A Practical Roadmap for Moving to Sitecore XM Cloud

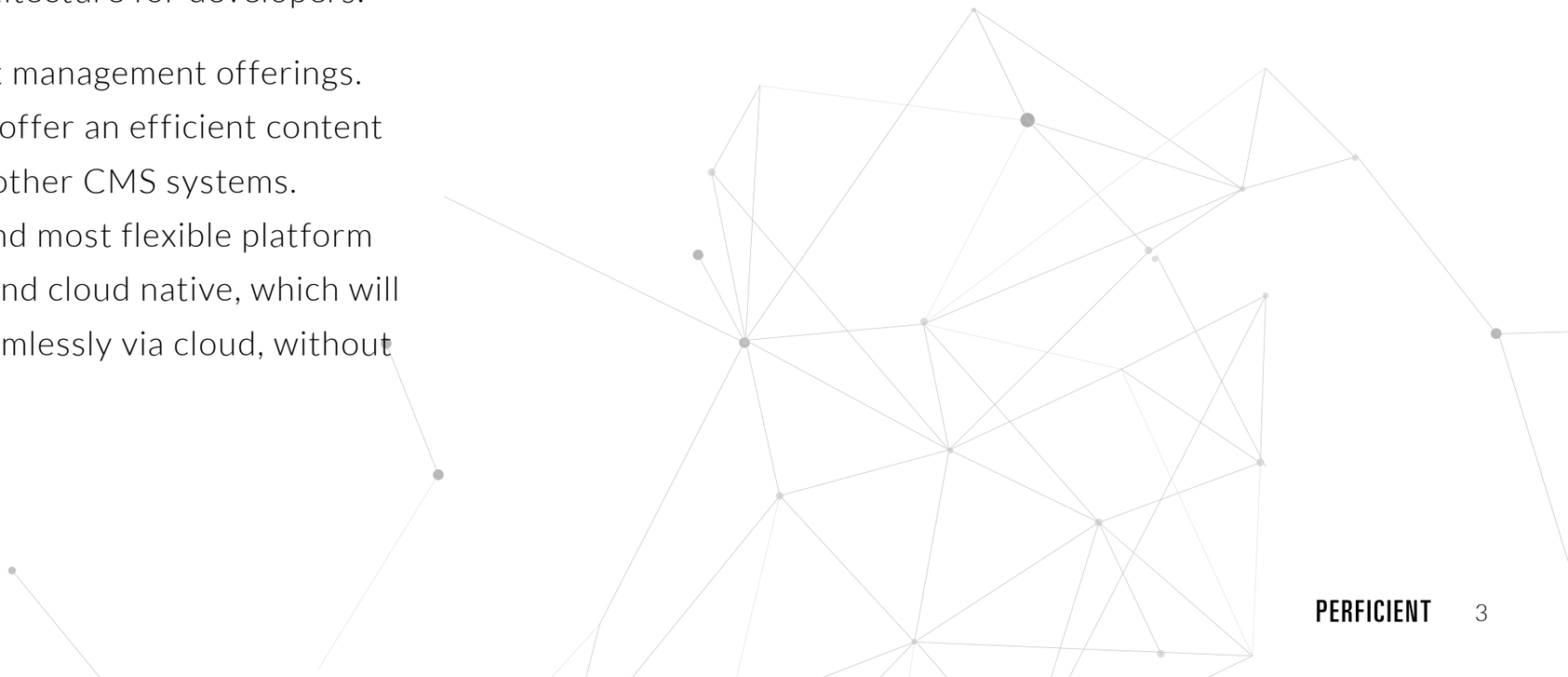
PERFICIENT®

Contents

What Is Sitecore XM Cloud?

Sitecore Experience Manager Cloud (XM Cloud) is a fully managed, self-service platform for developers and marketers to launch engaging omnichannel experiences using Sitecore's headless content management system (CMS). It's a cloud-native SaaS and embeds cutting-edge analytics and personalization technology from Sitecore CDP and Personalize. It also has market-leading visual authoring tools for marketers and an unrivalled modern architecture for developers.

XM Cloud is the future of enterprise content management offerings. The new sites, pages, and components tools offer an efficient content author experience that is not available with other CMS systems. With built-in analytics, you get the fastest and most flexible platform available today. XM Cloud is also fully SaaS and cloud native, which will provide even more capabilities over time seamlessly via cloud, without ever needing to manually upgrade.



