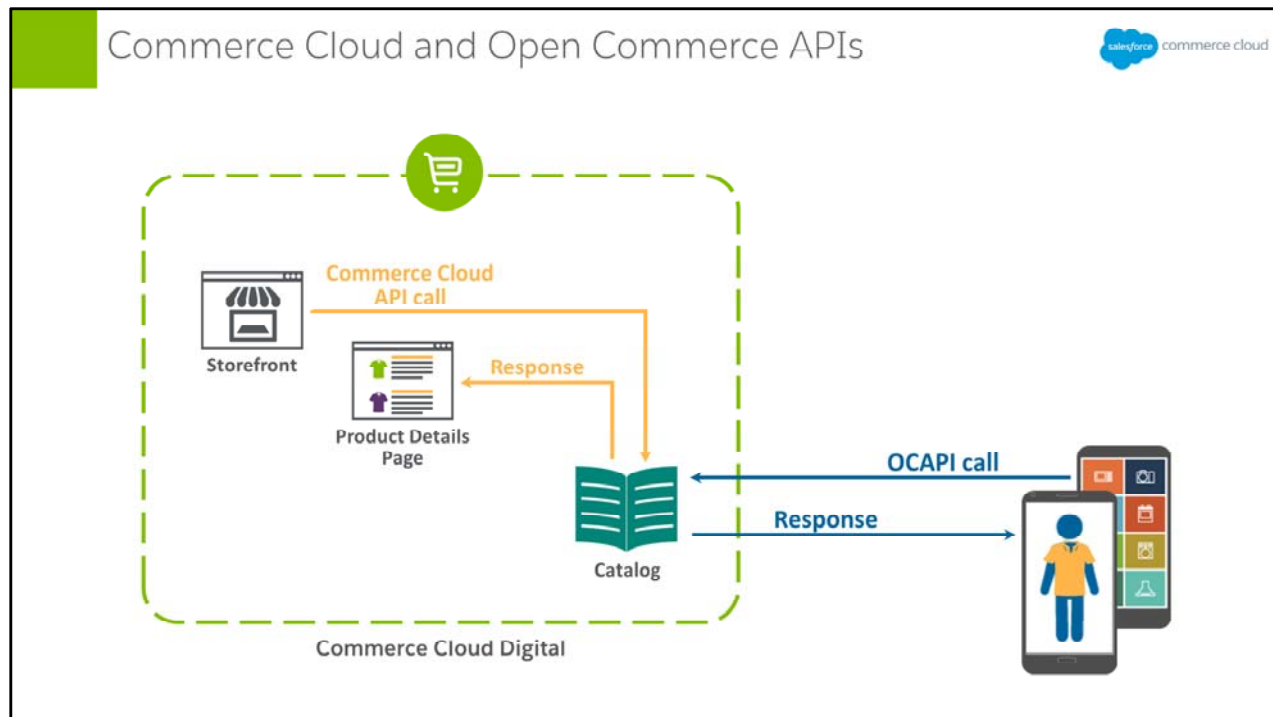
The functionality provided by a LINK cartridge is enabled by its configuration settings in the Business Manager.





When your specific needs are not addressed by a LINK cartridge, you can develop your own custom cartridge. Developers create their own cartridges for two reasons: to write custom code that supplements or customizes existing storefront functionality, and to write API calls that integrate external web services with a storefront.

