Software Engineering

Cloud-Based Software:
Virtualization and containers, Software as a service; Cloud Computing and Cloud Software Architecture
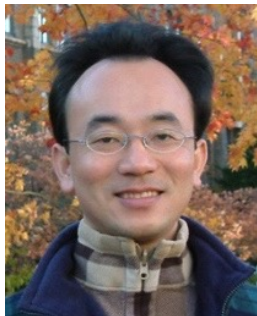
Min-Yuh Day, Ph.D,
Professor

Institute of Information Management, National Taipei University

https://web.ntpu.edu.tw/~myday

https://meet.google.com/
ish-gzmy-pmo

2024-04-22

Syllabus

Week   Date   Subject/Topics

1 2024/02/21 Introduction to Software Engineering

2 2024/02/28 Peace Memorial Day (Day Off)

3 2024/03/06 Software Products and Project Management:
               Software product management and prototyping

4 2024/03/13 Agile Software Engineering: Agile methods, Scrum,
               and Extreme Programming

5 2024/03/20 Case Study on Software Engineering I

6 2024/03/27 Features, Scenarios, and Stories

7 2024/04/03 Make-up holiday for NTPU Sports Day (No Classes)

8 2024/04/10 Midterm Project Report

Syllabus

Week   Date   Subject/Topics

9 2024/04/17 Software Architecture: Architectural design, System decomposition, and Distribution architecture

10 2024/04/24 Cloud-Based Software: Virtualization and containers, Everything as a service, Software as a service; Cloud Computing and Cloud Software Architecture

11 2024/05/01 Case Study on Software Engineering II

12 2024/05/08 Microservices Architecture, RESTful services, Service deployment

Syllabus

Week   Date   Subject/Topics

13 2024/05/15 Security and Privacy; Reliable Programming;
Testing: Functional testing, Test automation,
Test-driven development, and Code reviews;
DevOps and Code Management:
Code management and DevOps automation

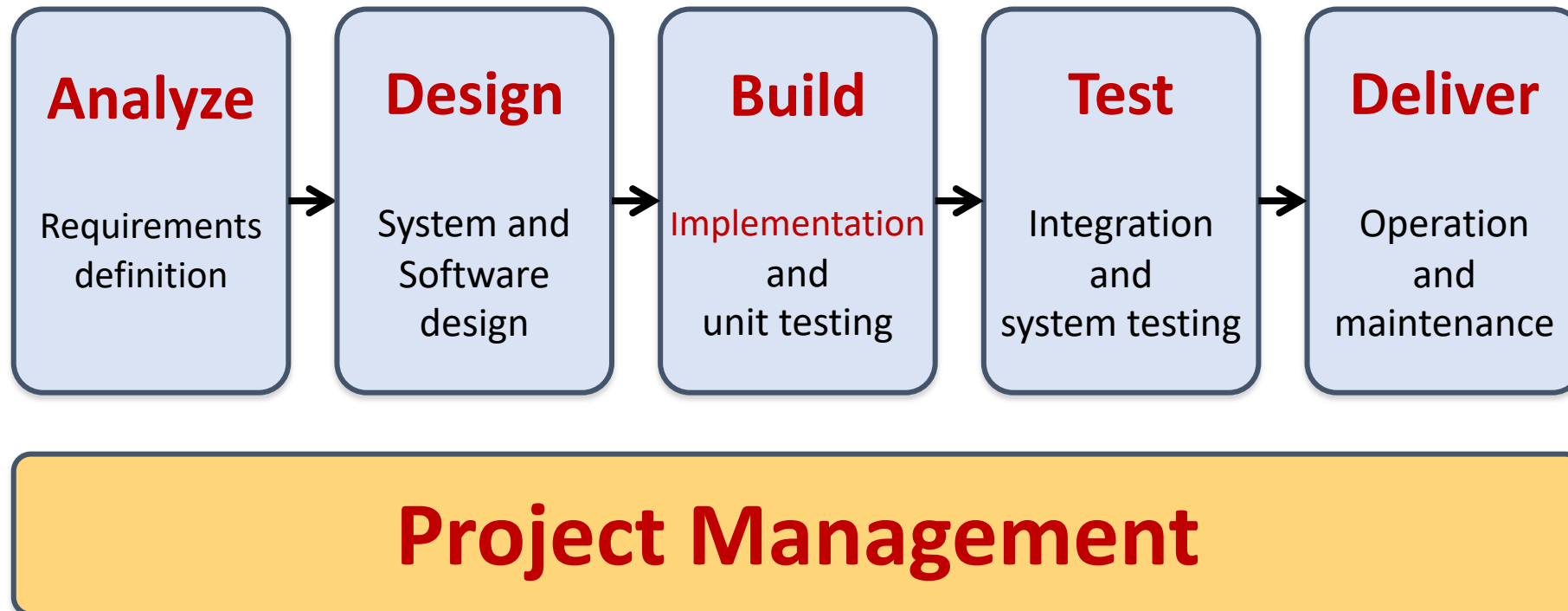14 2024/05/22 Industry Practices of Software Engineering

15 2024/05/29 Final Project Report I

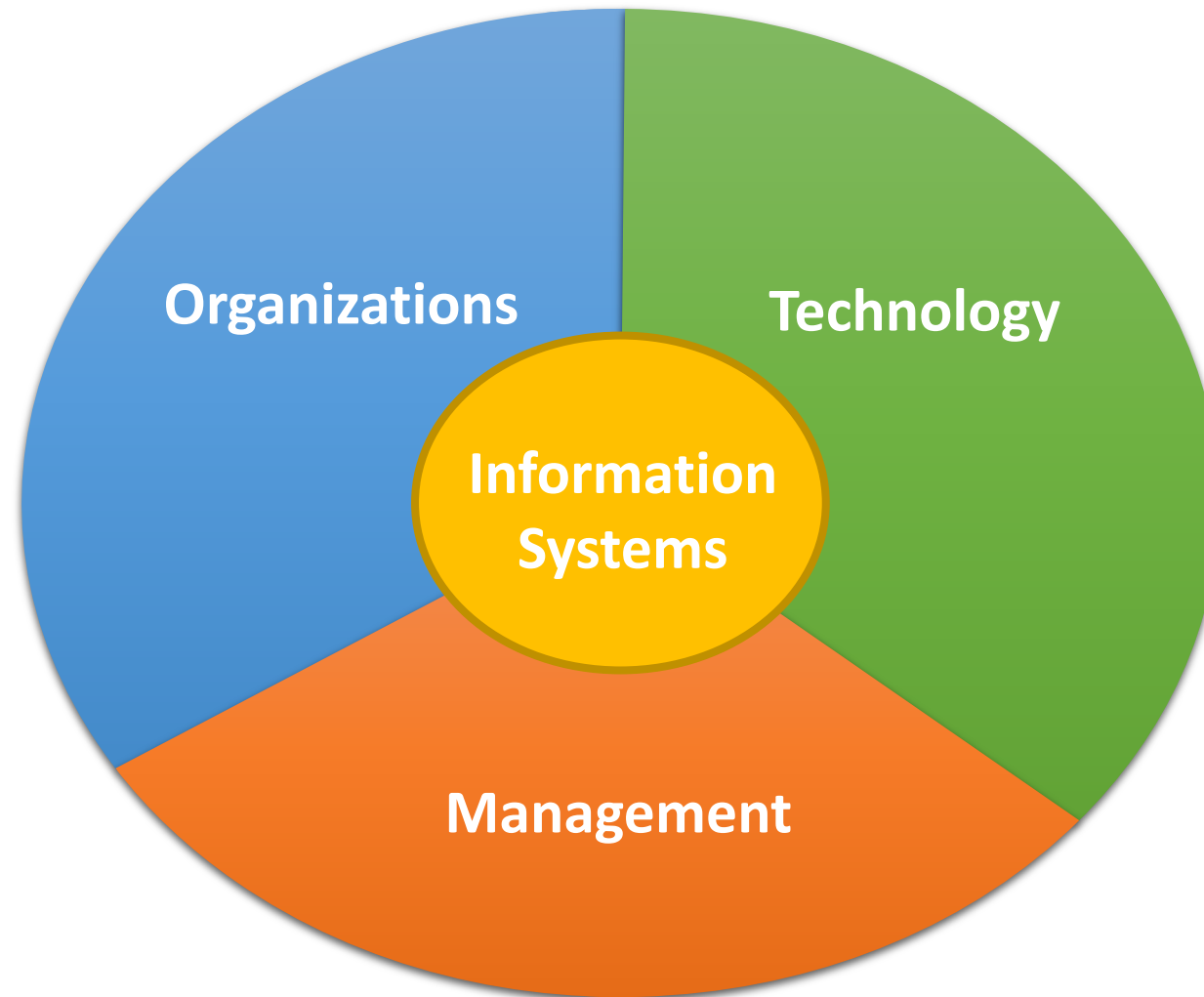16 2024/06/05 Final Project Report II

Cloud-Based Software:
Virtualization and containers,
Everything as a service,
Software as a service;
Cloud Computing and
Cloud Software Architecture

Software Engineering and Project Management

Analyze	Design	Build	Test	Deliver
Requirements definition	System and Software design	Implementation and unit testing	Integration and system testing	Operation and maintenance

Project Management

Information Management (MIS) Information Systems

Source: Kenneth C. Laudon & Jane P. Laudon (2014), Management Information Systems: Managing the Digital Firm, Thirteenth Edition, Pearson.
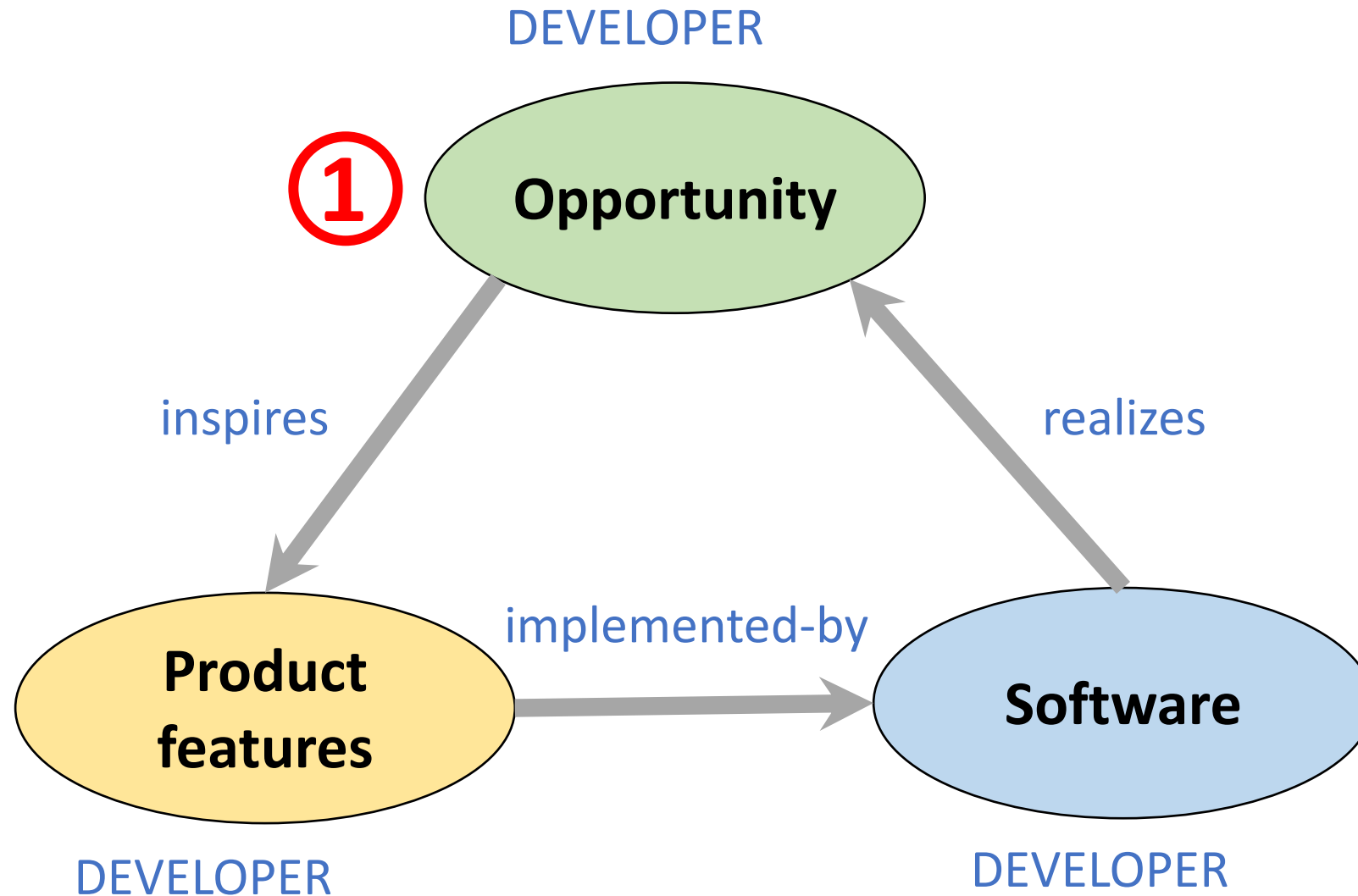
Fundamental MIS Concepts

8
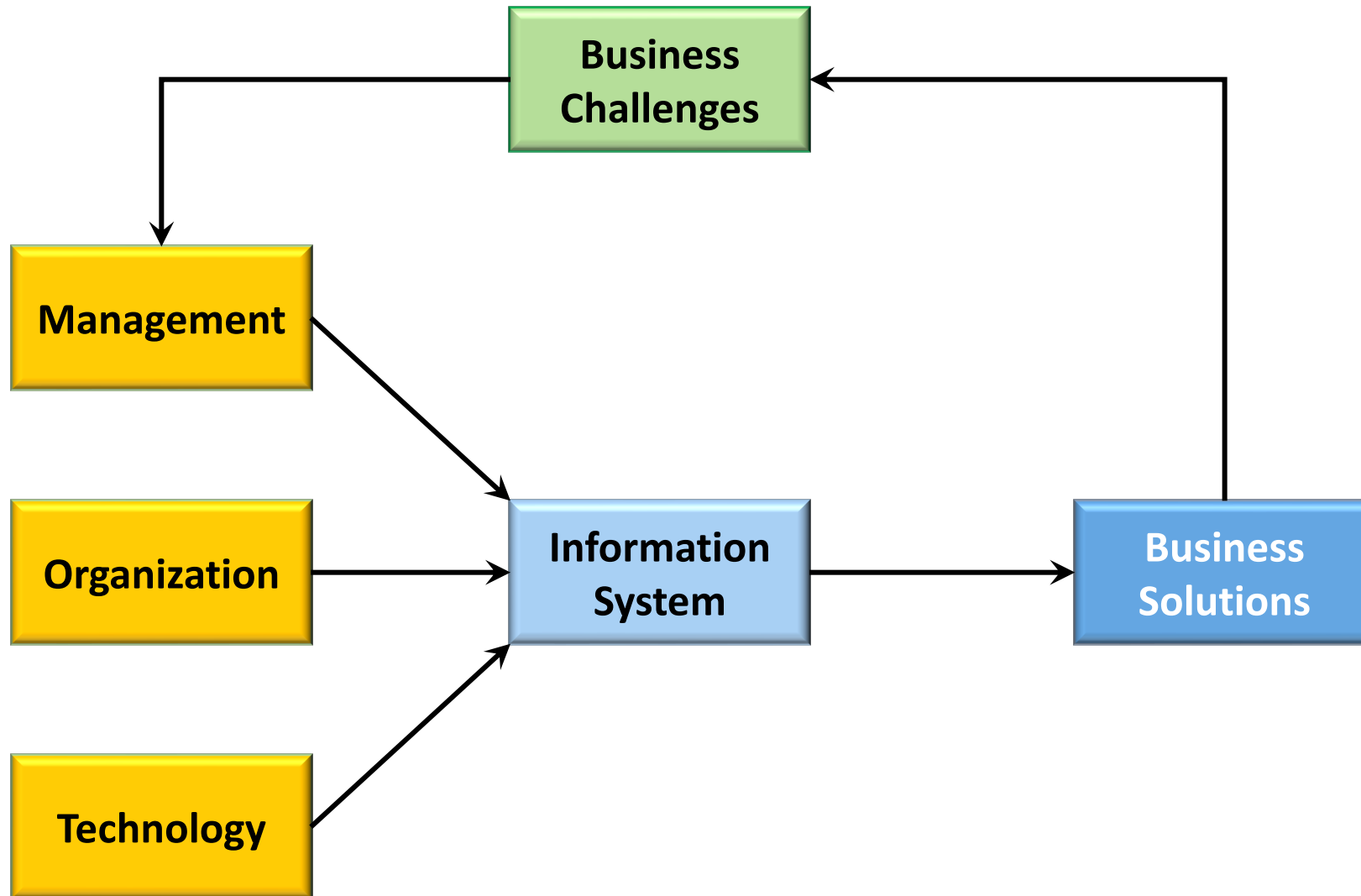
Project-based software engineering
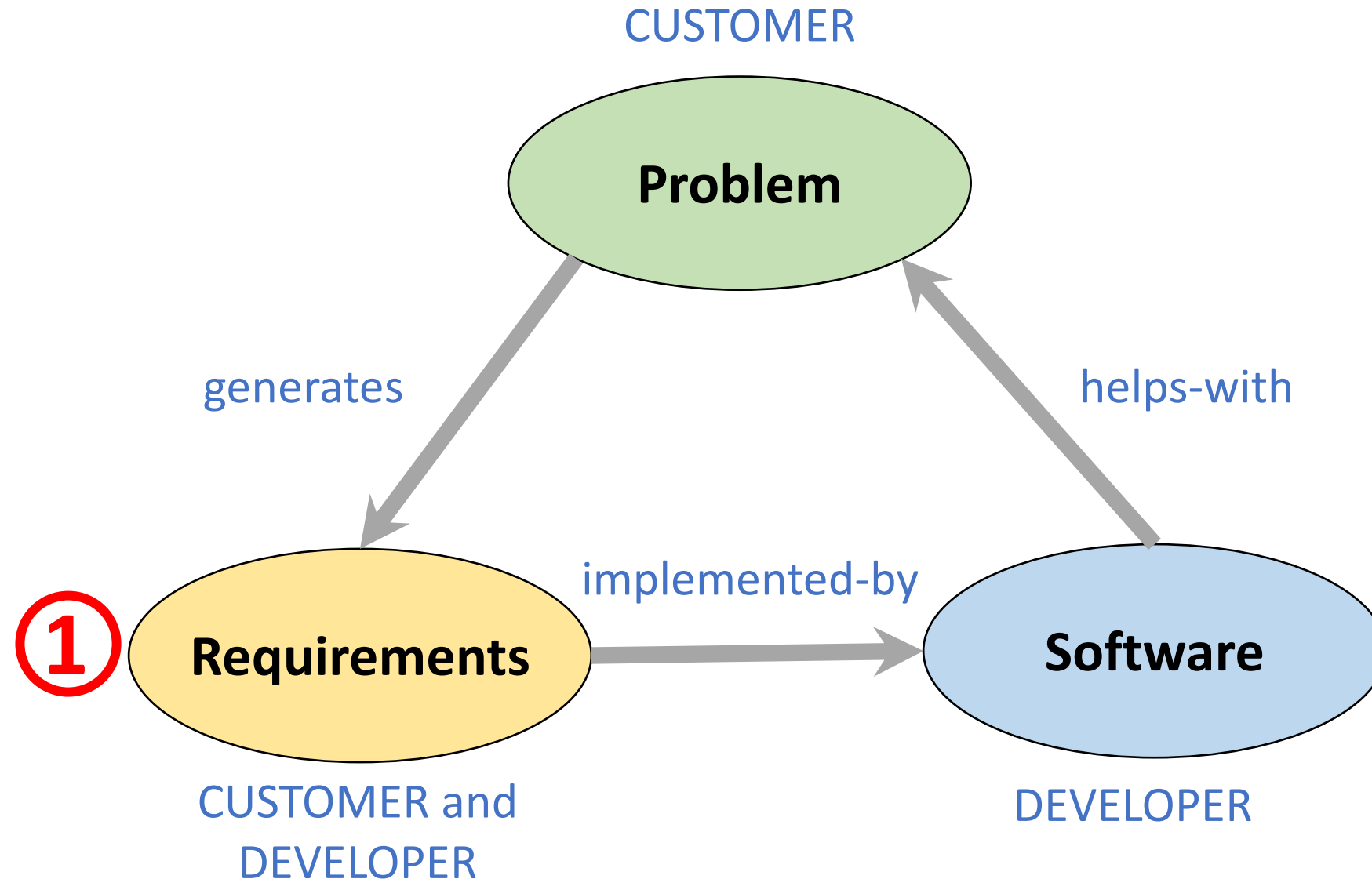
Product software engineering

Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Software execution models

Stand-alone execution

User's computer

User interface
Product functionality
User data

Product updates

Vendor's servers

Hybrid execution

User's computer

User interface
Partial functionality
User data

Additional functionality
User data backups
Product updates

Vendor's servers

Software as a service

User's computer

User interface
(browser or app)

Product functionality
User data

Vendor's servers
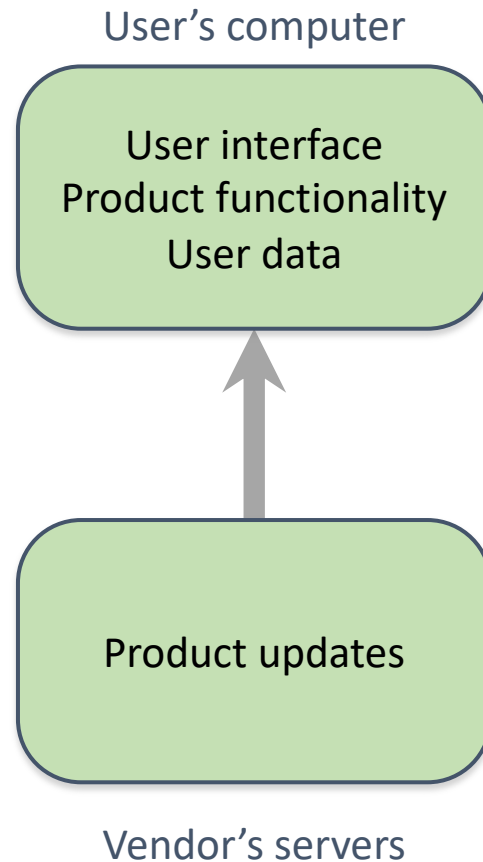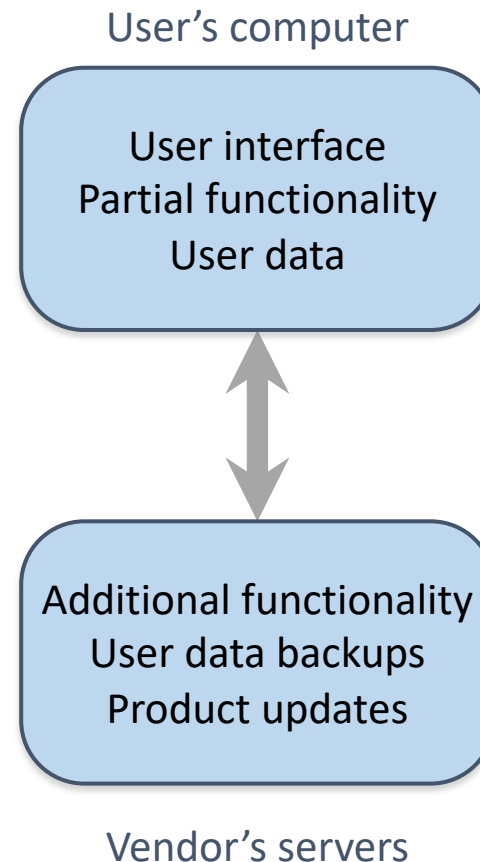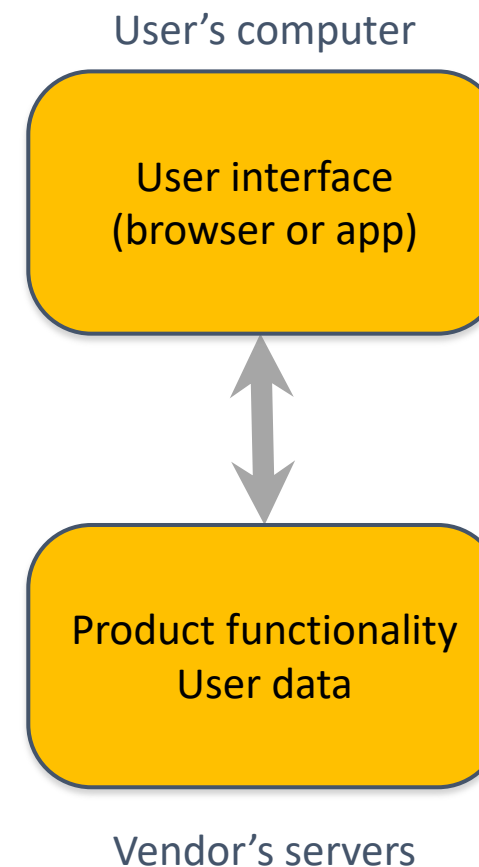
Product management concerns

Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.
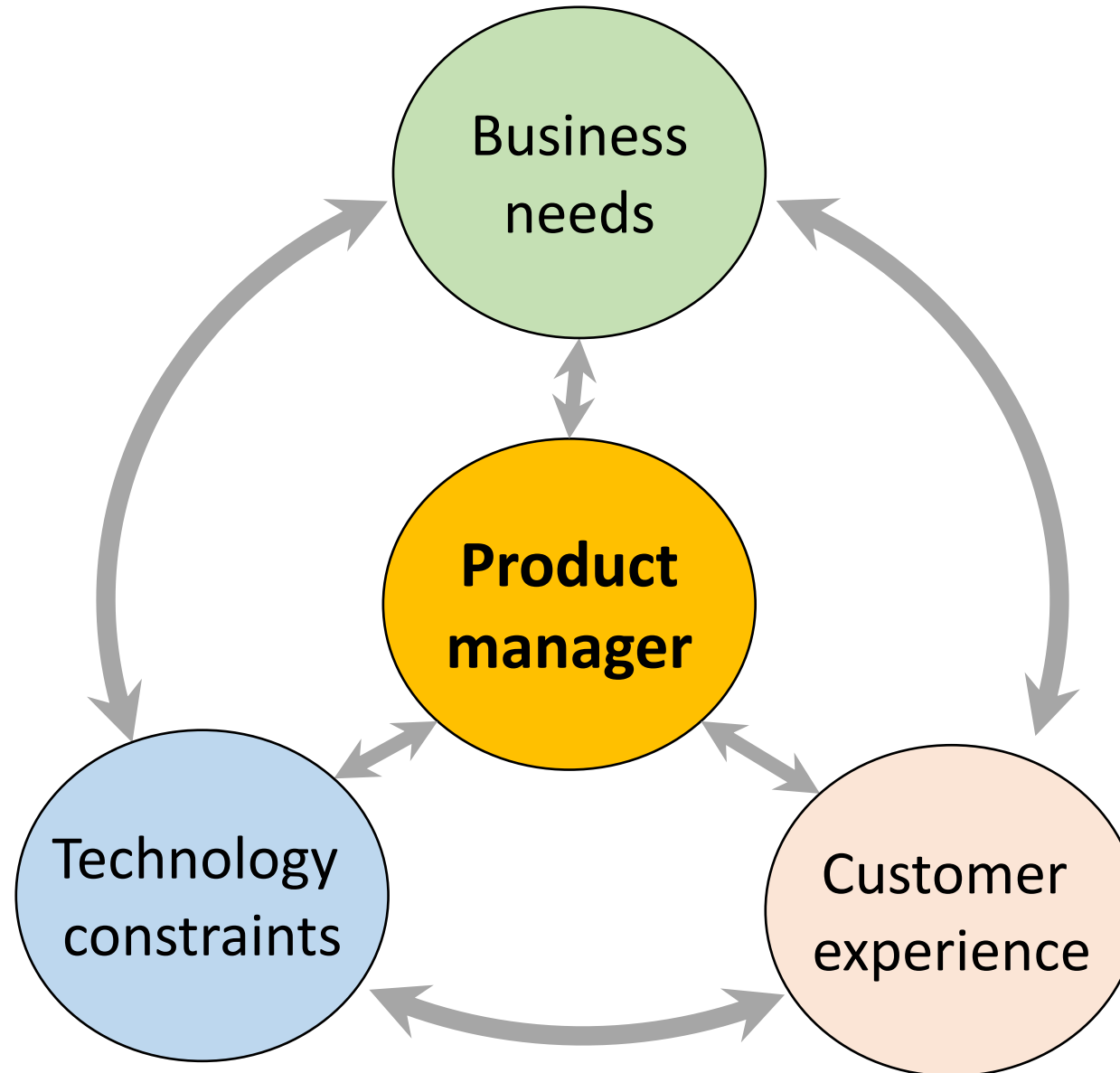
Technical interactions of product managers
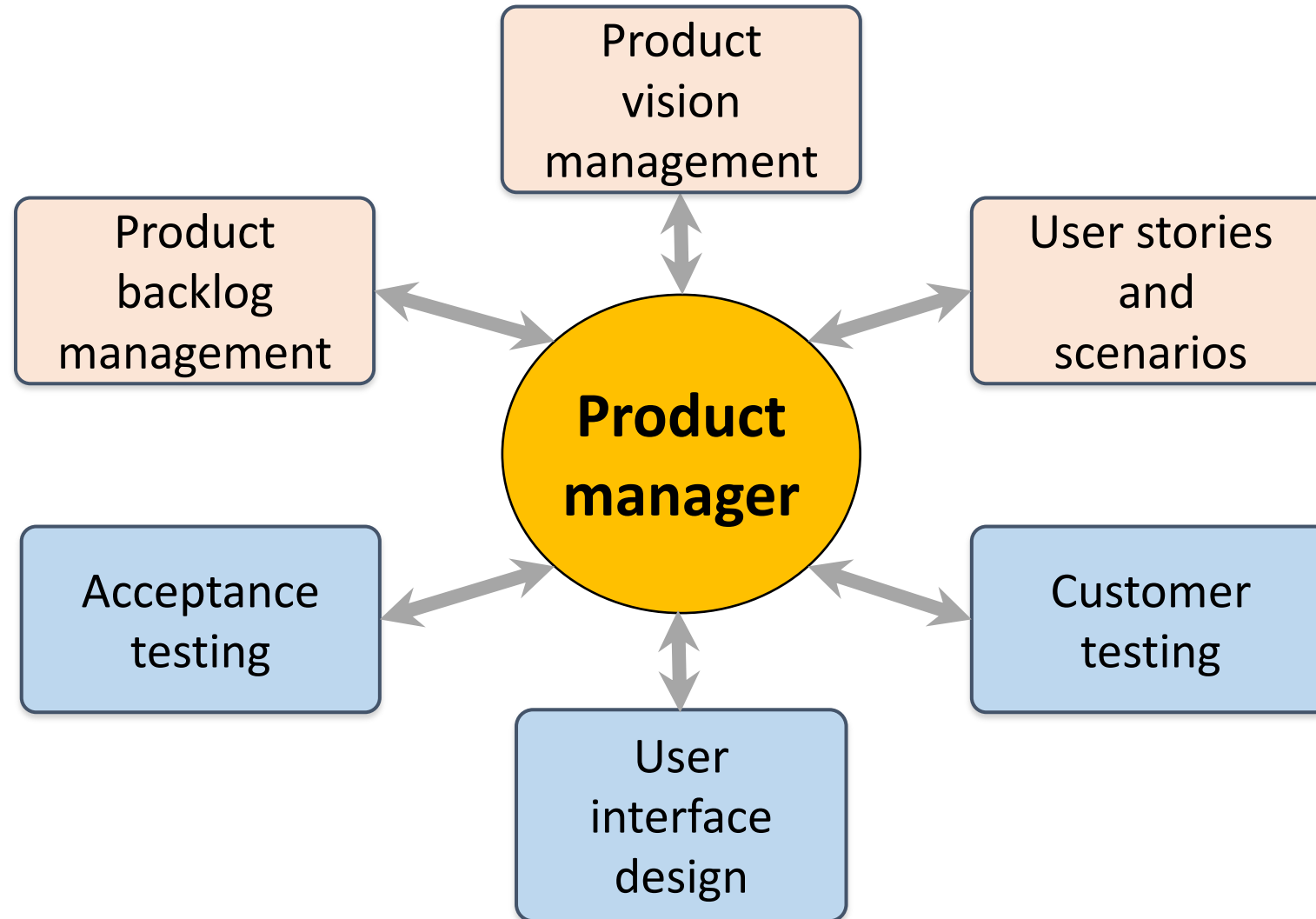
Software Development Life Cycle (SDLC)
The waterfall model
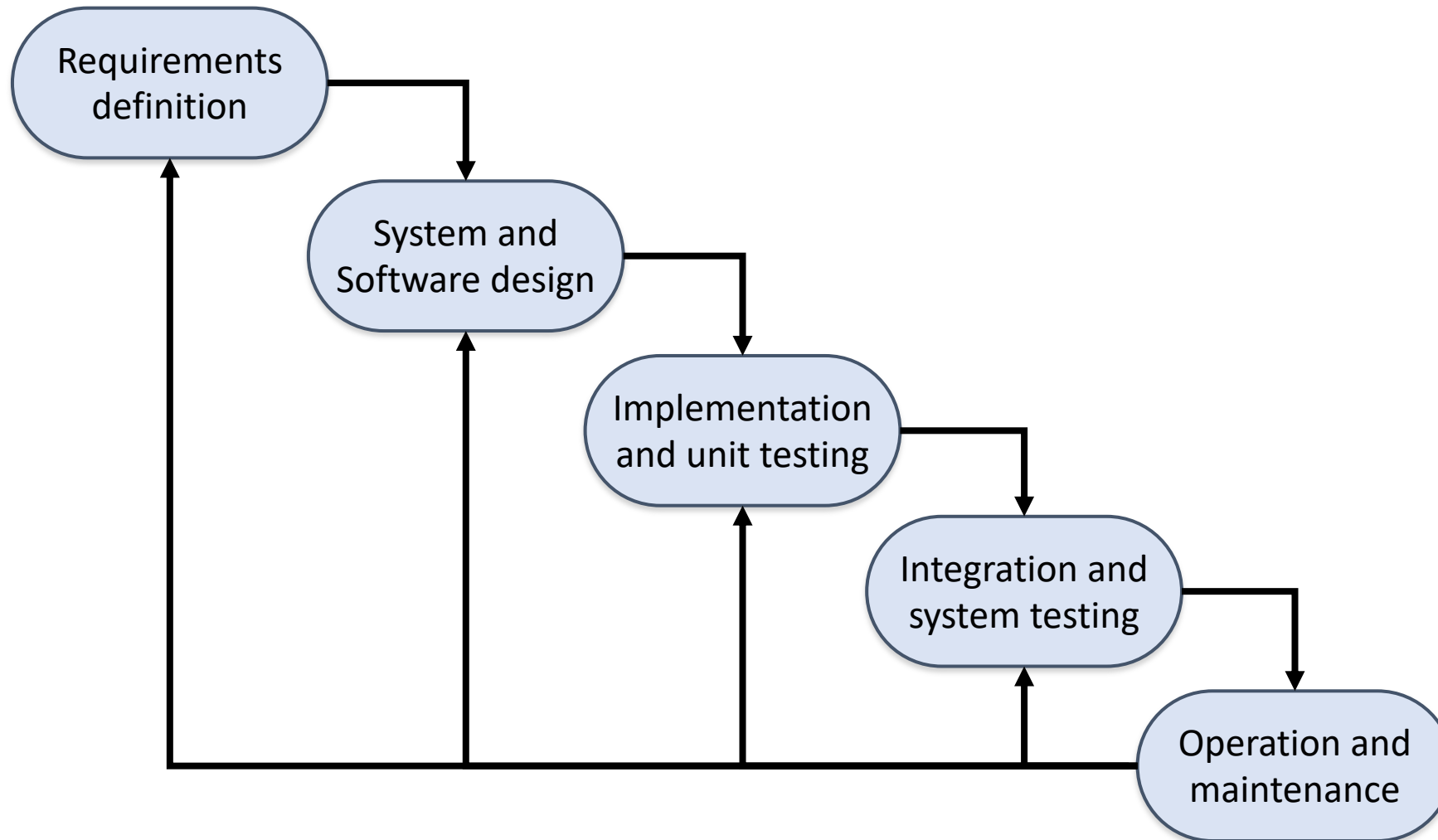
Plan-based and Agile development
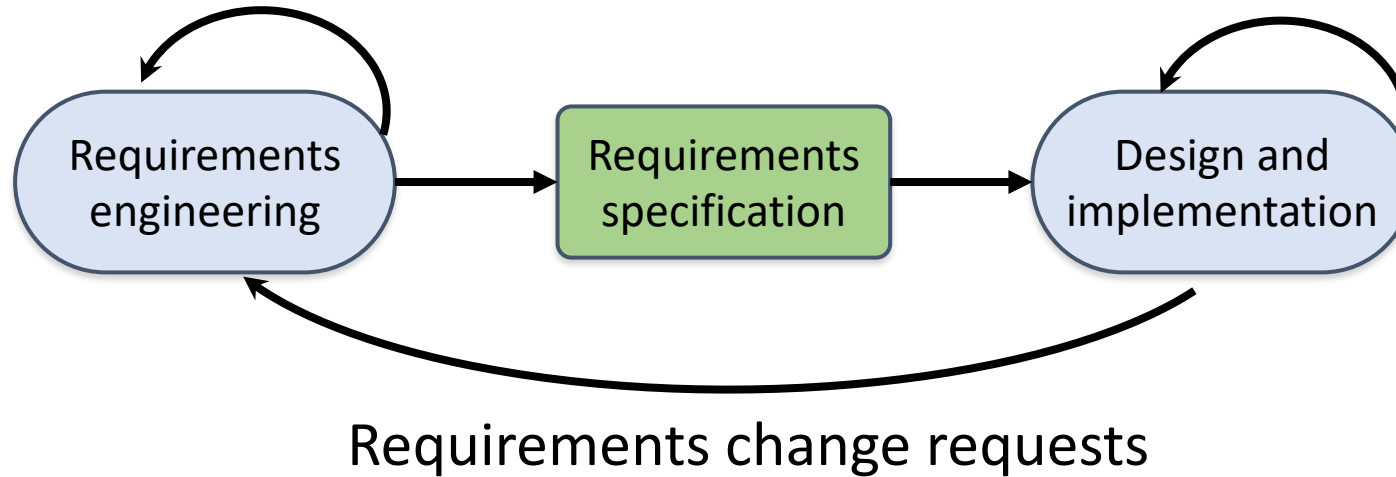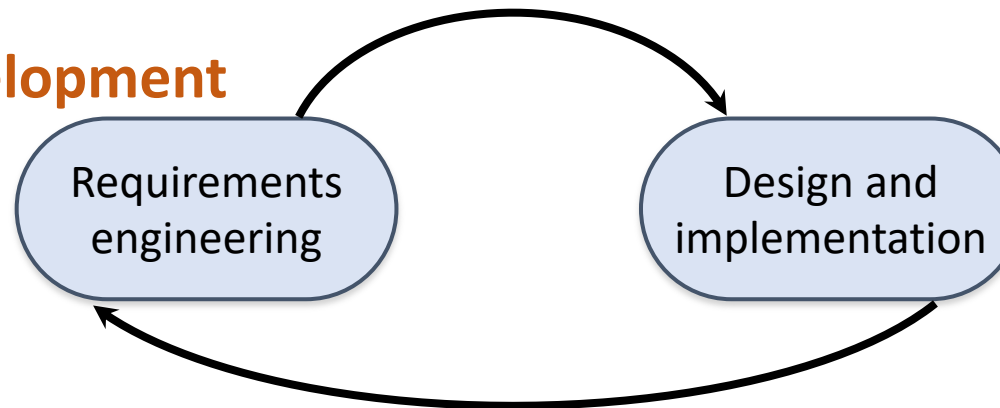
The Continuum of Life Cycles

