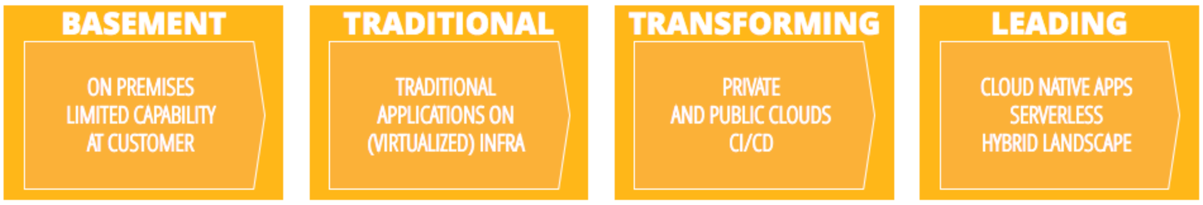
THE ROAD TO PUBLIC CLOUD

TECHNICAL AND ORGANISATIONAL CLOUD MIGRATION PREREQUISITES

Last update: 2nd July 2018

1. INTRODUCTION

The rise of virtualisation and cloud computing has totally changed the way applications are built. For enterprises around the world cloud computing has become the cornerstone of their business. Without it their business would simply seize to exist. In a perfect world every application is fully built according to the latest standards and a migration to the Public Cloud is done at the flick a switch. Unfortunately, no such perfect world exists and many applications, proprietary or not, are nowhere near perfect and cloud-ready, and neither are the organisations that build and operate them.



IT organisations can be labelled as 'basement', 'traditional', 'transforming' or 'leading' (as illustrated below). At Sentia we assist organisations in designing, planning and executing large-scale migrations to the Public Cloud as a part of a larger shift to become 'leading'. The advantages of cloud-native apps are abundant, e.g. a short time-to-market, scalability, flexibility and high resilience to any failure.

Prior to any migration to the Public Cloud we conduct a thorough assessment of all applications on cloud-readiness and help with the paradigm shift these adoptions demand. In this whitepaper we will provide you with insights into what a successful migration to the Public Cloud requires of your organisation and application(s).

1.1 JARGON AND MISUNDERSTANDINGS

In every profession types of language exist that can only be understood in a certain context. As a developer or engineer we are oftentimes confronted with ambiguous jargon. Interview ten CTO's on their definition of cloud-ready and you will probably end up with ten different definitions. It is convenient to share terminology, but it can be major a pitfall when there is no clear consensus on what the terms truly entail. We hope this whitepaper creates some consensus on common used terms and prevents any misunderstanding in the future.

“It is convenient to share terminology, but it can be major a pitfall when there is no clear consensus on what the terms truly entail.”

1.2 HYPERBOLIC DISCOUNTING

In recent years we have gained a lot of experience on migrating large clusters of applications to the Public Cloud and the accompanied organisational change. If preparations fall short these large-scale cloud adoptions can quickly become massive and burdensome. Applications come in all forms, shapes and sizes and assessing each application on cloud-readiness requires lots of hours spent on digging through code and doing research. Apart from the technical assessment it is paramount to assess the development culture of the organisation as well.

It can be tempting to skip this phase and start migrating each application on a lift-and-shift basis. Periodically these kinds of migrations eventually lead to a next project which involves refactoring and replatforming the application, but often that is the end of it. A lot of technical debt remains, CTO’s can boast all their applications are on the Public Cloud, but nothing really changed for the better. All true advantages of the Public Cloud go to waste and more often than not times the long-term costs go up significantly.

This tendency to go for the sooner-smaller reward instead of the longer-larger reward is called *hyperbolic discounting*. The value of a later reward is discounted by a factor that increases by the length of the delay, and in a hyperbolic fashion. For most CTO’s and businessowners technology can be challenging to keep up with and an evident solution for the long-term is often befogged by todays challenges. Combined with the exploding amount of new services, features and technologies Public Cloud vendors like AWS (in 2017 AWS introduced a staggering 1430 new services and features), Google and Azure introduce on a daily basis, organisations are under constant pressure to take the right decisions in a complex, volatile and disruptive world.

“This tendency to go for the sooner-smaller reward instead of the longer-larger reward is called hyperbolic discounting.”

We see it as our duty to prevent any organisation from making the mistake to migrate to the Public Cloud under false premises and conditions. If any application is migrated to the Public Cloud it should get the scrutiny it deserves. The consequences of not investing in a good assessment and change of culture can be dyer, expensive, and it can make a seemingly simple project quickly balloon out of control.

More important, any migration to the Public Cloud should be part of a larger paradigm shift. Without it most projects will most likely fail sooner or later.

