**Name: Ida Ayu Sri Pramesti Dewi**

**Student Id: E2100299/210030472**

1. Explain the fundamental difference between virtualisation using VM and virtualisation using container.

Answer:

Virtual Machine (VM) and container have a fundamental difference in how they virtualize and utilize resources. VM virtualizes hardware using a hypervisor by running a complete operating system (guest OS) on each unit. This provides strong isolation but requires large resources such as memory and storage, as well as longer startup times. For example, running Windows Server through VirtualBox on a Windows laptop requires a separate OS, which can increase overhead. In contrast, containers virtualize at the operating system level and share the host OS kernel, making them lighter and more efficient. Containers only bring the application and its dependencies without an additional OS, so the startup process is fast and resource usage is lower. For example, using Docker Desktop on Windows allows running several applications within seconds with minimal overhead. VMs are suitable for environments that require strong isolation, while containers are ideal for modern, efficient, and fast applications.

1. Referring to Fig. 1., explain the management responsibilities for SaaS, PaaS and IaaS.

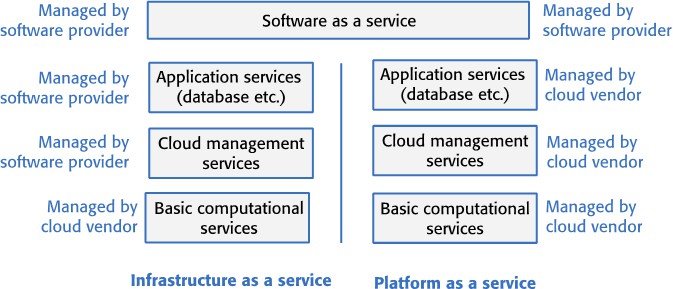


Figure 1: Management Responsibilities of IaaS, PaaS and SaaS.

Answer:

* IaaS (Infrastructure as a Service): In the IaaS model, such as Google Compute Engine or Cloud Storage, users are responsible for managing almost all aspects of the infrastructure. They must install the operating system, configure servers, and manage the deployed applications. Service providers like Google only manage the physical infrastructure, such as servers and networks, while users manage their own software, settings, and applications.
* PaaS (Platform as a Service): In the PaaS model, such as Heroku, which is used for web application deployment, users are only responsible for the application code they create. Heroku, as the service provider, manages the infrastructure, operating system, and server scalability. Users do not need to manage servers or operational settings but still have full control over the application they deploy. Users must ensure their application code is secure and works well, but the service provider manages the infrastructure and platform.
* SaaS (Software as a Service): In the SaaS model, such as Google Workspace which includes applications like Gmail, Google Docs, and Google Sheets, users only need to focus on using the applications. All infrastructure-related aspects, such as system updates, security, and server maintenance, are fully handled by the service provider, Google. Users do not need to worry about managing servers or data security as this is the responsibility of the service provider.

1. What are the benefits to software product vendors of delivering software as a service. In what situations might you decide not to deliver software in this way?

Answer:

The benefits to software vendors of delivering software as a service (SaaS) are more predictable revenue through a subscription model. An example is Google Workspace, which includes applications such as Gmail, Google Docs, and Google Sheets. With this subscription model, vendors receive regular payments from users, either monthly or annually, providing a stable revenue stream. Additionally, users always get the latest version of the software automatically without needing to perform manual updates. SaaS also reduces distribution and installation costs, as well as enables flexible scalability for users with businesses of various sizes.

However, in situations that require high levels of security or regulatory compliance, such as in the healthcare or financial sectors, some organizations prefer on-premise software solutions like Microsoft Office (desktop version) to have better control over their sensitive data. Furthermore, if the internet connection is unstable, using SaaS services like Google Workspace becomes less practical. For companies with highly specialized workflows, SaaS may need to be more flexible compared to on-premise software solutions that can be further customized.

1. Why is cost not the most important factor to consider when choosing a cloud platform for development and software delivery?

Answer:

Cost is still important but not the main factor when choosing a cloud platform. In my opinion, it is more important to choose a platform that has the features necessary for the application and is easy to use. A cheap platform that is difficult to use or lacks the required features can actually waste time and effort. For example, Google Cloud or AWS offers many features needed for large-scale application development, such as scalability and high-level security. However, if a cheaper platform like a small hosting service does not have these features, it could lead to performance and security issues in the future. In addition, system reliability, security features, and ease of use are often more important than just cost. For example, in the financial industry, where sensitive data is highly protected, a platform offering high levels of encryption and compliance, like Microsoft Azure, is preferred even though it may be more expensive because its security is more guaranteed compared to cheaper options that lack such features.

1. Explain why microservices should have low coupling and high cohesion.

Answer:

Microservices should have low coupling and high cohesion to ensure the system remains flexible, maintainable, and scalable. Low coupling means each microservice operates independently and does not rely heavily on other services. This is important so that if one service fails or is updated, other services are not affected. For example, in an e-commerce platform, the payment service should continue functioning even if the product catalog service changes. Meanwhile, high cohesion means each microservice focuses on a single, clear business function or task. A highly cohesive service is easier to understand, test, and maintain because its functionality is focused and not mixed with unrelated tasks. For instance the user authentication service handles only login and security without being involved in inventory management or payment processing. By adhering to these principles, microservices can evolve and scale independently, making the system more flexible, easier to test, and faster to develop.

1. Explain the differences between synchronous and asynchronous microservices interactions.

Answer:

In synchronous interactions, communication between services occurs directly and in real-time, where the sender service sends a request and waits for a response from the receiver service before proceeding further. For example, in an online payment system users must wait for payment confirmation before moving to the next step, such as order processing. The advantage of this interaction is receiving immediate feedback which is essential for operations requiring instant decisions. However the disadvantage is that system performance can be hindered if the response time is too long, and if the receiver service fails, the sender service will also be blocked.

Meanwhile, in asynchronous interactions, the sender service sends a request and continues processing without waiting for an immediate response from the receiver service. For example, when users register for a system, the registration request can be processed first while the confirmation email is sent later without interrupting the registration flow. The advantage of this interaction is that the system is more flexible and can handle multiple requests simultaneously without waiting for responses, thus improving scalability and efficiency. However this system is more complex to manage, as it requires mechanisms to handle message queues and ensure timely delivery of responses.

The main difference between the two lies in response speed and communication processes. Synchronous interactions require both parties to be available simultaneously with immediate responses, while asynchronous interactions allow communication to occur at different times with potentially delayed responses.

Readings : [https://www.freecodecamp.org/news/a-beginner-friendly-introduction-to-containers-](https://www.freecodecamp.org/news/a-beginner-friendly-introduction-to-containers-vms-and-docker-79a9e3e119b/) [vms-and-docker-79a9e3e119b/](https://www.freecodecamp.org/news/a-beginner-friendly-introduction-to-containers-vms-and-docker-79a9e3e119b/)