Predictive Life Cycle

Analyze → Design → Build → Test → Deliver
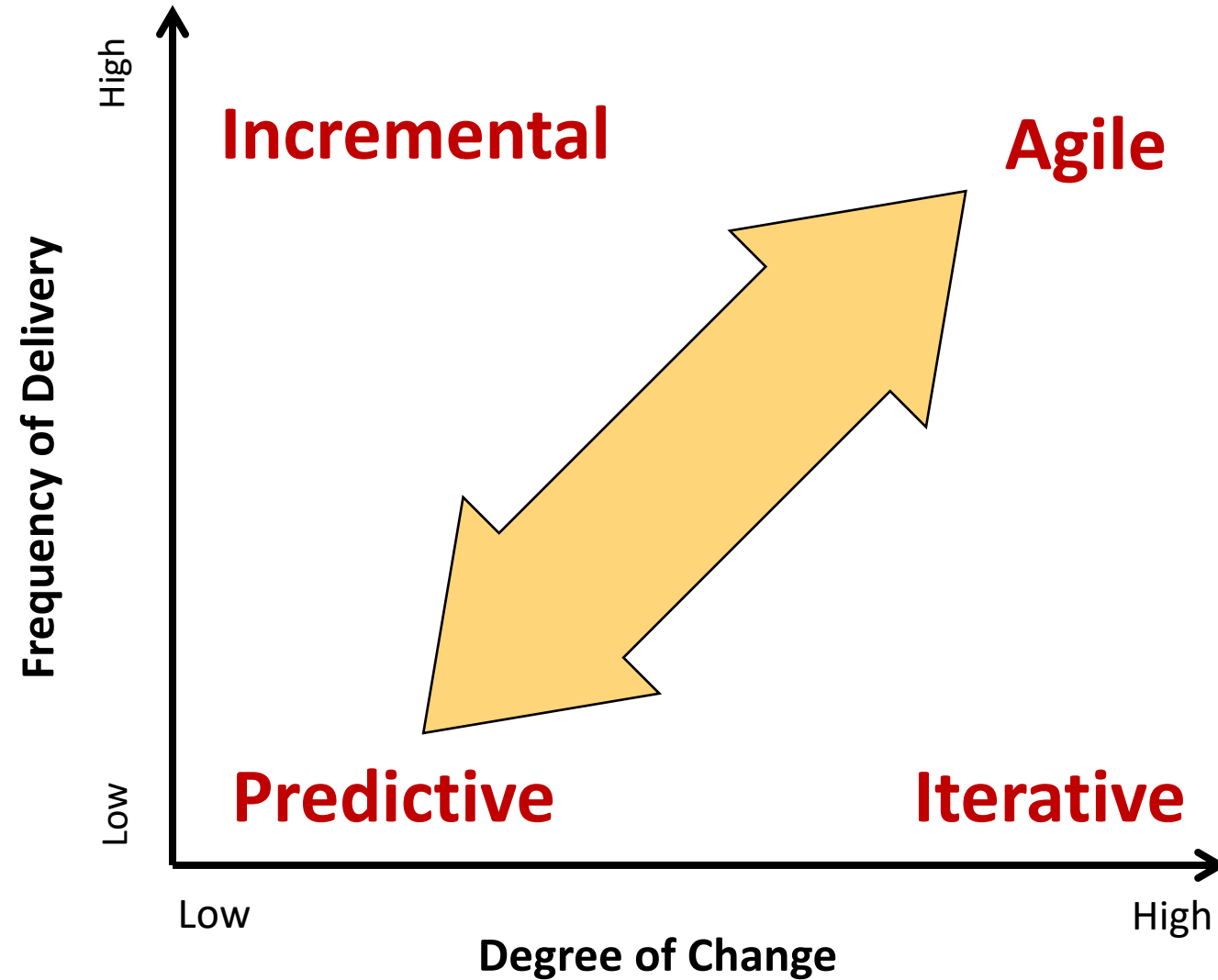
Source: Project Management Institute (2017), Agile Practice Guide, Project Management Institute
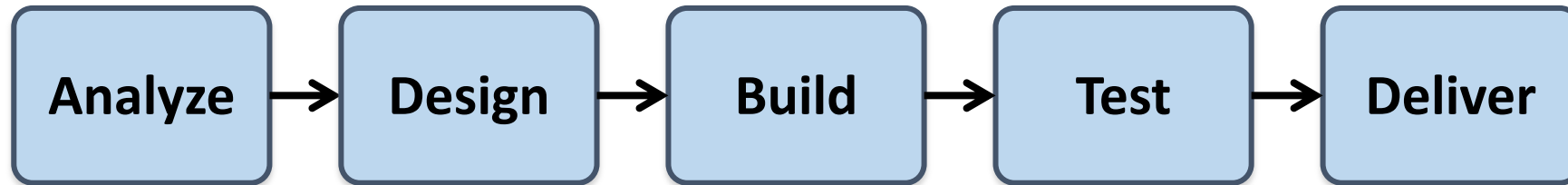
Iterative Life Cycle
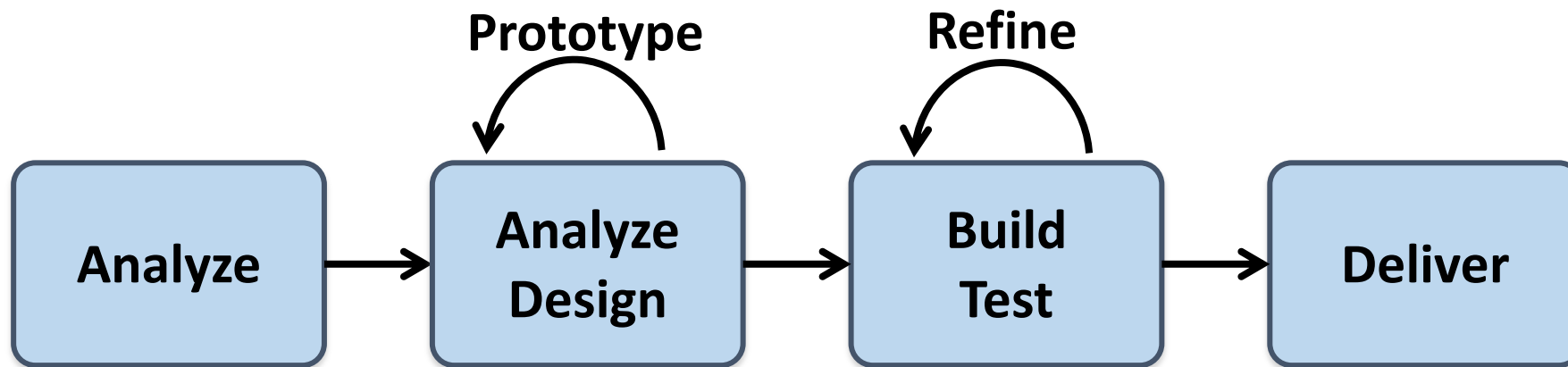
A Life Cycle of Varying-Sized Increments

Analyze
Design
Build
Test
Deliver

Analyze
Design
Build
Test
Deliver

Analyze
Design
Build
Test
Deliver

Source: Project Management Institute (2017), Agile Practice Guide, Project Management Institute
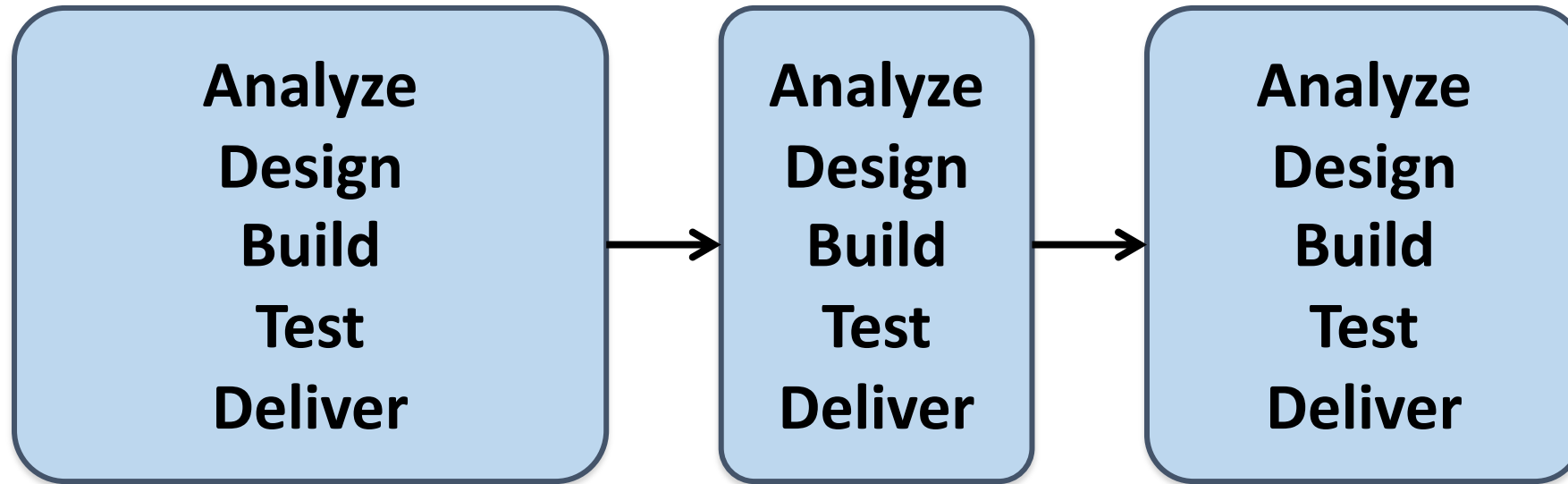
Iteration-Based and Flow-Based Agile Life Cycles

Iteration-Based Agile

Requirements Analysis Design Build Test	Requirements Analysis Design Build Test	Requirements Analysis Design Build Test	Requirements Analysis Design Build Test	Repeat as needed ...	Requirements Analysis Design Build Test	Requirements Analysis Design Build Test

Flow-Based Agile

Requirements Analysis Design Build Test the number of features in the WIP limit	Requirements Analysis Design Build Test the number of features in the WIP limit	Requirements Analysis Design Build Test the number of features in the WIP limit	Repeat as needed ...	Requirements Analysis Design Build Test the number of features in the WIP limit	Requirements Analysis Design Build Test the number of features in the WIP limit

From personas to features

1 **Personas** — A way of representing users

inspire

2 **Scenarios** — Natural language descriptions of a user interacting with a software product

are-developed-into

3 **Stories** — Natural language descriptions of something that is needed or wanted by users

define

inspire

4 **Features** — Fragments of product functionality

21

Multi-tier client-server architecture

Service-oriented Architecture

VM

Container

Virtual web server

Virtual mail server

User 1 Container 1

User 2 Container 2

Server software	Server software
Guest OS	Guest OS

Application software	Application software
Server software	Server software

Hypervisor

Container manager

Host OS

Host OS

Server Hardware

Server Hardware

Everything as a service

Photo editing

Software as a service (SaaS)

Logistics management

Cloud management Monitoring

Platform as a service (PaaS)

Database Software development

Storage Network

Infrastructure as a service (IaaS)

Computing Virtualization

Cloud data center

Software as a service

Software customers

Software provider

Cloud provider

Software services

Cloud Infrastructure

Microservices architecture – key design questions

What are the microservices that make up the system?

How should data be distributed and shared?

How should microservices communicate with each other?

Microservices architecture design

How should the microservices in the system be coordinated?

How should service failure be detected, reported and managed?

Types of security threat

An attacker attempts to deny access to the system for legitimate users

Availability threats

Distributed denial of service (DDoS) attack

An attacker attempts to damage the system or its data

Integrity threats

SOFTWARE PRODUCT

PROGRAM

DATA

Virus

Ransomware

Data theft

Confidentiality threats

An attacker tries to gain access to private information held by the system

Software product quality attributes

A refactoring process

Start → **① Identify code 'smell'** → **② Identify refactoring strategy** → **③ Make small improvement until strategy completed** → **④ Run automated code tests** → (back to ①)

Functional testing

Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Test-driven development (TDD)

Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

DevOps

Multi-skilled DevOps team

Code management and DevOps

DevOps automation

| Continuous integration | Continuous deployment | Continuous delivery | Infrastructure as code |

Code management system

Branching and merging

Recover version information

Code repository

Save and retrieve versions

Transfer code to/from developer's filestore

DevOps measurement

| Data collection | Data analysis | Report generation |

Cloud-Based Software:
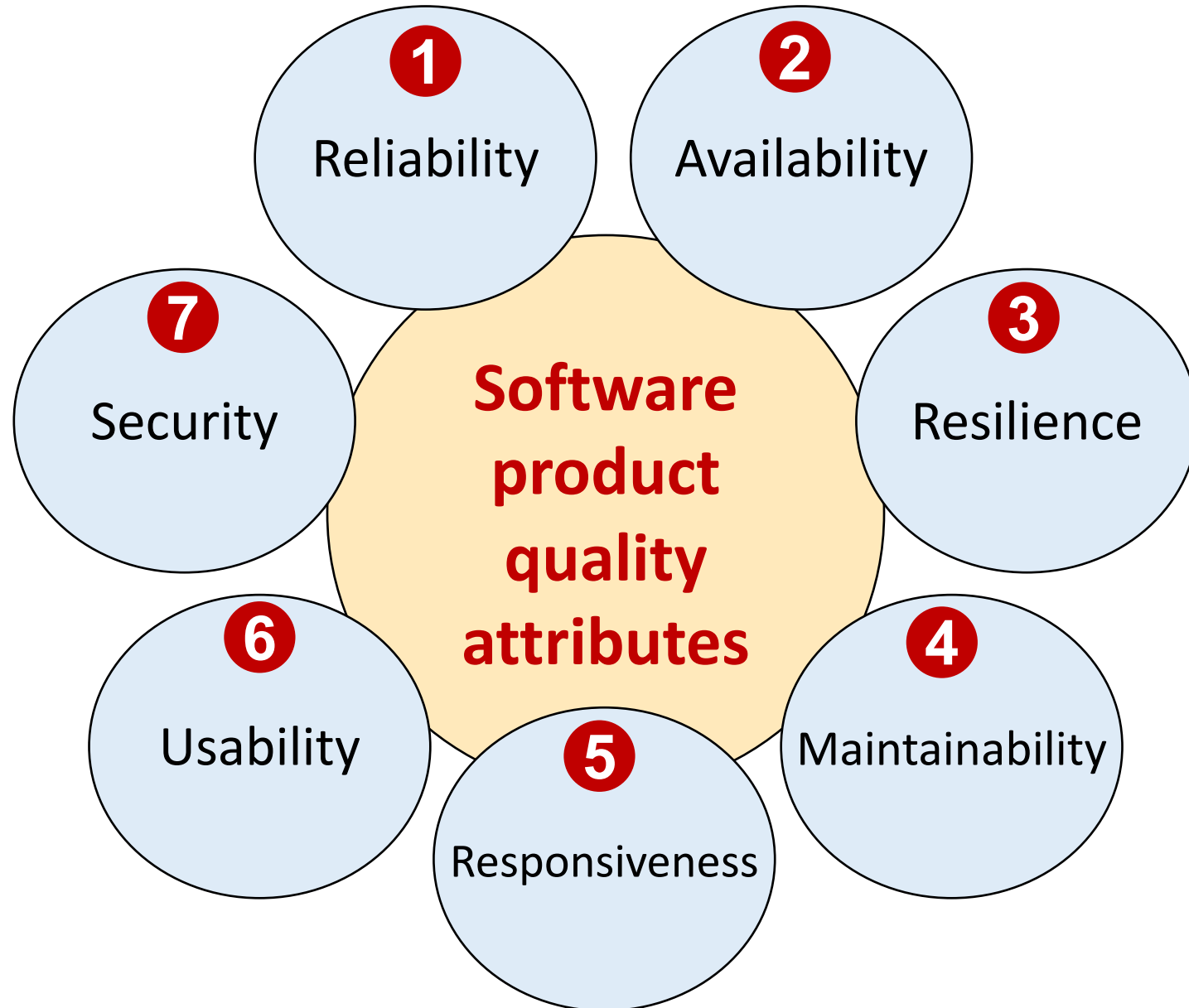**Virtualization and containers, Everything as a service, Software as a service; Cloud Computing and Cloud Software Architecture**

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-based servers are 'virtual servers',
    which means that they are implemented in
    software rather than hardware.**

The cloud

- **You can rent as many servers as you need, run your software on these servers and make them available to your customers.**

  - **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.**

Cloud Software:
Scaleability, elasticity and resilience

Scaleability

Maintain performance
as load increases

Elasticity

Adapt the server configuration
to changing demands

**Cloud software
characteristics**
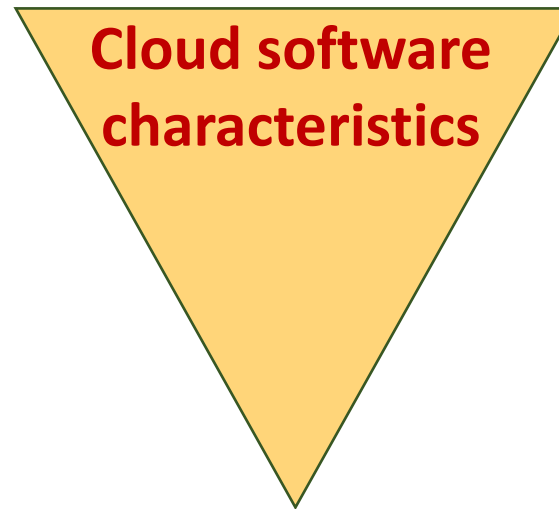
Resilience

Maintain service in the
event of server failure

Scaleability

- **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

- **Elasticity** is related to scaleability but also allows for scaling-down as well as scaling-up.

  - You can monitor the demand on your application and add or remove servers dynamically as the number of users change.

Resilience

- **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.

Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

41

Benefits of using the cloud for software development

- **Cost**
  You avoid the initial capital costs of hardware procurement

- **Startup time**
  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.

