



ilionx

Application Modernization 2021

Luc Gorissen

Table of Contents

1.	Introduction	3
2.	Applications in the Cloud	5
2.1	Cloud Phase 1: Lift-and-Shift	8
2.2	Cloud Phase 2: Re-factoring	10
2.2.1	Cloud Native Application Development	12
2.2.2	Cloud Native Technologies	15
2.3	Cloud Phase 3: Application Modernization	19
3.	Digital Transformation and Application Modernization	24
3.1	Business opportunities	25
3.2	Organization aspects	30
3.3	Application Modernization success factors	33
4.	Summary	35
	About Rubix	37
	References	38

1. Introduction

Late 2011, a colleague pointed out AWS Cloud to me. It sounded interesting enough, so I got started and installed a middleware platform on it. I specifically remember being pleasantly surprised by the powerful servers that were at my disposal. When thinking about use cases, there were also some concerns about costs, network latency, security, and some more. Then, early 2012 I followed a training for the same middleware platform that I already ran in the AWS Cloud. On Monday, the training started, but the trainer unfortunately had some difficulties with getting the training environments up and running. So, Monday evening, I copied my environment and the remaining days of the week, we had four fully functioning training environments. All it took was some simple copying actions and \$60,- from my credit card. I was sold: running applications in the cloud was going to be big!

Now it's 2021 and the cloud is no longer raising eyebrows. Running IT workloads in the cloud is – from an infrastructure point of view - the main goal for most IT departments. Public clouds were 'invented' a bit over 10 years ago and some big promises were made. Running applications in the Cloud would make our IT lives much easier, flexible, less costly, more scalable,

etc. Also, the business department would finally get what they always wanted – with the right quality and in time. But, did they really?

Over the years, it became clear that running IT workloads in the cloud was different from running them in an on-premises environment. Especially for bespoke applications. Just doing a simple lift-and-

shift, i.e. running the application on 'a computer in the cloud' often turned out to be costly and more importantly, did not deliver the anticipated advantages. It became clear that applications which were to benefit from what the cloud had to offer, had to be built in a special way, with technologies that were developed especially for that purpose. The terms being coined were 'cloud native applications' that were developed using 'cloud native technologies'.

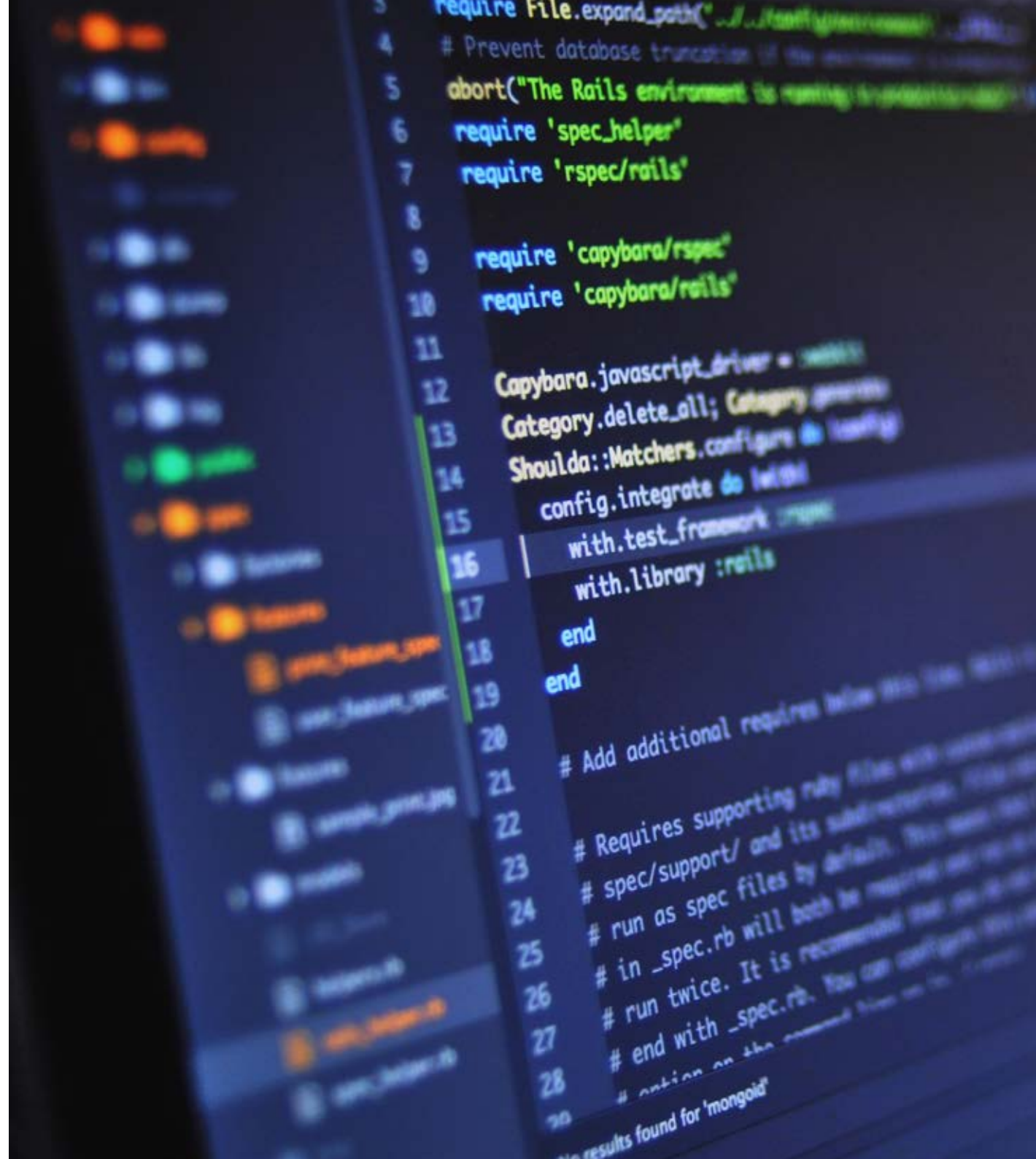
After it became clear how these cloud native applications had to be built, and the required technology was available, there was one remaining hurdle that had to be taken: how to deal with existing large bespoke applications? These applications often represented a significant investment, but were not fully suited to run successfully in the cloud. Fortunately, it turned out not

all such applications have to be re-built from scratch. Some will, but for others it is sufficient to only change them partially, in order to make them deliver their business value successfully in a cloud environment for many years to come.

But running cloud applications is not only an IT department thing: modern cloud applications and how they are developed can bridge the gap between business and IT – though that does require changes in how IT is embedded in an organization. Because a faster application development process requires close alignment between business and IT.

To assist you in finding your way on your Application Modernization journey, this white paper describes the strategies to modernize applications, i.e. to run them in the Cloud. Additionally, it sets the organizational context in which this can successfully be done.

Enjoy!



2. Applications in the Cloud

It was only in 2009^{1,2} that NIST gave their official definition of what 'Cloud' was and what to expect of it. Over the course of time, it became clear that most organization's IT can benefit greatly from the public cloud services. The cloud delivered its promises of lower IT spending, hyper scalability, increased speed and agility, lower operations costs, more focus on business value, etc. Including the birth of a completely new range of technologies that is also geared to maximizing the cloud benefits. Nowadays, many organizations have a cloud strategy in place that outlines how they want to take advantage of what cloud has to offer.

One of the topics that a cloud strategy has to address is how to migrate bespoke applications to the cloud. Already in an early stage, the options for migrating applications to the cloud were identified^{3,4}:

1. **Re-host:** make no changes to the application, but run it on a computer in the cloud. Also known as lift-and-shift
2. **Re-platform:** make minimal changes to the application by incorporating some of the advantages of the cloud. E.g.

replace an on-premises Oracle DB by an Amazon RDS DB

3. **Re-purchase:** replace the application by another application
4. **Refactor:** completely rebuild the application, in order to have a better fit with the possibilities the cloud offers
5. **Retire:** abandon the application
6. **Retain:** keep the application in the on-premises data center, usually with the intention to abandon it.

Which of the above options is chosen is a lifecycle question, and its answer should match with the business requirements. Lifecycle management topics are often difficult to deal with, as they usually are interwoven with an organization's historic vendor choices, sourcing policies, engineering culture, etc. This may complicate a discussion that should be centered around how IT solutions can meet the business requirements. To structure the lifecycle discussion, the application and its technical realization must be separated:

- **Application lifecycle:** will the application be able to meet future business requirements? Especially for bespoke applications, it is important to understand how the current technical state influences the feasibility to incorporate future requirements.

Examples of complicating factors are 'bad design' and 'technical debt', which both may complicate further development work on the application.

- **Technology lifecycle:** what is the roadmap of the technologies involved in running the application? For example a technology stack that is no longer supported may require a partial application refactoring.

» **The application and technology lifecycle, together with input from all involved application stakeholders, are the input for defining an optimal application migration strategy.**

In the early days of cloud, execution of an application migration was difficult, because no best practices existed on how to best execute the various migration strategies. Over time, these best practices have arisen and evolved. We distinguish three phases that each contributed a certain method:

Cloud Phase 1: Lift-and-Shift

The focus of organizations was on migrating existing, on-premises IT to the Cloud. The migration of bespoke applications was 'part of the IT infrastructure migration to Cloud'.

