ORACLE

Scalable, Resilient, Flexible:

Why Cloud Native's Time Is Now

Modular software offers major advantages.
Here are pain points to tackle today.

Table of contents

Are monolithic apps holding you back?

By Alan Zeichick
Senior Writer

Most enterprise IT teams maintain dozens or even hundreds of monolithic applications that serve the unique needs of their organizations. Written by developers using techniques honed by decades of experience, those pieces of software are highly useful. But they're also difficult to maintain, and adding new functionality is often deemed too risky.

By contrast, cloud native design based on web services promises easier upgrades to security, user experience, and application features. It improves developer productivity by enabling flexible code reuse and simplifying software testing and bug finding and fixing while supporting better application scalability and interoperability.

The fact is that many older monolithic apps were built for usage models that are fading from existence. Software may have been developed based on the notion that it would always function within a secure perimeter, that data formats wouldn't change or be extended, or that users would always be equipped with a certain type of device.

These are not good assumptions today.

It's time to take advantage of an ever-growing library of services available in the cloud that let developers easily expand the reach and capabilities of their applications. Whether it's the capacity to use advanced authentication services such as biometrics, incorporate AI capabilities, embed analytics, or create data visualizations, the path to building rich applications that move the organization forward begins with cloud native development.

> Cloud native design based on web services promises easier upgrades to security, user experience, and application features

The benefits of cloud native development

We're using the term "cloud native development," but it's also known as web services development or microservices design. All imply development for the cloud, and it's likely that you're using this methodology with one or more hyperscale cloud providers.

However, this approach isn't limited to applications that live fully in the cloud. You can use the same microservices techniques to build applications that run in your own data center, in multiple clouds, or in a hybrid model. In fact, developing cloud native software that runs side by side with your existing systems gives you options, including how you'll add features and when to move existing applications to a cloud services model, if doing so makes sense for your organization.

A popular place to start is by creating a flexible, secure, and accessible data management architecture for your primary data storage. Today, that means developing a system that can manage structured and unstructured data and make it available to applications wherever, whenever, and however they need it.

You have data. You use that data. What happens when your needs change?

Imagine a monolithic sales management application. Its database contains information about customers: key contacts, support interactions, purchase history, past transactions, current orders. It tracks which salespeople are associated with those accounts, discount status, credit terms, and more. Customers can access their data using one set of interfaces; your sales, sales management, billing, and shipping teams have other interfaces.

That's baseline functionality; no rocket science here.

What about when you do need some rocket science, though? Helpful features that aren't included in that already-huge sales management application? If it's an off-the-shelf licensed CRM platform, you can ask the vendor to add those features…and then wait. If it's a home-grown system, you can ask your development team…and then wait. Maybe the capabilities will be added if there isn't too much of a backlog, the features don't require substantial changes to the code and its data source, and the addition isn't deemed too expensive or risky given the importance of the application.

But what may well be incredibly costly is delay.

Let's say, for example, you want to automatically collect and add transcripts of each sales teleconference to customer records and start mining those conversations for opportunities to improve service and to perform sentiment analysis. Is customer satisfaction increasing or decreasing? What insights might offhand comments reveal about your competition?

Adding that capability to your existing system is likely to be a nightmare. Transcripts with audio create a lot of data. Where are you going to store it, and what will it take to convince your database managers to spend that money? Success will demand timely data analysis and even generative AI functionality. Do your developers have those skills? Will they be willing to make such substantial changes to a crucial piece of enterprise software? It's an uphill battle.

> The cloud native development approach offers a new way—an easier way—to make such changes by leveraging microservices.

