

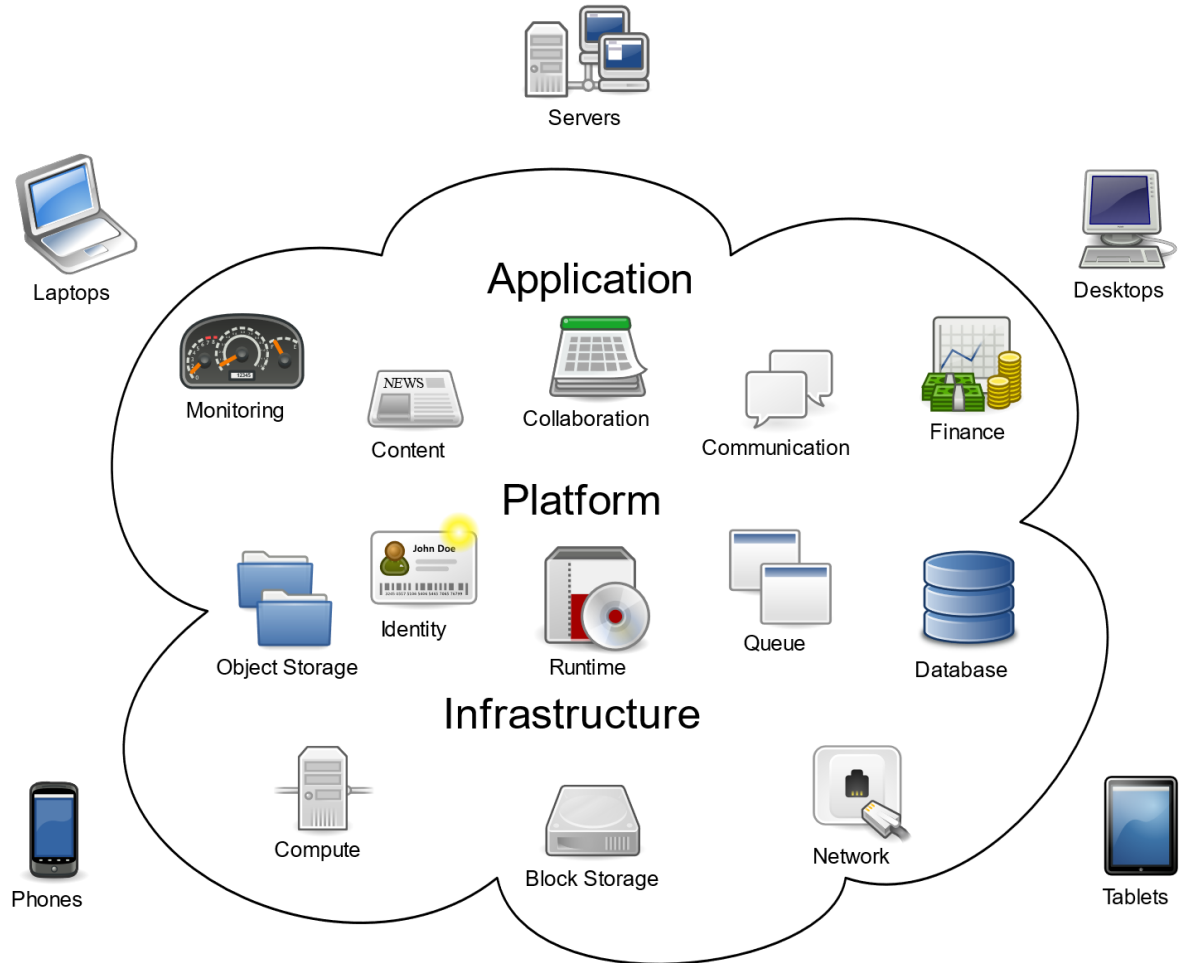
# *Cloud-based software*

**Professor Hossein Saiedian**

EECS 348: Software Engineering

Spring 2023

- On-demand availability and delivery of computing services
  - Servers
  - Storage
  - Databases
  - Networking
  - Software
  - Analytics

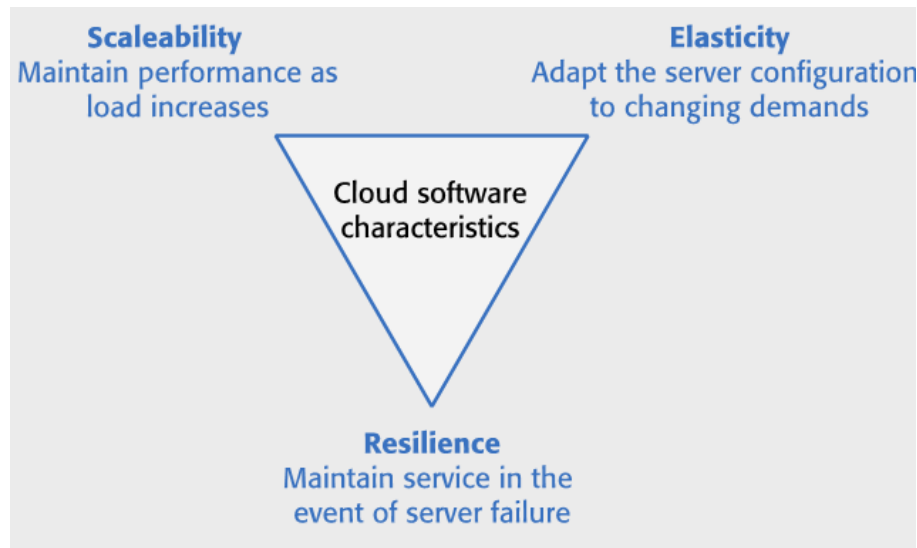


- The cloud is a very large number of remote servers that are offered for rent by companies that own these servers
  - You can rent as many servers as you need, run your software on these servers, and make them available to your customers
  - Your customers can access these servers from their own computers or other networked devices
  - You may rent a server and install your own software
  - You may pay for access to software products that are available on the cloud

- Remote servers are virtual (implemented in software, not hardware)
  - Many virtual servers can run simultaneously on each cloud hardware node
  - Very powerful hardware to run multiple virtual servers
    - \* Performance not affected



- Cloud management software makes it easy to acquire and release servers on demand
  - If you need resources for only a short time, you simply pay for the time that you need
  - This means that software that runs on the cloud can be **scalable**, **elastic**, and **resilient**



- **Scalability** reflects the ability of your software to cope with increasing numbers of users
  - As the load on your software increases, your software automatically adapts so that the system performance and response time are maintained

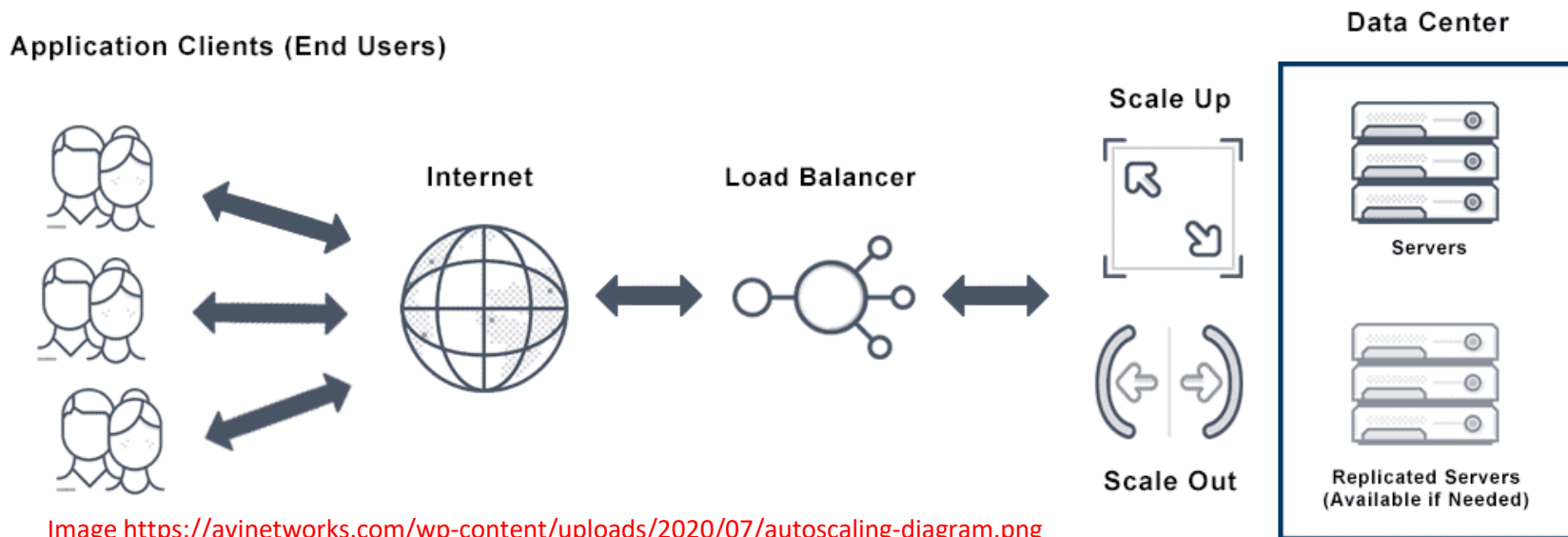
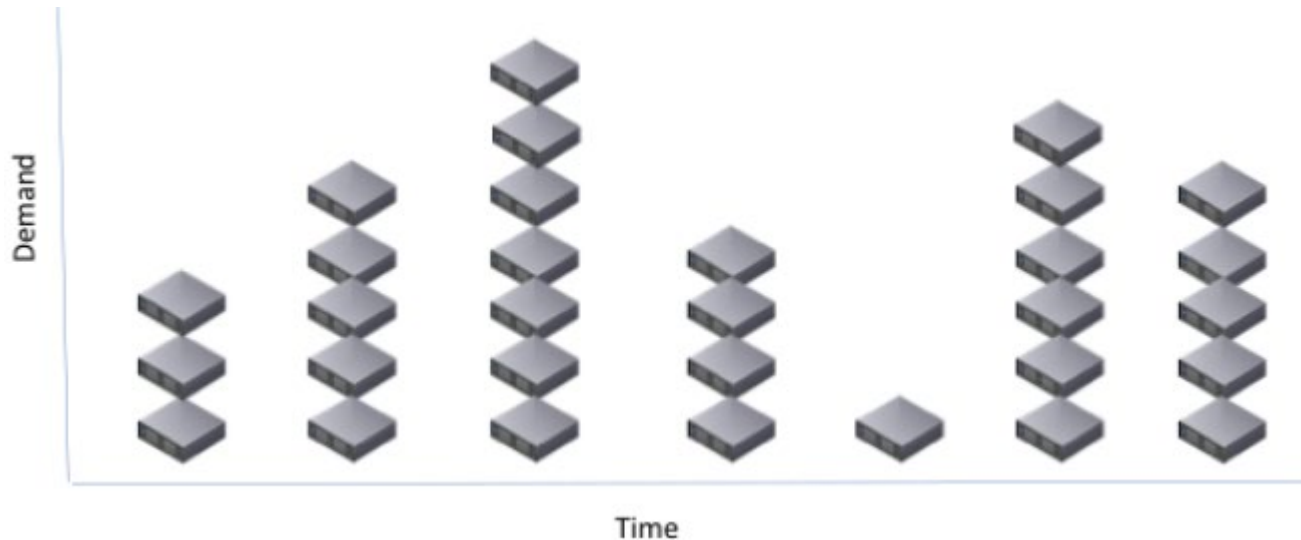


Image <https://avinetworks.com/wp-content/uploads/2020/07/autoscaling-diagram.png>

- Elasticity is related to scalability but also allows for scaling down as well as scaling up
  - That is, you can monitor the demand on your application and add or remove servers dynamically as the number of users changes



- **Scalability:** handles the changing needs of an application
  - Statically adding or removing resources to meet applications demands if needed
  - Provisioning new servers to meet static demand growth
- **Elasticity:** to match the resources allocated with the actual number of resources needed at any given point in time
  - Scalability of the resources (CPU, memory, ...)



