A Novel Architecture for AI-as-a-Service Applications using Docker Containers and Kubernetes

ABHI AKSHAT[1], KRITIKA TRIPATHI[2], DEVANSHI MALIK[3], SANDEEP KUMAR[4]
[1, 2, 3, 4] *Department of Computer Science and Engineering, Sharda University, Greater Noida, India*

Abstract*— AI services can broadly be separated into three main categories: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Businesses are using AI microservices for automation, customer engagement, speech recognition, data analysis, and other tasks to improve operations, customer experiences, and innovation. However, the platforms for adopting these services differ for different purposes and business domains. This paper proposes a simple architecture for a container-based AI as a Service (AIaaS) Application that will provide all three Services - SaaS, PaaS, and IaaS to the business on a single platform, thereby, making it uncomplicated for them to adopt and manage these services. This paper also discusses how using the containers-based architecture can help in creating such a platform with the help of Kubernetes and Docker

Indexed Terms*— AIaaS, SaaS, PaaS, IaaS, microservices, containers, Kubernetes, Docker, Containerisation.

I. INTRODUCTION

With the speed at which technology is developing these days, artificial intelligence (AI) has developed from a theoretical idea to a practical technology with the potential to revolutionize many different fields. AI's capabilities have redefined what is achievable from healthcare to finance, manufacturing to entertainment.

In this field, researchers require thorough knowledge of every concept and this is where the "AI as a Service" comes as a game changer. AIaaS provides researchers with on-demand access to AI tools, models, and infrastructure, democratizing AI and enabling researchers to focus on their core objectives. The term "AI as a Service" (AIaaS) refers to a paradigm shift that gives researchers access to advanced AI tools and resources without the requirement for specialized knowledge or large-scale computing equipment. Traditionally, using AI in the research field required significant expenditures to hire people with specialized training, software, and hardware. AI as a service has significantly altered this, which offers a cloud-based framework that makes AI tools, models, and resources available as needed. This leads researchers from diverse backgrounds and domains to fully utilize their potential.

This paper aims to deliver a platform based on AI as a service broadly classified into four categories.

Software as a Service: The software-as-a-service (SaaS) framework allows service professionals to digitally engage and collaborate with clients using web-based software accessible on any internet-connected device [1]**.** In the software as a service (SaaS) model, the software is accessed by users via web browsers, mobile applications, and APIs, and is managed and updated by the service provider on remote servers. Users can only utilise the programme by investing in pricey infrastructure because it is housed on the servers of the service provider. The cost and complexity of implementing and operating SaaS applications have been significantly lowered because of developments in cloud computing, which have made it possible to provide software applications over the Internet utilizing distant servers.

Platform as a Service: A cloud service paradigm called Platform as a Service (PaaS) offers a platform for the creation and implementation of cloud applications without worrying about hardware or system software [2]**.** PaaS includes middleware, development tools, database management systems, business intelligence (BI) services, and more in addition to infrastructure (servers, storage, and networking). You may save money and simplify your life by using PaaS instead of buying and maintaining middleware, application infrastructure, container orchestrators like Kubernetes, development tools, and software licenses.

Infrastructure as a Service: An essential part of cloud computing, infrastructure as a service (IaaS) offers networking, processing, and storage capabilities to consumers [3]. You may save money on hardware, get real-time business insights, and lessen the upkeep of on-premises data centres by migrating your organization's infrastructure to an IaaS provider. You only pay for a particular resource at the times when you really utilise it. Each resource is offered as a separate service component.

Container as a Service: CaaS, or Container as a Service is a form of implementing micro services architecture using containers. Containers are isolated environments that replace the need for Virtual Machines. It is built on a host OS, but every container can also have its own OS.

Therefore, this paper aims to build a system that can integrate all three kinds of services together in one platform so that it is easier for businesses to adopt and manage the services without any technical and managerial problems. This paper discusses an architecture for this type of platform by making a basic platform for three kinds of services, namely- Cold Email Writer, SEO Optimisation, and Image Generation.

Open AI API and Llama model are used in order to make it possible to implement the backend. Artificial intelligence (AI) technologies are the primary focus of OpenAI, a technology startup**.** It seeks to guarantee that AI is advantageous to all people. AI research is carried out by OpenAI is used in several domains, including natural language processing, computer vision, and reinforcement learning. It has produced several powerful AI models, such as GPT-3, which is renowned for its capacity to produce prose that resembles that of a human. Chatbots, content creation, and language translation are just a few of the uses for OpenAI's technology. Enhancing text and picture production as well as chatbot functionality in mobile apps has shown potential when OpenAI's AI models are combined with Flutter, an application development platform [5].