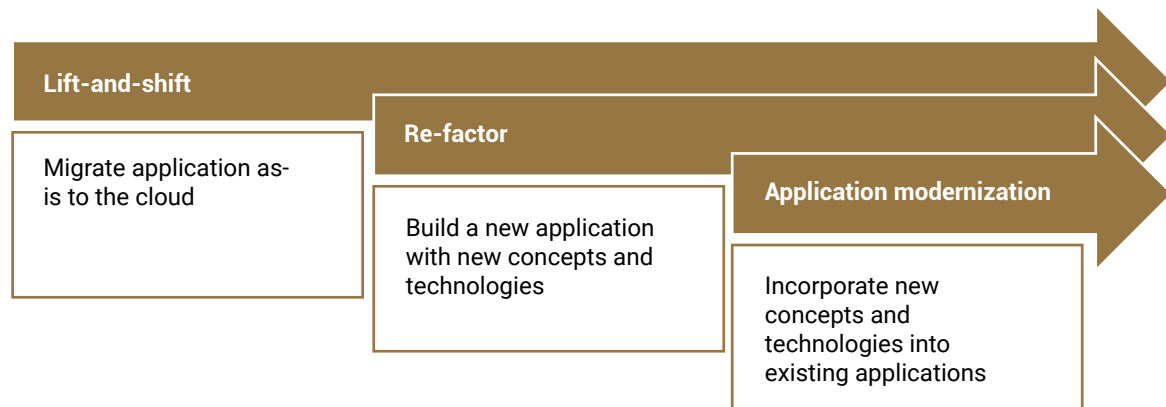
Cloud Phase 2: Refactoring

It became clear that a mere as-is migration of bespoke applications to the cloud did not deliver on the cloud promises. Focus with respect to bespoke applications shifted to refactoring: building the application from scratch – with new cloud technologies - in such a way that it

maximally benefits of what the cloud has to offer. This is also referred to as cloud native application development.

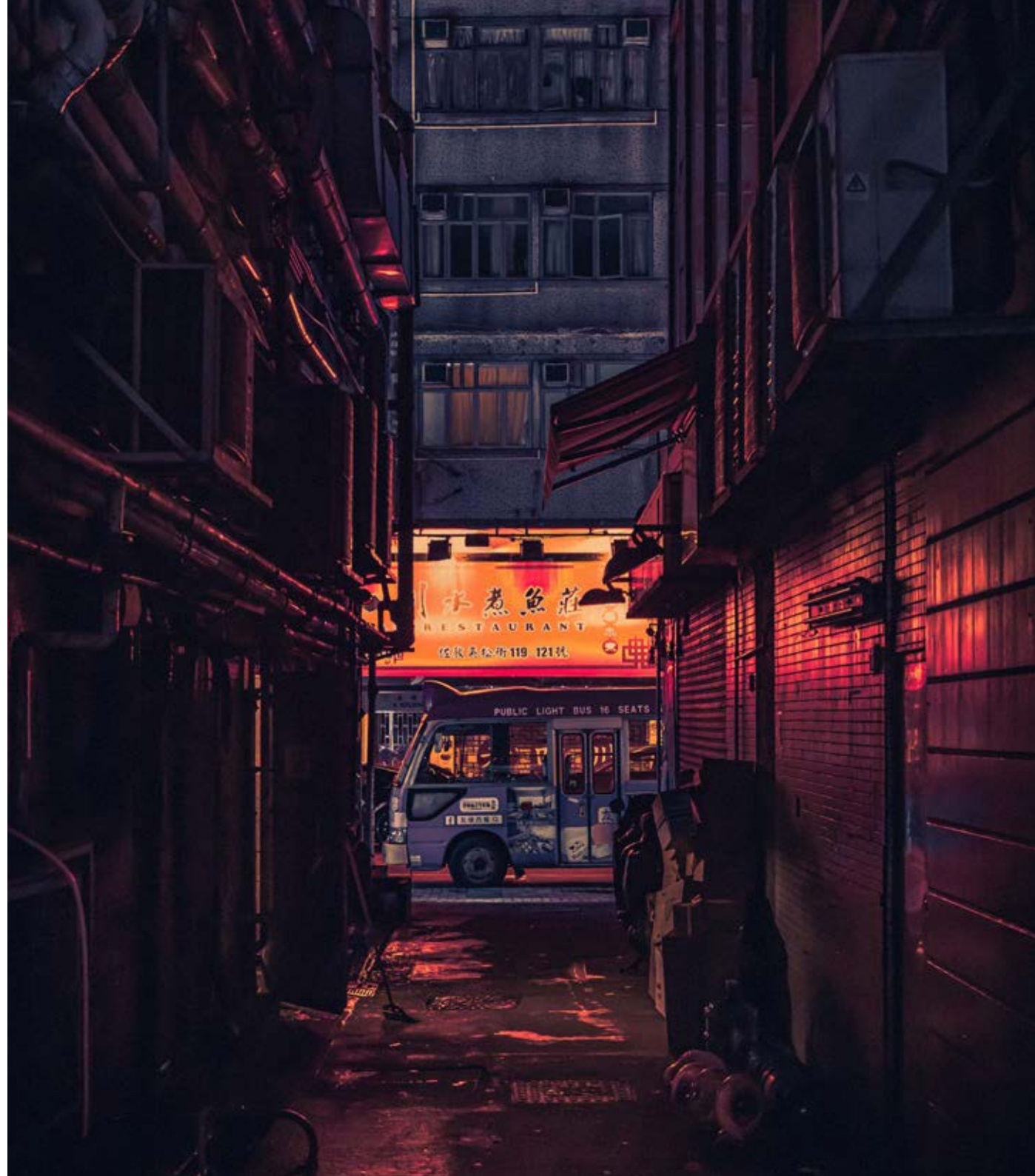
Cloud Phase 3: Application Modernization

The focus is currently shifting to the larger applications, that often implement the core business processes and represent a large investment. These applications are often complex, for example because of added functionalities and integrations, and can therefore not be refactored easily. As a consequence, future business requirements may be hard if not impossible to implement.



In such situations, a tailored strategy can be designed in which the application evolves in steps wherein new business requirements are implemented. These steps are – where necessary – realized in accordance with the modern way of building cloud applications. This results in a partially modernized application– only in the parts where it is justified by business requirements. In doing so, the investment in the application is protected as much as possible.

Do take note that depending on business requirements, a lift-and-shift migration to a public cloud or a complete refactoring are still valid options. As well as the other options from the 'original' list of six. The best choice depends on your organization's goal and context.



2.1 Cloud Phase 1: Lift-and-Shift

In 2006, Amazon launched its Elastic Cloud Service. Other parties like Google and Microsoft followed and it was the NIST cloud definition in 2009/2011^{1,2} that helped us understand what we were to expect from 'the Cloud'. The cloud paradigm was quickly embraced by both providers as well as companies. Providers like Amazon and Google had a head start. Amazon, because they were visionary: they had excess computing resources from their webshop, and decided to offer theirs to other companies. Google also got off to a quick provider start because of their experience with large, world scale implementation of applications. Other companies quickly followed with acknowledging the potential of cloud computing and embracing the concept. Some companies, e.g. Dropbox and Netflix, provided services that could not have been realized without cloud

computing. Other companies set their first cloud steps by using a non-business-critical SaaS application, e.g. an HR application.

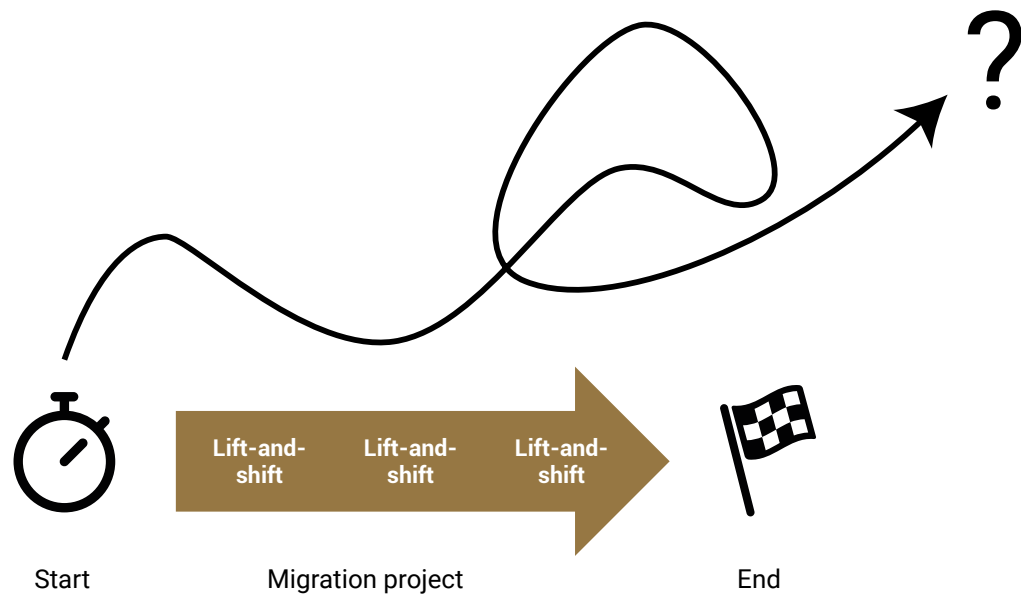
By now, after the first decade of cloud computing, most companies have cloud incorporated into their IT strategy, often with the goal of moving all their IT into the cloud and abandoning their data centers. But, what also became clear is that running existing applications on cloud computers will most likely not realize all of the expected cloud benefits. For example:

- 'that Cloud DB with lower costs' is never realized
- running an application in the cloud does not make it a better fit with the future needs
- an application does not have the

horizontal scalability that was expected

- introduction of new business functionality into an application is still difficult ...

