



LIFE SCIENCES

Selecting the right deployment model and approach for Life Sciences Cloud

Authors: **Luke Emberton, James Hemsley**

Contributors: **Rich Pucci, Rich Beadle, Ellie Rice, Mark Wraith**

slalom

A pivotal moment in life sciences CRM

With Veeva's announcement to move away from Salesforce onto its own platform, life sciences customers are looking at their CRM solutions and deciding whether to migrate to Veeva Vault CRM or remain with Salesforce and move to Life Sciences Cloud. While this could merely be a replatforming migration, with advancements in technology and evolving industry demands—including the rising prominence of AI, personalization, and patient-centricity—organizational leaders should approach it as an opportunity to transform their business.

Determining system environment strategy

Life sciences CRM implementations are often global, spanning many geographical regions and business units as well as thousands of users. For companies operating in more than one country, leaders need to determine whether it's best to use one global instance or multiple, separate instances (single vs. multi-org).

Answering the following questions can help you decide which option will work best for your organization:

- **What business model/process commonalities and differences exist between countries and regions?**
- **How many lines of business do you have, and can they be managed in a single org?**
- **What local compliance and data privacy regulations are in place?**
- **Are there export controls, sanctions, or sovereignty rules to consider?**

The first step is to review and categorize the regions and countries where the solution is to be deployed based on how you answered the questions above. At Slalom, we recommend that our life sciences customers evaluate data privacy regulations across all their regions and build a global baseline that takes the most restrictive requirements into account. This baseline then becomes a company-wide standard and is used as a reference when looking to introduce new systems into the enterprise.

The next step is to identify rules on sovereignty and export controls and map them to data type and country. These rules have the most direct impact on what data needs to be kept within a specific geographical region.

Companies opting for a Life Sciences Cloud implementation should conduct a thorough discovery to gain an understanding of the countries the solution will be rolled out to, the functional capabilities to be delivered, and the corresponding conceptual data model needed to support those capabilities. If regulatory/policy rules and business processes are common across multiple countries, using a single Salesforce instance may be the best option. Where there are variations across geographies, you may need multiple Salesforce instances (single org vs. multi-org).

Single-org solutions, however, could still be viable where regional variations are present. With Salesforce platform enhancements and the speed at which the technology is evolving, org leaders have a variety of options, particularly when looking at a native Salesforce solution instead of a package built on top of Salesforce. Regional variations can be supported by configuration switches in a single instance as opposed to replicated functionality across separate instances. Robust data visibility rules can restrict data access, so users only view data they are allowed to see based on their roles and responsibilities.

With careful and thoughtful design, regulatory and compliance requirements can also be supported within a single-org implementation. Leveraging Salesforce's packaging features, modules subject to the Prescription Drug Marketing Act (PDMA), 21 CFR Part 11, and regulations around reporting medical questions and adverse events can be isolated from other functionality and managed in separate code repositories. Each module can therefore follow different—and, where required, more meticulous—software delivery lifecycles than other parts of the wider implementation.

The goal should generally be to limit the number of instances as much as possible, as many historic justifications for managing multiple instances no longer apply. For example, the General Data Protection Regulation (GDPR) in Europe was often cited as a reason for separating US and EU instances, but with appropriate data controls and clarity on where data is held, it is possible to host the same instance in a US or European data center. Similarly, many life sciences companies that implemented Veeva over a decade ago adopted a multi-organization (multi-org) approach due to heightened concerns about regulations and data privacy. Regardless of the platform chosen, it is now crucial to re-evaluate this multi-org strategy, as the conditions and platform capabilities that led to the initial decision may no longer apply.

