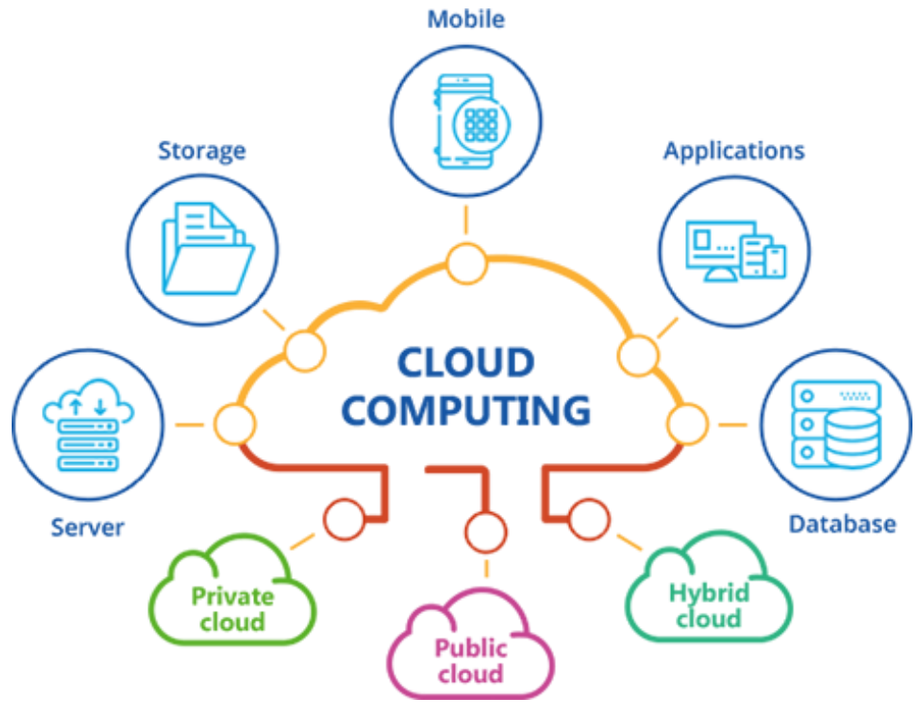
Chapter 5: Cloud-based software

CSC 4307

The cloud



The cloud is made up of very large number of remote servers that are offered for **rent by companies** that own these servers.

Cloud is the leveraging pools of **computing resources** to minimize costs and maximize compute efficiency.

Cloud Characteristics

- On-demand
- Self serviced
- Ubiquitous access
- Resource pooling
- Rapid elasticity
- Pay per use

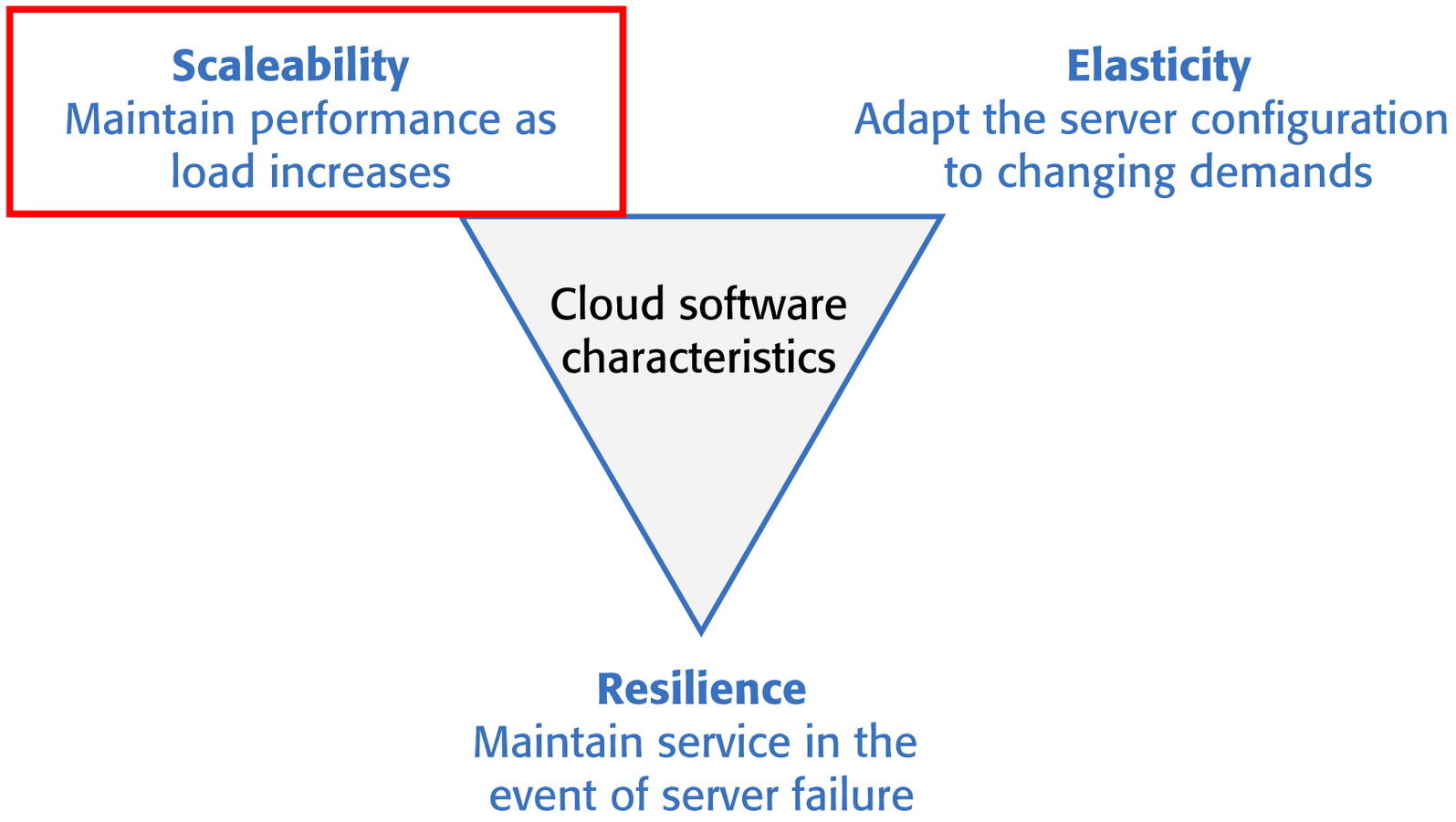
Cloud Deployment & Delivery modes

Deployment:
Private
public
hybrid,
Multi-cloud
community

Delivery :
PaaS, IaaS, SaaS, CaaS, FaaS

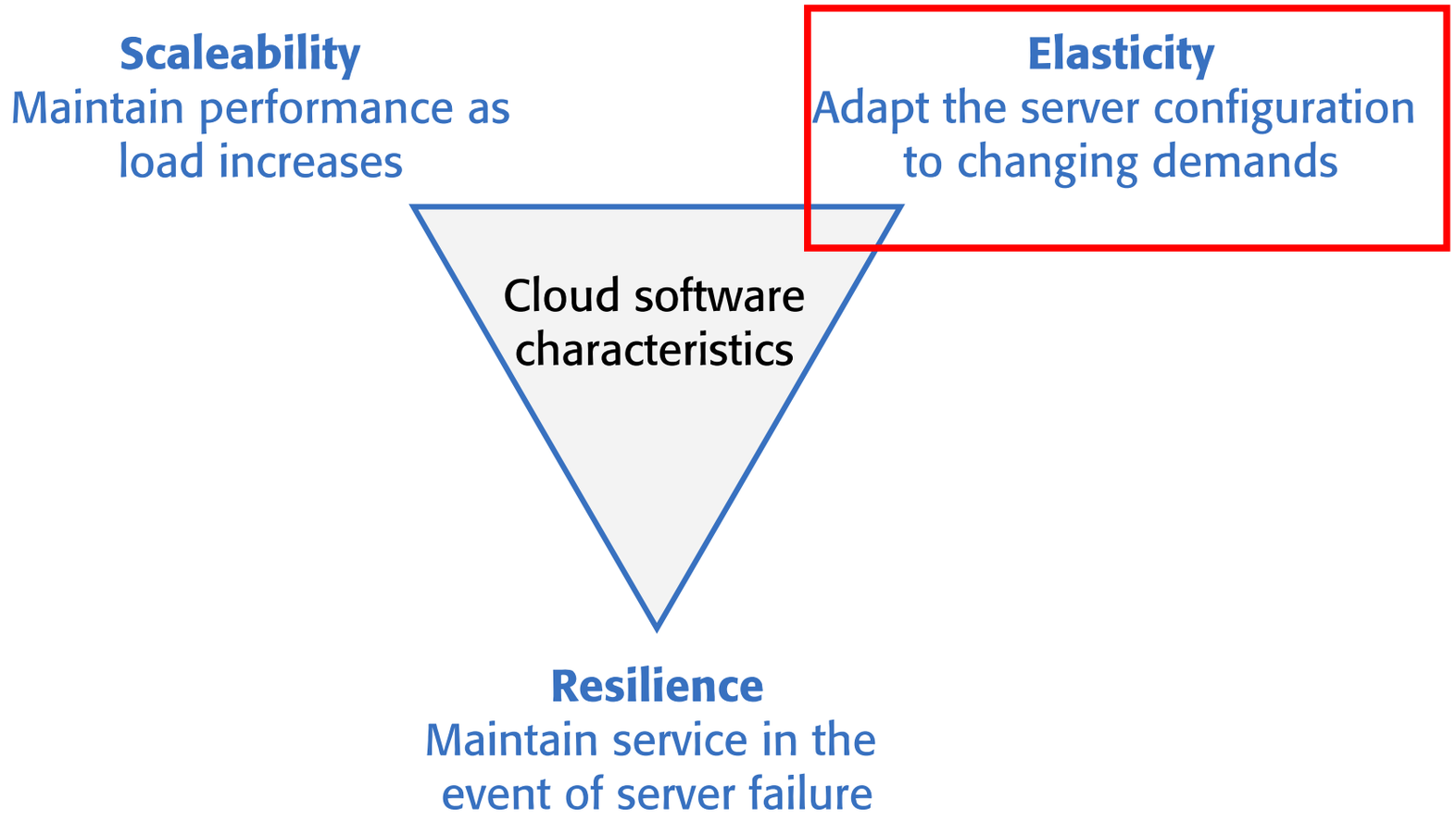
Scalability, Elasticity and Resilience

The ability of your software to cope with increasing numbers of users.

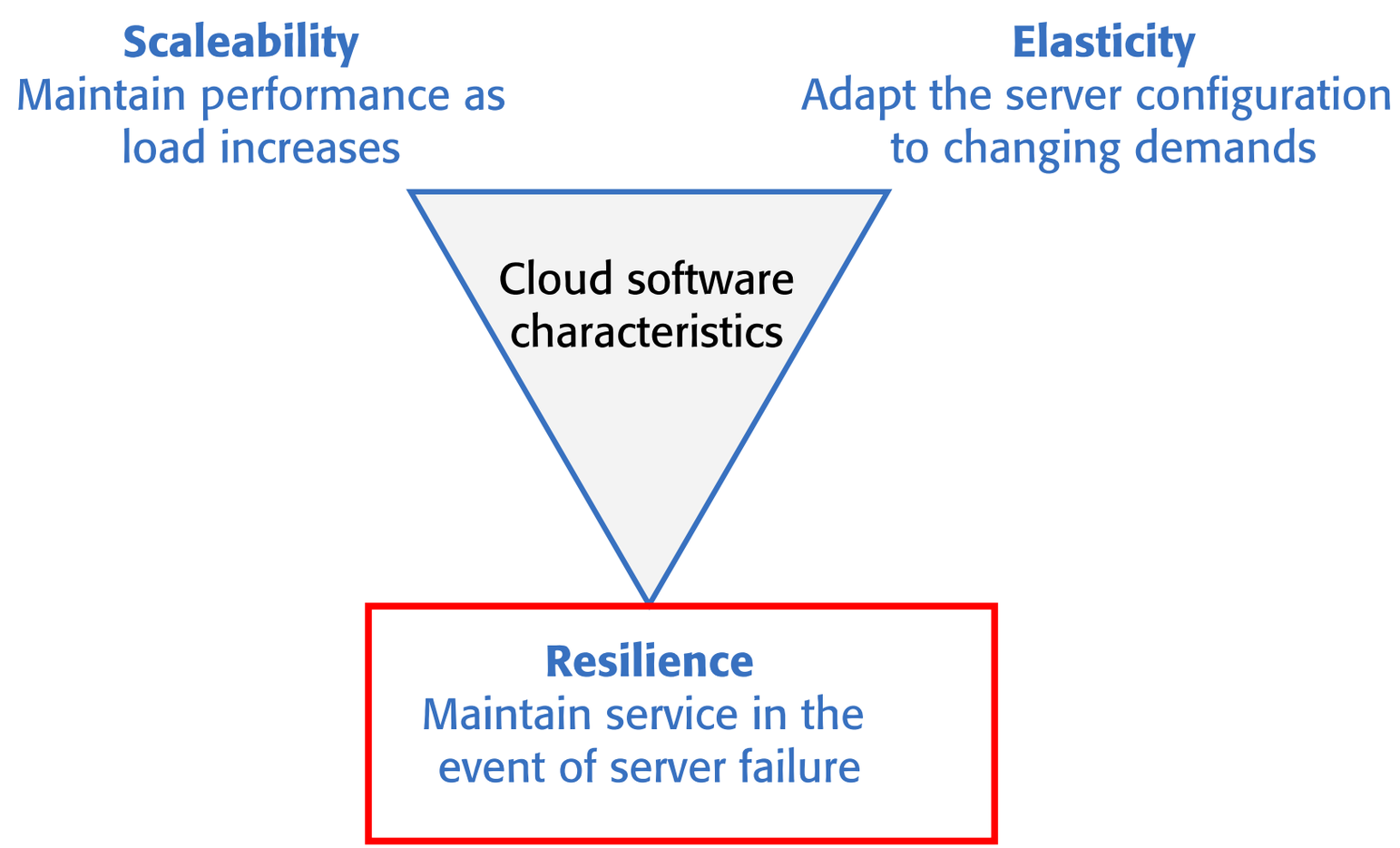


Scalability, elasticity and resilience

related to scalability but also allows for scaling-down as well as scaling-up.