Virtual cloud servers
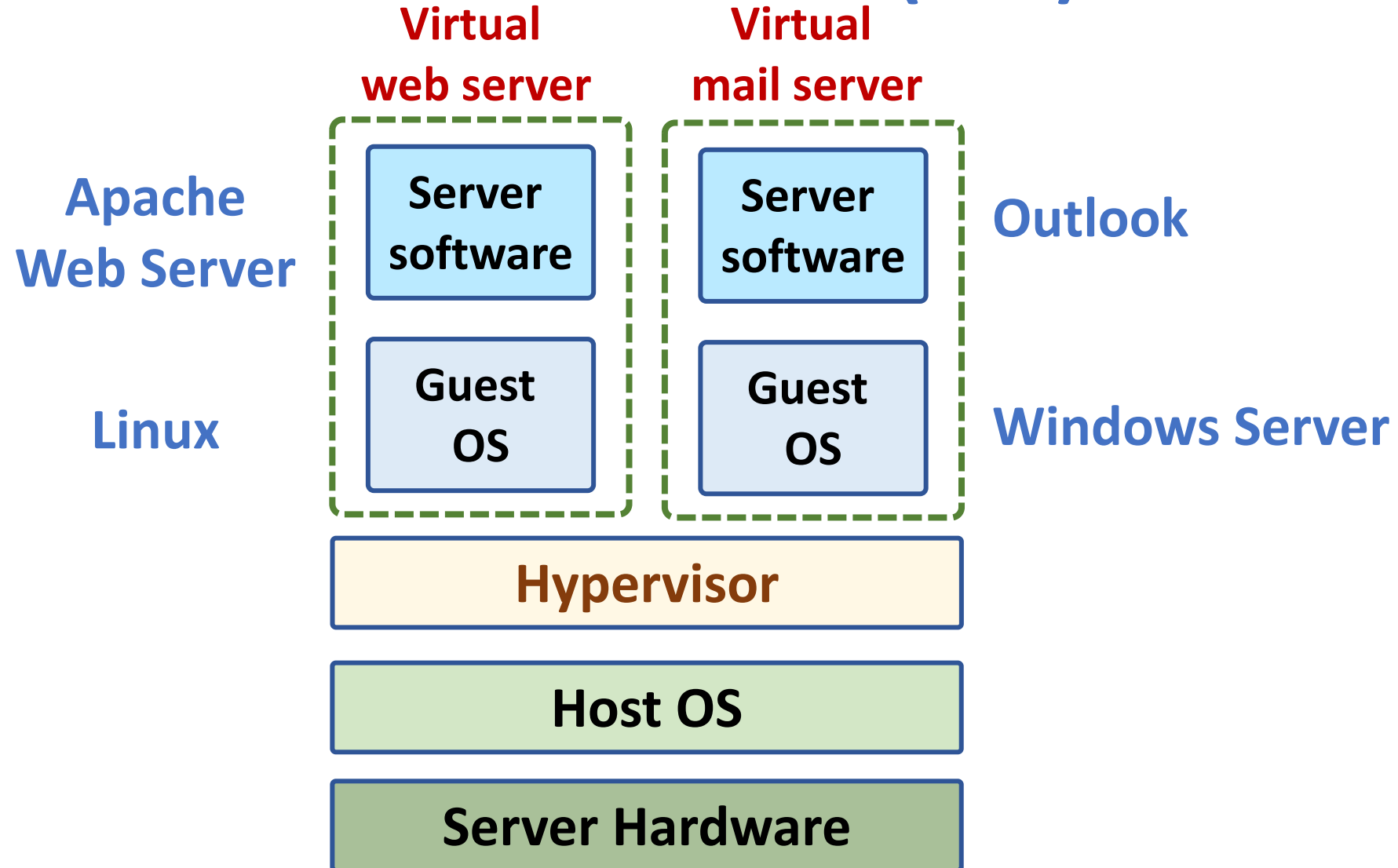
- A **virtual server** runs on an underlying **physical computer** and is made up of an operating system plus a set of software packages that provide the server functionality required.

- 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 cloud servers

- **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.

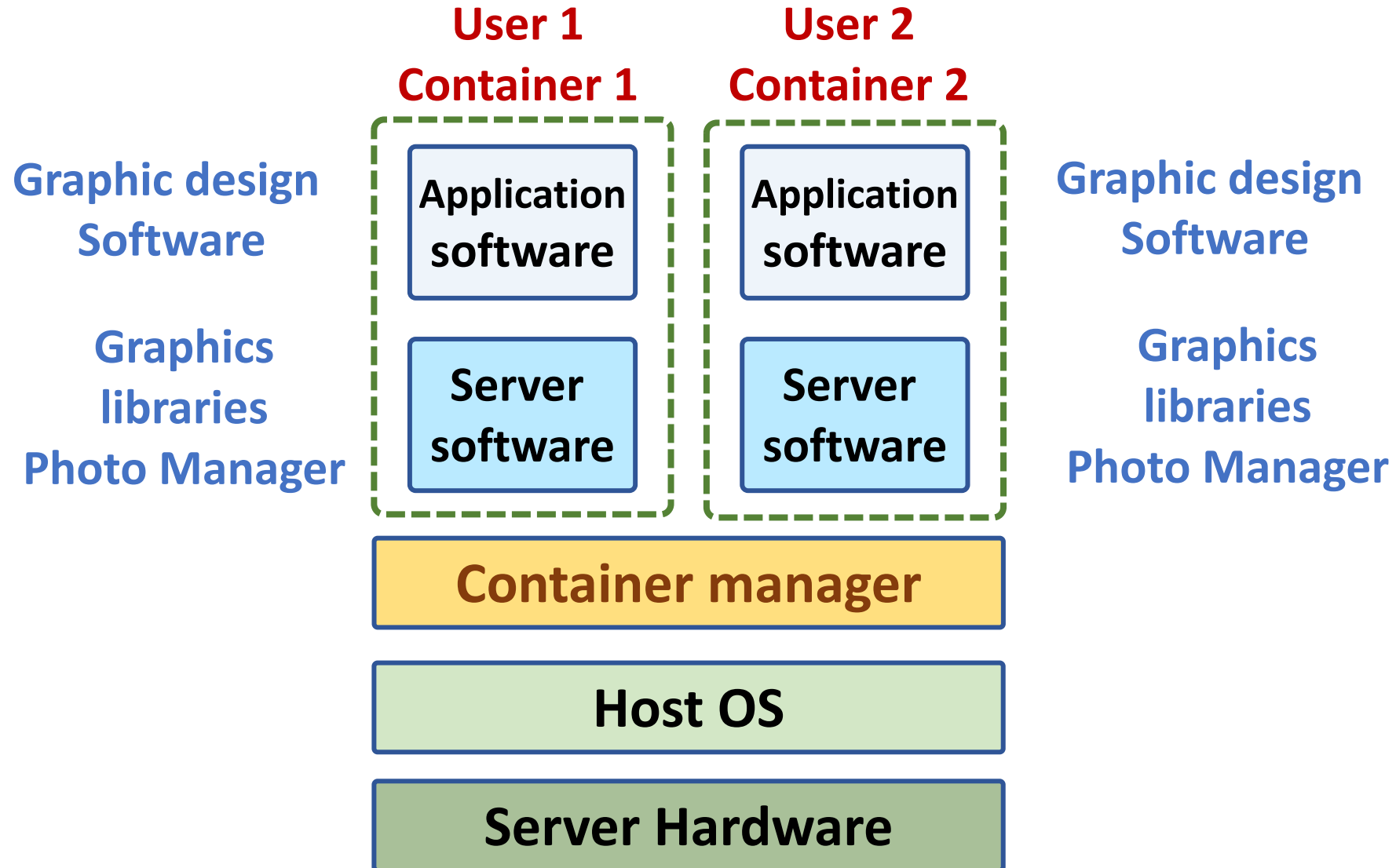# Implementing a virtual server as a Virtual Machine (VM)

Virtual web server

Virtual mail server

Apache Web Server

Server software

Server software

Outlook

Linux

Guest OS

Guest OS

Windows Server

Hypervisor

Host OS

Server Hardware

45

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.**
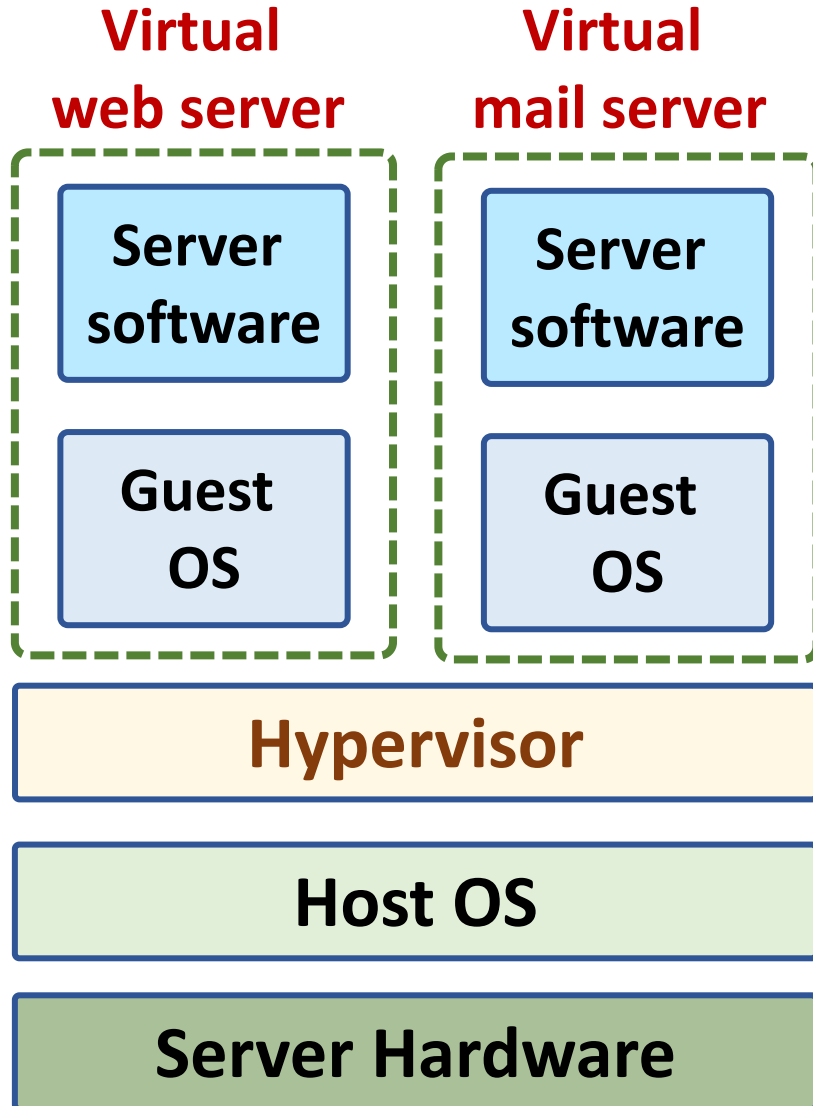
Container-based virtualization

- **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

VM

Virtual web server

Server software

Guest OS

Virtual mail server

Server software

Guest OS

Hypervisor

Host OS

Server Hardware

Container

User 1 Container 1

Application software

Server software

User 2 Container 2

Application software

Server software

Container manager

Host OS

Server Hardware

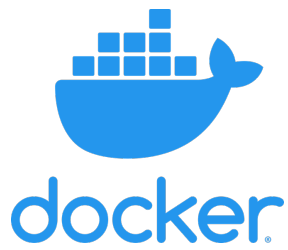Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

49

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

Registries

Docker hub

Images

Docker client

Docker Daemon

Containers

Dockerfiles

Docker host

Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.
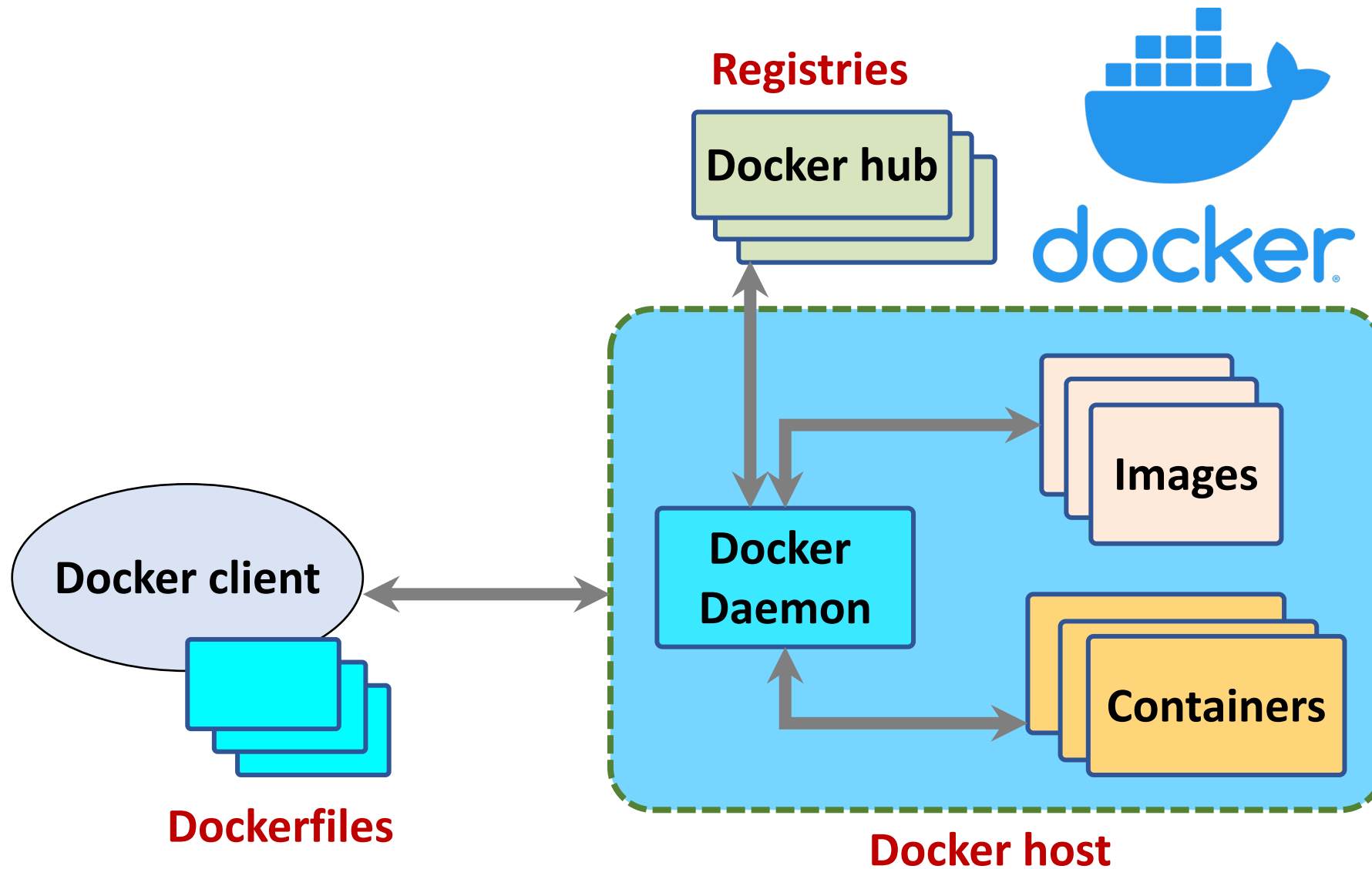
The elements of the Docker container system

- **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

The elements of the Docker container system

- **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 bby 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.

Docker images

- **Docker images** are directories that can be archived, shared and run on different Docker hosts. Everything that's needed to run a software system - binaries, libraries, system tools, etc. is included in the directory.

- 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.

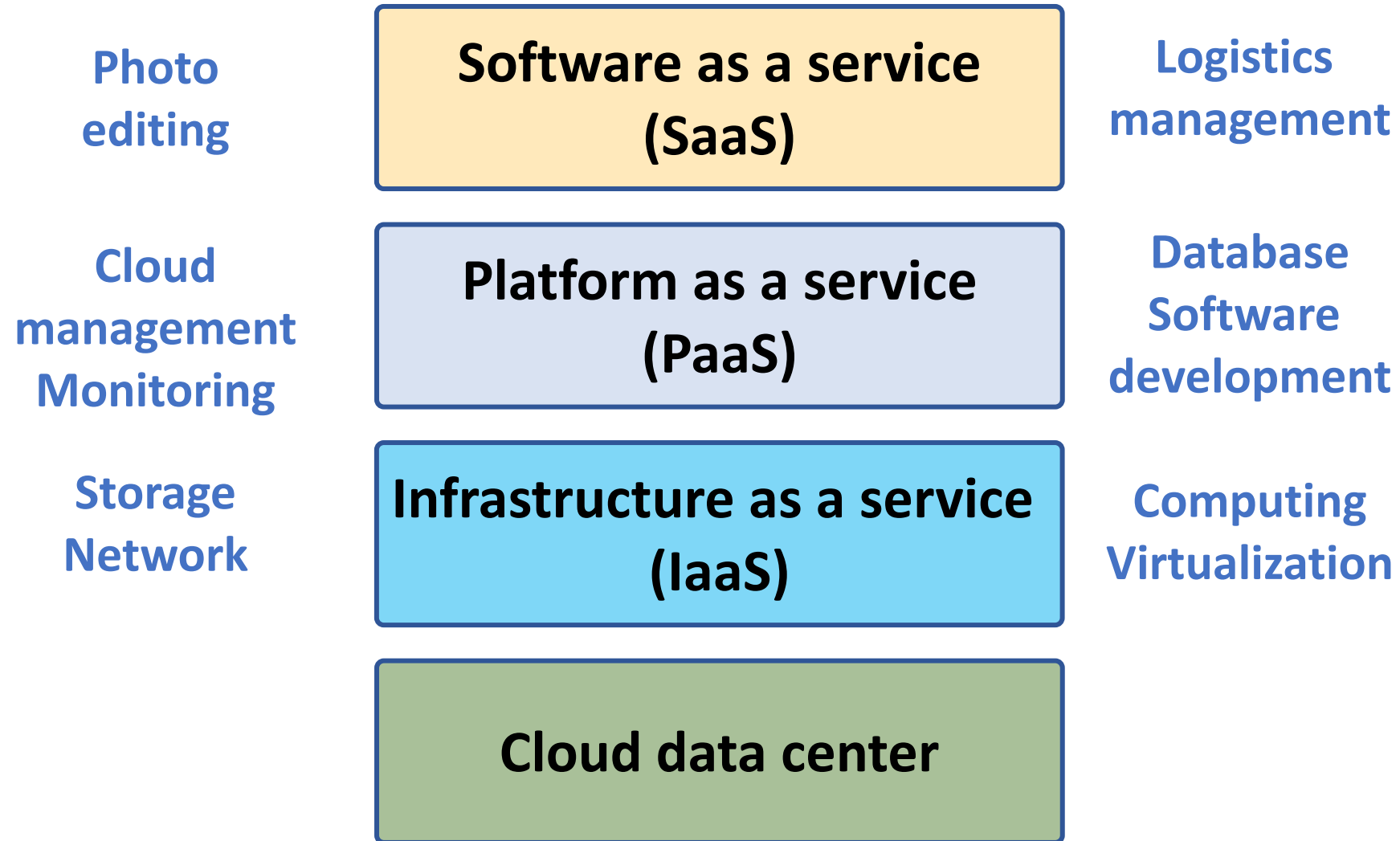# Benefits of containers

- **They solve the problem of software dependencies.**

  - **You don't have to worry about the libraries and other software on the application server being different from those on your development server.**

  - **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.**

Benefits of containers

- **They provide a mechanism for software portability across different clouds.**

  - **Docker containers can run on any system or cloud provider where the Docker daemon is available**

- **They provide an efficient mechanism for implementing software services and so support the development of service-oriented architectures.**

- **They simplify the adoption of DevOps.**

  - **This is an approach to software support where the same team are responsible for both developing and supporting operational software.**