Salesforce multi-org deployments

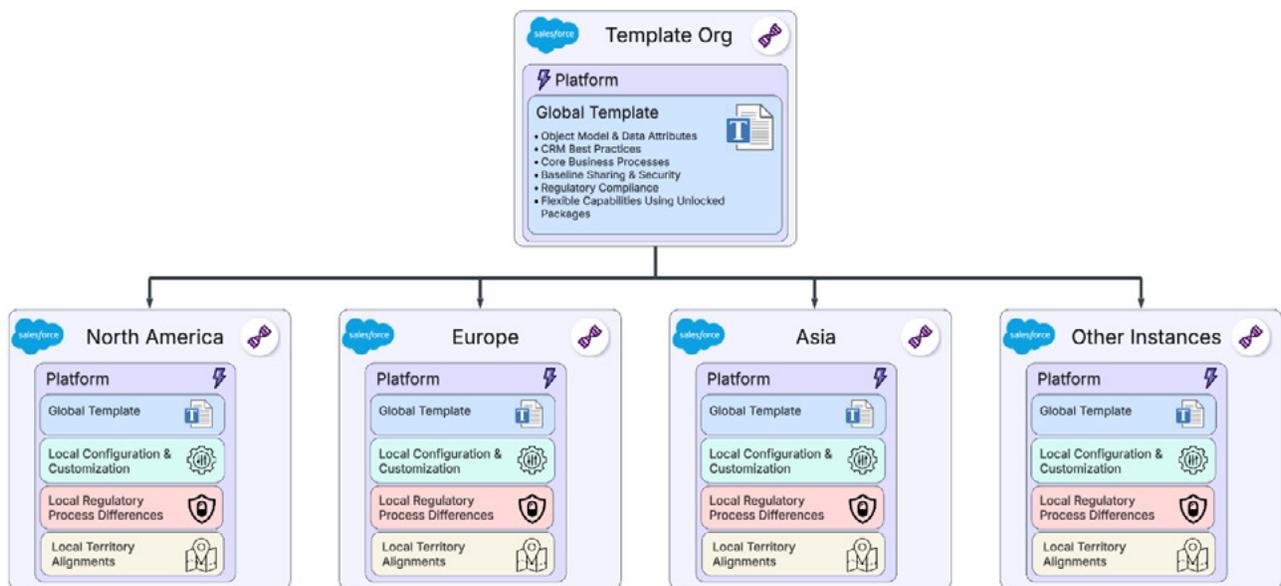
Should multiple Salesforce instances be necessary, it's important to consider taking the following actions:

- **Optimize development time and effort by reusing developed functionality in as many regions as possible**
- **Automate deployment to each of the Salesforce instances to speed the process and remove error-prone manual steps**
- **Automate regression testing to boost quality, particularly where functionality may be subject to computer system validation (CSV)/computer software assurance (CSA) requirements**
- **Streamline efforts to support multiple orgs integrating into global solutions, such as customer master data management (MDM) and business intelligence tools**
- **Carefully analyze and compare orgs before migration to determine whether any variability between orgs exists**

Core template development and rollout

Once you've defined your regions and country groupings with similar regulatory requirements and business processes, you can build a core or global template to promote reuse, disseminate best practices, harmonize, and standardize. The template can then be rolled out and tailored for requirements in specific regions based on deployment priority.

This approach can pose some challenges. It can be difficult to formulate a broadly representative core template without using specific regions' requirements, which is why the template should be built in parallel to the pilot deployment to a representative region. The implementation team then needs to have a clear understanding of which requirements are region-specific and filter those from the core build.



Options for managing and rolling out the global template

Salesforce provides two options for building and maintaining the core template and deploying it to region-specific instances. Importantly, both options leverage a source-driven development approach that maintains the source of the core template as opposed to an org-based model with a "packaging org."

Source-driven development offers support for version control and more resilience by providing an audit of changes that are made to the application and the ability to roll back to previous versions. Incorporating a source-driven approach as part of a broader DevOps program also enforces a change control process and robust deployment procedures. This will also help teams to be aware of and sign off on all changes. These capabilities are essential if the template includes functionality that is categorized as requiring validation under FDA guidelines.

Leveraging either of these new approaches reduces the potential for deployment issues and costly regression testing that need to be identified and resolved in the org-based deployment models commonly adopted in the past.

