Cloud-based software

Youssef Ghatas

Acknowledgement

- Presentation is adapted from Ian Sommerville “Engineering Software Products”

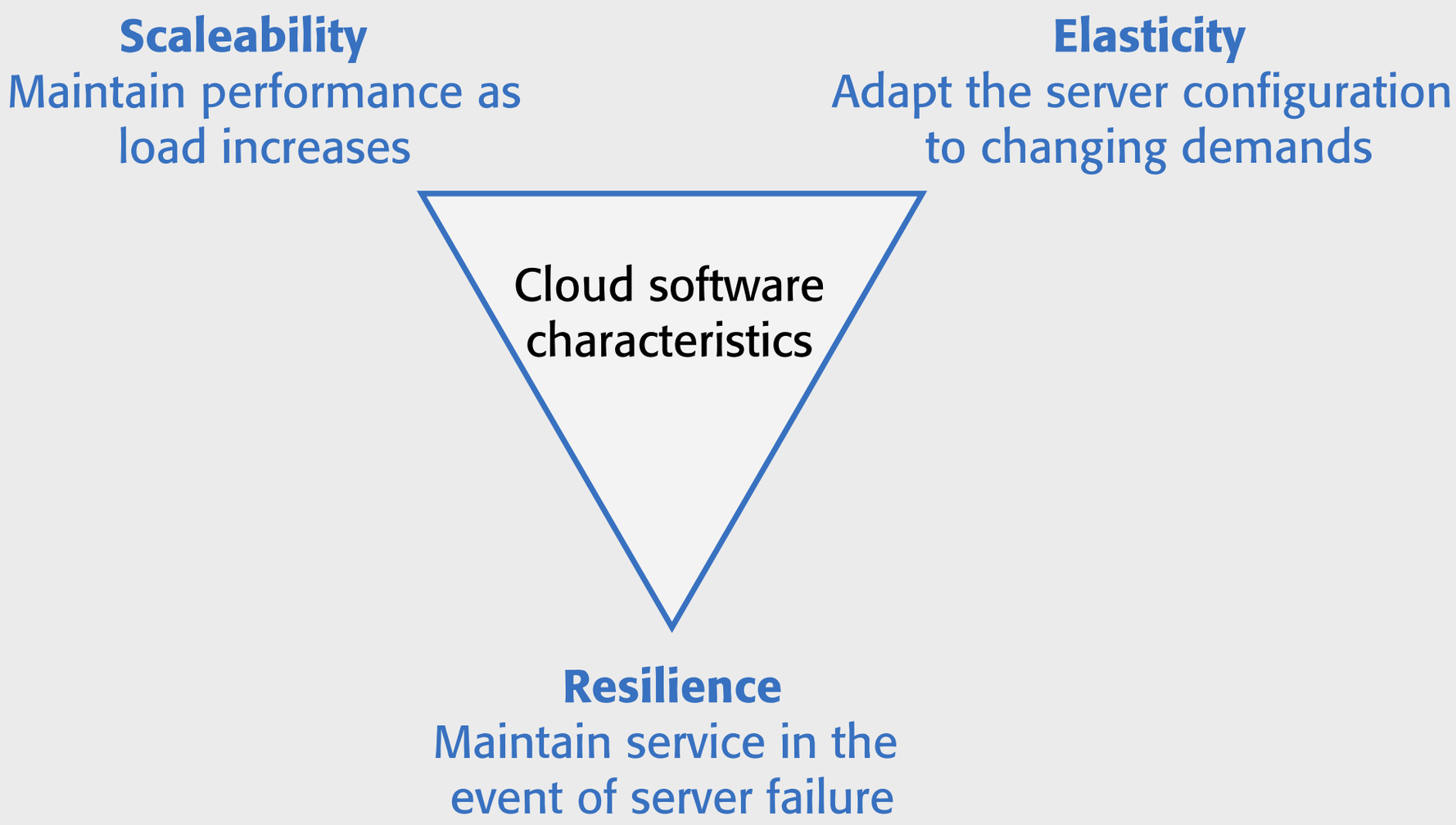
Agenda

- What is cloud computing?
- VM concepts.
- Containers.
- Docker.
- .. as a Service
    - IaaS
    - PaaS
    - SaaS
- Single vs Mutli-tenancy.
- SaaS Customization
- Intro to Resiliance
- How to choose Cloud Platfrom?

The cloud

- The cloud is made up of ve**ry large number of remote servers that are offered for rent** by companies that own these servers.
  - Cloud-based servers are '**virtual servers**', which means that they are implemented in software rather than hardware.
- 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 such as a tablet or a TV.
  - Cloud servers can be started up and shut down as demand changes.
- You may rent a server and install your own software, or you may pay for access to software products that are available on the cloud.

Scaleability, elasticity and resilience



Scaleability, elasticity and resilience

- Scaleability 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 is maintained.
- Elasticity is related to **scaleability 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 change.
- 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.

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. 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.

Why virtualization?

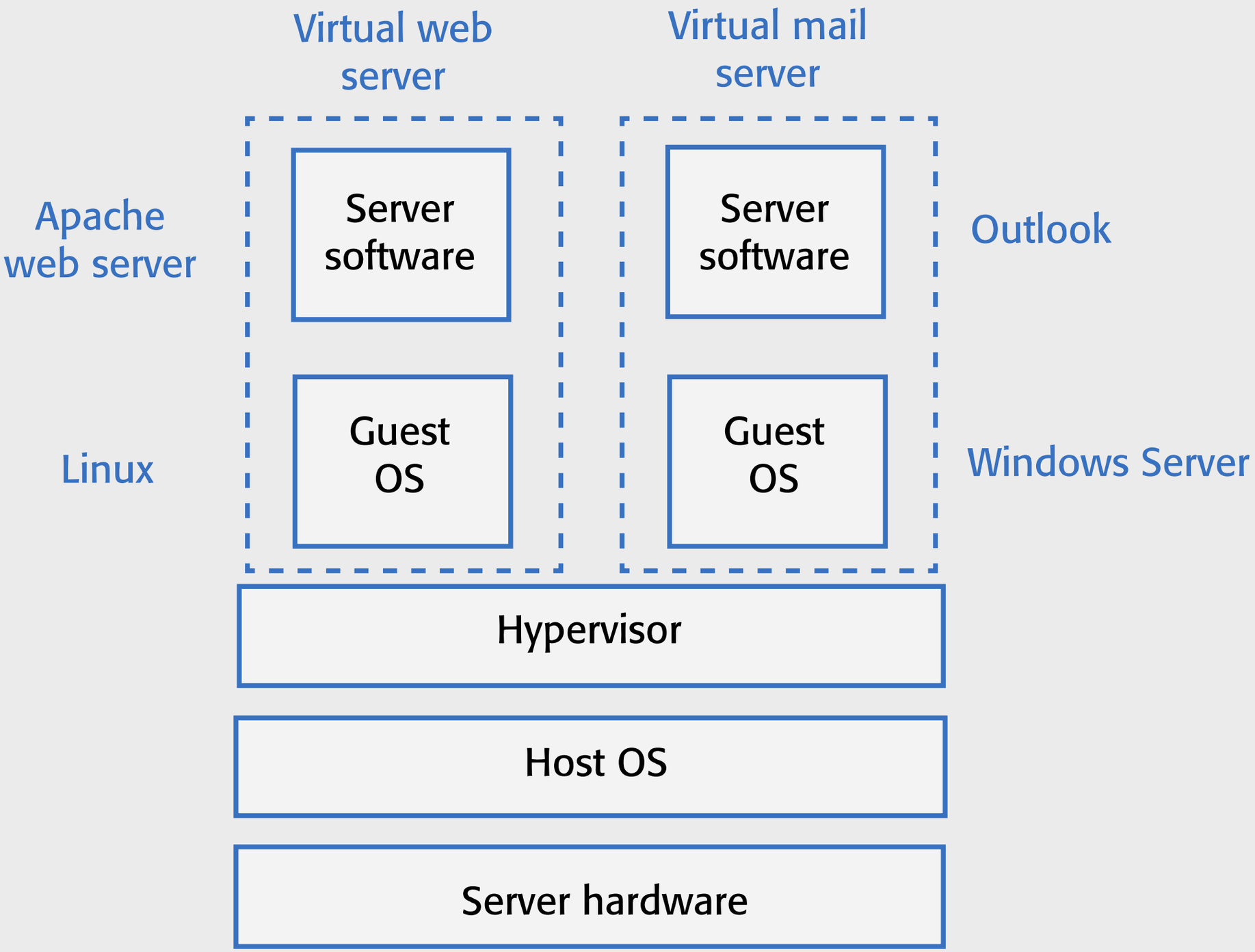
Types of virtualization

- Virtual Machine
  - Why?
- Container-based

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.
- If you use a virtual machine to implement virtual servers, you have (almost) exactly the same hardware platform as a physical server.

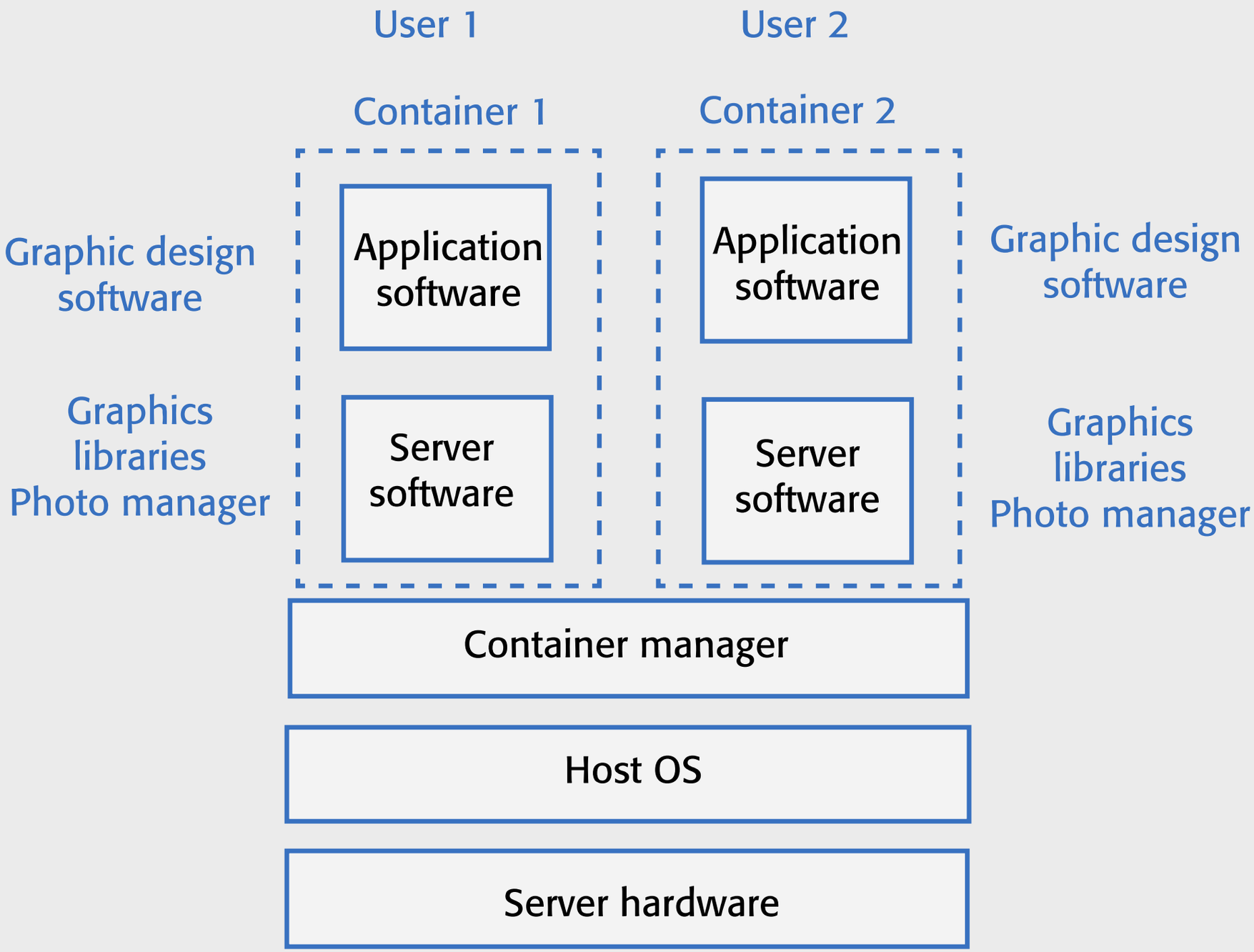
Implementing a virtual server as a virtual machine