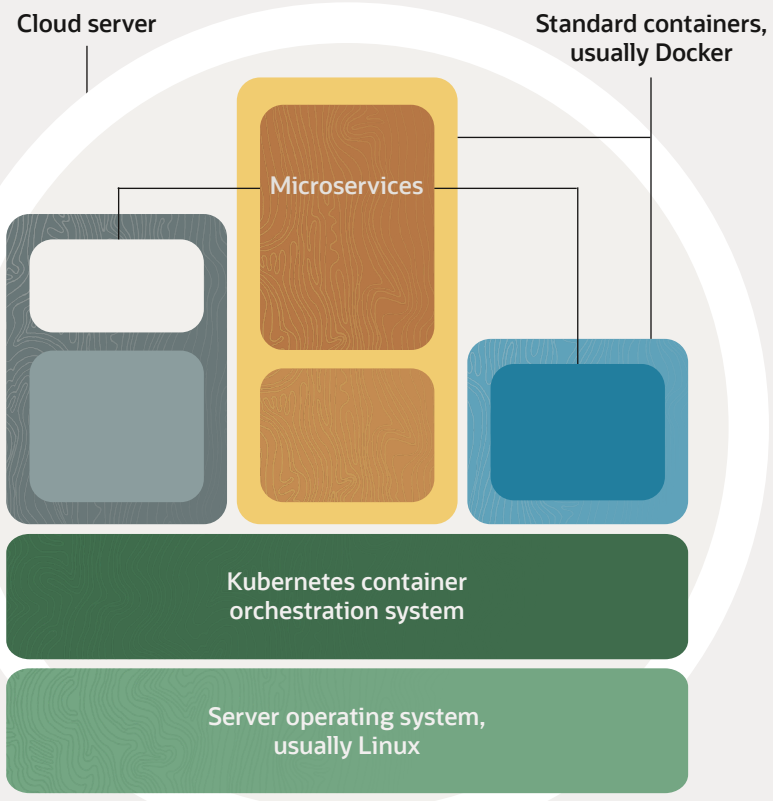
Microservices let you extend applications by stitching them together

Cloud native applications are composed of microservices, which are freestanding pieces of software—complete applications in themselves—that support a single task or perhaps a small number of related tasks. Each microservice manages its own resources. It has its own memory, its own processors, its own communications capabilities, and its own access to databases and storage. Each microservice is deployed separately, and management is automated.

The easiest way to leverage cloud native technologies is to design and construct applications using microservices. That's why the microservices approach is fast becoming the leading way to develop new software, even when it will run entirely within your own data center.

How does a cloud native architecture work?

A cloud consists of many servers, each of which contains an operating system, a container management system, and many containers. Each container runs one or more microservices.



Now let's go back to that sales management application. What if you can't rewrite your application to use microservices and are stuck with that monolith, at least for now? No problem. Write a new cloud native application that runs alongside the existing app and delivers the additional functionality that you need. As long as the new cloud native application has access to the original sales software's data stores, it can provide the functionality you're after.

In this scenario, then, your existing sales application runs unchanged. Worst case, you may need to make a one-time adjustment to provide hooks so that the new application can securely request data. The new microservices-based application will then store those valuable customer transcripts, with links to original customer records.

Over time, you can continue to add functionality to the cloud native application. Because of the modular approach enabled by microservices, you'll find the new system more responsive, easier to upgrade, and more flexible than the old. Eventually, you may wish to phase out the legacy software by adding its functionality, piece by piece, to the new application. And then, one day, you can retire the old app entirely—your now cloud native sales application can take on any new challenge.

> The easiest way to leverage cloud native technologies is to design and construct applications using microservices.

What it takes to do cloud native

In a microservices world, a software architect defines which services are needed and how they will communicate and interoperate. Developers can then build, test, and deploy each service separately—or, often, take advantage of services that are already built.

Essential elements of a cloud native application

Cloud native apps can be faster to build and easier to maintain because they use containers and APIs to break applications down into smaller, more manageable parts.

Containers
Containers are a lightweight and portable form of virtualization technology used to segment applications into smaller, independent services that can be developed, deployed, updated, and scaled independently.

Container orchestration system
A container orchestration platform, such as Kubernetes, automates the deployment, scaling, and management of containerized applications.

APIs
Application programming interfaces that expose application functionality through well-defined functions make for easier integration.

Best practices

DevOps and SysOps	A DevOps culture fosters collaboration between development and operations teams, which is vital for cloud native success. Once an app is deployed, SysOps makes sure that the app and the infrastructure run reliably.
Infrastructure as code (IaC)	Per this methodology, managing networks, servers, and storage programmatically allows teams to configure systems to dynamically meet an application's needs.
Observability	Teams can monitor cloud native application behavior through metrics, logs, and other granular methods.
Security	Robust security measures—including authentication, authorization, and encryption—should be configured for web applications.
Continuous integration/continuous delivery (CI/CD)	Automating the building, testing, and deployment of applications helps ensure frequent and reliable releases with strong version control.

Security functions in a cloud native application are typically handled by their own microservices, so programmers must include both authentication and authorization to access data or use services. A security advantage of the microservices approach is that data access privileges can be managed separately from the application. In our example of storing transcripts of sales calls for sentiment analysis, the application will check which customers each salesperson deals with. If a salesperson changes roles or leaves the organization, HR will change the employee record, and that information will be available to the application. Customer records that the salesperson could access before won't be accessible anymore because the system can hook into that HR data.

Contrast that with monolithic applications that often attempt to manage their own user access controls.