However, an organization that wants to move all applications to the Cloud quickly, can't be bothered to design an individual strategy for all its applications. The sheer amount of applications, their interactions, the application complexity and peculiarities will be too much to handle in an already complex program to move to the Cloud. When confronted with this, organizations often change their strategy into what is best described as 'let's first re-host all applications in the Cloud and then take it from there'. This results in a more manageable project that can be completed in a more acceptable time.



And this is the exact situation where many organizations find themselves in nowadays: they have migrated large chunks – if not all – of their IT to the cloud, at high costs. Only to discover that they have high monthly cloud costs and not the expected benefits they thought of in their original business case. All of these benefits of lower IT spending, hyper scalability, increased speed and agility, lower operations costs, more focus on business value ... are only partly met. I ask you: what else is to be expected when you run your existing applications on someone else's computer!?

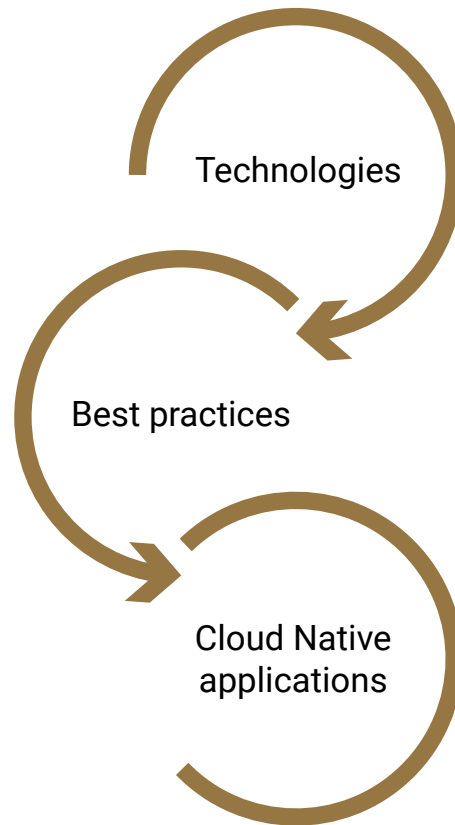
2.2 Cloud Phase 2: Re-factoring

Applications that were built to run in on-premises data centers don't automatically benefit from what the cloud has to offer. For example, they may not be designed for horizontal scalability. Or a switch to another database may be very hard, if not impossible. Or their internal design may make it hard to include new functionality. Or their accompanying development tools and development life cycle may not be up to the demands of quick and iterative software development. Or [fill in one of many other reasons].

Building applications that truly benefit from what the cloud has to offer, aka refactoring, was at first only achievable for companies like Google, Netflix, Spotify, Facebook, SoundCloud, LinkedIn, Uber, Zalando, etc. These companies not only have extreme requirements, but also the means to develop applications that meet these requirements.

Often, these companies had to develop new technologies, like for example LinkedIn, that developed Kafka when they needed a high throughput event processing platform. These technologies are known as cloud native technologies: technologies that are used to build applications that unleash the cloud's potential. Technologies that are built to live in a cloud environment. When you build a modern cloud application, your application platform naturally will consist of technologies that didn't exist 10 years ago. Another big change with respect to technology comes from the cloud providers, who are making cloud services available as 'managed cloud services'. This shift means that the cloud provider not only offers specific technology platforms, but also will do the corresponding cloud operations tasks, up to and including Lifecycle Management. Using technology





as a managed service in your IT solutions is therefore much faster and easier.

Of course, changing technologies is not the only reason for refactoring gaining in popularity. There is also a better understanding of how to build applications that make optimal use of the possibilities offered by the cloud. The best practices for cloud native applications are well understood by now.

Conclusion: with a better understanding of how to develop cloud native applications and with the right cloud native technologies at hand, application refactoring became a viable option for more and more organizations!

2.2.1 Cloud Native Application Development

Google, Netflix, Spotify, Facebook, SoundCloud, LinkedIn, Uber and Zalando are typical examples of companies that build Cloud Native applications with extreme requirements on scalability, data processing volumes, availability, etc. Their cloud applications impacted the way cloud applications are developed nowadays. It resulted in what is often referred to as Cloud Native Application Development, which is defined as:

The development of bespoke (enterprise) applications that make the best possible use of the possibilities that modern cloud technologies have to offer.

Meaning, cloud native applications must deliver business value: quick, at low cost and of required quality, by making the best

possible use of what the cloud has to offer. Typical cloud native applications have the following characteristics:

Low cloud resource usage


Modern cloud native technologies all focus on a small resource footprint: low usage of memory, cpu and network. They do so in 2 different ways. On a platform level, there is on-demand scaling, both scaling out and scaling in. The more demand for capacity, the more resources are claimed by the platform: scale out. When the demand for capacity decreases, those resources are released: scale in. This scaling out/in is preferably highly automated. On top of that, the new emerging framework technologies to build the application's business logic have considerably smaller footprints. A typical examples is Kubernetes (platforms) and Micronaut (framework).

Flexibility for change

Cloud native applications and their technologies aim for distributed applications that consist of parts – microservices or functions – that are maximally independent of each other. This is achieved by pursuing a maximum separation of concerns, which on a system level often translates to a Domain Driven Design approach^{12,11}. As a consequence, the various application parts can be easily replaced by a modified version.

High developer productivity

The cloud native frameworks strive to enable developers to focus more and more on implementing business logic instead of coding boilerplate stuff. Also, the availability of managed cloud services reduces the dependency on system administration, thus further improving



developer productivity.

Furthermore, development best practices like CI/CD and automated tests further improve the ability to quickly deliver new application functionality whilst staying in control.

Hyper scalability

A key characteristic of cloud native applications is horizontal scalability. But having the right platform and framework technology in place is not enough: the application must be built as a distributed, stateless application where the different parts can scale independently according to the capacity demand. A typical scalability bottleneck are ACID transactions on a traditional relational Database. Elimination of ACID transactions requires the adoption of the BASE – a data consistency model where the data in the system is eventually consistent across the whole system. Refactoring an application that relies on ACID database transactions

to an application that relies on eventual consistency of data has a major impact on how the application is built, and hence on the developer's work. This is also the main reason why existing current applications often can't easily be made to scale horizontally.

Reduced operations effort

The operations effort for a distributed application is higher than for 'a monolith application with its single application log file'. This is addressed in many different ways, mostly focusing on more and more automating of operations tasks. An obvious example is the introduction of managed cloud services, where a lot of the operations work is shifted to the cloud provider. Also, operations tasks are highly automated for areas like CI/CD, automated tests and observability.