Container-based virtualization

- 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.

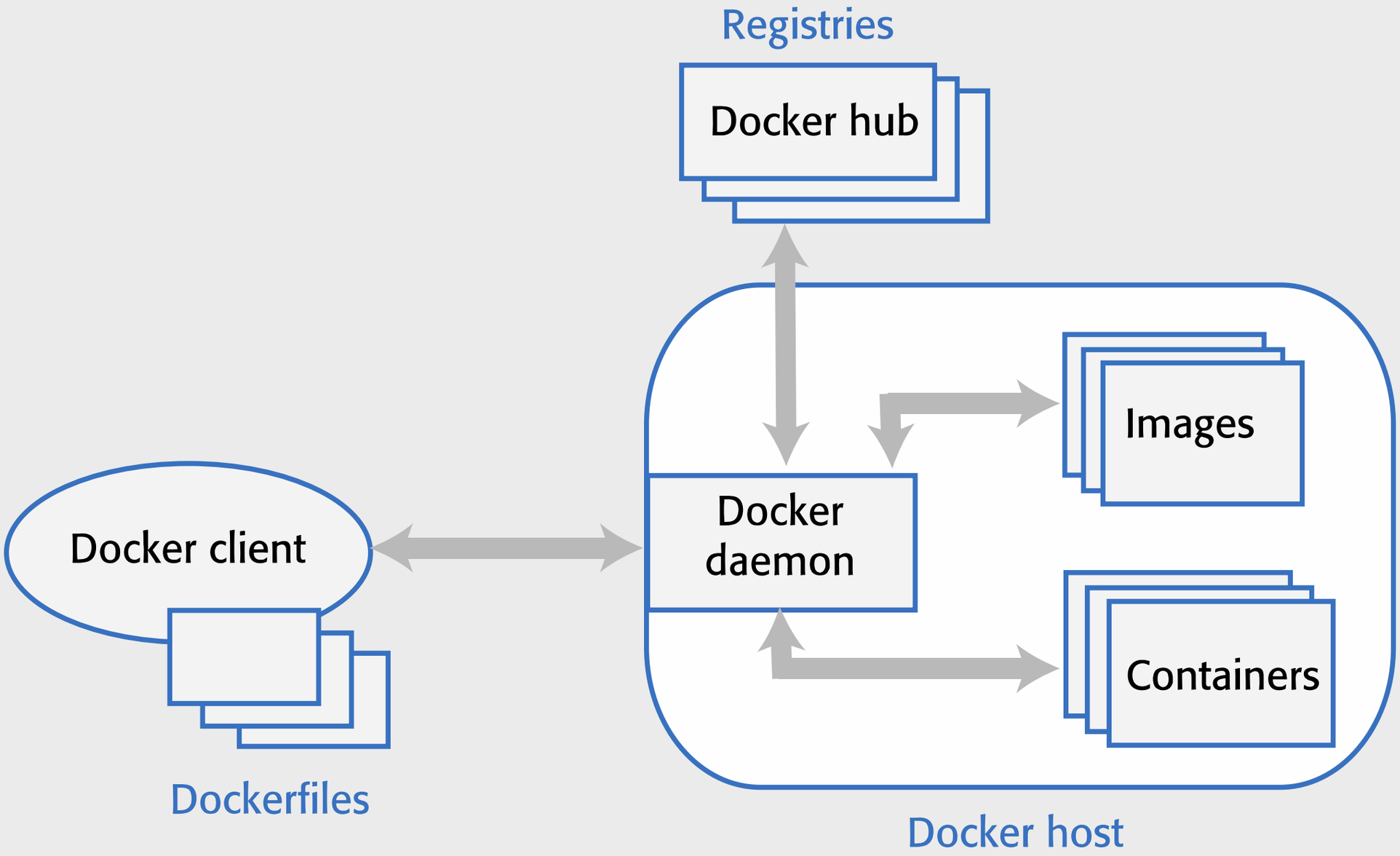
Using containers to provide isolated services



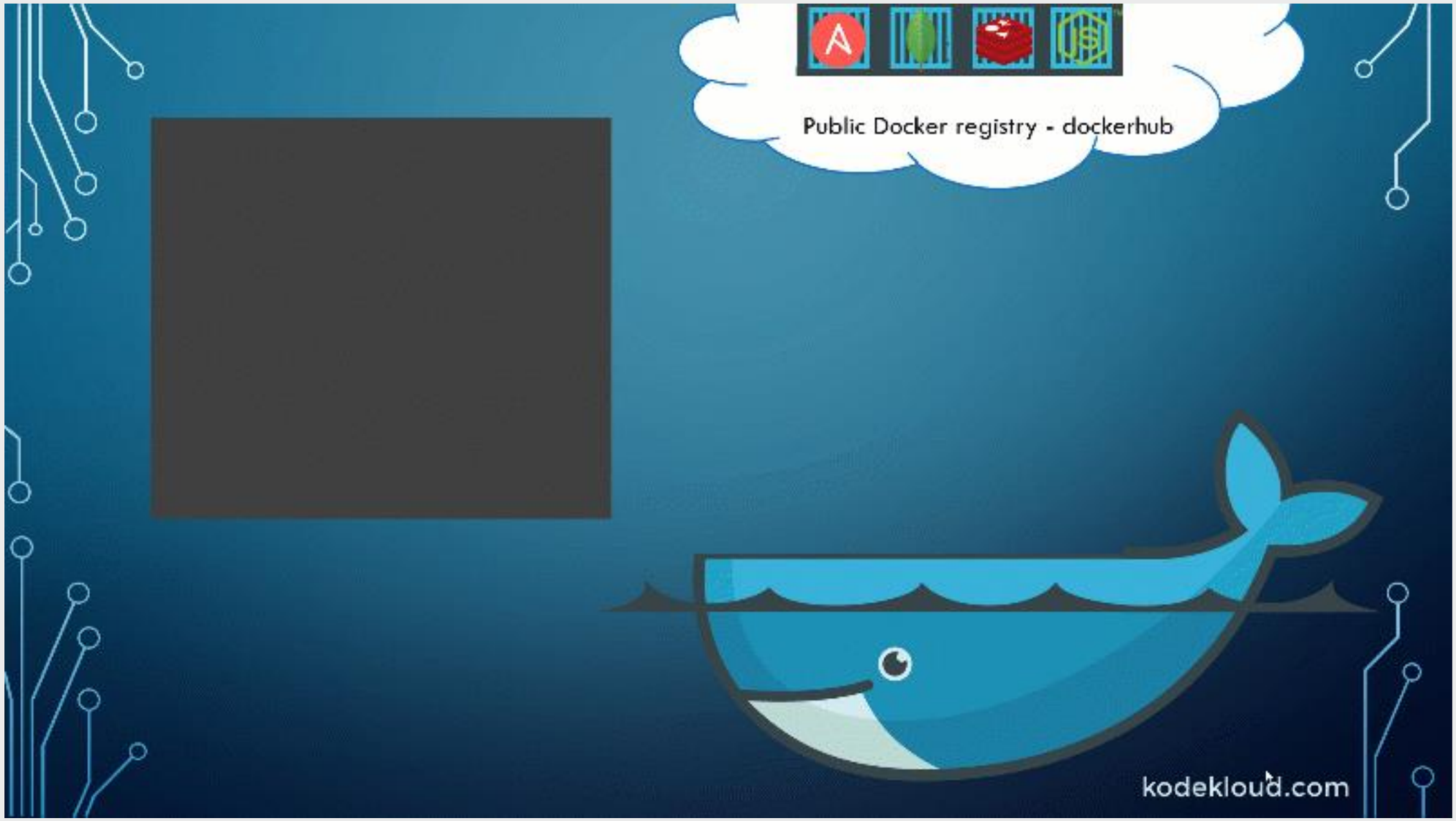
Docker

- 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



The Docker container system



https://r42724.medium.com/load-a-machine-learning-model-inside-the-docker-container-252be24e69d4

The elements of the Docker container system

Docker daemon
This is a process **that runs on a host server and is used to setup, start, stop, and monitor containers**, as well as **building and managing** local images.

Docker client
This **software is used by developers** and system managers to define and control containers

Dockerfiles
Dockerfiles 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.

