Layered Service Model Architecture for Cloud Computing

Vishal Kaushik, Prajwal Bhardwaj, and Kaustubh Lohani(✉)

School of Computer Science, University of Petroleum and Energy Studies, Dehradun 248007, India
vkaushik@ddn.upes.ac.in, kaustubhlohani25@gmail.com

Abstract. The National Institute of Standards and Technology defined three services models for cloud computing in 2011, namely IaaS, PaaS, and SaaS. Since then, a decade has passed, and cloud delivery methodologies have evolved, and newer ways of delivering cloud services have emerged. As a result of these new methodologies, lines between traditional service models are constantly fading, and new subcategories are emerging with totally different approaches to tackle cloud service delivery. This paper introduces a layered cloud service model to replace the three-category approach suggested by the NIST (IaaS, PaaS, and SaaS) in 2011. Moreover, we also discuss the existing approaches prevalent in the industry and the need for a new approach. Furthermore, we have mapped the existing services in the industry are mapped on the proposed architecture.

Keywords: Cloud service delivery model · Cloud computing · IaaS · PaaS · SaaS · Cloud definition · NIST definitions

1 Introduction

In the late 1990s internet was booming; consequently, several products and services were being created specifically for the internet. Hosting all of these products onto the internet meant leasing out servers. This proved expensive and inefficient as the organization had to pay for the leased servers even if they were not used. Cloud computing emerged as a cost-effective way for the digitalization of organizations by enabling computational resource sharing [11, 12, 15].

In 2011 NIST formally defined cloud computing and introduced three service models to offer cloud computing services based on business requirement, functionality, and control offered to the consumer. The three models are Infrastructure-as-a-Service (IaaS), Software-as-a-Service (SaaS), and Platform-as-a-Service (PaaS). A decade has passed since the introduction of these three service models, and since then, newer technologies like containers and serverless computing have come up that are not fitting in the categories of IaaS, PaaS, and SaaS [16].

Serverless computing provides a feature list that combines IaaS, PaaS, and SaaS; consequently, classifying serverless as one of the NIST service models is incorrect.

M. R. Khosravi et al. (Eds.): CloudComp 2021, LNICST 430, pp. 91–106, 2022.
https://doi.org/10.1007/978-3-030-99191-3_8

Similarly, containers are considered as an IaaS offering because both of them seem to abstract the underlying computational layers. This argument breaks apart once we consider that IaaS virtualizes over the underlying hardware using a hypervisor. In contrast, containers abstract over the underlying operating system deployed over the hardware using a container engine, making them slightly different from the IaaS service model.

A slight change in approach by contemporary cloud service models is creating new categories of delivery methods that are not falling entirely in a single category of services defined by the NIST (IaaS, PaaS, and SaaS) in 2011. Soon more contemporary delivery methodologies like these may be introduced, further blurring the lines between rigidly defined NIST service models. Thus, a novel service model architecture is the need of the hour that accommodates the exceptions of NIST architecture at the same time should be scalable to accommodate future delivery methodologies.

Unfortunately, not many such approaches are being developed actively. Johan Den Haan proposed one such framework in 2013 [1]. Johan suggested seven categories starting from bare metal and going up to SaaS applications. Another such approach actively utilized in the current industry to categorize cloud services is anything-as-a-service (XaaS). XaaS refers to any service delivered over the internet through the cloud, previously delivered through on-site methods. Both these approaches try to propose a way out of the NIST models; however, they have some shortcomings which we try to address with our proposed architecture.

In this paper, first, we describe the challenges posed by serverless and container, which prompts the need to think beyond NIST service models. Furthermore, we outline the existing approaches along with their shortcomings. Next, we introduce the proposed architecture. Moreover, we plot the current services offered by various CSPs onto our model. Finally, we highlight future research directions and potential challenges.

2 Important Terms and Definitions

2.1 Cloud Computing

NIST defines cloud computing as a "computing model that enables ubiquitous, convenient and on-demand access to a shared pool of computing resources (e.g., servers, storage, networks, applications) that can be rapidly provisioned and released with minimal effort from the consumer or the CSP" [2].

2.2 Cloud Service Provider or CSP

CSPs are organizations that provide on-demand cloud services to their clients. These services can provide computing power or infrastructure, a platform to execute application libraries and code, or an application to satisfy a business requirement. Moreover, CSPs also establish and manage public, private, and hybrid clouds [3].

Some examples of prominent CSPs are Google Cloud, Microsoft Azure, and Amazon AWS.

2.3 Infrastructure as a Service or IaaS

NIST defines IaaS as a service that provides the consumer with the ability to choose the required amount of computational resources – in the form of processing power, storage, and networks – upon which the consumer can run any software such as the operating system [2].

