FUJITSU

White Paper
Containerized Applications – A Way to Fast IT?

Digitalization requires Fast IT with development and operations going hand in hand. In order to make the idea of DevOps, micro-services and containerized applications a reality, infrastructures and management capabilities are needed that provide the speed and agility which are decisive for the success of the business.

Content

IT challenges continue to grow

Since the birth of IT, businesses have been reliant on it to support and drive innovation/productivity. Yet the slews of challenges that have followed the growth of IT have overridden the emerging flux of new technologies built to solve them. Despite the ever-growing progress in technology, the challenges have been persistent.

Nothing is as constant as change; that's why IT has to prove everyday its capability to flexibly adapt to the demands of the business and to keep pace with the light-speed development digital transformation has across all industries. And however much you try to keep your ear to the ground regarding innovations and processes for services, the priority should always be maintaining business continuity without disrupting the workflows already invested in your Robust (traditional) IT infrastructures. Collaborating Fast IT with your current infrastructure to formulate a hybrid environment is the focus of all IT managers, as is the understanding where the real strategic value is in the applications that provide the services essential to the business.

A history of applications

Applications have undergone various changes over the last decades. In the early days, people used **monolithic applications**, developed and deployed as large, single entities. The upside: They were easy to deploy. However, their further development was extremely expensive, because developers had to know and understand the inner workings of the applications. Maintenance was extremely complex, because the effects of minor changes in any other part of the application were not foreseeable. Moreover, any change had to run through a complete testing and deployment process, which was the reason for those long release cycles. Furthermore, monolithic applications cope with increasing volumes of data using a scale-up approach; so when they reached the limitations of the available server resources, further growth demands can only be met by replicating the entire application to one or multiple servers (scale-out) which is costly and a waste of resources.

The next step in the history of application development was **multi-tier applications**, consisting of usually 3 tiers: data, business logic and presentation, or even further tiers, such as authentication and reporting. The strict separation enables the distribution of the individual layers to different platforms, and therefore a more efficient scalability. When the workload increases, the business logic layer can be scaled, independently from the other tiers. Likewise the data can be replicated independently, too. However, it is worth mentioning that the business logic layer is still rather large, which poses challenges similar to monolithic applications.

To counteract the drawbacks of monolithic and multi-tier applications, the concept of a **Service-Oriented Architecture (SOA)** was born. The idea was to create applications as a collection of smaller services developed for a certain purpose, communicating with one another through a so-called Enterprise Service Bus (ESB). Compared with the aforementioned concepts, SOA enabled a more effective scalability and connecting a wide range of applications written in different languages. The flip side of the coin: introducing the ESB as an extra layer caused costly configurations, and at the end of the day this concept could hardly hit the speed of the business, due to long implementation periods.

The new trend: Micro-services

That's why it was high time to bring something into the discussion which can keep pace with the speed of the business: This is the micro-service approach. The idea behind it is fairly simple and similar to the SOA approach. You decompose an application (as complex as it might be) into small atomic, encapsulated entities, so-called micro-services, with a limited functional scope, working together. The development of these micro-services can happen fully independent from each other. After their deployment, they may be dynamically composed to complex applications, while the communication between the micro-services happens through language-agnostic APIs (Application Programming Interfaces). A flexible distribution of micro-services to various platforms enables a concurrent processing of multiple tasks.

Micro-services simplify development and maintenance, improve responsiveness to changing demands, enable scalability at individual service levels and contribute to cost-effectiveness.

As said before, the basic idea is not too different to SOA; but everything is just simpler and leaner. That's why some people denote micro-services as "service-orientation done right".

Micro-services – A driver for DevOps

Micro-services do not just represent a new form of application design; they are above all the basis for another new trend which is DevOps. DevOps is an artificial term used for blending the tasks performed by a company's application development team and its systems operation team. DevOps supporters do not see development and operations as separate activities, but rather the necessity for a close collaboration and harmonization of the two (i.e. architects, developers, testers, and operational experts) from the very start.

The benefits of DevOps are obvious: software can be built, tested and released more quickly, more frequently and more reliably. Time to deployment is cut and you will get customer feedback directly, which in turn will lead to better quality products, more innovation, and faster resolution of problems. The ability to continuously update application functions will increase user satisfaction. All this is perfectly supported by micro-services. Moreover, your IT infrastructure will be better utilized, which translates into cost savings – with regard to CAPEX and OPEX.