Image
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 elements of the Docker container system

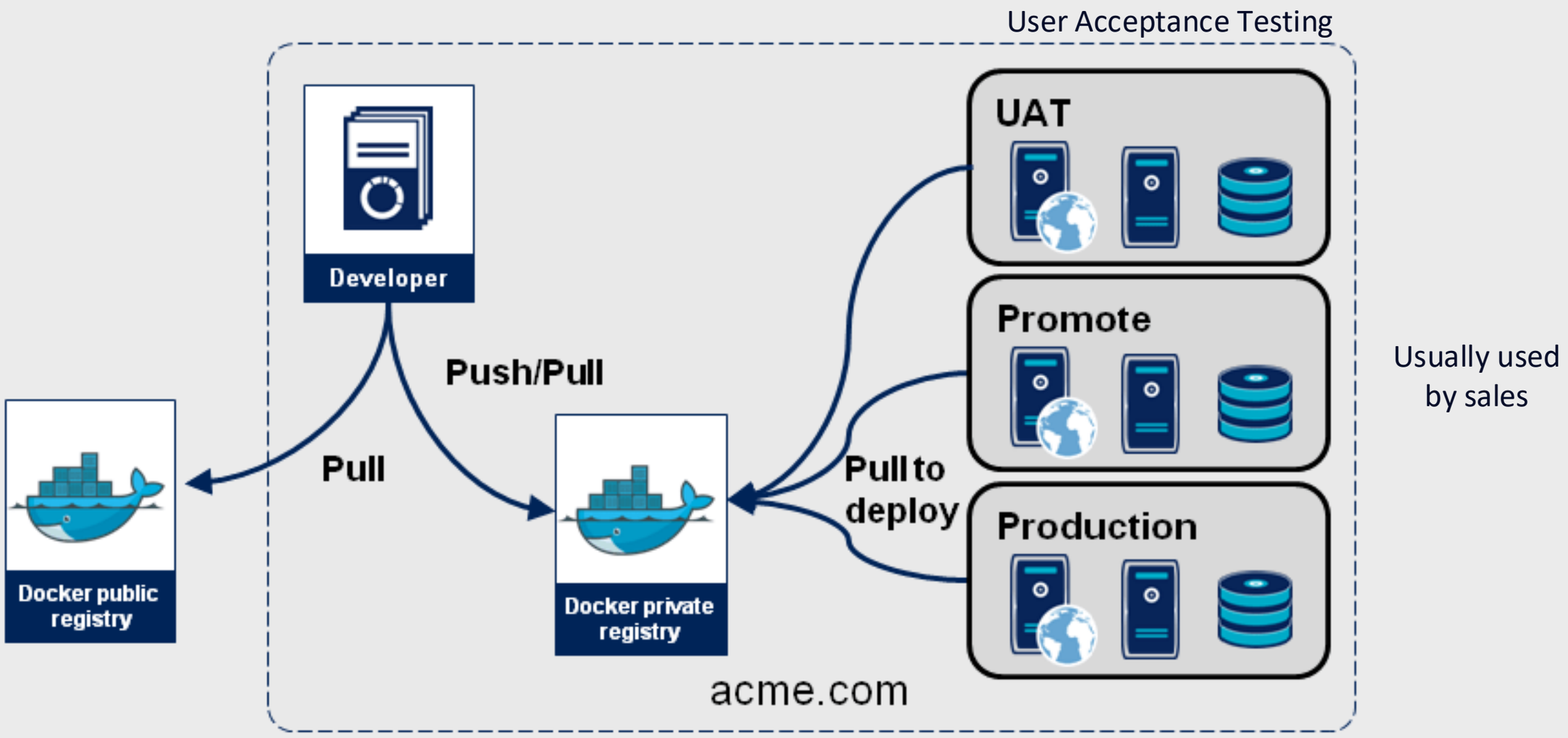
Docker hub

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.

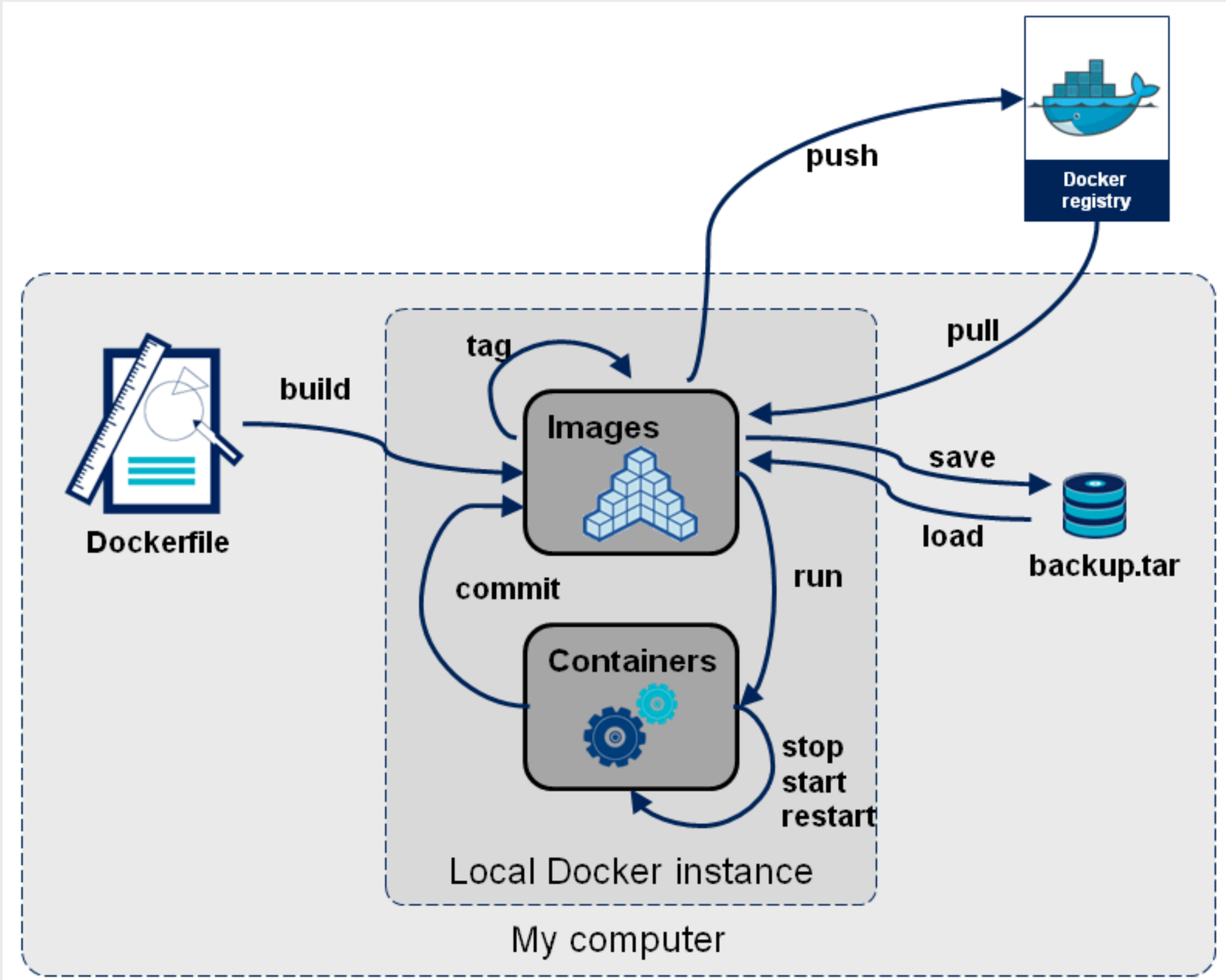
Real life scenario: P1



Usually used by sales

https://blog.octo.com/docker-registry-first-steps/

Real life scenario: P2



https://blog.octo.com/docker-registry-first-steps/

Let’s see a practical example!

Docker images

- Everything that's needed to run a software system - binaries, libraries, system tools, etc. is included in the DockerImage.
- A Docker image is a base layer, usually taken from the Docker registry, with your own software and data added as a layer on top of this.
- The **layered model** means that updating Docker applications is **fast and efficient**. Each update to the filesystem is a layer on top of the existing system.
- To change an application, all you have to do is to ship the changes that you have made to its image, often just a small number of files.

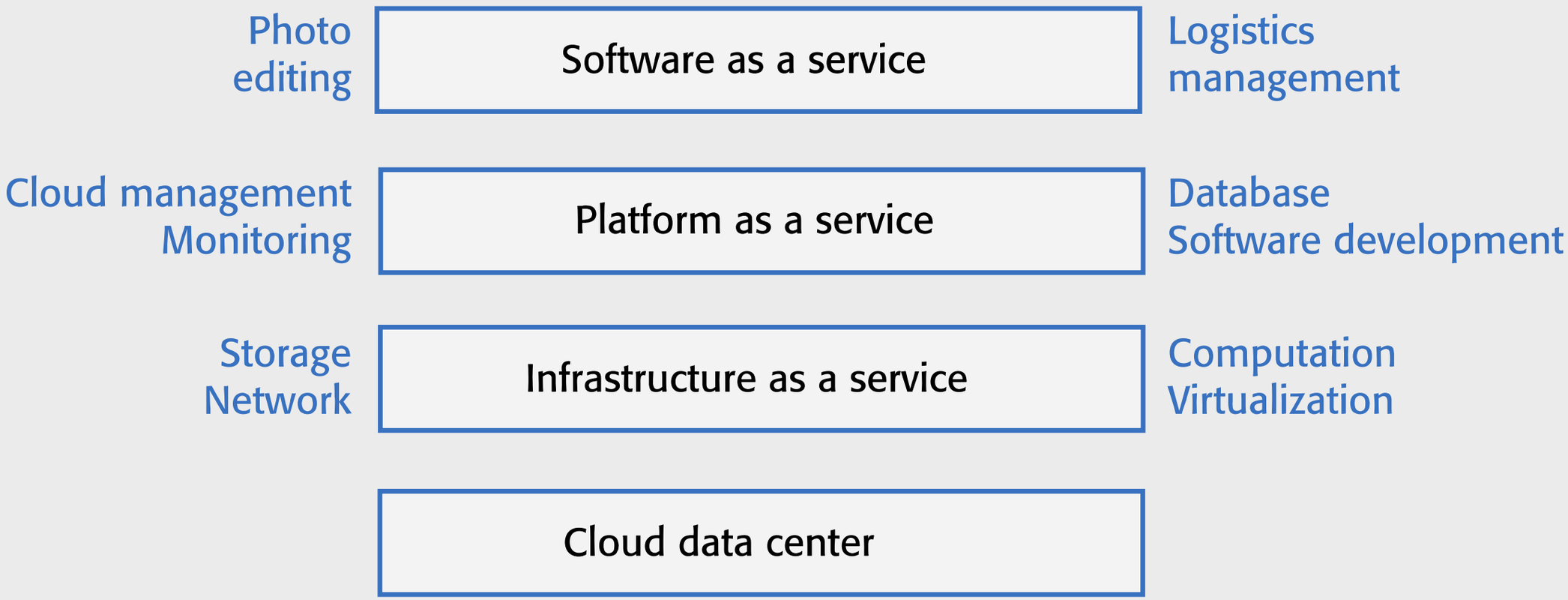
Benefits of containers

- They solve the problem of **software dependencies**.
  - Instead of shipping your product as stand-alone software, you can ship a container that includes all of the support software that your product needs.
- Software **portability across different clouds**. All what u need is docker daemon.
- Efficient mechanism for implementing software services and so support the development of **service-oriented architectures**.
- **They simplify the adoption of DevOps**.