Everything as a service

Photo editing

Cloud management Monitoring

Storage Network

| Software as a service (SaaS) |
| Platform as a service (PaaS) |
| Infrastructure as a service (IaaS) |
| Cloud data center |

Logistics management

Database Software development

Computing Virtualization

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)**

- **Platform as a service (PaaS)**

- **Software as a service (SaaS)**

Infrastructure as a service (IaaS)

- **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.**
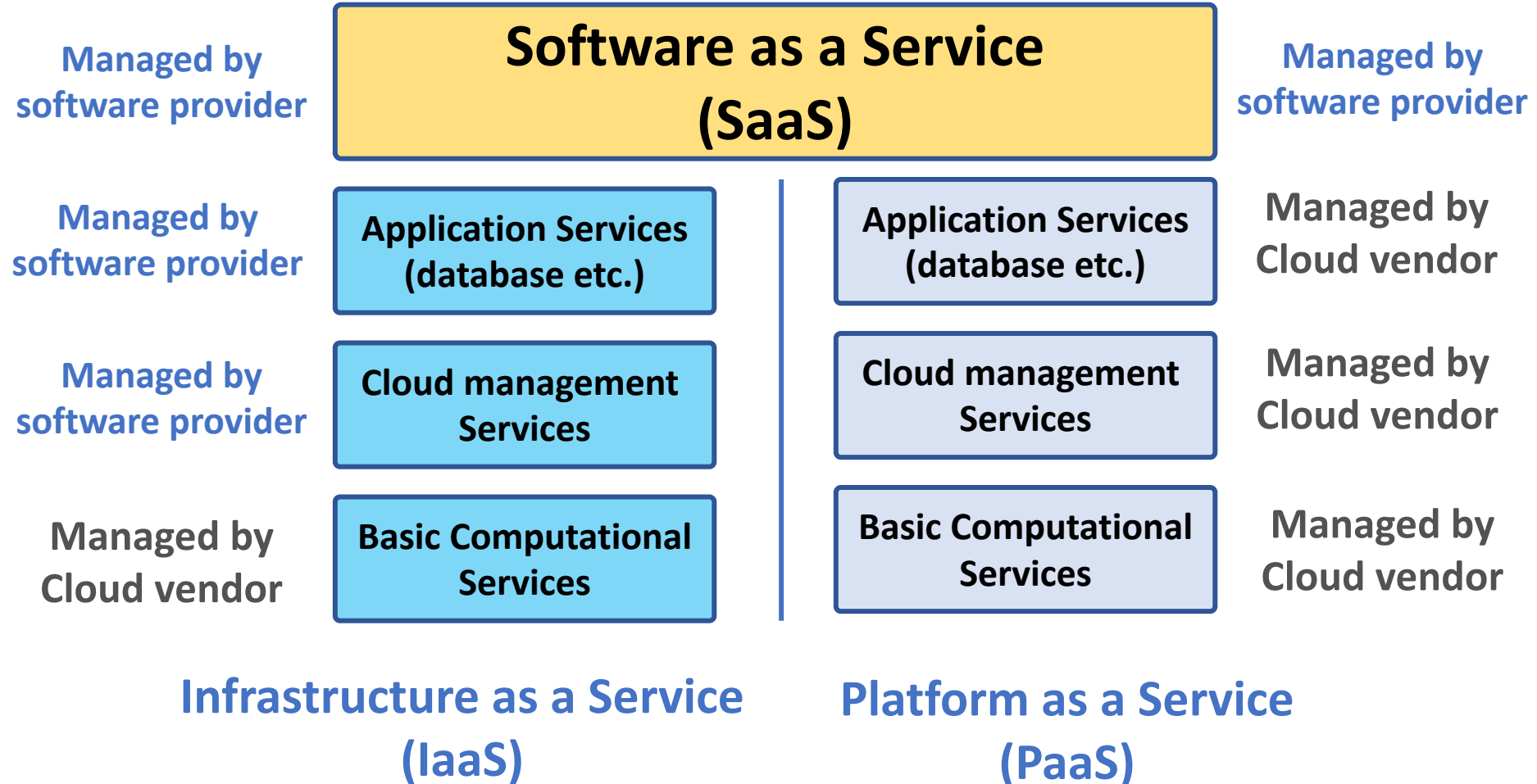
Platform as a service (PaaS)

- **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.**

Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

61

Software as a service (SaaS)

- **Software as a service (SaaS)**
  - **Your software product runs on the cloud and is accessed by users through a web browser or mobile app.**

Management responsibilities for IaaS and PaaS

Managed by software provider

Software as a Service (SaaS)

Managed by software provider

Managed by software provider

Application Services (database etc.)	Application Services (database etc.)
Cloud management Services	Cloud management Services
Basic Computational Services	Basic Computational Services

Managed by software provider

Managed by Cloud vendor

Managed by Cloud vendor

Managed by Cloud vendor

Managed by Cloud vendor

Infrastructure as a Service (IaaS)

Platform as a Service (PaaS)

Software as a service

- **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

Software customers

Software provider

Cloud provider

Software services

Cloud Infrastructure
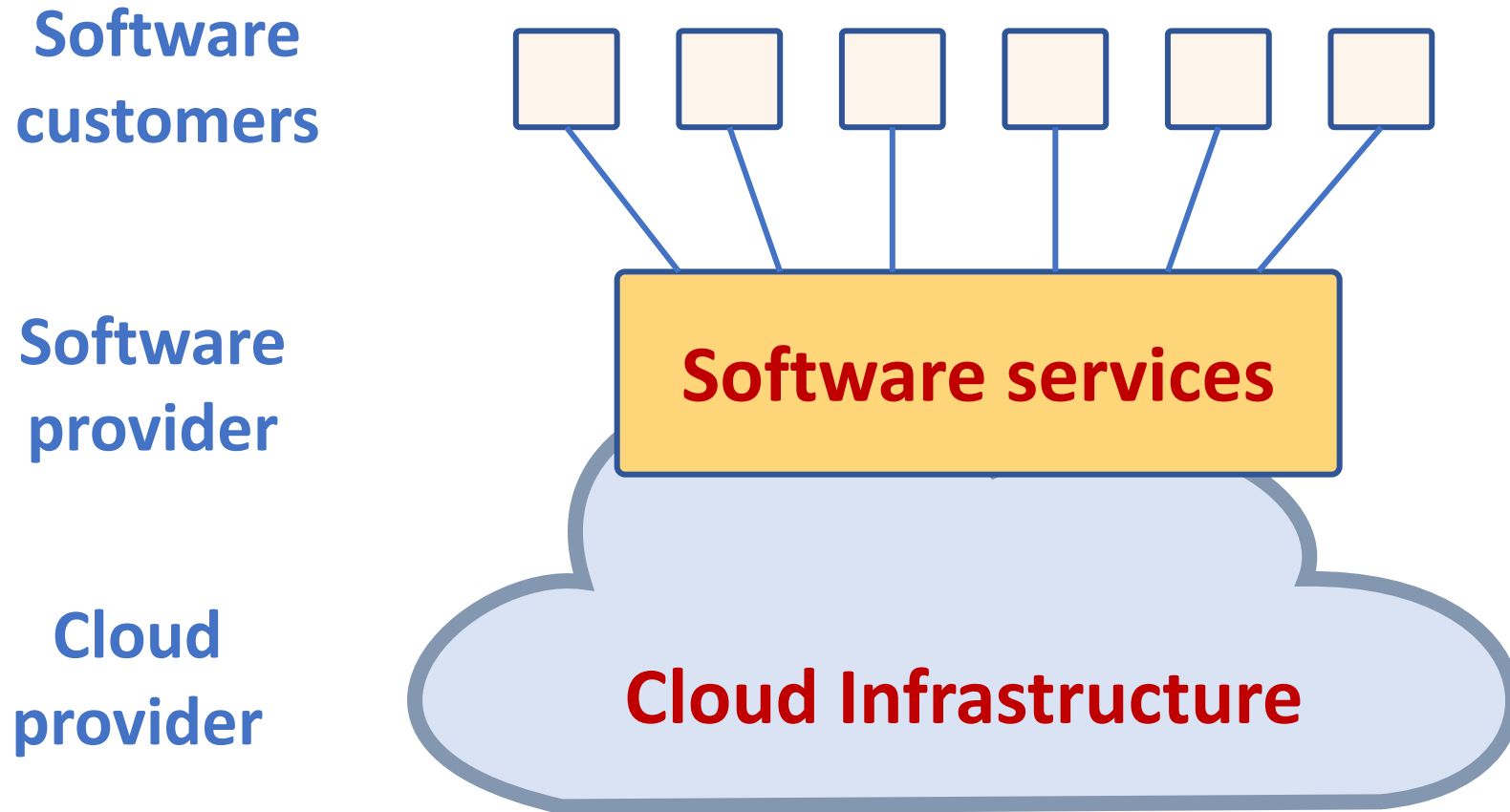
Benefits of SaaS for software product providers

- ## 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. 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.

Benefits of SaaS for software product providers

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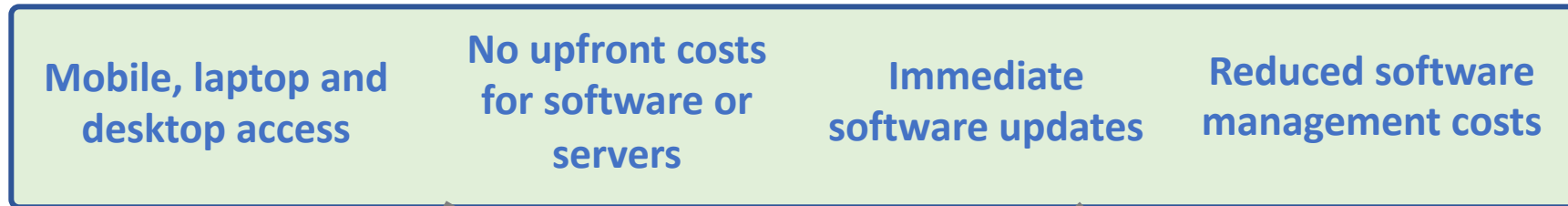
- **Try before you buy**
  You can make early free or low-cost versions of the software available quickly with the aim of getting customer feedback on bugs and how the product could be approved.

- **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.
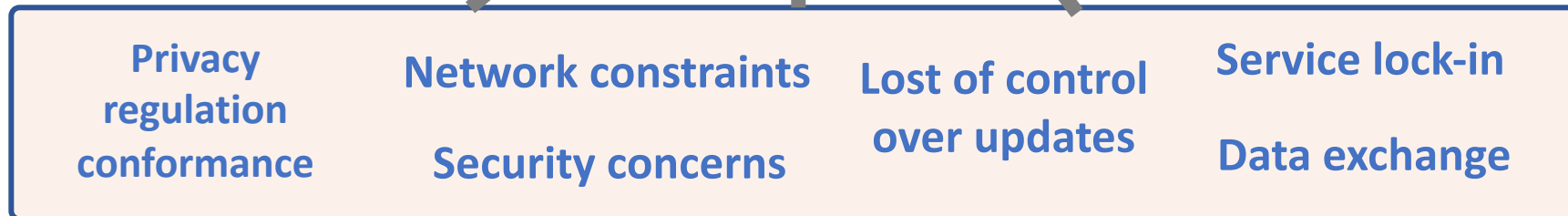
Advantages and disadvantages of SaaS for customers

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

Lost 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. 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 storage and management issues for SaaS

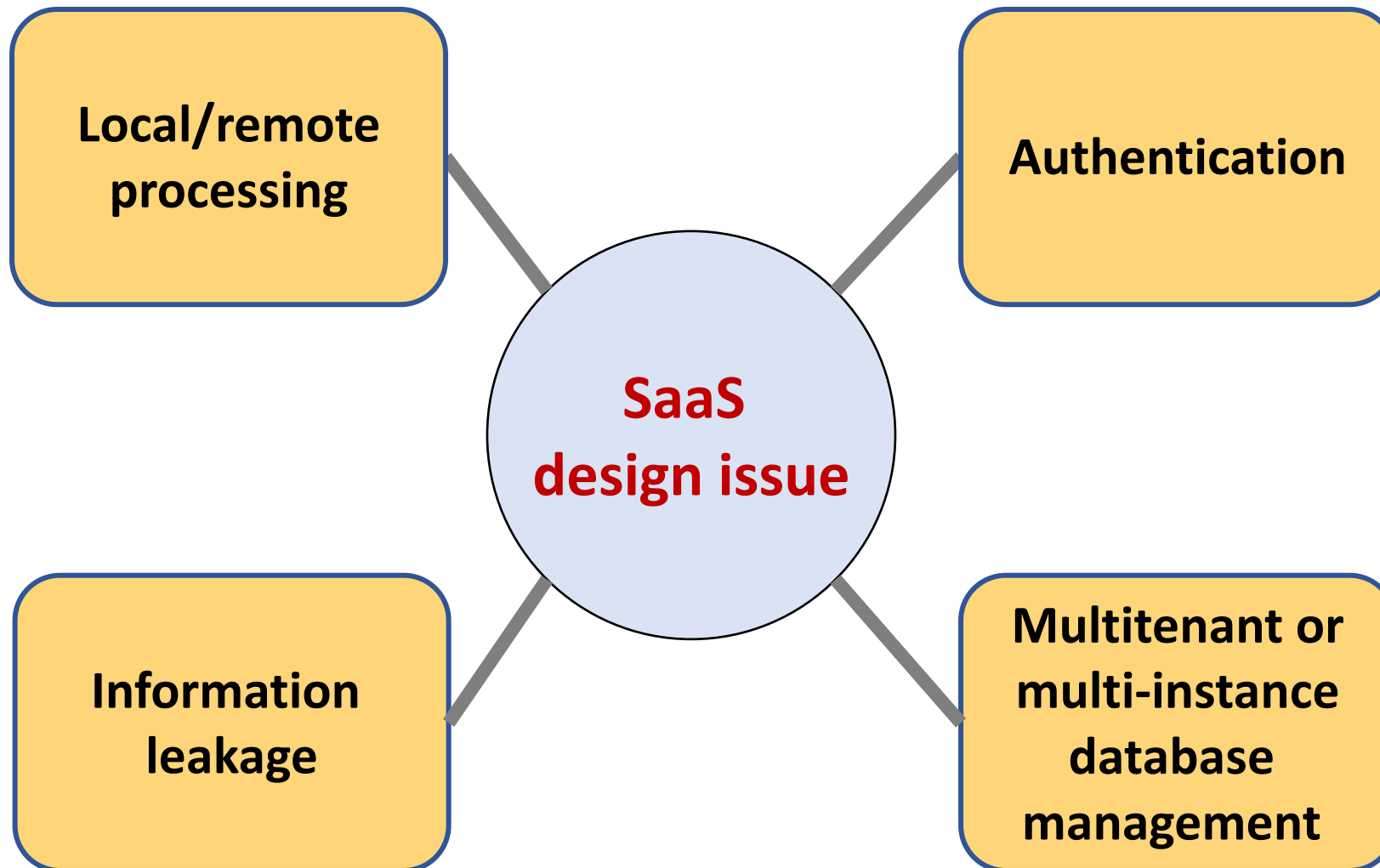- ## Data security

  Companies dealing with sensitive information may be unwilling to hand over the 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.

Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

70

Design issues for software delivered as a service

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.

Possible customisations for SaaS

- **Authentication**
  Businesses may want users to authenticate using their business credentials rather than the account credentials set up by the software provider.

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

Possible customisations for SaaS

- ## 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.

- ## Access control

  Businesses may want to be able to define their own access control model that sets out the data that specific users or user groups can access and the allowed operations on that data.

Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

74

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**
  Multitenant databases have to be designed for security because the data for all customers is held in the same database. They are, therefore, likely to have fewer security vulnerabilities than standard database products. 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. I explain possible database adaptations later in this section.
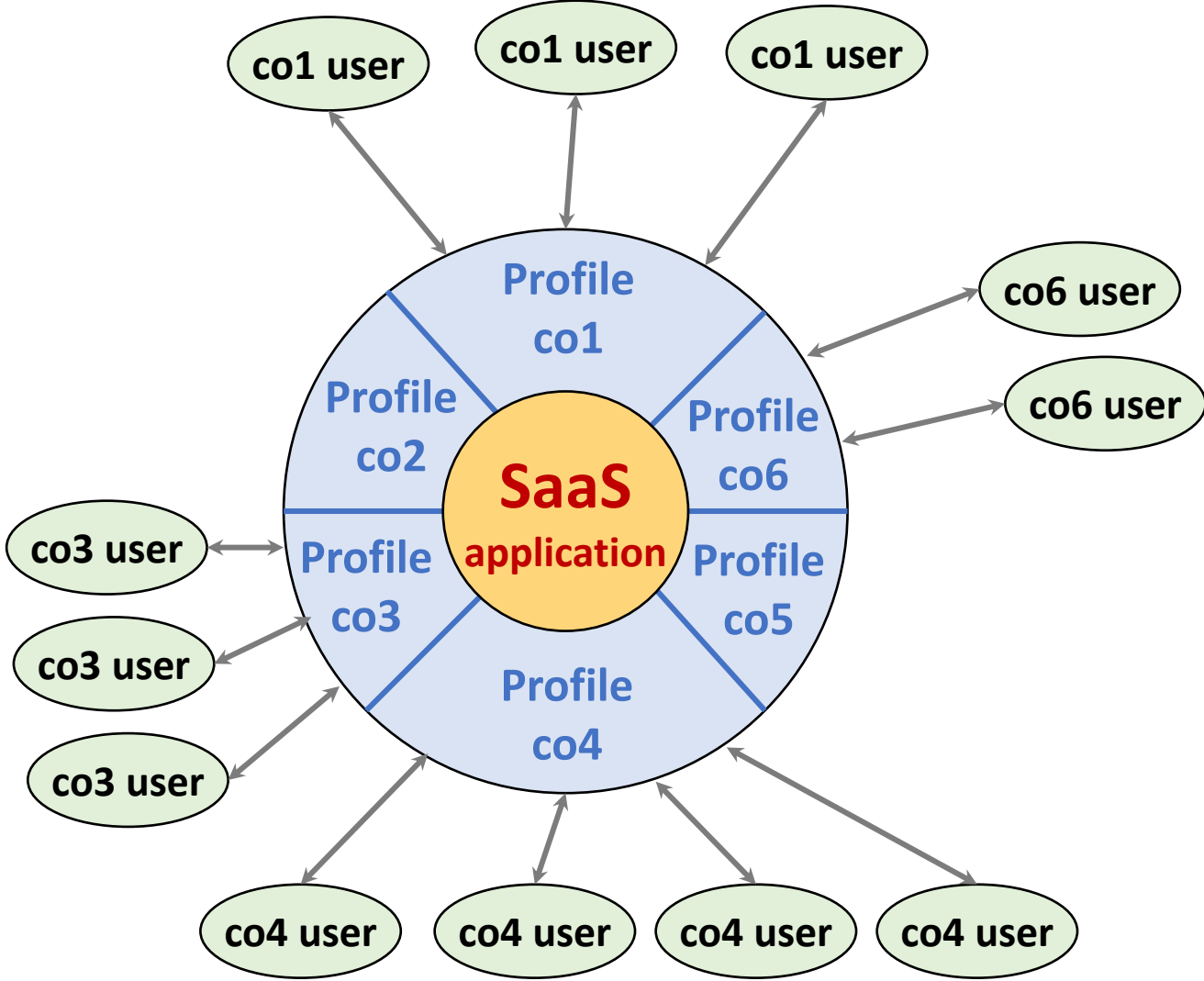
- ## Security
  As data for all customers is maintained in the same database, then there is a theoretical possibility that data will leak from one customer to another. In fact, there are very few instances of this happening. More seriously, perhaps, if there is a database security breach then it affects all customers.