Docker and Kubernetes for containerization offer a practical and efficient approach to deploying and managing microservices**.** Therefore, this paper uses the implementation of Docker and Kubernetes for the deployment of microservices [4].

By using these concepts and technologies this paper implements a Container based architecture for AI as a Service (AIaaS) Application. Three types of services are tackled - Software, Platform, and Infrastructure.

Container as a Service: CaaS, or Container as a Service is a form of implementing micro services architecture using containers. Containers are isolated environments that replace the need for Virtual Machines. It is built on a host OS, but every container can also have its own OS.

By using these concepts and technologies this paper implements a Container based architecture for AI as a Service (AIaaS) Application. Three types of services are tackled - Software, Platform, and Infrastructure.

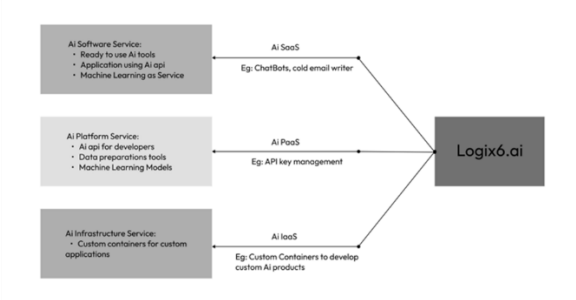

Figure 1: Basic Overview of the all-in-One-Platform for AIaaS Services

In Figure 1 an example organization (Logix6.ai) is mentioned. This organization has three components – AI SaaS, PaaS, and IaaS services. In the SaaS component, the organization is ready to use AI tools. The PaaS component provides APIs and SDKs for a developer to build their own applications on top of the platform. In the IaaS component, the organizations build customs solutions for a client using containers. For this organization, this architecture would be fruitful as it would provide scalability and flexibility.

II. LITERATURE SURVEY

A model and template-based method that creates test cases and scripts automatically was introduced by Zhou et al. [6] in 2014; this technique lowers the cost of performance testing and finds possible problems with performance.

Pahl et al. [7] conducted a thorough analysis of the application of clusters and containers in edge cloud architectures in 2015. It assesses how well edge-cloud computing infrastructures based on containers operate and identifies areas where they may be improved to better satisfy the demands of industrial applications that require speed.

It offers an orchestration and containerization approach for robotic software applications on heterogeneous systems, based on Docker and Kubernetes.

Cluster as a Service (ClaaS), a container-based method that facilitates cluster sharing and offers multi-user support, was suggested by Wei et al. [8] in 2016. It enables the majority of programmes to function without change by virtualizing the cluster environment

for application frameworks. To prove its viability, ClaaS uses lightweight containers and incorporates an actual system named Docklet. It also emphasises application definition in a way that makes deployment easier. The issues of executing diverse programmes and efficiently sharing cluster resources are addressed by ClaaS, which provides a container-based solution to cluster sharing with support for many users.

Container-as-a-service (CaaS) architecture is a service paradigm that allows cloud computing to deliver scalable systems while avoiding the drawbacks of conventional hypervisors. Tao et al. [9] published about it in 2018. It may be used to business process management, enabling organisations to more effectively and performance-focusedly outsource their procedures to the cloud. In order to optimise business process deployment on cloud containers, a linear programme is suggested, and its efficacy in comparison to conventional deployment tactics is demonstrated.

Zhang et al.'s study [10] compared the performance of virtual machines and containers in a large data context, and it was completed that same year. According to the study, virtual machines—more precisely, KVMs and Xen virtual machines—were slower than Docker containers. The study also looked at scaling containers using the Kubernetes framework, which enhanced Docker container performance over alternative frameworks.

Using the container technology that was demonstrated, Yao et al. [11] suggested a distributed architecture in which each service container is matched one at a time with a target service. A distributed architecture built on the foundation of container technology makes up the cloud platform. The innovation provides a containerized cloud platform with container management functionality. Without the need to maintain intricate Kubernetes yaml configuration files, the cloud platform furthermore offers container application administration services.