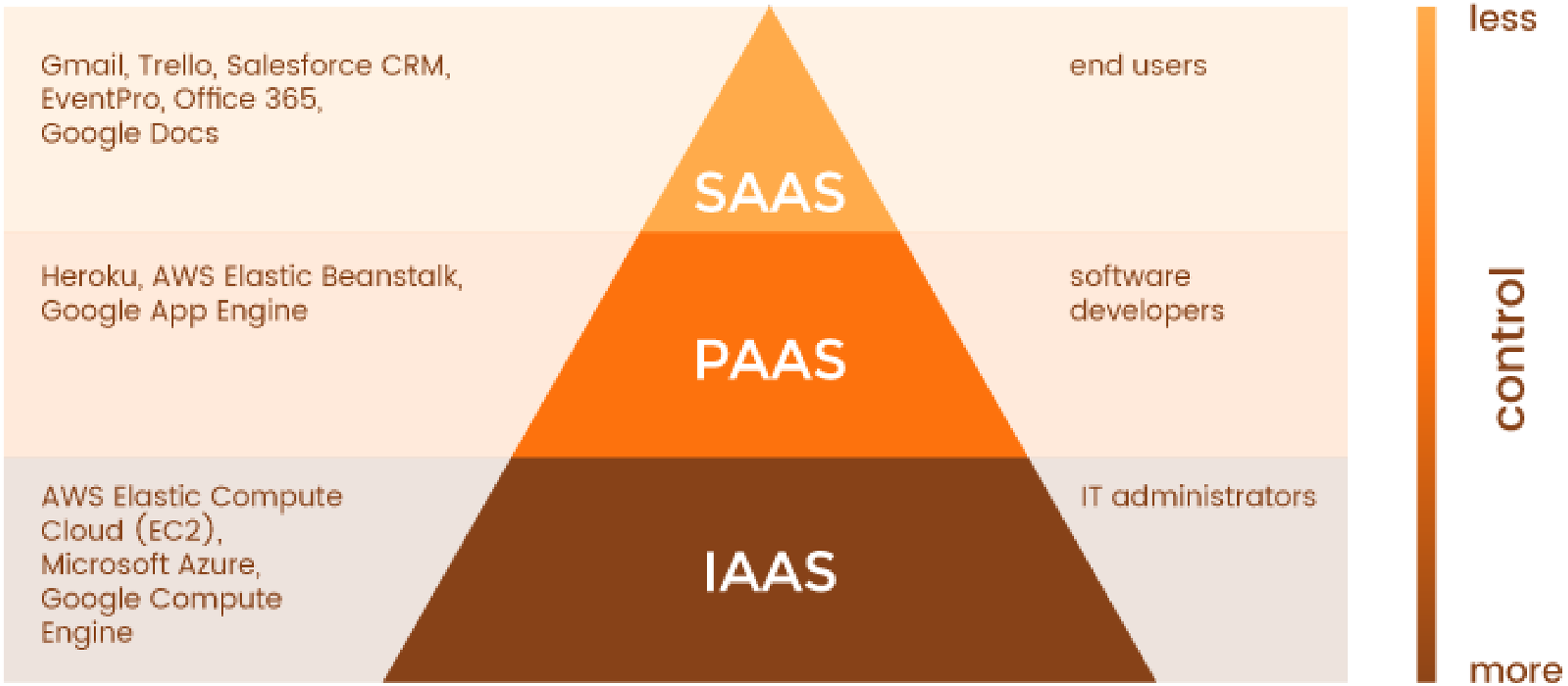
Everything as a service

- 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**.

Everything as a service

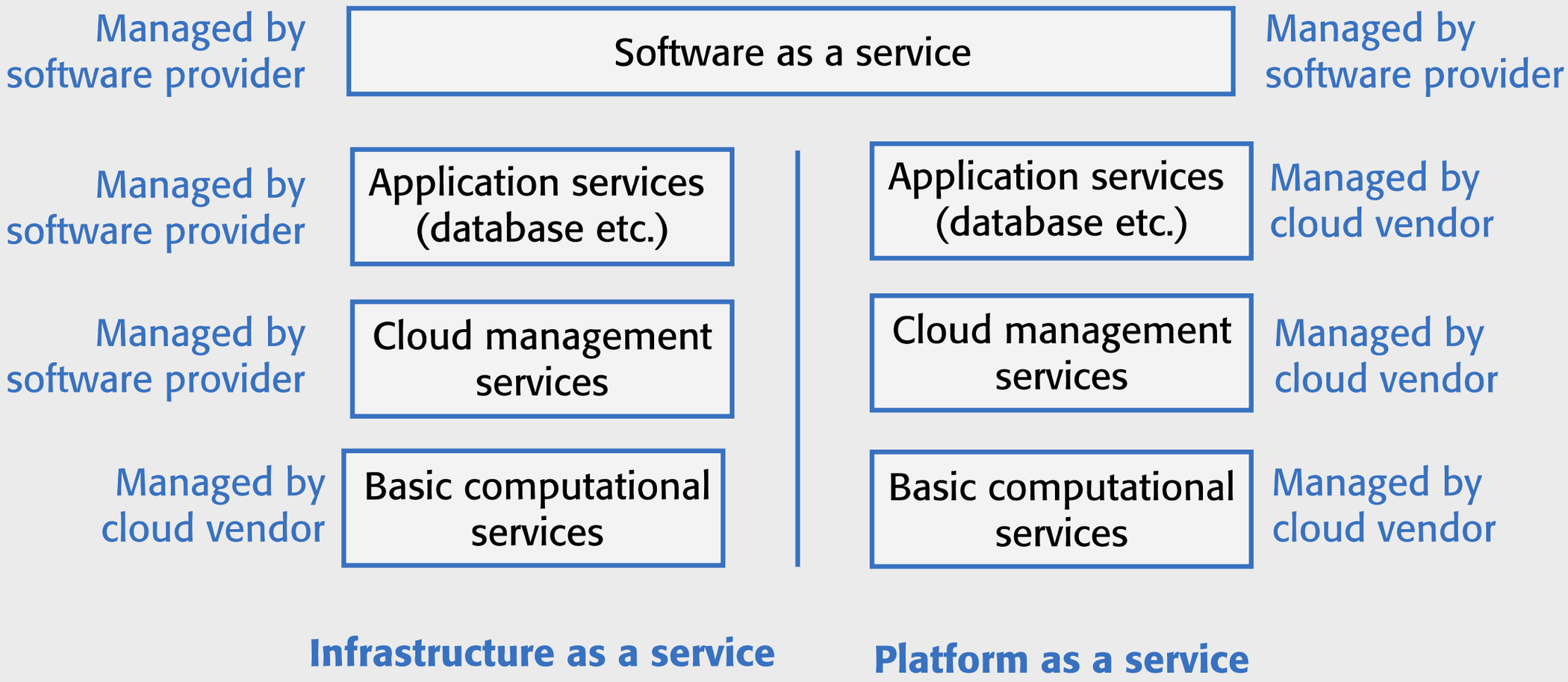


Everything as a service



https://rubygarage.org/blog/iaas-vs-paas-vs-saas

Management responsibilities for IaaS and PaaS



Group Activity (Different Slides)

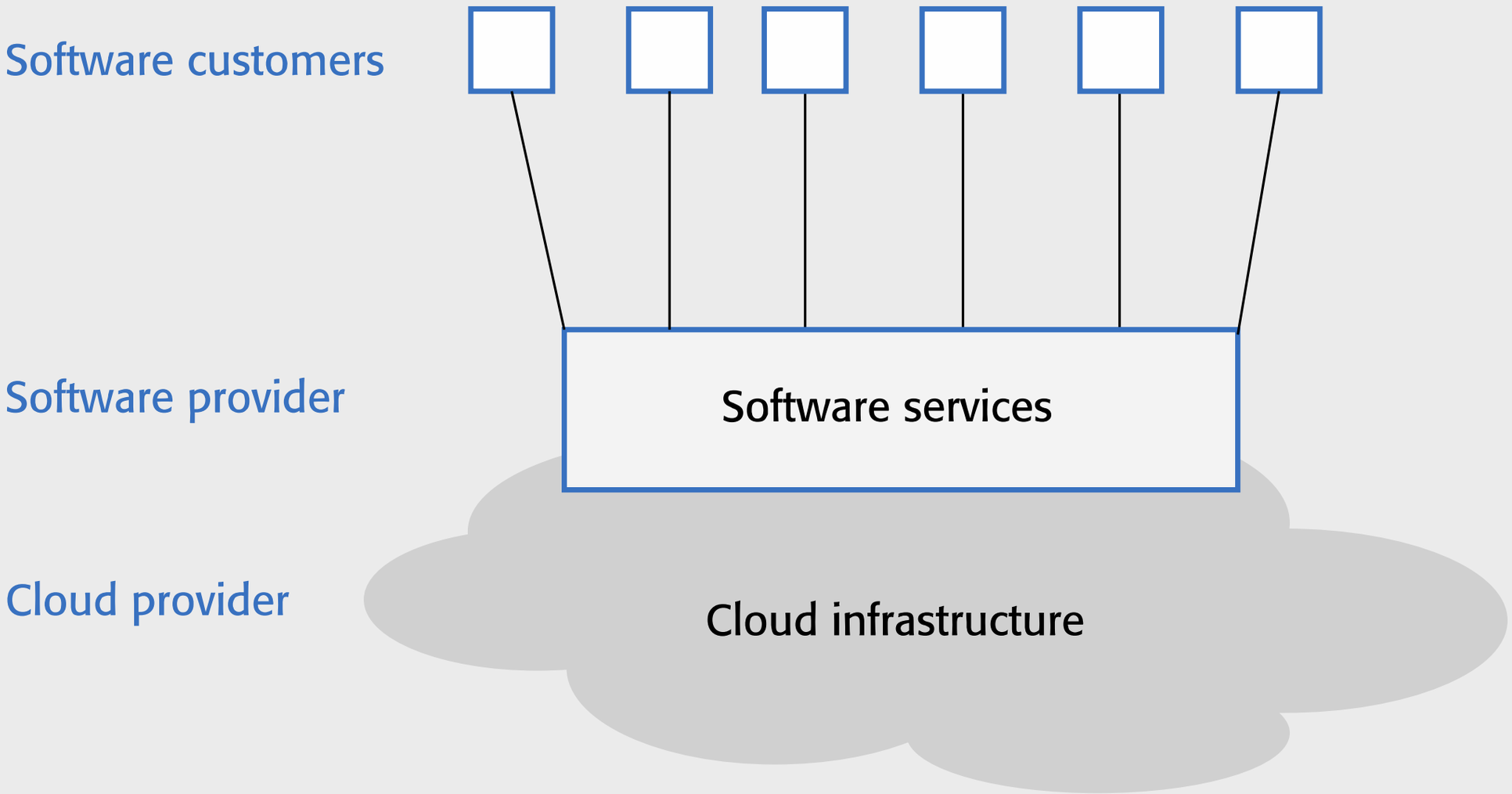
Group Activity (Different Slides)

- General Architecture (Dedicated Server)
  - DB Choice.
  - Static files
  - Caching
  - Others?
- AWS Tools:
  - S3
  - EC2
  - Lambda
  - RDS
  - DynamoDB
  - CloudFront
  - SES
  - (API Gateway)
  - (Amazon Route 53)

Software as a service

- Increasingly, **software products are being delivered as a service**, rather than installed on the buyer's computers.
- You run the software on your servers, which you may rent from a cloud provider. Customers don't have to install software.
- The payment model for software as a service is usually a **subscription model**.