## Scalability Vs Elasticity

Increasing the capacity to meet the increasing workload	"Increasing or reducing" the capacity to meet the increasing or reducing workload
In a scaling environment, the available resources may exceed to meet the future demands	In elasticity environment the available resources matches the current demands
Scalability adapts only to the workload increase by provisioning the resources in an incremental manner	It adapts to both the workload increase & workload decrease in an automatic manner
Scalability enables a corporate to meet expected demands for services with long term strategic needs	Elasticity enables a corporate to meet the unexpected changes in the demand for service with "short-term", tactical needs

# Scaling can be vertical or horizontal

## Vertical Scaling



1 CPU / 1 GB RAM  
~ \$10/mo



2 CPU / 2 GB RAM  
~ \$20/mo



4 CPU / 8 GB RAM  
~ \$80/mo

## Horizontal Scaling



1 CPU / 1 GB RAM  
~ \$10/mo



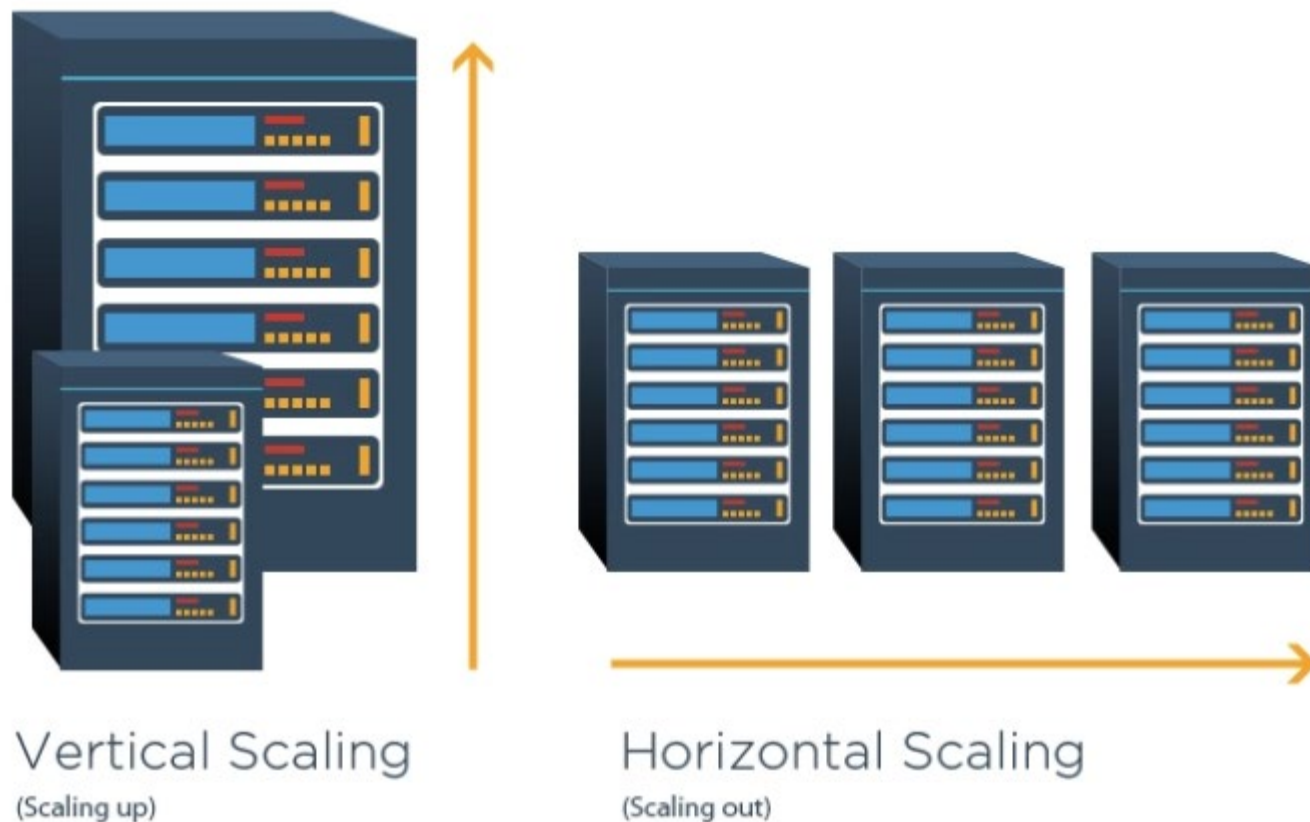
2 x (1 CPU / 1 GB RAM)  
~ \$20/mo



4 x (1 CPU / 1 GB RAM)  
~ \$40/mo

# Scaling can be vertical or horizontal

- It is also called scaling up or scaling out



- Resilience means that you can design your software architecture to tolerate server failures
  - You can make several copies of your software concurrently available
  - If one of these fails, the others continue to provide a service

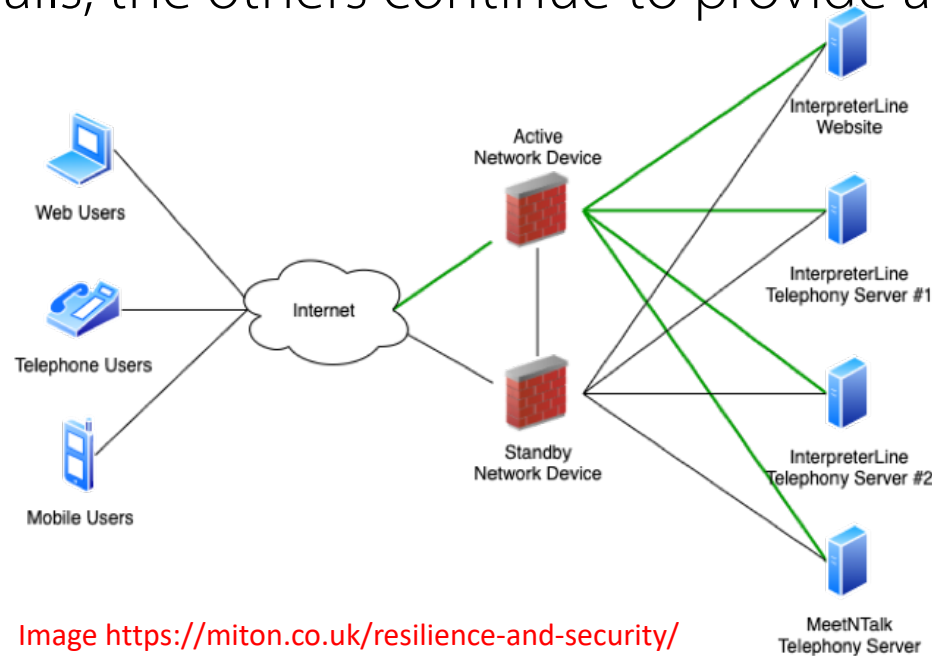


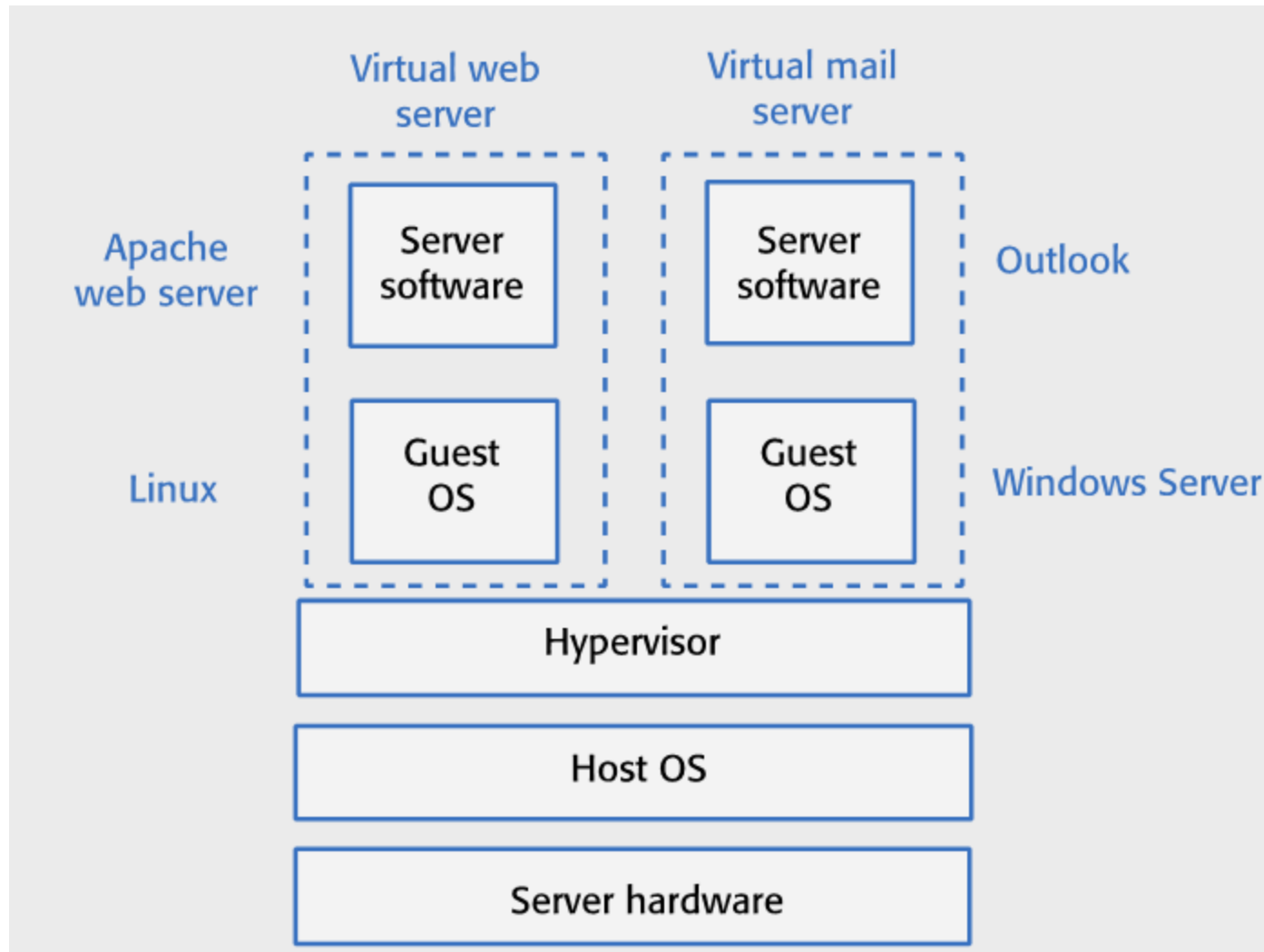
Image <https://miton.co.uk/resilience-and-security/>

# Why cloud computing?

- **Cost:** You avoid the initial capital costs of hardware procurement
- **Startup time:** You don't have to wait for hardware to be delivered before you can start work; using the cloud, you can have servers up and running in a few minutes.
- **Server choice:** If you find that the servers you are renting are not powerful enough, you can upgrade to more powerful systems
  - You can add servers for short-term requirements, such as load testing.
- **Distributed development:** If you have a distributed development team, working from different locations, all team members have the same development environment and can seamlessly share all information

- A virtual server runs on an underlying physical computer and is made up of an operating system plus a set of software packages that provide the server functionality required
- A virtual server is a stand-alone system that can run on any hardware in the cloud
  - This 'run anywhere' characteristic is possible because the virtual server has no external dependencies
- Virtual machines (VMs), running on physical server hardware, can be used to implement virtual servers
  - A hypervisor provides hardware emulation that simulates the operation of the underlying hardware
- If you use a virtual machine to implement virtual servers, you have exactly the same hardware platform as a physical server