The flexibility and reusability of the container-as-a-service (CaaS) framework for service migration, which forms the infrastructures, applications, and services for various vendors, is demonstrated by Wang et al. [12], demonstrating the potential benefits of this approach for business workflow orchestration. Resources are virtualized and segregated through the use of containers, making it easier to create and launch several apps independently across various vendor systems. Computational activities and worker processes are contained inside web services inside a container to take use of service-based workflow technologies and create efficient and timely workflows for a smart city environment.

Platform-based containers are a kind of container intended for transportation and storage, according to Kong et al. [13]. They may be stacked for effective storage and transportation, and they are quick and easy to use. Usually, these containers are made up of corner pieces, side columns, and a bottom frame. Two switchable workstations are made possible by the side columns' rotatably attached bottom structure. This design reduces expenses and enhances the ease of carrying and storing empty containers.

A microservice-based framework was developed by Kousiouris et al. [14] in 2019; it is capable of carrying out activities like supply chain monitoring and items location analysis by consuming and annotating data streams from online systems. Three blocks make up the system: Red, Node-Red, and AI layer. The AI layer was a key component of the study. TensorFlow is the foundation of the AffectUs AI framework, which uses Python 3. Dockerization of services is another. The service's Dockerization made it possible for all of the components to be quickly and seamlessly deployed, as well as for their automatic configuration, connection, and discovery.

According to Chen et al. [15], quick deployment is possible with a big data platform architecture system built on a container cluster. The big data platform's cluster management and automated deployment are made possible by container virtualization technologies and a descriptive file that serves as a service running unit. Using a multi-tenant architecture, the system offers large data applications safe isolation and sharing techniques in a multi-tenant setting.

Singularity is a containerization strategy intended for ARC on shared HPC clusters, according to Newlin et al. [16]. Although there are few studies assessing the performance trade-offs of deploying AI workloads

using a container-based approach, it is a feasible scientific container solution. The paper's authors are not aware of any previous research that has been done on this subject. By utilising newly created community-developed benchmarks, they want to investigate the trade-offs in performance that arise while executing AI workloads in a containerized Singularity environment.

A machine learning (ML) pipeline and use cases in several industries are highlighted in Parsaeefard et al.'s [17] architectural scheme for providing AIaaS on SDIs. To address the model creation and operational stages of AI applications, the model incorporates a novel training plane and an AIaaS plane. A placement design that takes resource utilisation into account and reduces the number of instantiated virtual machines (VMs) and active physical machines (PMs) in a cloud environment was presented by Hussein et al. [18] in the same year.

The difficulties of delivering AI capability "as a Service" (AIaaS) in corporate settings were covered by Casati et al. [19] along with some suggested solutions. The answers are based on the researchers' experience creating, implementing, and testing AI services with several ServiceNow clients. ServiceNow is an Application Platform as a Service that streamlines complicated tasks into a single cloud platform and facilitates digital processes. A framework and architecture are chosen that takes care of every problem, including automation, security, performance, efficacy, simplicity of use, and economical resource usage.

According to Wang et al. [20], containerization technology encapsulates HPC programmes and their dependencies, providing a performance-efficient way to install them. The findings demonstrate that while default Singularity delivers performance close to bare metal, fine-grain multi-container deployments are not supported.

Using decentralised policies based on reinforcement learning, Rossi et al. [21] investigated the run-time adaptation of containerized applications with Quality-of-Service requirements, deployed over heterogeneous computing and networking resources, in the same year. The applications were deployed with elasticity and container migrations in mind.

Debauche et al. [22] used IoT and microservice architecture in 2020 to integrate AI into the supply chain. The Edge computing post-cloud techniques enable jitter and latency to be improved. These infrastructures manage containerization and orchestration processes to enable automated and rapid deployment and migration of services such as ontologies, reasoning mechanisms, and customised AI algorithms. They provide a novel architecture in this study for implementing AI-driven microservice models at the edge.

A framework for managing containers in a CaaS environment—which may be used to implement AI applications—was developed by Boukadi et al. [23]. By offering a framework to fill CDO, deploy apps on a container orchestration system, and improve interoperability across cloud providers, the paper seeks to ease container administration for both users and cloud providers.

Containers, as introduced by Subil et al. [24], have become widely used in cloud systems because of their lightweight design, application mobility, and deployment flexibility. Additionally, they have been advantageous for applications involving data analysis and machine learning in high-performance computing (HPC) settings. But in HPC, containers have issues with I/O and performance, particularly with regard to throughput. Numerous container technologies have been studied for HPC settings, including Docker, Podman, Singularity, and Charliecloud. The effects of these solutions on HPC I/O throughput and how they work with Lustre and other parallel file systems have been taken into consideration. Scientific computing can benefit from the usage of container-based virtualization since it has demonstrated near-native speed and reduced overhead in HPC applications.