Scalability, elasticity and resilience



Resilience means that you can design your software architecture to tolerate server failures.

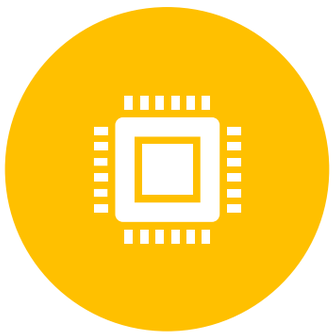
Benefits of using the cloud for software development

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.



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.

Virtualization & Containerization

Both containers and virtual machines (VMs) are software technologies that create self-contained virtual packages. Beyond that commonality, they differ in their operations, characteristics and use cases.

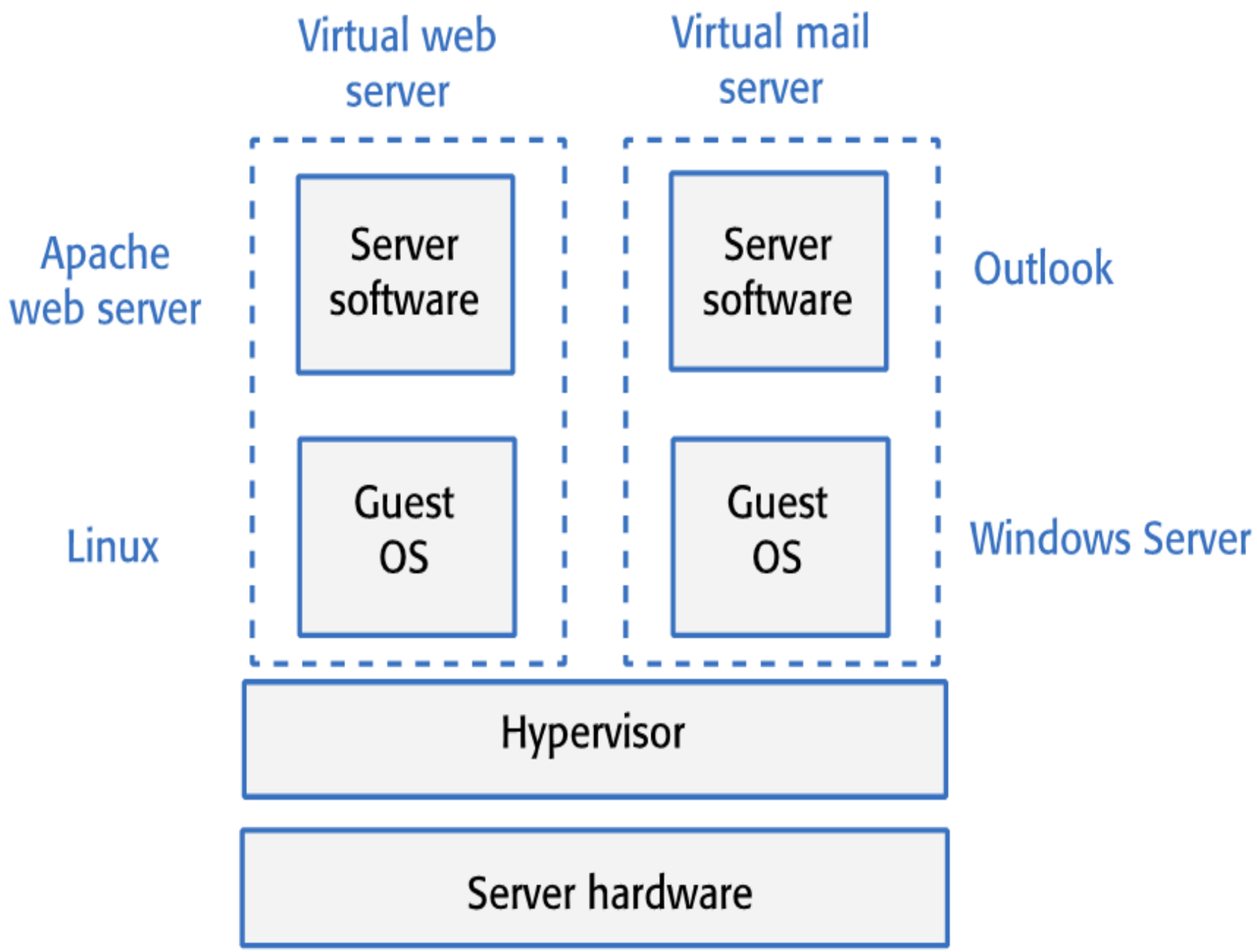
Virtual Cloud Servers

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.

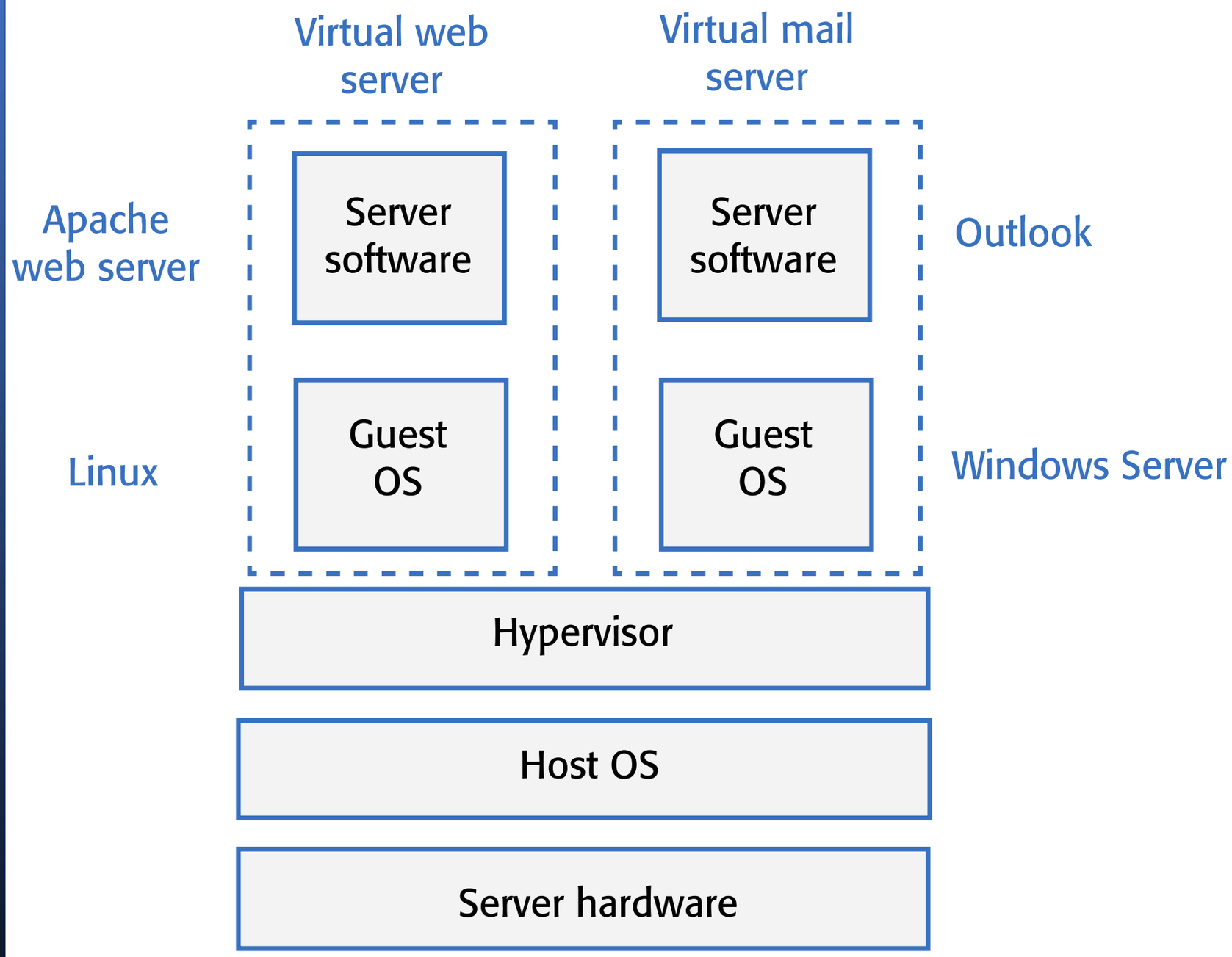
Virtual machines (VMs), running on physical server hardware, can be used to implement virtual servers. **A hypervisor** provides hardware emulation that simulates the operation of the underlying hardware.