Planning a Roadmap to Adopt XM Cloud

Existing Sitecore customers that want to adopt XM Cloud may feel overwhelmed at first. If you intend to build a new site from scratch or have plans to redesign and reimplement, starting a greenfield implementation is relatively straight forward. If you have significant investment in Sitecore XP, MVC, or ASP.Net, however, you may not have the desire or budget to rebuild from scratch, at least not at once.

To adopt XM Cloud, you'll need careful planning and execution. Most importantly, you'll need a roadmap.

This guide lays out five steps to building that roadmap.

Step 1: Determine Your Platform Migration Strategy

XM Cloud needs to be implemented using Sitecore Headless Services, meaning that existing MVC sites need to be rewritten to leverage this new architecture.

The first decision you need to make is whether to migrate your solution to headless on your existing platform or jump straight to XM Cloud as you move to headless. The answer depends on several factors including:

Current Sitecore Version: While there are older versions of JSS that support Sitecore 9, to establish a solid foundation and leverage Next.js and headless SXA, it's necessary to be on at least Sitecore 10.1. If you're on a very old version of Sitecore, it is worth considering whether the trouble of an upgrade is worth the effort, or whether you would be better off investing directly in moving to headless in a new environment.

Business Priorities: Migrating a solution to headless can take several months to complete. Are there other business priorities that cannot be put on hold during this migration? If so, migrating gradually is a better approach.

Solution Complexity: More-complex solutions take longer to rebuild in a headless fashion and can be more difficult to upgrade. Both factors need to be considered when determining your XM Cloud migration approach.

Approach 1: Upgrade, Move to Headless, Move to XM Cloud

The first migration approach involves upgrading your instance of Sitecore, moving your solution to a headless architecture, and then migrating your solution to XM Cloud.

Advantages:

- Allows you to take your time with the migration. There is no time pressure due to maintaining multiple infrastructures or licensing implications due to not completing the project on time.
- Allows you to wait for XM Cloud to mature. XM Cloud has improved and, because it's a SaaS product, it will improve over time. When you move to XM Cloud after migrating to headless, there may be new features that help with the migration or capabilities you can leverage.

Disadvantages:

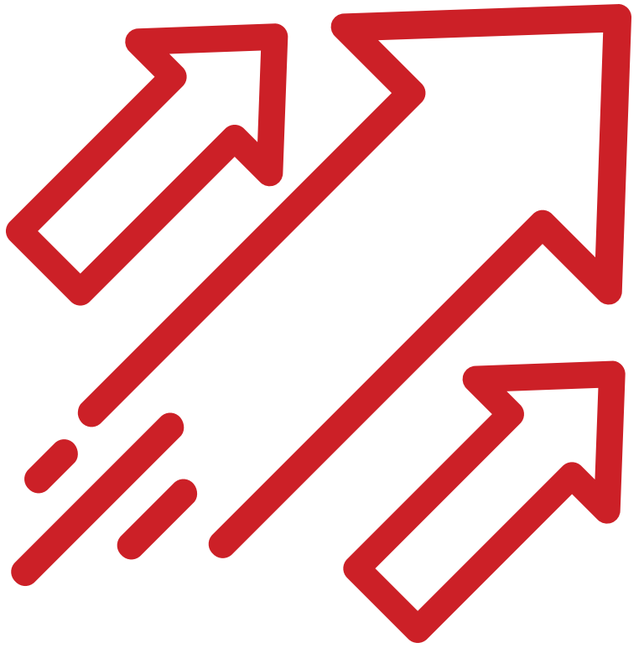
- Sitecore upgrades, especially from an older version, can be challenging. You will need to consider the use of WFFM forms, the introduction of Identity Server and depreciation of the AD module, compatibility issues with other modules, and dependencies like Glass Mapper, depreciation of Microsoft Azure Search, Migration of xDB data, and more.
- Without time or cost pressure, it may be difficult to prioritize moving everything to headless, or moving to XM Cloud, leaving you with a bifurcated site that is more difficult to maintain.
- Effort spent upgrading your solution could be focused on moving your solution to a headless architecture.
- This approach usually takes longer and is more expensive in the long run.

Upgrade Considerations

If you choose this route, consider whether to downgrade from XP to XM as part of the upgrade. To properly move to XM Cloud, all dependencies on xDB and XP's marketing features need to be removed since it is not supported in XM Cloud. If these dependencies are used minimally, remove them as part of the upgrade. If the dependencies are more deeply rooted, upgrade to Sitecore XP and remove the dependencies during the feature migration.

