


Super Cloud or Meta Cloud: A Problematic Approach

Stephane H. Maes¹ 

November 25, 2022

Abstract

Over the last year, several vendors and evangelist have taken on to promote, and push, super clouds, also known as meta clouds, as the future of cloud computing, or at least a latest trends.

It is not clear that it is an actual trend yet, or just the pe project of a few. Part of the problems that we see with super clouds may come from shifting all-encompassing definitions, which can mean whatever you want them to mean.

In any case we argue that such a super cloud, as an abstract layer across clouds, may be a problematic approach, from a business viability, architecture and efficiency point of view.

Using management tools and practices for multi cloud, hybrid cloud and developing cloud native applications is a better approach, with more manageable challenges.

1. Introduction

Clouds used to be categorized as on-premises/private, public and hybrid, the latter being combinations of the on-premises and public cloud, as well as sometimes legacy data centers. Distributed computing and distributed applications can also be added to the list.

Clouds themselves have evolved to include, or distinguish, between hypervisors on servers and / or bare metal, IaaS, PaaS, and possibly additional services available as SaaS. Then containers have been added, and with them we brought in cloud native concepts (a combination of [2,3], and [4]), and with orchestration services like Kubernetes (K8S) [1], available roughly on every environment as a managed PaaS service or container execution service or for installation on the IaaS.

Separately, virtualization vendors have provided infrastructure/cloud in a box, think Nutanix or HPE Evergreen, or cross cloud and data center environments: think of VMware vSphere and vCenter on AWS.

Since roughly 2010, different IT Operation Management (ITOM) tools have been developed (non-exhaustive examples):

- Automation tools that automate deployment or provisioning of IaaS, PaaS and /or applications
- Infrastructure as code, which define desire state of the infrastructure and let it be provisioned and maintained
- (Self-service) cloud orchestrators or Cloud management systems (e.g. HP CSA [6], then Micro Focus HCMX, or VMware vRealize), that allows to design, and order particular services deployments or applications.

¹ shmaes.physics@gmail.com

- Declarative description of deployment, and desire states, like cloud formation terraform (multi-cloud) and YAML for Kubernetes.
- Full packaging declarative descriptions like HELM for Kubernetes.
- Day-two management tools of infrastructure, and / or workloads
- Self-healing solutions as with AIOps and / or Kubernetes.
- DevOps and CI/CD tool chains and pipelines to develop, deploy and manage applications on cloud.
- Pre-defined cloud deployment templates as say SaaS or DevOps environment like Gruntworks.io [5].
- Tools to move VMs from on-premise to clouds, and even sometimes, from physical servers to VMs first. For example consider what was Micro Focus PlateSpin [7], or Veeam offerings.
- Etc.

Recently, super cloud, or meta cloud have been proposed and promoted as the future of cloud.

2. Super clouds

Many have promptly claimed that super cloud amount to their preferred offerings arguing:

- That super cloud goes beyond multi-cloud to include on-prem and the edge [8]. That would imply that super cloud is just what was called hybrid or multi-cloud. At least that's what HP /HPE for example called hybrid cloud year ago [8].
- That super cloud is "a cloud architecture that enables application migration as a service across different availability zones or cloud providers" [8].
- That offer a seamless experience across clouds and on-premises data centers through an abstraction layer and offering one place to build, develop and deploy, it intends to eradicate the troubles of maneuvering through different clouds [10].

[10] then provides a whole list of super clouds.

With this almost any tool, or solution, supporting the ability to deploy on, manage, or move between than one cloud "qualifies".

However, we believe that the intention captured behind super cloud is rather the latter bullet above, including in particular the abstraction layer. We will take it as the definition for the rest of the paper, unless if indicated otherwise [].