# Implementing a virtual server as a VM



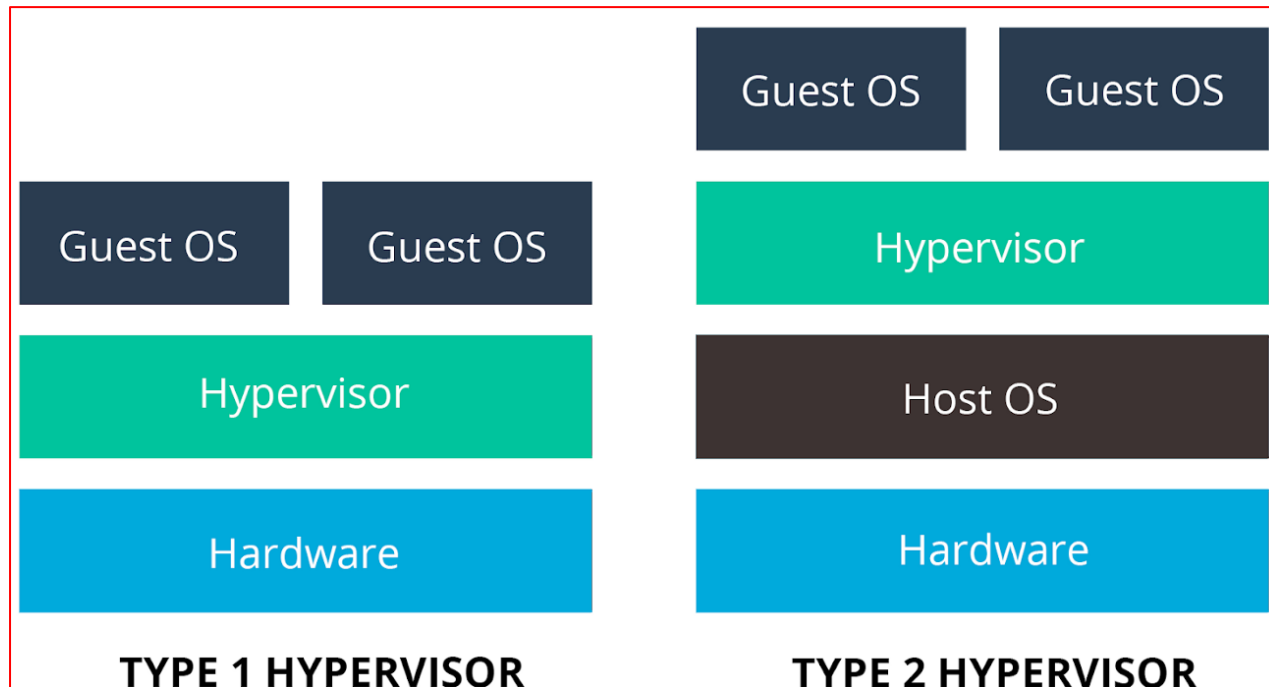
# What is a hypervisor

- **Hypervisor:** A type of computer software, firmware or hardware that creates and runs virtual machines
- **Host machine:** A computer on which a hypervisor runs one or more virtual machines is called a host machine
- **Guest machine:** Each virtual machine is called a guest machine



# Hypervisor type 1 and type 2

- Type 1 hypervisor (bare-metal): runs directly on the physical hardware of the host machine (e.g., MS Hyper-V)
- Type 2 hypervisor (hosted): installed on an OS (e.g., MS Virtual PC)



# Implementing a virtual server as a VM

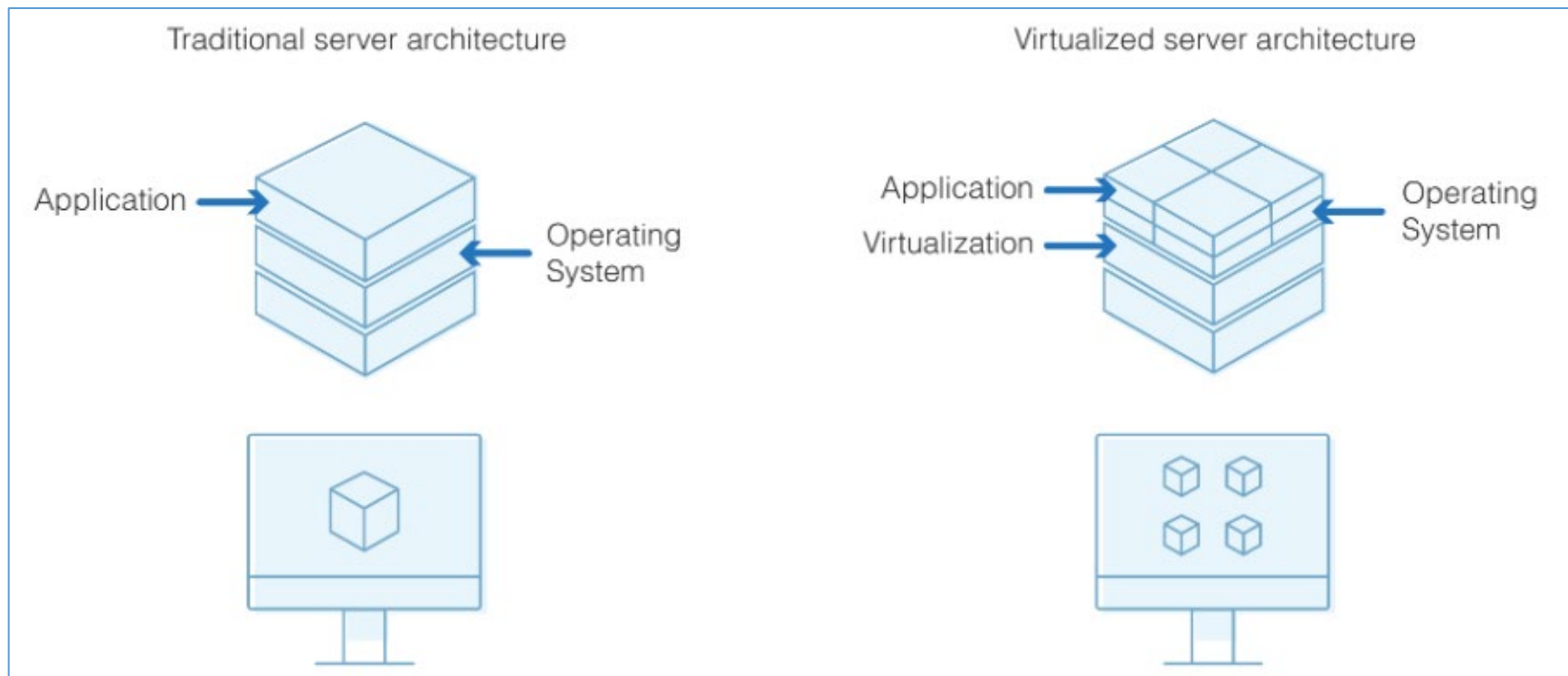
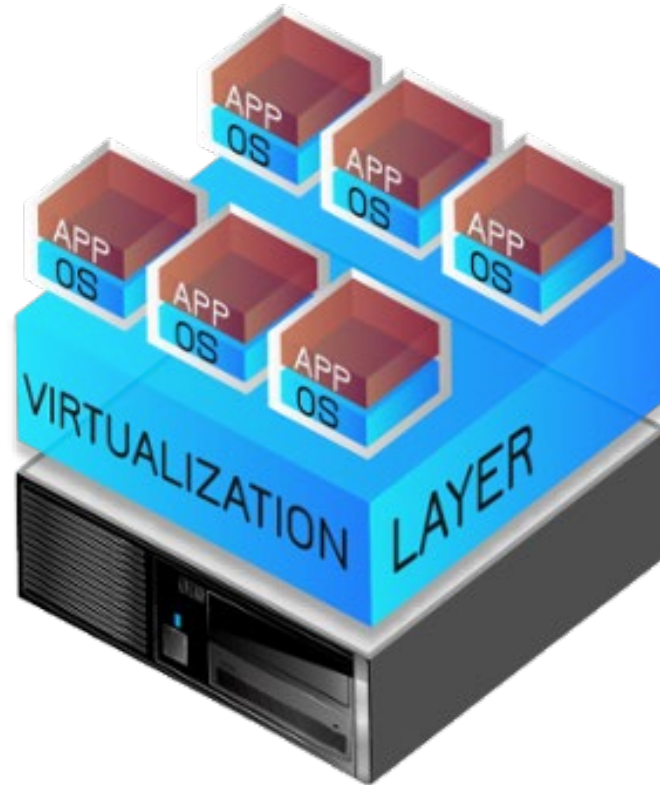


Image <https://www.dnsstuff.com/what-is-virtualization>

# Implementing a virtual server as a VM



**Traditional Server  
Architecture**



**Virtualized Server  
Architecture**

# Implementing a virtual server as a VM



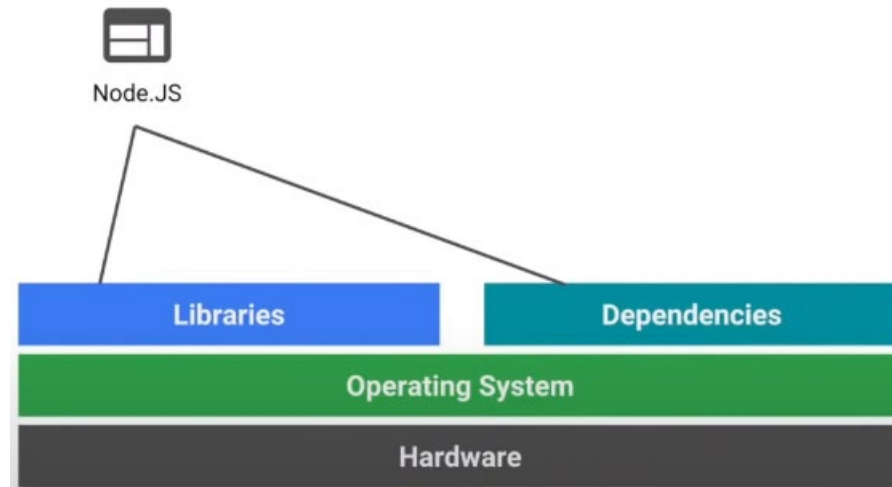
- Containers are used to ship various goods transported across the seas
- What about the code you write?



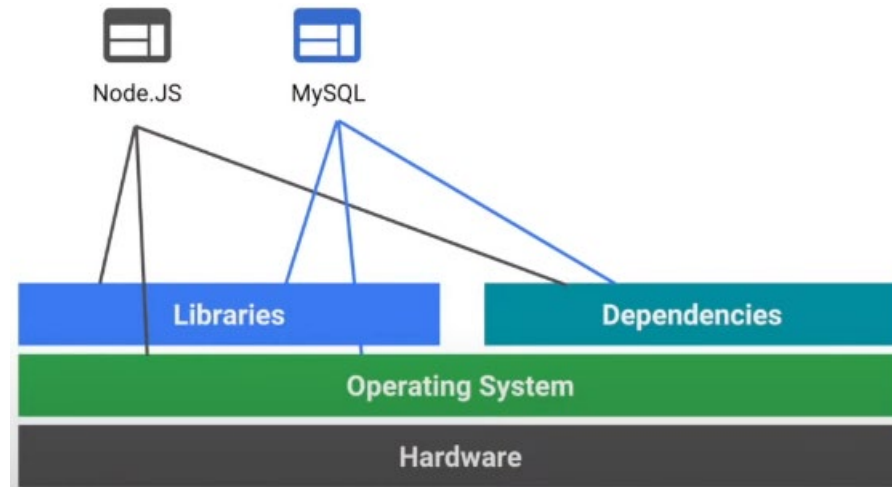
- Traditional software deployment
  - Hardware
  - Operating systems
  - Apps installed on top of the operating systems



- Traditional software deployment
- Each new software app will have its own
  - Libraries
  - Dependencies