Implementing a virtual server as a virtual machine



Hypervisor Type 1 / Native /Bare Metal

Implementing a virtual server as a virtual machine

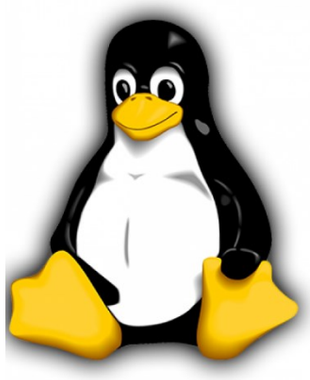


Hypervisor Type 2/ hosted

Virtualization tools

vmware

Microsoft
Hyper-V



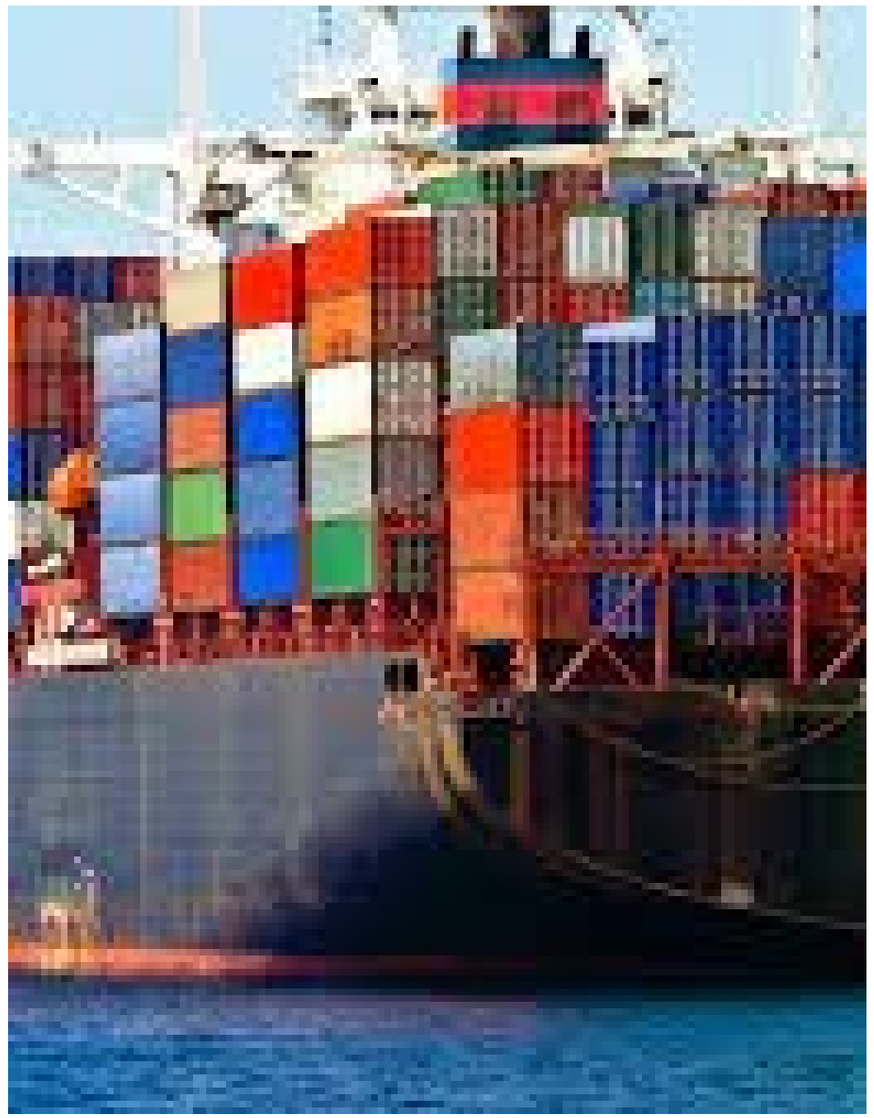
Linux
KVM

citrix

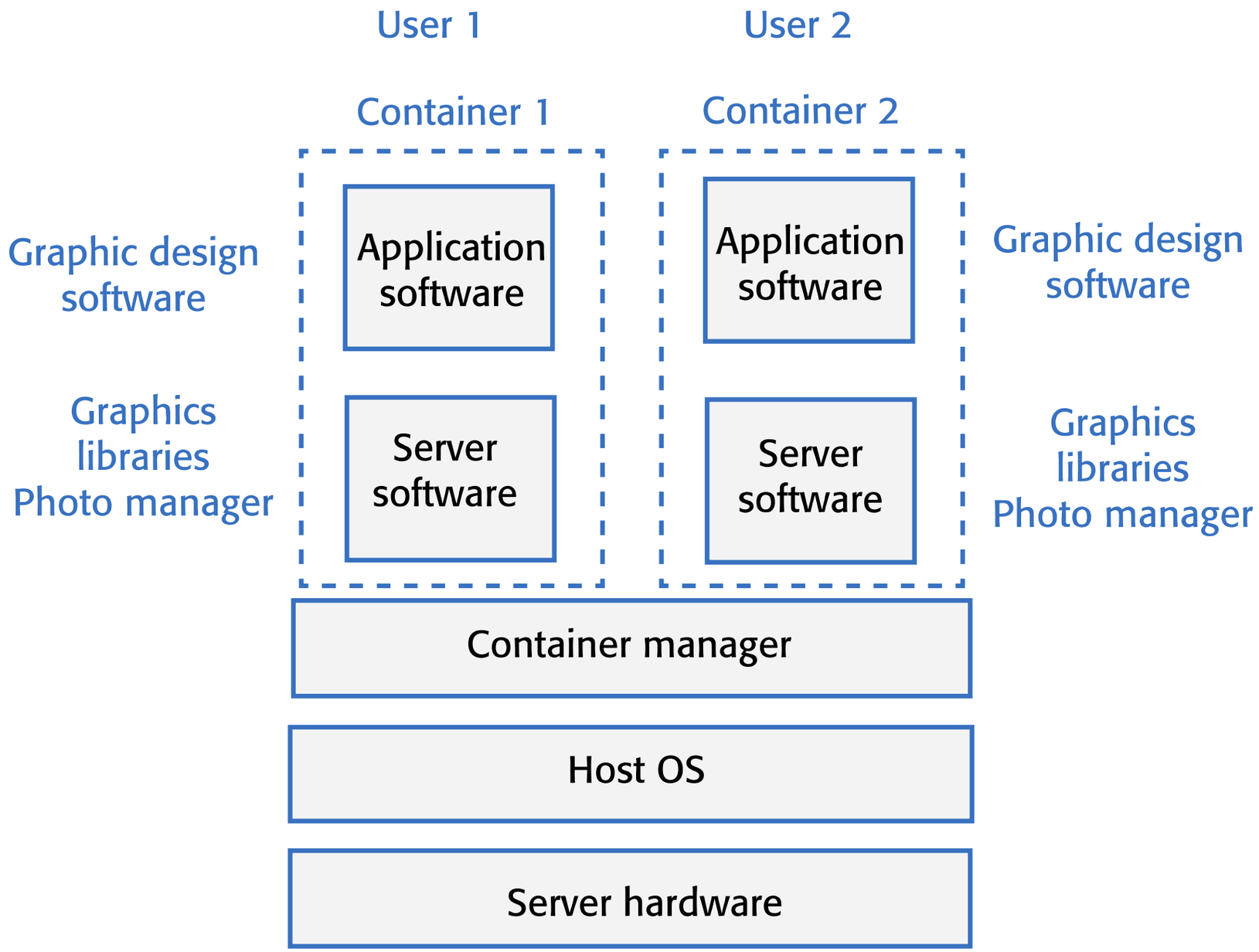
Container-based virtualization

- When running a cloud-based system with many instances of applications or services, these all use the same OS, you can use a simpler virtualization technology called ‘containers’.

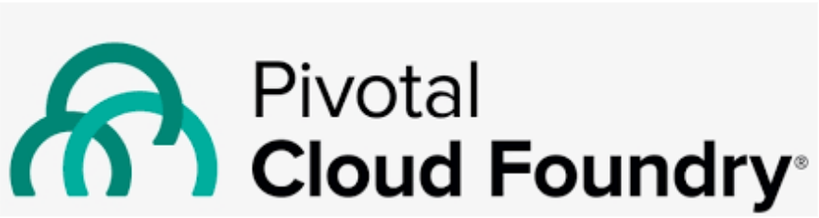
- Containers are **an operating system virtualization** technology that allows independent servers to share a single operating system.



Using containers to provide isolated services



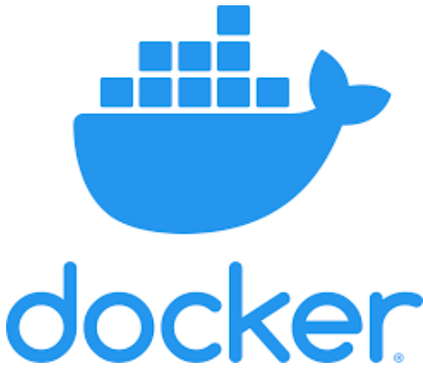
Containerization tools



VMs vs Containers

VALUE	CONTAINER	VM
Boot Speed	X	
size	X	
Maturity		X
Security		X
Ease of patching	X	
Developer Agility	X	

Docker



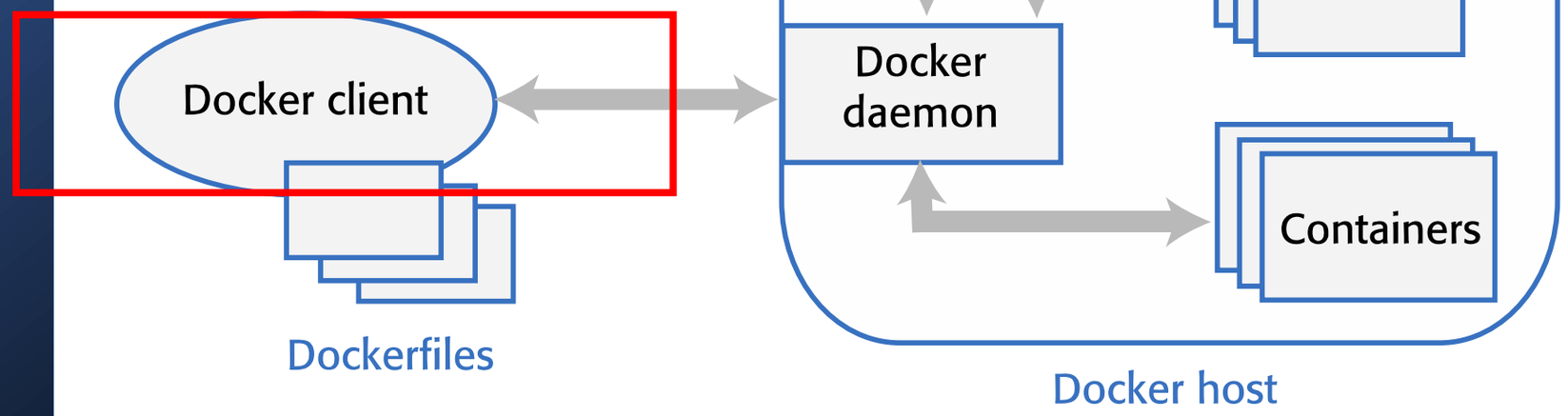
- An open-source project called Docker provided a standard means of **container management** that is fast and easy to use.

- Docker is a container management system that allows users to **define the software to be included in a container** as a Docker image.

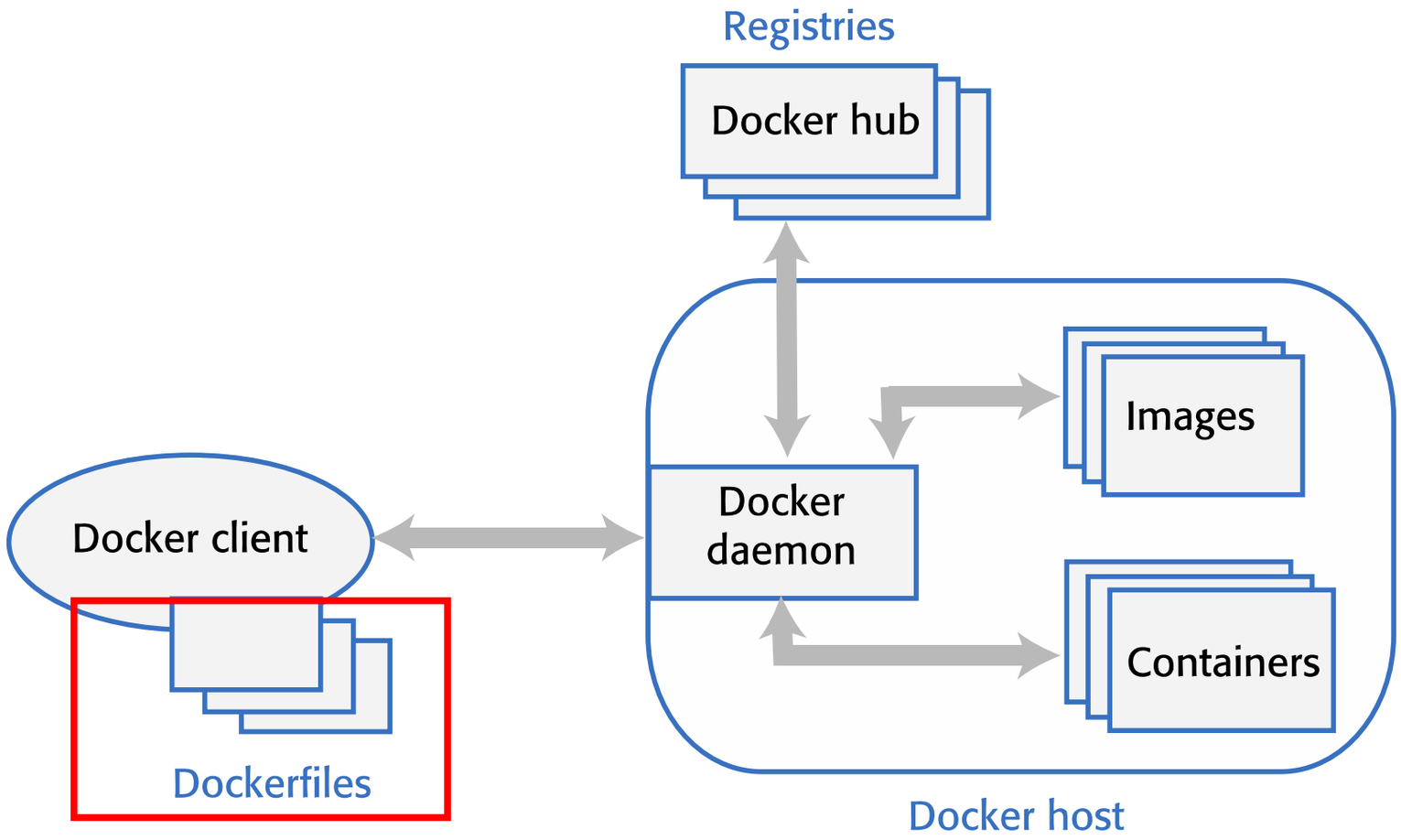
- It also **includes a run-time system that can create and manage containers** using these Docker images.

The Docker container system

Used by developers and system managers to define and control containers

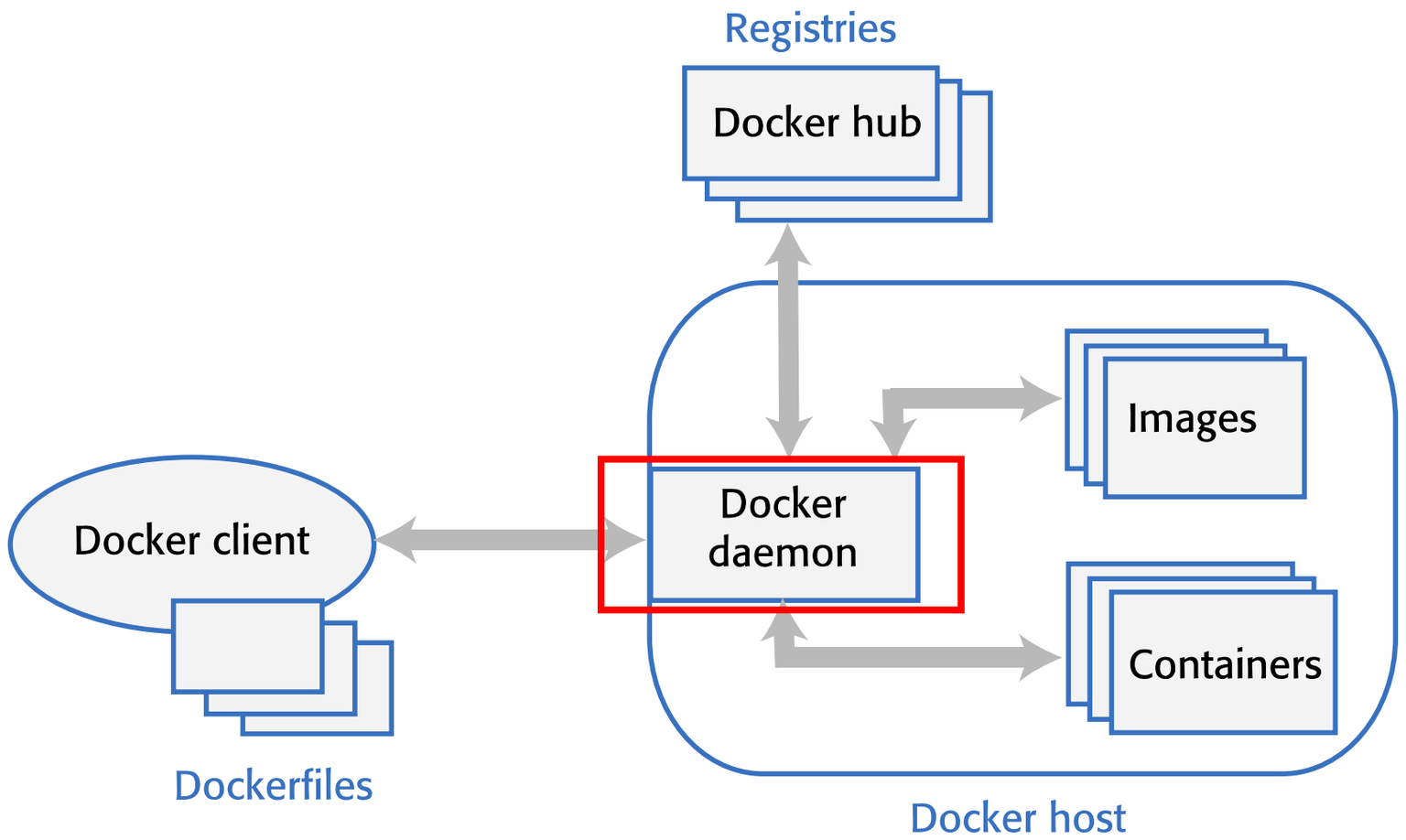


The Docker container system



Define runnable applications (images) as a series of setup commands that specify the software to be included in a container. Each container must be defined by an associated Dockerfile.