Approach 1 Variation: Don't Upgrade, Stand Up New XM Environment

Instead of upgrading, you could stand up a new Sitecore XM 10.3 environment and rebuild your solution there directly. While this is possible, you will need to deal with additional hosting and support costs for dual environments, so make sure your Sitecore license can support the approach. Dual environments make sense if the headless migration is constrained to a fixed, shortened timeline.



Approach 2: Rebuild Using Headless Directly on XM Cloud

Another approach is to skip the upgrade and migrate your solution to headless on XM Cloud directly.

Advantages:

- This approach doesn't waste any resources upgrading your existing solution.
- This approach is usually faster and gives you access to new XM Cloud features, including personalization, analytics, components, and new content editing features with Sitecore Sites & Pages.

Disadvantages:

- There will be licensing, and infrastructure costs related to maintaining the solutions in parallel.
- The longer the migration takes, the more it costs, putting pressure on the organization to complete the move. This can deprioritize other business initiatives for the duration of the migration.

Big Bang Migration vs Gradual Migration

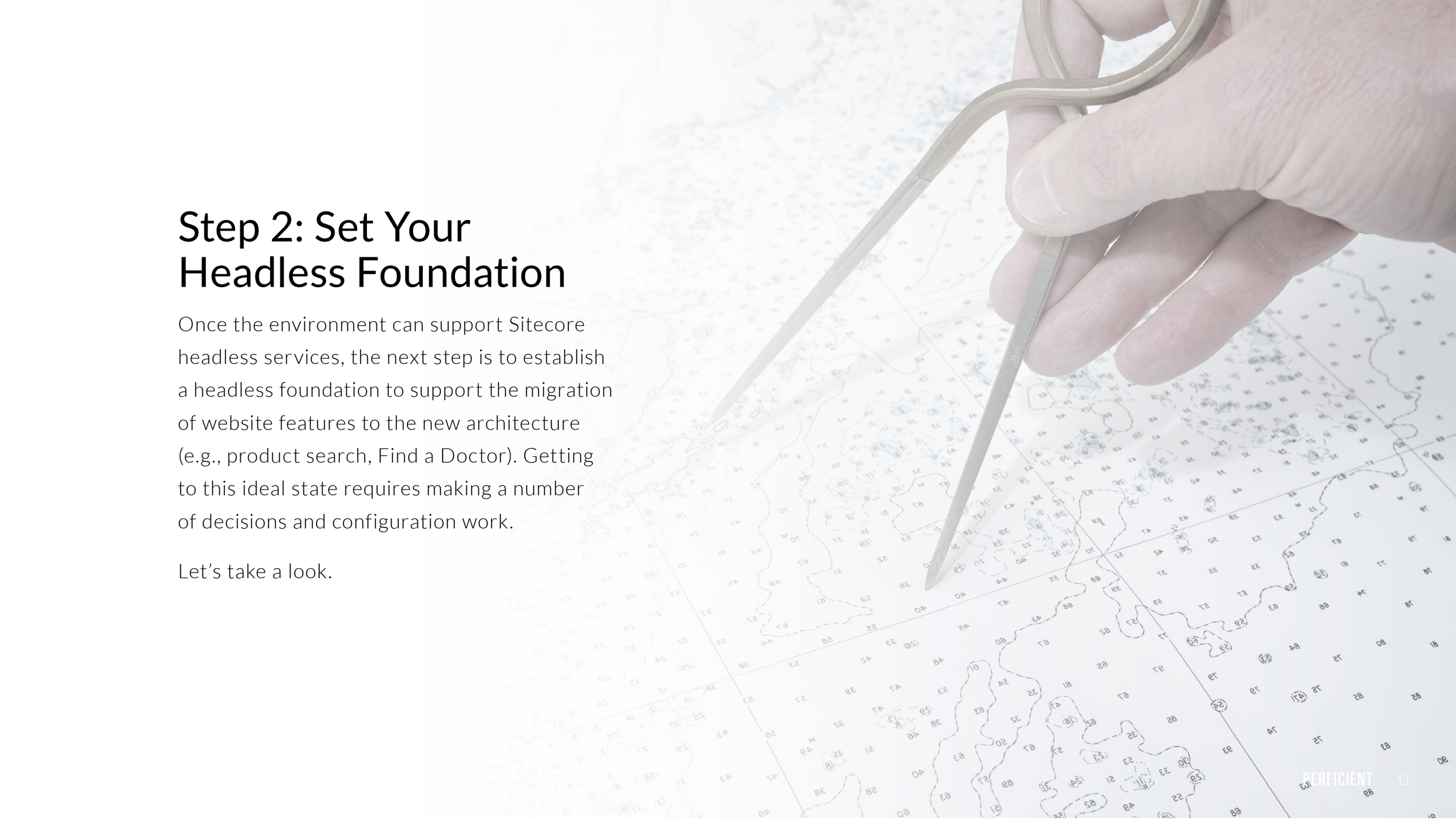
Besides the platform approach, examine whether you are looking to migrate your solution in one step, moving all features and pages to headless and bringing over the entire solution or whether you want to take a more gradual approach.