In IaaS consumer does not manage or control the underlying cloud infrastructure, including the hardware in the server farm on which the IaaS services are provisioned. Popular IaaS offerings include Amazon EC2, Google Compute Engine (GCE), and Azure Virtual Machines.

2.4 Platform as a Service or PaaS

According to NIST, the PaaS service model provides the consumer with the capability to directly deploy a custom application or acquired application on the cloud infrastructure using the programming languages, libraries, tools, and services provided by the CSP [2, 18].

In PaaS, consumer does not manage underlying cloud infrastructure. Moreover, the consumer is not concerned with computational resources, network, storage configurations, and operating systems. However, the consumer controls the hosting environment and hosted application code and execution [2].

Popular PaaS offerings include AWS-Elastic Beanstalk, Microsoft Azure, and Redhat Openshift.

2.5 Software as a Service or SaaS

NIST defines SaaS as a service model where the CSP provides the consumer with ready-to-use applications on the cloud infrastructure. These applications are accessible through several client interfaces such as a web browser or a program interface [2].

In SaaS consumer neither controls the underlying cloud setup nor the computational, network, or storage resources. Further, the control of the operating system and other such software capabilities resides with the CSP. However, in some cases, the consumer has limited control to alter some user-specific settings [17].

Popular SaaS offerings include Microsoft Office 365, Salesforce.com, and Google applications such as Gmail or Google Drive.

2.6 Serverless Computing

A cloud computing service model in which CSP allocates the computing resources as per the consumer demand, moreover maintaining the servers on behalf of the consumer. Serverless provides the consumer with the convenience to deploy application code or external libraries without the hassle of maintaining the underlying servers. Consequently, the client does not need to worry about keeping the server configurations up to date with the latest offerings and security patches; the CSP takes care of all of this. Moreover, serverless also provides the user with immense flexibility and scalability with features

like auto-scaling, which translates to scaling up the resources as the demand increases [10].

Some popular serverless hosting providers: AWS Lambda, Google Cloud Functions, IBM Openwhisk.

2.7 Containers and Containers as a Service (CaaS)

A container is an executable unit of a pre-defined or customized container image that generally encompasses one or more software programs to deliver an application or service [4]. A container is virtualized instance over the operating system created with the help of a container engine such as Docker.

CaaS or container as a service aims at providing users with services and tools to create, manage and organize container instances.

Some popular CaaS providers include Amazon EC2 Container Service, Azure Container Service, and Google Container Engine.

3 Need to Think Beyond IaaS, PaaS, and SaaS

As discussed in the introduction section above, contemporary technologies such as serverless and containers cannot fit in the NIST service models. Serverless and containers provide a slightly fundamentally different approach to the NIST-defined models in architecture, control, scalability, and cost.

3.1 Serverless vs IaaS, PaaS, and SaaS

Both serverless and PaaS attempt to hide the back-end for a more straightforward service model. In the case of PaaS, users are expected to manage the hosting environment – keeping the hosting environment up to date with the latest feature set – along with the deployed applications. However, as described above in Sect. 2.6 in serverless computing, the users are only expected to manage the deployed application with CSP taking up the responsibility of managing the servers and in turn executing the application code uploaded by the client, taking the control away from the client and placing it in the hands of CSP, consequently making the PaaS model more controllable for the client. Moreover, in serverless features, the pay-as-you-go subscription model where the client only needs to pay for the resources utilized by them as opposed to paying for the whole platform in the case of PaaS. Furthermore, Serverless supports auto-scaling, which means that the client automatically gets additional computational resources as the processing need increases; further, the allocated resources are taken back to the CSP pool once the need has gone down. Whereas PaaS usually does not feature auto-scaling and any jump in the requirement of the computational resources has to be taken up with the CSP, which will manually allocate the additional resources, however in most cases, there is no provision to take back once provisioned resources after the demand has decreased [13, 14].

Serverless is very different from the SaaS service model. Serverless provides the clients with the feature of executing application code on the cloud with the flexibility of CSP managing their servers, while SaaS is a whole end-to-end product offering over the

cloud. In serverless, the client can code, compile, and execute their application, giving the client control over the application and its configuration. In contrast, SaaS is a product that is in a ready-to-use state from the moment the subscription starts. Further, in a SaaS offering client only controls certain user-specific settings.

IaaS is an infrastructure offering that gives the client complete control over the processing, networking, and storage aspects. In contrast, serverless takes away most of the control offered by the IaaS for a more straightforward approach providing clients with a more flexible and scalable approach without worrying about maintaining the underlying servers. Moreover, IaaS gives the client freedom of choosing the operating system and altering its settings. In contrast, in serverless, the client is provided a bundles package that contains all the software and the dependencies required to smoothly run the application making the client responsible only for the application code, and CSP takes care of executing the application on their servers.