Option 1: Leveraging unlocked packages

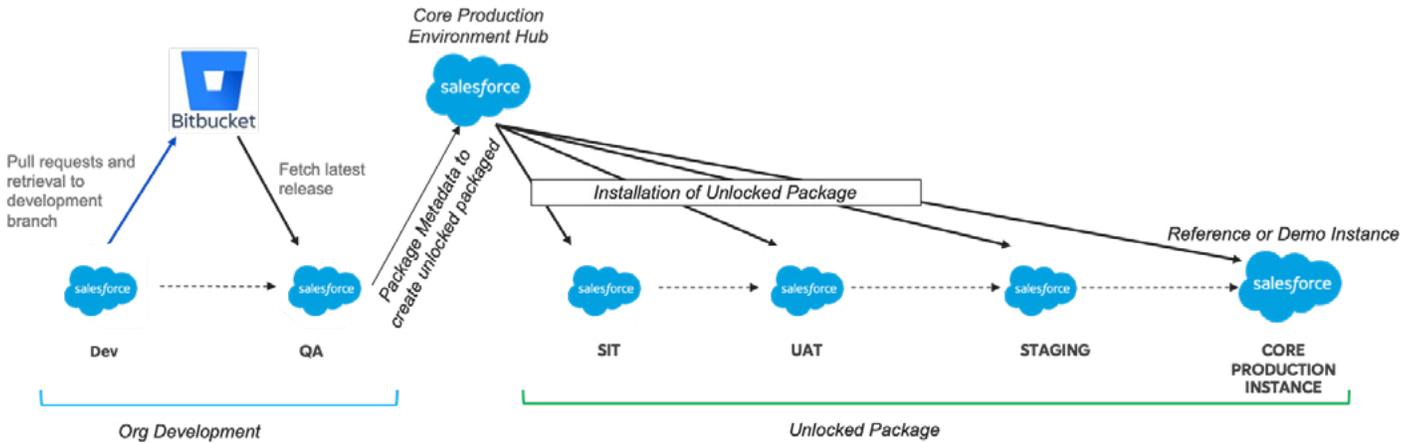
Salesforce provides packaging functionality that supports bundling source code to self-contained applications. This is an approach that has been leveraged by independent software vendors (ISVs) since Salesforce introduced the AppExchange. More recently Salesforce has evolved this capability to provide “unlocked packages” that work with the Salesforce deployment technology Salesforce DX to support deployment management for internal business applications.

Unlocked packages support modularization of source code, managing dependencies across metadata and other packages while supporting subsequent configuration of the source contained within. Source metadata can be organized into packaged directories with everything managed in a version control system. Unlocked packages can be deployed to any Salesforce instance and upgraded as they evolve. Unlocked packages have versioning, providing traceability of what configuration is present in each system where the package is installed. Leveraging packages also supports greater consistency to deployment as they can be deployed rapidly and with the change history that is often needed.

With this approach, a core Salesforce org will be leveraged with a series of development sandboxes to create the global template, which can then be published as an unlocked package and installed into region-specific instances.



Core template development flow



The core template development flow is shown above. It begins with a standard “org development” model. This supports a faster build and aligns with the traditional Salesforce development approach that experienced Salesforce developers are familiar with. User stories are developed as features and checked into the “develop” branch for deployment to a continuous integration (CI)/quality assurance (QA) sandbox supporting unit testing and automation of core build processes such as code quality checks, automated regression test execution, and peer reviews. Developers are thereby freed from the complexity and time demands of the packaging process.

Once the user stories have been developed and unit tested, they are packaged up by a DevOps/release management team with guidance from the development team. This forms the template core package, which is then installed in system integration testing (SIT), user acceptance testing (UAT), and other preproduction environments for quality assurance. After successful testing the package is installed in production for the core environment. The core production environment is not used directly for business operations by any of the regions; it serves as a reference or demo instance.