More robust systems

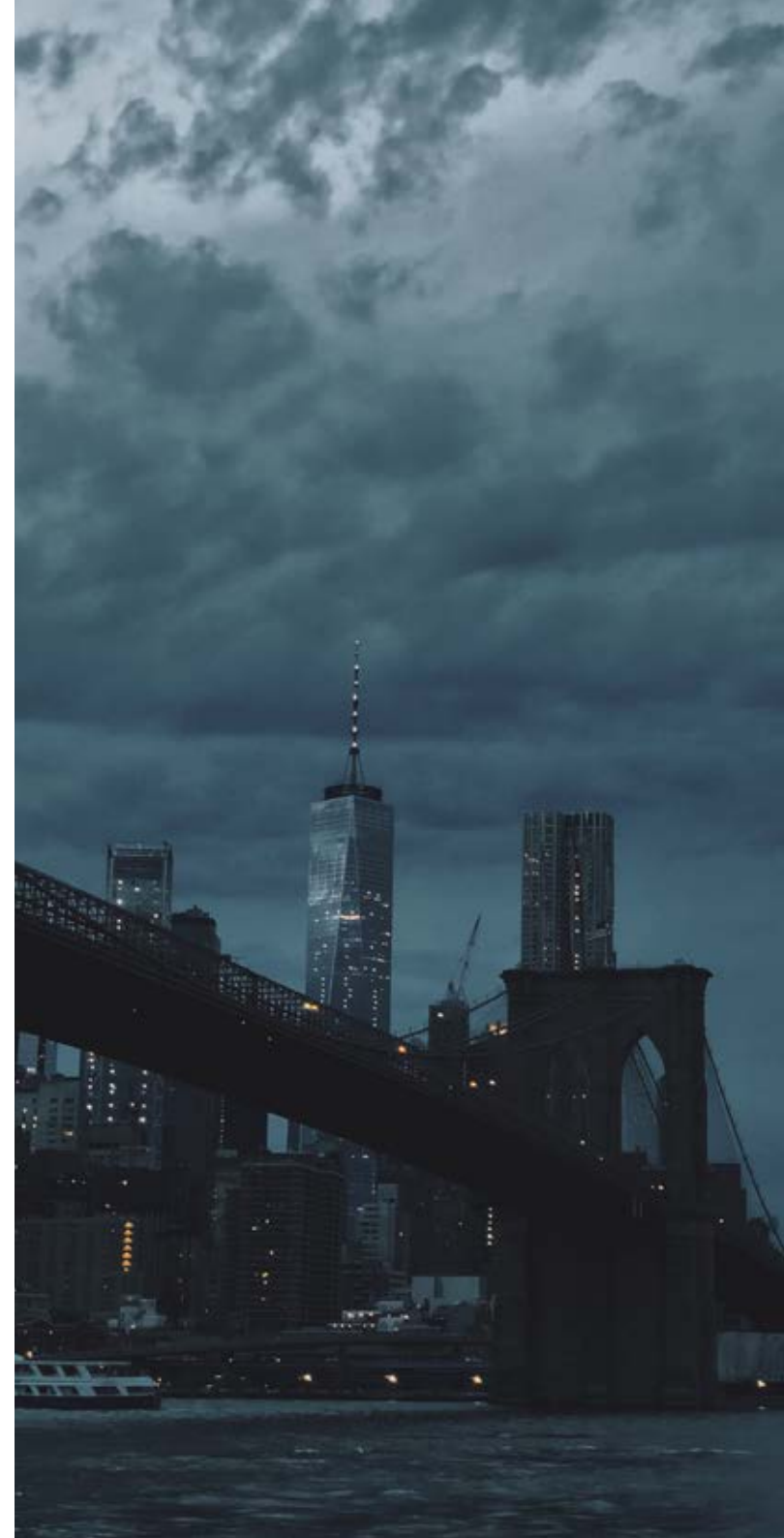
Cloud native applications are built in such a way that failures of application components should not lead to a complete failure of the whole application, but only to defaulting specific parts of the application. This, together with platform capabilities for auto-restart and auto-healing of applications, can significantly contribute to application functionality availability.

New cloud technologies

New cloud technologies are often available as managed cloud services and therefore easy to incorporate in applications. Typical examples here are AI, face recognition, IoT platforms etc. All available with just a few mouse clicks and a credit card. And if no longer needed, they're disposed of just as quickly.

Mind you, these characteristics do not magically result from picking the right technologies: it is crucial that they are

incorporated in the design and building of the application. This is often referred to as Twelve-Factor Apps¹³.



2.2.2 Cloud Native Technologies

The previous sub section touched upon the characteristics of cloud native application development. In order to properly develop applications with these characteristics, an enabler is needed: Cloud native technologies. Technologies that are built to live in a cloud environment, like Kubernetes, Containers, Kafka, GraphQL , GraalVM, Serverless functions, DynamoDB, Jaeger, Istio, Prometheus, Terraform, etcd, Helm, Micronaut. As mentioned, when building a modern cloud application, your application platform will consist of technologies that didn't exist 10 years ago. These technologies can be categorized in three service models.

IaaS, PaaS and SaaS

The NIST cloud definition¹ distinguishes these three different service models to recognize what capabilities are provided to

the customer:

1. **Infrastructure as a Service – IaaS:** provides computing resources like processing, storage and networks;
2. **Platform as a Service – PaaS:** provides programming languages and tools that can be used to deploy applications. Many consider PaaS to be THE application developer's toolbox.
3. **Software as a Service – SaaS:** provides applications.

What service model(s) an organization decides to use, is an important decision. For example, focus on the IaaS service model will limit vendor lock-in. But using advanced PaaS building blocks may speed up application development and reduce operations effort significantly. And the SaaS service model could very well be part of a buy-before-build approach. A

detailed discussion about how Enterprise Architecture principles guide the choices for the cloud service models is outside the scope of this paper, however, specifically relating to application migration to the cloud:

- **IaaS** is mainly used in the **lift-and-shift** of an existing application to the cloud.
- **PaaS** is mainly used for **developing bespoke applications**.
- **SaaS** is mainly used when **replacing** an existing application with an application that the cloud provider offers.

PaaS building blocks

Many cloud providers have a PaaS offering with many application technologies. Technologies that once would have taken a long time to implement in an on-premises data center are now readily available.

Examples are:

- Container platforms like Kubernetes
- Artificial Intelligence (AI) and Machine Learning (ML)
- Virtual and Artificial Reality
- IoT platform
- Integration platforms and API Gateways
- Data streaming solutions like Kafka
- Observability solutions like Elastic Observability
- Data analytics and search engines
- Content Delivery Network solutions
- Databases
- Face recognition
- Blockchain
- Digital assistant – chatbot

When such technologies are needed for an application, using readily available PaaS offerings should be considered. Otherwise, applying the desired tech can prove too complex or too costly to handle. In general, using as many PaaS offerings

as possible – or managed cloud services as they are often referred to – makes sense. Picking the right PaaS services will enable the application developer to focus on business logic and minimize time spent on developing boilerplate code. And, of course, delegate a lot of operations and lifecycle management tasks to the cloud provider.

The accessibility to PaaS services is continuously being improved, partly because many cloud native technologies are maturing themselves, which translates to more and better support for non-functionals. Simultaneously, the cloud providers are improving the accessibility of their PaaS building blocks, often by adding abstraction layers or combining PaaS building blocks. The most advanced PaaS building blocks are nowadays labelled as ‘serverless’, i.e. a PaaS building block where the underlying hardware is no longer visible to the consumer but completely

managed by the cloud provider.

Example

A typical example of a well known cloud native technology is container management platform Kubernetes. The Kubernetes platform is currently evolving into a direction where it focusses solely on ‘managing containers’. Other, non-core parts are abstracted through the introduction of various APIs:

- CRI – Container Runtime Interface: to support Container Runtimes other than Docker
- SMI – Service Mesh Interface: to support different service mesh implementations
- CSI – Cluster Storage Interface: to support various types of storage
- Cluster API: for managing Kubernetes clusters



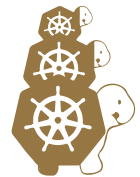
Container Runtime Interface



Service Mesh Interface



Cluster Storage Interface



Cluster API

As a result, AWS for example can offer a Kubernetes platform that:

- supports AWS Firecracker runtimes,
- uses the AWS App Mesh as service mesh,
- uses AWS Elastic Block Storage,
- and is managed within the AWS cloud via the Cluster API.

Then you also have RedHat OpenShift; a Kubernetes platform that:

- is currently moving from Docker to CRI-O container runtime,
- uses the Istio service mesh,
- uses amongst others RedHat Ceph storage,
- and is managed via the RedHat OpenShift console

However, these two examples are not complete application platforms. To use them as an application platform, additional components have to be added, like tooling

for Observability, CI/CD, automated testing etc. RedHat OpenShift already contains some pre-selected add-ons and integrates those into the OpenShift building block. Whereas e.g. the Oracle Container Engine for Kubernetes, is a bare Kubernetes offering where the developer has to add all of these components himself and do lifecycle management for the complete combination.

Kubernetes is nowadays the de-facto standard for container management platforms. However, there are cloud providers that offer simple, managed services for running containers. Examples are Azure Container Instances and AWS Fargate. These managed services do not offer all of the Kubernetes functionalities, but they can run containers. And the cloud provider manages the underlying Kubernetes and hardware. Simple and easy accessible!

In short, the selection of a PaaS building block for running containers is not straightforward and requires a careful selection process. The next paragraph is meant to help you on your way in this process.

PaaS building block selection

When developing an application, the right PaaS building blocks have to be picked: they must support the cloud native application design. Technology is a key enabler!

Using PaaS building blocks to build an application often results in adopting a multi-cloud strategy. This is because PaaS building blocks differ greatly from one public cloud provider to the next. E.g. one

cloud provider may have an excellent AI/ML offering, whereas another provider has a great IoT platform and yet another has a great chatbot. With a best-of-breed strategy, this will lead to a multi-cloud solution. It should then be carefully considered whether the expected benefits outweigh the added complexity of a multi-cloud solution.

One more thing about refactoring

A fair word of warning is in order. Developing cloud native applications is more than just 'putting the right application design and technology in place'. Some considerations are:

- Developing such applications requires a skill set that is likely new to people.

- Application scopes need to be carefully set, i.e. applications that cross organizational borders may be severely impacted in their flexibility by having too many stakeholders that need to be consulted when making decisions. A typical consideration is to let IT and organization be a mirror image of each other, as described by Conway's law⁵.
- A shift to a more DevOps style of working can be a large step for an organizations.

Especially organizations that are new at cloud native application development must take these considerations into account.



2.3 Cloud Phase 3: Application Modernization

Over the last years many best practices were developed around how to migrate applications to the cloud and how to build new applications for the cloud. But that does not address the issue of the huge installed base of bespoke applications that – when moved to the cloud – do not benefit from the cloud. However, there is an increasing understanding that these applications can often be modernized. Such Application Modernization updates legacy applications by replacing or adding new cloud concepts and technologies to the existing application. Thus, application modernization adds new business value to existing applications whilst protecting the current investment.

