Capgemini | Microsoft

DESIGNING A SCALABLE, HIGH AVAILABILITY AND FUTURE-PROOF PLATFORM

for Cloud native compute

Introduction

Designing a scalable, high availability and future-proof platform is a cornerstone for many cloud projects. A popular design pattern to achieve this goal is a microservice architecture. In this architectural style, a single application is architected as a set of multiple, loosely coupled and independently deployable components called microservices. In order to build and run enterprise level cloud native applications, cloud providers offer services with modern technologies that are cost-efficient, fast and reliable.

In Kubernetes, all microservices are packaged into containers that incorporate software code and all of its dependencies into a standardized unit. The container orchestration and all configuration are managed by Kubernetes which acts as a platform to execute and run multiple and complex container workloads. The second option is to use serverless functions that execute the software code. Cloud providers allocate and manage the underlying infrastructure on demand, taking care of the servers on behalf of their customers. A single Function App can contain multiple functions whereas each function corresponds to a method in programming language. Within microservice architecture, a Function App corresponds to a single microservice and can therefore scale independently.

The existence of both options already indicates that there is no silver bullet. In the next chapter the readers will find the necessary guidance to make a well-founded decision between the two approaches. A special focus is on how to combine the advantages of both approaches. The presented solution allows you to start with functions to minimize time to market and then evolve into Kubernetes later on. Turnkey solutions like Kubernetes Event-driven Autoscaling (KEDA) and Distributed Application Runtime (DAPR) are discussed as well. KEDA adds autoscaling in Kubernetes to achieve parity between functions. DAPR is intended as general distributed application platform on top of Kubernetes with support for application patterns such as publish-subscribe. It reduces integration complexity and allows to reach beyond Kubernetes due to its abstraction. The base for this insight has been the close collaboration with Microsoft and Capgemini along with cloud project experiences such as Microsoft gold partner.

Function Apps versus Kubernetes

Function Apps come in various flavors and Kubernetes is often combined with other software such as service meshes. A stricter definition regarding the assumed configuration is therefore needed. Function Apps can be based on a consumption, premium or app service plan. The App service plan provides additional features which are rather relevant for special cases such as high-end scenarios.

Therefore, Function Apps in this paper refer to the maximum functionality resulting from the consumption and premium plan only. Service meshes focus on simplifying the internal communication between containers in Kubernetes. Function Apps do not come with orchestration support anyway and therefore service meshes are not relevant for this comparison.

The focus of the comparison is on differentiators, since they decide which option to go for. Areas with comparable support are as follows:

- **Security**

Both services provide configurations that minimize the attack surface. Function Apps benefit from the managed infrastructure and additional features such as private deployment. Azure Kubernetes Services also allows rich security features coming out-of-the-box, e.g. by using the private cluster option.

- **Local Deployment**

Both can be run on-premises to reduce costs (Kubernetes via Minikube, Azure Stack, OpenShift; Function Apps in multiple ways as downloadable software).

- **Scalability**

Both provide scaling out-of-the-box. Scaling in Kubernetes comes with several options that require thorough design considerations. Options include manual versus automated scaling and scaling on a cluster versus on a pod level. The Cluster Autoscaler scales on node level targeting things like node pools or Virtual Machine Scale. The scaling on the pod level can be achieved by using the Horizontal Pod AutoScaler which scales on metrics like CPU or memory. By using KEDA, external services can be used as triggers.

Autoscaling is provided in Function Apps in all three plans. However, the consumption and premium plan use a different mechanism which reacts faster than the App Service Plan.

All differentiators are rated using a three-value scale. The values have the following meaning:

- **'+'** : feature is supported with no additional effort
- **'o'** : feature is supported, but requires additional efforts in administration and configuration
- **'–'** : feature is not supported or would requires tremendous additional efforts

Area	Differentiator	Function Apps (Microservice)	Kubernetes (Microservice Platform)
Architecture	Orchestration features for multiple microservices (high availability, recovery)	–	+
	High compute power	–	+
	Time-to-market	+	o
Development	Application configuration feature toggles)	+	o
	Portability (developed code)	o (Possible by wrapping into container)	+
	Integration in Azure native services	+	o (Starting point with additional frameworks)
	State support	+	o (Possible but easier supported with function apps)
Operations	Runtime restrictions	–	+
	Portability (platform)	o (Possible by wrapping into container)	+
	Complex Deployments e.g. rolling update)	– (Deployment slots can support but do not allow fine-grained component control as in Kubernetes)	+
	Monitoring	o (App insights required needs to be created and linked)	o (Azure Monitor, Application Insights and OSS components like Prometheus can be used)
	Costs	+ (If below threshold consumption) – (If premium since more expensive VMs than Kubernetes)	o (No costs for Kubernetes orchestration control plane, but no free tier)

A clear platform indication can be given if you have an edge case scenario that is clearly small or complex. Complex might refer to big computing power or orchestration support in a full blown microservice platform.