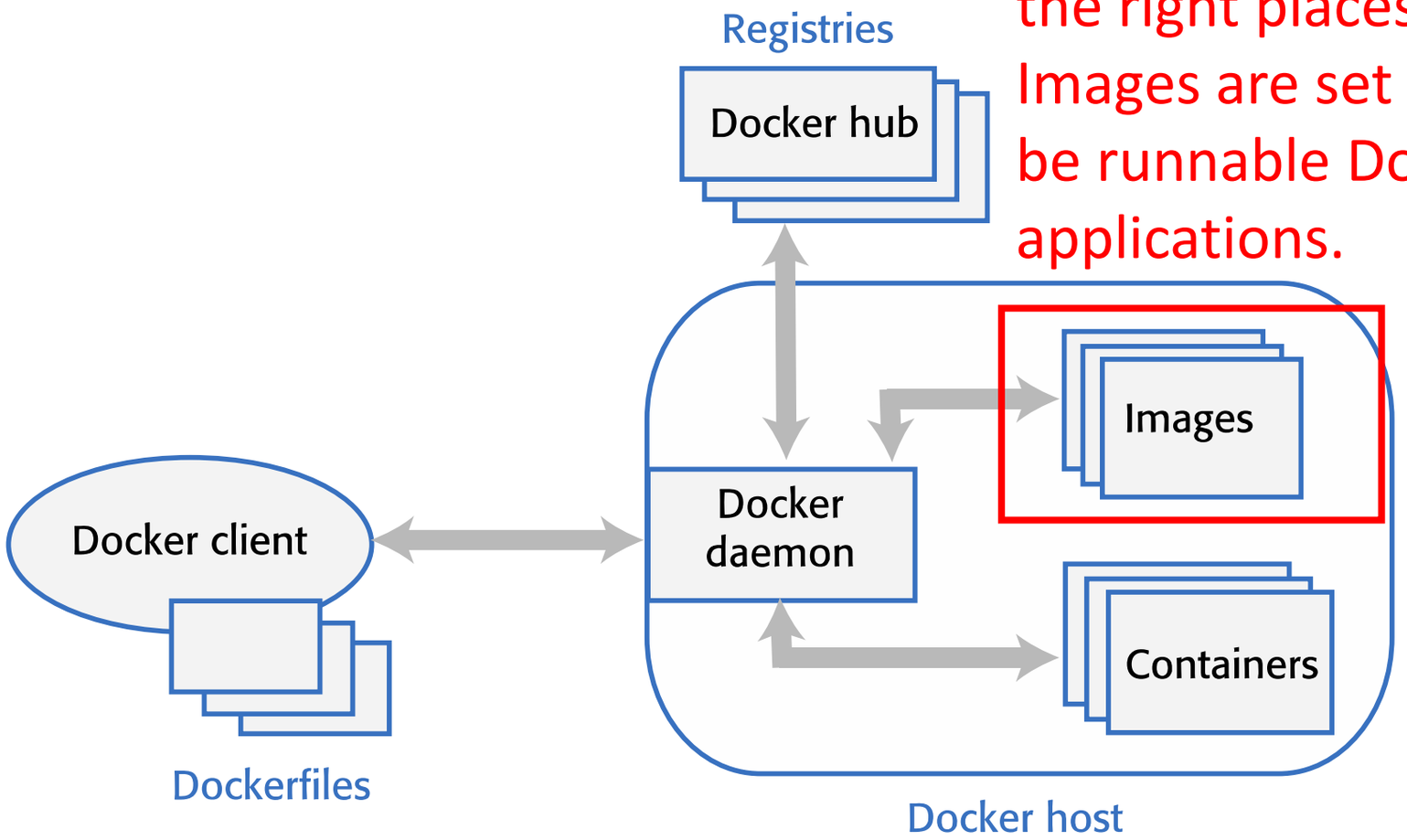
The Docker container system



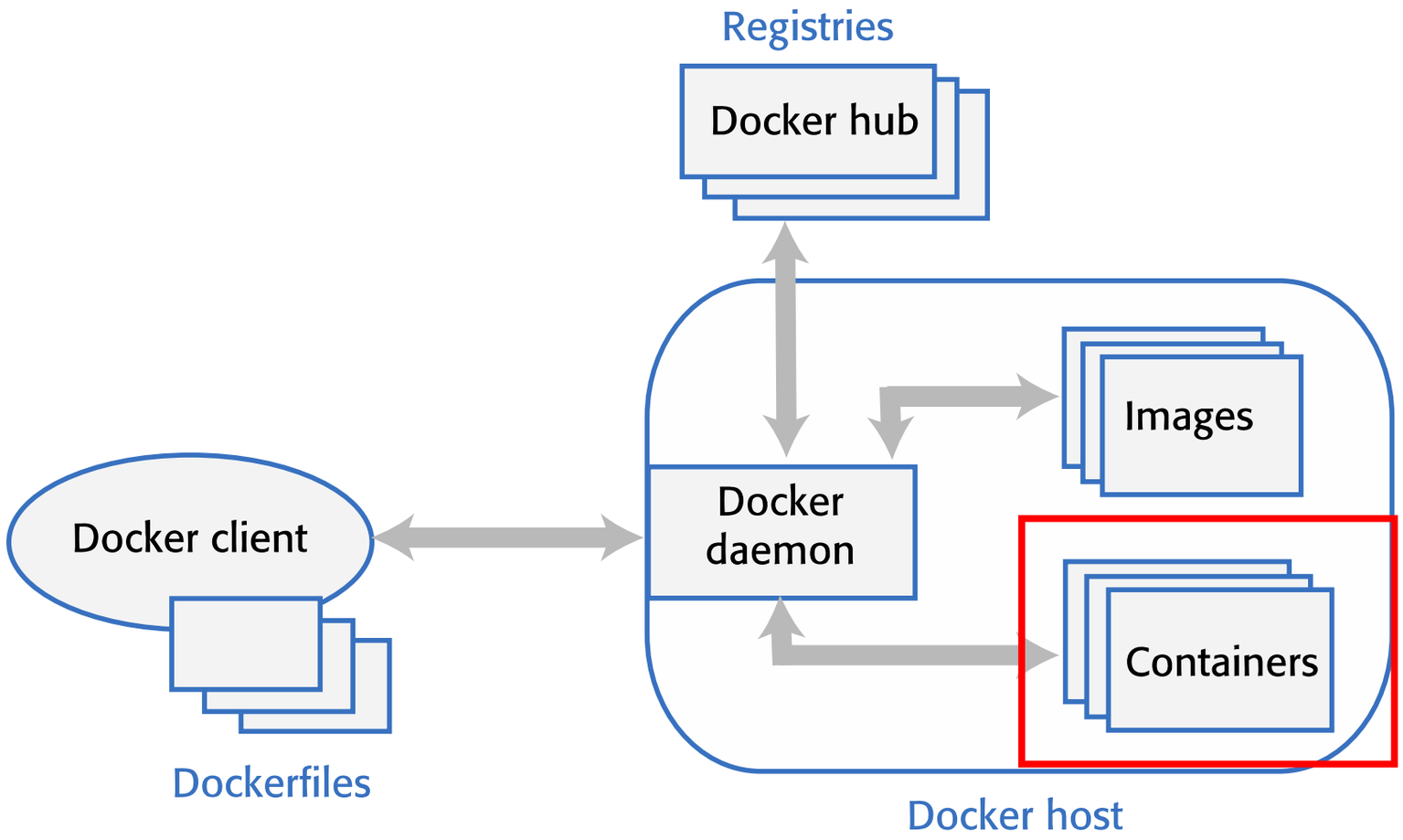
Runs on a host server and is used to setup, start, stop, and monitor containers, as well as building and managing local images.

The Docker container system

A Dockerfile is interpreted to create a Docker image, which is a set of directories with the specified software and data installed in the right places. Images are set up to be runnable Docker applications.



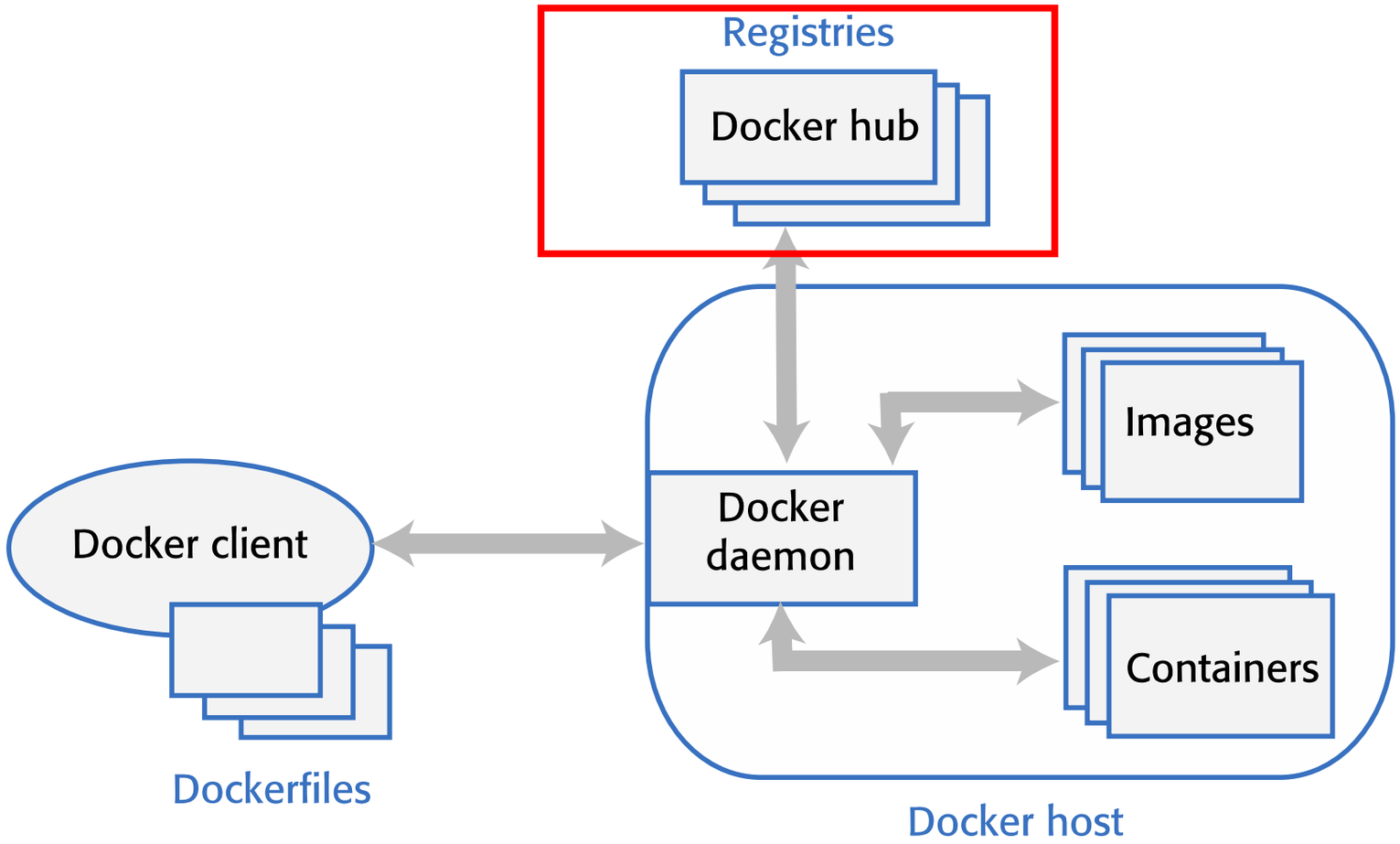
The Docker container system



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 Docker container system

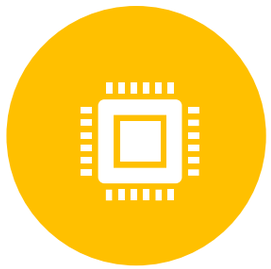
A registry of images that has been created. These may be reused to setup containers or as a starting point for defining new images.



Benefits of containers

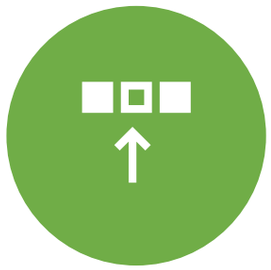
No software dependencies.

Increased portability across clouds.



Less System Resources

Greater Efficiency and Scalability



Support the development of service-oriented architectures.

Simplify the adoption of DevOps.

Homework

- Install VirtualBox and VirtualBox Extension Pack, Create a VM, Install Ubuntu 22.04 . Screenshot your machine CPU and memory usage before and after running VM
- Install docker, run ubuntu 22.04 Docker container. Screenshot your machine CPU and memory usage before and after running the container

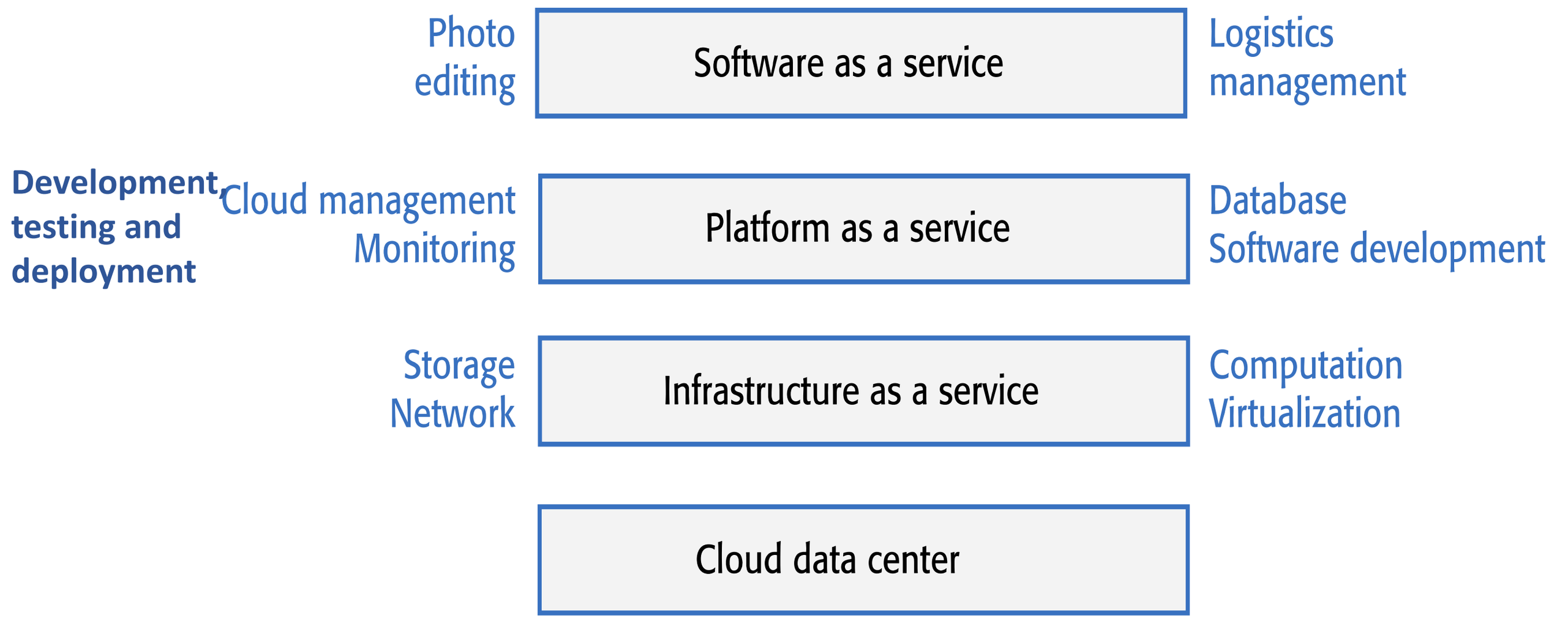
Deadline : November 8.