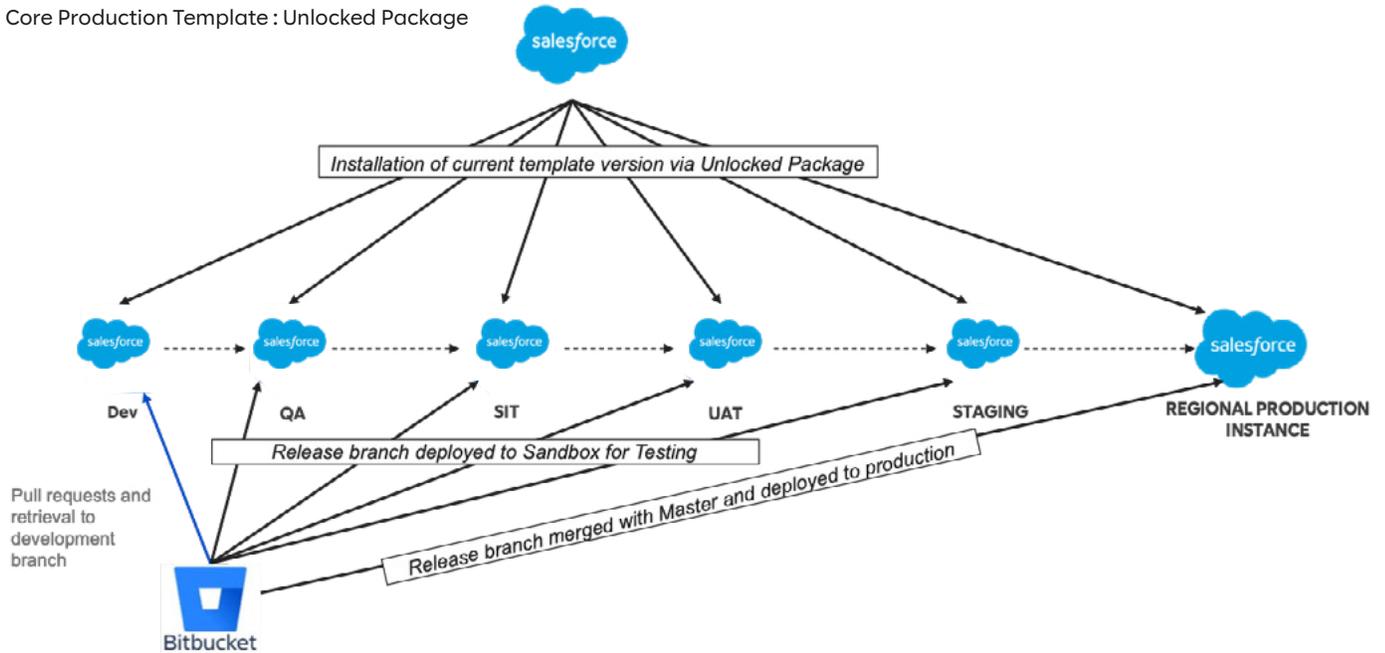
Some best practices that should be followed when developing or enhancing the unlocked package include using package snapshots that reduce the time to create a snapshot as the source is modified and packaging the delta instead of the full package.

Region configuration/customization flow

The development and configuration for a region is then performed in a separate set of sandboxes. The appropriate version (normally the latest) of the unlocked package is installed into each of the dev, test, and production instances as part of the release process.

Region-specific customizations are then built on top of the unlocked packages. These are committed to a region-specific repository leveraging standard Salesforce development tools (such as VS Code or Copado/Gearset) and branching models (e.g., Git-Flow).

Core Production Template : Unlocked Package



Unlocked package content is visible to the developers within the target instances. Developers can configure and tailor functionality included within the unlocked package based on the requirements of the region, but a governance process should be created to determine what content developers can change and what should be locked.

The final deployment to production includes the installation of the unlocked package and the deployment of the region-specific functionality from the region-specific repository.

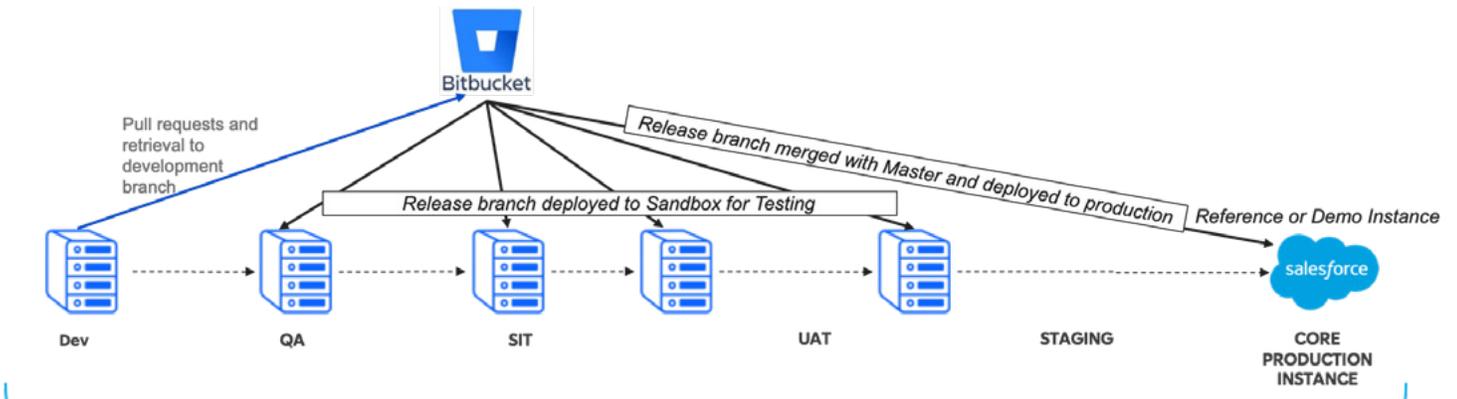
Further sophistication and control can be added to the process by developing the region-specific customization in its own unlocked package. The benefit of modularizing the metadata should balance out the complexity of packaging and the time needed to evolve the solution.