3. Kubernetes

Building applications on Kubernetes, as containers, or cloud native applications in general, can be a way to achieve most of the objectives of super clouds: If the applications are built to be able to use both container run times, or K8SaaS (aka CaaS), as well as Vanilla CNCF Kubernetes, portability across clouds is straight forward, except for some issues, related possibly to the annotations for ingress, and services, or resources relied on. External services can be used via API, like REST interfaces. In general, these would not be portable. So, for example, if an Azure service, say cognitive or chat bot, is needed, then the application will have to continue to use it, even if moved on premises, or to AWS. Otherwise, it could also be developed with the ability to use another bot service if deployed on AWS.

Alternatives like developing for a K8S platform like OpenShift, installable, or provided, on most cloud and on-premises, is an alternative. Only that, in some cases, OpenShift is to be installed also any new target cloud, before deploying or moving to a different cloud or data center. Again, in general, external service services would have to remain on their cloud.

Resources needed by the application are also to be considered. A persistent volume, or a database, may have to be migrated, or synched, and if the data service behave differently, the application may have to be tested an adapted for the different cases.

In our example with the chat bot, API may changes, but also one chat bot may be NLU based and require training for a domain, while another bot may be more rule based and require grammar and dialog rules instead. This is an effective way to see also as described later on, why it is impossible to abstract many of the cloud services, certain beyond common denominators at the PaaS level or even sometimes at the IaaS.

4. Platforms

As an extension to K8S, platforms running on different clouds, e.g., think a J2EE web service can be similarly considered as satisfying the super cloud use cases, as long that external services and resources are similarly handled.

OpenShift (as middleware, like it J2EE middleware) is a good example.

Note that platforms like public cloud PaaS, are in general not fitting the super cloud use cases: services, resources and environments can be quite different from one cloud to another. Adapting the application or developing it to fit the different PaaS would be required.

5. Resources

Some resources, like databases, ingress and ingress controller, firewalls and routers can be deployed on all cloud (and on-premises), or even be found as IaaS managed services across them. Like for example Oracle, Postgres, F5, etc.

But even then, if the ingress service is provided by different vendors, things may change with some some implementing proprietary annotations, while other requiring separate configuration of routing rules.

It is really important to understand that even an apparently same service can differ. For example, for a long time on Azure, Postgres was the window version, at the difference of AWS or Google Cloud, where Postgres was implemented on Linux. As a result, performances and capabilities, like acceptable numbers of open data base connections, were quite different: applications ported to Azure required new settings, and sometimes rewrite or different architecture, if I/O intensive or data centric (analytics, AI etc.).

6. Virtual Machines

Of course migrating workflows to VMs, and the moving of virtual machines from on-premises to clouds, and in between clouds, is another popular approach. A good example is Micro Focus Platespin [7].

With tools like Densify, formerly Cirba, for example, the moves within a data center or across cloud can optimize performance and workload [10]. More advanced optimizers like what was offered by Composure.ai, formerly MosaixSoft, can do the same for containers or Kubernetes [11].

As long that network is correctly configured, and external resources handled as prescribed above, this will also satisfy the super cloud use cases.

VMs can now also take advantage of Kubernetes management with tools like KubeVirt [13], and many related variations.

7. Cloud service brokers

Cloud service brokers like HPE CSA, now Micro Focus HCMX, vRealize, and VMWare Tanzu, especially in its Telco variation, can be seen as powerful ways to have a single pane of glass for ordering, deploying and day-2 management across multiple cloud, data centers and edges. They also satisfy the use cases of super clouds.

ITOM and DevOps tools, are and can be similarly handled, orchestrating the different cloud and providing a single pane of glass.

8. On-premise or edge clouds in a box

Offerings like for example Amazon Outpost, Microsoft Azure stack, and similar offerings of others, also provide the ability to efficiently get cloud services on-premises or at the edge. This approach also addresses the use cases of super clouds.

9. Limited need for the Super Cloud Patterns

In the above sections, we have already seen that it is possible to satisfy the use cases of super cloud patterns in many different ways, without requiring the invention of new concepts or new offerings and abstractions like super clouds. It is a question of having the appropriate tools, or designing, migrating the applications appropriately. Designing new applications to be cloud native facilitates addressing much of the needs.