Lab 1 & 2 report deadline : November 13.

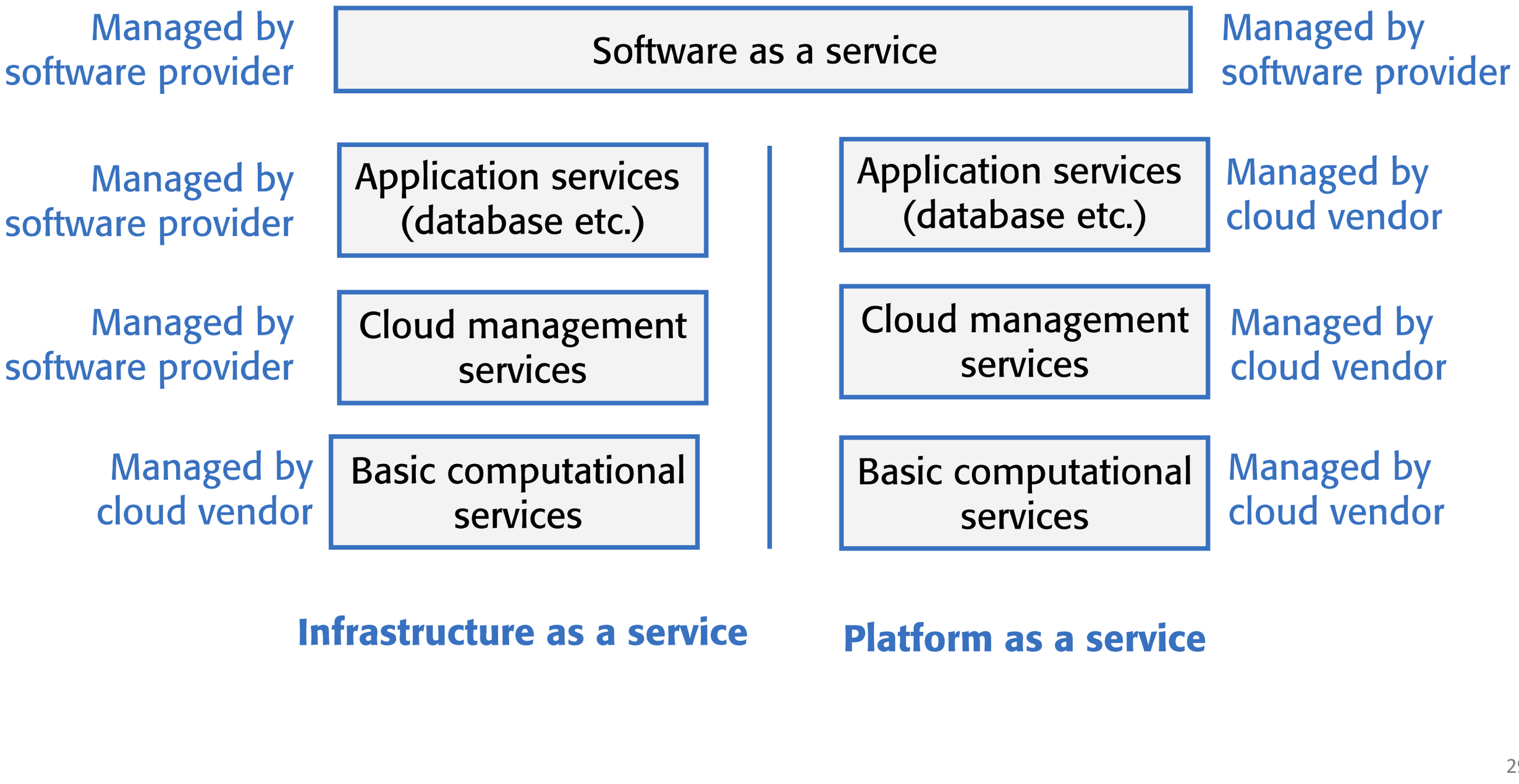
Everything as a Service

Everything as a service or Anything as a Service (EaaS, XaaS, *aaS) is **a concept of being able to call up re-usable, fine-grained software components across a network**.