2. SILOS FROM THE PAST

Many articles, books and whitepapers exist on how a cloud-ready application should be designed and what the team designing and building the application should look like. A developer should take all basic guidelines in account when designing and building each application but most of the time this is not the root cause of an application not being cloud-ready. Enterprises often still use ancient applications, built in a time where concepts like virtualisation were nothing more than pie in the sky and dedicated servers still dominated the market. Times have changed but most applications have not.

"Times have changed but most applications have not."

Most of these ancient applications were built by teams of developers with little or no knowledge on the operational side of the story. Operating in their own silo and working towards big complex releases, landing these monolithic monsters on a production environment was left for the engineers or operations to sort out. This dichotomy between development and operations has some major implications. Most developers still build applications that perform outstanding in their own development playground but fail to scale in the real world and true production/development environment parity is often nowhere to be found. The result is a substantial longer time-to-market and less room, time and budget for innovation.

In the early days it is easy to miss the first tell-tales but when confronted with a growing demand and the subsequent logical move to the Public Cloud, these shortcomings can stop any organisation dead in its tracks.

For this to change we not only need to change the way the application works and is designed, but foremost the workings of the team that builds and operates application(s).

2.1 AN INVESTMENT IN KNOWLEDGE PAYS THE BEST INTEREST

Luckily, the strong dichotomy between operations and development has become more and more a thing of the past. DevOps, a clipped compound of 'development' and 'operations', is the popular software engineering culture that unifies both silos. DevOps aims to reduce the time between committing changes to an application (or system) and the changes being deployed in an environment, whilst ensuring high quality through leveraging automated testing and monitoring of all steps.

In this culture it no longer suffices as a developer to know little to nothing about operations and vice versa. This changes the way applications are built and maintained in a profound way. Developers should refrain from large releases and instead continuously deliver small changes which are automatically tested and oftentimes automatically delivered and deployed to a real-life environment.

Doing so prevents scarce developers spending scarce time on large changes that eventually do not yield any useable results and little time has to be spent by operations on landing the application in a real-life environment. A proper implemented CI/CD (Continuous Integration/Continuous Delivery) pipeline helps developers adopt this new way of working and should be enforced as the only mechanism to contribute to any existing codebase. Designing, building, and maintaining this pipeline requires discipline, time and budget, but it is the only way to break free from untested, unautomated, labour-intensive, fault-prone and immensely complex releases. Adopting this way of working requires developers to change their old ways, learn how new tooling works and understand what CI/CD can do for them.

"Designing, building, and maintaining this pipeline requires discipline, time, and budget, but it is the only way to break free from untested, unautomated, labour-intensive, fault-prone, and immensely complex releases."

Successfully migrating to the Public Cloud has to go hand-in-hand with a culture change and an investment in the knowledge of developers on operations, and vice versa, is therefore incumbent.

Developers need to understand the implications of their day-to-day work on the scalability of the application in the Public Cloud. There are many ways to share this knowledge. We have found that creating a proof-of-concept and a dedicated playground to be a vital and integral first part of any move to the Public Cloud but letting developers loose on these virtual playgrounds should never be your sole effort.

2.2 MUTATIS MUTANDIS

Even though DevOps has been largely adopted and it has helped pave the way for a less segmented way of working, we are still far distant from a world in which every developer understands what the operational or security consequences of seemingly small parts of code are. To create this understanding education is the most logical route, but there are complementary measures an organisation can take to ensure applications are built in a cloud-friendly manner.

As much as developers have become involved in operations, system-engineers have moved into the domain of development as well. From a historical point of view Sentia predominantly operated in the operations silo but in recent years has claimed larger parts of the development silo.