Strangle the monolith

A good approach to the modernization of existing applications is using the

Strangler Fig pattern¹⁴. The pattern was first described by Martin Fowler – way back in 2004 already! The approach the Strangler Fig pattern takes is that specific application parts are extracted from the monolith application and implemented as separate microservices. This involves the following steps¹⁵:

1. Design the application

A to-be design of the application is required. A design that ensures that the application has optimal flexibility for handling future changes. A best practice to make such a design is to apply the concepts of Domain Driven Design (DDD)¹².

2. Choose an application part to modernize

A specific part of the application that has to be modernized should

be selected. Preferably, the selected application part has limited dependencies on the other parts of the application, thus limiting the technical risk of the modernization step. In general, there will be a trade-off between business value and technical risks. It is important to strive for a good balance between the two.

3. Plan out the microservices within the selected application part

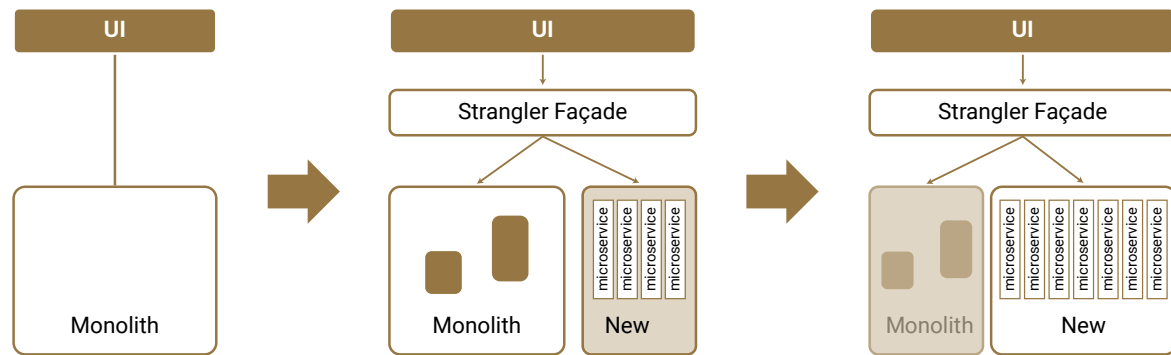
Design the microservices that are to be built for the selected application part, including how their data will be decoupled from the data store(s) of the monolith.

4. Plan a Strangler Façade

Planning for a Strangler Façade means that an implementation pattern must be designed that describes how the new microservices will co-exist with

the monolith. This should cover how the new implementation of the selected application part will be introduced and co-exist with the existing implementation. The next step is planning how the existing functionality is best removed from the monolith.

These steps are illustrated in this figure:



While the chosen architectural approach and technologies can differ, the steps themselves clearly show that Application Modernization can be done in a step-by-step approach. It is important to point out that Application Modernization can stop when no further modernization steps can be identified to create additional business value. This ensures an optimal protection of the investments that were already done in the monolith.

Application Modernization is not only about replacing existing functionalities with a new, better implementation. The described approach can also be used to add new functionalities to an already existing application.

Cloud native application characteristics in Application Modernization

In the section on developing cloud native applications, the characteristics of cloud native applications were listed.

When a new cloud native application is developed, all of these characteristics can be met, when taken into account from the start. However, modernizing an existing application that does not have all of these characteristics is different. It makes no sense to just blindly pursue all of these characteristics. It only makes sense if the effort is justified by the business value that it yields:

- **Lower resource usage**

For a simple enterprise web application with 20 users, resource usage is not that big of a deal. The costs of a couple of Virtual Machines with a public cloud provider will by far outweigh the costs of modernizing the application. However, things are different when it concerns an application that is offered – as a SaaS application - to many customers. Then, the resource consumption really starts to matter, as it is directly taken out of your profit. Imagine a lot of VMs doing nothing outside of office hours... Similarly, if an application has to deal with spiky or unpredictable workloads. The parts of the application that cater for most of the unused Cloud resources can be addressed by refactoring them into microservices that have a better resource usage profile.

- **Flexibility for change**

When an application is changing rapidly, and when multiple teams are working on

the same application, the application must easily absorb changes. This capability can be largely improved by breaking out the rapidly changing parts and implementing them as microservices. These microservices should have a single responsibility and clear boundaries, something called a bounded context in Domain Driven Design¹². The implementation of the microservices should then be done in such a way that deployment of new versions all the way into production is largely automated. Including automated tests.

- **High developer productivity**

Developer productivity is greatly enhanced by using the newest cloud native technologies. They ensure that developers, more than ever, can focus on business logic, instead of spending much time on boilerplate code. Examples of application parts that may benefit from a replacement that yields higher developer productivity, are 'that old UI framework' or 'that

homebrew business rule engine'. The first can be replaced by a low-code platform front-end, and the second by a PaaS Rule Engine offering by your public cloud provider. If these technical components have a lot of future implementation work planned, replacement may be worthwhile.

- **Hyper scalability**

Best practices show that it is best to start an application as a monolith, which is then broken up into microservices when the application user base keeps growing⁹. The advantage of this approach is that the application's behavior is already understood, making it easier to identify the right microservices needed to be split off. Still, there can be good reasons to deviate from this best practice. Maybe the user base starting from day 1 is known and very large. Maybe the expected user base is not known at all, but the application may need to scale up within a period of days. Or the workloads are very spiky by nature.

Reduced operations effort

Similar to ‘high developer productivity’, the newer cloud native technologies also aim to reduce operations effort. Parts of an application may greatly benefit from modernization steps that bring better monitoring, better CI/CD capabilities, more flexible scaling, etc. to the table. For example, there can be significant business value in being able to often release new functionality or to release functionality to a limited user base. In this case, it makes sense to move to a CI/CD implementation and application platform that supports blue-green and canary-in-the-coalmine deployments.

From an operations effort point-of-view, it is important to understand the managed services that a cloud provider offers. The aim with these services is to shift as much of the operations work as possible to the cloud provider. The more managed cloud services can be incorporated into an application, the lower the operations effort. Beware that there may be a trade-off

with managed services. Often, offloading operational tasks also comes with constraints on how the managed service’s technology can be used.

More robust systems

With distributed applications, the failure of one of the components may likely only lead to a partial application failure. Whether that aspect is important for a specific application depends on the application functionality and the business impact of complete application failures. Imagine a webshop application having a failure that affects the tracking of placed orders. If that failure would result in a complete application outage, it would have great impact. If, on the other hand, only the part of the application for tracking placed orders would not be available, but customers can still shop and place new orders, the impact would be significantly less. You need to determine if this is important for your application. If yes, it should be considered to break the

application into separate components that can be run on a platform with auto-healing capabilities like Kubernetes. Note however, that a robust system requires more than ‘just running it on Kubernetes’. In our example: the other application parts must be able to handle the failure of the ‘order tracking’ part. Building such application landscape requires an overall strategy to handle these partial failures.

New cloud technologies

New cloud technologies can contribute to business value in several ways: new functionality can be implemented much faster, more advanced technologies suddenly come within reach, and the cost of failures is minimized as no large up front investments are required. This, together with collected best practices from the early adopters, will make adding new technologies to an application more feasible than ever.

Application Modernization or rebuild?

Application Modernization makes sense in a lot of situations: only changing some specific aspects of an application may offer the business their required value. On the other hand, a complex Application Modernization process may in the end be more costly than a complete rebuild from scratch. The answer to this question is not an easy one. It all centers around what the future of the application looks like from a business requirements point of view: the application roadmap. When this roadmap is clear, different Application Modernization steps and even a complete rebuild can be evaluated, eventually resulting in an optimal approach for a specific application.

An Application Modernization example

A good example on Application Modernization is shown in [this video](#)¹⁰. This example was presented on AWS re:Invent and concerns the Amazon shopping application.



This presentation covers two Application Modernization steps:

1. **Flexibility for change:** breaking up the application into microservices in order to be able to incorporate changes much quicker.
2. **Reduced operations effort:** Moving from a self-managed Kubernetes platform to a managed service platform, i.e. AWS EKS.

Some takeaways:

- Even a multinational as the AWS shopping branch of Amazon has only recently (2017) broken up their monolith shopping application into microservices.
- Also this multinational found Kubernetes a too difficult platform to manage by themselves. It took away focus from the actual goal: improving the shopping application. Hence, they decided to switch to a Kubernetes platform as a managed service from a cloud provider – obviously AWS EKS in this case.

3 Digital Transformation and Application Modernization

An important aspect that has not been mentioned so far is the organization. The optimal scenario for Application Modernization or even a complete rebuild should also be a scenario that your organization can handle. Can you quickly adopt new technologies? Can you handle quick deliveries of new versions of an application? It's important to consider the bigger picture here, with the bigger picture being Digital Transformation.

A formal definition of Digital Transformation is 'the adoption of digital technology to transform services or businesses, through replacing non-digital or manual processes with digital processes or replacing older digital technology with newer digital technology'¹⁷.

In practice, for your organization it may look like this:

- The (IT) world is rapidly changing, partly driven by the impact that cloud

has, and it's difficult to establish the right way forward in business and organizational matters.

