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.

But what does reality often look like?

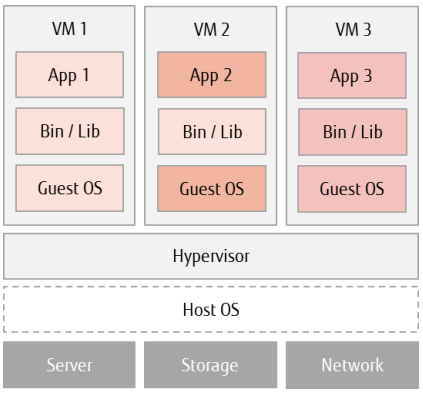
Considering the evolution of applications which are vital to the strategic value of an organization, we have to admit that reality often looks different in today's organizations. And by the way, this is a situation that we all know too well.

Application developers write programs in a development environment with all dependencies in place. The application is forwarded to the IT department for deployment. But then it often transpires that not all dependencies are in place in the production environment, or at least not the right software versions of them. The result: the application is unable to run and trouble-shooting starts. A lot more frequently the question that comes up is whether there is a way to put together what is needed for an application and package it in a single entity.

A possible answer: Virtual Machines

One possible answer is certainly virtual machines, which include all that is needed for applications. The impact: a simplified and fast deployment, a fast reaction on new business demands, and a guarantee that things will be functioning at the end of the day. So, why should the development team not just pack an application and its dependencies into a virtual machine and then pass it on to the operations team?

Because there are some drawbacks. In every virtual machine you need the guest operating system, as well as binaries and libraries which are often identical in multiple virtual machines. This will have a negative impact on the resource requirements; i.e. workloads running on a server will reach the given limits earlier. Furthermore, the time taken to migrate virtual machines or downloading them, as well as backup and restore will be time-consuming. And not to forget: the process might have to be repeated multiple times, depending on the number of virtual machines. This applies to updates and upgrades of the software and the virtual machines maintenance in general. It also applies to implementing high availability mechanisms and the likes having a decisive impact on business operation. All this takes effort and time. In addition, virtual machines cannot be moved around easily, because there are different types and formats for different environments, such VMware, Microsoft, KVM and others.

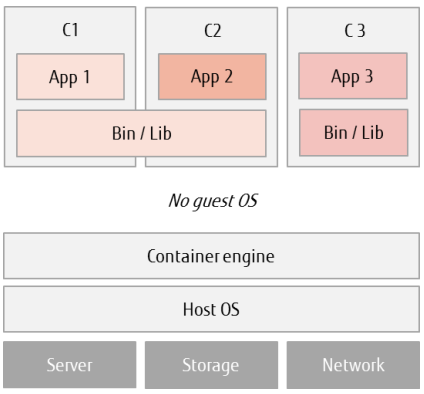


So the idea suggests this question: is there a more efficient way to deal with these challenges?

Containers

Of course there is a more efficient way - and the answer is containers - although they are not exclusively designed for micro-services. Containers are executed as isolated processes in the user space of the operating system and include usually one application. They share the same operating system kernel, the relevant binaries, e.g. device drivers, and necessary libraries. . At runtime, everything which is necessary to run the application will be packaged inside the container.

In comparison with virtual machines, containers provide a number of advantages. Containers are small, consume fewer resources, and enable running heavier workload per server. They cause less swapping, and increase the overall performance. The start-up happens much faster; we are talking of less than one second (if the image has already been downloaded to the local registry), whereas booting the operating system of virtual machines usually takes minutes. This means, that the typical performance bottlenecks caused by boot storms occurring in conjunction with VDI (Virtual Desktop Infrastructure) - when many users boot their virtual workplaces at the same time - are a thing of the past. Migrating containers between servers, downloading them from their repository, as well as backup and restore happens much faster.



A compelling advantage is portability: Development can happen locally on a laptop, and the result can run in any environment, be it on said laptop, the data center on the customer's premises or some cloud provider. Building and configuring applications once and running them anywhere is the motto of containerization. The speed advantages underline that new business demands can be met much faster.

The applications can be managed much easier, because environments are more homogenous and therefore simpler to manage. And the fact that there is just a single instance of the operating system for multiple containers results in simplified maintenance, too.

The capabilities in terms of scalability are enormous; whenever necessary, a new instance of the same container may be run on the same or on multiple servers.