3.2 CaaS vs IaaS, PaaS, and SaaS

PaaS can be considered a complete application delivery system encompassing application development and delivery in a single package. Here, the CSP provides the user with the platform bundled with the required specifications allowing the client to deploy application code and execute it. In contrast, CaaS is a different service approach that provides the clients with a service to create, organize and manage containers using tools such as Docker, a container image, an orchestrator, and infrastructure to deploy. Essentially, CaaS provides clients with just a container service on which they can set up their hosting environment comprising of calibrating hardware configurations, required software like an operating system, and external dependencies or libraries. After setting up the hosting environment, the client can deploy application code to be executed via the software packages installed by the client.

On the other hand, PaaS is sort of like a bundled package in which the hosting environment setup is taken care of by the CSP according to the specifications set by the user, and the user can focus on application development and delivery. Consequently, CaaS as a service generally offers more control when compared with PaaS. Generally, CaaS and PaaS are used hand in hand, PaaS for the application development and CaaS for application deployment, making this system CaaS-PaaS hybrid [9].

On the other hand, if the service approach, feature set, and control offered are considered, SaaS and CaaS can be thought of as opposite poles of a magnet. SaaS is a complete ready-to-use (from the moment subscription starts) application used by the organizations for its functionality and benefits them in conducting their business, such as an electronic-mail service. In contrast, CaaS is a container service on which a hosting environment can be set up, and a custom-made application can be developed and delivered through it.

Finally, CaaS has a fundamentally different approach to providing service compared with IaaS, resulting in a difference in control offered by both the services; consequently, classifying CaaS same as IaaS will not be entirely correct. Both of these technologies use the technique of virtualization for service delivery. However, the difference is that IaaS virtualizes on operating system placed over the data center hardware and creates VMs (Virtual Machines) whereas, CaaS uses a container engine to virtualize over the operating

system, which is generally placed over an IaaS service. Using a container engine provides the client with easier horizontal scaling, meaning they can create additional containers without involving the CSP. In contrast, initializing more VMs in IaaS requires assistance from the CSP's end. However, since there is an additional layer involved in CaaS before the virtualization happens, which is absent in the virtualization process of IaaS, the level of control offered is more in IaaS.

4 Existing Approaches and the Need for New Approach

As discussed previously in the introduction section, very few novel frameworks talk about replacing the prevailing NIST system. However, people like Johan Den Haan [1] and Christine Miyachi [5] are suggesting changes since 2013. Furthermore, since the NIST system is ambiguous, the industry has decided to take a whole new approach wherein they categorize services based on the functionality they offer; for example, according to this system, a platform that provides disaster recovery tools and services is called DRaaS (Disaster Recovery as a Service) rather than PaaS. The model where categorizing services basis their functionality is done is referred to as XaaS (Anything as a Service). This section tries to explain the existing approaches (other than NIST service models), their shortcomings, and the need for another novel approach.

4.1 XaaS (Anything as a Service)

An approach used widely in recent times by CSPs is XaaS. XaaS refers to any service delivered over the internet that was previously delivered through physical means.

Naming a service on the lines of XaaS has gained popularity among the CSPs due to the marketing clarity it provides to their offering [6]. For example, a "Privacy as a Service (PRaaS) solution to take care of your privacy needs based on the geography" seems lucid than something like "Web-based SaaS for taking care of your privacy needs based on the geography."

Another factor contributing to organizations not using NIST service models to label their cloud offerings is the broad qualification parameters associated with each service. As time passed and new delivery methodologies came, these defined parameters became more outdated, leading to oversimplification of the features offered in a product. Essentially, the cloud technology matured, giving rise to novel service models such as serverless and containers. These novel delivery methods provided features that were a little different than those defined for IaaS, PaaS, and SaaS. So, if a CSP decided to label their product as per the NIST model, prospective clients could assume it provides the features described by the NIST and not further explore the product features, proving detrimental for the CSP's business.

At first, XaaS seems like a potential replacement for the NIST model. However, XaaS methodology solely focuses on the functionality to name service, meaning that there might be several variations to categorize the same service. For example, a service such as HaaS or "Healthcare as a Service" can also be called "Medicare on Demand as a Service." This variation might be legal as per the XaaS framework. However, multiple variations of the same service cause ambiguity. Essentially, XaaS is confusing

and unintentionally unclear as the "aaS" suffix could potentially refer to anything other than cloud services too (Fig. 1).