- The business sees an ever increasing number of opportunities, but also a corresponding number of new threats.
- Cloud technologies are more powerful than ever, but applying them the correct way is complex.
- Your (IT) organization is working harder than ever to keep up and adapt where necessary.

Application Modernization – as it is all about adding or replacing parts of existing applications with new cloud concepts and technologies – clearly has its place in a Digital Transformation program. So in this section we will address the business and organizational aspects of Cloud and more specifically Application Modernization. Finally, some success factors are shared for Digital Transformation programs that are linked to Application Modernization.

3.1 Business opportunities

It is important to realize that the current IT technology trends are more than a mere replacement of some technical components by better ones. The new cloud native technologies are part of what is described by the World Economic Forum as ‘the Fourth Industrial Revolution’¹⁸. The Fourth Industrial Revolution is building on the Third – the digital revolution. What makes it different is that a fusion of many technologies – physical, digital and biological – is happening at an exponential pace. It wouldn’t be a revolution of course if it wasn’t disruptive, but the most striking part is the velocity at which the disruptions happen!

The new cloud native technologies are a part of this Fourth Industrial Revolution and often serve as a technology push in general for new business opportunities. Examples of

some relevant IT technology trends are⁶:

- **Hyperautomation**

According to the Gartner definition, hyperautomation from technology point of view ‘deals with the application of advanced technologies to increasingly automate processes and augment humans’. Hyperautomation is about Robotic Process Automation (RPA) and intelligent Business Process Management Suites (iBPMS) where the traditional business process automation is extended with software robots (bots) or AI. Hyperautomation is also about including the information from a Digital Twin Organization²⁰ in the application. It is this combination of technologies that makes new applications possible. However, such new applications will have a huge influence on how employees interact with them.

Therefore, they can only be introduced successfully when employees are involved from the start.

- **Multiexperience**

Developments like Artificial Reality (AR), Virtual Reality (VR) and Mixed Reality (MR) will have more and more impact on how we interact with the digital communications world around us. Simultaneously, input and output devices are changing at rapid pace. Imagine Apple stopping the production of iPhones in a couple of years and replacing it with ...?

Huge changes on how we interact with both systems and other humans are expected and some are already here. For example, product visualization in real estate and webshops or VR-based training are not hard to imagine.

- **Edge computing**

Edge computing – and fog computing – refer to solutions where (part of) the application functionality is moved to edge devices. These types of distributed solutions were originally driven by IoT solutions that needed to avoid latency and need for bandwidth. However, the edge devices keep evolving into drones, autonomous vehicles, robots, ... bringing new applications within reach. These solutions are also redefining the cloud as we know it: at first, the cloud was centralized, but edge computing has decentralized it.

- **Distributed cloud**

When we refer to ‘the cloud’ in this paper, we are mostly referring to Public Cloud services, i.e. services that are offered from a Public Cloud provider’s data center. The term Private Cloud refers to services with cloud characteristics like e.g. scalability, but then offered from a company’s private

data center. Hybrid Cloud refers to an integration of both Public and Private Cloud services.

Implementation of Private Cloud services with all expected cloud benefits proved to be hard. Implementation of combinations of technologies in a Hybrid Cloud also proved to be hard.

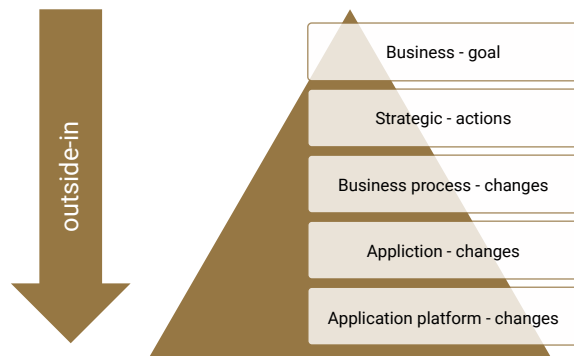
This has led to the Distributed Cloud, in which Public Cloud Services are offered to locations outside of the Public Cloud provider’s data centers. The Public Cloud provider then still executes operations tasks like maintenance and updates. Advantage of this solution is that requirements like on data location and on latency can still be met, without compromising the other Cloud advantages. Distributed Cloud, e.g. AWS Outposts, technically delivers on most of the requirements of a Hybrid Cloud approach.

Obviously, this list is far from complete. Trends like Internet of Things, Artificial

Intelligence and Machine Learning are not mentioned, but are expected to have a significant impact in the years to come. And then, newer trends like Quantum Computing are quickly evolving, and will offer even more possibilities.

These technology trends offer unprecedented opportunities and it is tempting to consider many disrupting business services as being the result of a technology push. However, there is more in play here. More and more companies are, for example, moving from an inside-out to an outside-in marketing approach¹⁹, which has a profound effect on the requirements of their application platform(s). Basically, an inside-out approach focusses on improving the company’s strengths that are already there. A huge drawback is that it has more focus on the current business than on what is needed for the future business. The outside-in marketing approach starts with the customer’s needs

and translates these to the business goals. The outside-in model will incorporate innovations that align with customer needs in a more natural way. These innovations will lead to new business goals and the corresponding strategic actions. In turn, these strategic actions then bubble down via business processes to applications - running on an application platform.



The application platforms have to accommodate all these changes and innovations:

- faster than before

- with better user experience
- with better quality
- with an open and interoperable digital ecosystem

As a result, organizations must build applications and application platforms that match with their business needs. As mentioned in the introduction, the link between business and IT is getting tighter and tighter. Which, not unimportantly, in many organizations reflects by moving IT budget directly to the business.

Application Modernization and the Business

Most organizations have a – sometimes large – investment in bespoke application platforms. These were developed to fulfill business requirements that were needed at a certain point in time. Over time, the world has changed and at some point, the match between business requirements and application functionality becomes

insufficient: a gap that needs to be addressed. There are several ways to do so. Rewrite the bespoke application from scratch is an option. Or, perhaps currently a COTS (SaaS?) application is available that meets the business requirements. But both of these options may result in major application implementation projects and they both do not protect the investment in the existing bespoke application. In many cases, application modernization may be an appealing business option for updating the current, existing application.

Changing business needs can pose application challenges that Application Modernization can address:

- **Vendor lock-in / application lifecycle management**

Migration of (parts of) an application from an old or even obsolete technology stack to a more modern technology stack.

- **Scalability**

Increase scalability of (parts of) an application so it can handle larger workloads.

- **Agility and maintainability**

Increase agility and maintainability of an application by breaking it up in mutually independent parts that are hence easier to change and maintain.

- **Robustness**

Increase robustness by applying the appropriate Cloud technologies and optionally breaking the application up in mutually independent parts, i.e. parts that can operate/fail without affecting each other.

- **Advanced functionality**

Adding new application functionality that uses new technologies, e.g. adding a Virtual Reality capability.

- **Security and privacy**

Security and privacy demands are getting more important, and older applications may not be able to keep up with these demands.

- **Rebuild**

Rebuild (parts of) the application to optimally use the Cloud advantages e.g. for optimizing costs, better security, monitoring, etc.

Application Modernization – as previously explained– usually is a process where an application is changed – modernized – in multiple steps. These steps will not only avoid big-bang approaches, but will also allow for focus on the business needs. A step-by-step approach will ensure that each step delivers the business value most needed at that moment.

By now, it is clear that Application Modernization protects investments in

a bespoke application. However, most organizations do not really establish the value of their existing IT systems, let alone base their decisions on it. The decisions for Application Modernization are mostly made by the business, and are heavily influenced by velocity: Application Modernization can often deliver solutions in a shorter term.

However, some application challenges can't be addressed by Application Modernization. Therefore, a thorough analysis of the application roadmap and the current application (technical) state must be done up front, in order to understand what exactly Application Modernization can offer for a specific application.

An example – the oil terminal

A typical example of Application Modernization starts with an Oil Terminal that has an Oil Terminal Management

application that keeps track of all product movement and what product is stored in an oil tank. The Oil Terminal Management application is a bespoke application that has many specific features for the Oil terminal and hence represents a large investment and – also important – it cannot be simply replaced by a COTS Terminal Management application.

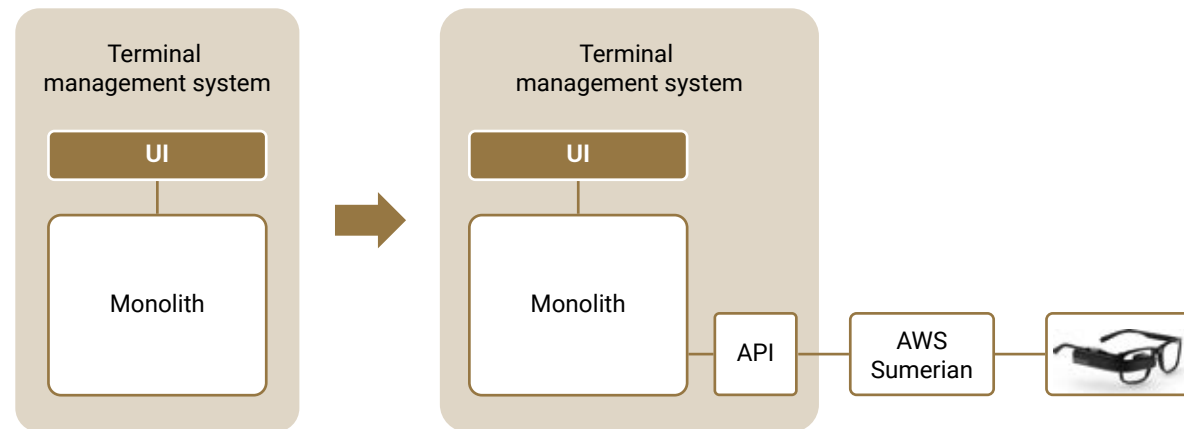
Now, a business initiative is started that aims to increase safety on the Oil terminal, and one of the ideas is to provide field workers with glasses that they put on when starting their work on the Oil terminal. When they look at an oil tank, the glasses would display information about what is stored in the oil tank.