Apart from all this, it should not be ignored that utilizing fewer resources and simplified maintenance translate to CAPEX and OPEX savings. Containers provide customer value by enabling in an easy way to focus on new features, fixing issues and shipping new or updated software.

Due to all these advantages, it is no surprise that containers have become increasingly attractive. Analysts predict that by 2018, over 50% of all new applications will be deployed in containers.

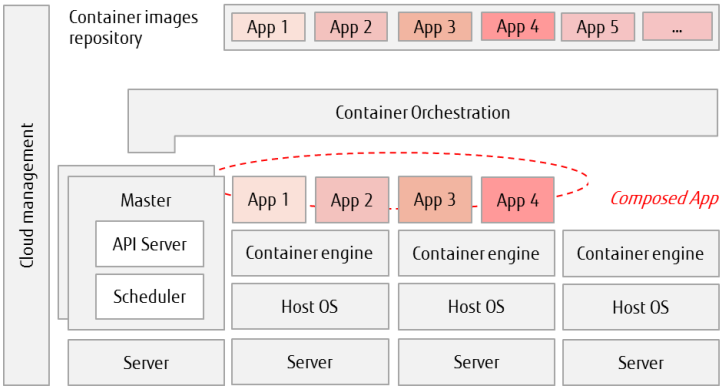
Micro-services and containers: Challenges

Despite all the advantages that micro-services and containers provide, there are also some challenges we should not ignore. In practice, you might have to deal with a big number of services; and you will be bound to describe all these services and their dependencies. You will have to deploy this great number of services and set up a test environment for the composed application. Apart from this, it will be all about a solid and reliable orchestration, which should also help reduce production incidents by catching problems before they occur. In order to always ensure a predictable and acceptable performance, functions such as failover, scaling and load-balancing should be conducted in an automated manner. In other words, it is all about simplified management and maintenance of the infrastructure, in order to save countless hours of wasted effort and frustration.

Container deployment

Let us now have a closer look at the infrastructure needed for containers and how to deploy it. In theory, containers can be deployed on a single server; however, for the sake of increased performance and availability it is highly recommendable to deploy multi-container applications on a set of nodes controlled by orchestration software. Therefore, usually one of the first steps is to create a cluster consisting of a master node (which typically should exist redundantly) and a number of worker nodes. The master and worker nodes will then be deployed with an operating system and the runtime system for running the containers, which is also denoted as container engine. In addition, a sort of an agent is needed to ensure communication between the workers (slaves) and the master. If it is only containers running on the nodes, it is recommended to choose a lean operating system specially optimized for containers in order to get the most out of the available resources and increase security.

Although it is not a must, we should always take into consideration that a (private) cloud infrastructure might provide enormous advantages due to its self-service feature and rapid provisioning capability. It gives software developers the freedom to quickly request and get what they need, perfectly supporting the DevOps idea. In this case, the deployment phase will also include getting the respective cloud management software up and running.



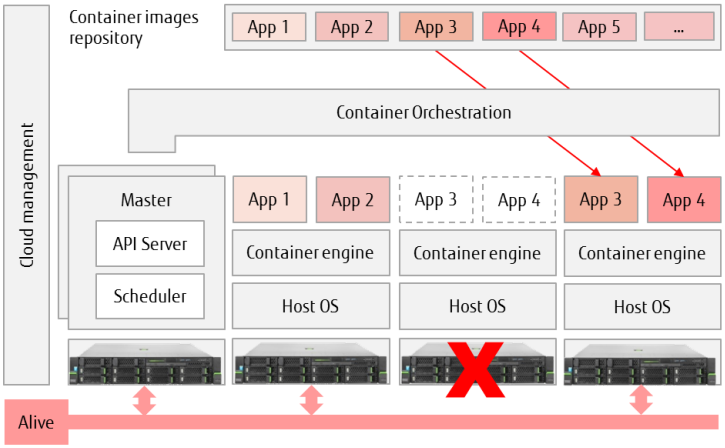
Meanwhile developers can build the micro-services and store the images including all their dependencies in a container repository. When starting a containerized application, the container orchestrator implicitly starts all containers running the micro-services that make the application; due to the defined dependencies, they are needed as prerequisites. The master node selects the worker nodes available for running the containers, while the workers in turn download the respective container images from the container repository and execute them. It's also possible that all workers share a shared storage volume where the images from the registry are stored, so the image just needs to be downloaded once.

Finally, the downloaded containers are interconnected according to their dependencies.

