amdocs

analysys mason

White paper

Preparing your organization for a cloud-native future: best practices for people, processes and platforms

January 2020

Caroline Chappell

RESEARCH

Contents

List of figures

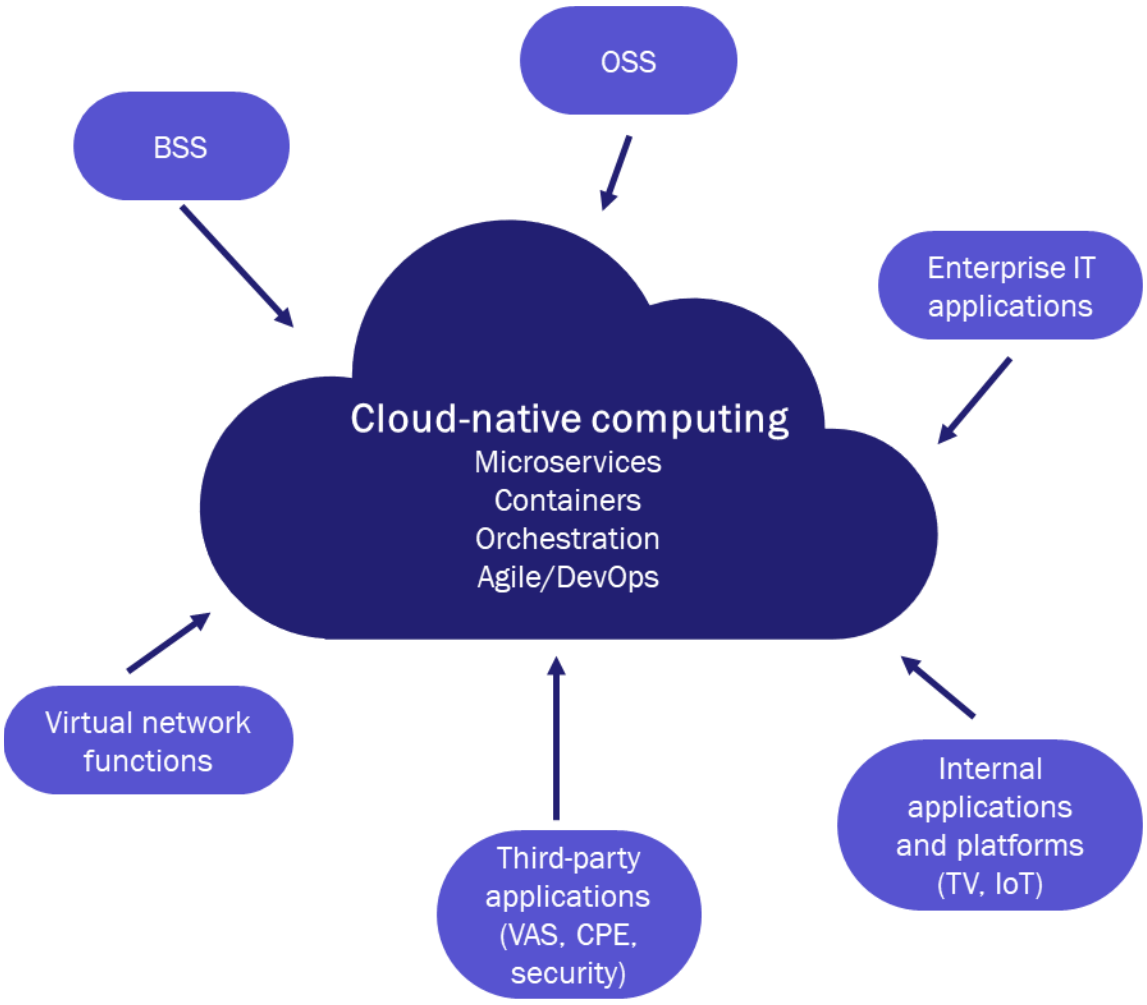
1. Executive summary

1.1 CSPs must embrace a cloud-native approach to software

The business drivers for developing and deploying cloud-native software are clear. Communications service providers (CSPs) want to gain the flexibility and speed advantages that webscale companies have when bringing new products and services to market. Adopting their cloud-native way of building software is key to achieving this goal. In addition, CSPs' vendors, from BSS/OSS suppliers to network function owners, are developing their products using a cloud-native approach. CSPs need to become both familiar with cloud-native software and skilled in its production and deployment so that they can rapidly introduce innovation to their customers, regardless of whether they have developed such software in-house or procured it from a vendor.

The use of cloud-native software accelerates the rate at which a business can respond to the market. Cloud-native software therefore represents the next big wave of change for CSPs' employees, processes and technologies. CSPs need to ensure that their approach to cloud-native software development and deployment is right to minimize their risk. It is not feasible to adopt a new way and velocity of working overnight, so instead they must start small and transform their development and operational teams gradually, applying best practices from the outset. On the other hand, CSPs cannot afford to delay. Their peers and competitors outside the telecoms industry already have, or are acquiring, cloud-native software capabilities. Without these capabilities, CSPs will be unable to maximize the benefits of the next generation of operations and network-based software components that are already available to them today and which will underpin 5G service opportunities in the future.

Figure 1.1: All software in use in a telecoms organization will eventually converge onto a common cloud-native foundation



Source: Analysys Mason, 2019

1.2 Making the right organizational changes

People are the key to success with cloud-native software. CSPs need a critical mass of software engineers with the mindset and skills to use cloud-native processes, platforms and tools effectively. However, changing people and organizational culture is the hardest aspect of any transformation. Every organization is different and each CSP should change at its own pace, potentially engaging an external partner to help manage organizational challenges and bridge skills gaps.

Best practices that can help with organizational change include implementing the right methodology from the start so that it can be rolled out consistently to other teams, making allowances for the time that it will take people to change their practices, providing the right support so that changes are meaningful, identifying critical influencers who can inspire and lead change and putting the right metrics in place to keep the change process on track and promote its benefits.