A cost-effective and lightweight edge intelligence architecture based on containerization technology was presented by Mabrook et al. [25]. This architecture manages, distributes, and launches edge/cloud apps across clusters of low-power devices, such Raspberry Pis, using Docker technology. When compared to typical cloud computing, this method has advantages in terms of processing time, energy efficiency, and bandwidth savings. The efficacy of the suggested

architectural approach has been confirmed by experimental outcomes using Raspberry Pi clusters.

A path for integrating AI into corporate processes is presented by Reim et al. [26], who stress the importance of organisational capacities, internal competency development, and AI understanding.

The MODAK tool was presented by Mujkanovic et al. [27]. It maps ideal parameters to target infrastructure and creates optimised containers to optimise application deployment in software-defined infrastructures. Custom-built, optimised containers perform better than official DockerHub images, according to the evaluation. The complexity of neural networks and the target hardware affect graph compiler performance.

Janbhi et al. [28] put out a paradigm for DAIaaS provisioning in IoE and 6G systems the same year. The framework splits the computations involved in training and inference into smaller, concurrent tasks that are appropriate for the cloud, fog, and edge layers of the network. In addition to evaluating the framework using performance measures and presenting case examples, the article offers suggestions for improving performance.

According to Neuhüttler et al. [29], the use of AI has led to significant changes in the components of business models, including improved value propositions, data utilisation, ecosystem collaboration, and modified revenue models.

The idea that AIaaS providers should keep an eye on client usage to guarantee proper use of AI services was first presented by Javadi et al. [30]. Generally speaking, AIaaS services offer general capability that is "at a click" away. Given the growing interest in AIaaS, it is critical to address the associated concerns. The study focuses on the problems related to accountability, regulation, and ethics in AI.

An AI service open middle station, or system that offers a uniform and standardised platform for accessing AI services, was proposed by Jiayi et al. [31]. It consists of many modules, including a project competition system, an automatic training optimisation system, a data standardisation processing system, and a standardisation system for service APIs. The modules that make up this system are the access layer, business layer, stable layer, and service layer. This technology increases the system's flexibility and deployment efficiency while enabling the quick and continuous deployment of service applications.

A cloud architecture based on containers with improved autonomy and scalability was proposed by Kim et al. [32]. In order to provide the container-based cloud service, user terminals are connected to the network via a web socket, authentication keys are generated and stored in the in-memory cluster, users are authenticated using the keys, cloud services are provided via the service module, event data is stored in the in-memory cluster, it is periodically stored in a database, and the service module is scaled. Performance and scalability are guaranteed by the system and technique, even when many people are logged in at once.

According to Craig et al. [33], container-based analysis environments enable researchers to access and analyse large-scale datasets wherever they are stored, hence facilitating low-barrier access to research data. Using containers also removes the need to physically move big datasets and addresses privacy issues by allowing research apps to be run and evaluated on the data owner's infrastructure. By giving researchers access to data that would otherwise be unavailable and enabling the reproducibility and reuse of scientific findings, these container-based techniques have the potential to completely transform the analysis of research data.

According to Kim et al. [34], a web socket is used to link user terminals and an access server to the network in order to deliver a cloud service based on containers. An authorised user receives the cloud service via a container-structured service module, and the in-memory cluster stores event data created during service provisioning. Periodically, a database is used to store the event data. Databases, a server module, and an in-memory cluster are all part of the system. A container-structured data access control module allows the server module to access the databases and leverages the event data to deliver the cloud service.

An AI-as-a-service stack implementation was presented by Lins et al. [35] in 2021, which enables the construction of a system with user-friendly features and smooth sailing. The stack featured building blocks and ready-to-use AI applications in the form of AI software services (which relate to the traditional SaaS cloud layer). Tools for helping developers build code to bring forth AI capabilities are known as AI developer services (related to the standard PaaS cloud layer). However, there was no implementation offered for the same.

A container-based edge computing system for smart healthcare applications was presented by Tuan et al. [36] in 2021. The solution centralises patient data in a safe, scalable, and fault-tolerant database by using a strong cloud computing architecture. The suggested system also uses a lightweight container orchestration framework to create AI applications for high availability, scalability, deployment automation, and efficient administration.