Function Apps are a very good match for the small end, e.g. due to a single application focus. Since they maximize infrastructure abstraction, they are very easy to set up with a fast time-to-market. Certain features such as state support and integration with other Azure services are better supported compared to Kubernetes. Especially for test environments where the call threshold is no problem, they are a great leverage for saving money. Here, the deployment slots allow an A/B testing. The following reasons can speak against functions. Kubernetes needs not automatically to be an alternative due to its additional complexity:

- Function runtime restrictions
- Existing ecosystem/knowledge different from functions
- Portability

Kubernetes is clearly the winner in complex scenarios since it comes with a lot of built-in orchestration functionalities and since it can be designed more specifically on the user needs. E.g. clusters can be designed to use VM types with GPU acceleration or other requirements and associate them only to specific apps hosted on the platform. Additionally, Kubernetes enables much more mature deployment options—for example, if you need a platform which supports A/B testing or a component wise rolling update.

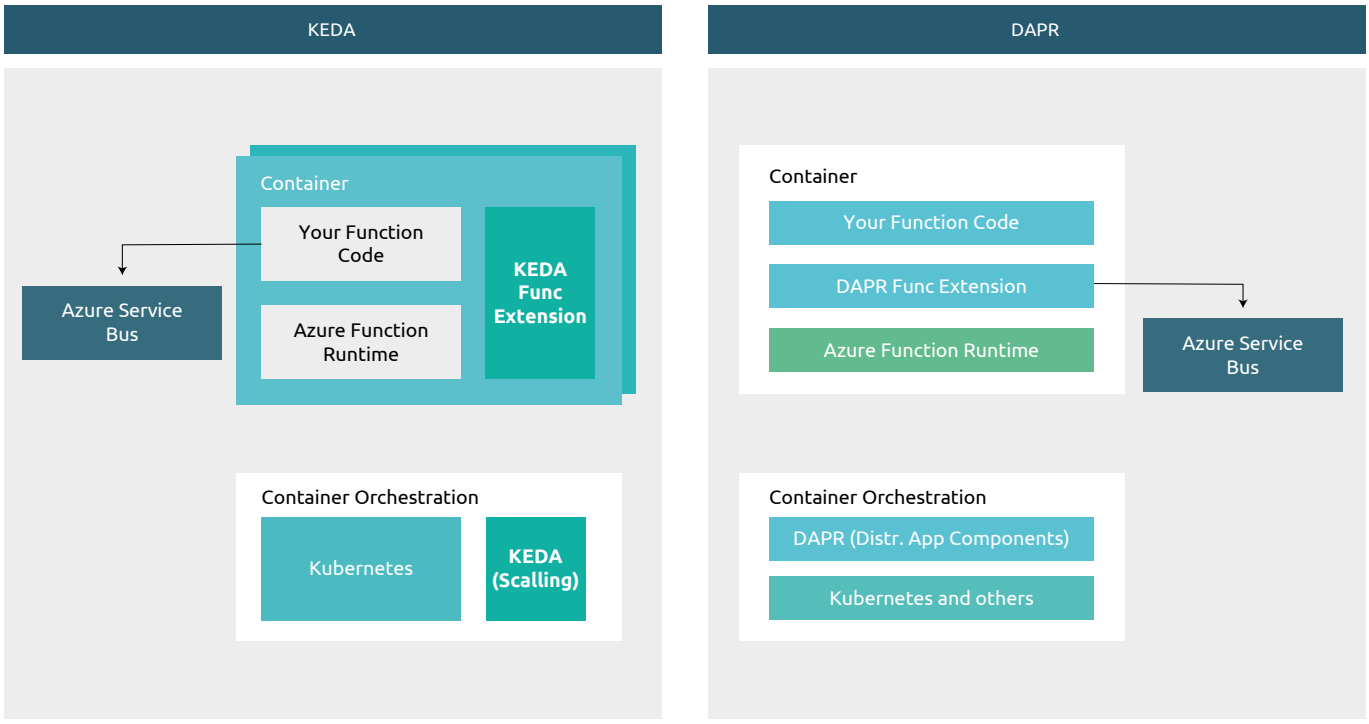
Kubernetes can be combined with additional technologies such as KEDA and DAPR that both support functions. Supported scenarios in conjunction with Function Apps and Kubernetes will be discussed in detail in the next chapter.

Can Kubernetes and function apps work together?

KEDA is an open source, single-purpose and lightweight component that can be added into any Kubernetes cluster. It works alongside the standard Kubernetes components and enhances the Horizontal Pod Autoscaler functionality to scale container instances. KEDA is designed to activate and scale Kubernetes deployments depending on the needed capacity and triggered by event outside of the Kubernetes cluster. KEDA acts as a metrics adapter that forwards internal or external metrics to the Horizontal Pod Autoscaler to drive scale out. Typical examples of triggering external metrics include RabbitMQ and Azure Service Bus etc.