1.3 Making the right process changes

Cloud-native software mandates a lean and agile development and operations process. This starts with the software design, which is different from the traditional way of building monolithic applications. CSPs need to adapt their software design processes to accommodate microservices-based cloud-native software and automate the delivery of that software into production. In a cloud-native approach, the design and delivery of software also includes designing and delivering the way in which the code will be operated (managed) when in production. As such, the development, delivery and operations of cloud-native software (that is, the software development lifecycle (SDLC)) should all governed by the same continuous process, and should be automated using a 'pipeline' of integrated software development, delivery and operations tools. This process is fundamental to achieving velocity in a cloud-native software environment.

CSPs will need to master three inter-related activities in order to adapt their software development and deployment processes for a cloud-native world:

- a new approach to software design using microservices
- the optimization of the end-to-end build, integrate, test, deploy, operate process for pushing software into production and managing it once it is there
- the automation of this end-to-end process to gain speed and consistency benefits.

1.4 Acquiring the right platform

Developers need a tools platform to support automated delivery processes in a cloud-native environment. This platform should be established at the very beginning of a CSP's cloud-native transformation; it should be curated to support the needs of the first team to pioneer a DevOps approach, and subsequently managed centrally for the benefit of all teams.

A centrally managed platform provides the set of tools needed to automate a CSP's SDLC, guides to optimal design patterns and development best practices (such as those that ensure the performance, scalability and portability of cloud-native software) and the CSP's approved set of 'non-functional' services. These non-functional services are those services needed for the correct operation of any application, such as services that support the monitoring, security, auditing and data storage needs of an application. These services should be common across all applications, whereas 'functional' microservices encapsulate business logic for a specific application.

Implementing a centrally managed platform across the CSP's organization has a number of advantages over allowing groups of developers to put together their own 'pipelines' of tools to support the SDLC. It prevents the uncontrolled and fragmented proliferation of tools in the organization, thereby enabling people to move freely between development teams without having to reskill. It prevents problems from arising in production, when cloud-native software from a team using one set of tools and non-functional requirements needs to interoperate with software developed by another team with completely different tooling. It also supports the consistent, end-to-end automation of the SDLC process.

CSPs will need to decide whether they build their own tooling platform or acquire such a platform from a third party. This paper sets out the evaluation criteria that CSPs can use to help them to make this decision.

The paper describes Amdocs's adoption of a cloud-native software approach, as well as the best practice decisions that have guided the company's changes to its organization and processes, and informed its development of a centrally managed tooling platform.

2. Recommendations

Analysys Mason believes that CSPs can best prepare themselves to work with the cloud-native systems of the future if they follow the three principles listed below.

- **Do not delay.** Cloud-native software is bring rapidly adopted because of the clear benefits that it confers. Your suppliers are already migrating to cloud-native environments, and are, in many cases, rebuilding their systems to be fully cloud-native. Their release cadence is changing and customers that can take advantage of this will be better positioned to address market opportunities and challenges. A cloud-native transformation is a long journey, so CSPs should start on it as soon as possible, because there will be no easy way to catch up.

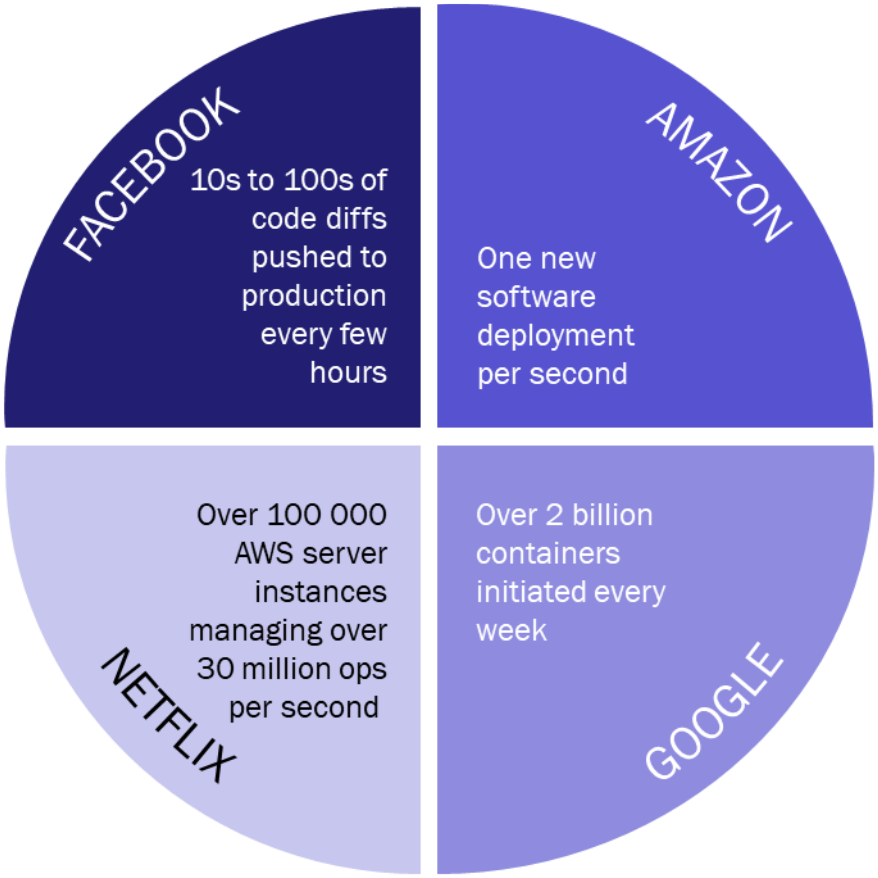
- **Start small, with a well-defined business pain point and then scale, with an end goal in mind.** CSPs can implement cloud-native transformations gradually and incrementally, rather than having to plan for and execute changes to culture, processes and technology on a disruptive scale. CSPs can apply cloud-native practices in a greenfield project just as easily as when extending an existing system. CSPs should begin with a small team and a simple, well-scoped business pain point. The team should produce a minimal viable solution as their first deliverable, which can be scaled up through small iterations using cloud-native processes and platform support.

- **Apply best practices from the outset.** CSPs should adopt cloud-native technologies, processes and mindsets in the right way from the start to avoid the pitfalls associated with non-optimal ways of working with cloud-native software. They should understand and adopt the best practices regarding changing the culture of their software development and deployment organizations (people), their development and deployment methodologies (processes) and their supporting tooling platforms (technology).