Software as a service



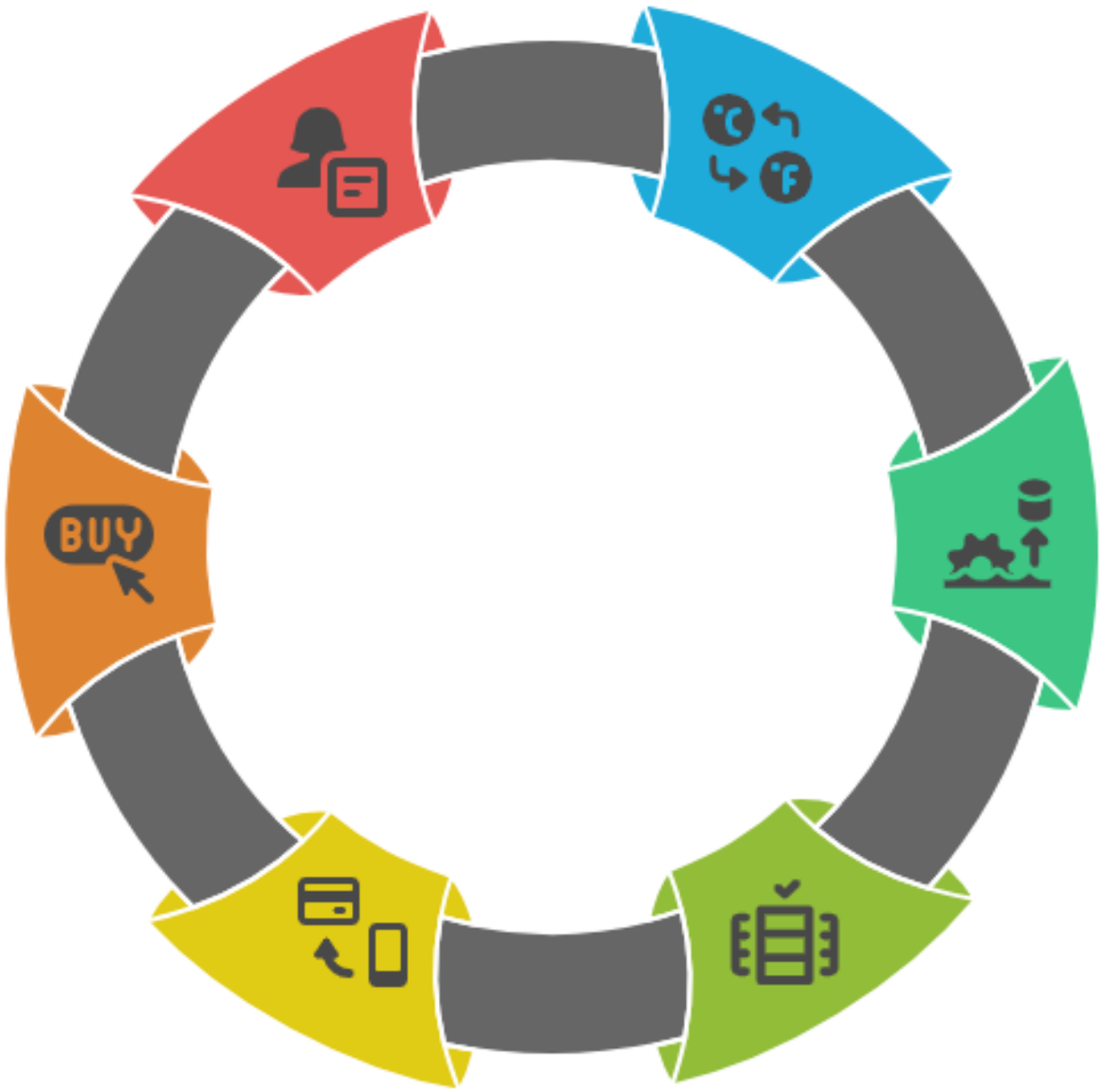
Advantages of SaaS for Providers

Data Collection

Insights from user data

Cash Flow

Predictable revenue streams



Try Before You Buy

Trial periods for users

Update Management

Automated software updates

Payment Flexibility

Customizable payment options

Continuous Deployment

Frequent feature releases

Benefits of SaaS for software product providers

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.

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. (WHEN CAN THIS HAPPEN EVEN AFTER USING CLOUD?)** 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.

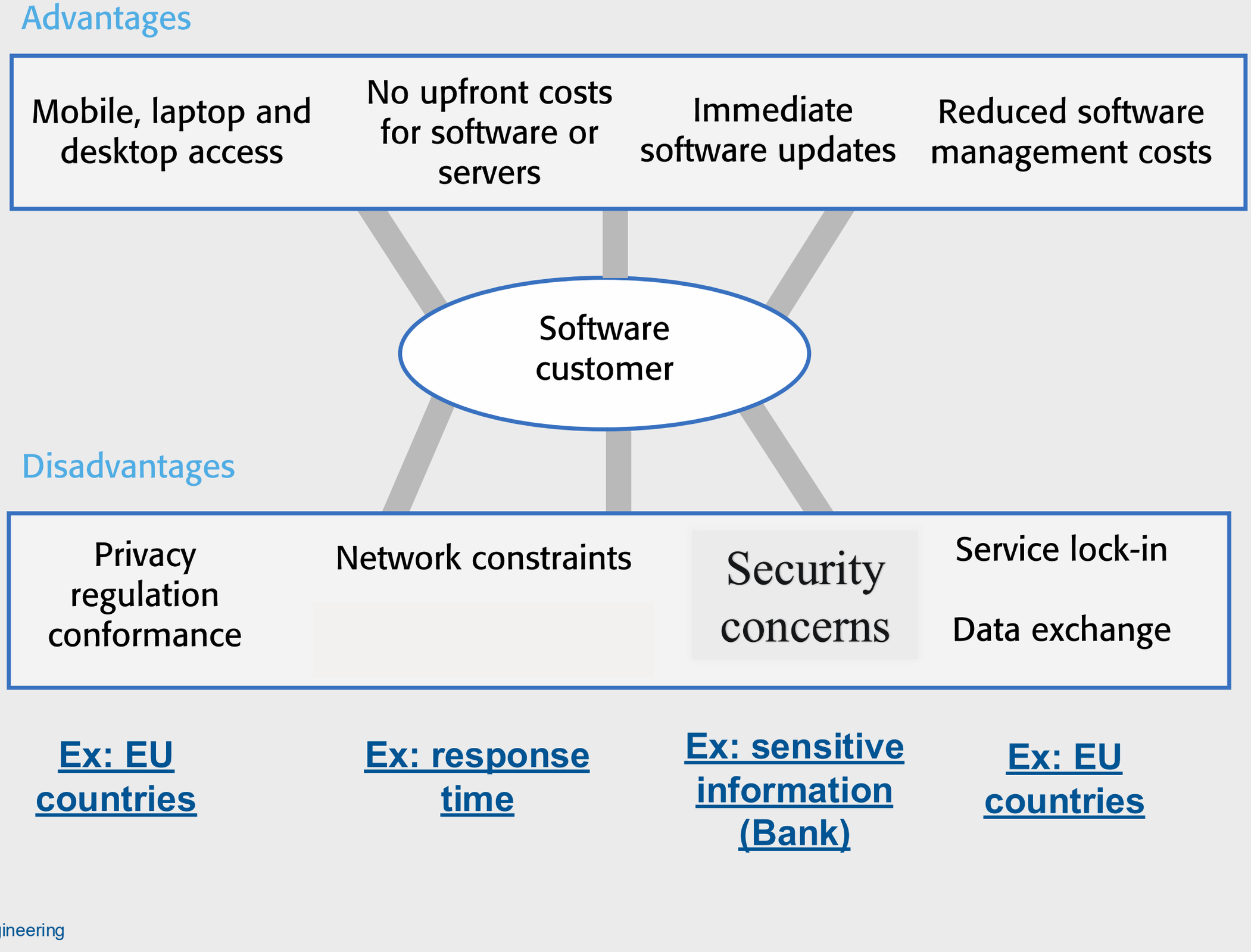
Benefits of SaaS for software product providers

Payment flexibility
You can have several different payment options.

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.

Advantages and disadvantages of SaaS for customers



Design issues for software delivered as a service

If the system will be used in another company?

?

Local/remote processing

Network Traffic vs. Processing Power

Authentication

Own Auth
External Auth (Google)
Business: Federated Auth (College)

SaaS design issues

?