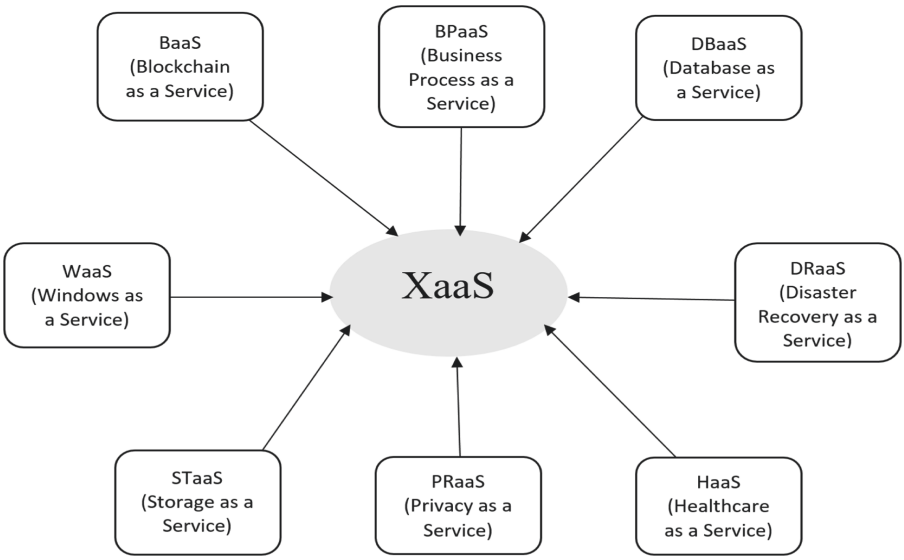

Fig. 1. Examples of anything as a Service (XaaS) offerings

4.2 Johan Den Haan Framework for Categorizing Cloud Services

In the article written in October 2013, Johan argues that "saying that I work for a PaaS company is as broad as saying that I work for a software company." Furthermore, he says, "The lines between IaaS, PaaS, and SaaS are blurring, and subcategories are emerging within these categories that describe a different range of approaches" [1].

Johan Den Haan, in his article, suggested a seven-layer model as shown in Table 1; starting with the bare-metal hardware (data centers) and defining each additional layer as an abstraction over the hardware layer until he gets the final layer where applications are used to execute the cloud service model.

The main advantage of this framework is the clear segregation of cloud services without overlapping between categories. For example, Serverless computing can now be considered as a separate service model as opposed to it previously being categorized simply as PaaS. However, CaaS is not included as a separate layer in this framework; instead, it is included in layer 2 (Foundational PaaS).

Foundational PaaS bears slight differences when compared with CaaS. Foundational PaaS is the layer on top of the infrastructure layer that gives the developer pre-packaged tools and a ready-to-use hosting environment [7]. In contrast, CaaS uses a container engine that virtualizes on the operating system deployed on the infrastructure; in CaaS, the CSP provides tools to manage these containers instead of a pre-packaged development environment, in case of CaaS how the organizations set up these containers to

Table 1. Johan Den Haan framework to categorize cloud services [1] further modified by Christine Miyachi [5]

S.No.	Layer name	Potential users
Layer 6	SaaS	End users
Layer 5	App services	Citizen developers
Layer 4	Model-driven PaaS	Rapid developers
Layer 3.5	Serverless computing	Speed developers
Layer 3	PaaS	Developers/Coders
Layer 2	Foundational PaaS	DevOps
Layer 1	Software-defined data centre	Infrastructure engineers
Layer 0	Hardware	–

deploy their application is not the concern of the CSP. Due to the slight architectural differences between CaaS and Foundational PaaS, CaaS generally offers more control than Foundational PaaS.

Essentially Foundational PaaS is a bigger umbrella under which many types of similar services come; CaaS could be one. According to Gartner, by 2022, more than 75% of applications will be running on containers instead of less than 30% today [8]. This stat by Gartner shows that the technology is not yet peaked; if the technology keeps on growing, the approach of delivering CaaS services may change and consequently bear less and less resemblance with Foundational PaaS, which would again prompt for the model to be updated. Thus, we believe that CaaS should receive its own category.

5 Proposed Architecture

5.1 Preview

The proposed service model classification architecture comprises eight primary layers starting from layer 1 to layer 8. Compute, network, and storage or CNS are the building blocks of the model.

Layer 0 denotes the hardware layer onto which further abstractions are constructed to create other layers (layer 1 till layer 8). Layer 0 represents on-site deployment or data centers set up by the CSPs and is included to provide a clear baseline for the above layers. Layer 0 is not a cloud service model; rather, it represents the fundamental building blocks of further layers. Layer 1 till Layer 8 represent delivery methodologies.