3. Business drivers for a cloud-native-capable organization

Webscale companies have pioneered cloud-native software development approaches and technologies to achieve awe-inspiring metrics for business velocity and scale (see Figure 3.1). Swathes of IT applications, and even software formerly embedded in dedicated hardware, are now being developed to run ‘natively’ in the cloud, and are able to take maximum advantage of cloud benefits such as speed, scale and low cost.

Figure 3.1: Webscale companies have pioneered the speed, scale and cost benefits of cloud-native software



Source: Analysys Mason, 2019

Leading CSPs have been following webscale companies' example for some time. CSPs are excited by the opportunity that 5G represents to build a new cloud-native network and service portfolio that will provide them with webscale speed and scalability advantages. For example, AT&T's Network Cloud is cloud-native infrastructure that underpins its 5G launches and is ready to run cloud-native network functions (NFs). Open Network Automation Platform (ONAP), which AT&T and other CSPs see as critical to 5G network management, is being developed using microservices principles. CSPs have, or are investing in, cloud-native development skills and tools so that they can become software companies in their own right. They want to be able to create a new generation of digital services, including applications that will support 5G use cases. A growing number of CSPs are undergoing IT transformations, and are supporting business processes with cloud systems implemented in a cloud-native way.

There are compelling, well-documented reasons to build and deploy cloud-native software at a time when "software is eating the world"[1] and is becoming critical to achieve a competitive advantage. Traditional methods of developing, deploying and updating software are significantly slower than cloud-native approaches. Cloud-native software supports the rapid introduction of innovation, regardless of whether a company is developing applications itself or implementing third-party systems that have been developed in a cloud-native way.

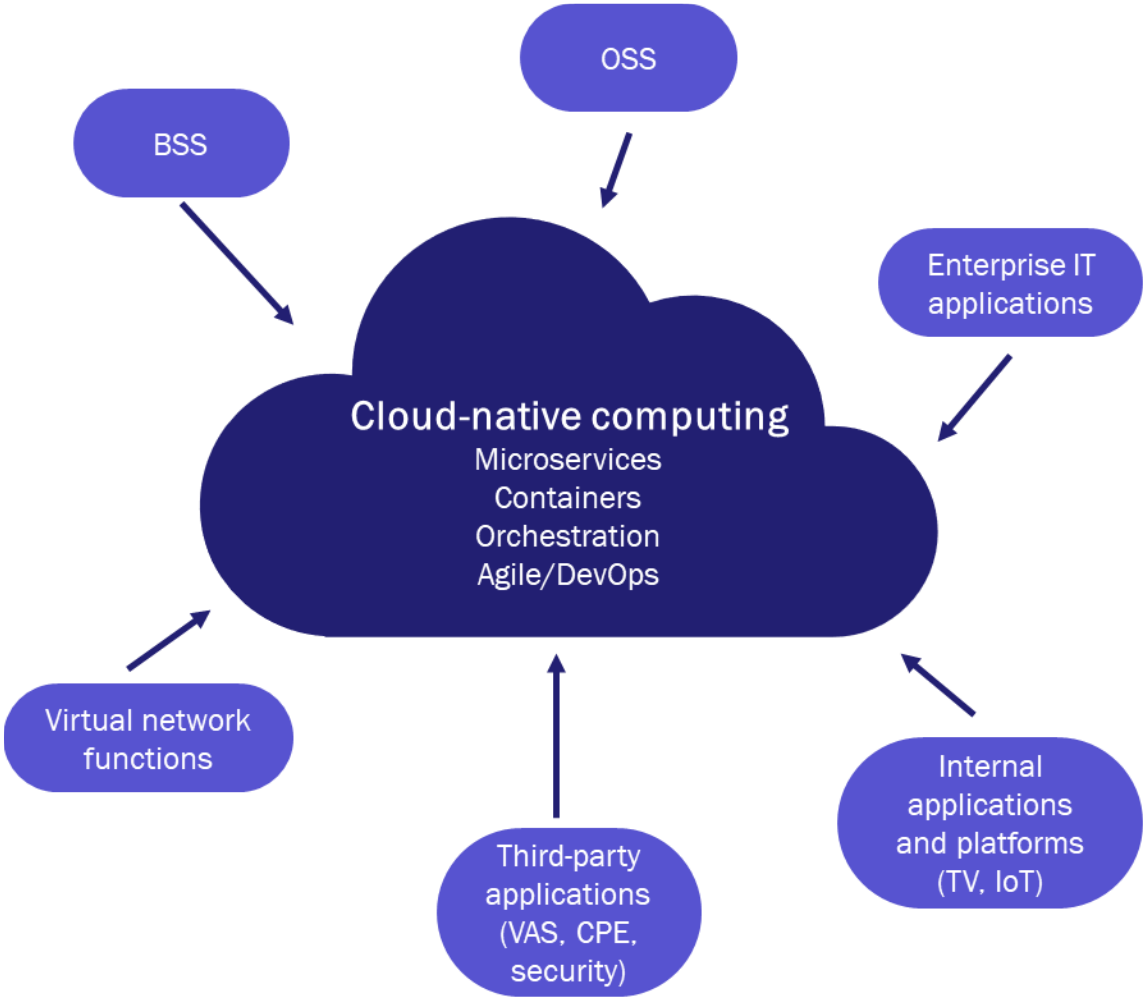
Individual microservices are quick and cost-efficient to develop and test, so features can be added and changes made incrementally, thereby de-risking their introduction. A microservices architecture also makes use of the

1 Quote from Marc Andreessen, Andreessen Horowitz, 2000.

scaling and automation capabilities of the cloud, making cloud-native software inherently resilient and 'elastic'. These properties have been difficult to achieve with previous generations of software design. Cloud-native software is often built using a microservices architecture, which produces small, stateless software components that are organized into business domains and can run in the lightweight virtualization technology of the future: 'containers'. Every aspect of a microservice's lifecycle is automated for agility and efficiency.

In a telecoms context, every type of software that a CSP uses to run its business will become cloud-native over the next few years, from consumer and enterprise customer applications to business support systems, and from operations management software to network functions (see Figure 3.2).

Figure 3.2: All software in use in a telecoms organization will eventually converge onto a common cloud-native foundation



Source: Analysys Mason, 2019

All CSPs will sooner or later engage with cloud-native software, even if they do not develop it themselves. CSPs will need to know how to deploy cloud-native software, and many will want to engage with their suppliers to 'co-create' customized, differentiated functionality on top of vendor's cloud-native software platforms. In this way, CSPs will be able to advantage of the ease with which cloud-native software can be extended to introduce innovation at a rapid and continuous pace.