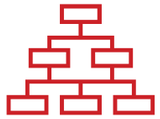
Instead of a single step, you could set up your platform to allow the migration of features one at a time, moving functionality to headless based on priority and business need. There may be additional setup needed to support this approach, but it will allow you to be more responsive to business priorities during the migration.

Step 2: Set Your Headless Foundation

Once the environment can support Sitecore headless services, the next step is to establish a headless foundation to support the migration of website features to the new architecture (e.g., product search, Find a Doctor). Getting to this ideal state requires making a number of decisions and configuration work.

Let’s take a look.





Select a Framework

Sitecore headless services support several front-end frameworks including React, Angular, Vue, .NET Core, and Next.js. We recommend using Next.js for its runtime static generation capabilities that help improve performance. Sitecore is investing more into this framework as well, building in support for headless SXA . Leverage headless SXA features for their automation, page design, and component variant support.

Select a Rendering Host

Your front-end application needs to be hosted somewhere. The only requirement is supporting Node.js, so there are numerous options available. If your goal is high performance, select a serverless hosting platform that supports edge functions and static generation. Vercel and Netlify are popular options, and Sitecore's partnership with Vercel, and ownership of Next.js, make this a logical option. You can choose to self-host to save costs, but remember to account for additional DevOps, maintenance, and operational costs.