Abstraction over a layer signifies an additional software deployment on the said layer. This software deployment essentially gives the product situated in the said layer the features of a cloud service model. Abstraction encapsulates the layer within a custom software, enabling the CSP to modify the feature set of the layer on which it is deployed, consequently changing how the service is offered on that layer. Hence, creating a new cloud delivery model with a different feature set and control offered to the client. For example, a container engine – a software – if deployed on top of Layer 1 changes the

fundamental characteristics of the Layer 1 service model. Thereby creating a new layer comprising of CaaS or container services (layer 2).

Furthermore, we have divided the layers into three categories on the lines of the NIST model for easy understanding and to enable seamless between the proposed architecture and the NIST model. These categories are:

1. Basic Infrastructure: Includes layers that deal primarily with offerings that give some form of hardware with or without the abstraction.
2. Platform: Includes layers that offer a platform for development and application to the clients.
3. Application: The application category contains layers that offer products pre-packaged as applications.

5.2 Design Methodology and Trends

The model has been designed with CNS (Compute, Network, and Storage) as the fundamental components. Any level of abstraction or encapsulation onto CNS components of a layer creates a new layer on top of the previous layer.

We have constructed the architecture such that the level of control over CNS given to the client decreases as the layer number increases. So, layer 1 offers the maximum control over CNS, and layer 8 offers the least.

Furthermore, as the layer number increases, the control of CSP over the product increases. Essentially, the CSP manages most part of the products placed on layer 8, including the underlying infrastructure.

5.3 Architecture Explanation

The architecture is represented as shown in Fig. 2; here, the layers are marked along the vertical direction, and the components (CNS) are shown along the horizontal direction. The intersection of layers with the fundamental components (CNS) clarifies the product's functionality and is represented with a symbol. Cloud services or products offered by the CSPs are placed at a suitable layer with coordinates of suitable intersection determined by the product purpose or functionality.

For example, According to the proposed architecture matrix, Amazon EC2 is a service suitable for layer 1 at coordinates [1, C].

The layers, as shown in Fig. 2, are as follows:

– Layer 0: Physical Hardware
– Layer 1: Virtualized Hardware
– Layer 2: CaaS
– Layer 3: Foundational PaaS
– Layer 4: PaaS
– Layer 5: Serverless Computing
– Layer 6: Built-up PaaS
– Layer 7: Application Services
– Layer 8: SaaS

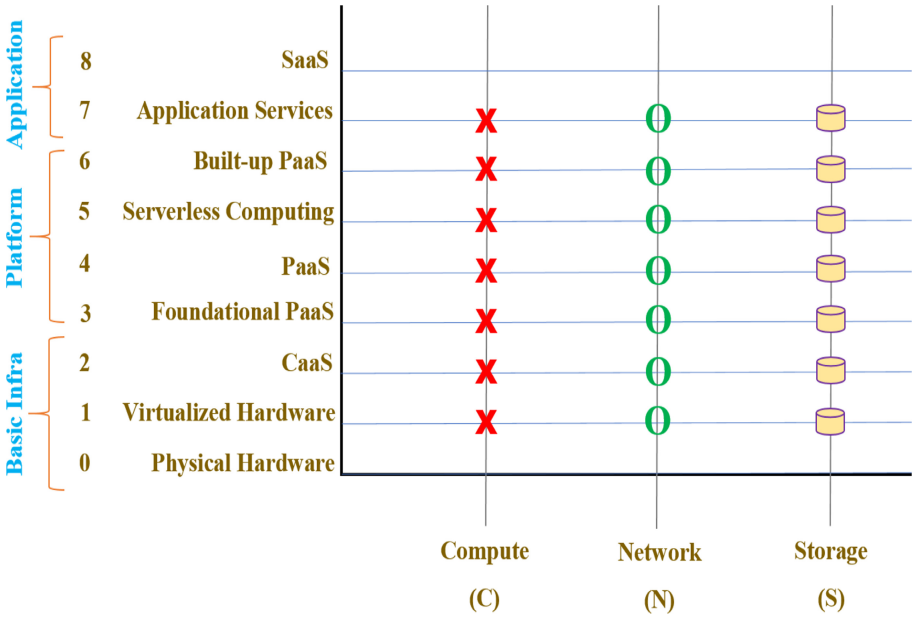

Fig. 2. Proposed classification architecture for cloud service models

5.4 Layers Explanation and Criteria

Layer 0 – Physical Hardware
This layer is the physical layer and is placed for setting a baseline on which abstractions could be placed to create cloud offerings. Independently this layer can be considered as an on-site setup which is opposite to cloud computing.

Abstractions over the CNS components of this layer create different feature sets, thereby creating additional layers.