Using openSUSE Kubic, Pratama et al. [37] concentrated on the technical implementation and deployment of CaaS, the fusion of containers with cloud computing, emphasising its dependability and user-friendliness for software development. Software developers may increase the speed and scalability of software development with the use of Caas.

The practises and difficulties involved in creating deep learning (DL) and machine learning (ML) models as components of large software-intensive embedded systems are examined in a multi-case study presented by John et al. [38] in 2022. In order to maximise model design and business integration, they provide a conceptual framework complete with activities, iterations, and triggers.

In order to ensure that AI frameworks operate in an isolated container environment with user credentials, Kumar et al. [39] describe how Docker containers may be utilised in safe shared multi-GPU systems without requiring any changes or raising security issues.

Pre-processing, post-processing, and the model itself are all included in the AI service architecture that Shah et al. [40] suggested for edge devices that same year. The suggested design is appropriate for microservice architecture as it specifies interfaces for the AI service's configuration and access. The study also discusses an explicit content-blocking device that checks if material is explicit based on an explicit URL list using explicitness determination model files.

According to a research by McMillan et al. [41], artificial intelligence (AI) can offer crucial cognitive insights for handling and executing complex and unpredictable systems. Tasks like time series prediction, consumer segmentation, and energy management systems design have been tackled with the use of artificial intelligence techniques like deep learning and unsupervised learning. Improved service outcomes, flexibility in response to changing circumstances, and increased productivity across a range of industries, including industry and healthcare, are some of the possible advantages of incorporating AI into infrastructure systems.

Software Testing: An Analysis and Comparison of Artificial Intelligence Methods [42] because manual testing is not a practical solution. The difficulties of reducing software lifecycles and meeting deadlines have been addressed by AI-powered automated testing, which has made it possible to execute complete test suites on every update in a timely manner. The use of artificial intelligence (AI) principles, such as neural networks, machine learning, deep learning, and expert systems, may make software testing less difficult and time-consuming.

III. METHODOLOGY

The entire architecture is based on two levels - SaaS level and PaaS level. This model is an architecture for any company that has the following two features. The entire architecture is container-based. Each application or service is hosted in its separate container. Each docker container has an infrastructure layer on top of which is the host operating system. The Docker engine runs on top of the host operating system. It provides a way to run several isolated containers. As each container does not need any shared data among them. Figure 1 shows the architecture of a container.

Here each model is a separate container with its own OS, libraries, and data. Using this architecture makes

the system scalable if traffic increases, we can simply add more containers and use the docker image file of the previous container to replicate it. The main architecture is divided into two parts - SaaS and PaaS. Firstly, SaaS level. SaaS level will provide ready-to-use services. It is a frontend that is used to interact with the services. It will have modules such as authentication, user subscriptions, and more.

Docker containers employ a client-server model where the Docker client interacts with the Docker daemon, responsible for managing containers and images. Docker images are immutable, layered templates that include an application and its dependencies. Containers are runnable instances of images, isolated from each other and the host system, ensuring consistency and portability. Docker registries store and distribute images, with Docker Hub being a prominent public registry. Docker Compose simplifies multi-container application management, and Docker provides networking and volume solutions for inter-container communication and data persistence, making it a versatile and widely used technology for containerization and application deployment.

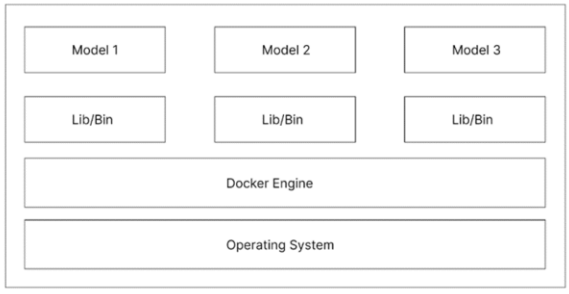

Figure 2: Basic Architecture of a General Docker Container

The server runs a Kubernetes cluster which is responsible for managing the containers. The deployment and management of containerized apps are made easier by the open-source Kubernetes container orchestration platform. It functions inside a group of computers that consists of a master node and several worker nodes. Worker nodes execute containers within Pods, while the master node holds essential components including the API server, etc, controller manager, and scheduler. To maintain the necessary duplicates, these Pods may be maintained by Replication Controllers and ReplicaSets. ConfigMaps and Secrets handle configuration and confidential data, while Services and Ingress offer networking options. Kubernetes is a crucial tool for automating the deployment and scaling of applications in a containerized environment because it makes sure that the cluster's real state matches the expected state even in the face of node failures or scaling requirements.