DAPR is a portable, open source, serverless and event-driven runtime that makes it easy for developers to build resilient, stateless and stateful microservices that run on the cloud and edge and embraces the diversity of languages and developer frameworks driven by Microsoft. DAPR addresses developer needs or 'application-level constructs' such as manage state or to subscribe to a pub/sub messaging system. It abstracts from the underlying services implementing these constructs. This applies for the orchestration platform as well as the application level constructs it provides. Thus, a subscribe implementation of a RabbitMQ-based publication can be replaced with Azure Service Bus without having to change the application code.

The following figure illustrates major architectural components that are used in both set-ups:



Both extend the Kubernetes as the orchestration platform and the function part to integrate with DAPR and KEDA. In both cases, the function needs to be containerized to be executable under Kubernetes including the Azure Function Runtime. The KEDA function extension is only required if you use durable functions. The two pictures also illustrate the different focus of the two frameworks. KEDA concentrates on scaling container instances. It is possible to run Function Apps without KEDA but then you lose the autoscaling feature provided by the consumption/premium plan. DAPR abstracts from application level constructs such as the service bus and Kubernetes as execution platform.

It cannot be executed on function apps as a hosting environment and the 'DAPR Func Extension' is not yet released.

Running function apps unchanged in Kubernetes including autoscaling is therefore only possible with KEDA. Known Function App features still work under KEDA. This includes bindings and state support by using the durable functions extension for KEDA. Also, other services frequently used in conjunction with Function Apps integrate well with Kubernetes. This includes Azure app configuration service and API management by combining them with an ingress controller within Kubernetes.

Starting with Function Apps and DAPR is not possible since the extension is not yet recommended for production, and Azure Function Apps as a hosting platform is not supported. However, DAPR is a future way to plug-in function code in Kubernetes, but it requires a porting if you start with Azure Function Apps. In most cases the porting is straightforward since comparable concepts, such as for state management and binding, exist. Differences only result from different syntax or different involved APIs. No counterpart exists for the orchestration patterns provided by durable functions such as fan in/out or time based.

Summary

As illustrated, there is no single approach that addresses everything. When making the right choice, the starting point is to determine the required level of complexity in terms of computing and orchestration support. Thanks to KEDA, Function Apps can be a fast entry point even if the project switches to Kubernetes later on. Kubernetes is a portable container platform, but the features you can use in conjunction with functions (State support or bindings targeting certain Azure services) are Azure specific. On the other hand, DAPR does not support Function Apps as hosting platform, but aims higher in terms of portability. Azure specifics regarding state support and bindings are hidden from the application code. Moreover, Functions in conjunction with DAPR are not yet ready for production. However, creating distributed apps that abstract from Kubernetes and the underlying application components are already possible.

Used Resources

- Azure functions documentation: https://docs.microsoft.com/en-us/azure/azure-functions/
- API management documentation: https://docs.microsoft.com/en-us/azure/api-management/api-management-kubernetes
- Azure App Configuration: https://docs.microsoft.com/en-us/azure/azure-app-configuration integrate-kubernetes-deployment-helm
- Kubernetes home: https://kubernetes.io/
- Azure Kubernetes Services: https://docs.microsoft.com/en-us/azure/aks/intro-kubernetes
- DAPR Concept Overview: https://docs.dapr.io/concepts/overview/
- DAPR on Kubernetes: https://docs.dapr.io/operations/hosting/kubernetes/kubernetes-overview/
- Microsoft – Technical customer story on DAPR: https://customers.microsoft.com/en-gb story/1335733425802443016-ignition-group-speeds-development-and-payment-processing-using-dapr-and-azure
- KEDA home: https://keda.sh/
- Azure Functions on Kubernetes with KEDA: https://docs.microsoft.com/en-us/azure/azure-functions functions-kubernetes-keda
- Azure Functions on Kubernetes with KEDA (part 1): https://microsoft.github.io/AzureTipsAndTricks/blog/tip277.html
- Azure Functions on Kubernetes with KEDA (part 2): https://microsoft.github.io/AzureTipsAndTricks/blog/tip278.html
- Repo KEDA Func Extension: https://github.com/kedacore/keda-external-scaler-azure-durable-functions
- Repo DAPR Func Extension: https://github.com/dapr/azure-functions-extension.

Authors

Christian Weber
christian.a.weber@capgemini.com

Güncel Düzgün
guencel.duezguen@capgemini.com

Albrecht Schönfeld
albrecht.schoenfeld@capgemini.com

Co-authors

Andreas Mock
andreas.mock@microsoft.com

Christian Dennig
christian.dennig@microsoft.com

About Capgemini

Capgemini is a global leader in partnering with companies to transform and manage their business by harnessing the power of technology. The Group is guided everyday by its purpose of unleashing human energy through technology for an inclusive and sustainable future. It is a responsible and diverse organization of 290,000 team members in nearly 50 countries. With its strong 50 year heritage and deep industry expertise, Capgemini is trusted by its clients to address the entire breadth of their business needs, from strategy and design to operations, fueled by the fast evolving and innovative world of cloud, data, AI, connectivity, software, digital engineering and platforms. The Group reported in 2020 global revenues of €16 billion.

Get the Future You Want | www.capgemini.com