Comparing containers & microservices

A post from [Ev Kontsevoy](https://goteleport.com/blog/microservices-containers-kubernetes/" \t "_blank) summarized the comparison of these two terminologies in an interesting way:

*“A container is a useful resource allocation and sharing technology. It’s something DevOps people get excited about. A microservice is a software design pattern. It’s something developers get excited about.”*

In other words, we can sum this up as:

* Microservices are about the design of software.
* Containers are about packaging software for deployment.

So, we can choose whether to use a container for hosting a microservice. But to get full value from both, it is significantly better to run microservices within containers.

Deploying an entire application to a single VM introduces a single point of failure risk, whether or not a microservice architecture has been used. But spreading the application through microservices across multiple containers results in fully exploiting the value of both by providing resilience as well as agility through scaling and improvements targeting specific services without negatively impacting the entire application.

Flexibility is also introduced in that [developers](https://www.bmc.com/blogs/application-developer-roles-responsibilities/) can write applications in the [language of their choice](https://www.bmc.com/blogs/programming-languages/) since the container will allow them to deploy across whatever environment is provided. Efficiency comes from containers using less resources compared to VMs.

An added benefit comes in the form of security through isolation and a broader attack surface that limits the impact should a single microservice or container be subject to a security breach such as a hacking attack.

## Challenges with containers & microservices

The drawbacks that come with using containers and microservices are tied to the management overhead especially when dealing with large scale distributed deployments. This means that deployment, monitoring, and management of containers and microservices at such environments would be quite challenging and require specialized tools that can support orchestration and ensure consistency in deployment.

Other challenges include:

* Complexity from managing microservices written in different languages
* Cost implications of network resource usage from remote calls across multiple services
* Investigating [root causes](https://www.bmc.com/blogs/root-cause-analysis/) or auditing systems becomes challenging when dealing with log management across distributed services, as log aggregators would be required

## Future of containers & microservices

The need for modern applications that provide both agility and resilience cannot be understated in the age of cloud and mobile apps. Benefits such as faster time to market and enhance security are value propositions that the business will gladly accept.

Containers and microservices are currently the preferred approach for scaling and refactoring [legacy](https://www.bmc.com/blogs/application-software-modernization/) applications to make them cloud native.

Powered by Kubernetes and Docker as well as the growth of hybrid cloud deployments and edge computing, the market for these capabilities is expected to continue growing, with [MarketWatch](https://www.marketwatch.com/" \t "_blank) predicting a CAGR of 12.7% for the global cloud microservices market, reaching a value of $1.7 billion by 2027.