Choose a Delivery Strategy

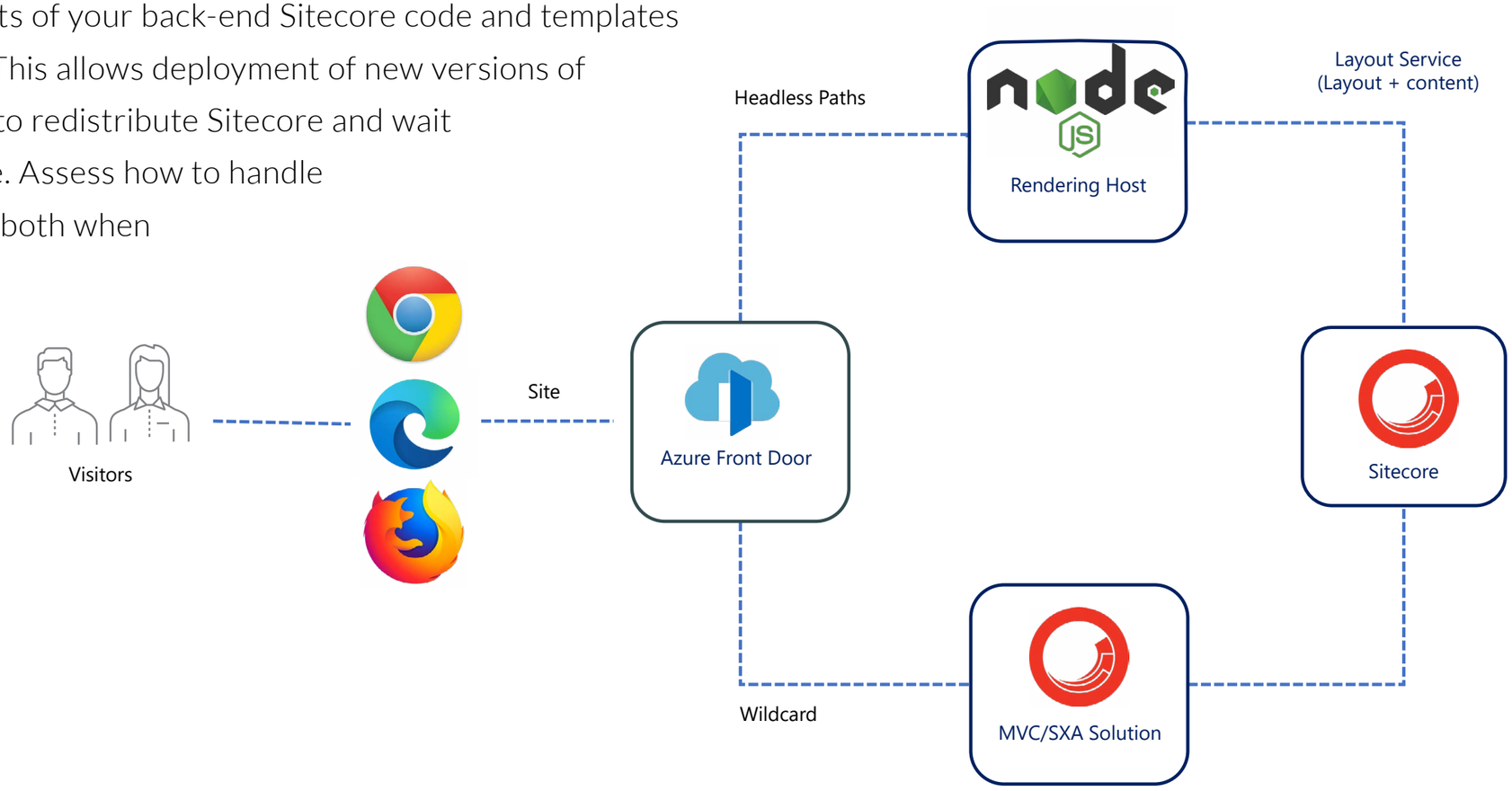
After installing Sitecore headless services, you'll have access to Sitecore's layout service and graphQL API. By default, the front-end application will receive content and layout details using the services that are hosted on your CD servers. An alternative approach is to utilize Sitecore's Experience Edge to seamlessly publish content and layout details to Edge, which makes content globally available and performant. This option requires additional subscription costs but eliminates all dependencies on CD servers to better scale your infrastructure. It also requires users to think through the architectural implications of not having CD servers. This is required when moving to XM Cloud so doing it now will make this process much easier when you move to XM Cloud later.

Configure a Reverse Proxy

If you do not intend to rebuild entire sites in one step, leverage a reverse proxy, such as Microsoft Azure Front Door, to direct traffic between your existing MVC site and newly rebuilt headless features based on URL patterns. Including this as part of the headless foundation will allow you to migrate features incrementally without the pain of redirects, which can impact SEO rankings.

Create DevOps Processes

Next is to create DevOps processes that fully support automated deployments. Having a separate front-end application provides additional flexibility and considerations that are unavailable with an MVC-based site. Once you create DevOps processes, you can now separate deployments of your back-end Sitecore code and templates from your front-end application. This allows deployment of new versions of the application without the need to redistribute Sitecore and wait for CM and CD servers to recycle. Assess how to handle coordinating deployments across both when updates are needed.



Step 3: Create a Prioritized Backlog of Features to Migrate

Once a solid headless foundation is in place, put together a prioritized backlog of features to migrate. There are several things to consider when crafting your plan.

Highly Visible and Impactful Features

To maintain the support of stakeholders, it's important to demonstrate the value of the project by gradually migrating features to headless. Choose highly visible pages and features that will demonstrate to the organization that the investment in headless is a wise one.

Foundational Features

Be sure to prioritize feature migration accordingly. It's important to understand the dependencies between multiple features and how moving one feature can impact another.
A highly dependent feature that's commonly migrated is site search. For example, when migrating content from an MVC site to a headless site in the content tree, account for having a search feature that will provide results from both the migrated and non-migrated content trees. Prioritizing an early site search migration to enhance and support unified search results is common.

Features With Planned Enhancements

If there are existing backlog items that will lead to significant enhancements to existing features, prioritize those features for a headless rebuild to save yourself from having to implement them twice. It also demonstrates responsiveness to business stakeholders and helps build organizational goodwill.

Features That Benefit From a Performance Boost

A great way to display the benefits of a headless architecture is to prioritize the rebuild of poorly performing site areas. Collect performance metrics of the MVC version of the feature and for the headless version. Also, take lighthouse scores of the before and after and use this data to promote the migration effort.