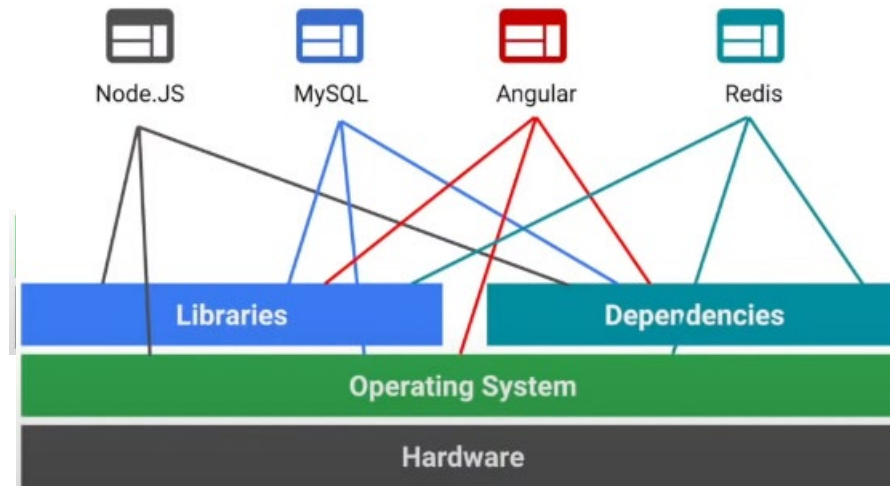


- Traditional software deployment
- Each new software app will have its own dependencies

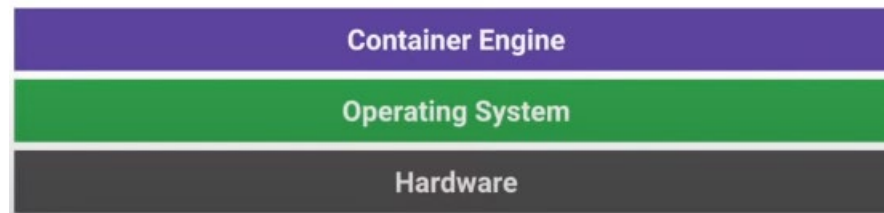




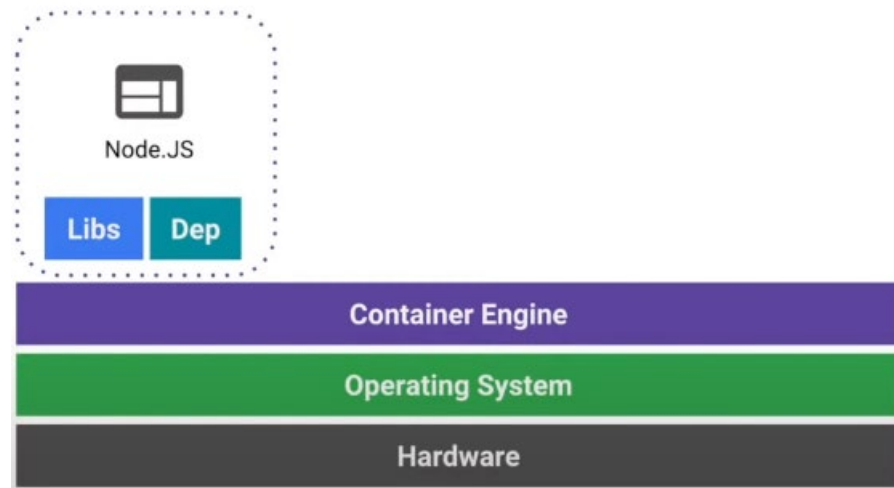
- Traditional software deployment
  - Each new software app will have its own dependencies
  - Some dependencies will introduce conflicts
    - \* Different versions of the same library
    - \* How to solve problems like build, ship, deploy, scale?



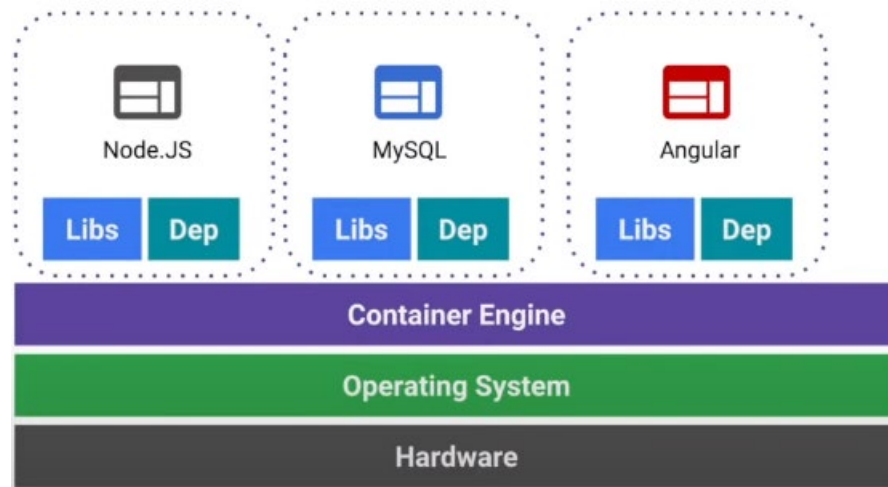
- The container technology
  - Examples: Docker, AWS ECS, Microsoft Azure



- The container technology
  - Help develop containers
  - The libraries and dependencies are included in the container
  - Easily move from one machine to another



- The container technology
  - Libraries and dependencies come as part of the package

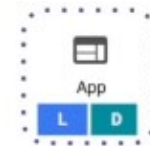


- The container technology
  - A container can be moved across virtual environments without worrying about the underlying dependencies



- The container technology

## Applications - Build, Ship, Deploy, Scale



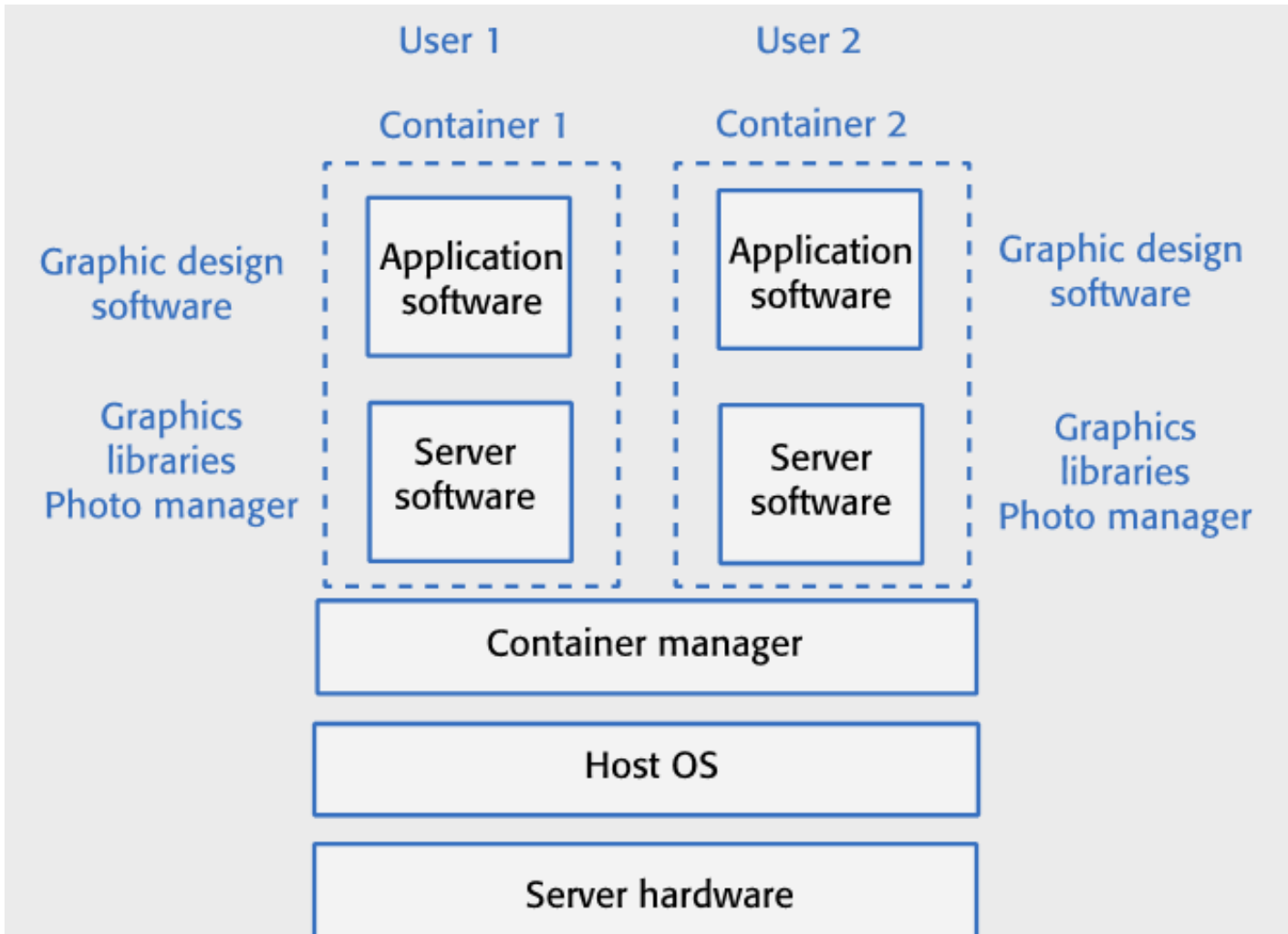
	Traditional	Containers
Installation	<ul style="list-style-type: none"><li>• Time Consuming</li><li>• Multiple Commands</li></ul>	😊
Software Dependencies	<ul style="list-style-type: none"><li>• Dependency Hell</li></ul>	😊
Packaging	<ul style="list-style-type: none"><li>• Not Easy</li></ul>	😊
Dev -> Stage -> Prod - Shipping	<ul style="list-style-type: none"><li>• Duplicate Efforts</li><li>• Compatibility issues</li></ul>	😊
Isolation	<ul style="list-style-type: none"><li>• Not Possible</li></ul>	😊
Scalability	<ul style="list-style-type: none"><li>• Not Easy</li></ul>	😊

- If you are running a cloud-based system with many instances of applications or services, these all use the same operating system, you can use a simpler virtualization technology called 'containers'
- Using containers accelerates the process of deploying virtual servers on the cloud.
  - Containers are usually megabytes in size whereas VMs are gigabytes.
  - Containers can be started and shut down in a few seconds rather than the few minutes required for a VM

- Containers are an operating system virtualization technology that allows independent servers to share a single operating system
  - They are particularly useful for providing isolated application services where each user sees their own version of an application