Sure, it could be conceptually ideal to have an abstract layer that is implemented across all relevant cloud: public, on-premises and at the edge, and then services would be obtained from the layer as available APIs and workloads would abstractly run on the abstracted cloud.

The author having pushed such approaches with Telco SDPs at the Parlay, then OSE (OMA Service Environment) at OMA (Open Mobile Alliance) [14-17], and offerings like SDP (Service Delivery Platforms, which is an ancestor of middleware and web services for service providers, Cloud APIs, PaaS, API GW and micro services patterns) at IBM, Oracle, Huawei and Ericsson, agrees with the a priori value proposition, but argues that the situation is different with clouds, as discussed in the next section.

Unfortunately, such a design can easily lead to technical inefficiencies, business challenges and application design and management limitations.

10. The challenges with super cloud patterns

There is a risk that the abstraction layer design will introduce too many inefficiencies versus designing to be able to accommodate equivalent services in different clouds. Let us consider the following examples:

- It is easier and more efficient to have a model where workloads run on explicitly identified known clouds rather than abstracting again the capabilities of a server, network or storage offered in an abstract cloud itself abstracted on the services already abstracted by the different cloud providers and their implementations.
- Services beyond basic IaaS capabilities, but that can also apply to them like for computing services, may be uniquely differentiated in what to configure, or what features and behaviors are offered. Remember the chat bot example of section 3. Abstractions will at best be able to offer the lowest common denominator, or, always use the same cloud service for certain capabilities thereby unable to satisfy the super cloud use case. Offering all possible options to handle all cloud versions is plausible but hard and it risks to rapidly become a nightmare for developers, customers and super cloud providers.
- Traffic out of, and / or into a public cloud is typically expensive, and “slower”:
 - Relying on a services always implemented in a cloud, from another cloud may be outrageously expensive for the super cloud provider, or its customer. With the abstraction layer, this may be harder for developers or customer to understand and optimize.
 - Oracle approach to do so with Azure [18], AWS and Google cloud is an example showing that it is possible. But that is just for one service, one database vendor, and only these 3 public clouds and OCI. Furthermore, it is not an abstraction but mostly rather a deal to cover costs and boost speed and bandwidth to use on Azure, or the other clouds, Oracle autonomous DB, actually provided and managed in OCI.
 - Moving data, or synchronizing data from a cloud to another, even for equivalent services can be extremely inefficient, costly for service provider or customer and hard to manage if through abstractions that hide these activities.
- Who pays what?:
 - If services are abstracted, to switch / delegate implementations, to one or another cloud, then who is responsible for the inefficiencies of the implementations, and, in particular, of the cost that will depend on hidden, because abstract, implementation choices and policies / settings.
 - What to use or what is used will typically be very hard to understand, and truly hidden to the customer if pure abstraction is achieved?
 - Again think of the Oracle DB example from above. It is clear because Oracle bills and then pays azure or AWS. That is what a super cloud service provider would have to do. But across many services and dynamic changes, good luck to be able to easily build a price list or explain what happens when/
- Prices for services across clouds are often non-comparable, and dynamics; sometimes function of the customer tier in each cloud provider view. Abstract services prices would be even harder to determine by the customer, and a challenge for the super cloud provider.

- Services unique, or uniquely differentiated, for a cloud, are examples where abstraction makes little sense, and customer or developers must understand, and be able to control what when we know that traffic from one cloud to another will introduce extra cost and inefficiencies. Otherwise, again, super cloud services will be the lowest denominator, or will behave differently based on what cloud is used at a given moment to provide the services. That is simply not a viable model.
 - Developers can't develop to handle that through an abstract API. Multi cloud is possible if the cloud target is known. It all falls apart if it isn't because of abstraction, and yet capabilities or behaviors change.