Additional Architecture Considerations

As you prioritize your migration roadmap, there are several architectural considerations to assess:

Content Reuse and Management

Dependencies involve content. If a feature is migrated and other non-migrated pages refer to this content, consider how it will be managed and check that the content authors understand the impacts or limitations.

Global Component Library

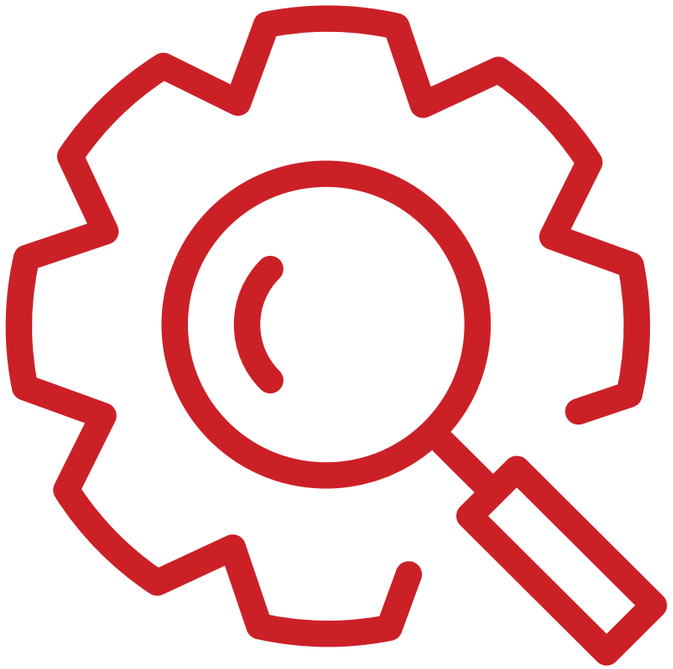
As you build out new headless templates and components, maintain a global component library to ensure proper documentation and long-term governance of the solution. Create a pattern library site to document available components or leverage component and maintenance tools to avoid component and maintenance issues.

Controller and API Use

When moving MVC features to headless and Next.js, keep in mind that with XM Cloud, there are no CD servers that host custom API endpoints. While some approaches will work with the hosted XM solution, you'll need to remediate the approach when migrating to XM cloud.

Remove Dependencies on xDB

If utilizing Sitecore XP, when migrating features remove any dependencies on xDB and other features that will not be available in XM Cloud



Add Composable Features

It's important to consider whether it makes sense to introduce composable features during the migration. It is not necessary to wait until the end to introduce these capabilities. Adding personalization and testing with Sitecore CDP and Personalize will deliver greater business value than just migrating content pages.

Search

XM Cloud has no web database or indexes, meaning that any code that leverages content search API will be rewritten. Composable solutions like Sitecore Search, Coveo Cloud, and Search Staxx Studio provide easy to integrate search solutions. Otherwise, a custom solution is needed to handle search results and indexing.

Step 4: Migrate Features

Once the headless foundation is set and the backlog is ready, begin migrating features. Establish a sprint and release cadence to ensure consistent progress and consider creating a communication plan to inform stakeholders of the progress being made.

With each release, traffic will steadily move from the MVC application to the headless rendering host. Because of this, make sure to evaluate the performance and load of your servers. You may be able to reduce the number or sizing of the servers as the load decreases to your CD servers.

Lastly, keep track of how many features are left before the completion of the migration. As the migration to headless is complete, begin planning the transition to XM Cloud, including the procurement of required licenses.

Step 5: Move to XM Cloud

Once features have been migrated to a headless architecture, it's time to migrate to XM Cloud. It's important to evaluate the features of XM Cloud to make sure it has the capabilities you require. Once ready, you'll need to account for the following:

Rework to Deal With Technical Debt

If features are properly constructed when moving to headless, the migration effort should be minimal. Remember that XM Cloud does not have CD servers, so any code that relied on a CD server to handle a request will need rework.