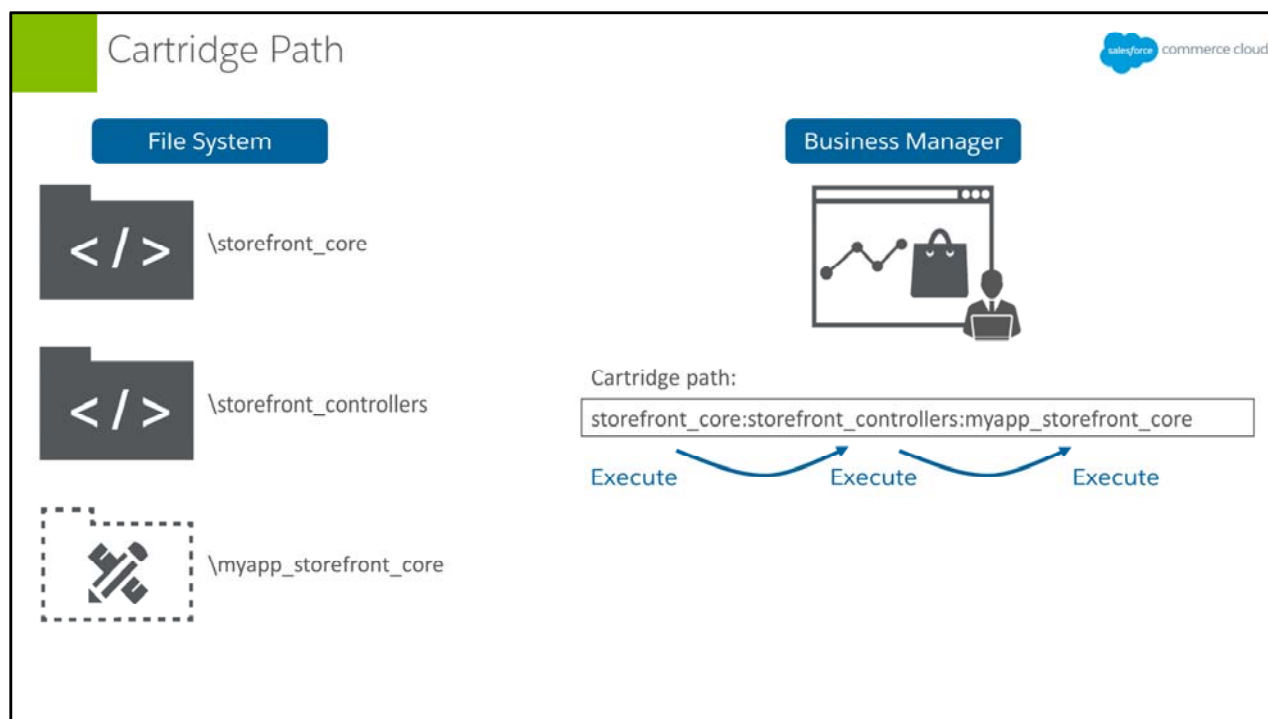
Developers also create discrete cartridges for code that can be reused by other sites within a production instance, and for resources that are used by a language-specific site.



Custom code uses either the Commerce Cloud API or the Open Commerce API (OCAPI).

Developers access core layer functionality from within a Digital instance by using the Commerce Cloud API. For example, you might have a custom product attribute that you want to display on the product details page. Use the Commerce Cloud API to access the product attribute from the product details page within the storefront.