Since this layer is entirely set up on the organization's premises, it offers maximum control over the CNS components. Physical data centers can be considered as part of this layer.

Layer 1 – Virtualized Hardware
This layer comprises virtual machines, networks, or virtual storage created using abstracting over the physical hardware using something like hypervisors by the CSP.

The products at this layer offer virtualized hardware components that can be utilized the same as regular hardware. Hardware virtualization is more efficient and is also less costly than actual physical hardware deployment. Also, hardware virtualization offers almost a similar level of control as physical hardware, but at the same time, it is more reliable than using physical systems.

Organizations use the products at this layer, but mostly hardware virtualization is the primary way used by the CSP to deliver customized platforms and software. Further

abstracting this layer creates customized cloud solutions that come under the platform and application category.

The primary criteria for a product to be placed in this layer are that it offers physical hardware components via placing an additional virtualization layer upon it.

Layer 2 – CaaS or Container as a Service

This layer comprises container management products and products that virtualize upon hardware using an additional layer of container engine over the host operating system that can be possibly placed on virtualized hardware.

This layer is an abstraction over layer 1 (Virtualized Hardware Layer). This layer provides the consumers with container management systems that can create, manage, and organize containers. The clients are given complete control of the containers that they create. Moreover, the user can customize the container environment to suit their application development; this customization includes installing required OS, libraries, tools, and additional dependencies.

This layer offers control over the OS as opposed to the virtualized hardware layer, which focuses more on the infrastructure layer.

We have added a separate layer for CaaS services instead of including it in the foundational PaaS as the level of control offered here is more when compared with foundational PaaS (Layer 3).

Layer 3 – Foundational PaaS

This layer comprises products that offer the client a platform for their development tasks. The provided platform contains pre-installed libraries and dependencies according to the specifications provided by the client. However, the client is given the freedom to deploy additional libraries as the need arises.

The compute vertical on this layer focuses on services like DevOps, where a custom platform is provided to ensure compliance with DevOps. However, the user can modify the platform and include additional dependencies to cater to their organization's specific requirements. The network vertical deals with solutions that cater to communications inside and outside the platform, ensuring intelligent load balancing, scheduling, and queuing the incoming application access requests. Finally, the storage vertical deals with something like object storage.

The level of control offered here is the most among the platform category, with features like the inclusion of custom dependencies absent in further layers of the platform category.

Layer 4 –PaaS

The PaaS layer includes products that provide the consumer with a platform as per the specifications required by the client to facilitate application development and deployment. However, offerings at this layer offer less control than platform offerings at layer 4. Essentially, if a client purchases a PaaS product from the CSP, they will get a platform that will be pre-equipped to perform the required task and managed and maintained by the CSP. Products in this layer do not give the user control to deploy libraries other than those provided by the CSP. Essentially, if a client purchases a PaaS product for SQL Database they will get a platform with pre-installed libraries to facilitate just the

SQL development ultimately; the user will not get any feature to install libraries for MongoDB in the same product, for that they will need to purchase a different platform. However, the users can manage and control the code execution over the servers.

This layer offers specific language compilers such as java or python deployed over a container in the compute vertical. The communicate vertical encompasses products that facilitate communication among applications deployed over the cloud and applications deployed on-site. The storage vertical offers specialized storage solutions provided as a platform and uses a popular database technology like SQL or DynamoDB to store the inserted data.

Products at this layer should provide a specialized platform for a specific task – such as code execution and deployment, communication between applications, and storage solutions – with the condition that the provided platform will perform the said task with the pre-installed libraries and no libraries could be added that are not in the bracket of requirements to perform the said task.

Layer 5 – Serverless Computing

This layer comprises products that offer a platform for carrying out application development and deployment. However, like other platform category layers, products at this layer do not let the users control their execution servers. The CSP manages everything after uploading the code and pressing execute. Furthermore, products at this layer offer the clients to only pay for the time they have engaged the servers and not for the whole platform as in products at other platform category layers.

Products at this layer come as a platform equipped with all necessary libraries for the task chosen by the client. The client controls the application code and application deployment, but the servers for code execution are controlled by the CSP and not by the client.

Layer 6 – Built-up PaaS

Products at this layer can be best described as drag and drop PaaS. The product offered at this layer specifically targets rapid developers. Rapid developers are developers who want to develop an application but do not have an in-depth knowledge of various technologies required to do so. So, products offered at this layer provide such developers with a platform containing ready-made parts which they can assemble as per their application requirements. They do not need to make changes to these ready-made components. However, products at this layer provide the functionality to customize a read-made offering according to their use-case.