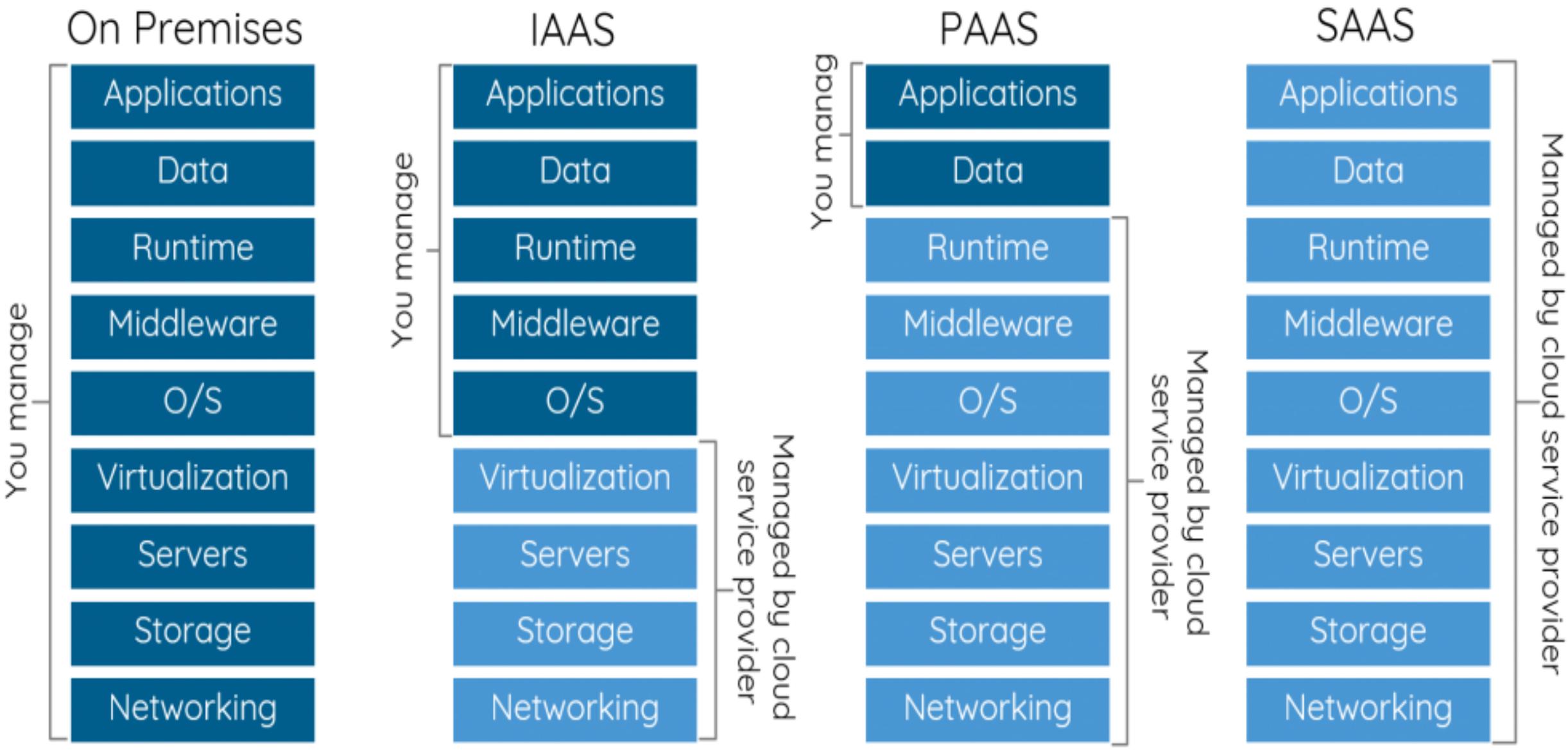
Everything as a service

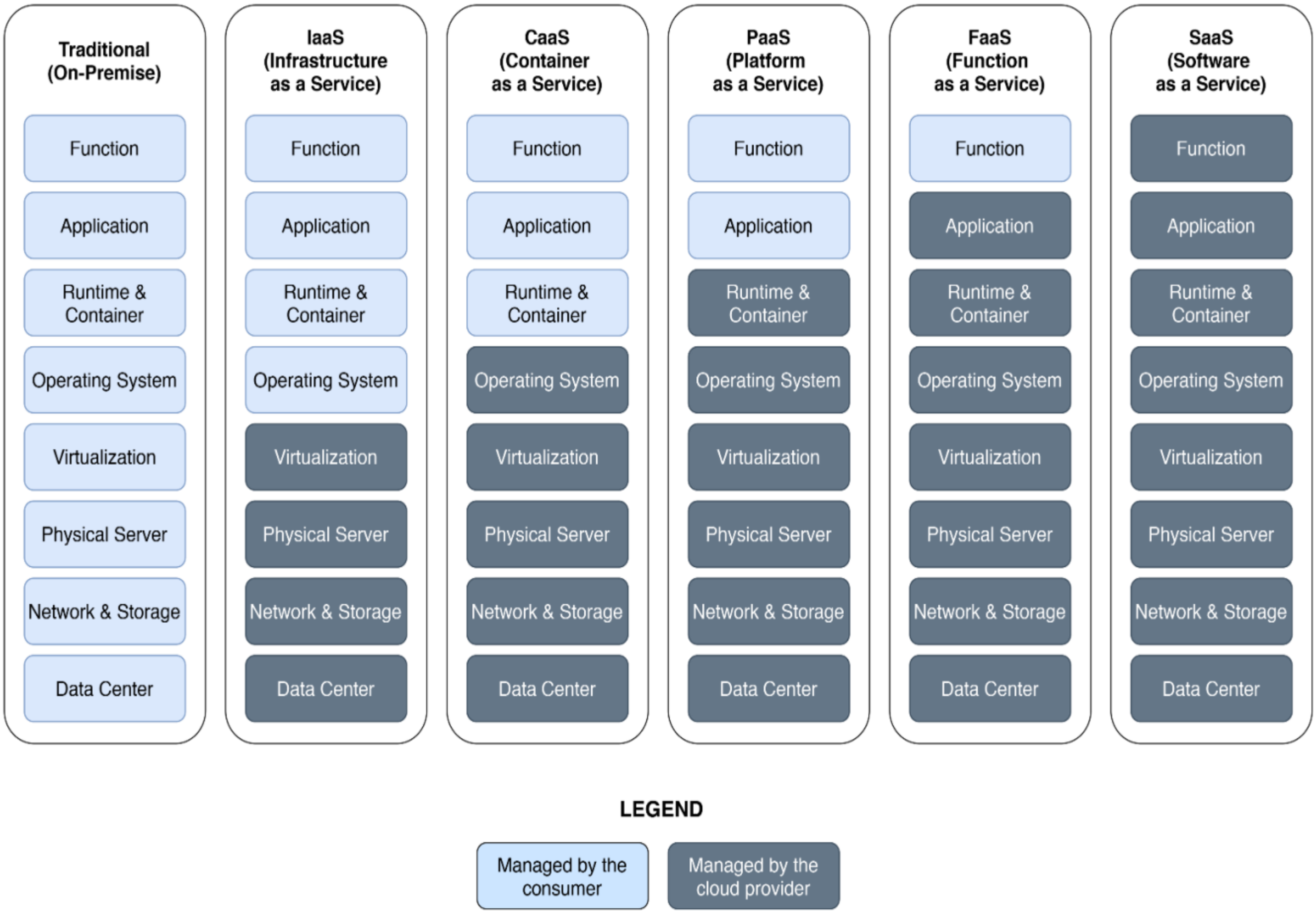


Management responsibilities for IaaS and PaaS



IaaS, PaaS, SaaS

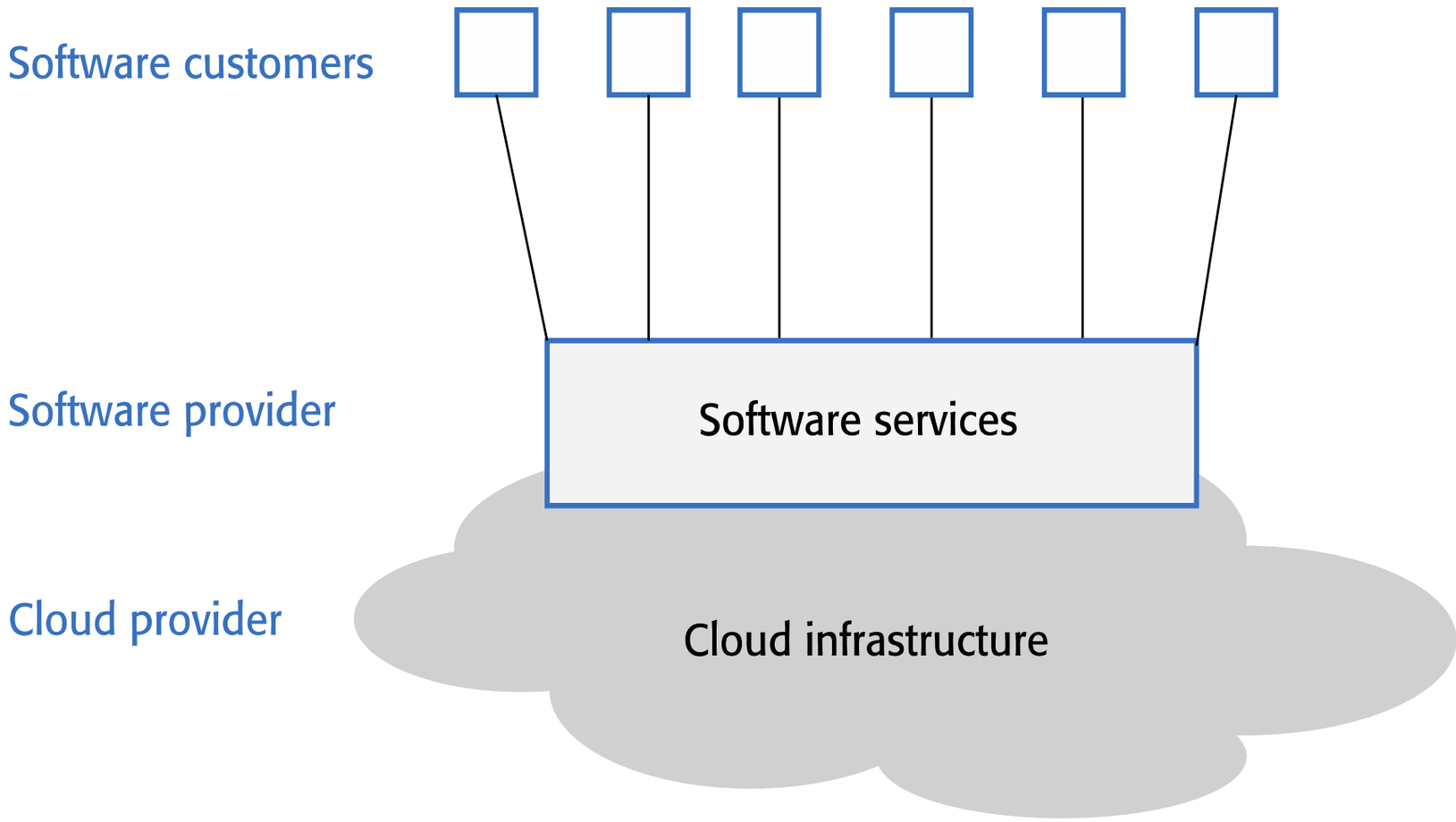



Traditional (On-Premise)
Function
Application
Runtime & Container
Operating System
Virtualization
Physical Server
Network & Storage
Data Center
IaaS (Infrastructure as a Service)
Function
Application
Runtime & Container
Operating System
Virtualization
Physical Server
Network & Storage
Data Center
CaaS (Container as a Service)
Function
Application
Runtime & Container
Operating System
Virtualization
Physical Server
Network & Storage
Data Center
PaaS (Platform as a Service)
Function
Application
Runtime & Container
Operating System
Virtualization
Physical Server
Network & Storage
Data Center
FaaS (Function as a Service)
Function
Application
Runtime & Container
Operating System
Virtualization
Physical Server
Network & Storage
Data Center
SaaS (Software as a Service)
Function
Application
Runtime & Container
Operating System
Virtualization
Physical Server
Network & Storage
Data Center
LEGEND
Managed by the consumer
Managed by the cloud provider

Software as a Service

Software product runs on the cloud and is accessed by users through a web browser or mobile app.

Software as a service

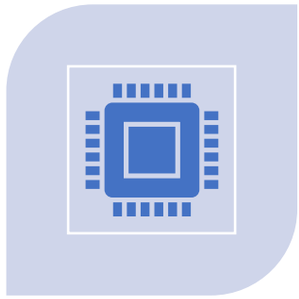


- Customers don’t have to install software and they access the remote system through a web browser or dedicated mobile app.

- The payment model for software as a service is usually a subscription model.

Benefits of SaaS for software product providers

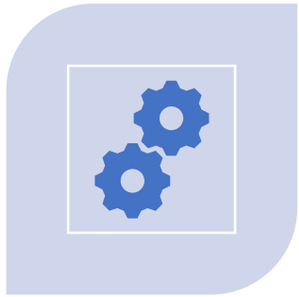
PAYMENT FLEXIBILITY



TRY BEFORE YOU BUY

DATA COLLECTION

CASH FLOW



UPDATE MANAGEMENT

CONTINUOUS DEPLOYMENT

Advantages and disadvantages of SaaS for customers

Advantages

- Mobile, laptop and desktop access
- No upfront costs for software or servers
- Immediate software updates
- Reduced software management costs

Software customer

Disadvantages

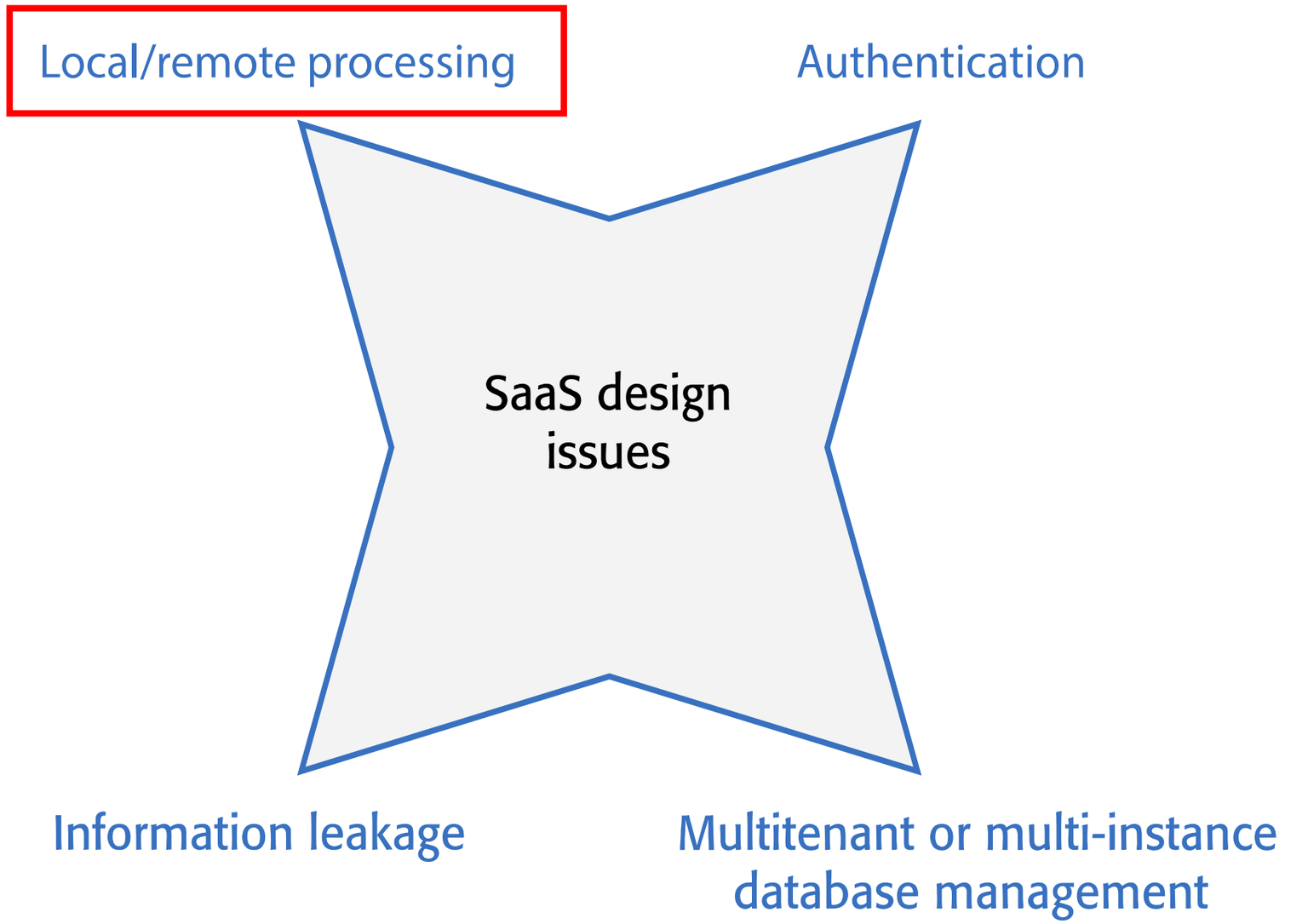
- Privacy regulation conformance
- Network constraints
- Security concerns
- Loss of control over updates
- Service lock-in
- Data exchange

Data storage and management issues for SaaS

- ***Regulation:*** Some countries, such as EU countries, have strict laws on the storage of personal information.
- ***Data transfer:*** ***T*****he software response time may be limited by the network speed.** Affordability problem for individuals and smaller companies.
- ***Data security:*** Companies with sensitive information may be unwilling to hand over the control of their data to an external software provider.
- ***Data exchange:*** exchange data btw a cloud service and other services or local software applications, can be difficult **unless the cloud service provides an API.**

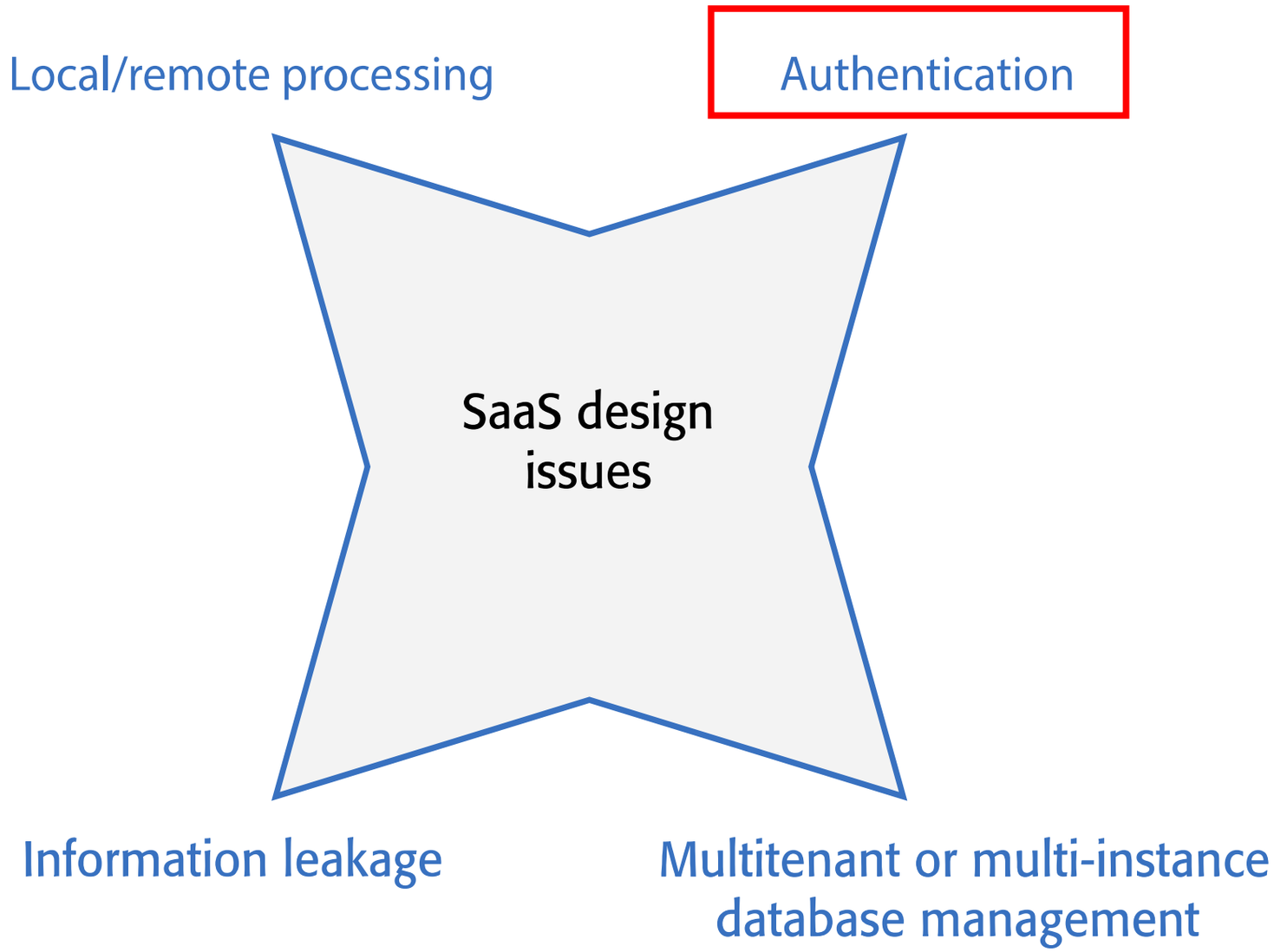
Design issues for software delivered as a service

- Local execution reduces network traffic and → increases user response speed.
- Local processing increases the electrical power needed to run the system.