- ## 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.

User profiles for SaaS access

Architectural decisions for cloud software engineering

Database organization

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

Scalability and resilience

What are the software scalability and resilience requirements?

Software structure

Should the software structure be monolithic or service-oriented?

What cloud platform should be used for development and delivery?

Cloud platform
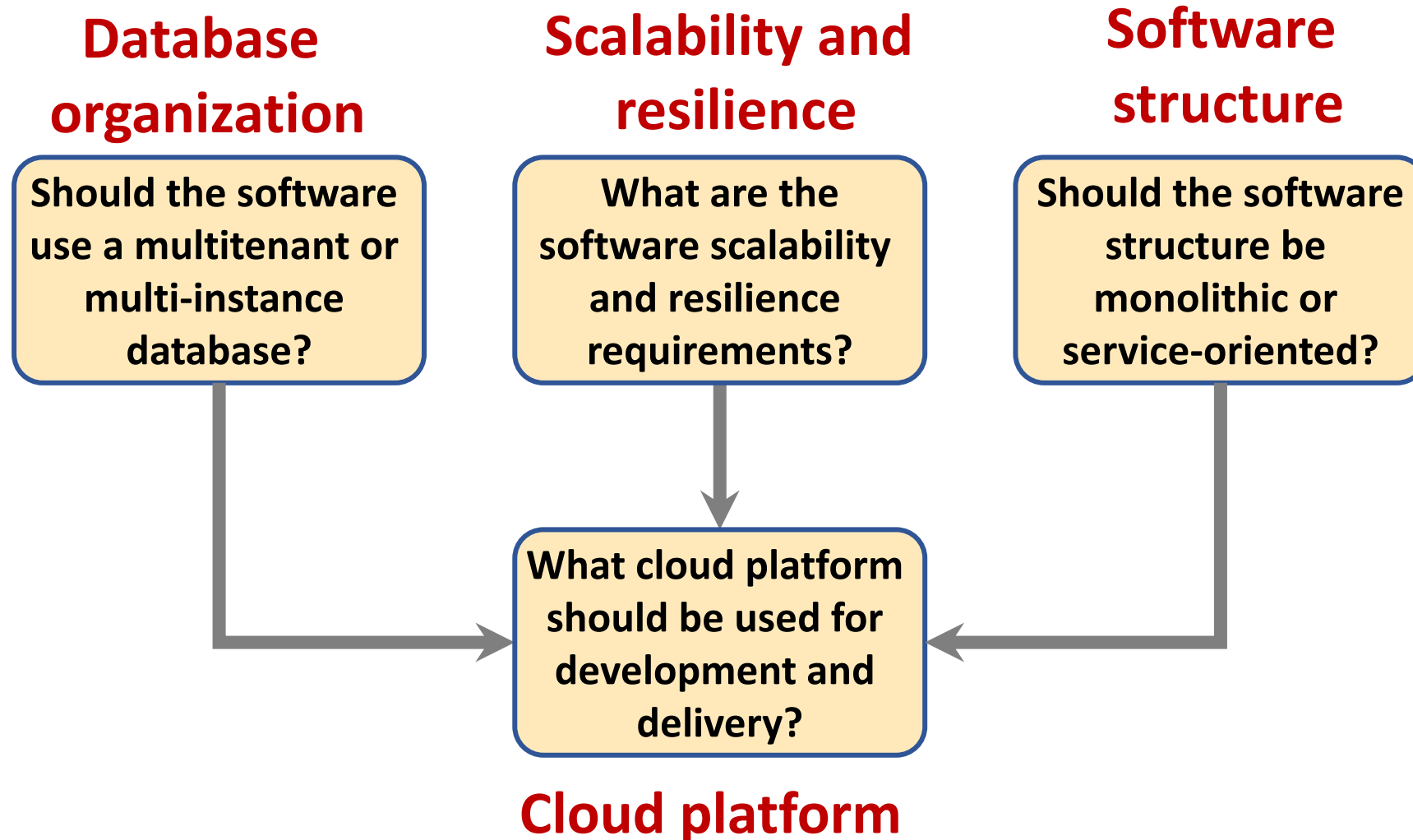
Questions to ask when choosing a database organization

1. **Target customers**

2. **Transaction requirements**

3. **Database size and connectivity**

4. **Database interoperability**

5. **System structure**

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.

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

Questions to ask when choosing a database organization

- ## 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 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

- **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

- 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.

Scalability and resilience

- **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

System monitor

Location A

Location B

Active system

Standby system

Database mirror

Database 1

Database 2

Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

85
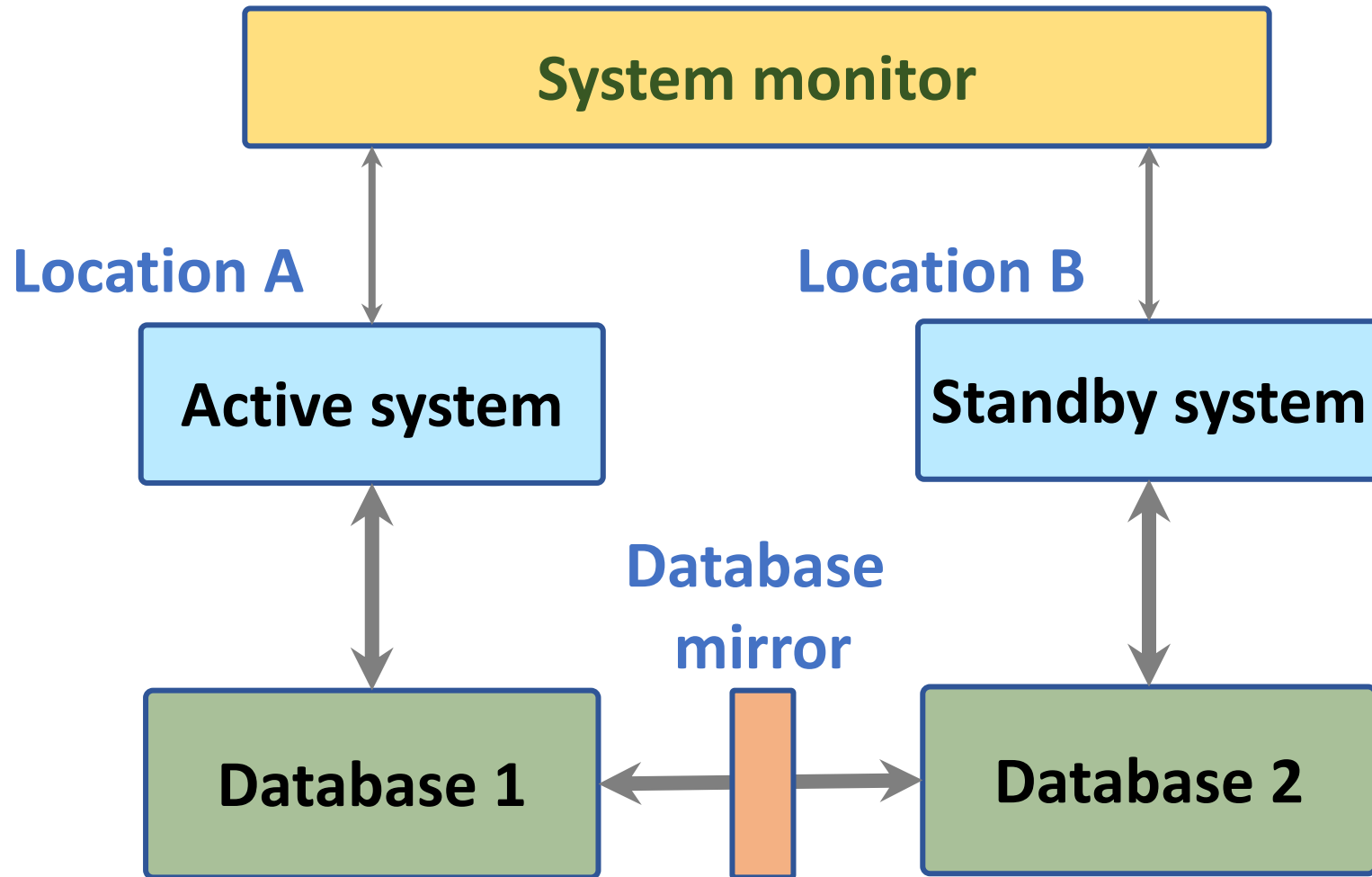
Resilience

- **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.**

Resilience

- **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 operation can be switched automatically to the software copies elsewhere.**

System structure

- An **object-oriented** approach to software engineering has been that been extensively used for the development of client-server systems built around a shared database.

- The system itself is, logically, a **monolithic system** with distribution across multiple servers running large software components. The traditional **multi-tier client server architecture** is based on this **distributed system model**.

System structure

- **The alternative to a monolithic approach to software organization is a service-oriented approach where 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.**

Cloud platform

- 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 are also smaller national providers that provide more limited services but who may be more willing to adapt their services to the needs of different customers.

- 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

Expected load and load predictability

Resilience

Cloud platform choice

Supported cloud services

Privacy and data protection
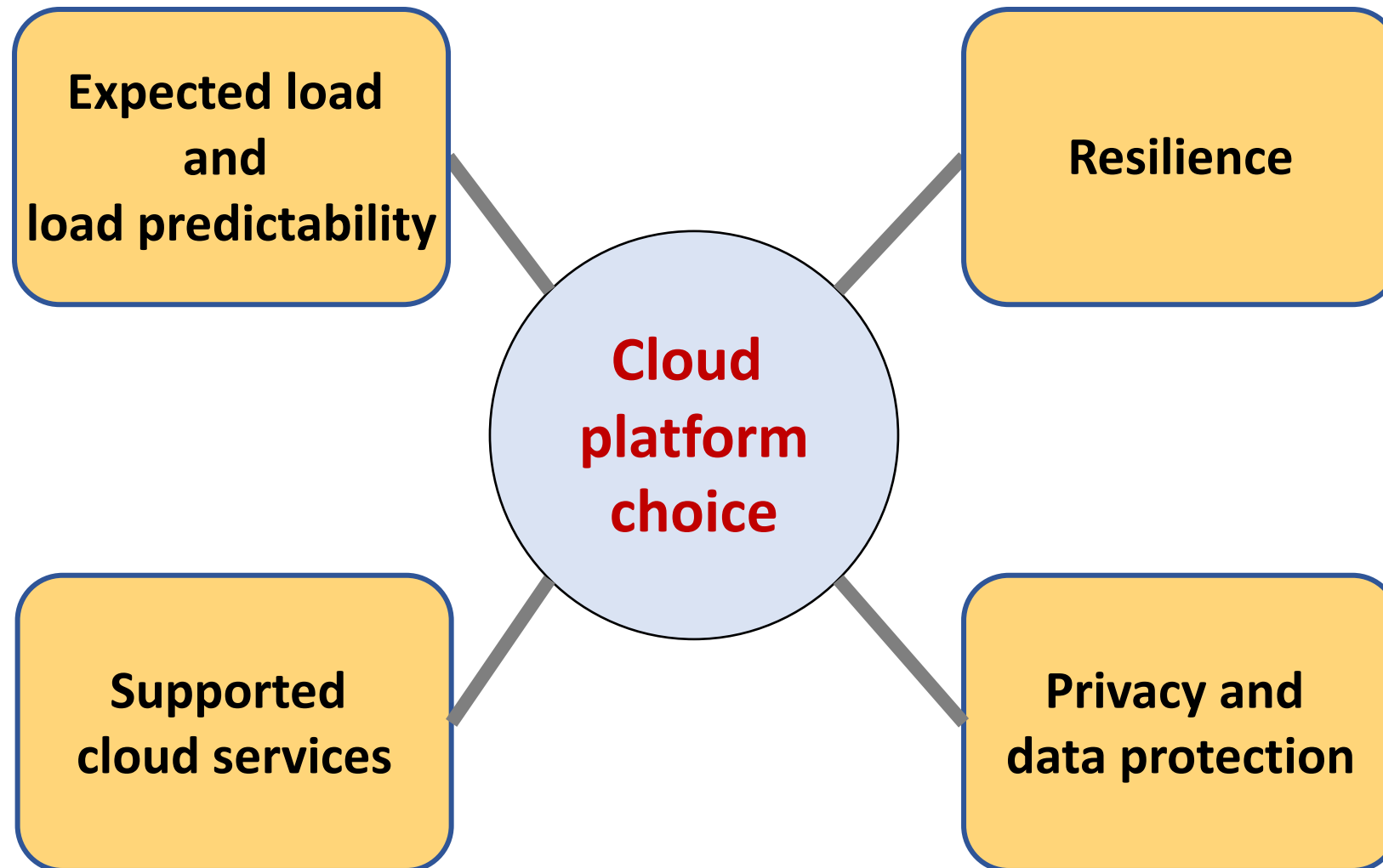
Business issues in cloud platform choice

Cost

Developer experience

Target customers

Business issues

Service-level agreements

Portability and cloud migration

Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.
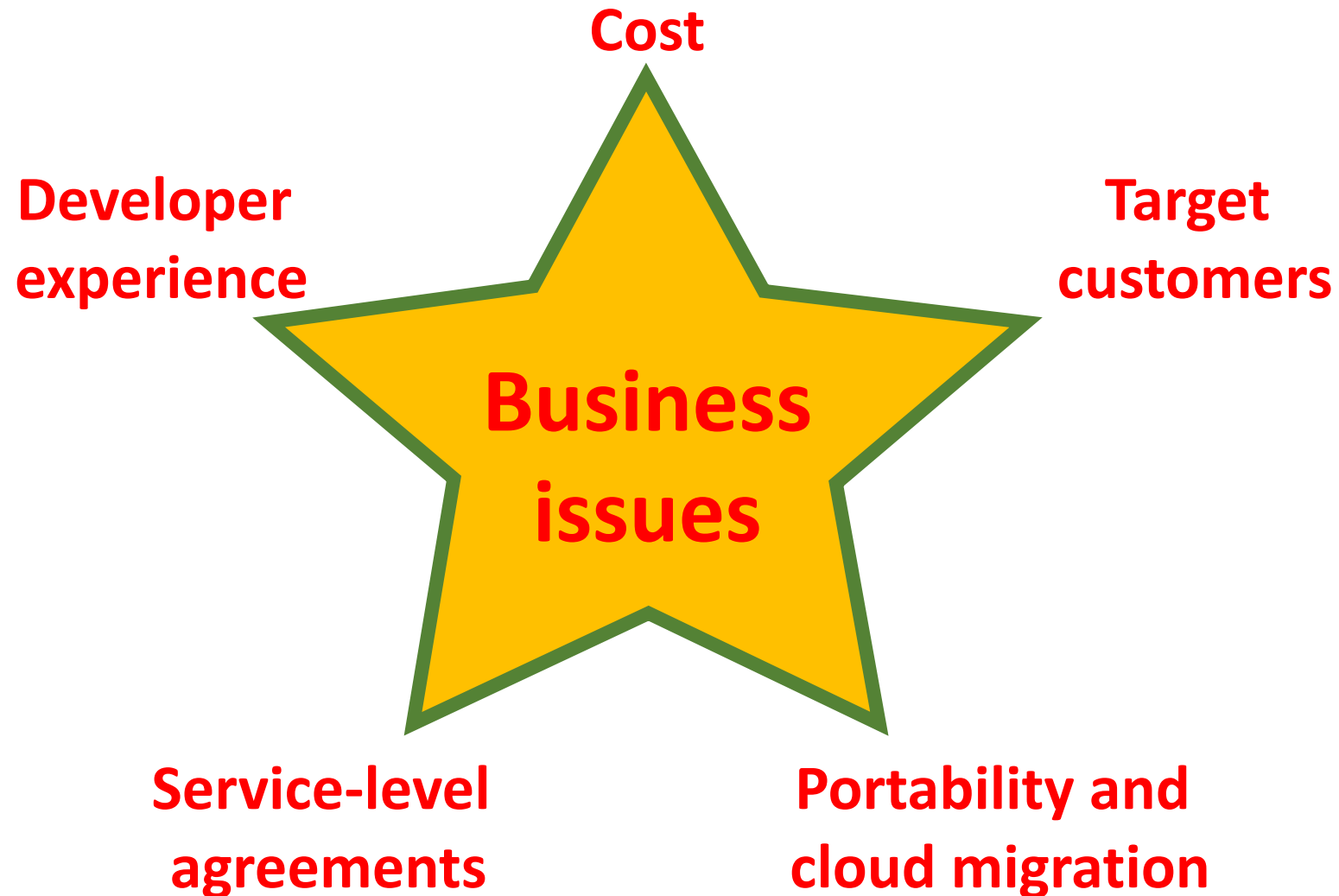
Cloud Computing and Cloud Software Architecture

Web Application with
AWS Core Services

AWS Products and Services

Analytics	Application Integration	AR & VR	**AWS Cost Management**	Blockchain
Business Applications	**Compute**	Customer Engagement	**Database**	Developer Tools
End User Computing	Game Tech	Internet of Things	Machine Learning	Management & Governance
Media Services	Migration & Transfer	Mobile	**Networking & Content Delivery**	Quantum Technologies
Robotics	Satellite		**Security, Identity & Compliance**	**Storage**

Source: https://aws.amazon.com/

95

AWS Compute

Compute

Amazon EC2
Virtual servers in the cloud

Amazon EC2 Auto Scaling
Scale compute capacity to meet demand

Amazon Elastic Container Registry
Store and retrieve docker images

Amazon Elastic Container Service
Run and manage docker containers

Amazon Elastic Kubernetes Service
Run managed Kubernetes on AWS

Amazon Lightsail
Launch and manage virtual private servers

AWS Batch
Run batch jobs at any scale

AWS Elastic Beanstalk
Run and manage web apps

AWS Fargate
Run containers without managing servers or clusters

AWS Lambda
Run code without thinking about servers

AWS Outposts
Run AWS infrastructure on-premises

AWS Serverless Application Repository
Discover, deploy, and publish serverless applications

AWS Wavelength
Deliver ultra-low latency applications for 5G devices

VMware Cloud on AWS
Build a hybrid cloud without custom hardware

Source: https://aws.amazon.com/

96

AWS Database

Database

Amazon Aurora
High Performance Managed Relational Database

Amazon DynamoDB
Managed NoSQL Database

Amazon DocumentDB (with MongoDB compatibility)
Fully managed document database

Amazon ElastiCache
In-memory Caching System

Amazon Managed Apache Cassandra Service
Managed Cassandra-compatible database

Amazon Neptune
Fully Managed Graph Database Service

Amazon Quantum Ledger Database (QLDB)
Fully managed ledger database

Amazon RDS
Managed Relational Database Service for MySQL, PostgreSQL, Oracle, SQL Server, and MariaDB