For example, a built-up PaaS product to develop a website rapidly would have ready-made HTML form plug-ins, design plug-ins to customize the look and feel of the webpage, and authentication systems already available to be used off the shelf as per the developer's will. The developer can utilize these off-the-shelf components directly or make adjustments to them to suit the application better.

The product at this layer offer platform control to the client to some extent. Moreover, the freedom to change the default offerings to suit a specific application better is also provided. However, the developer cannot control the code execution and the basic infrastructure settings like the operating system and language libraries; the user has to

work with the libraries and languages provided by the platform and cannot install external libraries. Essentially, this layer is the result of amalgamating the features of platform category layers with application category layers which in turn means that products at this layer will have more control than application category layers but the least level of control among platform category layers.

Layer 7 – Application Services
The products at this layer offer developers an application that provides off-the-shelf services that can be repackaged to become a part of a customized application according to a client-specific use case.

At first, the products at this layer may seem the same as those on the previous layer (Layer- 6: Built-up PaaS), but the crucial difference lies in controlling and customizing the ready-to-use components. In this layer, the products offer a choice to create a final application by repackaging numerous ready-made components. In contrast, the products on the Built-up-PaaS layer offer the developers control to choose from the components and customize those components according to a specific use-case by uploading custom code in the language supported by the product. Control to customize every component and then repackage those customized components as an application is absent in the products offered at this layer. Product at this layer offers control only to repackage the already available components as they are provided inside the product offered by the CSP.

However, the depth of control while repackaging can vary, which means that users can select a single feature inside a particular component to include in their application and ignore the rest. However, what the user cannot do is add a feature of their own inside the component. For example, in Application Services for privacy, the client can select if they want the feature to "automatically create privacy policy documents" after they answer a "questionnaire" from the "Generate Documentation Module." However, the client cannot modify the "privacy policy generator" and include additional code that outputs an additional line as per the data in the questionnaire.

Layer 8 – SaaS
The last layer contains products that are ready-to-use applications in which the user can make customizations that only affect themselves or their organizations, such as company geographical location changes.

Among all the layers, SaaS offers the least amount of control to the client, and almost all the management is taken care of by the CSP.

Products classified as SaaS are essentially pre-packaged software delivered through the cloud and are used by the organizations for the functionality it provides rather than creating or hosting their application.

In SaaS services, each of the CNS components' involvement cannot be determined separately as the level of abstraction is relatively high, and the underlying configurations are invisible to the user. Thus, there are no intersections shown in Fig. 2 on layer 8.

The primary criteria for a product to be classified as SaaS are that it should be a pre-packed software delivered over the cloud that serves a particular purpose and offers no environmental or infrastructure control, meaning almost all of the components are controlled by the CSP rather than the client.

6 Plotting Cloud Products in the Proposed Architecture

In this section, we have mapped the prevalent cloud product of popular CSPs. According to our understanding, the services are mapped as follows (Table 2):

Table 2. Mapping popular cloud products on the proposed architecture

CSP	Product	Co-ordinates
Amazon	Amazon EC2 (Elastic Compute Cloud)	1, (C)
Amazon	Amazon RDS (Relational Database Services)	4, (S)
Amazon	Amazon S3 (Simple Storage Service)	3, (S)
Amazon	Amazon Lambda	5, (C)
Amazon	Amazon CloudFront	8
Amazon	Amazon Glacier	
Amazon	Amazon SNS (Simple Notification Service)	8
Amazon	Amazon Elastic load balancing	3, (N)
Amazon	Amazon Kinesis	8
Amazon	Amazon IAM (Identity and Access Management)	8
Amazon	Amazon Elastic Beanstalk	4, (C)
Amazon	Amazon Redshift	4, (S)
Amazon	Amazon Cloudwatch	8
Microsoft	Azure DevOps	3, (C)
Microsoft	Virtual Machines	1, (C)
Microsoft	Azure Active Directory	8
Microsoft	Azure Logic Apps	6, (C)
Microsoft	Azure Site Recovery	7, (C,N,S)
Microsoft	Azure Bots	4, (C)
Microsoft	Office 365	8
Docker	Container	2, (C)
Docker	Docker Swarm	3, (N)
Salesforce	Salesforce.com	7, (C,N,S)
Google	Google Compute Engine	1, (C)
Google	Google Container Engine	2, (C)
Google	Google Cloud Functions	5, (C)

7 Future Research Directions

The proposed model would work for the foreseeable innovations in cloud service methodology. Nevertheless, after a while, the number of layers might increase to a number beyond manageable. Another dimension could be added to the framework that could take care of minor fluctuations in the delivery approach.

In the subsequent papers, we will look at exact parameters for the third dimension. Moreover, we would look at how another dimension could change the proposed framework and the number of layers.