The idea in itself sounds feasible, but the solution is not straightforward. The new functionality is linked with the existing Terminal Management system for its data, process states and authorization

mechanisms. Simultaneously, it is clear that implementing even more new functionality in the already existing application and its outdated technology stack would not be feasible. It would simply be too complex and costly.

As an alternative route, an application modernization approach can be taken:

1. The existing application is extended with some APIs.
2. The AR solution is built in the AWS Sumerian managed cloud service.
3. A Sumerian / AR compatible set of glasses is selected.



3.2 Organization aspects

Digital Transformation initiatives will affect most, if not all departments in an organization. A complete Digital Transformation strategy will cover the impact on all departments, but that is, understandably I presume, beyond the scope of this paper. This section focuses solely on how the application development activities in your IT department are affected by such a strategy.

The consequences of the current technology trends can be huge, sometimes even disruptive to your business and organization. That being said, it's important to have the right perspective on what 'disruptive' means for your company. When you Google the term 'disruptive quotes'⁷, you will find a lot of inspirational quotes and interesting people and companies. However, most of us are not working

at the likes of Apple, LinkedIn, Google, Spotify, or Netflix, and it is important to understand that this matters. All of these hyper disruptive companies are organized in such a way that they are able to leverage the Cloud's potential to meet their business goals. They have high IT budgets, the best IT engineers, and can deliver results to the end user much quicker than their competition. If no existing technology fits their needs, they develop it themselves. Meaning what is disruptive for them, is something else than what is disruptive for you.

What you need to consider within the organizational context, is that even though the technologies for building advanced cloud applications are maturing and becoming more and more readily available, the real question is if your IT department is

able to develop and maintain applications with these technologies? Are they up to that task?

To answer that question, let's look at some characteristics of cloud applications, to understand where it is different from traditional application development:

Human resource skillset

Most companies have already experienced that the cloud brings along a shift in the required skillset from their IT staff. But not only the skillset changes: also the speed goes up. Everything needs to go faster. Faster application delivery, faster adaption of new technologies. Making the most important change in the required skillset: employees must absorb change at higher speed!

Some other examples of changes in the skillset:

- Less work on platforms and systems management, especially when managed services in a public cloud are used.
- More focus on development of application (business) logic.
- Network and security aspects require more attention, especially when using (multiple) public clouds.
- Costs control shifts from an APEX to an OPEX which has to be taken into account when designing solutions.
- In general, the (speed of) introduction of new technologies requires a different mindset.
- The application development and delivery is more and more automated, making infrastructure-as-code, CI/CD and automated testing essential elements.

➤ This transition in skillset should not be underestimated. A good sourcing strategy will help to make this transition a success. In addition, a trusted partner that can supply the required knowledge and resources can speed up this transition.

Teams

Typically, cloud application development is done best in DevOps teams, i.e. teams where Application Development and Operations are combined. Lately, the term BusDevOps has been coined to stress that also the business should be represented in the application development team. The idea behind (Bus)DevOps is that when the involved disciplines are combined into one single team, their communication will be easier and faster. Hence, new business requirements can be implemented much quicker. Furthermore, the feedback loop on how the application works on the production system is also much quicker.

A key success factor for this to work is team size. A popular quote by Jeff Bezos is that 'a team shouldn't be larger than two pizzas can feed'⁸. It is easy to see that if a team gets too large, it will become slow. However, there is also Conway's Law⁵ that implies that an application must reflect the (structure of) the organization [...] and the development teams organization must reflect the structure of the application.

➤ In short, flexible and high speed application development speed requires an application design and development teams that match with how the application is or will be used in the organization.

But, a team is not a stand-alone unit in an organization. A team may need to discuss topics with managers, influencers, users, business process designers, etc. Resulting in the boundary condition that all the non-team resources must be easily accessible, or they will slow down the team.

Supporting activities

This 'application-development-need-for-speed' leads to autonomous BusDevOps teams in which as many decisions as possible are made within the team itself. However, this decision making has its limits. For example, an architect may need to approve technology choices. Or a training department may need to prepare training material for a new application functionality. Or the release board may need to coordinate with the marketing department when something goes into production. All of these supporting activities need to be considered carefully: if the release board only allows new application release 4 times per year, it does not make much sense to be able to deliver new application functionality on a weekly basis... So, supporting activities may become a bottleneck that needs to be addressed.

Supporting activities that may need to change (or need to be implemented) are:

- Release strategy
- Disaster recovery
- Consistency bridges between old and new
- Fallback scenario's
- Exit strategy

Transition phase

Most organizations don't start from scratch on development of cloud applications. For example, an organization may already have moved part of their IT infrastructure to a public cloud, so its IT department is already familiar with certain cloud technologies. Also, application development may already be working on agile projects that work in DevOps style. Nevertheless, it is recommended to clearly assess if the IT department's current status matches with what is required for successful cloud application development.

If the IT department is not ready, measures must be taken to address the shortcomings. Think of measures like:

- training current IT staff;
- hiring external staff ;
- establishing relations with trusted IT partners, and partnering with expert companies;
- establish a separate cloud expertise center.

These measures are often only temporary, 'just to get started'. However, a well-balanced set of measures will help an organization to get up to speed quickly (again ... speed). Therefore, it's recommended to explicitly plan a 'transition phase', which ensures the organization transforms into one that can handle whatever the cloud throws at them.

3.3 Application Modernization success factors

As said, an application modernization project is likely to be part of an overall digital transformation program in an organization. But, even if it is not, there are many digital transformation success factors that can be directly applied to an application modernization project. A list of the main ones, inspired by AWS¹⁶:

- **Executive Buy-In**

Modernizing an application is not a simple IT-only technical activity. It will affect many stakeholders throughout the organization and therefore support on the highest level is needed, especially when things get difficult (and I promise you, they will). If the application modernization is driven by the IT department, the CIO may be a good choice. However, if the application

modernization is driven by a business initiative, an executive that is linked to that business is recommended to represent the program on the highest level.

- **IT Staff Alignment**

All IT staff needs to be on board, so the application modernization activities need to be clear to all involved IT Staff. The need to be involved, updated regularly and their



concerns must be addressed.

Application Modernization Strategy

When an application is modernized, there will be some intermediate stages. A strategy must be determined that outlines what steps will be taken and how the transition phases look like. Such a strategy takes business needs and technical and organizational constraints into account.

The link between the strategy and the vision and mission of the organization must be made.