Automated failover

Cluster orchestration is more than just allocating containers once to available worker nodes. What if a server fails? To detect a server failure in good time, the master node continuously checks, if the worker nodes are alive.

Once a node failure is detected, because no alive-signal is returned from that worker node, the master node selects another worker node to take over the work of the failed node. The selected node downloads the respective container images from the container repository, starts the containers, and the operation can go on. This failover mechanism should of course become effective in an automated fashion.

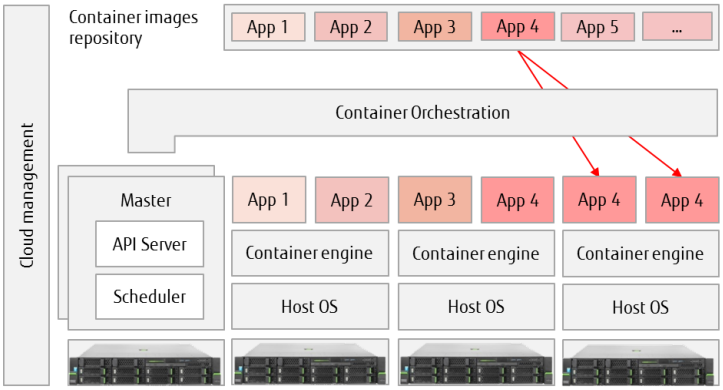


Automated scaling

The objective of container operation is to save resources when their utilization by the applications is low, but on the other hand, ensure that the applications are responsive, even if resource utilization is high. For this reason, the master should permanently measure resource utilization and moreover even apply various additional metrics and evaluate logs.

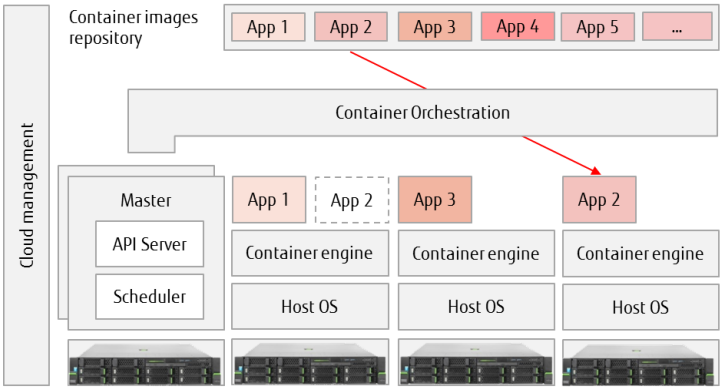
Once the bottlenecks are identified at runtime with regard to a certain container, the container will be replicated. The new container instances can be created on the same worker node, and / or on one or multiple additional worker nodes and run there in order to improve responsiveness. For web applications, the network traffic is automatically dispatched to the newly created container.

This automated scaling is also applied vice versa. If it turns out that it is no longer necessary to run several instances of a container in parallel, containers will be removed.



Automated load-balancing

A beloved method to get the best performance out of an existing infrastructure is distributing the overall workload equally to the available worker nodes in the cluster. This is achieved by automated load-balancing, which is another essential function of container orchestration.



Storing container data

Containers are a perfect fit for stateless applications, which means that data generated in one session is not stored for the usage by another session. But many applications require data to be stored persistently. A typical example is a 3-tier application, where the front-end and the middle tier are typically stateless, but the database is stateful.

Container orchestration simplifies the usage of persistent storage for containers: Block storage provided by the infrastructure management system can be registered to the container clusters as "data volumes". These are attached to the requesting container at time of the container start-up. For the container, the volume appears as a regular mounted directory which can be used to store data. Once the container stops, e.g. because of a node failure, the data remains in the data volume and can be used by another container.