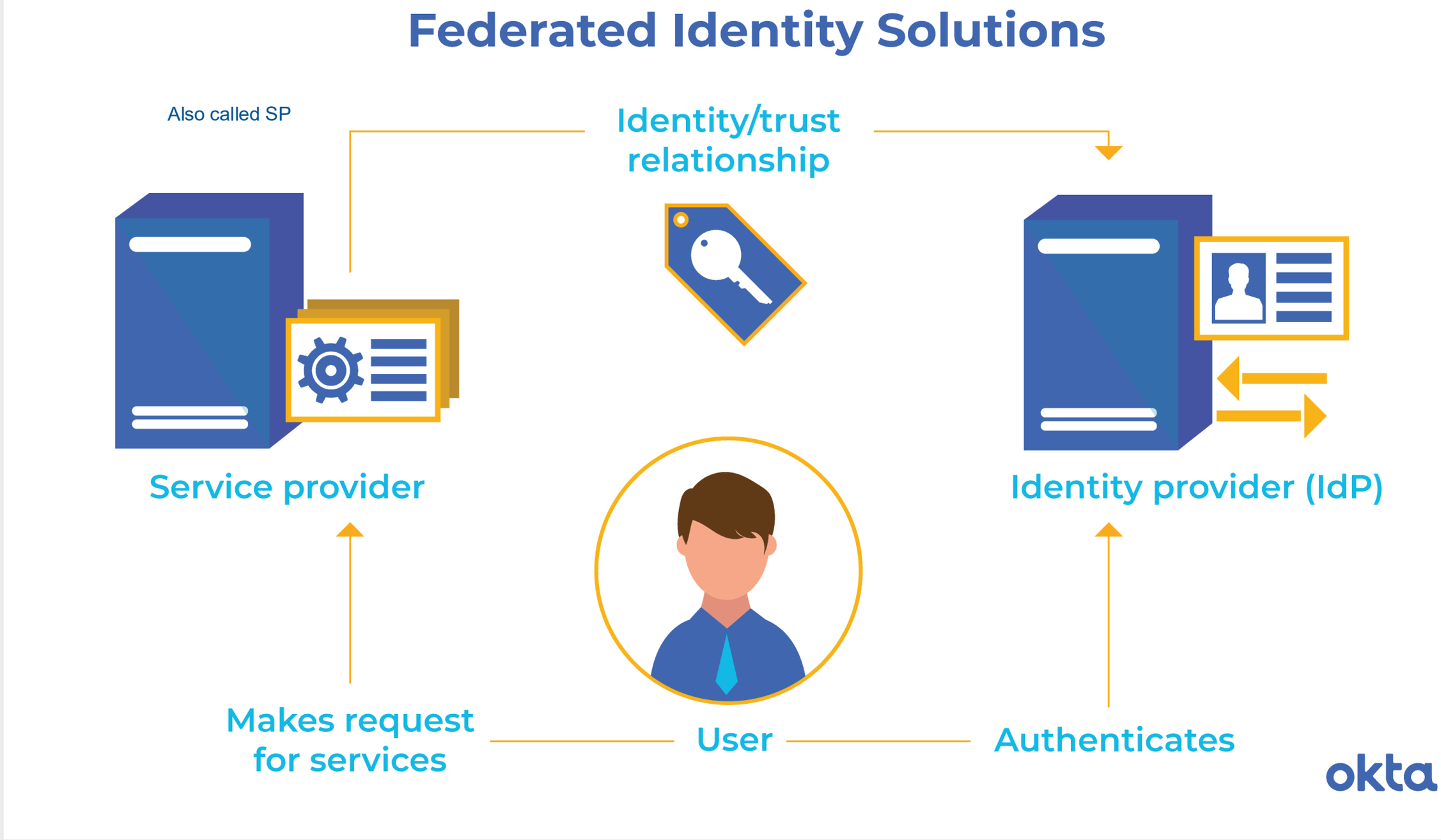
Information leakage

multiple users from multiple organizations

Multitenant or multi-instance database management

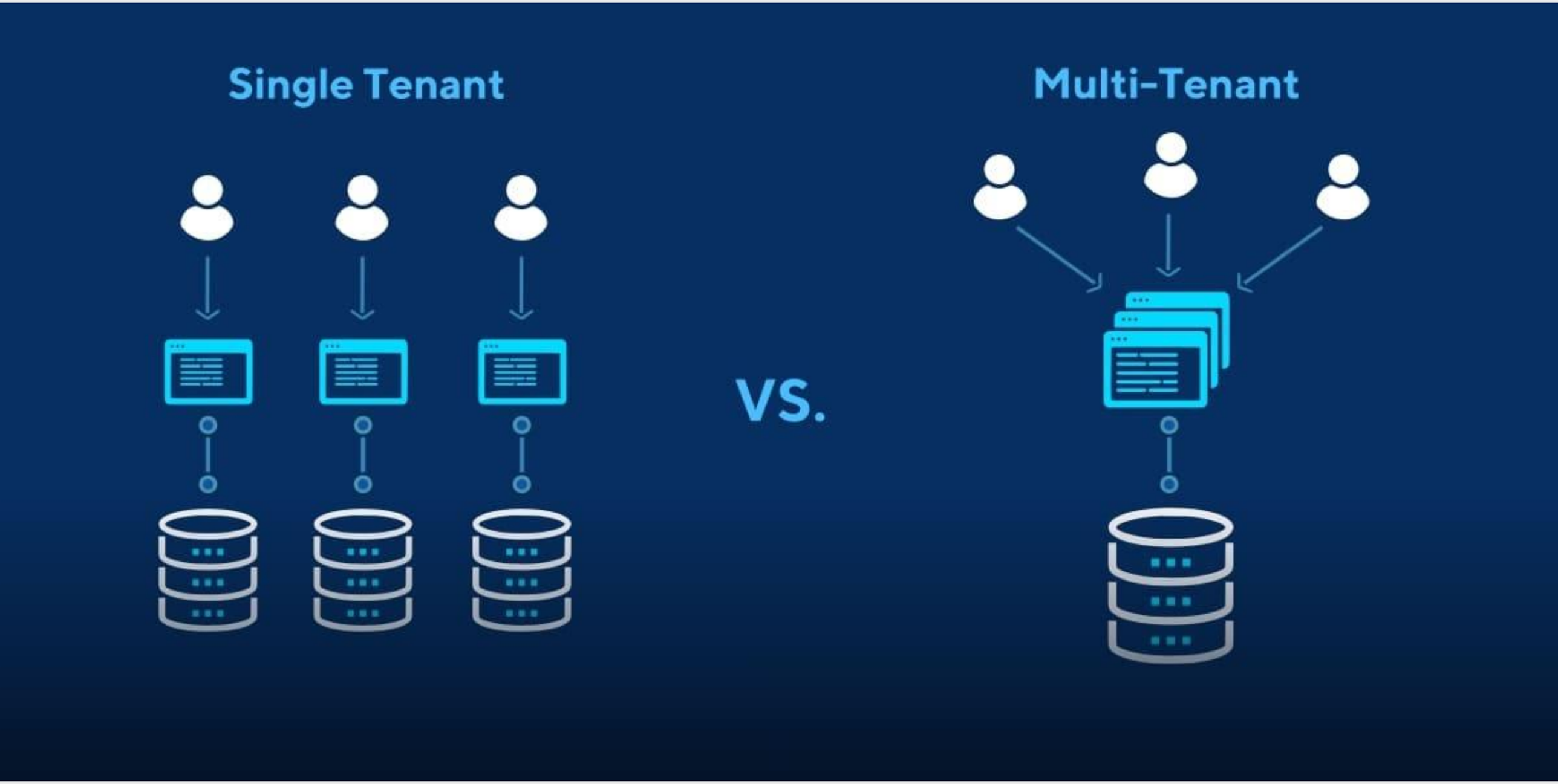
Multitenant (Single instance): One software instance & one DB.
Single-tenant (Single Codebase): One software instance & multiple separate DB
Multi-instance: Separate software & DB

Federated Auth



https://www.okta.com/identity-101/what-is-federated-identity/

Single Tenant vs. Multi-tenant



HEADS UP!

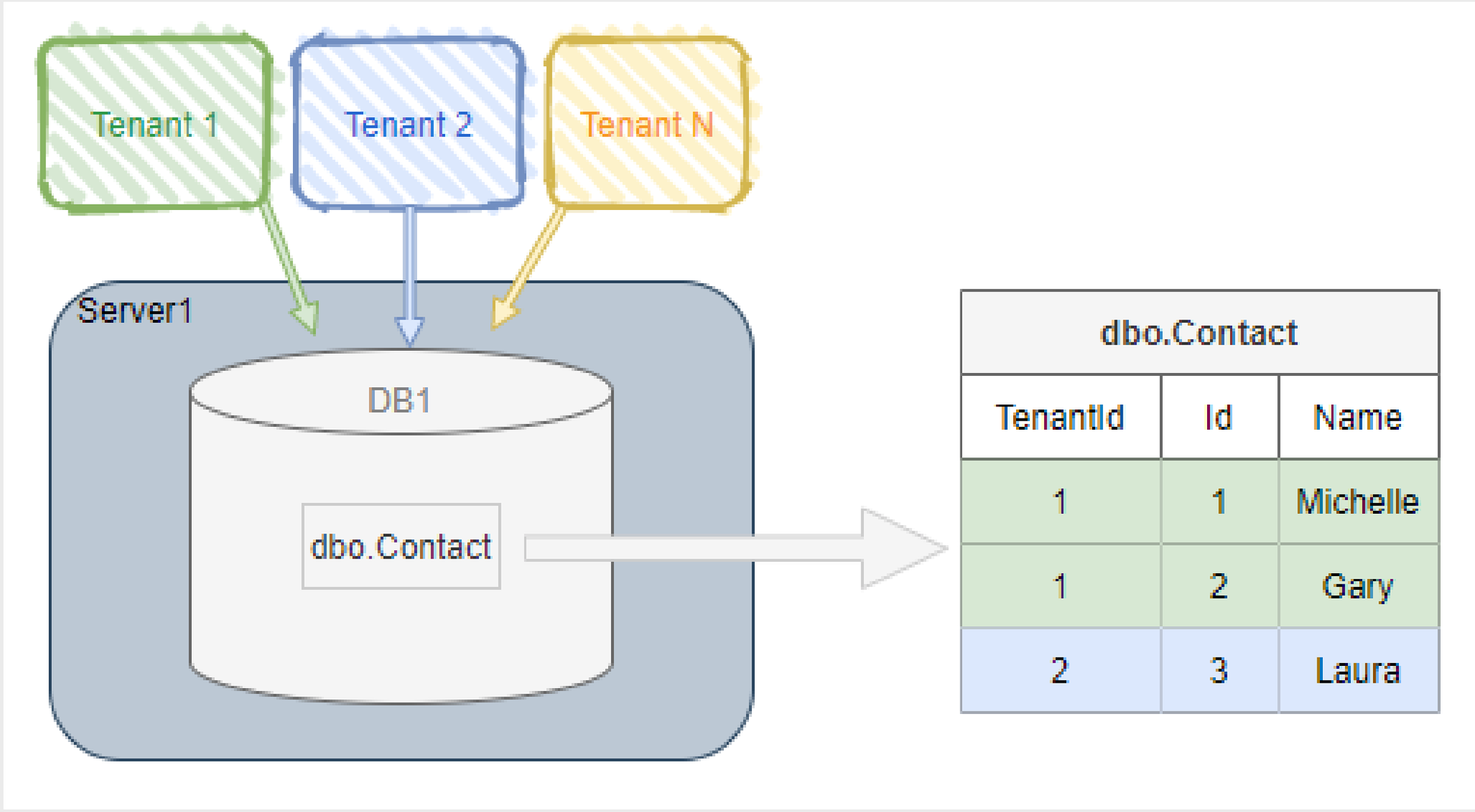
Some points and design factor can be considered both an advantage and disadvantage based on context.

Multi-tenant systems

- 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 of 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.