It is important that the differences come between this and patterns like OMA OSE and EPEM, SDPs and API gateways, come from the fact that:

- Telco enablers are expected to be standardized.
- Network capabilities (functions) are expected to be standardized.
- And OSE was supposed to address exposure internally or to third parties à la Telco 2.0 [20], from a given operator or a multi-national / multi-network operator à la Huawei multinational SDP. The capabilities were same or equivalent and standard and traffic between networks were internal to the operator or operator alliances. Therefore it was not encountering the same issues, inefficiencies or cost. That being said, Telco 2.0, not SDPs, failed because different operators did not expose all the same capabilities at the same time, forcing the over-the-top (OTT) developers to only be able to target subscribers of a same operator or alliance and never reaching scales favorable to their business model. The entry of device based APIs with iOS and Android frameworks, were not operator dependent and sealed the faith of Telco 2.0.

The IT/Internet/Cloud world is not the same as Telco. Interoperability across provider is not essential at the system level, and standards are seen as too slow and stifling innovation. Only underlying protocols are typically standardized, and possibly some programming language specific syntax or APIs, as was the case for example for Java JCP [21]. Open source for are also alternatives like for Linux [22] or CNCF for Kubernetes and cloud native ecosystem [3].

Without standards, new cloud services, capabilities behaviors and APIs can and will always be added and breakneck speed. Super cloud will always be behind with a subset, or lowest common denominator of services and at the mercy of any change of service, capability, features and prices, and new services. As public cloud providers will most probably not align to encourage such initiatives, it is naïve to expect their cooperations. And it will be so easy for them to break the principle at any time.

The jury is maybe open on Sylva, the new cloud initiatives of European Telcos [23-25]. Frankly, there also the prospect is grim, because risk are high that telcos will show that they have learned nothing, and create a telco vertical stack, not based on same technologies as IT and internet clouds. And they will not be able to compete without relying on the same economies of scales. It's been the curse of the telco industry to always believe that telco specific use case warrant vertical solutions instead of at best new requirements, contributions and best practices to use the same horizontal technologies as everybody else. Furthermore, [25] does not describe super cloud but rather Sylva being a cloud provider or Cloud stack.

11. Conclusions

The proposals for super cloud, aka meta cloud, as abstraction across public clouds, on-premises and edges, are red herring. They are not good technical designs, with many inefficiencies and not sustainable as a competing and business offering.

Most of the use cases behind the proposal of such an approach are desirable, but they can be supported with suitable solution and application design for cloud native, multi-cloud and hybrid cloud.

Despite the drawback, we know that many may be distracted by false prophets and follow the sirens of the idea. So super cloud may indeed be tried and offered in the industry. To do so, the vendor, implementor or provider would have to address the challenges we captured in this paper.