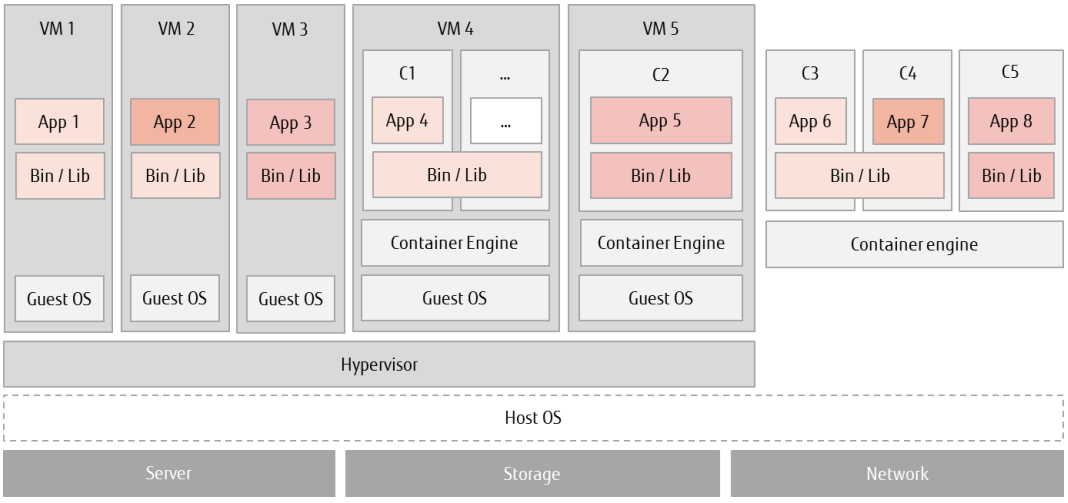
What about security?

An important aspect has not been considered until now, which is security. Containers run in the user space of the operating system, thus sharing the operating systems kernel. Consequently, a failure of the operating system can create a system-wide outage, i.e. at least this particular server plus all containers running on it will be affected. This insufficient isolation may also cause security concerns, for a kernel attack could compromise all containers. With trusted applications and single tenancy, e.g. applications originating all from the same development community, it would be quite unlikely that this will happen. In the event of multi-tenancy, there might be more uncertainty, and people are well advised to counteract this uncertainty. But, is there a way out and what does it look like?

Combine virtual machines and containers

The aforementioned issues can be circumvented by combining containers and virtual machines. You create a virtual machine or a set of virtual machines with Linux as operating system, e.g. per each tenant, for isolation. You run the container engine in the virtual machine and your containerized applications on top of it. The virtual machine isolates the container engine from the underlying operating systems kernel. Thus, a compromised container will not affect other containers. All told, the benefit is a higher degree of isolation and hence a higher level of security. Advanced technologies such as network virtualization can be used to improve security and embedding the container landscape into an overall Software Defined strategy for the Data Center.

There is no general recipe what is better or worse. Our advice is to decide at deployment time what to choose, depending on the level of isolation required. The subsequent figure shows the various concepts described.

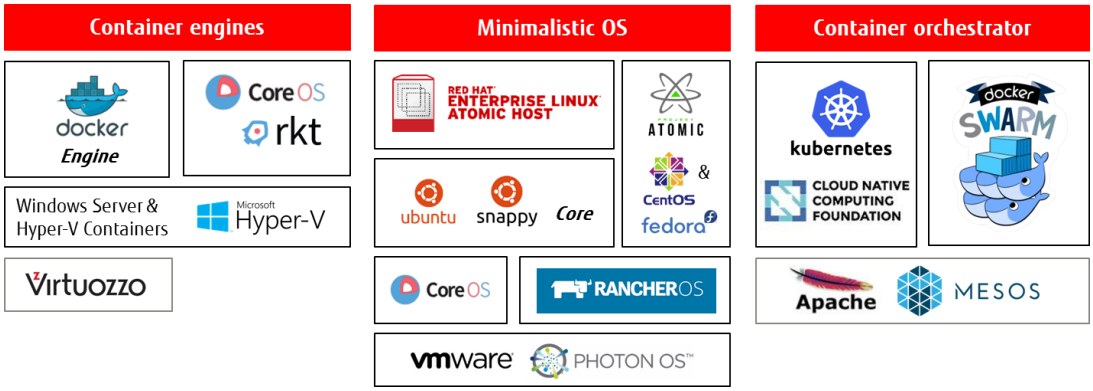


Container eco-system and its analogy

After having laid the foundation for the theme of micro-services and containers, let us summarize the components you will always meet in a container eco-system. Firstly, there are the container images which represent the container applications or micro-services running in containers. Secondly, there is the container engine, which is the runtime environment enabling the execution of containers. The container engine may run on a traditional operating system or a minimalistic operating system which is specifically designed and optimized for running containers. And thirdly, there is a container cluster manager or container orchestrator, in charge of deploying containers across the cluster, as well as orchestration, monitoring and management in general.

To make the roles of these components better understandable, an analogy could help. The container orchestrator may be compared with an architect whose task is to have a grand building built from many bricks. In our example, the grand building would be a complex application, and the bricks would be the containers. But there is a huge difference between constructing the grand building and running our micro-services: constructing the grand building may be a years-long project, whereas running containerized applications will happen with a few clicks of a mouse.