Amazon RDS on VMware
Automate on-premises database management

Amazon Redshift
Fast, Simple, Cost-effective Data Warehousing

Amazon Timestream
Fully managed time series database

AWS Database Migration Service
Migrate Databases with Minimal Downtime

AWS Storage

Storage

Amazon Simple Storage Service (S3)
Scalable Storage in the Cloud

Amazon Elastic Block Store (EBS)
EC2 block storage volumes

Amazon Elastic File System (EFS)
Fully managed file system for EC2

Amazon FSx for Lustre
High-performance file system integrated with S3

Amazon FSx for Windows File Server
Fully managed Windows native file system

Amazon S3 Glacier
Low-cost Archive Storage in the Cloud

AWS Backup
Centralized backup across AWS services

AWS Snow Family
Physical devices to migrate data into and out of AWS

AWS Storage Gateway
Hybrid Storage Integration

CloudEndure Disaster Recovery
Highly automated disaster recovery

AWS Networking & Content Dilivery

Networking & Content Delivery

Amazon VPC
Isolated Cloud Resources

Amazon API Gateway
Build, Deploy, and Manage APIs

Amazon CloudFront
Global Content Delivery Network

Amazon Route 53
Scalable Domain Name System

AWS PrivateLink
Securely Access Services Hosted on AWS

AWS App Mesh
Monitor and control microservices

AWS Cloud Map
Application resource registry for microservices

AWS Direct Connect
Dedicated Network Connection to AWS

AWS Global Accelerator
Improve application availability and performance

AWS Transit Gateway
Easily scale VPC and account connections

Elastic Load Balancing
Distribute incoming traffic across multiple targets

AWS Security, Identity & Compliance

Security, Identity & Compliance

AWS Identity & Access Management
Manage User Access and Encryption Keys

Amazon Cognito
Identity Management for your Apps

Amazon Detective
Investigate potential security issues

Amazon GuardDuty
Managed Threat Detection Service

Amazon Inspector
Analyze Application Security

Amazon Macie
Discover, Classify, and Protect your Data

AWS Artifact
On-demand access to AWS compliance reports

AWS Certificate Manager
Provision, Manage, and Deploy SSL/TLS Certificates

AWS CloudHSM
Hardware-based Key Storage for Regulatory Compliance

AWS Directory Service
Host and Manage Active Directory

AWS Firewall Manager
Central Management of Firewall Rules

AWS Key Management Service
Managed Creation and Control of Encryption Keys

AWS Resource Access Manager
Simple, secure service to share AWS resources

AWS Secrets Manager
Rotate, Manage, and Retrieve Secrets

AWS Security Hub
Unified security and compliance center

AWS Shield
DDoS Protection

AWS Single Sign-On
Cloud Single Sign-On (SSO) Service

AWS WAF
Filter Malicious Web Traffic

Source: https://aws.amazon.com/

100

AWS Cost Management

AWS Cost Management

AWS Cost Explorer
Analyze Your AWS Cost and Usage

AWS Budgets
Set Custom Cost and Usage Budgets

AWS Cost and Usage Report
Access Comprehensive Cost and Usage Information

Reserved Instance Reporting
Dive Deeper into Your Reserved Instances (RIs)

Savings Plans
Save up to 72% on compute usage with flexible pricing

AWS Services

- **Amazon EC2**

  - **Virtual servers in the cloud**

- **Amazon Simple Storage Service (S3)**

  - **Scalable storage in the cloud**

- **Amazon Aurora**

  - **High performance managed relational database**

- **Amazon DynamoDB**

  - **Managed NoSQL database**

- **Amazon RDS**

  - **Managed relational database service for MySQL, PostgreSQL, Oracle, SQL Server, and MariaDB**

AWS Services

- **AWS Lambda**
  - **Run code without thinking about servers**
- **AWS Elastic Beanstalk**
  - **Run and manage web apps**
- **Amazon VPC**
  - **Isolated cloud resources**
- **Amazon Lightsail**
  - **Launch and manage virtual private servers**
- **Amazon SageMaker**
  - **Build, train, and deploy machine learning models at scale**
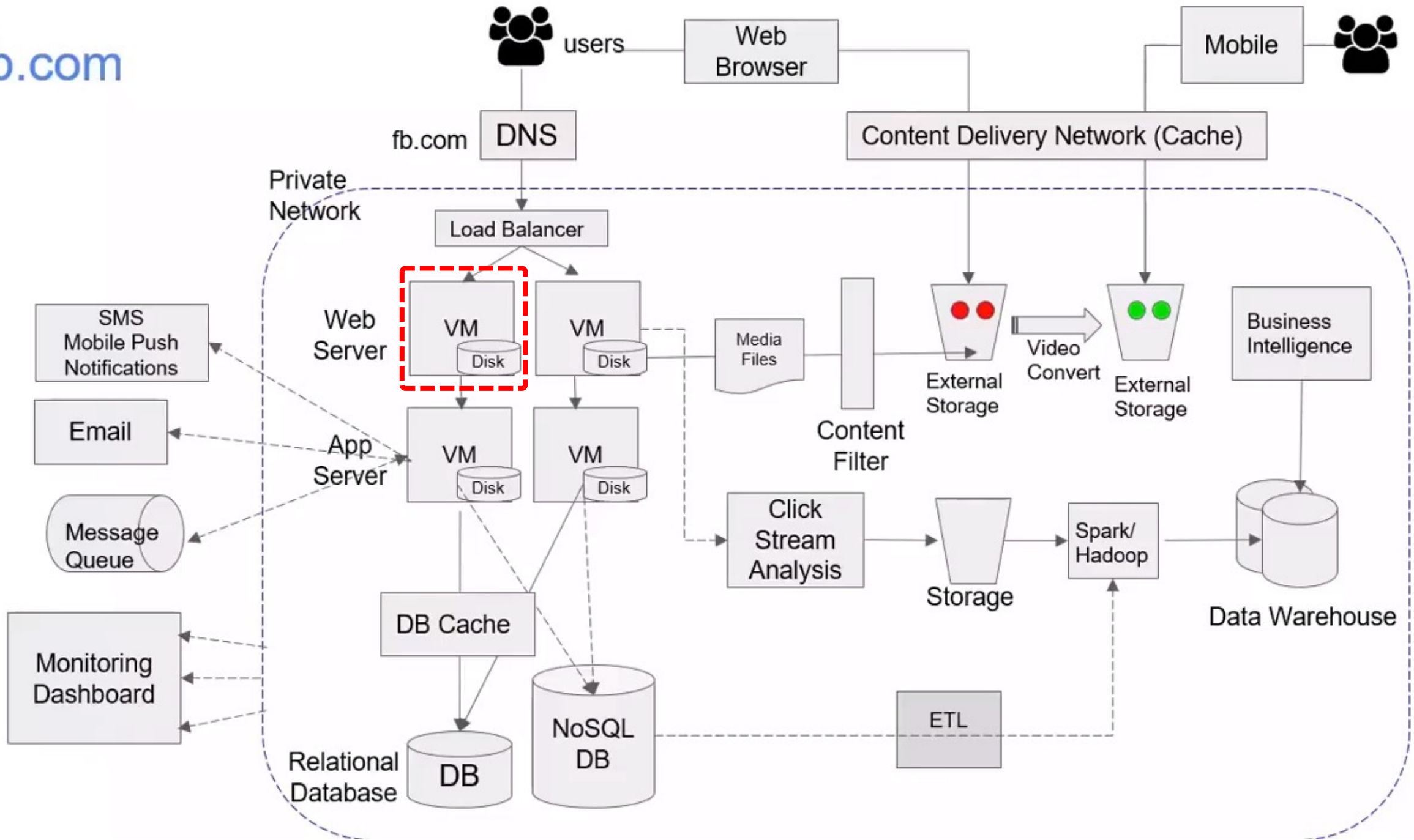
fb.com on AWS
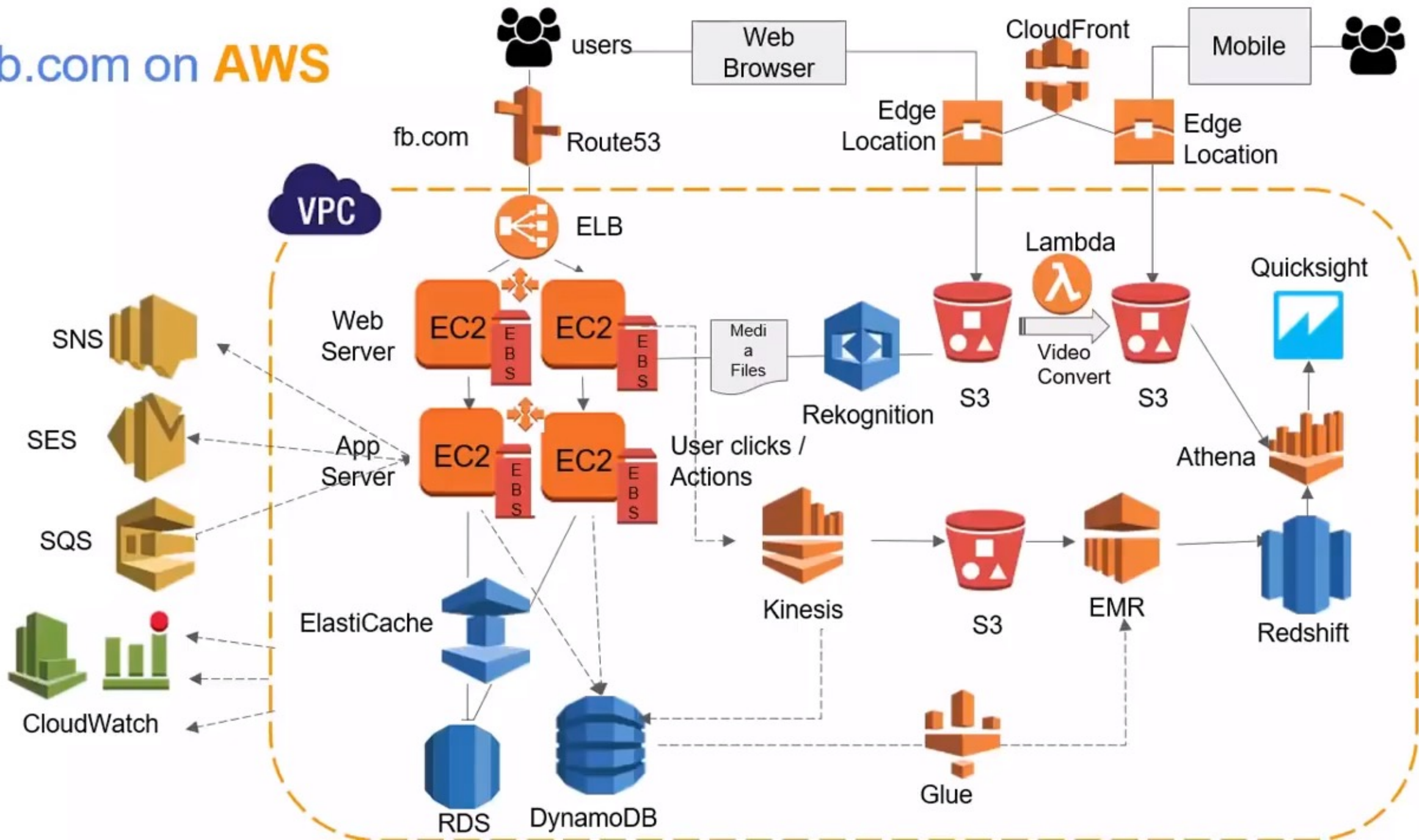
fb.com on AWS
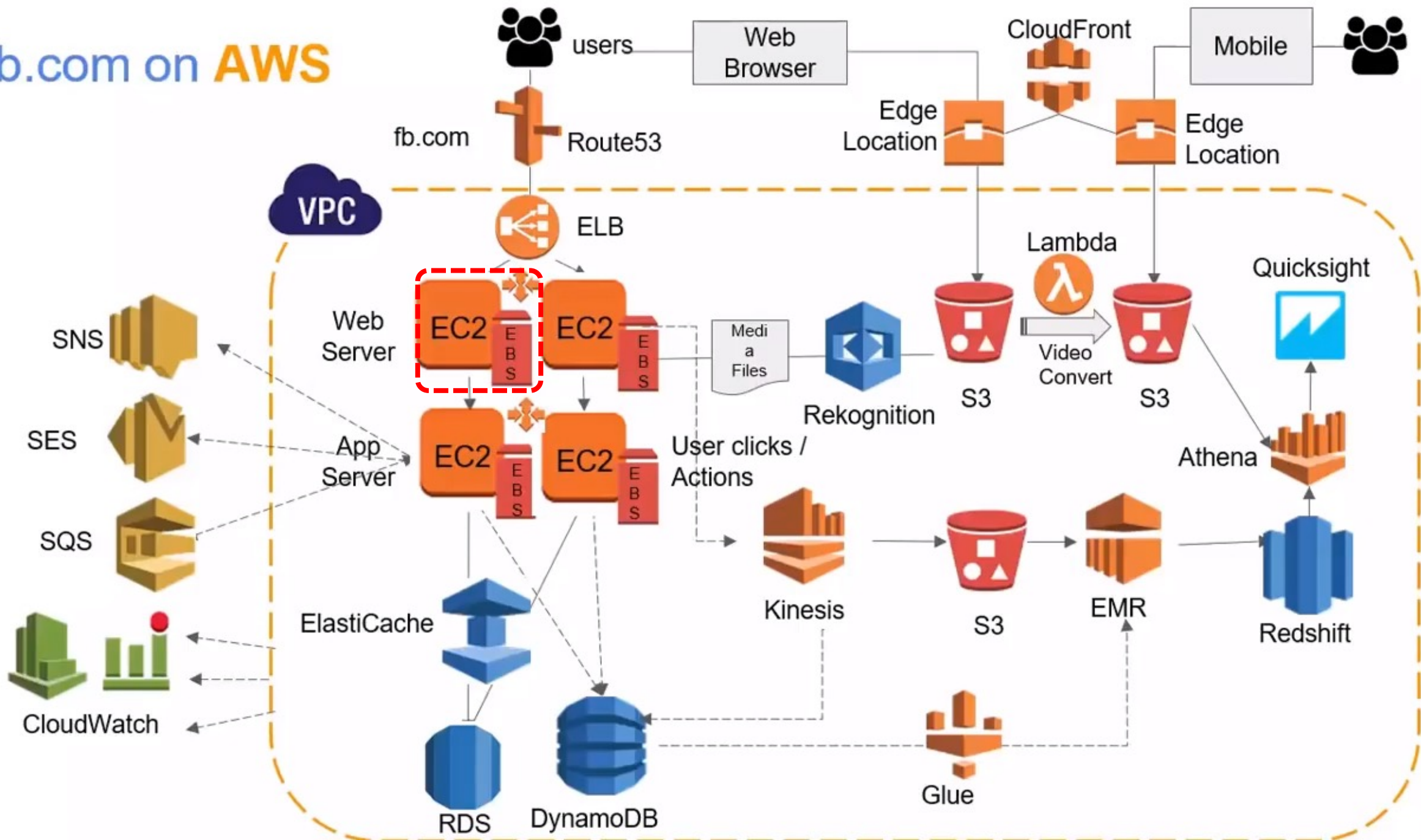
fb.com on AWS
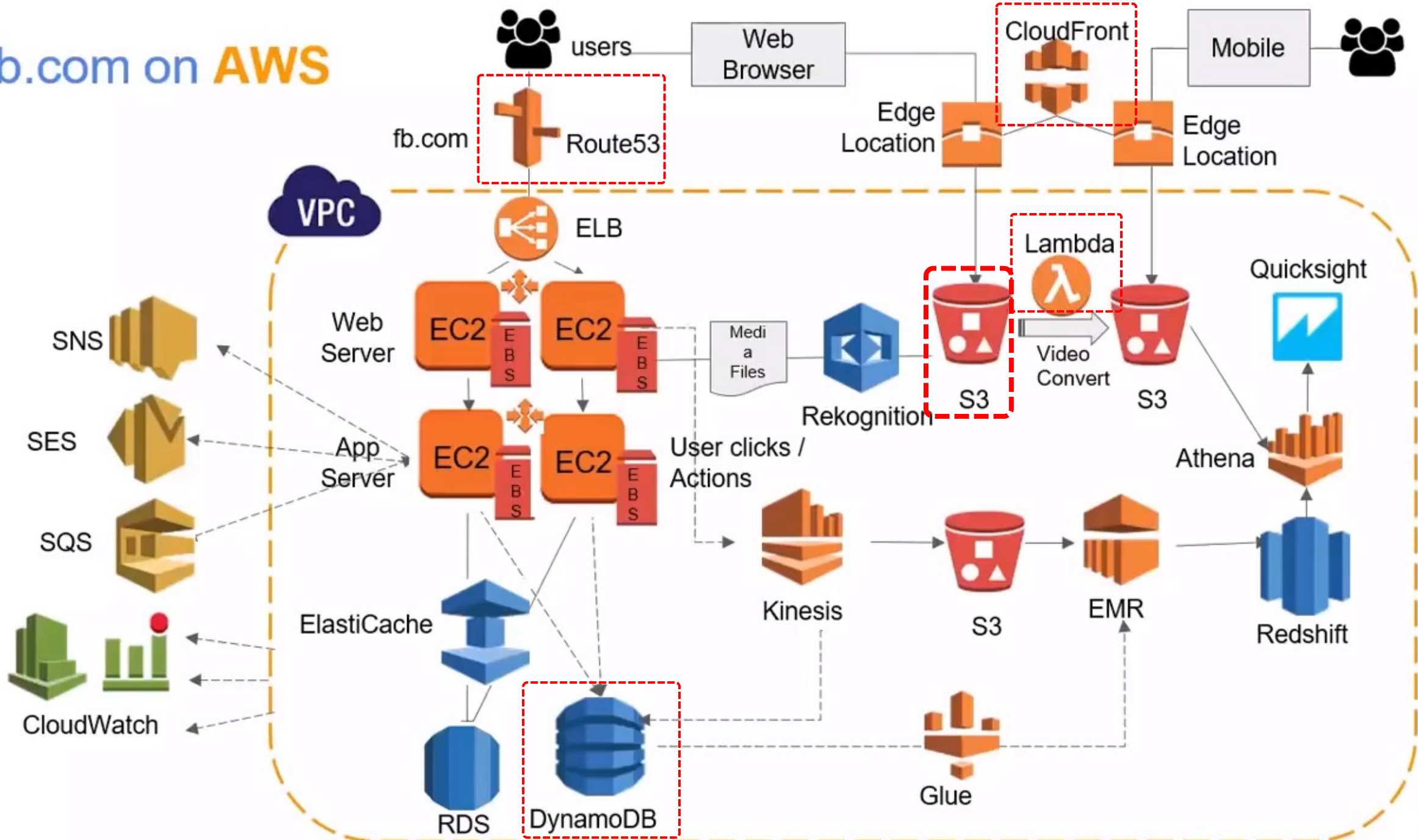
AWS Application Services
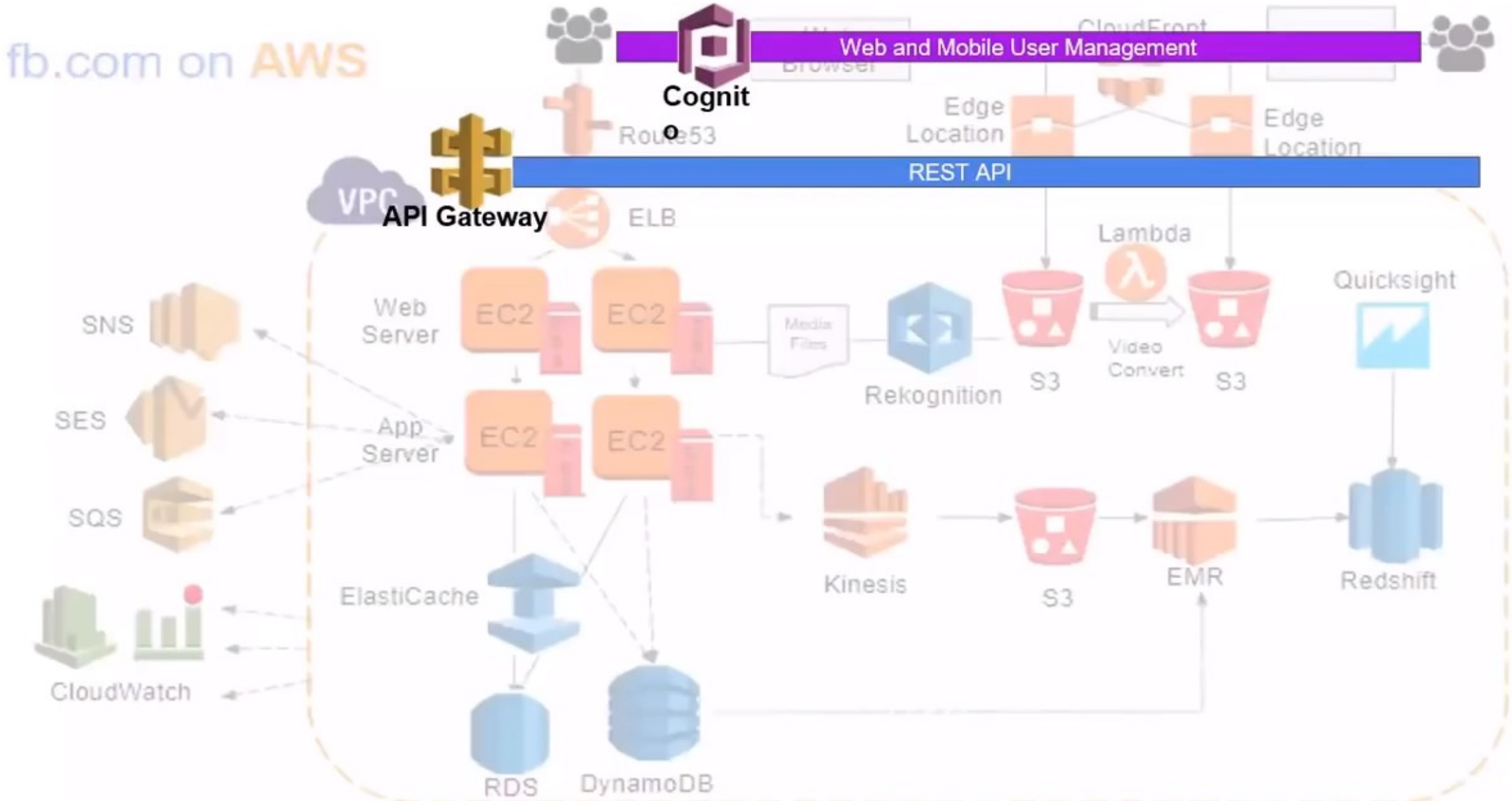
AWS Security Services
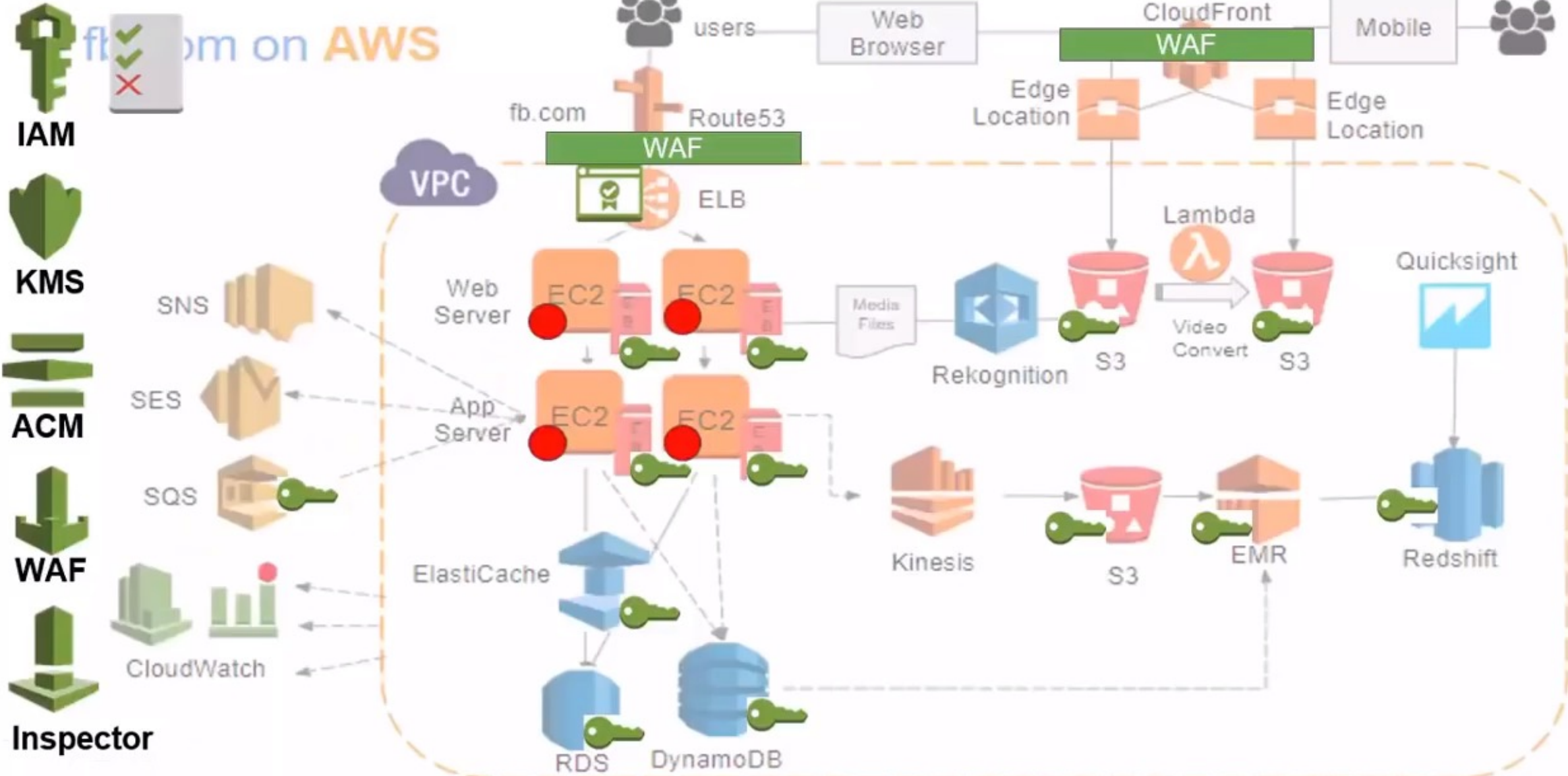
AWS Development and DevOps Services
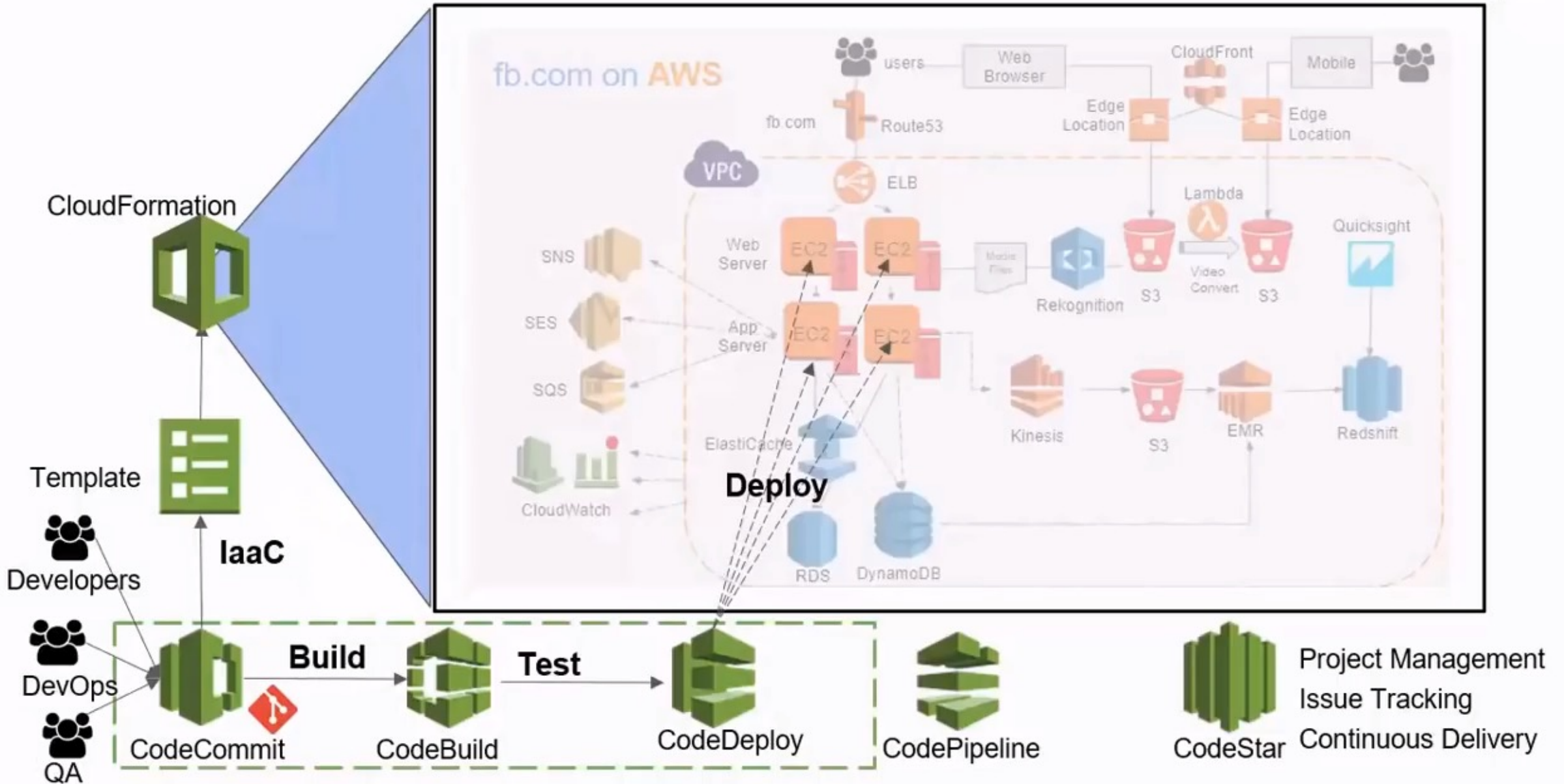
AWS Serverless Architecture