Cloud-native software represents the next big wave of change for CSPs' employees, processes and technologies. CSPs need to get it right to minimize risk. Fortunately, cloud-native practices and systems can and should be introduced incrementally, and the support of a mature technology platform can significantly reduce any transformation pain.

4. Putting people first: making meaningful changes to your organization

4.1 Drivers of changes to employee practices

People are the key to success with cloud-native software. CSPs need to have a critical mass of software engineers that have the mindset and skills to use cloud-native processes, platforms and tools effectively. Most CSPs will follow a mixed strategy of reskilling existing employees and hiring new talent to ensure that they have the right resources in place. Such resources have a high market value and are critical to CSPs' long-term future, so CSPs must ensure that they create a congenial and innovation-focused working environment that both attracts and retains these staff.

Cultural changes are the hardest aspect of any transformation. Every organization is different, and there is no magic 'cookie-cutter' approach that will work for every company. Each CSP should change at the right pace for its organization, and should potentially engage an external partner to help manage organizational challenges and bridge skills gaps. CSPs may undertake different transformation journeys, but there are best practices that they can follow to smooth their paths and reach a positive outcome (see Figure 4.1).

Figure 4.1: Best practices for changes to employee practices

Source: Analysys Mason, 2019

4.2 Best practices for changes to employee practices

- **Be consistent.** Teams that are responsible for change management should use the processes and tools that they advocate for the organization as a whole. Most CSPs are adopting the Scaled Agile Framework (SAFe) methodology for cloud-native software development due to the widespread industry support and the flexibility that it gives them for customization. The change management team should also use the SAFe, if that is the chosen methodology.

- **Take your time.** It is hard to change hearts and minds, ways of working and skillsets, and it is easy to underestimate how slowly transformation will progress. CSPs need to be realistic about the speed at which benefits will be realized, disseminated across the company and recognized as worthy of adoption by the mainstream development community, not just by a handful of early enthusiasts.

- **Make change meaningful.** People do not change just because their job titles do. They need to be guided and supported, ideally through training from an experienced partner that helps them to understand their new

roles and responsibilities. They need formally defined processes that reinforce behavioral change, and good role models and leadership.

- **Start small.** CSPs should start with a handpicked, elite team that has the enthusiasm to change and the right team-working and technical skills to succeed in doing so. Focus on one team to start with and begin the transformation at a natural time of change, for example, at the start of development for a new product.

- **Choose champions.** A critical factor in transformation success is the identification of critical influencer(s) who command respect within the organization, display leadership skills and are passionate about change. Such champions for change will become the future trainers of other teams, thereby enabling the transformation to scale.

- **Measure appropriately.** CSPs need to put metrics in place that accurately gauge the efficiency of their development teams, such as the velocity of development, the duration of projects and the predictability of both of these measures. It is important to capture operational readiness metrics too, including the rate of 'commits' (the number of software changes successfully accepted and deployed), the readiness to deploy and the level of automation achieved within different DevOps processes, such as testing.

4.3 Best practices when working with a partner organization

The introduction of a partner organization to facilitate a transformation can be a good short-term alternative to hiring new staff. New employees have an incentive to 'fit into' the existing culture and may not have the authority to push for change, even if they have the right skills to do so. CSPs have often been disappointed by new hires' inability to make headway against their prevailing organizational cultures.

A partner organization is appointed to deliver transformation and is invested in making it happen. However, the partner must be the right fit for the CSP. The people it brings into the CSP's organization to effect change should be able to win the respect of the CSP's employees through their expertise and personal qualities. The partner's employees must maintain their independence from the CSP's culture in order to change it, but they must nevertheless be good team players that can act as exemplars of the desired behavior. They should work alongside the CSP's employees as true colleagues, and should participate fully in multiple DevOps roles, such as product owners, software architects and developers. They should not simply prescribe new ways of working; they should demonstrate the effectiveness of agile processes and technologies, and illustrate the benefits both for the business and for developers themselves. Showing (not telling) is the best way of persuading CSPs' software developers that their working lives can be less stressful, more productive and more enjoyable through the adoption of cloud-native practices.

A partner organization is likely to have broad experience of the platform needed to support the CSP employees that will participate in cloud-native software development and operations. As a result, a partner can guide the introduction of such a platform, and can work with the pioneering CSP team to implement it in a way that optimally reinforces the new operating model. The activity of building the cloud-native tools platform should act as a proof of concept, both for the way in which the new team should work and for the technologies in the platform itself. A partner can help a CSP to 'be consistent' in this respect and to implement the platform more quickly than it could on its own.

5. Transforming the process: domain-driven design and value stream mapping

5.1 Introducing cloud-native operations

People and processes are intrinsically linked. CSPs want to change their people so that they can take advantage of a new way of working: in this case, a lean and agile software development and operations (DevOps) process.

The starting point for this operational transformation is the software design process. In a cloud-native environment, software applications are built as sets of highly modular microservices, and they require a different design approach to that of traditional monolithic applications. CSPs therefore need to rethink their design processes to accommodate cloud-native software, especially as discipline is required to ensure that developers only create microservices that have a clear purpose.

The process of developing and deploying microservices (the software development lifecycle (SDLC)) needs to be automated as much as possible in order to gain time to market and continuous integration benefits. The set of automated processes that support the SDLC (known as a 'pipeline') should be standardized and should use tools that are selected centrally, rather than by individual DevOps teams.

As we have suggested, CSPs should 'start small' by introducing these practices to a single, elite project team first, before attempting to scale them across their organizations. The set of software-design-to-delivery processes (including the SDLC pipeline) should be optimized and refined as it is rolled out based on the value that it is providing to the CSP business. The technique for understanding the process improvements that are needed is known as value stream mapping. Incorporating value stream mapping into CSPs' cloud-native software development environments ensures a culture of continuous improvement, which is of benefit to the business.