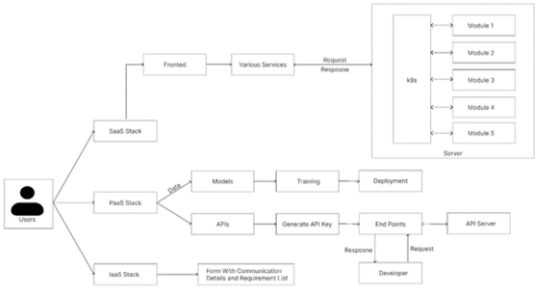

Figure 3: Architecture of the implementation of the Proposed Methodology

The Representational State Transfer (REST) architectural style is followed in the construction of a RESTful server. A stateless server, or one that does not retain any state information about its clients or their requests, is one of the major characteristics of a RESTful server. Rather, every request that comes in from a client has all the data that the server needs to process and reply to it. Because the server does not need to maintain track of client sessions or other stateful data, the stateless design helps to simplify and increase the scalability of the server architecture.

With the help of the following technologies, a robust and non-redundant architecture could be developed. It is simple to implement and deploy. Also, with the help of Docker Container, scaling.

IV. RESULTS AND DISCUSSION

Using the architecture, a simple AI-as-a-service was made. It has two components - SaaS and PaaS and targets 5 services - Cold Email generator, Image Generator, Copywriting tools, SEO optimizer, and advertisement assistant. For the backend server NodeJS was used and to manage the docker container Kubernetes was used.

For the SaaS part, a front end is built for the user to interact with. A Server is made that hosts all the containers for each service is managed by Kubernetes, and will be deployed to the cloud.

The server is made RESTful so that it's stateless and does not require to know the type of request that is coming to it. With the help of this feature, a single server was made to handle both end-user applications and API requests for services to be used by other applications.

For the PaaS part, there is an API gateway that handles all the API requests and sends the request to the same backend server. API management service is used to handle all the requests. The same server is used for both the SaaS and PaaS implementations as it is a RESTful server.

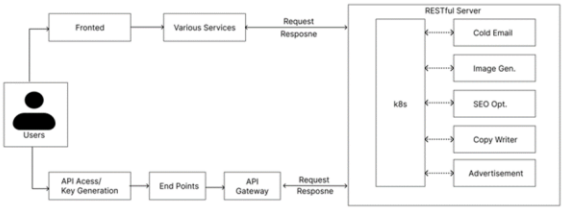

Figure 4: Architecture of the proposed AIaaS Application

This implementation provided a sense that this architecture can be used by any organization that wishes to implement Ai-as-a-service model and provide SaaS and PaaS features. Having contains eliminates the need of having multiple backend services that do the same thing. Use the RESTful architecture a single backend server was able to handle both the End-user applications and API requests. Although the implementation done in this paper was pretty basic, this architecture holds the potential for larger applications.

CONCLUSION

This paper focuses on the architecture and purpose of a simple and efficient model for an AI product that has SaaS and PaaS components. A simple application was implemented using this architecture. Technologies like Docker, Kubernetes, streamlit, and Python were used. This paper was able to accomplish the following using the architecture.

FUTURE SCOPE

As this paper provided only the architecture and proper testing was not done. Further rigorous testing is required. Also, many more technologies could be added like Kafka, load balancers, and many more. Also, a proper application following the architecture is to be built. Optimizing the architecture is also a major task for future work. Making the model more scalable is also a major concern. Although because of docker and Kubernetes, the model is highly scalable but adding cloud integration could potentially make it more scalable.

REFERENCES

[1] Raghu, Chilukuri., Aymeric, Grassart., Karnig, Kerkonian., Harry, Madanyan., Rudy, Minasian., Thanadham, Thaveesaengsiri., John, Tran. (2015). Software as a service framework for the digital engagement and conclusion of clients by service professionals.