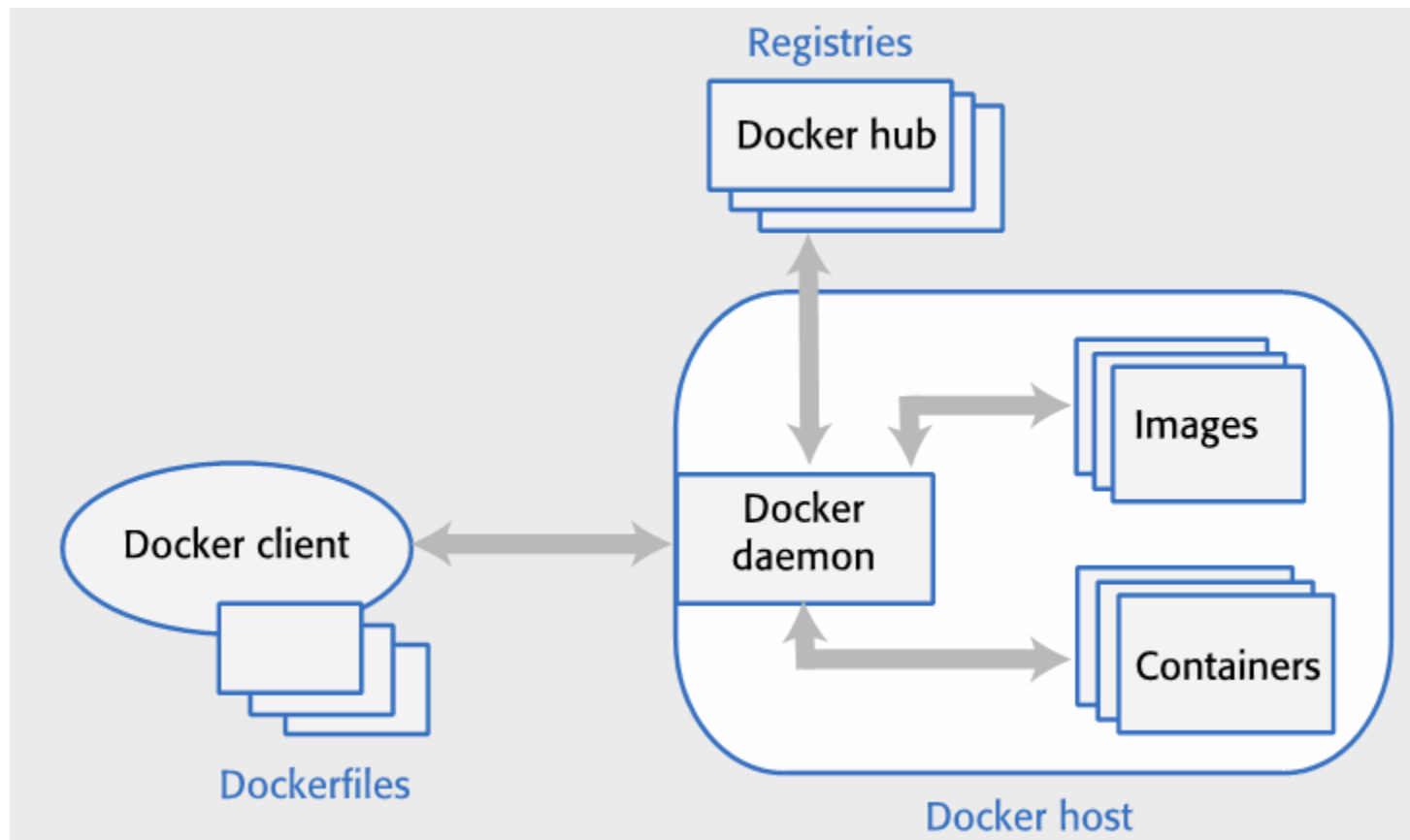


# What is a container



- Containers were developed by Google around 2007 but containers became a mainstream technology around 2015
- An open-source project called Docker provided a standard means of container management that is fast and easy to use
- ***Docker is a container management system that allows users to define the software to be included in a container as a Docker image***
- It also includes a run-time system that can create and manage containers using these Docker images


# The Docker container system



- **Docker hub:** This is a registry of images that has been created. These may be reused to setup containers or as a starting point for defining new images.
- **Containers:** Containers are executing images
  - An image is loaded into a container and the application-defined by the image starts execution
  - Containers may be moved from server to server without modification and replicated across many servers.
  - You can make changes to a Docker container (e.g. by modifying files) but you then must commit these changes to create a new image and restart the container.

- The idea of a service that is rented rather than owned is fundamental to cloud computing.
- Infrastructure as a service (IaaS)
  - Cloud providers offer different kinds of infrastructure service such as a compute service, a network service and a storage service that you can use to implement virtual servers
- Platform as a service (PaaS)
  - This is an intermediate level where you use libraries and frameworks provided by the cloud provider to implement your software. These provide access to a range of functions, including SQL and NoSQL databases
- Software as a service (SaaS)
  - Your software product runs on the cloud and is accessed by users through a web browser or mobile app

# Various things as a service

 User managed

 Provider managed

On premises

Application

Data

Runtime

Middleware

Operating system

Virtualization

Networking

Storage

Servers

IaaS

Application

Data

Runtime

Middleware

Operating system

Virtualization

Networking

Storage

Servers

PaaS

Application

Data

Runtime

Middleware

Operating system

Virtualization

Networking

Storage

Servers

SaaS

Application

Data

Runtime

Middleware

Operating system

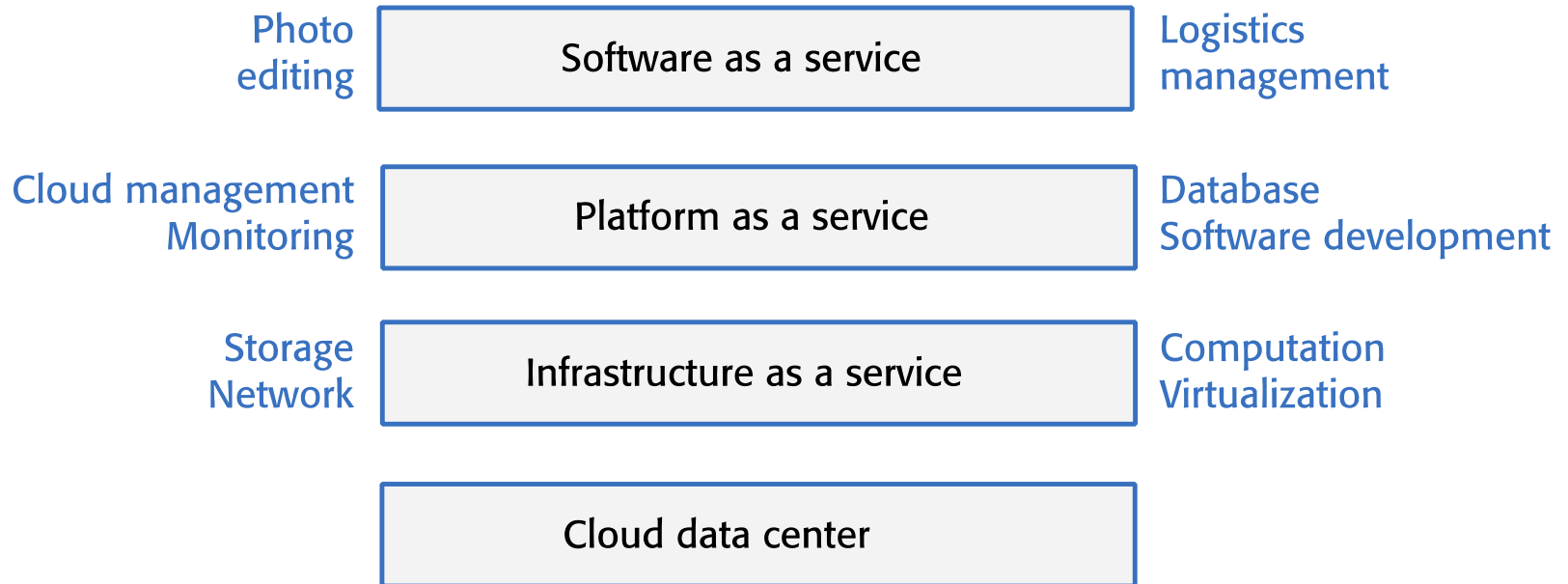
Virtualization

Networking

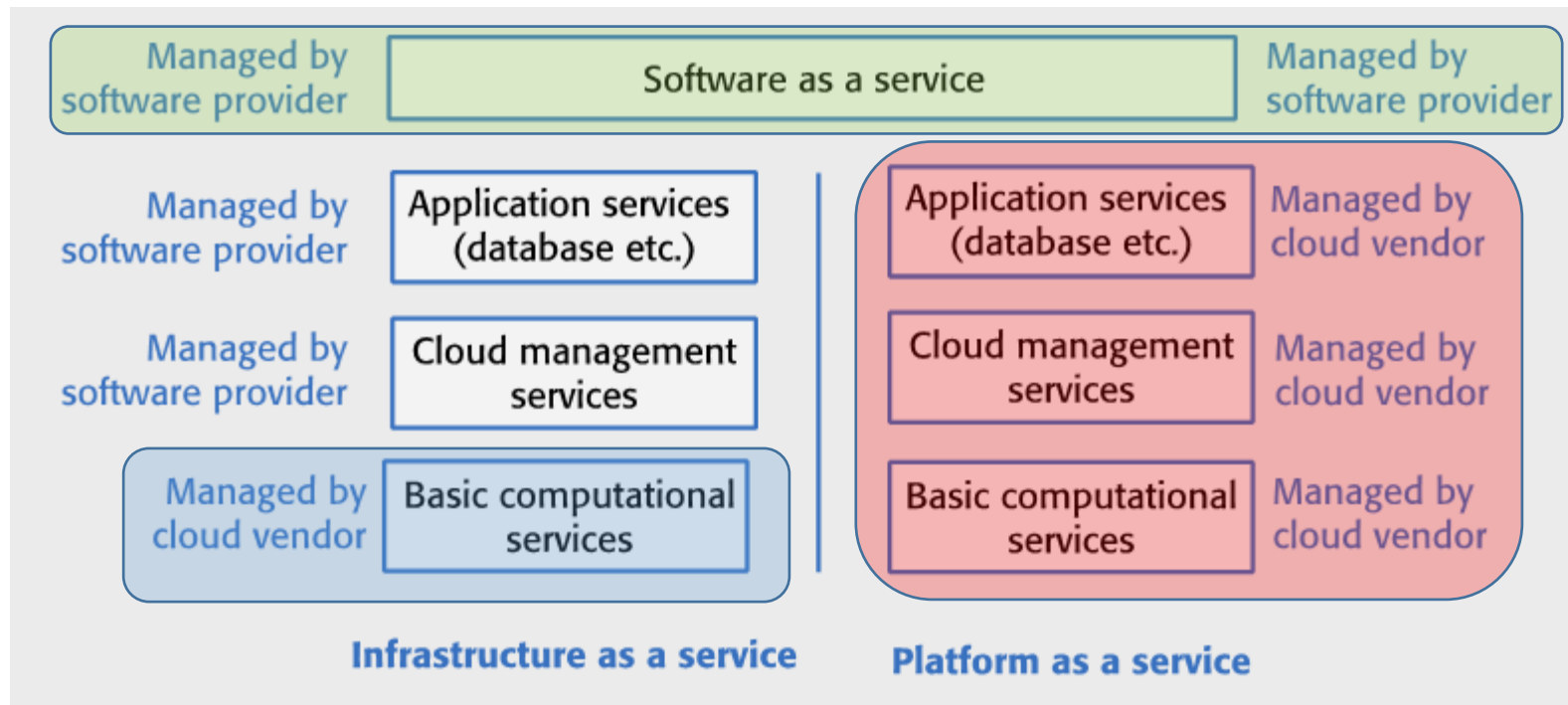
Storage

Servers

# Various things as a service



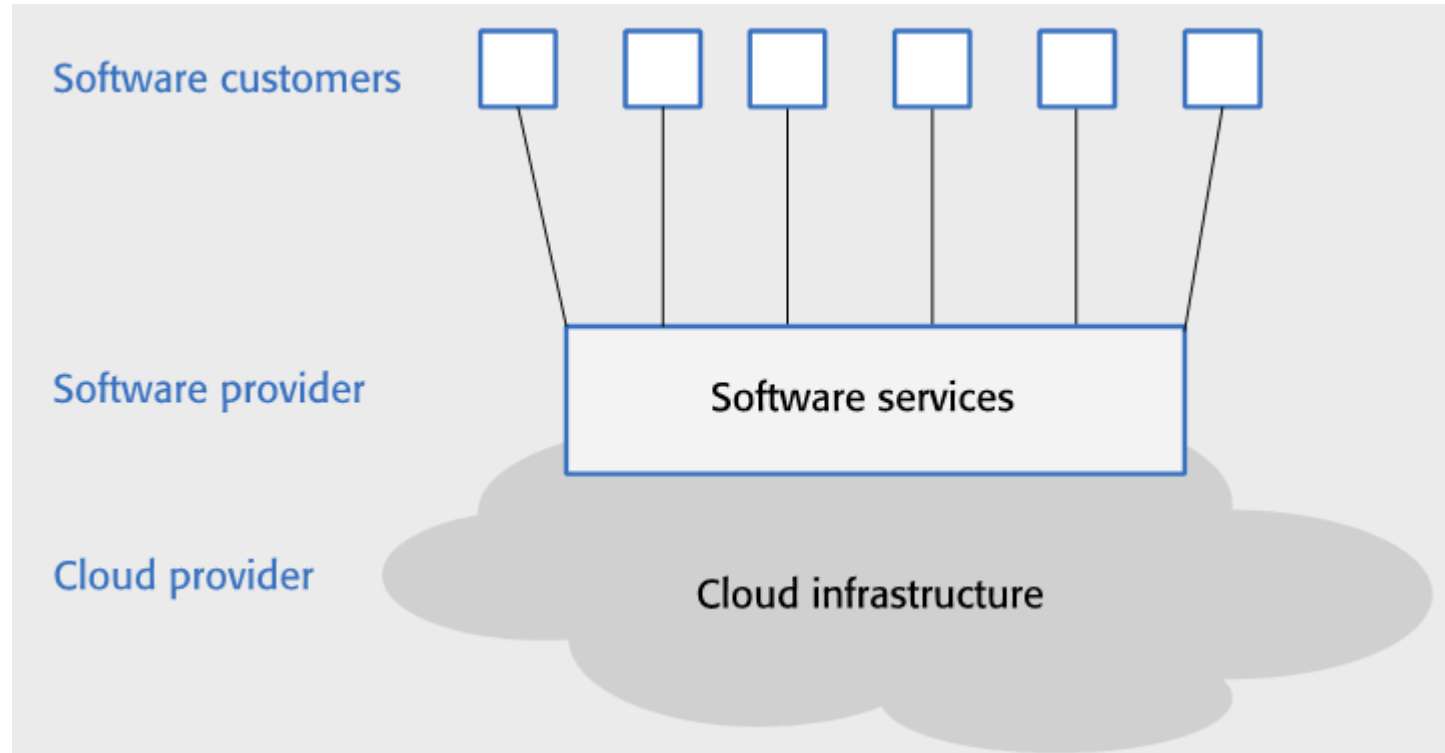
# Management responsibilities for IaaS and PaaS





- Increasingly, software products are being delivered as a service, rather than installed on the buyer's computers.
- If you deliver your software product as a service, you run the software on your servers, which you may rent from a cloud provider.
- Customers don't have to install software and they access the remote system through a web browser or dedicated mobile app.
- The payment model for software as a service is usually a subscription model.
  - Users pay a monthly fee to use the software rather than buy it outright.