5.2 Rethinking your approach to software architecture

The majority of software applications are currently built as self-contained, monolithic systems that contain all the business logic and data that is needed to carry out an end-to-end function. Monolithic applications can therefore have very large code bases that are difficult to modify and expensive to maintain and operate. Sportswear manufacturer, Adidas, describes them as 'elephant' applications, and contrasts them with cloud-native applications, which it likens to 'ants'. Applications composed from multiple 'ant-like' microservices are inherently easier and more cost-effective to develop, change and manage on top of a cloud-native platform. However, designing and building microservices requires developers to apply a very different and rigorous process to that used for developing monoliths.

A domain-driven software design process supports the decomposition of the desired end state of a large software system into modules that will work together. Such modules, or functional domains, should be designed for maximum reusability. For example, a concept such as 'customer' will be used in multiple different applications, so it is sensible to separate it out as a domain that can be shared by others. All customer-related microservices will then have a 'home' domain to live in, thereby avoiding the replication and siloing of customer data in different applications, and supporting data consistency across applications that use the shared domain. Domains are loosely coupled, so microservices within them can change without affecting any other domain. This is a key principle that supports the rapid speed of development and integration in a cloud-native software environment.

The adoption of a domain-driven software design process guards against the development of microservices for the sake of building them. Some CSPs have spent months building individual microservices without using proper architectural principles, only to find that they are not reaping the speed and integration benefits of modularity and composability in their application development.

5.3 Defining a new cloud-native software development and delivery process

The end-to-end, cloud-native software delivery and deployment process should be highly automated for speed, consistency and reliability. The resulting pipeline is intimately linked to its supporting technology; that is, a platform containing a standardized set of DevOps tools. This platform will be discussed in Section 6.

Process automation will include the following activities.

- **Build automation.** This covers the set of activities associated with creating a software build. In a cloud-native process, this typically includes **continuous integration** (CI) (the automated practice of merging tested developer code into a shared repository of 'mainline' code at frequent intervals (usually several times a day)) and applying quality control. It is easier to apply quality control often to small code increments than it is to test an entire system at the end of a development. CI helps with early detection of code integration problems and results in a higher quality end product. This is vital when software applications will be composed of different combinations of microservices developed by different teams.

- **Test automation.** This is the automation of the critical but repetitive tasks that are needed to validate built software. Test automation is necessary for continuous integration; without it, developers would struggle to support the required frequency of 'commits' or risk merging untested or partially tested code into the main repository, which could destabilize an entire build.

- **Deployment automation.** This integrates with other tools in the pipeline to support deployment activities such as automated version control and the configuration of microservices that are ready for release. Cloud-native software uses **continuous deployment** principles. Individual microservices can be delivered into production by the pipeline as soon as they have been developed/upgraded/fixed. In advanced continuous deployment environments, hundreds or thousands of code changes may be deployed each day. This would not be possible without a high level of automation.

5.4 Implementing the process: value stream mapping

To establish a new cloud-native software delivery process, CSPs should start small by creating a single, cross-functional team as a proof of concept (PoC). This new team should be given the target of building a minimum viable solution (MVS) with the new cloud-native process. The CSP's executive leadership team should determine the business area that the PoC and MVS will address, based on the CSP's business goals, strategic direction and an initial readiness assessment. The PoC team should therefore aim to produce business value within 6–9 months through the application of a lean and agile cloud-native process.

The starting point for process change is an understanding of the existing software delivery processes. In this discovery phase, a CSP should aim to understand its current processes from the perspective of all participating roles. It can be helpful for those responsible for defining the new process, and/or their partner organization, to join project meetings to understand the dynamics and challenges associated with the current way of delivering software. They should then use a value stream mapping process to help them to define the new, lean and agile process to be used for cloud-native software development and deployment.

A value stream mapping process uses velocity and quality metrics to identify key areas in a value stream (in this case, a software delivery process and its associated tooling) that can be improved, enhanced or eliminated. The output of a value stream mapping process is a customer-centric, data-driven visualization of how work should flow through the software delivery process. This enables the CSP to identify waste and process disconnects and to come up with a well-articulated desired state for the new delivery process. It is recommended that the new process is aligned with SAFe best practices for the reasons mentioned above. The process should also be instrumented to ensure that the right results are being achieved. Important metrics here include mean time to value, amount of waste/toil, committed versus completed tasks and defect density.

Sometimes an organization thinks that its process is agile, but in fact, it is not following SAFe best practices. It can be useful for a CSP to audit its process if it does not feel that it is achieving the expected benefits from cloud-native software delivery. For example, daily meetings may reveal a blame culture, poor co-operation or a leader who assigns work rather than empowering the team to select its next tasks. In this case, teams need to be shown the right path and make small changes in order to meet SAFe requirements.

6. Adopting the right platform for DevOps efficiency

6.1 What is a DevOps platform?

In a cloud-native environment, developers need a tools platform to support automated delivery processes, or pipelines. This platform should be established at the very beginning of a CSP's cloud-native transformation: it should be curated to support the needs of the first team to pioneer a DevOps approach and subsequently managed centrally for the benefit of all teams.

The platform has three critical capabilities. It should:

- be the single source of tools for automating an operator's build, testing and deployment processes
- provide developers with a 'paved road' to guide them, based on optimal design patterns and best practices that enable them to develop efficient, secure, reliable and compliant microservices
- automate non-functional tool-supported requirements (such as observability, security, data storage and audit) and ways of working to achieve performance, scalability and portability across different environments, and decouple these non-functional requirements from functional requirements so that developers can focus on developing value-added business functions.

The designated owners of the platform should act as a center of excellence for best practice software delivery knowledge and tools expertise.

6.2 Drivers for a centrally defined and managed platform

A centrally managed platform enables developers across the entire organization to reuse the same non-functional microservices. Such consistency promotes efficiency and security. Because the platform is shared, developers can move freely between teams, supported by the same tools. Teams know that features developed by any other team can be safely reused because they all use the same pipeline and supporting technologies. Automated processes such as CI/CD become frictionless because every microservice pushed into production through the CI/CD pipeline has been built in a consistent way with predictable behavior.