Provision and Configure XM Cloud

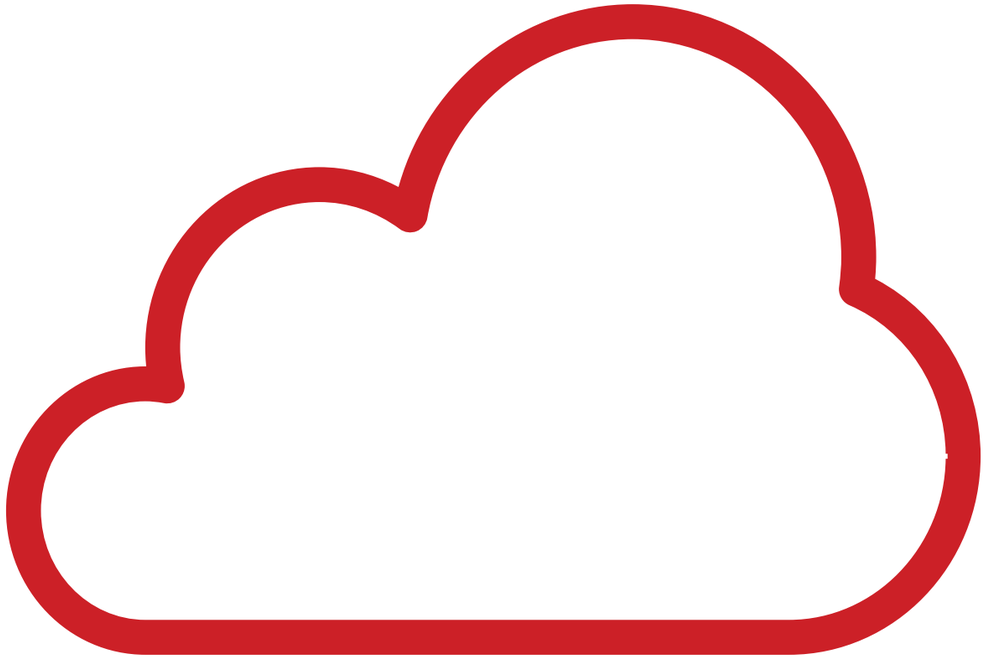
Once you've secured a license for XM Cloud and can access the cloud portal, you'll need to provision projects and environments. Think through the development and management processes and create environments to support quality assurance requirements. Also consider how to manage administration. XM Cloud uses a cloud portal where administrators can issue invitations and provision access.

Migrate Content and Cut Over

With the solution ready to move to XM Cloud, the final step is to migrate the content and cut over. Migrating content can be as simple as packaging everything from your old instance and installing it on your XM Cloud instance. Once content is migrated and testing is complete, the transition should be as simple as updating your reverse proxy to point to your new front-end application.the approach when migrating to XM cloud.

Plan and Execute

While XM Cloud supports Sitecore Experience Editor, it is important to consider enabling other XM Cloud features. Sitecore Pages provide an enhanced editing experience. There are new analytics and personalization capabilities that come with new interfaces for content authors and marketers to master.



Update DevOps Processes and Deploy Templates and Code

XM Cloud currently integrates Vercel and Github, with other providers planned. You'll need to evaluate the built-in capabilities against managing your build and release pipelines in an external tool like Azure DevOps. Given that a headless solution is managed with an external tool, making modifications to target XM Cloud instead of a content management server is often the most straightforward approach. XM Cloud allows for deployments to be initiated in the cloud portal. Whether you want to leverage this is another question.

Once your DevOps processes are updated, the best way to test is to deploy them to your XM Cloud environment. You will need to update your front-end application to talk to XM Cloud's Experience Edge endpoints.

Getting Started

Migrating to XM Cloud is not a quick endeavor or one to be taken lightly. You want an experienced partner like Perficient that can guide you through the decision-making and planning process. We have helped many clients on their headless journey to XM Cloud and can help you, too. Whether you're looking to lift-and-shift your existing solution to headless, redesign your sites as part of the migration, or want a partner to work together on a migration in a co-development model, we offer standardized approaches that can help.

If you're unsure how to proceed, our headless and composable roadmap offering can help you build a customized roadmap in as little as four weeks.

