Migrating legacy Software to the Cloud

Legacy applications require adjustments before migrating - a necessity but also a unique opportunity.

codesphere

Executive summary

- Migrating legacy software to the cloud strategic imperative
- Lift & shift often not feasible because of license, performance and security considerations
- No best practices for structural infrastructure decisions - only trade-offs
- The move is getting easier as cloud native approaches mature

Strategic advantages of migrating legacy applications

The transition from on-premises infrastructure to cloud computing is a pivotal transformational project for modern businesses. This shift necessitates the modification of legacy software systems, a critical step for several reasons. Firstly, legacy applications often lack the scalability and flexibility required for cloud environments, hindering potential benefits such as cost efficiency and performance optimization. Secondly, cloud migration presents an opportunity to update security protocols, as legacy systems may not meet current cybersecurity standards.

Additionally, adapting legacy software for the cloud enables integration with advanced cloud services and technologies, fostering innovation and competitive advantage. Failure to modify these systems can lead to increased operational risks and missed opportunities for business growth and efficiency. Therefore, altering legacy software is not just a technical necessity but a strategic imperative in the journey towards digital transformation and cloud adoption.

As these technologies mature and businesses collect experiences it gets easier to judge when to migrate and when not. Hybrid options become readily available and new solutions help with the daunting transformation & complexity that comes with the new architecture.

Lift-and-shift often not an option

To understand why applications can often not be moved from on premise infrastructure to a cloud environment without significant changes to the underlying software it is important to understand the technical differences between typical on prem environments and cloud environments.

	On Prem	**Cloud**
Runtime	Applications run directly on dedicated servers or virtual machines	Applications run within isolated containers distributed across machines
Network	Applications run within an environment protected by network rules that can only be accessed via VPN connection	Applications run in distributed containers that often need to communicate via the internet
Settings	Environment does not change unexpectedly, every configuration can be controlled	Environment can change over time, cloud offerings deprecate and not all settings can be controlled
Ressources	Amount of resources is determined by the hardware and cannot be changed short term	Resource allocation is dynamic, and containers (size & number) can scale up or down

Recovery	Self-healing capabilities are limited, failing hardware leads to outages	Self-healing mechanisms can automatically restart or replace failed containers
Deployments	Manual deployment processes, often involving downtime	Automated deployment through container orchestration tools like Kubernetes
Favored architecture	Monolithic applications with database in the same low latency network	Distributed architectures like microservices with more abstraction

Different environments allow or favor different types of applications and architectures. On premise applications were often built in monolithic architectures where all the application parts are packaged together into binaries and executables. They are often built very close to the databases as these run within the same network on the same hardware.

Cloud native applications are built with a flexible amount of resources in mind, often packaged into microservices that independently scale, abstractions that create adaptability to changing environment settings and more. Let's call these aspects soft limitations because they might make a lift and shift unattractive or inefficient but not impossible. However, there are hard limits that can make a lift & shift impossible.

Here are the most common limitations to consider:

Licencing & compatibility considerations:

- On prem architecture often requires licenses for (older) software i.e. oracle DBs or Java runtimes that are bound to a physical machine
- Not all (old) software has cloud compatible counterparts like hard dependencies on windows services that will be unavailable in most cloud container environments

Performance considerations:

- Your cloud database is now likely deployed separately (i.e. different service or container) adding network latency
- Communication is now more exposed to the web requiring additional authentication for each request which can add security latency

Security considerations:

- Per design a cloud environment is more exposed, you can not physically isolate your entire landscape behind network rules and firewalls
- Updates and changes (controlled by the cloud provider) can lead to security breaches and need to be monitored

Observability considerations:

- The things you want to / need to monitor to measure your applications health status are different on-prem vs. on the cloud

Breaking down monoliths - considerations & trade-offs

Once you have decided to move your legacy application(s) to the cloud it helps to consider the following aspects. While noting that there are no general best practices for structural infrastructure decisions - only trade-offs, let's discuss some of these trade-offs:

- How big should a service be? (Agility vs. Performance)
- Which functionality can/should reasonably be bundled? (Performance vs. Deployment effort)
- How do I decompose the data so that I can query the correct context when I need to? (Security vs. Performance)
- How do I manage/automate the deployment of now many smaller services? (Complexity vs. Efficiency)
- How do I make sure that one failing service doesn't take down the entire application? (Stability vs. complexity)

There isn't a one-size-fits-all solution or a single book that holds all the answers; the optimal choice varies based on numerous factors. Nonetheless, when executed correctly, the benefits are substantial. Only one thing is clear: **Mastering software coding is essential before reaping the rewards.**

Re-Enable developers to manage their own infrastructure

What is true as well: The advent of cloud-based resources transformed the code deployment process significantly compared to traditional bare metal deployment. This new process required in-depth knowledge of virtualization and containerization. Without these skills within the development team, the end-to-end software development process became fragmented. DevOps teams emerged as the essential blend of skills that bridged this gap. A costly compromise, which is time consuming, too.

Codesphere as a self-service platform is designed for developers. It simplifies the creation of virtual machines instantly using zero-config principles. We set out to make the entire deployment orchestration easy enough so that no specialized operations knowledge is required. Individual developers are empowered to deploy complex application landscapes (those typically requiring Docker & Kubernetes) without the burden of learning how to set up, maintain and operate such systems:

- Deploy as if locally, from a terminal, without the need to learn Kubernetes, Docker, Serverless, or Buildpacks.
- Enable provisioning (production) workspaces from the UI, handle routing via domains - staging, progressive releases, or A/B testing
- Seamlessly integrates with all large Git providers (GitHub, GitLab and Bitbucket) with a strong focus on efficient developer experiences
- Proprietary deployment algorithm that coldstarts resources in seconds - cost savings for low traffic apps and during autoscaling
- Hardware agnostic - clients can run on our data centers, on (private) cloud subscriptions & hyperscalers or on bare metal on-premise hardware.

Codesphere returns the end2end process authority to the development teams. And it allocates cloud resources smarter and faster, which leads to a higher density of virtual machines and reduces overprovisioning. In other words: **Bridging the deployment divide leads to quicker and more cost-efficient software innovations run in the Cloud.**