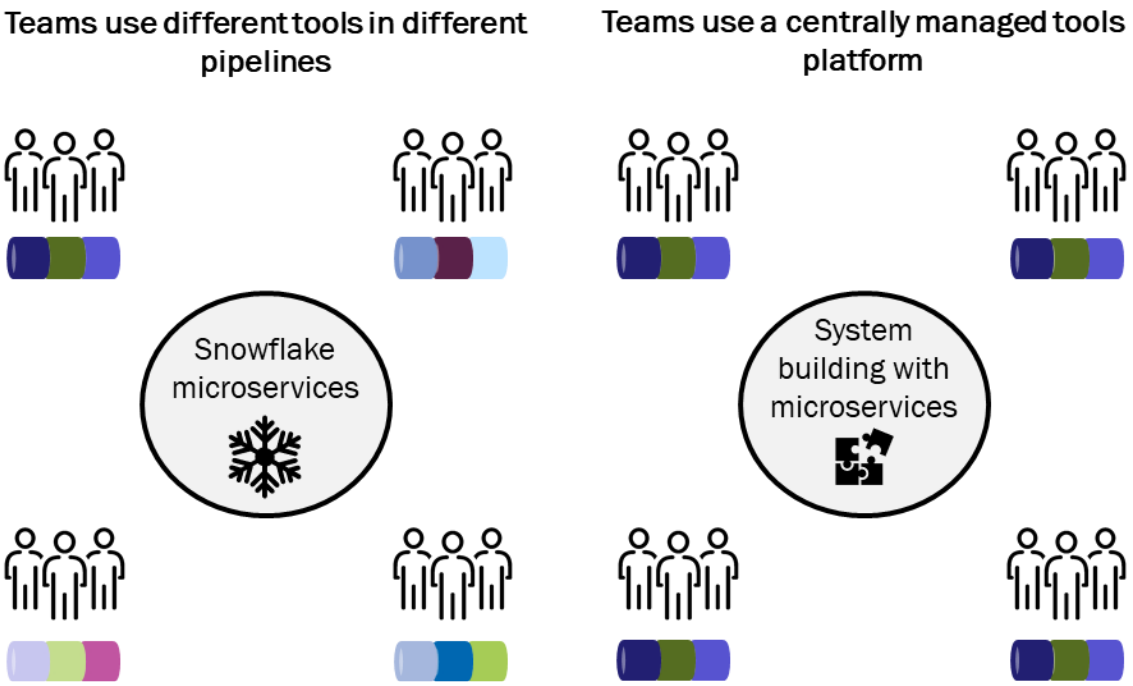
In summary, the three key reasons for using a common platform to underpin all DevOps activities are as follows (Figure 6.1).

- **A centrally managed platform prevents uncontrolled tool proliferation.** Improving the velocity of feature development is a key reason for adopting a cloud-native software approach, so developers are incentivized to 'get the job done' as fast as possible. They naturally look for tools to help them and, left to their own devices, will bring in tools that they know. The tool stack for cloud-native application development is more complex than, and very different from, that used to build monolithic applications. In the cloud-native environment, multiple tools are available to address each task in the pipeline, and many of these tools come from open-source communities. If each developer or team creates their own pipeline supported by their preferred set of tools, an operator's tooling landscape is likely to be highly diverse. 'Tool sprawl' is difficult and expensive to manage, prevents the creation of 'tribal knowledge' around specific tools and practices and results in 'snowflake' microservices that do not play well with others.

- **A centrally managed platform prevents 'technical debt' in production.** The tools used to deploy a new functional microservice determine the non-functional ('middleware') requirements that accompany it. If each functional microservice uses a different pipeline that is supported with different tools, it will interoperate with an idiosyncratic middleware stack. This causes problems known as a 'technical debt' when it is brought together as part of a larger system and needs to interoperate. The unintentional results of tooling diversity include bugs, security vulnerabilities, data leakage and poor performance in production. Developers may use further tools to try and fix these technical debts, thereby compounding the original problems, or they may bypass tooling altogether, introducing other vulnerabilities. The additional rework adds cost and time and prevents the reuse of business microservices across multiple applications, both of which negate a major benefit of cloud-native development: speed.

- **A centrally managed platform supports end-to-end delivery process automation.** It is difficult to fully automate certain aspects of the delivery process when developers use a diverse choice of tools without effective governance. Test automation is an example. Different testing tools have different strengths and emphases, so a pipeline using one testing tool may not adequately handle test cases applied by a tool in another pipeline. This leads to inconsistencies in the test automation process that may require a manual fix. Without automation, operators will find it difficult to roll out microservices at scale or make rapid and frequent changes to microservices in production.

Figure 6.1: A centrally managed platform supports the building of consistent, end-to-end systems



Source: Analysys Mason, 2019

6.3 Approaches to acquiring a platform: build-it-yourself versus third-party-provided

CSPs face a choice when it comes to implementing a cloud-native software delivery platform. Should they build such a platform themselves? Should they use a general cloud-native IT application delivery platform? Or should they invest in a telecoms-specific, carrier-grade platform that is tuned for telecoms industry design patterns and requirements, such as high availability, resilience and operator use cases?

CSPs gain several advantages from using a third-party platform. They can delegate the following responsibilities to the third-party platform provider.

- **Curation of a best of breed set of tools.** The cloud-native tool landscape is at an early stage of evolution and is therefore expanding rapidly. If operators elect to build their own platform, they will need to keep track of hundreds of small tool vendors. Many such companies are start-ups that lack the ability to support large-scale users and produce tools that are designed for IT deployment environments. They do not have the resources to evolve their tools to support carrier-grade requirements. A third-party platform provides a curated set of cloud-native tools based on the platform provider's knowledge of the market and its ability to select and combine best of breed products.

- **Identification, onboarding and integration of new tools in line with the rest of the market.** A platform provider must invest in ingesting the new tools that continually emerge in the dynamic, cloud-native tools environment in order to provide a platform that is aligned with webscale best practices. As CSPs become increasingly DevOps-capable, they will want to explore and work with leading-edge tools, but will need these to be integrated with their pipeline in a safe and efficient way. A third-party platform provider has the right resources to ensure that the platform is kept up to date without compromising its consistency and security.

- **Provision of tool maintenance and support across the pipeline.** Maintaining a cloud-native tooling platform is not a CSP's core business, and supporting such a platform can be a distraction from value-added activities such as building new business microservices.

- **Participation in open-source communities that are developing key platform technologies.** Many of the tools used for cloud-native software delivery are based on open-source developments. It is important that a platform builder participates in these open-source communities in order to gain early experience of the technologies and to influence their evolution. Most CSPs are unlikely to have the resources to do this.