# Software as a service



- **Cash flow:** Customers either pay a regular subscription or pay as they use the software
  - This means you have a regular cash flow, with payments throughout the year
  - You don't have a situation where you have a large cash injection when products are purchased but very little income between product releases.
- **Update management:** You are in control of updates to your product and all customers receive the update at the same time
  - You avoid the issue of several versions being simultaneously used and maintained
  - This reduces your costs and makes it easier to maintain a consistent software code base.
- **Continuous deployment:** You can deploy new versions of your software as soon as changes have been made and tested
  - This means you can fix bugs quickly so that your software reliability can continuously improve.

- **Payment flexibility**

- You can have several different payment options so that you can attract a wider range of customers
- Small companies or individuals need not be discouraged by having to pay large upfront software costs

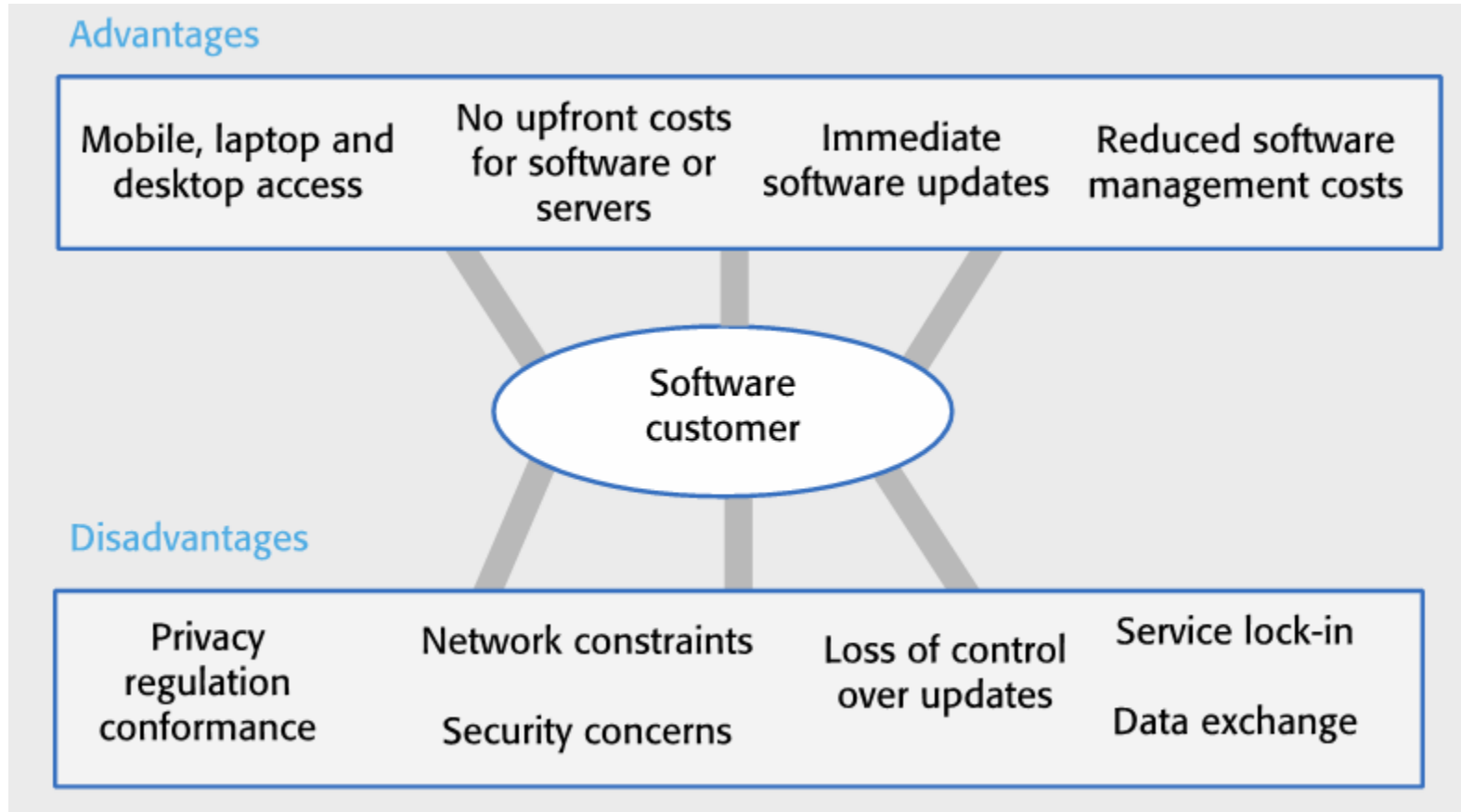
- **Try before you buy**

- You can make early free or low-cost versions of the software available quickly with the aim of getting customer feedback on bugs and how the product could be approved

- **Data collection**

- You can easily collect data on how the product is used and so identify areas for improvement
- You may also be able to collect customer data that allows you to market other products to these customers

# Advantages and disadvantages of SaaS for customers



- **Regulation**

- Some countries, such as EU countries, have strict laws on the storage of personal information
- These may be incompatible with the laws and regulations of the country where the SaaS provider is based
- If a SaaS provider cannot guarantee that their storage locations conform to the laws of the customer's country, businesses may be reluctant to use their product.

- **Data transfer**

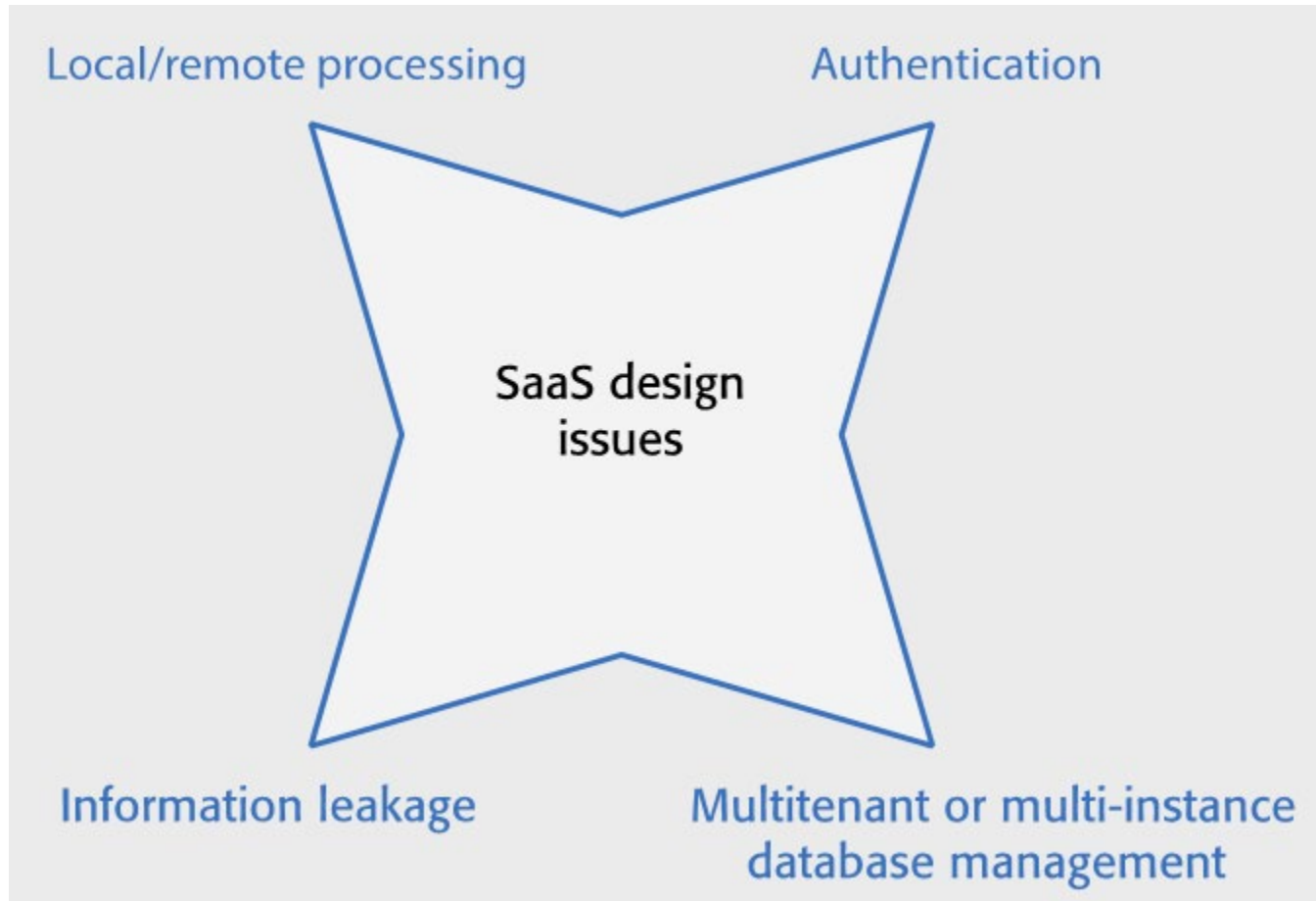
- If software use involves a lot of data transfer, the software response time may be limited by the network speed
- This is a problem for individuals and smaller companies who can't afford to pay for very high-speed network connections

- **Data security**

- Companies dealing with sensitive information may be unwilling to hand over control of their data to an external software provider
- As we have seen from a number of high-profile cases, even large cloud providers have had security breaches
- You can't assume that they always provide better security than the customer's own servers.

- **Data exchange**

- If you need to exchange data between a cloud service and other services or local software applications, this can be difficult unless the cloud service provides an API that is accessible for external use





- **Local/remote processing**

- A software product may be designed so that some features are executed locally in the user's browser or mobile app and some on a remote server
- Local execution reduces network traffic and so increases user response speed; this is useful when users have a slow network connection.
- Local processing increases the electrical power needed to run the system

- **Authentication**

- If you set up your own authentication system, users have to remember another set of authentication credentials
- Many systems allow authentication using the user's Google, Facebook or LinkedIn credentials
- For business products, you may need to set up a federated authentication system (SSO), which delegates authentication to the business where the user works

- **Information leakage**

- If you have multiple users from multiple organizations, a security risk is that information leaks from one organization to another.
- There are a number of different ways that this can happen, so you need to be very careful in designing your security system to avoid this.