The following figure shows examples of container engines, minimalistic operating systems and container cluster managers, which can be found in the market, as of this writing.



Building an infrastructure is complex

Hopefully by now we have a clear understanding of containers and their benefits. But how do we get there? There is no doubt that building a data center infrastructure is complex. The main reason for this is the complexity of the infrastructure itself, which is composed of diverse components, such as servers, storage, networks, virtualization layers for all these components, databases and other middleware, as well as applications. In addition, a management layer is needed to keep all components under control.

When building the infrastructure in a Do-It-Yourself (DIY) approach, you need to select the right components, sized in the right manner, from a myriad of possibilities, procure and configure them, before you integrate the individual components onsite. As the compatibility of the components is not guaranteed, extensive testing is a must. The fact that these components may originate from multiple vendors does not make things easier. All these activities are time-consuming and expensive, whilst at the same time presenting businesses with multiple risks if things don't work as desired. A deep knowledge of all components involved is required as is an understanding of their various dependencies on each other. Often, the coordination among the various administrators who are in charge of the individual components seems to be endless. And as every installation is different, maintenance will be complex, too. Therefore the question suggests itself, if there is a better way to go for container infrastructures.

FUJITSU Integrated System PRIMEFLEX

Of course there is a better way to go: FUJITSU Integrated System PRIMEFLEX, a pre-defined, pre-integrated and pre-tested combination of data center components, which provides computational power, storage capacity, network connectivity and software with management software being a mandatory part. Customers have the freedom of choice to implement PRIMEFLEX solutions for Robust IT, based on servers, storage and network components, or as a Fast IT approach; developed as serverized solutions for scale-out scenarios. Based on best practices and real-life project experience, PRIMEFLEX systems are designed in a way that their components will work optimally together.

The benefits resulting from Integrated Systems are manifold. All of a sudden, complexity is reduced; introducing a new infrastructure in your data center becomes much simpler. You will experience less trouble through trial-and-error testing, because the compatibility of all components is absolutely guaranteed. At the same time, risk is minimized and the skill sets required in your IT department will be less demanding. Apart from this, you need less time for planning, and deployment is tremendously accelerated which shortens time to production and time to value. Due to the optimized design of Integrated Systems, resource utilization is optimized, too. This can have a positive impact on data center space, cabling, energy consumption and cooling efforts. Moreover, an Integrated System represents a perfect foundation for efficient operations and reduced maintenance efforts. All these aspects help reduce cost, both CAPEX and OPEX. Finally, we should not ignore the fact that all these benefits enable IT organizations to focus on the really important aspects of the business. Moving away from a build and maintain focus means improved responsiveness to new business requirements, or even driving business to a new level.

The PRIMEFLEX line-up includes converged and hyper-converged systems, both built from best-in-class components, either from Fujitsu itself or leading technology partners. PRIMEFLEX systems are either pre-installed in the factory, and arrive ready-to-run at the customer's site; or they are delivered as reference architectures giving you the flexibility to adapt them to your specific requirements. On demand the adjusted configuration can be pre-installed and delivered ready-to-run, combining the advantages of reference architectures with those of ready-to-run systems. For all PRIMEFLEX reference architectures, installation and configuration guidelines are available as a standard.

PRIMEFLEX is supplemented by services throughout all lifecycle phases, either delivered by Fujitsu or our local partners.

Fujitsu has the longest track record in terms of integrated systems; the first system being shipped in 2002. Since then, we have continuously optimized our processes for end-to-end solutions, be it in product management, quality assurance, manufacturing and support. Meanwhile, we can refer to an impressive line-up market and many customer references.