References

- [1]: Kubernetes, "Production-Grade Container Orchestration", <https://kubernetes.io/>.
- [2]: Wikipedia, "Cloud native computing", https://en.wikipedia.org/wiki/Cloud_native_computing. Retrieved on November 25, 2022.
- [3]: "Cloud Native Foundation", CNCF, <https://www.cncf.io/>. Retrieved on November 25, 2022.
- [4]: "The Twelve-Factor App Introduction", <https://12factor.net/>. Retrieved on November 25, 2022.
- [5]: Gruntworks.io, "Your entire infrastructure. Defined as code. In a few days", <https://www.gruntwork.io/>. Retrieved on November 25, 2022.
- [6]: Wikipedia, "HP Cloud Service Automation Software", https://en.wikipedia.org/wiki/HP_Cloud_Service_Automation_Software. Retrieved on November 25, 2022.
- [7]: Wikipedia, "PlateSpin", <https://en.wikipedia.org/wiki/PlateSpin>. Retrieved on November 25, 2022.
- [8]: ZINNIA BANERJEE, (2022), "Supercloud, the Next Big Trend in Cloud Computing. Supercloud goes beyond multi-cloud to include on-prem and the edge", August 22, 2022, <https://analyticsindiamag.com/supercloud-the-next-big-trend-in-cloud-computing/>. Retrieved on November 23, 2022.
- [9]: BRIAN NJUGUNA, (2022), "Supercloud simplifies cloud architecture while leveling up automation", August 16, 2022, <https://siliconangle.com/2022/08/16/supercloud-simplifies-cloud-architecture-leveling-automation-supercloud22/>. Retrieved on November 23, 2022.
- [10]: Wikipedia, "Densify", <https://en.wikipedia.org/wiki/Densify>. Retrieved on November 25, 2022.
- [11]: LinkedIn, "'Composure.ai'", <https://www.linkedin.com/company/mosaixsoft-inc-/about/>. Retrieved on November 25, 2022.
- [12]: KubeVirt, "Building a virtualization API for Kubernetes", <https://kubevirt.io/>. Retrieved on November 25, 2022.
- [13]: Wikipedia, "Open Mobile Alliance", https://en.wikipedia.org/wiki/Open_Mobile_Alliance. Retrieved on November 25, 2022.
- [14]: Stephane H. Maes, et al.(at OMA Architecture), (2004), "OMA Service Environment", OMA, https://www.openmobilealliance.org/release/OSE/V1_0-20040907-A/OMA-Service-Environment-V1_0-20040907-A.pdf.
- [15]: Stephane H. Maes, (2007), "Service delivery platforms as IT Realization of OMA service environment: service oriented architectures for telecommunications", 2007 IEEE Wireless Communications and Networking Conference.
- [16]: Stephane H. Maes, (2010), "Understanding the relationship between SDP and the cloud", CLOUD COMPUTING, 2010.

- [17]: Stephane H. Maes, et al.(at OMA Architecture), (2008), "Policy Evaluation, Enforcement and Management Callable Interface (PEM-1) Technical Specification", OMA, https://www.openmobilealliance.org/release/PEEM/V1_0-20080805-C/OMA-TS-PEEM_PEM1-V1_0-20080805-C.pdf.
- [18]: Oracle, "Multicloud with OCI and Azure", <https://www.oracle.com/cloud/azure/>. Retrieved on November 25, 2022.
- [19]: Wikipedia, "Service delivery platform", https://en.wikipedia.org/wiki/Service_delivery_platform. Retrieved on November 25, 2022.
- [20] Kandy, Io, (2020), "What is Telco 2.0?", <https://www.kandy.io/media/glossary/glossary/telco-20>. Retrieved on November 25, 2022.
- [21]: JCP, "Java Community Process", <https://www.jcp.org/en/home/index>. Retrieved on November 25, 2022.
- [22]: Linux Foundation, "Decentralized innovation. Built on trust.", <https://www.linuxfoundation.org/>. Retrieved on November 25, 2022.
- [23]: Frederic Lardinois, (2022), "LF Europe's Project Sylva wants to create an open source telco cloud stack", <https://techcrunch.com/2022/11/15/lf-europes-project-sylva-wants-to-create-an-open-source-telco-cloud-stack/>. Retrieved on November 25, 2022.
- [24]: Linux Foundation, (2022), "Linux Foundation Europe Announces Project Sylva to Create Open Source Telco Cloud Software Framework to Complement Open Networking Momentum", November 15, 2022, <https://www.linuxfoundation.org/press/linux-foundation-europe-announces-project-sylva-to-create-open-source-telco-cloud-software-framework-to-complement-open-networking-momentum>. Retrieved on November 25, 2022.
- [25]: GitHubs, "Sylva", <https://gitlab.com/sylva-projects/sylva/-/tree/main>. Retrieved on November 25, 2022.
- [26]: David Linthicum, (2022), "The next frontier in cloud computing Supercloud? Metacloud? The race is on to name the emerging layer of abstraction and automation that will remove the complexity of multicloud.", InfoWorld, <https://www.infoworld.com/article/3667371/the-next-frontier-in-cloud-computing.html>. Retrieved on November 25, 2022.