If a CSP decides to use a third-party platform provider, it should evaluate it from the following perspectives.

- What level of support does the platform provider provide? Training and 24/7 support will be key to third-party platform acceptance.

- How widely is the platform used in the industry? Ideally, the platform provider will act as a center of tools excellence across a number of organizations to gain a wide range of insights into tool performance and problems. It will then be able to make use of economies of scale in maintaining the platform, to the benefit of all users.

- How are tools and best practices kept up to date? The platform provider should have one process for change management across all the tools and components in the platform for consistency and efficiency.

- Does its platform support architectural patterns that have been validated with leading industry experts? The platform should embed industry best practices for cloud-native software development and deployment, as recommended and tested by respected practitioners.

- What carrier-grade features does the platform provide in terms of performance, reliability, availability and scalability? The platform should be fit to operate in a demanding, mission-critical and high-scale telecoms environment.

7. Amdocs as a case study for cloud-native transformation

7.1 Amdocs's drivers for change

Amdocs is a world-class developer of mission-critical software systems for the telecoms industry. As a software company, Amdocs has always been committed to using best practice delivery processes and tools that enable it to deliver its systems to customers as quickly and cost-efficiently as possible.

Amdocs's customers have been coming under increasing pressure to innovate at speed. Amdocs knew that it had to become more agile in the delivery of its software to help its customers to respond quickly to unpredictable and dynamic market demands. Amdocs needed to bring new software features to market faster in order to support CSPs' changing requirements. Amdocs also needed to make it easier and quicker for CSPs to ingest these features. Amdocs wanted to change the cadence of its software releases so that new features could be delivered more frequently in smaller increments, instead of making CSPs wait 2 years at a time for a new release of its software, which had all the capabilities that the CSPs had asked for, but which they had to absorb all at once.

Amdocs had to change the way it designed and built its software in order to become agile. The company recognised the power of cloud-native technologies to increase the speed and efficiency of its own software delivery process and decided to pivot its business to incorporate them. Amdocs started on the radical path of redesigning its entire suite of applications in 2015. The company wanted to deliver business functions as loosely coupled microservices running on a standardized, centrally managed platform instead of providing its products as monolithic applications. The new approach required a transformation journey for Amdocs's employees, the re-engineering of its software delivery lifecycle and the building of a common pipeline and tools platform, Microservices360, as a 'paved road' to help developers focus on business logic.

7.2 Lessons from Amdocs's transformation journey

Amdocs recognised that it was putting its business at stake by undertaking such a far-reaching transformation, so it wanted to follow industry best practices every step of the way. The company attributes its success to several key factors.

- **Putting the right software architecture in place from the start.** Amdocs designed an end-to-end cloud-native architecture for its entire portfolio. Amdocs's end-to-end thinking ensured that it decomposed its portfolio into the right domains and subdomains; that is, into purposeful microservices that can be efficiently reused by its own applications and by those from third-party partners, as well as by applications co-created with customers.

  Amdocs understood that, in a cloud-native architecture, it is not enough to simply expose existing monolithic functions through APIs; it had to properly decompose and rebuild its software. Amdocs uses industry standard APIs (such as TM Forum's Open Digital Architecture APIs), where they exist, to guide its domain structure and the data that each microservice exposes. Open APIs underpin Amdocs's ability to recompose microservices into larger application structures.

  Amdocs incorporated another important principle into its end-to-end architecture: each domain must be loosely coupled and each microservice must have its own lifecycle so that it can be released through a CI/CD pipeline independently of any other microservice within its or in other domains. This is key to the agility that Amdocs wanted to build into its delivery process. It means that Amdocs can quickly and easily make changes to parts of its portfolio without affecting others, and gives customers drip-feed access to new features, thereby breaking Amdocs's traditional biannual release cycle.

- **Starting small and building incrementally.** Amdocs designed an end-to-end architecture for its portfolio, but it is implementing that architecture iteratively, in small steps, domain by domain. This has allowed Amdocs to test the right approaches, tools and practices over time and to win the hearts and minds of its developers by example. The result has been to create a motivated and effective development community that is eager to embrace new methods. Now Amdocs is seeding the wider industry with the best practices that it has put in place as it co-develops features and products with CSP customers and partners.

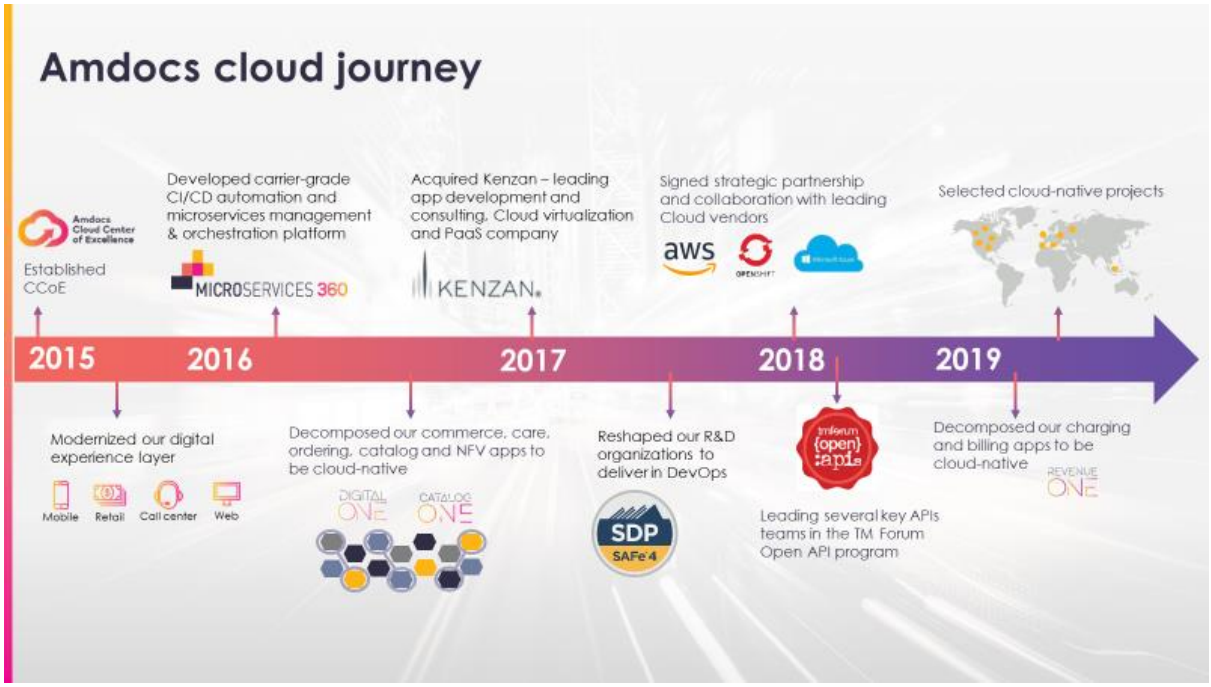
- **Embracing an open culture.** Amdocs took webscale companies as its models and understood that it cannot build all the software innovation that it needs in a fast-moving, cloud-native world, and nor does it need to. The company decided that its traditionally closed culture was no longer fit for purpose. Instead, Amdocs has embraced open-source software, which it is committed to incorporating into its products. Open-source software is rapidly becoming a source of industry standards, so its adoption enables Amdocs to closely align its products with those standards and to innovate more quickly with their support.