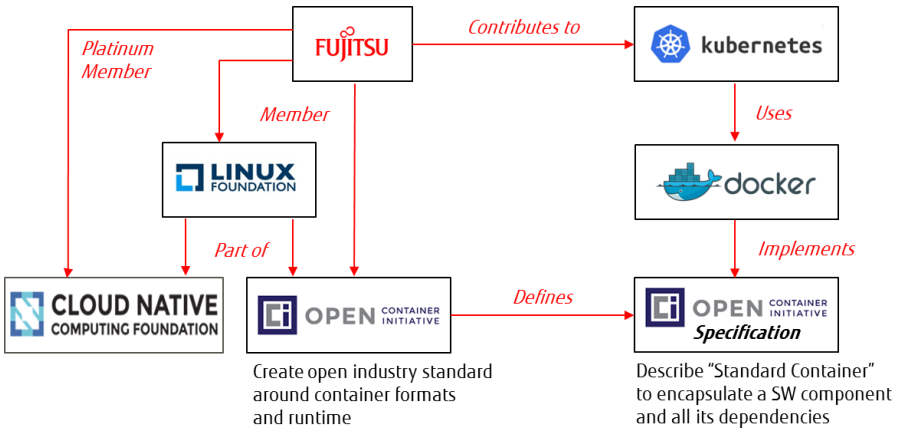
With our PRIMEFLEX family, we address use cases of high relevance. Among these are of course virtualization and cloud infrastructures supporting containerization. FUJITSU Integrated System PRIMEFLEX combines a high-performance and energy-efficient Fujitsu hardware stack, a market leading virtualization or cloud platform (VMware, Microsoft or OpenStack), one stop support and a comprehensive professional service portfolio –all in one solution package.

Fujitsu's engagement in container technology

What is Fujitsu's role in container technology? First of all, it is worth mentioning that Fujitsu is heavily engaged in various boards looking after container technologies, standards and related themes. Fujitsu is a member of the Linux Foundation and a platinum member of one of its projects, the Open Container Initiative, whose mission it is to create open industry standards around container formats and runtime environments. The Open Container Initiative defines the Open Container Initiative Specification, defining how to describe dependencies between containers and how to encapsulate software components. The Open Container Initiative Specification serves as a basis for implementing the Open Container Initiative RunC which in turn is used as a runtime system for Docker, which is the de-facto standard of container engines. Apart from this, Fujitsu is a platinum member of the Cloud Native Computing Foundation, helping to drive the adoption of computing paradigms optimized for modern distributed environments.

But it is not only the involvement in these boards that makes Fujitsu an important player in the area of containers. Our deep involvement also helps us recognize where the gaps are with existing open source products and what needs to be done in order to make some of these great ideas ready for the enterprise. Therefore, Fujitsu actively contributes e.g. features, documentation, bug fixes to the open source projects Kubernetes and Monasca; for Kubernetes especially in the area of dashboard and cluster provisioning, and for Monasca in the area of log management. Moreover, Fujitsu completes the container eco-system by a number of software products which make containerization even more attractive for organizations, and indeed already today. There is the Open Service Catalog Manager, a self-service portal open sourced by Fujitsu, which gives developers and other users all the freedom they need to easily request infrastructure resources on demand. And there is Fujitsu's Cloud Monitoring Manager based on OpenStack Monasca with advanced metrics enabling a better resource utilization, a better overall performance and a better end user satisfaction. By means of Fujitsu's UShareSoft AppCenter, existing applications can easily be separated from their runtime environment, making them appropriate for containerization.

As far as the cloud infrastructure and the cloud management software is concerned, Fujitsu selected OpenStack. The main reason is Fujitsu's decision to migrate all its internal IT to OpenStack. The vast experience gained from this undertaking is also used in Fujitsu's public cloud service offerings; and therefore it is quite natural that OpenStack is used for on-premise container deployments, too.



Summary

In the era of digitalization, everybody speaks of Fast IT. Fast IT may require DevOps, DevOps in turn require micro-services, containers are ideal for executing micro-services. Containers require the right infrastructure to run on. Open source software, such as OpenStack, Docker and Kubernetes, supplemented by Fujitsu's upper layer software products, such as ServerView Cloud Monitoring Manager, Enterprise Service Catalog Manager (also known as Open Service Catalog Manager open source project) and more, all based on Fujitsu's reliable server platforms PRIMERGY and PRIMEQUEST are certainly a perfect foundation. And Fujitsu's consulting expertise will guide you into the right direction. That's exactly what our Business-Centric Data Center approach is about. But at the end, it's about deploying the entire infrastructure – in a simple and fast way, at minimized risk – and it's about operational efficiency. And this is exactly where FUJITSU Integrated System PRIMEFLEX comes into play.
All told, if you have to have Fast IT, DevOps, micro-services and containers on your agenda, it's worth having a word with us.

Contact
FUJITSU Technology Solutions GmbH
Address: Mies-van-der-Rohe-Strasse 8,
D-80807 Munich, Germany
Website: www.fujitsu.com/primeflex
2018-11-03 WW EN