- **Multi-tenant and multi-instance systems**

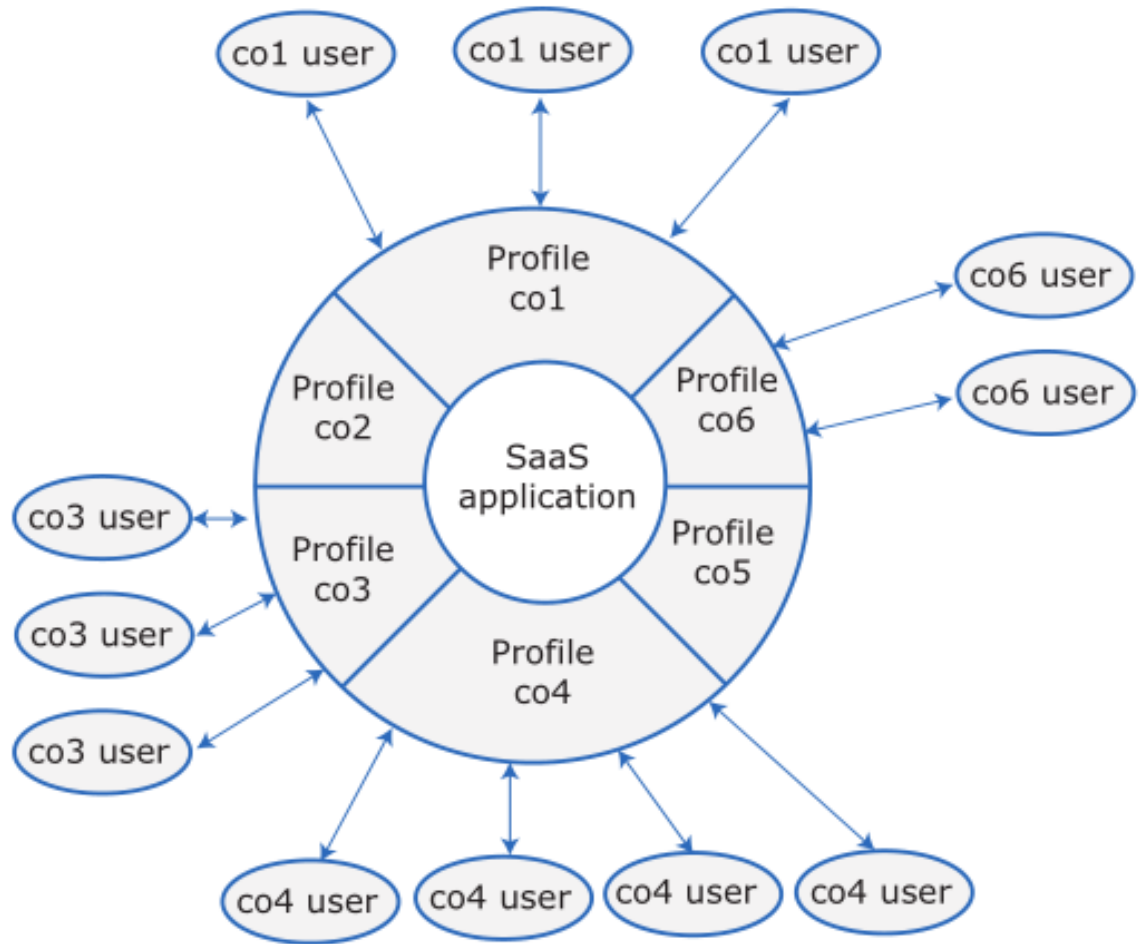
- In a multi-tenant system, all customers are served by a single instance of the system and a multitenant database.
- In a multi-instance system, a separate copy of the system and database is made available for each user

- A multi-tenant database is partitioned so that customer companies have their own space and can store and access their own data
  - There is a single database schema, defined by the SaaS provider, that is shared by all the system's users
  - Items in the database are tagged with a tenant identifier, representing a company that has stored data in the system
  - The database access software uses this tenant identifier to provide 'logical isolation', which means that users seem to be working with their own database

# Multi-tenant databases: pro and cons

Advantages	Disadvantages
<b>Resource utilization</b> The SaaS provider has control of all the resources used by the software and can optimize the software to make effective use of these resources.	<b>Inflexibility</b> Customers must all use the same database schema with limited scope for adapting this schema to individual needs. I explain possible database adaptations later in this section.
<b>Security</b> Multi-tenant databases have to be designed for security because the data for all customers are held in the same database. They are, therefore, likely to have fewer security vulnerabilities than standard database products. Security management is also simplified as there is only a single copy of the database software to be patched if a security vulnerability is discovered.	<b>Security</b> As data for all customers are maintained in the same database, there is a theoretical possibility that data will leak from one customer to another. In fact, there are very few instances of this happening. More seriously, perhaps, if there is a database security breach, then it affects all customers.
<b>Update management</b> It is easier to update a single instance of software rather than multiple instances. Updates are delivered to all customers at the same time so all use the latest version of the software.	<b>Complexity</b> Multi-tenant systems are usually more complex than multi-instance systems because of the need to manage many users. There is, therefore, an increased likelihood of bugs in the database software.

- Authentication
- Branding
- Access control
- Data scheme



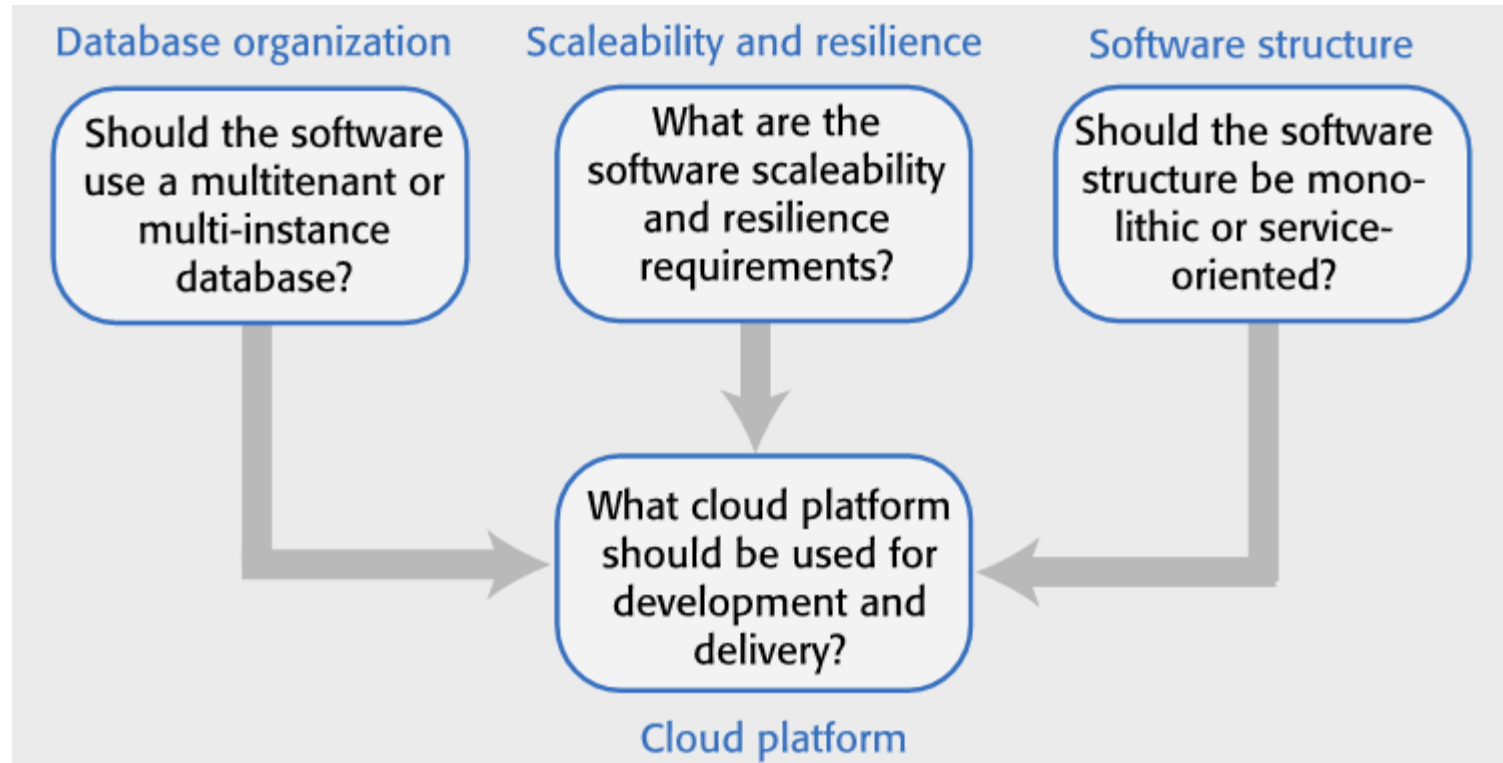
- Information from all customers is stored in the same database in a multi-multi-tenant system
- Key security issues are multilevel access control and encryption
  - Multilevel access control means that access to data must be controlled at the organizational level and the individual level
  - You need to have organizational-level access control to ensure that any database operations only act on that organization's data
  - Encryption of data in a multitenant database reassures corporate users that their data cannot be viewed by people from other companies if some kind of system failure occurs

- Multi-instance systems are SaaS systems where each customer has its own system that is adapted to its needs, including its own database and security controls
- Multi-instance, cloud-based systems are conceptually simpler than multi-tenant systems and avoid security concerns such as data leakage from one organization to another
- Two types: vm-based or container-based



# Multi-instance databases: pro and cons

Advantages	Disadvantages
<b>Flexibility</b> Each instance of the software can be tailored and adapted to a customer's needs. Customers may use completely different database schemas and it is straightforward to transfer data from a customer database to the product database.	<b>Cost</b> It is more expensive to use multi-instance systems because of the costs of renting many VMs in the cloud and the costs of managing multiple systems. Because of the slow startup time, VMs may have to be rented and kept running continuously, even if there is very little demand for the service.
<b>Security</b> Each customer has its own database so there is no possibility of data leakage from one customer to another.	<b>Update management</b> Many instances have to be updated so updates are more complex, especially if instances have been tailored to specific customer needs.
<b>Scalability</b> Instances of the system can be scaled according to the needs of individual customers. For example, some customers may require more powerful servers than others.	
<b>Resilience</b> If a software failure occurs, this will probably affect only a single customer. Other customers can continue working as normal.	