[2] Dinkar, Sitaram., Geetha, Manjunath. (2012). Platform as a Service. 73-152. doi: 10.1016/B978-1-59749-725-1.00003-2

[3] Shamim, Hossain. (2013). Infrastructure as a Service. 146-169. doi: 10.4018/978-1-4666-2187-9.CH002

[4] Sasaki,, Ken-ichi. (2022). Containerizing Microservices Using Kubernetes. 213-230. doi: 10.1007/978-1-4842-8832-0_9

[5] (2023). Building Smart Mobile Apps with Flutter and OpenAI AI-Powered Text and Images and Chatbots. International Journal For Science Technology And Engineering, 11(6):904-908. doi: 10.22214/ijraset.2023.53796

[6] Zhou, J., Zhou, B., & Li, S. (2014). Automated Model-Based Performance Testing for PaaS Cloud Services. 2014 IEEE 38th International Computer Software and Applications Conference Workshops, 644-649.

[7] Claus, Pahl., Brian, Lee. (2015). Containers and Clusters for Edge Cloud Architectures -- A

Technology Review. 379-386. doi: 10.1109/FICLOUD.2015.35

[8] Wei, Cui., Hanglong, Zhan., Bao, Li., Hu, Wang., Donggang, Cao. (2016). Cluster as a Service: A Container Based Cluster Sharing Approach with Multi-user Support. 111-118. doi: 10.1109/SOSE.2016.16

[9] Ye, Tao., Xiaodong, Wang., Xiaowei, Xu., Guozhu, Liu. (2018). Container-as-a-service architecture for business workflow. International Journal of Simulation and Process Modelling, 13(2):102-115. doi: 10.1504/IJSPM.2018.10012815

[10] Qi, Zhang., Ling, Liu., Calton, Pu., Qiwei, Dou., Liren, Wu., Wei, Zhou. (2018). A Comparative Study of Containers and Virtual Machines in Big Data Environment. arXiv: Distributed, Parallel, and Cluster Computing

[11] Yao, Xiabing., Hu, Linhong. (2018). Contain cloud platform and server.

[12] Ye, Tao., Xiaodong, Wang., Xiaowei, Xu., Guozhu, Liu. (2018). Container-as-a-service architecture for business workflow. International Journal of Simulation and Process Modelling, 13(2):102-115. doi: 10.1504/IJSPM.2018.10012815

[13] Kong, Heqing., Li, Shengqi., Zhang, Qianxian., Zhao, Jiangang. (2018). Platform-based container.

[14] Kousiouris, G., Tsarsitalidis, S., Psomakelis, E., Koloniaris, S., Bardaki, C., Tserpes, K., Nikolaidou, M., & Anagnostopoulos, D. (2019). A microservice-based framework for integrating IoT management platforms, semantic and AI services for supply chain management. *ICT Express, 5*, 141-145.

[15] Chen, Tong., Huang, Yongjian., Wang, Yongze. (2019). Big data platform architecture system based on container cluster.

[16] Marvin, Newlin., Kyle, Smathers., Mark, E., DeYoung. (2019). ARC Containers for AI Workloads: Singularity Performance Overhead. 1-. doi: 10.1145/3332186.3333048

[17] Parsaeefard, S., Tabrizian, I., & Leon-Garcia, A. (2019). Artificial Intelligence as a Services (AI-aaS) on Software-Defined Infrastructure. ArXiv, abs/1907.05505.

[18] Hussein, M., Mousa, M., & AlQarni, M. (2019). A placement architecture for a container as a service (CaaS) in a cloud environment. *Journal of Cloud Computing, 8*.

[19] Casati, F., Govindarajan, K., Jayaraman, B., Thakur, A., Palapudi, S., Karakusoglu, F., & Chatterjee, D. (2019). Operating Enterprise AI as a Service. *International Conference on Service Oriented Computing*.

[20] Yinzhi, Wang., R., Todd, Evans., Lei, Huang. (2019). Performant Container Support for HPC Applications. doi: 10.1145/3332186.3332226

[21] Fabiana, Rossi. (2019). Self-Management of Containers Deployment in Decentralized Environments. 315-318. doi: 10.1109/SERVICES.2019.00088

[22] Olivier Debauche, Saïd Mahmoudi, Sidi Ahmed Mahmoudi, Pierre Manneback, Frédéric Lebeau, A new Edge Architecture for AI-IoT services deployment, Procedia Computer Science, Volume 175, 2020, Pages 10-19, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2020.07.006.