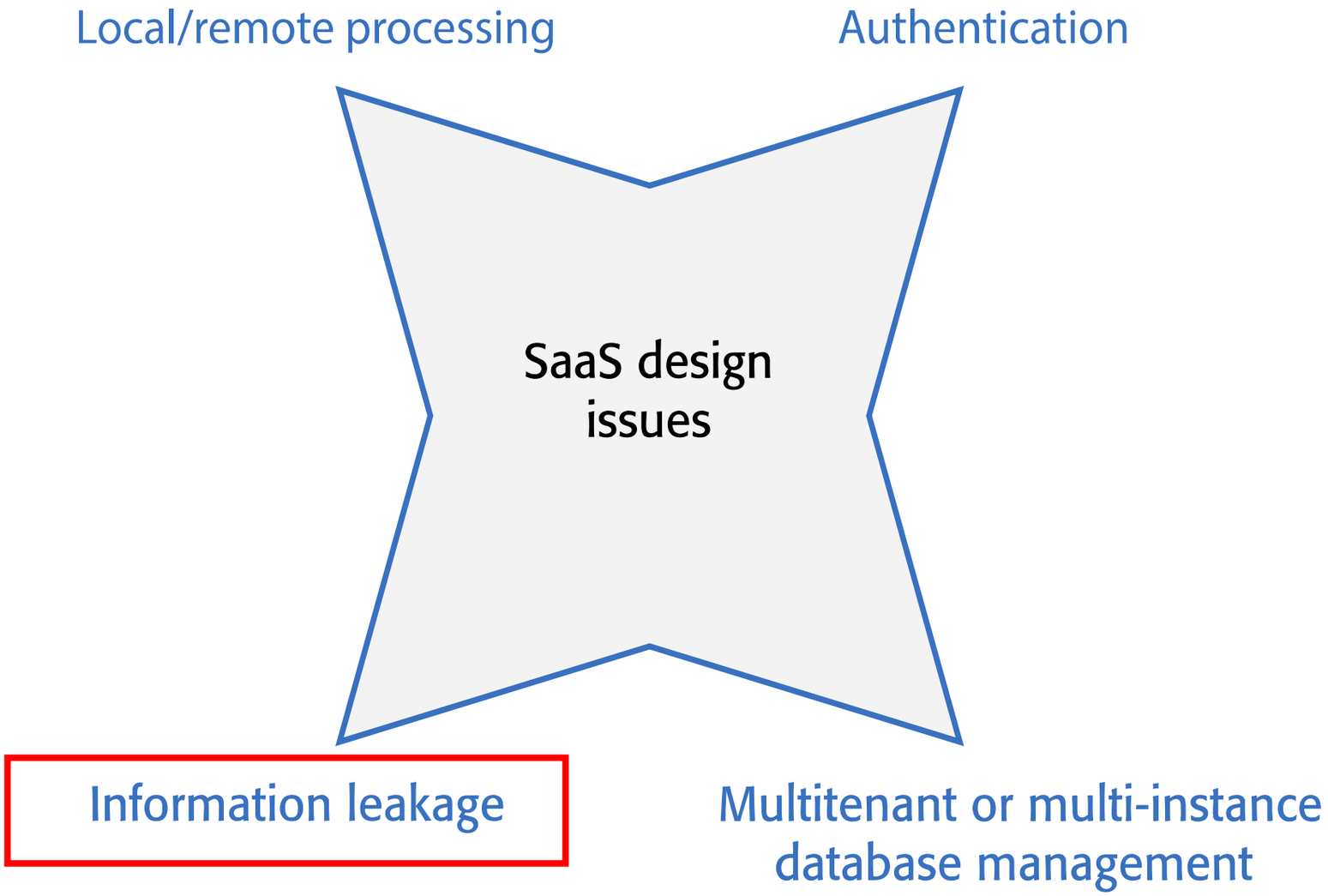


Design issues for software delivered as a service

- Users must remember another set of authentication credentials, (Many systems allow authentication using the user's Google, Facebook or LinkedIn credentials, federated authentication system)

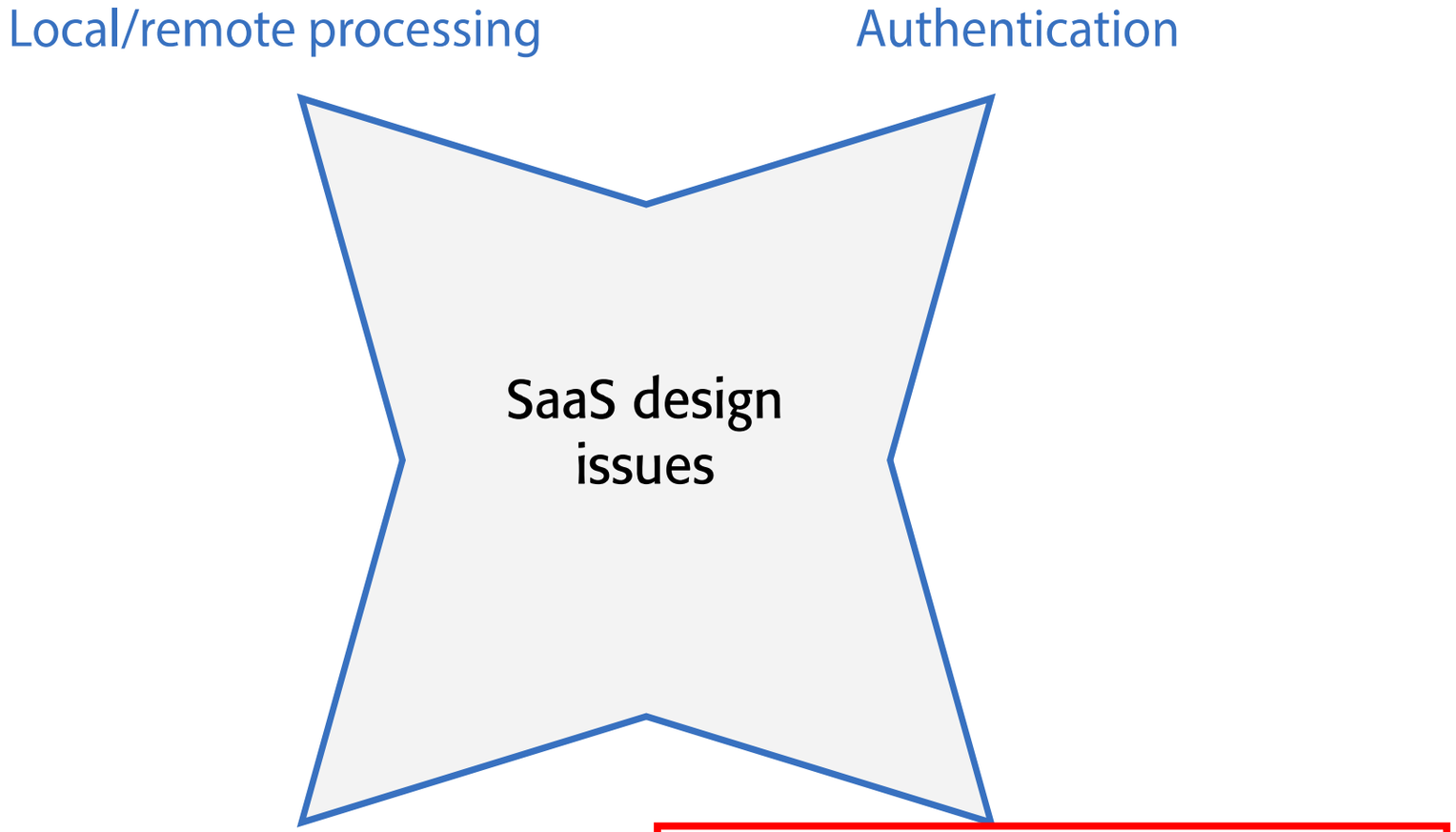


Design issues for software delivered as a service



- With multiple users from multiple organizations, a security risk is that information leaks from one organization to another.

Design issues for software delivered as a service

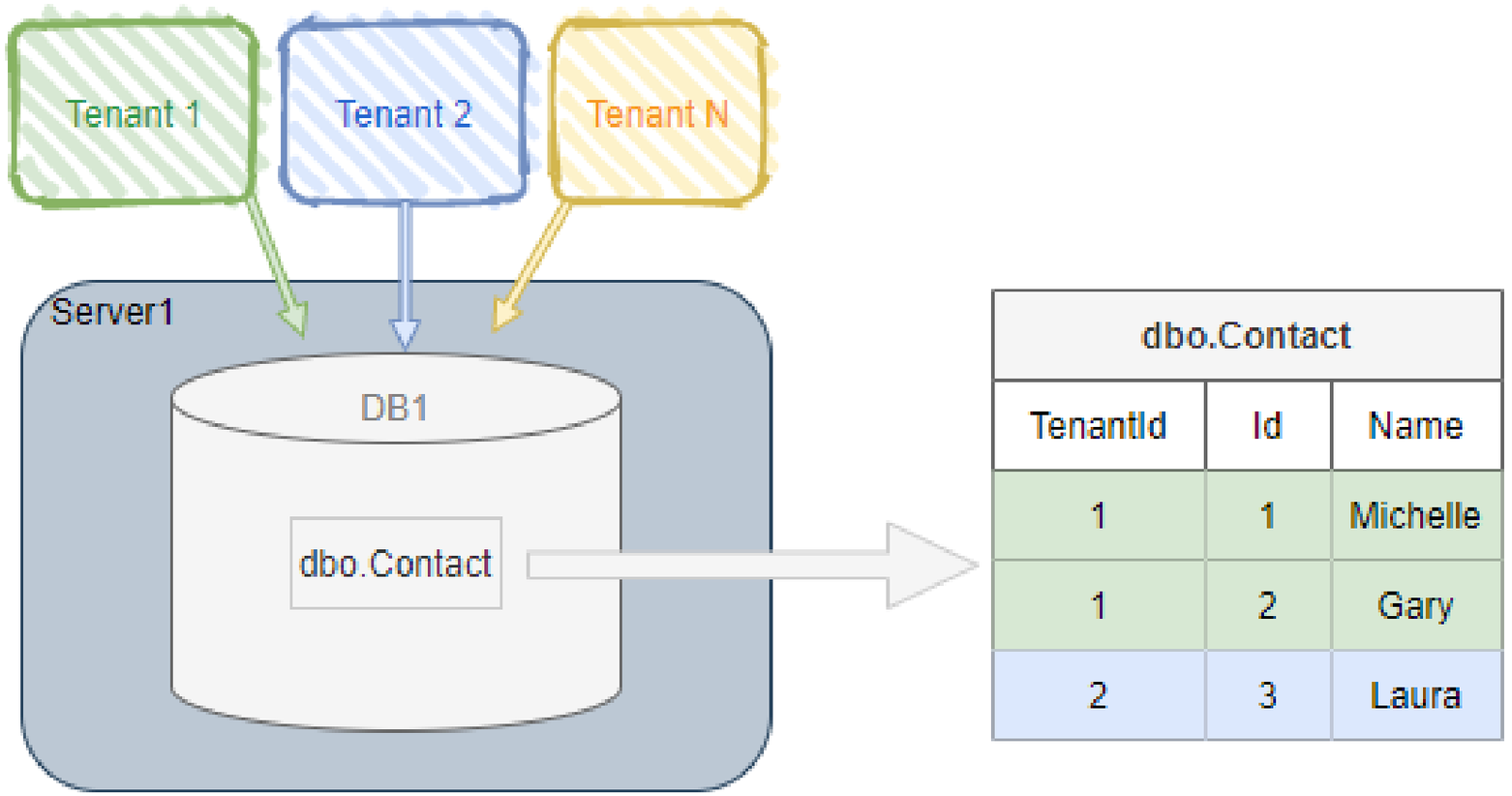


- All customers are served by a single instance of the system and a multitenant database.
- In a multi-instance system, a separate copy of the system and database is made available for each user.

Multi-tenant systems

All customers are served by **a single instance of the system** and a multi-tenant database.

Multi-tenant systems



dbo.Contact		
TenantId	Id	Name
1	1	Michelle
1	2	Gary
2	3	Laura

- There is a single database schema, defined by the SaaS provider, that is shared by all the system's users.

Advantages of multi-tenant databases

Resource utilization

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.

Security

The data for all customers is held in the same database. Security management is simplified as there is only a single copy of the database software to be patched if a security vulnerability is discovered.

Update management

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.

Disadvantages of multi-tenant databases

Inflexibility

Customers must all use the same database schema with limited scope for adapting this schema to individual needs.

Security

As data for all customers is maintained in the same database, then there is a theoretical possibility that data will **leak from** one customer to another.

Complexity

Multitenant 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.

An example of a multi-tenant database

Stock management					
Tenant	Key	Item	Stock	Supplier	Ordered
T516	100	Widg 1	27	S13	2017/2/12
T632	100	Obj 1	5	S13	2017/1/11
T973	100	Thing 1	241	S13	2017/2/7
T516	110	Widg 2	14	S13	2017/2/2
T516	120	Widg 3	17	S13	2017/1/24
T973	100	Thing 2	132	S26	2017/2/12

Possible Customizations for SaaS

AUTHENTICATION

BRANDING

BUSINESS RULES

DATA SCHEMAS

ACCESS CONTROL

Adding fields to extend the database

Add some extra columns to each database table and define a customer profile that maps the column names that the customer wants to these extra columns.

Stock management								
Tenant	Key	Item	Stock	Supplier	Ordered	Ext 1	Ext 2	Ext 3
T516	100	Widg 1	27	S13	2017/2/12			
T632	100	Obj 1	5	S13	2017/1/11			
T973	100	Thing 1	241	S13	2017/2/7			
T516	110	Widg 2	14	S13	2017/2/2			
T516	120	Widg 3	17	S13	2017/1/24			
T973	100	Thing 2	132	S26	2017/2/12			

Database extensibility using tables

Main database table

Tab1

Stock management						
Tenant	ID	Item	Stock	Supplier	Ordered	Ext 1
T516	100	Widg 1	27	S13	2017/2/12	E123
T632	100	Obj 1	5	S13	2017/1/11	E200
T973	100	Thing 1	241	S13	2017/2/7	E346
T516	110	Widg 2	14	S13	2017/2/2	E124
T516	120	Widg 3	17	S13	2017/1/24	E125
T973	100	Thing 2	132	S26	2017/2/12	E347

Tab2

Field names		
Tenant	Name	Type
T516	‘Location’	String
T516	‘Weight’	Integer
T516	‘Fragile’	Bool
T632	‘Delivered’	Date
T632	‘Place’	String
T973	‘Delivered’	Date

Extension table showing the field names for each company that needs database extensions

Tab3

Field values		
Record	Tenant	Value
E123	T516	‘A17/S6’
E123	T516	‘4’
E123	T516	‘False’
E200	T632	‘2017/1/15’
E200	T632	‘Dublin’
E346	T973	‘2017/2/10’
...		