And, since we're talking about web services, encryption of data in motion will be provided by HTTPS. Authentication can use OpenID standards, and authorization can be done via OAuth 2.0. So, while you will need to determine how to manage users and establish their access to data and services, the dev team won't have to create protocols to do so. Those capabilities are mature, vetted, and widely available.

> Developers can then build, test, and deploy each service separately—or, often, take advantage of services that are already built.

Cloud native collaboration

You've likely heard about DevOps and SysOps, for developer and systems operations, and that this collaborative approach is crucial for success with cloud native applications. Once developers build and test microservices that perform as expected, operations teams are charged with creating a secure, reliable, high performance environment for the new cloud native application. The two functions—development and operations—are equally important because choices in software architecture will affect scalability and reliability.

For example, developers package microservices into containers that hold everything a service needs to function and virtualize the environment so a service can run on any server, on premises or in the cloud. Containers are lightweight—they store just enough of the service's environment to let them start and stop quickly. Operations teams know a server may take minutes to boot up, while containers typically start in just a few seconds. Because of this, new containers can be easily started when loads are high and shut down when loads are low, as long as they're deployed properly.

5 side benefits of cloud native development

We've discussed how cloud native development lets you move away from monolithic architectures. But the benefits don't stop there.

1

High availability

Because microservices run in virtual containers, cloud native development allows for all sorts of useful configurations. If your application is available globally, imagine deploying instances of it on each continent for performance purposes. Or perhaps you'll duplicate services outside of a hurricane zone to improve reliability. You might also need to keep data and processing localized to meet compliance requirements.

2

Lower costs

The system that helps manage the virtual environment, Kubernetes, is an extremely popular open source platform that deploys, scales, and manages containers. Kubernetes lets you start, stop, and group containers individually and in clusters, using those capabilities to provide reliability and scalability. Despite all that functionality, its open source nature and popularity make skills acquisition relatively affordable.

3

Increased uptake

Kubernetes not only lets you manage your deployment, it helps users discover the microservices that will offer them the fastest response times. Our system for determining customer sentiment may be running in a distributed fashion in global regions, for example. With Kubernetes, sales teams in Asia can discover and use the microservices running nearest to them, which will differ from the services that teams in Europe discover. And each team will experience snappy performance.

4 Multicloud compatibility

Cloud native applications are designed to be portable and flexible, making them easier to migrate between different cloud platforms. This flexibility allows organizations to shop multiple cloud providers for the best features, localization, and pricing, and then move their applications accordingly. Thus, Kubernetes helps deliver the benefits of multicloud.

5 Facilitate unified data architectures

Finally, one of the biggest advantages of a cloud native architecture is that it's supported by every large data management system in common use. Oracle, Microsoft, AWS, Google, and many open source projects support data access via web services. Limiting the number of database systems and access methods makes creating and maintaining any app much more manageable. A unified database can improve data integrity by helping ensure that data is never duplicated or out of sync while reducing the time it takes to perform queries and analytics across multiple data types.

When adding, say, generative AI functionality, having all relevant data in a single converged database can speed up fine-tuning and day-to-day inferencing by orders of magnitude. In a scenario where you have a monolithic application running next to a modern microservices application, your goal might be to migrate to a single unified database solution. Doing so may require some changes to the monolithic software, but there are substantial benefits to having all that data in a single platform that's accessible via microservices.

Reasons to adopt cloud native development now

A cloud native approach will help your organization stay competitive. Cloud native architectures let you leverage the scalability, flexibility, and resilience of the cloud for faster deployment of applications and services. They can also help you foster collaboration and better position your business to leverage emerging technologies such as GenAI and zero trust security.

Top reasons why cloud native should become your default software strategy:

Cloud native facilitates rapid application development.

When your organization sees a problem or spots an opportunity, you don't want to spend months defining requirements, then yet more months determining the architecture for a new application to address the need. With a cloud native approach, the architecture is a simple description of the user interface, the core services and data required, and the communications needed to link those services. Development teams can then jump in to refine those services, define small microservices, write any custom code, and stitch everything together. It's a much faster process, especially because some—or most—of the services you need already exist.

Cloud native simplifies software fixes and updates.

Bugs happen. The trick is to find and fix flaws as quickly as possible and then deploy an update with the least possible disruption.

In a microservices world—with applications composed of independent, loosely coupled services—the scope of a bug is generally constrained. Teams can deploy updates to individual components without affecting the entire system, minimizing downtime and risk. Once a problem is found, developers can fix, update, test, and redeploy one service. Every application that uses that service gets the upgrade instantly.