Why Perficient

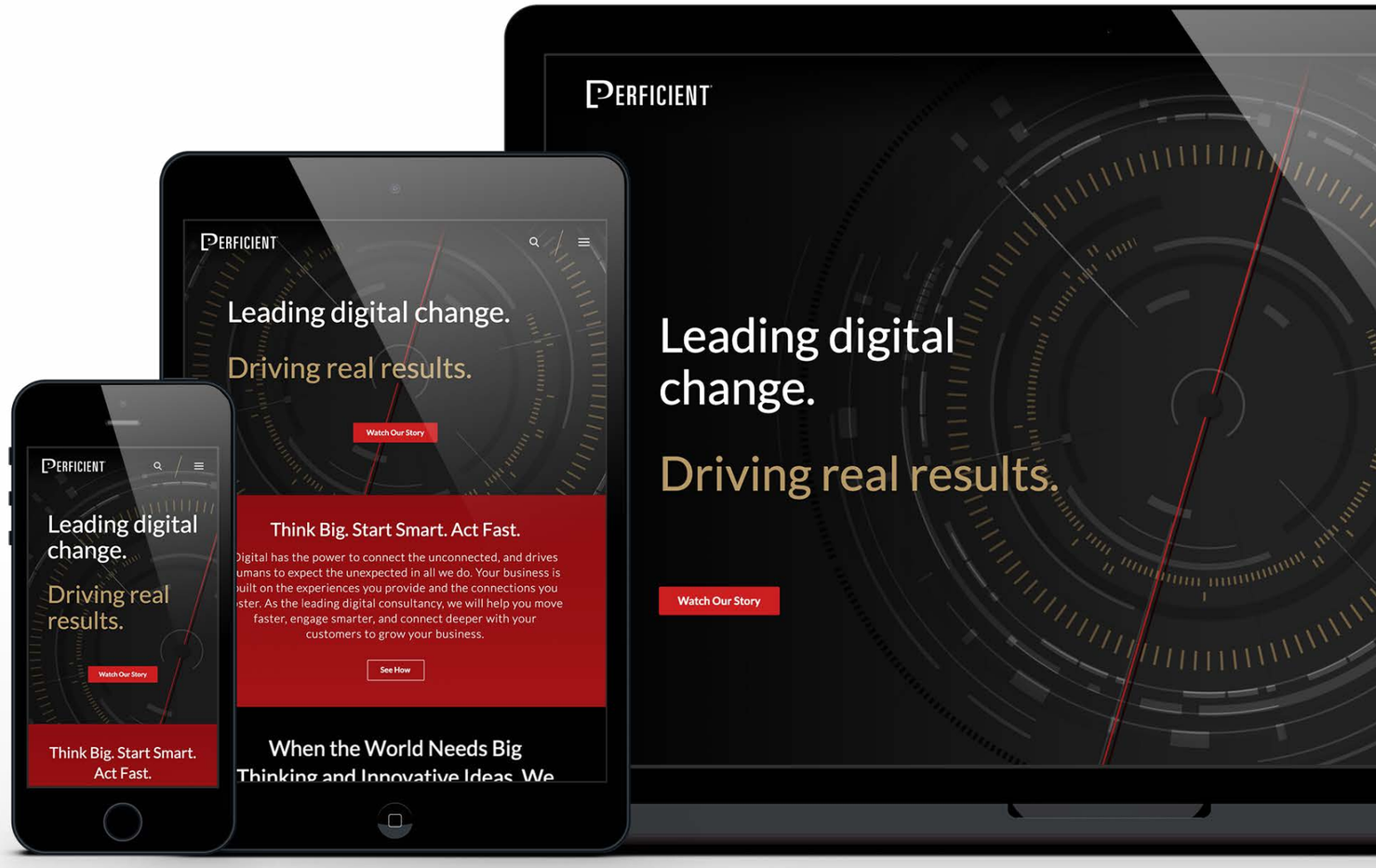
We are a Platinum Enterprise Solutions Partner helping our clients create meaningful and engaging digital experiences that delight customers, build brand loyalty, and drive revenue. From strategy through design and implementation, our global team of technical and industry experts will help you realize value with Sitecore faster than you thought possible.

We're the industry leader in leveraging the power of Sitecore's to empower the world's biggest brands to chart their future with a composable platform strategy. Combined with our expertise in experience design, strategy, digital marketing, and analytics, we create end-to-end solutions that deliver exceptional results.

Let Perficient help you on your digital transformation journey.

Perficient is the leading global digital consultancy helping transform the world's largest enterprises and biggest brands. As a trusted end-to-end digital provider, Perficient partners with its Global 2000 and other large enterprise customers across North America to design and deliver digital transformation solutions that exceed customers' expectations, outpace the competition and transform their business.

PERFICIENT®



PERFICIENT.COM/BLOGS

PERFICIENT.COM/INSIGHTS

(855) 411-PRFT(7738)

PERFICIENT.COM/CONTACT