Option 2: Org-based development using a source control management system to manage the core template

In this model the responsibility of managing and maintaining the core template falls on the source control management system, typically a Git solution, for example from Bitbucket. A core template repository is established where the solution relating to core functionality is committed and evolved.

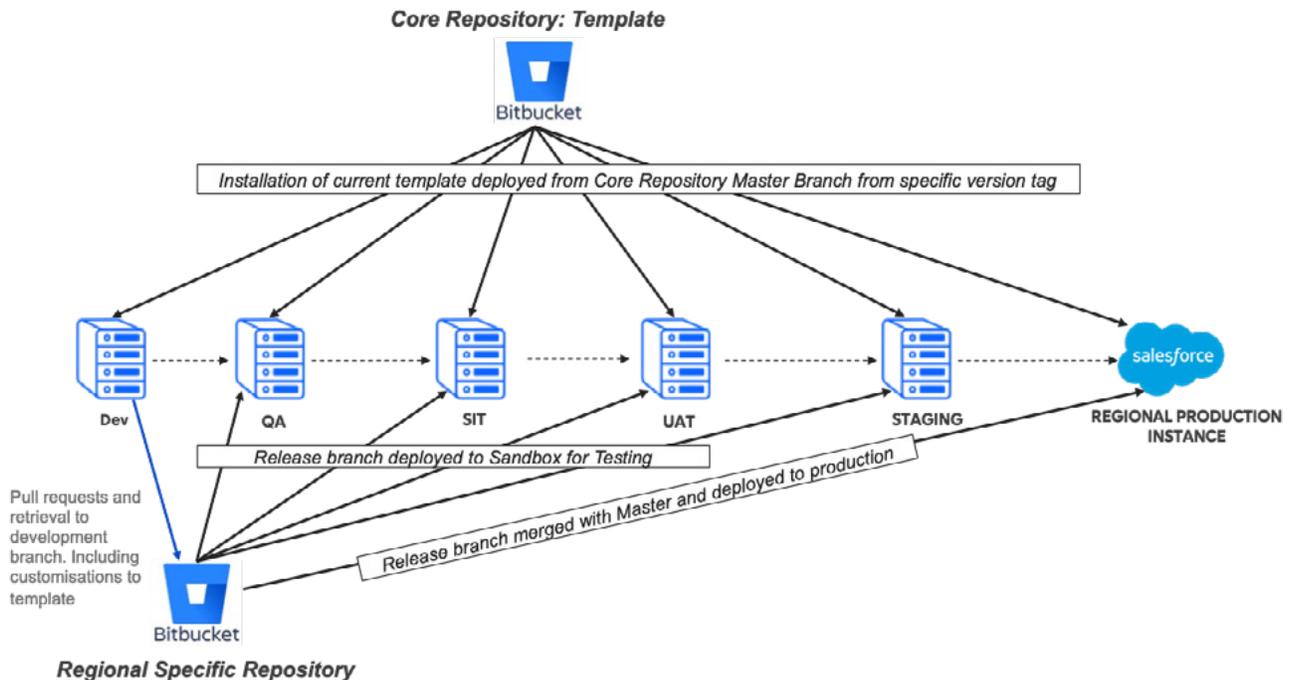
When a version of the core template is completed, it is then deployed to a region-specific org. The region-specific org has its own Git repository for managing configurations developed for that region. The deployment process includes the combination of both the version from the core template repository used during the region's development process and the version from the region-specific master repository.

Core template development flow



The development of the core template follows a normal dev flow with features baked into releases for testing before being merged into the master branch and tagged with a version.

Region configuration/customization flow



The region-specific configuration then has the core functionality deployed from the core template master repository with region-specific customizations developed in the region-specific sandboxes and maintained in their own source code repository as shown above.