Finally, we would like to suggest the following research directions that can be pursued to improve the proposed architecture:

- Addition of another dimension that can take care of minor fluctuations in the delivery approach resulting in a cleaner model.
- We have considered CNS (Compute, Network, and Storage) as the means of cloud service delivery; however, there might be more ways of delivering cloud services that remain to be explored.
- The criteria to place a product on a service layer are by no means concrete or complete, and there might be other parameters that should be included to make the architecture more relevant. These additional parameters remain to be explored.

8 Conclusion

NIST service models were defined a decade ago, and since then, cloud delivery systems have evolved a lot, and many novel service methodologies have emerged, such as Serverless Computing and CaaS are widening the cracks in the NIST-defined service models.

In this paper, we have tried to address the issue of NIST service models categorization getting outdated by proposing a layered architecture. We proposed a nine-layer framework with each layer defining a separate service model. We have also tried to shed light on the topic, why we should update the NIST model. We argued that technologies like Serverless and CaaS use a fundamentally different approach due to which a different level of control is offered by them, which makes it difficult to place in the buckets of IaaS, PaaS, and SaaS, which are again defined by NIST based on control given to the client. Finally, we have placed cloud services currently prevalent in the industry onto our model.

NIST models might be losing industry significance, but they still do a great job introducing the cloud delivery methodology to a novice in this field. Thus, we believe that the NIST models should introduce a novice to the field, but a more comprehensive model should be utilized for industry usage.

References

1. Haan, J.D.: The cloud landscape Described, categorized, and compared. The cloud landscape described, categorized, and compared, October 2013. http://www.theenterprisearchitect.eu/blog/2013/10/12/the-cloud-landscape-described-categorized-and-compared/. Accessed 10 Aug 2021

2. Mell, P., Grance, T.: The NIST definition of cloud computing. NIST (2011). https://doi.org/10.6028/nist.sp.800-145
3. Red Hat. What are cloud service providers? https://www.redhat.com/en/topics/cloud-computing/what-are-cloud-providers. Accessed 10 Aug 2021
4. Erl, T., Mahmood, Z., Puttini, R.: Cloud Computing: Concepts, Technology, & Architecture. Prentice Hall, Hoboken (2014)
5. Miyachi, C.: What is "Cloud"? It is time to update the NIST definition? IEEE Cloud Comput. **5**(3), 6–11 (2018)
6. Simmon, E.: Evaluation of cloud computing services based on NIST SP 800–145. NIST Spec. Publ. **500**, 322 (2018)
7. Mendix: What is platform as a service (paas)? Mendix, January 2013. https://www.mendix.com/blog/making-sense-of-paas/. Accessed 10 Aug 2021
8. Gartner: Gartner forecasts strong revenue growth for Global container management software and services through 2024. Gartner, June 2020. https://www.gartner.com/en/newsroom/press-releases/2020-06-25-gartner-forecasts-strong-revenue-growth-for-global-co. Accessed 10 Aug 2021
9. Pahl, C.: Containerization and the PaaS cloud. IEEE Cloud Comput. **2**(3), 24–31 (2015)
10. Castro, P., Ishakian, V., Muthusamy, V., Slominski, A.: The rise of serverless computing. Commun. ACM **62**(12), 44–54 (2019)
11. Sosinsky, B.: Cloud Computing Bible. Wiley Publication, Hoboken (2011)
12. Buyya, R., Broberg, J., Andrzej, G.: Cloud computing: Principles and paradigms. Wiley, Hoboken (2011)
13. Van Eyk, E., Toader, L., Talluri, S., Versluis, L., Uţă, A., Iosup, A.: Serverless is more: from PaaS to present cloud computing. IEEE Internet Comput. **22**(5), 8–17 (2018)
14. McGrath, G., Brenner, P.R.: Serverless computing: design, implementation, and performance. In: 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), pp. 405–410. IEEE, June 2017
15. Bahga, A., Madisetti, V.: Cloud Computing: A Hands-on Approach. Universities Press, Lanham (2014)
16. Gibson, J., Rondeau, R., Eveleigh, D., Tan, Q.: Benefits and challenges of three cloud computing service models. In 2012 Fourth International Conference on Computational Aspects of Social Networks (CASoN), pp. 198–205. IEEE, November 2012
17. Sowmya, S.K., Deepika, P., Naren, J.: Layers of cloud–IaaS, PaaS and SaaS: a survey. Int. J. Comput. Sci. Inf. Technol. **5**(3), 4477–4480 (2014)
18. Mohammed, C.M., Zebaree, S.R.: Sufficient comparison among cloud computing services: IaaS, PaaS, and SaaS: a review. Int. J. Sci. Bus. **5**(2), 17–30 (2021)