- **Quick Wins**

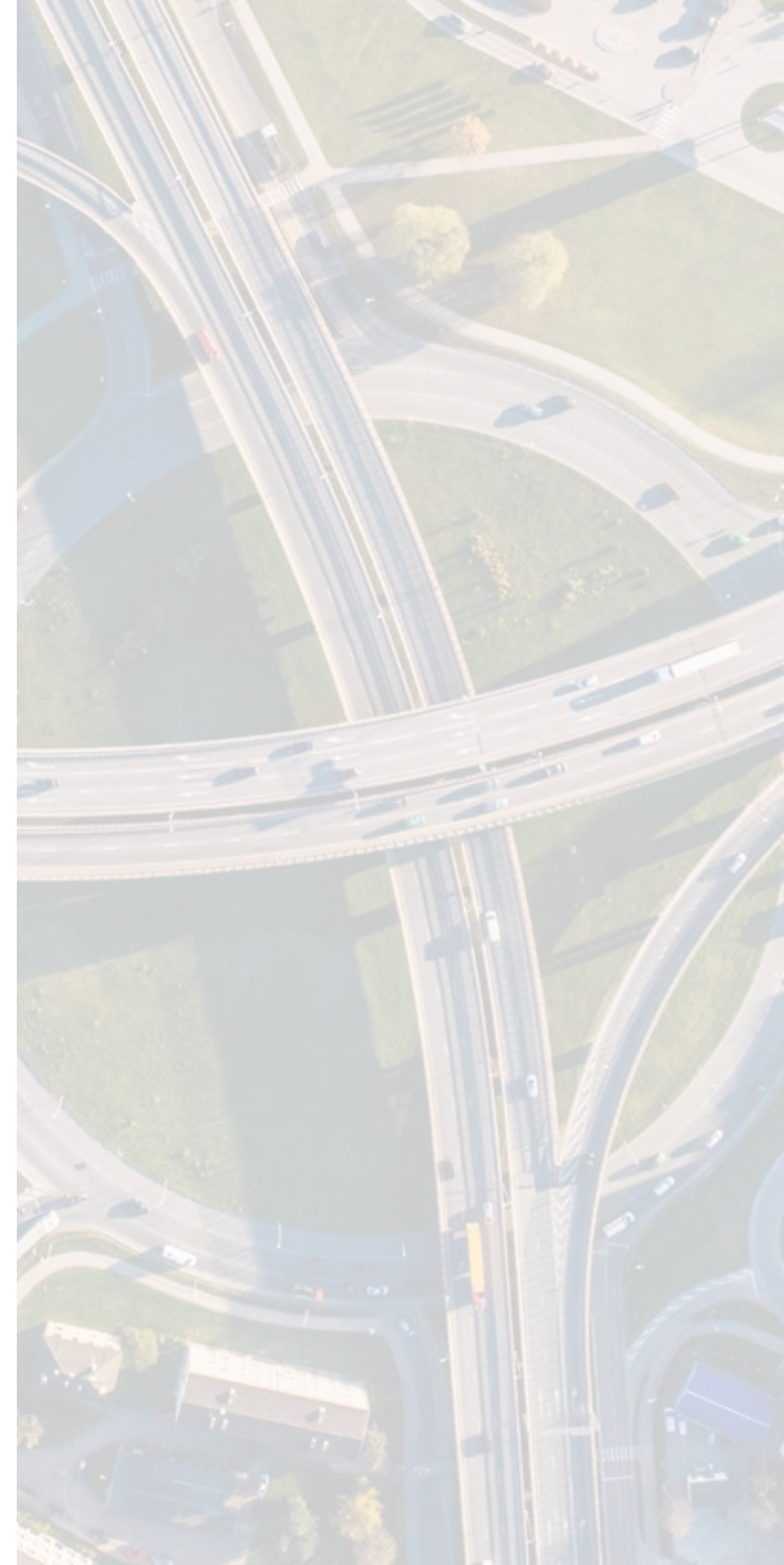
The application modernization project must deliver a quick win, to convince the whole organization that this is a worthwhile effort. Preferably this quick win has business value and low technical risk.

- **Cloud Center of Excellence**

Depending on the size and complexity of the application modernization project, it may be a good idea to establish a separate Cloud Center of Excellence team that helps other teams. This team can supply to other teams knowledge and best practices, set architecture guidelines, etc.

- **Partner Engagement**

Where the existing IT organization lacks knowledge and experience, external partners must be engaged to deliver this. These partners must not play a supplier role: the partners must commit to the program's results and even have decision-making powers.



4. Summary

Lots of organizations use public clouds like AWS, Azure and Google widely for running their IT workloads. Cloud applications promise many advantages: lower costs, more flexible applications, better scalability, etc. It is therefore no surprise that many organizations have a cloud strategy in place that outlines how they want to take advantage of the cloud. But, by now it is clear that cloud benefits aren't always achieved: simply running an application on a computer in the cloud does not make the application more flexible. Also, not all applications can be easily scaled. The traditional way of building applications had to change: new cloud technologies with their own set of best practices are maturing and make it possible for everyone to build cloud applications that benefit from what the cloud has to offer. As this maturation goes on, these technologies become increasingly accessible for organizations that don't have the size of a Google, Facebook, LinkedIn, Uber, Spotify, Netflix, Uber etc.

When it became clear that cloud applications should be built differently than traditional applications, a new challenge presented itself: what to do with the existing base of bespoke applications? These applications were built with traditional technologies and

using concepts that would block them from taking advantage of the cloud. Often, they represent large investments, which would be lost if the application was rebuilt from scratch. As it turns out, in most cases applications can be partially modernized: application modernization.

For example, parts of an applications can be restructured, or new cloud technologies can be incorporated in specific parts of the application. This tailored approach can improve an existing application step-by-step, making it better suited to run in the cloud and still optimally safeguard the large investment.

It is important to understand that the cloud does not solely affects the IT department. The key concept here is 'speed': almost without exception, organizations want their applications to faster deliver the desired functionality! This need for speed is only partially addressed by the technical solutions. It also requires the organization to transform the way business and IT work together. This often requires a full Digital Transformation approach that will impact many departments.

It will be clear that taking full advantage of what the cloud has to offer can be complex. Your organization may have to undertake a journey that affects many departments. However, increasingly available experience and best practices can teach you how to do so successfully. Making the only question left: what's holding you back?



ABOUT ILIONX

ilionx is an IT service provider that helps organizations stay ahead. Since its foundation in 2002, ilionx has been supporting its customers as a digital partner in the field of business innovations, applications, data analytics and cloud & security. ilionx offers specialist knowledge, is agile and flexible and has the ability and scale to successfully implement large projects. This has led to many successful projects for healthcare institutions, local authorities and companies, such as NN, KLM, UMCG, ABN AMRO, ASML, Sligro Food Group, KPN, VodafoneZiggo, various municipalities and a large number of hospitals.

Since 2017, Egeria has a majority stake in ilionx. In 2018, ilionx merged with QNH Consulting, in 2019 the company took over ICTZ and in 2021 integration specialist Rubix, consultancy Le Blanc Advies and Salesforce partner Redbook ICT also joined ilionx. The company has more than 1,000 employees and is located in Groningen, Amsterdam, Utrecht, Zwolle, Maastricht, Den Bosch, Eindhoven and Hoorn. More information can be found at www.ilionx.com or follow ilionx on [Twitter](#), [Facebook](#) or [LinkedIn](#).

At ilionx we have the people and knowledge to help you with all steps of your Application Modernization journey, whether you just embarked or are an experienced 'traveller'.

Are you interested in Application Modernization as a solution to meet progressing business requirements? Then feel free to contact Thijs Iding, Account manager at ilionx:

✉ tiding@ilionx.com
☎ +31 6 12 291 528

References

All internet based references are retrieved in Jan-Feb 2020.

1. NIST original Cloud Definition, July 2009, <https://www.nist.gov/system/files/documents/itl/cloud/cloud-def-v15.pdf>
2. NIST final Cloud Definition, Sep 2011, <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf>
3. 6 Strategies for Migrating Applications to the Cloud, Nov 2016, <https://aws.amazon.com/blogs/enterprise-strategy/6-strategies-for-migrating-applications-to-the-cloud/>
4. Open Platform Migration Guide, 2018, <https://www.redhat.com/cms/managed-files/platform-migration-guide-technology-detail-f10427bf-201802-en.pdf>
5. Conway's Law, https://en.wikipedia.org/wiki/Conway%27s_law
6. Gartner, Top 10 Strategic Technology Trends for 2020, Published: 21 October 2019, ID: G00432920
7. Google on disruption quotes, 2020, <https://www.google.com/search?q=disruption+quotes+>
8. Jeff Bezos 'two pizza rule', <https://www.cnn.com/2018/04/30/jeff-bezos-2-pizza-rule-can-help-you-hold-more-productive-meetings.html>
9. Microservices, 25 March 2014, <https://martinfowler.com/articles/microservices.html>
10. AWS re:Invent 2019: [REPEAT 1] Running Kubernetes at Amazon scale using Amazon EKS (CON212-R1), <https://www.youtube.com/watch?v=M-Fh0OzliJI>
11. Berke Sokhan, ThoughtWorks, Domain Driven Design for Services Architecture , <https://www.thoughtworks.com/insights/blog/domain-driven-design-services-architecture>
12. Domain Driven Design, Eric Evans, 2004, <https://www.amazon.co.uk/Domain-Driven-Design-Tackling-Complexity-Software-ebook/dp/B00794TAUG>
13. The Twelve-Factor App, <https://12factor.net/>
14. StranglerFigApplication, Martin Fowler, 29 June 2004 <https://martinfowler.com/bliki/StranglerFigApplication.html>
15. Time to strangle your Monolith to Microservices, Manish Tripathy, 9 April 2019, <https://medium.com/@manisht/strangle-that-monolith-the-strangler-pattern-40c9eeb94402>

16. 7 Essentials for a Successful Cloud-First Transformation, AWS, 2016, <https://pages.awscloud.com/7-Essentials-for-a-Successful-Cloud-first-Transformation-S.html>
17. Digital Transformation, Wikipedia, November 2020, https://en.wikipedia.org/wiki/Digital_transformation#cite_note-11
18. World Economic Forum, Klaus Schwab, The Fourth Industrial Revolution: what it means, how to respond, 14 Jan 2016, <https://www.weforum.org/agenda/2016/01/the-fourth-industrial-revolution-what-it-means-and-how-to-respond/>
19. Inside-Out Strategy VS. Outside-In Strategy: Which Marketing Approach Is Best?, Thomas Ahn, 27 maart 2019 , https://viralsolutions.net/inside-out-strategy-vs-outside-in-strategy/#.X8IRH7Mo_g
20. Digital twin, Wikipedia, https://en.wikipedia.org/wiki/Digital_twin

Photo credits: Pexels.com

Aleksandar Pasaric	2411688
Aleksejs Bergmanis	681335
Deva Darshan	1123972
Edgar Hernandez	783944
Guilherme Rossi	1755683
Kehn Hermano	3849167
Luis Gomes	546819

Layout and illustrations

Dikke Huisstijl / Wout Reinders