Value table showing the value of extension fields for each record

Multi-instance systems

Each customer has its own system that is adapted to its needs, including its own database and security controls.

Multi-instance databases

- **VM-based multi-instance systems**: the software instance and database for each customer runs in its own virtual machine. All users from the same customer may access the shared system database.

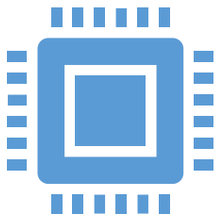
- **Container-based multi-instance systems:** each user has an isolated version of the software and database running in a set of containers.

Advantages and Disadvantages of multi-instance databases

Flexibility

Security

Scalability



Resilience

Cost

Update management

Cloud software architecture

You need to decide on the most important **software attributes, the delivery platform, and the technology used.**

Architectural decisions for cloud software engineering

Database organization

Should the software use a multitenant or multi-instance database?

Scaleability and resilience

What are the software scaleability and resilience requirements?

Software structure

Should the software structure be mono-lithic or service-oriented?

What cloud platform should be used for development and delivery?

Cloud platform

Database Organization

3 possible ways of providing a customer database in a cloud-based system:

1. As a multi-tenant system, shared by all customers for your product. **This may be hosted in the cloud using large, powerful servers.**
2. As a multi-instance system, with each customer database running on its own virtual machine.
3. As a multi-instance system, with each database running in its own container. The customer database may be distributed over several containers.

Questions to ask when choosing a database organization

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.**

Questions to ask when choosing a database organization

Transaction requirements

- Is it critical that your products support ACID transactions where the data is guaranteed to be always consistent?
- If so, use a **multi-tenant database or a VM-based multi-instance database.**

Questions to ask when choosing a database organization (cont.)

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.**

Questions to ask when choosing a database organization (cont.)

Database interoperability

- Will customers wish to transfer information from existing databases?
- What are the differences in schemas between these and a possible multitenant 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.**

Questions to ask when choosing a database organization (cont.)

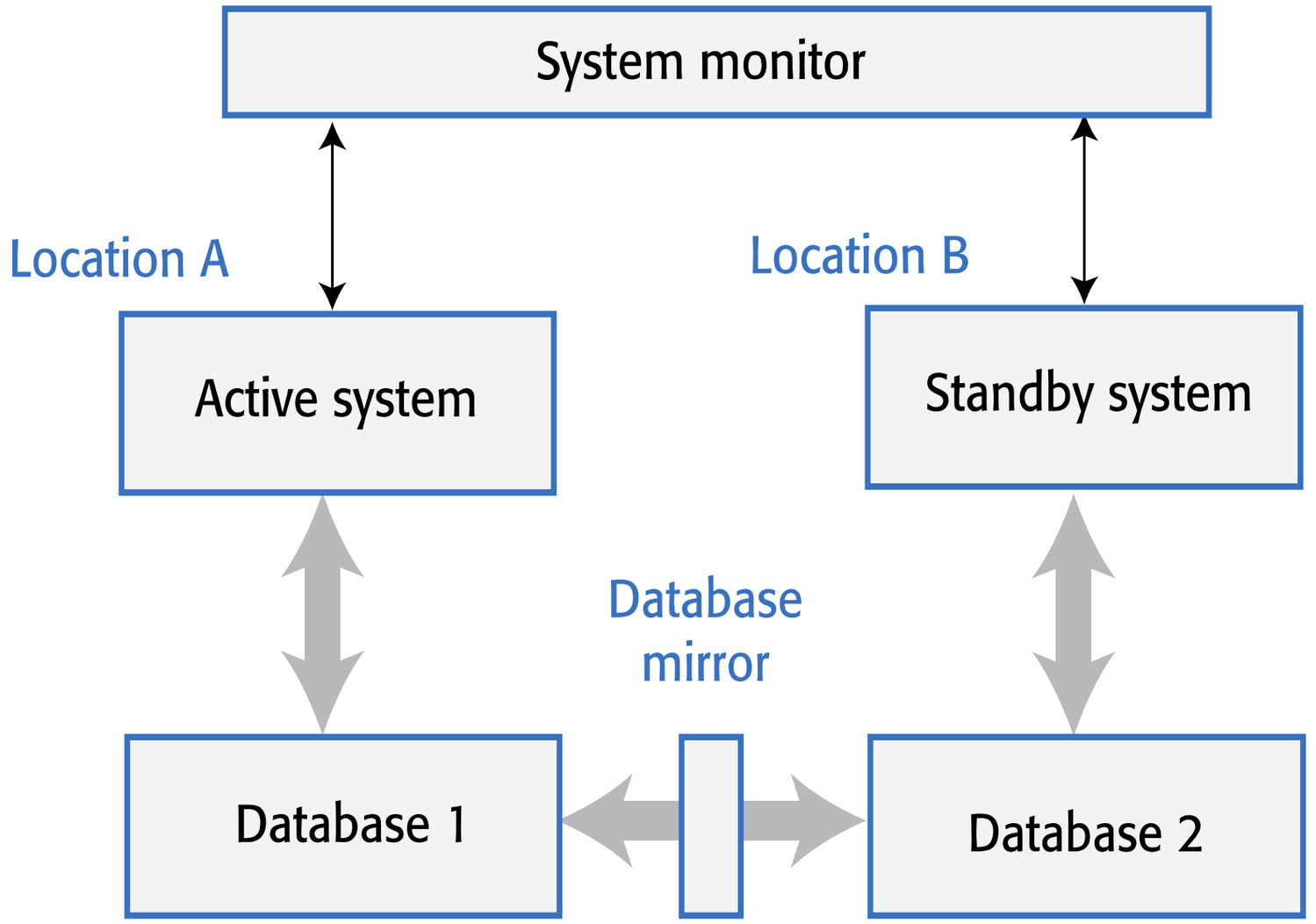
System structure

- 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.**

Scalability and resilience

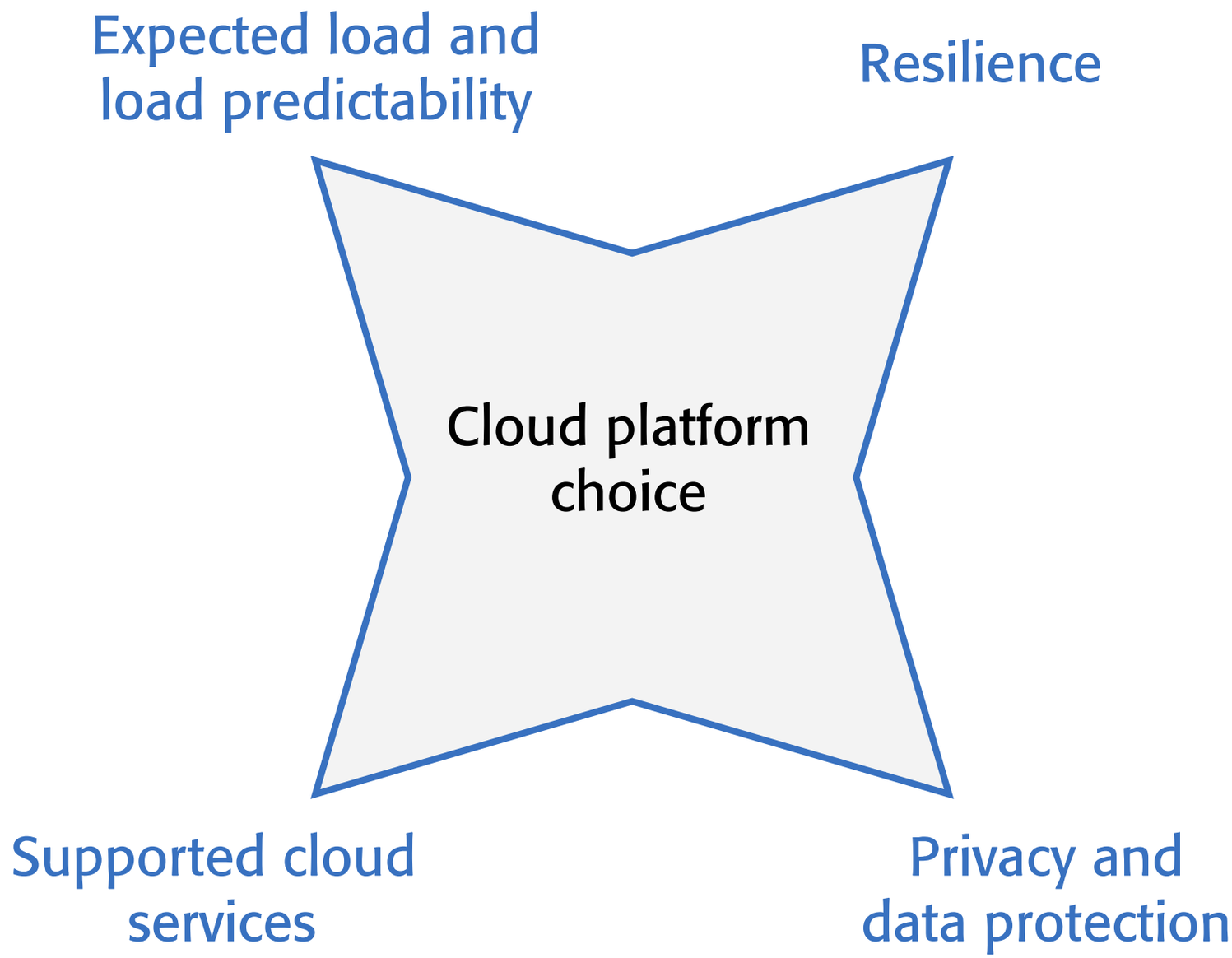
Using a standby system to provide resilience

- You should use redundant virtual servers that are not hosted on the same physical computer and locate servers in different locations.



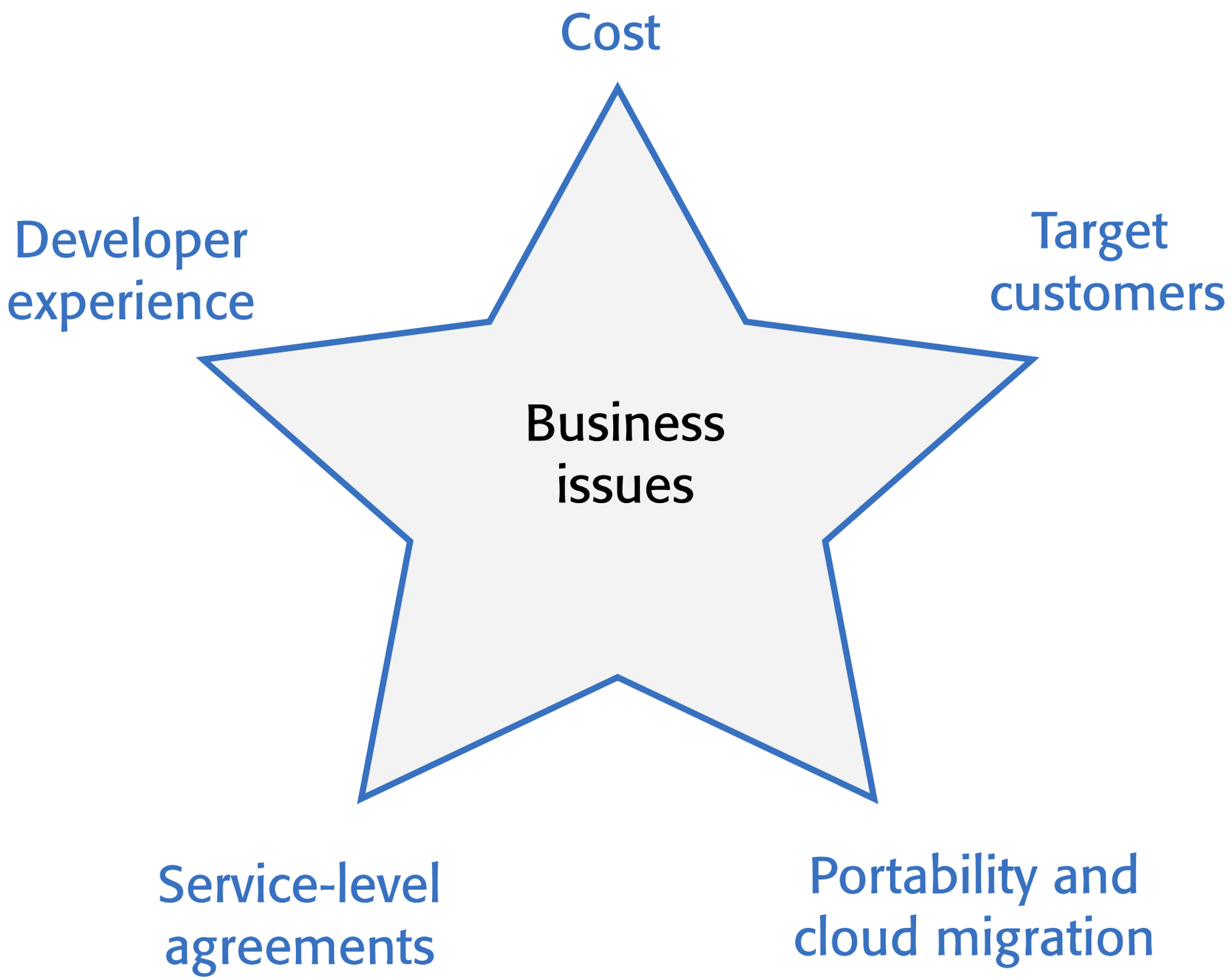
Technical issues in cloud platform choice

How to choose your Cloud platform?



Business issues in cloud platform choice

How to choose your Cloud platform?



Service Level Agreement

A cloud **SLA** is an agreement between a cloud service provider and a customer that ensures a minimum level of service is maintained.
It guarantees levels of reliability, availability and responsiveness to systems and applications; and describes penalties if service levels are not met.