However, Amdocs realised that the success of its strategy depended on managing the disruptive cultural changes that an open, cloud-native approach brings. It sought help in making the required organizational changes from leading industry practitioners, and bought a cloud-native change management consultancy, Kenzan, to bring the right cultural skillsets in house (see Figure 7.1).

- **Automating a common pipeline.** Amdocs understood that it could not give developers endless flexibility when building microservices. The company's software had to be flexible in deployment so that Amdocs could run it in different customer environments, such as AWS, Cloud Foundry or OpenShift. However, to build that software, its developers needed a single pipeline, supported by a common tooling platform. Amdocs recognised that a common approach to automation was the only way of delivering scalable, resilient microservices rapidly into production.

  The company therefore developed a centrally managed platform, Microservices 360 (MS360), based on well-established open-source technologies and tools as an end-to-end ecosystem for its developers. MS360 incorporates best practice software design patterns that Amdocs has validated with recognized cloud-native development experts, and the platform is a source of consistent, secure and carrier-grade non-functional microservices in order to boost developer productivity and compliance.

Figure 7.1: Amdocs's cloud-native transformation journey



Source: Amdocs

7.3 Benefits from Amdocs's cloud-native approach to software delivery

Amdocs put KPIs in place so that it can measure the impact of its transformation on software delivery. The company has seen large improvements in its ability to carry out the following functions.

- **Deliver new features at speed.** The velocity of Amdocs's software design, development and testing processes has increased by 25% as a result of new practices. These include the creation of a dedicated team for non-functional requirement development (which enables developers to focus on business logic and

reduce technical debt), the availability of reusable design templates, the cross-pollination of skills and ideas, the generation of boilerplate test code and improved end-to-end non-functional test automation.

- **Reduce the cost of operations.** Amdocs has reduced operational costs by 30% due to the shared automation behaviors of all its microservices, the development of higher quality code that needs less troubleshooting and the establishment of a dedicated team to implement operational features.

- **Move to a continuous release cadence**. Amdocs has decreased the customer release adoption time from 1 month to 1 day due to its high levels of test automation. It now releases new features every month, rather than in an 8-month cadence.

In addition, Amdocs can offer its customers greater flexibility in the choice of microservices that they consume. They can also choose to extend Amdocs's solutions with microservices from third-party partners and through co-creation. These capabilities make it easier for customers to integrate Amdocs's solutions into their BSS/OSS environments, and to customize them for competitive differentiation and/or in response to new market opportunities.

8. Conclusion

Like Amdocs, CSPs need to master cloud-native software delivery, but they do not need to act alone. They can benefit from others' transformation journeys and the wealth of experience that is accumulating in the software industry. This makes cloud-native transformations less daunting and improves the likelihood of success.

CSPs will need to address all three aspects of transformation: people, processes and technologies. Their developers and operations staff will need to be equipped to work in new, agile and collaborative teams with re-engineered and highly automated processes that are underpinned by new tools. The most effective means of governing new DevOps teams is through a centrally managed common platform to ensure that all teams produce well-tested microservices that can be deployed into production in a consistent way.

It is transformation best practice to start with a small, pilot project that trains a single team and focuses on a single domain. However, CSPs should first understand the architecture system that they are developing from an end-to-end perspective to ensure that they have scoped its domains correctly, with future reuse in mind. They should determine the delivery process to follow through a value stream mapping exercise and use an appropriate platform to support a high level of process automation from the outset.

CSPs can accelerate their transformations if they engage a partner to help them with key aspects, such as changing employee practices, process definition and platform building. An external partner brings valuable knowledge of industry best practices, successful techniques for overcoming organizational challenges and relevant skillsets, which can be difficult to acquire otherwise. CSPs that use a vendor-supplied platform to support cloud-native delivery do not have to spend resources on keeping up to date with the latest tool developments or on platform maintenance and support.

Amdocs's cloud-native transformation story demonstrates the benefits that can be achieved and the best practices that are critical for success. CSPs should start their transformation journeys as soon as possible to ensure that they can exploit an emerging generation of cloud-native software systems, from BSS/OSS to cloud-native network functions (CNFs) and IT applications.

9. About the author

Caroline Chappell (Research Director) co-ordinates Analysys Mason's digital transformation research and contributes to the *Digital Infrastructure Strategies* research program. Her research focuses on service provider adoption of cloud, and the application of cloud technologies to fixed and mobile networks. She is a leading exponent of SDN and NFV and the potential that these technologies have to enhance business agility and enable new revenue opportunities for service providers. Caroline investigates key cloud and network virtualization challenges, and helps telecoms customers to devise strategies that mitigate the disruptive effects of cloud and support a smooth transition to the era of software-controlled networks.

This whitepaper was commissioned by Amdocs. Analysys Mason does not endorse any of the vendor's products or services.

Published by Analysys Mason Limited • Bush House • North West Wing • Aldwych • London • WC2B 4PJ • UK
Tel: +44 (0)20 7395 9000 • Email: research@analysysmason.com • www.analysysmason.com/research

Registered in England and Wales No. 5177472