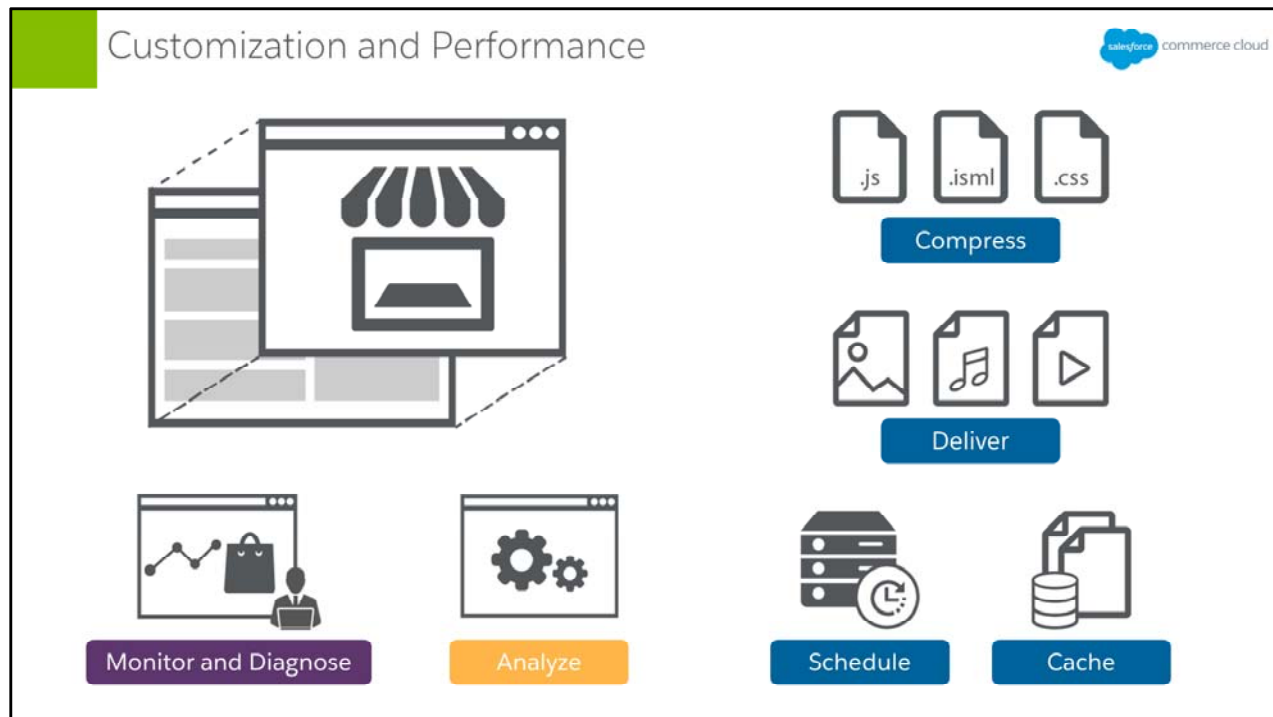
To access core layer functionality from outside a Digital instance, developers use OCAPI. For example, you might create a mobile phone app that gets the product image of a shirt, and uses the phone's camera to show how the customer would look wearing the shirt. Use OCAPI to access the product image from the app outside the storefront.



After you add a cartridge to your storefront, use the Business Manager to update the cartridge path so that the storefront recognizes the new cartridge.

The cartridge path is a colon-separated, ordered list of cartridges that defines code execution precedence. Cartridges at the beginning of the cartridge path take precedence over cartridges at the end of the cartridge path.

When the storefront executes code, it follows the cartridge path. To override core functionality, position a custom cartridge earlier in the cartridge path than the cartridge that you want to override.



During the early phases of storefront development, it is important to define strategies that enhance site scalability and performance.

Recommendations include:

- Compress CSS, JavaScript, and ISML.
- Identify the most efficient way to deliver multimedia content (image, audio, and video files).
- To avoid slowing storefront site performance, schedule jobs to occur outside heavy traffic hours.
- Minimize the number of calls to the database layer by using the page cache, and,
- When establishing integrations, be sure to:
  - Use the caching capabilities that are supported by the API in use.
  - Set web service timeouts.
  - Account for unresponsive services.

Maintaining a high-performance storefront requires monitoring performance and making adjustments when performance slows. The Business Manager offers analytics, custom logs, and a profiler to help with monitoring and diagnosing problems.

The Control Center provides statistical graphs to help with monitoring and understanding a storefront site's overall resource consumption, including connections and network utilization, data usage, file system usage, and processing time.



## Course Summary



*Having completed this course, you should now be able to answer these questions:*

- What does it mean to work in a software-as-a-service (SaaS) environment?
- How do I develop a storefront site?
- How are Commerce Cloud Digital and SiteGenesis related?
- How can I expand my storefront functionality?



## Congratulations!

You have completed *Commerce Cloud Digital Architecture Overview*.

Your satisfaction is very important to us. Click the **Course Survey** link emailed to you at the end of this course.