Cloud native provides scalability and high performance.

When an application is composed of dozens or hundreds of independent pieces, if one service creates a bottleneck—often detected automatically due to slowed response times—the Kubernetes container management system immediately autoscales by launching an additional containerized version of that service. That new resource helps share the load, and the problem is resolved. If another bottleneck appears under different workload conditions, the same scalability fix happens automatically. The reverse occurs, too. When demand for a service decreases, resources can be scaled back, resulting in cost savings.

Cloud native apps are resilient and fault tolerant.

There's no one critical cog in a well-designed microservices deployment—no piece that, if it fails, takes the whole application with it. That's because there are many layers of redundancy at the container level. If a service drops, Kubernetes will detect the problem and route requests to redundant resources while restarting the service or spinning up new instances.

A container crashed or a rack lost power? A well-architected design can start up new containers with services running in a different rack or a different data center, restart, and route network traffic accordingly. Done and done. And since the Kubernetes system is itself a microservice, it's also resilient. This extra level of fault tolerance, by the way, is why many cloud native applications are run in hyperscale clouds instead of single data centers.

> When demand for a service decreases, resources can be scaled back, resulting in cost savings.

Cloud native practices allow for better separation of duties.

In a cloud native infrastructure, all the teams involved in IT systems management have the autonomy to do their jobs. Granting these teams only the privileges necessary to perform their specified tasks helps organizations prevent unauthorized access and facilitate audits.

Say it's time to move to multifactor authentication. The group that manages security can implement and test new authentication models without involving application developers. Even when the system is rolled out, the web services app may need little or no modification to function under these new conditions. Meanwhile, data management teams can implement user access controls and create a governance structure, and the user experience group can do its work while network administrators manage changes and upgrades to the network, all operating independently. And, when it's time for the company to move into a new market, local teams can handle language and taxation differences, inventory management, contracts, and dozens of other service-related variables in their regions without affecting how the app operates in other markets.

A matter of protocol

The glue that makes cloud-native applications work are communications protocols that connect microservices, even when those services are running in different containers, physical servers, data centers, regions within a single cloud provider, or even in clouds from multiple providers. These protocols almost always use the request-response model via application programming interfaces, or APIs.

> The glue that makes cloud-native applications work are communications protocols that connect microservices.

The most common API protocol set for cloud-based architectures is representational state transfer, or REST, and it takes place over the same HTTP protocol as much of the internet. This means that services can exchange information with any other service reachable online. REST's many benefits include efficiency, compatibility, scalability, and security.

Cloud native development unlocks cutting-edge functionality, including AI.

Modern cloud platforms offer developers more than servers, networking infrastructure, and tools for managing containers and Kubernetes orchestration. They also bring a wealth of battle-tested APIs that don't need to be written, tested, or updated.

Oracle Cloud Infrastructure, for example, offers thousands of APIs to services that are vetted, maintained, and extensively documented, and that make it easier to adopt the newest technologies. One such example is OCI Generative AI Agents. This fully managed service combines the power of large language models (LLMs) with an intelligent retrieval system and other AI agents to create sophisticated automated processes or perform complex business analysis. If you have lots of documents, for example, these services can analyze and summarize them for your end users in seconds. Little to no coding is required, because—after all—these are API-accessible services that are there for you and your developers to use.

There are also AI services for speech-to-text, text-to-speech, image recognition, text analysis, language translation, and even prebuilt services to enable conversational, natural language chatbots. All are accessible as services via cloud native applications built using a microservices architecture.

Remember the sentiment analysis functionality we mentioned earlier? It's available in OCI. That means your developers don't need to master aspect-based sentiment analysis and natural language processing. All they need to do is use the API in your microservice, and away you go.

Oracle tools and technologies for cloud native development

At the center of Oracle's offerings for building and deploying cloud native applications is OCI, a distributed cloud that can run in the public cloud, in a multicloud environment, in a hybrid cloud using in-house resources, and as a dedicated cloud running entirely within your own data centers.

When you use OCI, you can leverage Oracle cloud native services to develop and manage applications based on tested and fully supported open standards, specifications, and APIs defined by the Cloud Native Computing Foundation (CNCF).

Accelerate innovation and cloud migration with the power and flexibility to deploy OCI Oracle Database services in any cloud. Combine the best of the cloud with your data to quickly build and modernize applications.

Learn more **AI Solutions Hub**

Connect with us

Call +1 800 363 3059 or visit oracle.com/ca-en/

Outside Canada, find your local office at oracle.com/ca-en/corporate/contact/