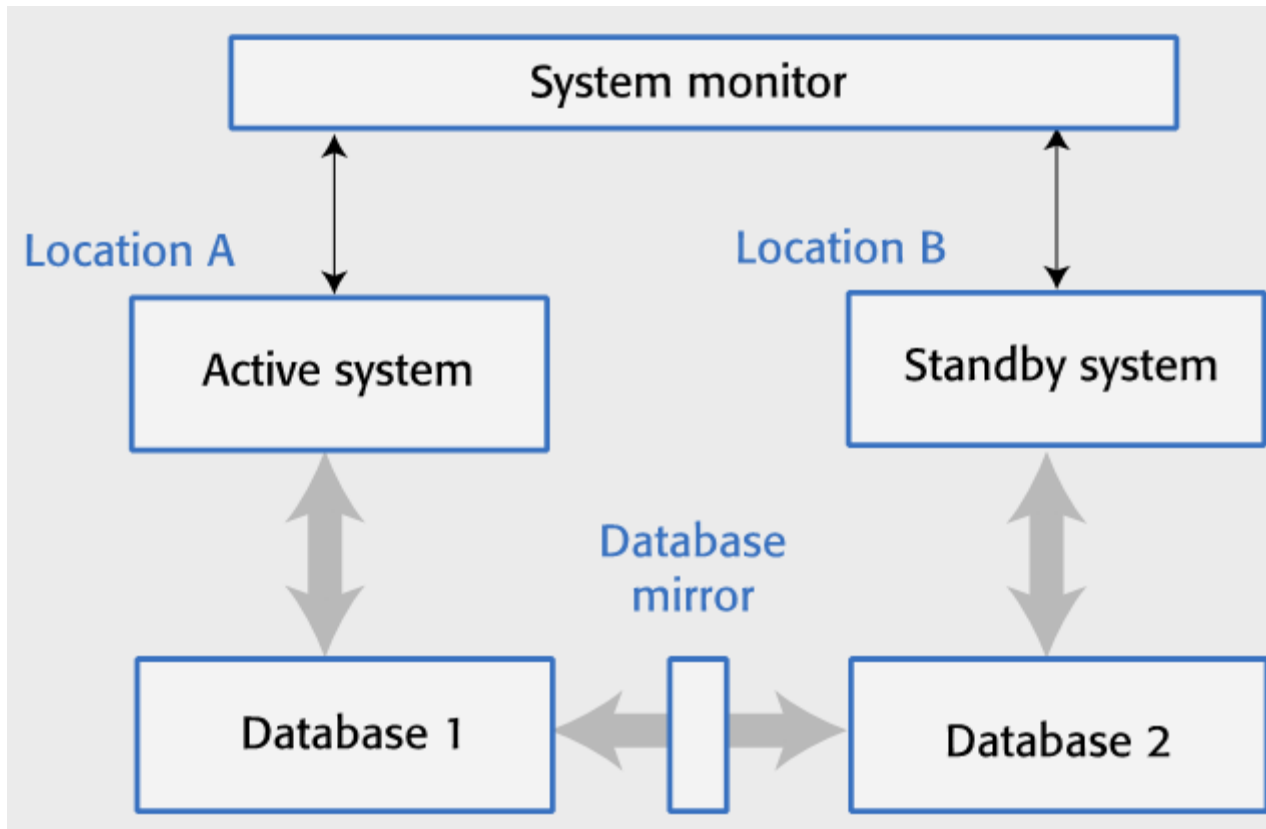


# Questions to ask when choosing a database organization

Factor	Key questions
Target customers	Do customers require different database schemas and database personalization? Do customers have security concerns about database sharing? If so, use a multi-instance database.
Transaction requirements	Is it critical that your products support ACID transactions where the data are guaranteed to be consistent at all times? If so, use a multi-tenant database or a VM-based multi-instance database.
Database size and connectivity	How large is the typical database used by customers? How many relationships are there between database items? A multi-tenant model is usually best for very large databases, as you can focus effort on optimizing performance.
Database interoperability	Will customers wish to transfer information from existing databases? What are the differences in schemas between these and a possible multi-tenant database? What software support will they expect to do the data transfer? If customers have many different schemas, a multi-instance database should be used.
System structure	Are you using a service-oriented architecture for your system? Can customer databases be split into a set of individual service databases? If so, use containerized, multi-instance databases.

- The scalability of a system reflects its ability to adapt automatically to changes in the load on that system.
- The resilience of a system reflects its ability to continue to deliver critical services in the event of system failure or malicious system use.
- You achieve scalability in a system by making it possible to add new virtual servers (scaling out) or increase the power of a system server (scaling up) in response to increasing load.
  - In cloud-based systems, scaling-out rather than scaling up is the normal approach used. Your software has to be organized so that individual software components can be replicated and run in parallel.
- To achieve resilience, you need to be able to restart your software quickly after a hardware or software failure.

- Using a standby system to provide resilience



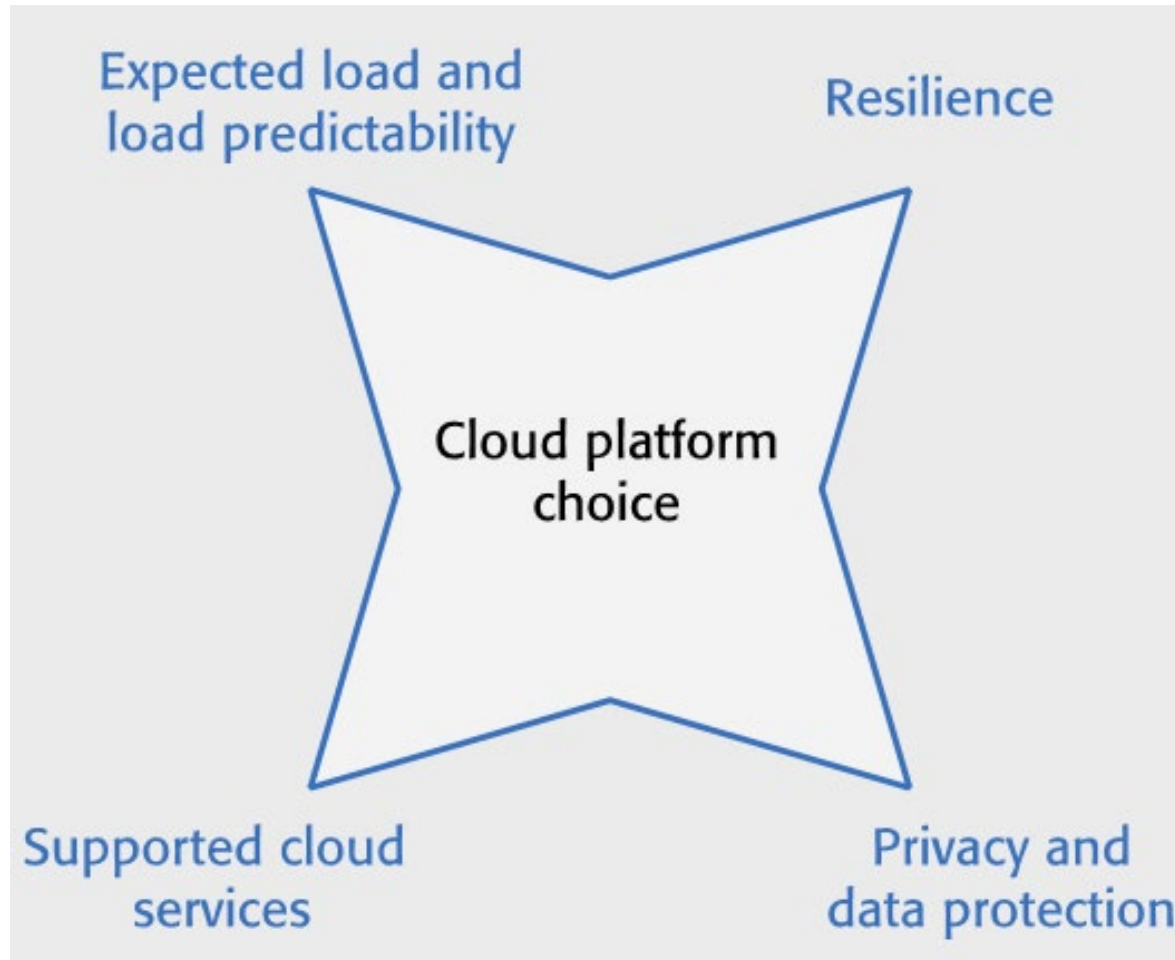
- Resilience relies on redundancy
  - Replicas of the software and data are maintained in different locations.
  - Database updates are mirrored so that the standby database is a working copy of the operational database.
  - A system monitor continually checks the system status. It can switch to the standby system automatically if the operating system fails
- You should use redundant virtual servers that are not hosted on the same physical computer and locate servers in different locations
  - Ideally, these servers should be located in different data centers.
  - If a physical server fails or if there is a wider data center failure, then the operation can be switched automatically to the software copies elsewhere

- A monolithic approach to software: distribution across multiple servers running large software components
- The alternative to a monolithic approach to software organization is a service-oriented approach
  - The system is decomposed into fine-grain, stateless services
  - Because it is stateless, each service is independent and can be replicated, distributed, and migrated from one server to another
  - The service-oriented approach is particularly suitable for cloud-based software, with services deployed in containers
  - Next chapter

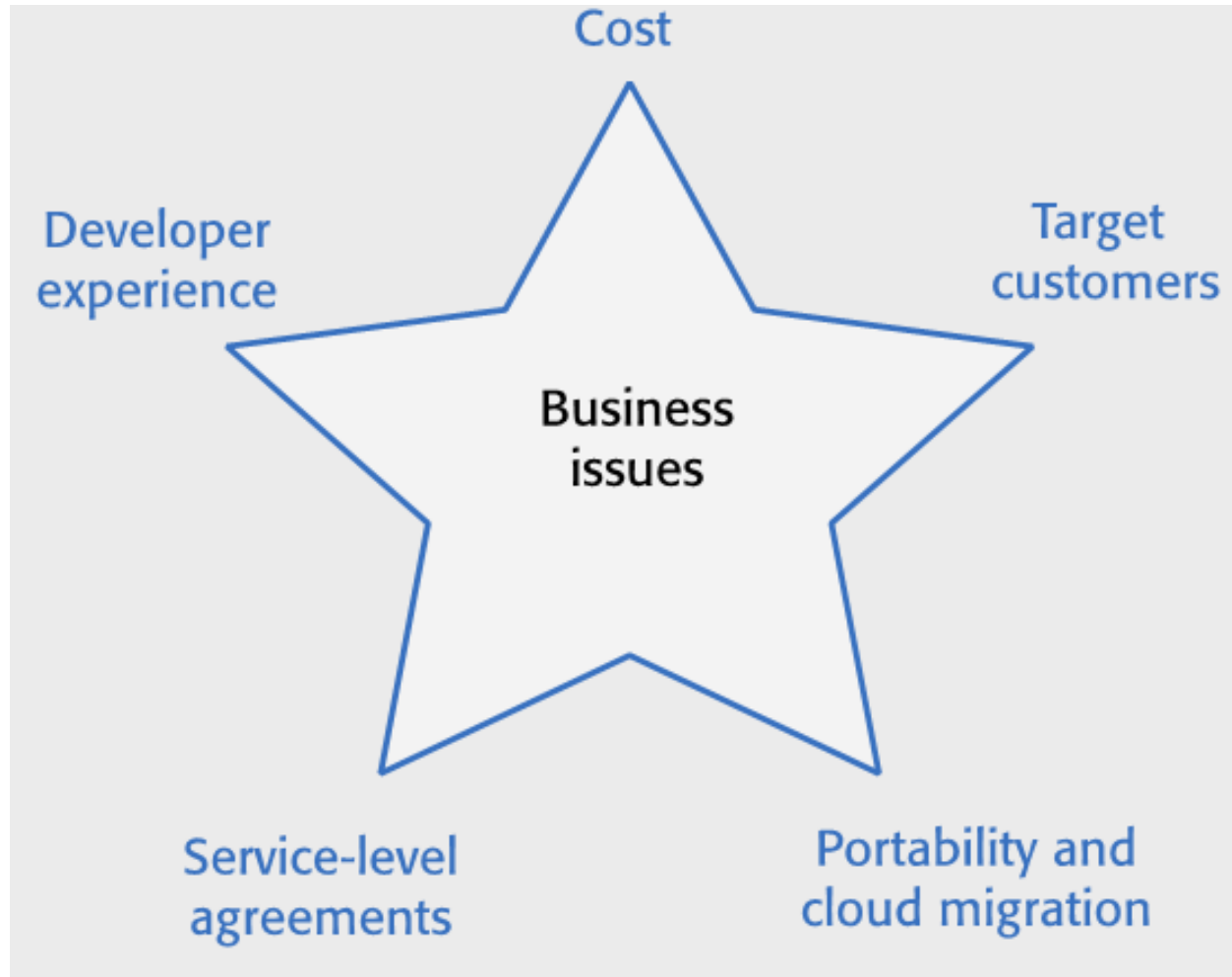
- Cloud platforms include general-purpose clouds such as Amazon Web Services or lesser-known platforms oriented around a specific application, such as the SAP Cloud Platform
- There is no 'best' platform and you should choose a cloud provider based on your background and experience, the type of product that you are developing, and the expectations of your customers
- You need to consider both technical issues and business issues when choosing a cloud platform for your product



# Technical issues in cloud platform choice



# Business issues in cloud platform choice



- The cloud is made up of a large number of virtual servers that you can rent for your own use
  - Remote access; pay for what you need
- Virtualization is a technology that allows multiple server instances to be run on the same physical computer
- Virtual machines are physical server replicas on which you run your own operating system, technology stack and applications
- Containers are a lightweight virtualization technology that allows rapid replication and deployment of virtual servers
- A fundamental feature of the cloud is that 'everything' can be delivered as a service and accessed over the internet

- Scalability, elasticity, resilience
- IaaS, PaaS, SaaS
- Multitenant vs multi-instance cloud platforms