Additional considerations

When using the source control approach, some of the benefits of packaging don't apply, such as modularizing the metadata and having control over the version of truth where conflicts in functionality exist. Deploying the core template and region-specific customization as separate steps also introduces certain risks. Should one of the steps complete and another does not, the system can be left in an indeterminate state.

Slalom has built best practices to help overcome these challenges:

- **Never overwrite/update Apex classes or flows in the regional org that are maintained in the core**
- **Clone page layouts/user interface layer components and profiles before customizing them in the region-specific instance**

Slalom has also developed automated scripts that merge the core template repository and region-specific repository before deploying to a Salesforce environment. This enables the source to be deployed in a single step that will either complete or roll back, thereby ensuring that the environment is not left in an indeterminate state. As part of the automated scripts, additional features are incorporated that replicate unlocked packaging functionality, such as preventing metadata in the template solution from being overwritten by regional-specific changes.

When to select which model

There are advantages and disadvantages to consider when choosing a development approach for building the core template. There is a pathway to migrate from one model to another, but this takes significant effort and is time-consuming, particularly when moving away from unlocked packages. It's therefore important to carefully consider your approach when beginning an implementation.

From Slalom's experience of implementing both approaches, we've found the following pros and cons for each:

Unlocked package

Benefits

- This approach allows for modularization and versioning and manages dependencies.
- Using package installation provides a more predictable approach compared to deployment.
- Partitioning the application into packages (core and region) provides protection against core components being overwritten, thereby supporting ease of upgrade.

Challenges

- There can be a time lag before metadata types are available for packaging, particularly for newer features. Examples include experience site configuration, Omnistudio, permissions, and profiles. These metadata types then either need to be excluded from the solution or a mixed installation approach using packaged and unpackaged metadata is required.
- Higher complexity to develop and maintain packages requires highly skilled and dedicated DevOps practitioners.
- The packaging creation process can take a long time and adds overhead to development. Development speed can be slower due to the need to build pipelines to execute (scratch org snapshots can help to reduce this).
- There are no "delta deploy" or "quick deploy" options, which means package installation durations can be long.
- It is more difficult and time-consuming to identify and resolve package installation errors because of the time needed to install and gain feedback. Less information is provided in error messages.
- The ecosystem around the use of packages is still maturing and therefore has less support than traditional deployment approaches.



Leveraging a source control management solution

Benefits

- This is a more familiar development path for those experienced in Salesforce development/configuration.
- Pipeline builds are much faster.
- There's better coverage of metadata components for a mostly autonomous deployment.
- Timelines for deployment can be reduced leveraging "delta deploy" and "quick deploy."
- It is possible to merge metadata before deployment to make deployment a single step.

Challenges

- This approach does not provide the same level of modularization/containerization as packages.
- It also doesn't provide dependency checking to verify whether build has a complete set of components.
- It does not natively support upgrading or protection for overwriting of core components.*
- A two-step installation process is necessary unless metadata is merged before deployment.*

*Slalom has developed several DevOps build and deployment scripts that overcome these challenges, including automated merging of core- and region-specific repositories and "approved overrides" to prevent core components from being modified.



Selecting an approach

For companies implementing Salesforce Life Sciences Cloud using a multi-org solution, the above considerations should be taken into account. Slalom can support and advise clients on their choice. The Salesforce technology continues to evolve with new features regularly added to support deployment. Based on the recent implementations using these deployment approaches Slalom has found that the source control management approach for building the core template has been more effective for clients, particularly where they are using newer features such as those which are present in Life Sciences Cloud. Slalom's prebuilt scripts enable clients leveraging the source code approach to streamline development and seamlessly support the deployment and configuration of region-specific instances.

Conclusion

Transitioning to Salesforce's Life Sciences Cloud provides a significant opportunity for life sciences companies to transform their operations, but doing so requires careful consideration. By aligning CRM solutions with evolving industry demands, such as AI, personalization, and patient-centricity, as well as requirements stemming from global business models and data privacy regulations, life sciences organizations will be better prepared to take on whatever challenges and disruptions the future holds. Partnering with Slalom and Salesforce provides you with trusted advisors who can help you ask—and answer—all the right questions and take the most informed steps during this transition.

**Let's solve
together.**

Ready to break free from legacy systems and embrace modern, scalable architectures? **Talk to one of our experts today.**