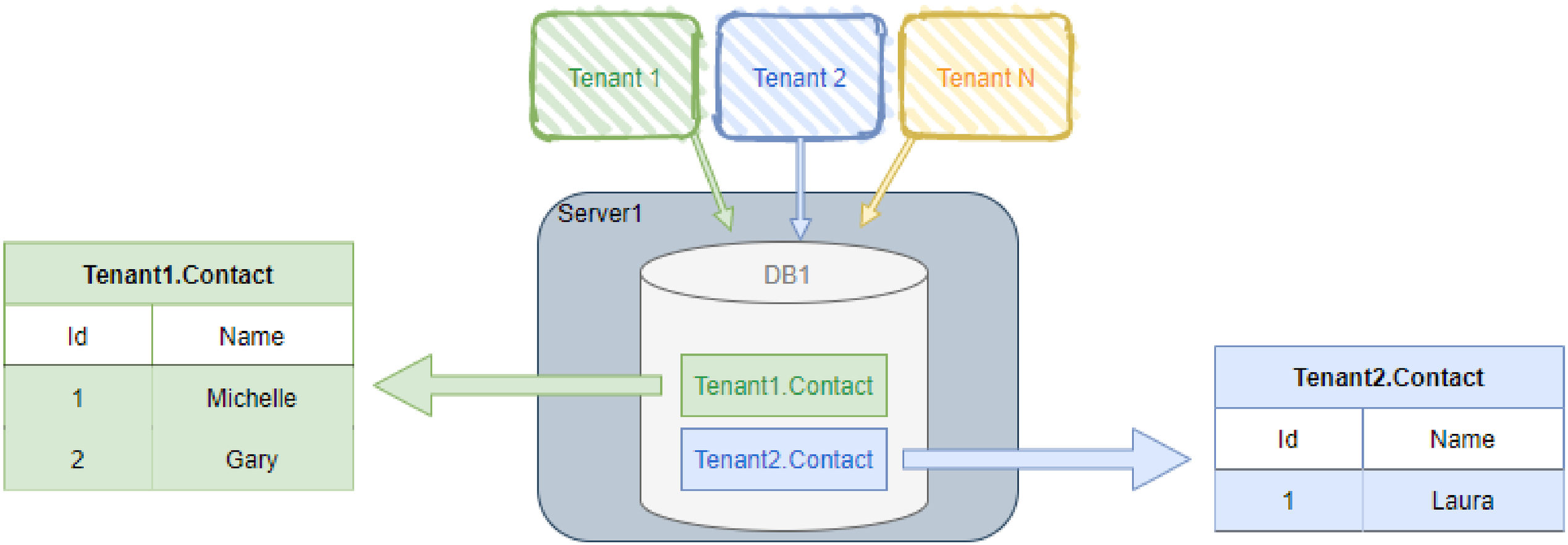
Multi-tenant systems (Variations)

- **[Basic & Text Book] Single database, shared schema**



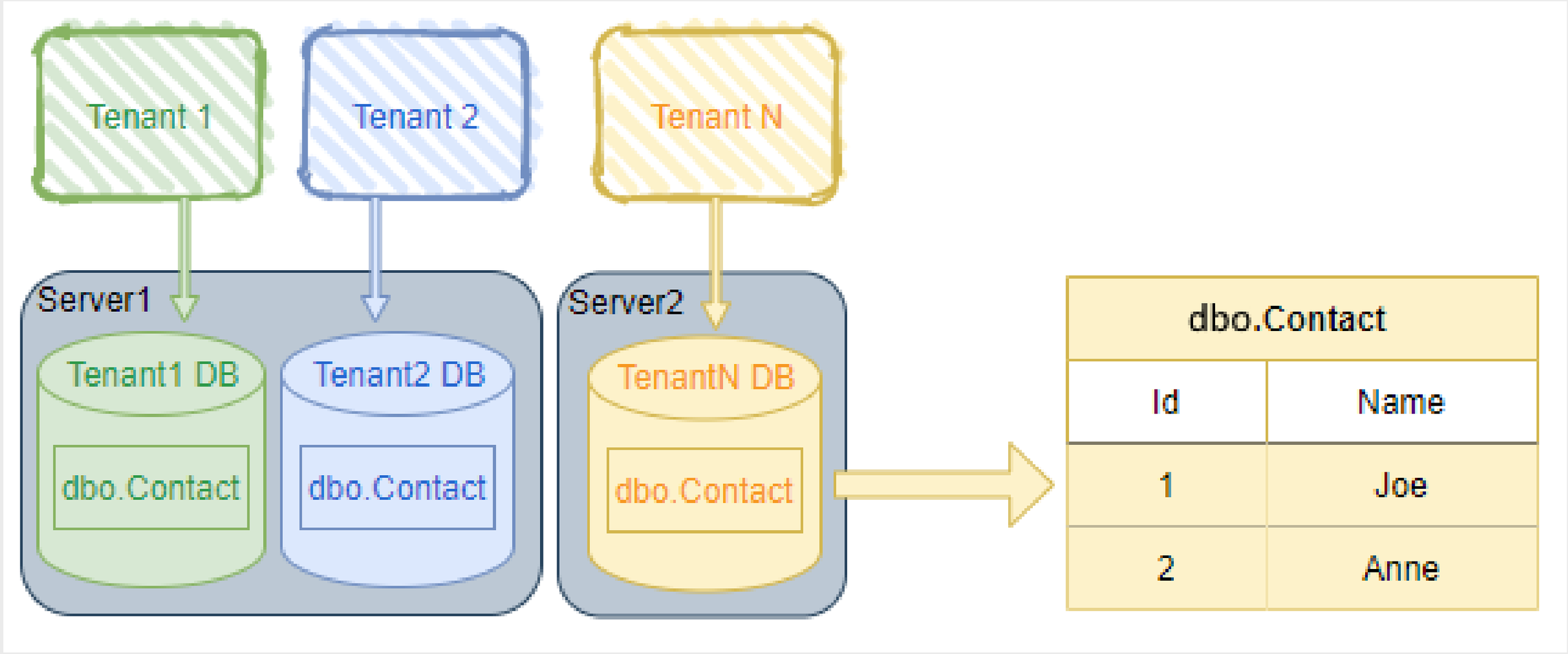
Multi-tenant systems (Variations)

- **Single database, Separate schema**



Single-tenant systems

- **Single database for each software instance**



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
Security management is **simplified** as there is only a single copy of the database software to be patched if a security vulnerability is discovered.

Is that always an advantage?

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**. I explain possible database adaptations later in this section.

Security

1. Same DB: possibility that data will leak from one customer to another.
2. More seriously, perhaps, if there is a database **security breach then it affects all customers**.

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.

Possible customisations for SaaS

Branding
Businesses may want a user interface that is branded to reflect their own organisation.

- White Labeling: A company can even buy the right to sell the system with its name (brand) on it.

Business rules
Businesses may want to be able to define their own business rules and workflows that apply to their own data.

Data schemas
Businesses may want to be able to extend the standard data model used in the system database to meet their own business needs.

Customizations for SaaS

You have made a SaaS system for padel reservation system. A company has asked you for their own customized version of the website (or App):

- Branding (logo & Colors)
- Different reservation rules.
- Extend your user model to get more info about customers.
- Coupon Module.

1. Pros and Cons?

   Sort from easier to most difficult.
2. Write your Choice of implementation for each feature:

   Separate Repo? (Different code)

   Same Repo?

   Why?

Adding fields to extend the database

Assume that a customer has asked for more data saved to the profile, coupons, or the product attributes?

Adding fields to extend the database

- You 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. However:
  - How many columns?
  - Too many? Too few?
  - Different customers are likely to need different types of columns.
    - String, Integer or other types?

Sols?

- Extra columns (previously mentioned)
- Text only (can be JSON)

  Problem?
- Another Table

  Problem?

Extending a database using tables

- An alternative approach to database extensibility is to allow customers to add any number of additional fields and to define the names, types and values of these fields.
- The names and types of these values are held in a separate table, accessed using the tenant identifier.
- Unfortunately, using tables in this way adds complexity to the database management software.
  - Extra tables must be managed and information from them integrated into the database.

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

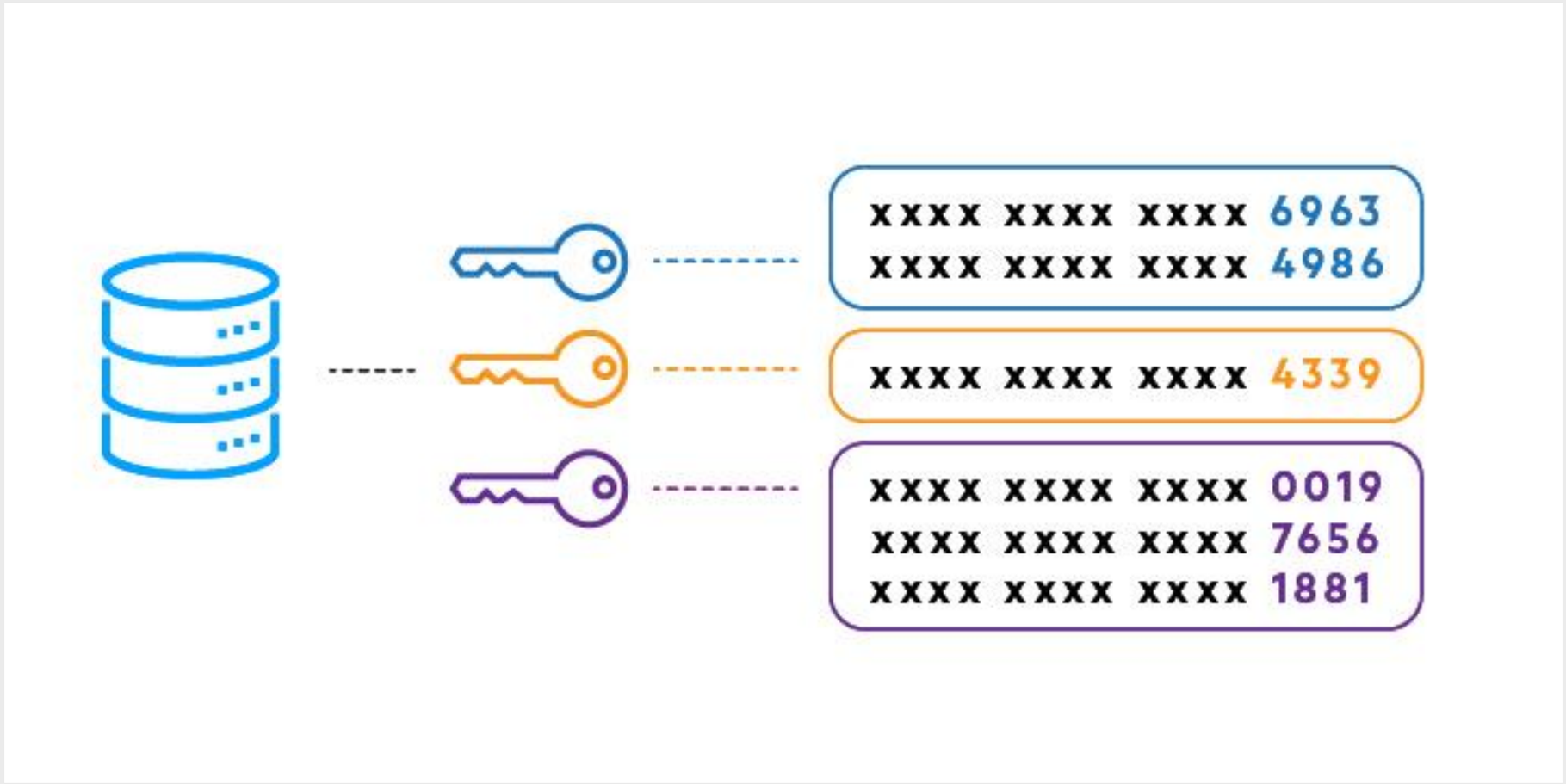
Field values		
Record	Tenant	Value
E123	T516	‘A17/S6’
E123	T516	‘4’
E123	T516	‘False’
E200	T632	‘2017/1/15’
E200	T632	‘Dublin’
E346	T973	‘2017/2/10’
...		