[23] Khouloud, Boukadi., Molka, Rekik., Jorge, Bernal, Bernabe., Jaime, Lloret. (2020). Container description ontology for CaaS. International Journal of Web and Grid Services, 16(4):341-363. doi: 10.1504/IJWGS.2020.110944

[24] Subil, Abraham., Arnab, K., Paul., Redwan, Ibne, Seraj, Khan., Ali, R., Butt. (2020). On the Use of Containers in High Performance Computing Environments. 284-293. doi: 10.1109/CLOUD49709.2020.00048

[25] Mabrook, Al-Rakhami., Abdu, Gumaei., Mohammed, Abdullah, Alsahli., Mohammad, Mehedi, Hassan., Atif, Alamri., Antonio, Guerrieri., Giancarlo, Fortino., Giancarlo, Fortino. (2020). A lightweight and cost effective edge intelligence architecture based on containerization technology. World Wide Web, 23(2):1341-1360. doi: 10.1007/S11280-019-00692-Y

[26] Implementation of Artificial Intelligence (AI): A Roadmap for Business Model Innovation.

[27] Mujkanovic, N., Sivalingam, K., & Lazzaro, A. (2020). Optimising AI Training Deployments using Graph Compilers and Containers. *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, 1-8.

[28] Janbi, N., Katib, I.A., Albeshri, A.A., & Mehmood, R. (2020). Distributed Artificial Intelligence-as-a-Service (DAIaaS) for Smarter IoE and 6G Environments. *Sensors (Basel, Switzerland), 20*.

[29] Neuhüttler, J., Kett, H., Frings, S., Falkner, J., Ganz, W., & Urmetzer, F. (2020). Artificial Intelligence as Driver for Business Model Innovation in Smart Service Systems. International Conference on Applied Human Factors and Ergonomics.

[30] Javadi, S.A., Cloete, R., Cobbe, J., Lee, M.S., & Singh, J. (2020). Monitoring Misuse for Accountable 'Artificial Intelligence as a Service'. *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*.

[31] Li, Jiayi., He, Tonglu., Yang, Fei., Guo, Xuedong., Ren, Yongliang. (2020). AI service open middle station and method.

[32] Kim, Kee, Baek., Cho, Soo, Hyun., Lee, Yong, Hyuk., Yang, Young, Jin., Bae, Seung, In., Song, Jin, Hee. (2020). Cloud system based on container and method for providing cloud service having enhanced scalability and autonomy.

[33] Craig, Willis., Mike, Lambert., Kenton, McHenry., Christine, Kirkpatrick. (2017). Container-based Analysis Environments for Low-Barrier Access to Research Data. 58-. doi: 10.1145/3093338.3104164

[34] Kim, Kee, Baek., Cho, Soo, Hyun., Lee, Yong, Hyuk., Yang, Young, Jin., Bae, Seung, In., Song, Jin, Hee. (2020). Method for providing cloud service based on container.

[35] Lins, S., Pandl, K.D., Teigeler, H. et al. Artificial Intelligence as a Service. Bus Inf Syst Eng 63, 441–456 (2021). https://doi.org/10.1007/s12599-021-00708-w

[36] Tuan, Le-Anh., Quan, Ngo-Van., Phuong, Vo-Huy., Dang, Huynh-Van., Quan, Le-Trung. (2021). A Container-Based Edge Computing System for Smart Healthcare Applications. 324-336. doi: 10.1007/978-3-030-77424-0_27

[37] Pratama, I.P. (2021). The implementation of Container as a Service (CaaS) cloud using openSUSE kubic.

[38] John, M.M., Olsson, H.H., & Bosch, J. (2022). Towards an AI-driven business development framework: A multi-case study. *Journal of Software: Evolution and Process, 35*.

[39] Kumar, M., & Kaur, G. (2022). Containerized AI Framework on Secure Shared Multi-GPU Systems. *2022 Seventh International Conference on Parallel, Distributed and Grid Computing (PDGC)*, 243-247.

[40] Fadia, Shah., Aamir, Anwar., Ijaz, ul, haq., Hussain, AlSalman., Saddam, Hussain., Suheer, Al-Hadhrami. (2022). Artificial Intelligence as a Service for Immoral Content Detection and Eradication. Scientific Programming, 2022:1-9. doi: 10.1155/2022/6825228

[41] Lauren, Kathleen, McMillan., Liz, Varga. (2022). A review of the use of artificial intelligence methods in infrastructure systems. Engineering Applications of Artificial Intelligence, 116:105472-105472. doi: 10.1016/j.engappai.2022.105472

[42] (2023). Implementation and Comparison of Artificial Intelligence Techniques in Software Testing. doi: 10.1109/iscon57294.2023.10112041