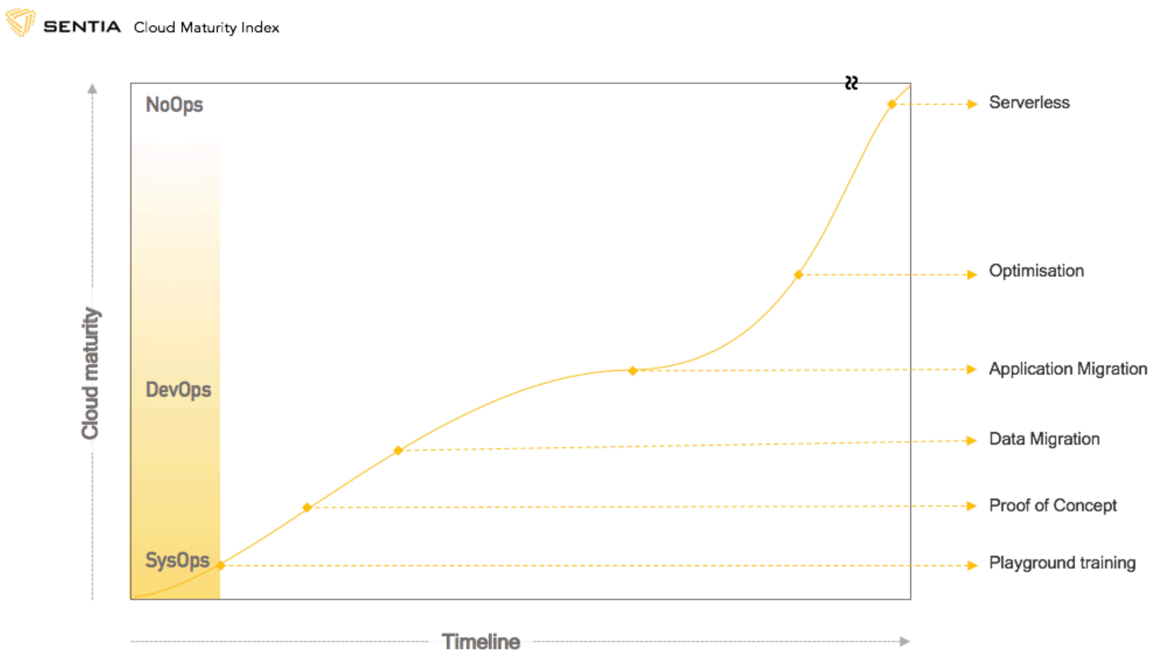
For example, in our Managed Application Continuity (MAC) proposition Sentia goes much further than taking responsibility for the platform and underlaying OS, network, and infrastructure. Sentia takes full and sole ownership over the complete stack, including the application(s). This requires engineers to have in-depth knowledge of the application, something that goes way and beyond their expected competencies in a traditional SysOps world.

Another example of this fading border between operations and development is the Infrastructure-as-Code methodology applied in many of our AWS projects. In this methodology every environment is built up from templates (e.g. Cloudformation templates), servers are considered ephemeral (hence the saying: treat your servers like cattle, not pets) and should be treated as such and the templates are stored in a version control system. In many projects Sentia has leveraged an advanced CI/CD pipeline to attain the highest degree of automation and quality assurance in the development of these templates. Infrastructure and application are becoming more and more intertwined.The similarities between developing applications and infrastructure are striking and demonstrate how the Public Cloud has opened the door for both developers and engineers to combine their strengths and build better applications collectively.

Involving Sentia as a partner in go-to-cloud migrations enables your team to benefit from our solid experience in these challenging projects, both from an operational and developer's perspective.

2.3 YOU ARE HERE: THE CLOUD MATURITY INDEX

All projects start with defining a desired outcome and assessing the current state of affairs. We have created a Cloud Maturity Index to illustrate what most go-to-cloud journeys look like and to help you define a point on the horizon.



Let us start with a small caveat. It might seem obvious that NoOps is the point on the horizon for every organisation, but it definitely is not. In our opinion for most organisations NoOps is a distant utopia. Only when designed for Public Cloud from the ground up some applications can be classified as NoOps, but even for these applications there are some very strong arguments against NoOps. The most important argument is that there will always be a strong need for operational support and, due to the digital transformation many companies are in the midst of, this need is more likely to grow in the upcoming years than diminish.

The second most important reason is the vendor lock-in that goes hand-in-hand with building your application on Public Cloud vendor specific services. We do not argue that a vendor lock-in is bad per se, but at the moment of writing it can be an argument against Public Cloud services-based NoOps.

Therefore, we conclude a high degree of optimisation using Public Cloud services is the point on the horizon in most go-to-cloud migrations.

3. TECHNICAL ASSESSMENT

As mentioned earlier, a technical assessment should be the basis of every go-to-cloud project and should give you a good indication on where to start. Whether that's a full-scale data migration or a small proof of concept, a technical assessment clearly shows what needs to be done to successfully migrate your application(s) to the Public Cloud.

For Sentia to successfully assess an application on cloud-readiness, we need to look at its foundations as modern day developers. It is important to comment that this approach is not self-evident, as most traditional hosting companies tend to steer away from anything that involves assessing or altering the source code of the application. In our view the lines between application and hosting are rapidly fading. Infrastructure is more and more becoming part of 'the code', clearly demonstrated in the Infrastructure-as-Code methodology we mentioned earlier.