Value table showing the value of extension fields for each record

Database security

- Information from all customers is stored in the same database in a multi-tenant system so a software bug or an attack could lead to the data of some or all customers being exposed to others.
- Key security issues are multilevel access control and encryption.
  - **Multilevel access control** means that access to data must be controlled at both the
    - **organizational level: Within company**
    - **individual level: User-based role**
- 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-tenant DB Encryption



https://baffle.io/blog/multi-tenant-data-security/

Multi-instance databases

- 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.

Advantages of multi-instance databases

Flexibility
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.

Security
Each customer has their own database so there is **no possibility of data leakage from one customer to another**.

Scaleability
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.

Resilience
If a software failure occurs, this **will probably only affect a single customers**. Other customers can continue working as normal.

Disadvantages of multi-instance databases

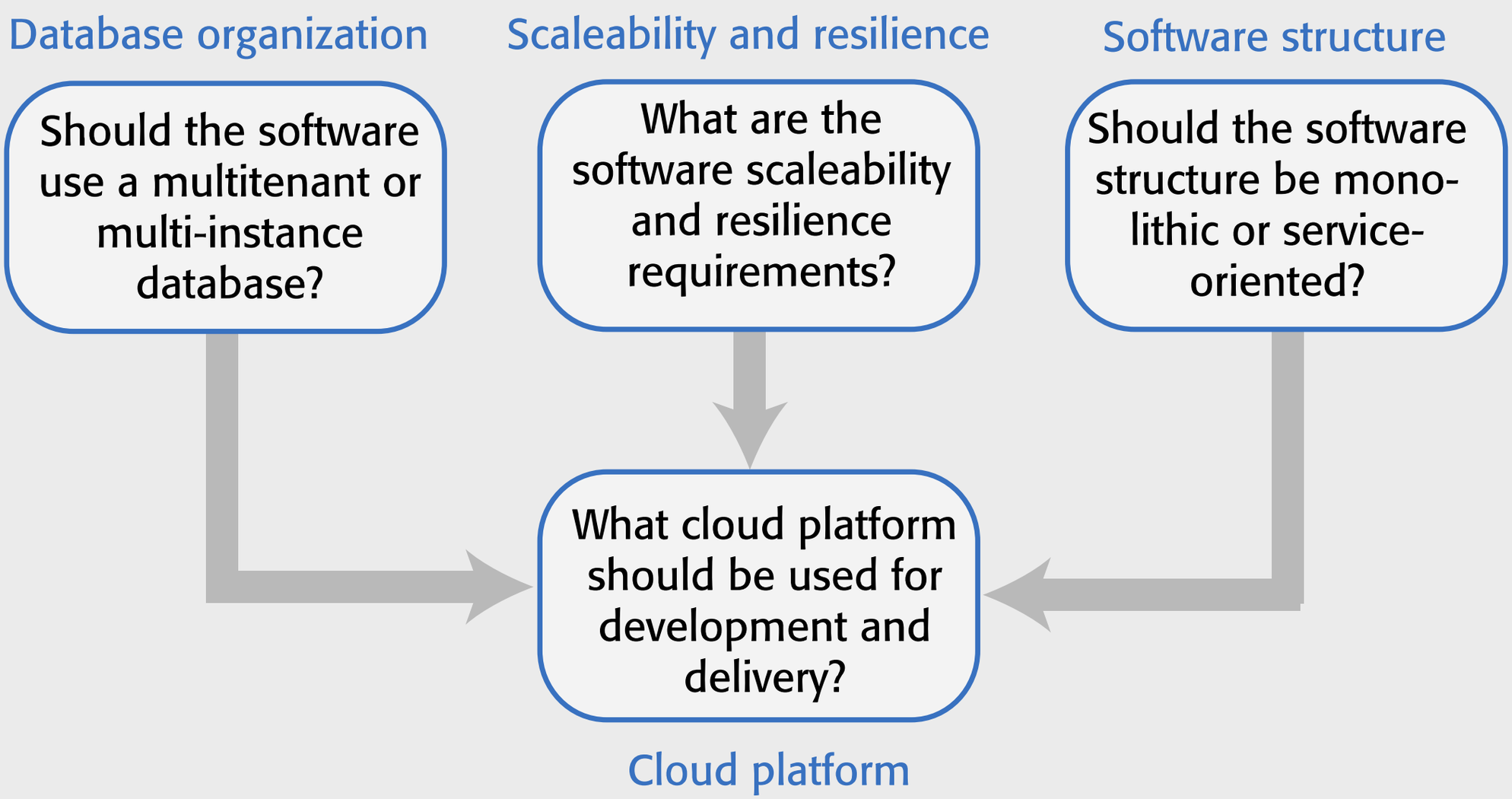
Cost

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.

Update management

It is more complex to manage updates

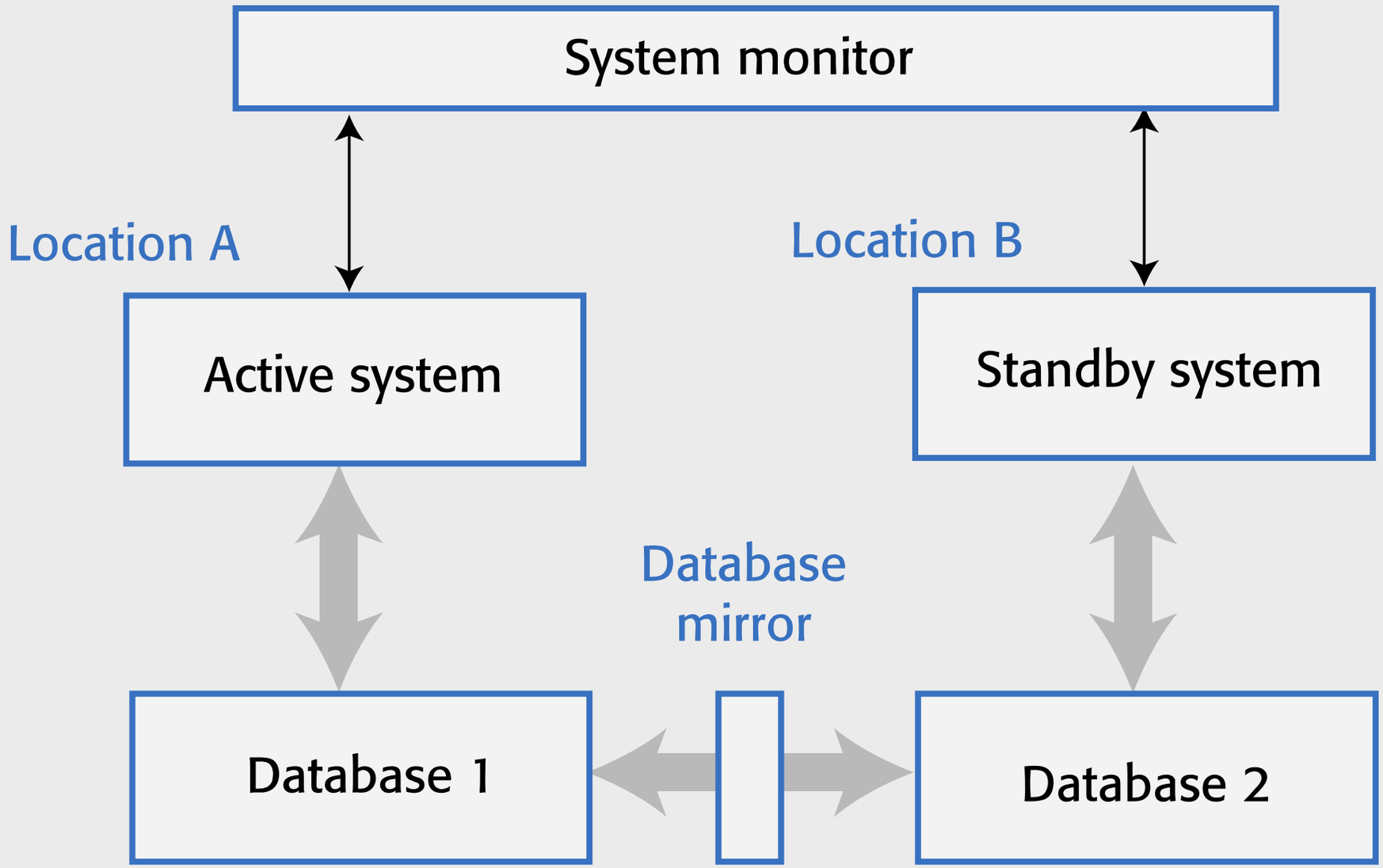
Architectural decisions for cloud software engineering



Scalability and resilience

- The scaleability 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 scaleability in a system by making it possible to add new virtual servers (**scaling-out [Horizontl Scaling]**) or increase the power of a system server (**scaling-up [Vertical Scaling]**) 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

- 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 operational system fails.
- You should use redundant virtual servers that are not hosted on the same physical computer and locate servers in different locations. **Why?**

Key points 1

- The cloud is made up of a large number of virtual servers that you can rent for your own use. You and your customers access these servers remotely over the internet and pay for the amount of server time used.
- Virtualization is a technology that allows multiple server instances to be run on the same physical computer. This means that you can create isolated instances of your software for deployment on the cloud.
- 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 allow rapid replication and deployment of virtual servers. All containers run the same operating system. Docker is currently the most widely used container technology.
- A fundamental feature of the cloud is that ‘everything’ can be delivered as a service and accessed over the internet. A service is rented rather than owned and is shared with other users.

Key points 2

- Infrastructure as a service (IaaS) means computing, storage and other services are available over the cloud. There is no need to run your own physical servers.
- Platform as a service (PaaS) means using services provided by a cloud platform vendor to make it possible to auto-scale your software in response to demand.
- Software as a service (SaaS) means that application software is delivered as a service to users. This has important benefits for users, such as lower capital costs, and software vendors, such as simpler deployment of new software releases.
- Multitenant systems are SaaS systems where all users share the same database, which may be adapted at run-time to their individual needs. Multi-instance systems are SaaS applications where each user has their own separate database.
- The key architectural issues for cloud-based software are the cloud platform to be used, whether to use a multitenant or multi-instance database, the scaleability and resilience requirements, and whether to use objects or services as the basic components in the system.