"For Sentia to successfully assess an application on cloud-readiness, we need to look at its foundations as modern day developers."

As we have assessed countless applications and developed a long track-record in migrating applications to the Public Cloud, our starting point is our very own experience. Some applications are too monolithic and ancient to ever successfully land on the Public Cloud. These dinosaurs have rooted in on-premise datacentres and should never be attempted to move anywhere near a Public Cloud environment. Heavily infested with dependencies and old design patterns these applications are either to be retired or retained, depending on what makes sense for your business. We can spot them from a distance and our experience will save you valuable time and budget.

3.1 THE 12-FACTOR APP

When we assess an unknown application we start out with the 12 factors, originating from *the 12-Factor App*. In the early days of the Cloud, the founders of Heroku (now a subsidiary of Salesforce) offered customers hosting for their applications that was entirely worrisome-free. All you needed to do was upload your code and they would take care of the rest. Their promise comprised you no longer needed to worry about hosting or infrastructure. Far ahead of its time and absolutely ground-breaking. It was a grand promise which heavily leaned on the ability of developers to build cloud-ready applications. Quickly confronted with a flood tide of questions, Heroku developed a manifesto to provide guidelines on how to write cloud-ready applications.

This manifesto has become the guideline for many developers and is still used today. Some of the factors are kind of straight-forward and basic, but still valuable to take into account when designing, building or assessing applications.

The Heroku manifesto consists of twelve factors. It would exceed the aim of this whitepaper to deliberate on each factor but let us summarize the factors.

1. **Codebase**: *One codebase tracked in revision control, many deploys*
2. **Dependencies**: *Explicitly declare and isolate dependencies*
3. **Configuration**: *Store configuration in the environment*
4. **Backing Services**: *Treat backing services as attached resources*
5. **Build, release, run**: *Strictly separate build and run stages*
6. **Processes**: *Execute the app as one or more stateless processes*
7. **Port binding**: *Export services via port binding*
8. **Concurrenc**y: Scale out via the process model
9. **Disposability**: *Maximise robustness with fast start-up and graceful shutdown*
10. **Development and Production parity**: *Keep development, staging and production as similar as possible*
11. **Logs**: *Treat logs as event streams*
12. **Admin Processes**: *Run admin/management tasks as one-off processes.*

3.2 CONSIDERATIONS ON APPLYING THE 12 FACTORS

The 12-Factor App manifesto was written by pioneers in a time the Cloud was still exotic and novel, and it should be read and applied as such. Some factors are more applicable and useful than others. Whilst assessing applications on cloud-readiness some factors are considered as knockout criteria, e.g. the application being stateful.

“In certain cases exceptions can be allowed, if dealt with in a structural matter.”

In certain cases exceptions can be allowed, if dealt with in a structural matter. A good example is the use of sticky sessions. A sticky session means the application is caching user session data in the memory of the applications process. It assumes the visitor will remain to be routed to the exact same server, which can be risky. The application is still able to scale out but in the event a server goes down and needs to be replaced, the data on this server will be lost, and subsequently the user is faced with errors or missing data. A good way to solve this problem is storing the session data in a datastore, such as Redis. Although not solving this problem is against good practice, we find it to be tolerable in the short-term and not blocking for a migration to the Public Cloud if not combined with other major deficiencies.

4. THE BOTTOM LINE

Obviously, disregarding any red flags in the technical assessment will eventually cause problems further down the line. If refactoring is not an option for you it is time to look at alternatives. Retaining the application, keep it as-is, can be a justified approach in some situations. Replacing the application with another application altogether can make sense as well, especially if there is a good SaaS equivalent available.

“Retaining the application, keep it as-is, can be a justified approach in some situations.”

Any of these alternatives should be carefully considered, not just from a technical perspective. A SaaS platform can profoundly reduce the total cost of ownership but as a side effect create a vendor lock-in which, in some cases, can make things go from bad to worse.

If you decide to and have the means to refactor the application, Sentia can advise you on feasibility and approach. Most of times there are more ways to hook a fish. We are more than happy to help you with our in-depth knowledge and expertise.

4.1 WHAT NEXT?

A migration to the Public Cloud and changing the development culture of an organization can be challenging. Sentia has the know-how, experience, resources and proven methodologies to help you with every step along the way. We can provide you with a well-defined strategy and resources to execute your migration and help you improve your competitive advantage with a shorter time-to-market and fewer operational costs.

4.2 YOUR FIRST ENCOUNTER WITH SENTIA?

If this is your first contact with Sentia, know we are happy to support you in applying new technologies and methods in your ICT landscape. Please feel free to contact us if you have any questions or remarks.