AWS Serverless Airline Booking

AWS Serverless Airline Booking Stack

UI/UX
- Quasar framework
- Vue.js
- AWS Amplify
- Stripe Elements
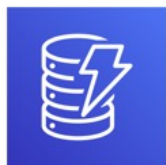
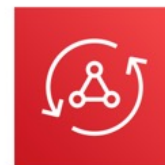## Data/Lang
- Amazon DynamoDB
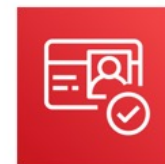- Python
- Typescript
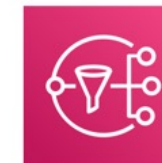- JavaScript

API/Auth
- AWS AppSync
- Amazon API Gateway
- Amazon Cognito

Messaging
- Amazon SNS
- AWS Step Functions

113

AWS Serverless Airline Booking
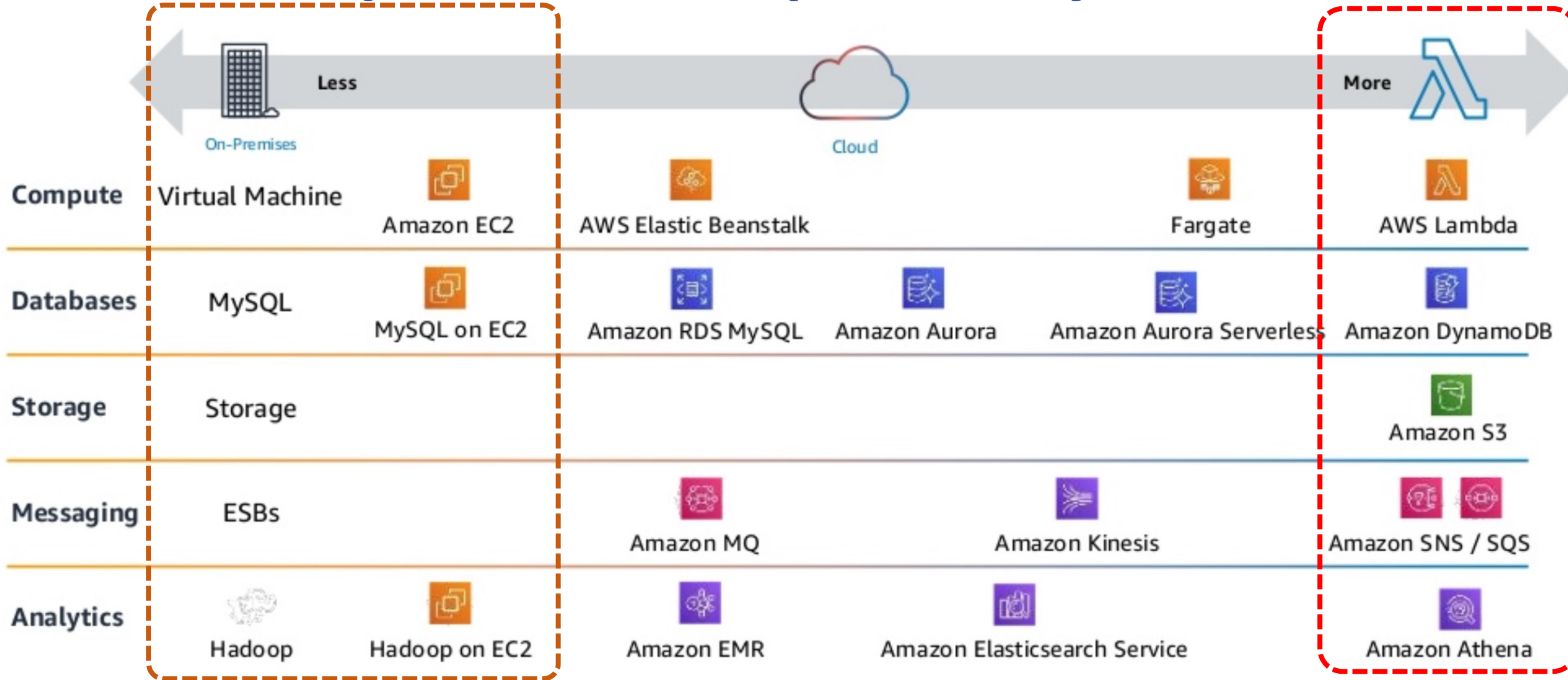# High level infrastructure architecture
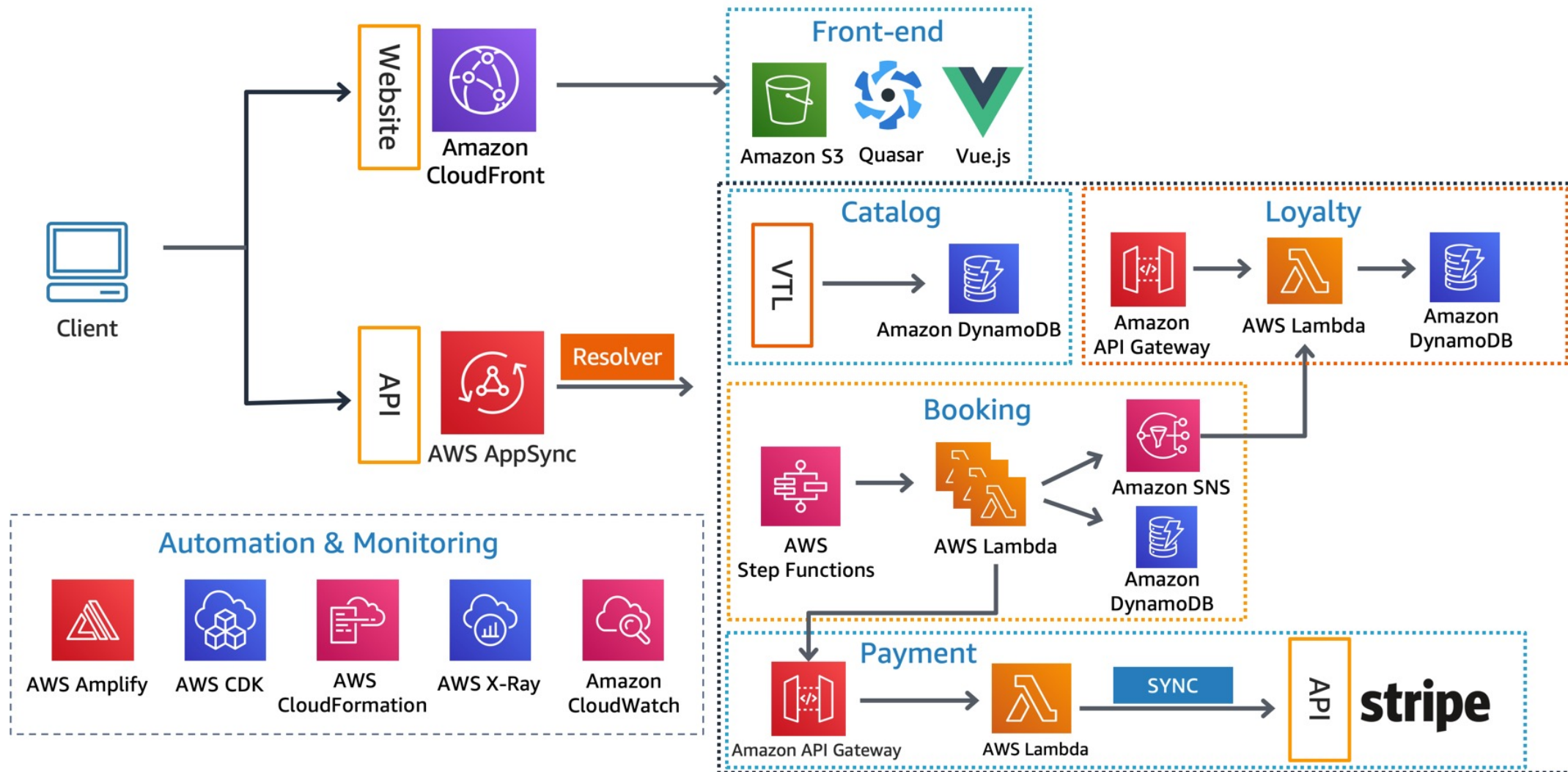
Source: https://github.com/aws-samples/aws-serverless-airline-booking

AWS Serverless Architecture
AWS Operational Responsibility Models

	On-Premises		Cloud		
	Less				More
Compute	Virtual Machine	Amazon EC2	AWS Elastic Beanstalk	Fargate	AWS Lambda
Databases	MySQL	MySQL on EC2	Amazon RDS MySQL / Amazon Aurora	Amazon Aurora Serverless	Amazon DynamoDB
Storage	Storage				Amazon S3
Messaging	ESBs		Amazon MQ	Amazon Kinesis	Amazon SNS / SQS
Analytics	Hadoop	Hadoop on EC2	Amazon EMR	Amazon Elasticsearch Service	Amazon Athena

Build a Serverless Web Application

Build a Serverless Web Application

Projects on AWS:

Build a Serverless Web Application

with AWS Lambda, Amazon API Gateway, Amazon S3, Amazon DynamoDB, and Amazon Cognito

Introduction   ① **Host a static website**   ② **Manage users**   ③ **Build a serverless backend**   ④ **Deploy a RESTful API**   ⑤ **Terminate resources**

Overview

In this tutorial, you'll create a simple serverless web application that enables users to request unicorn rides from the Wild Rydes fleet. The application will present users with an HTML based user interface for indicating the location where they would like to be picked up and will interface on the backend with a RESTful web service to submit the request and dispatch a nearby unicorn. The application will also provide facilities for users to register with the service and log in before requesting rides.
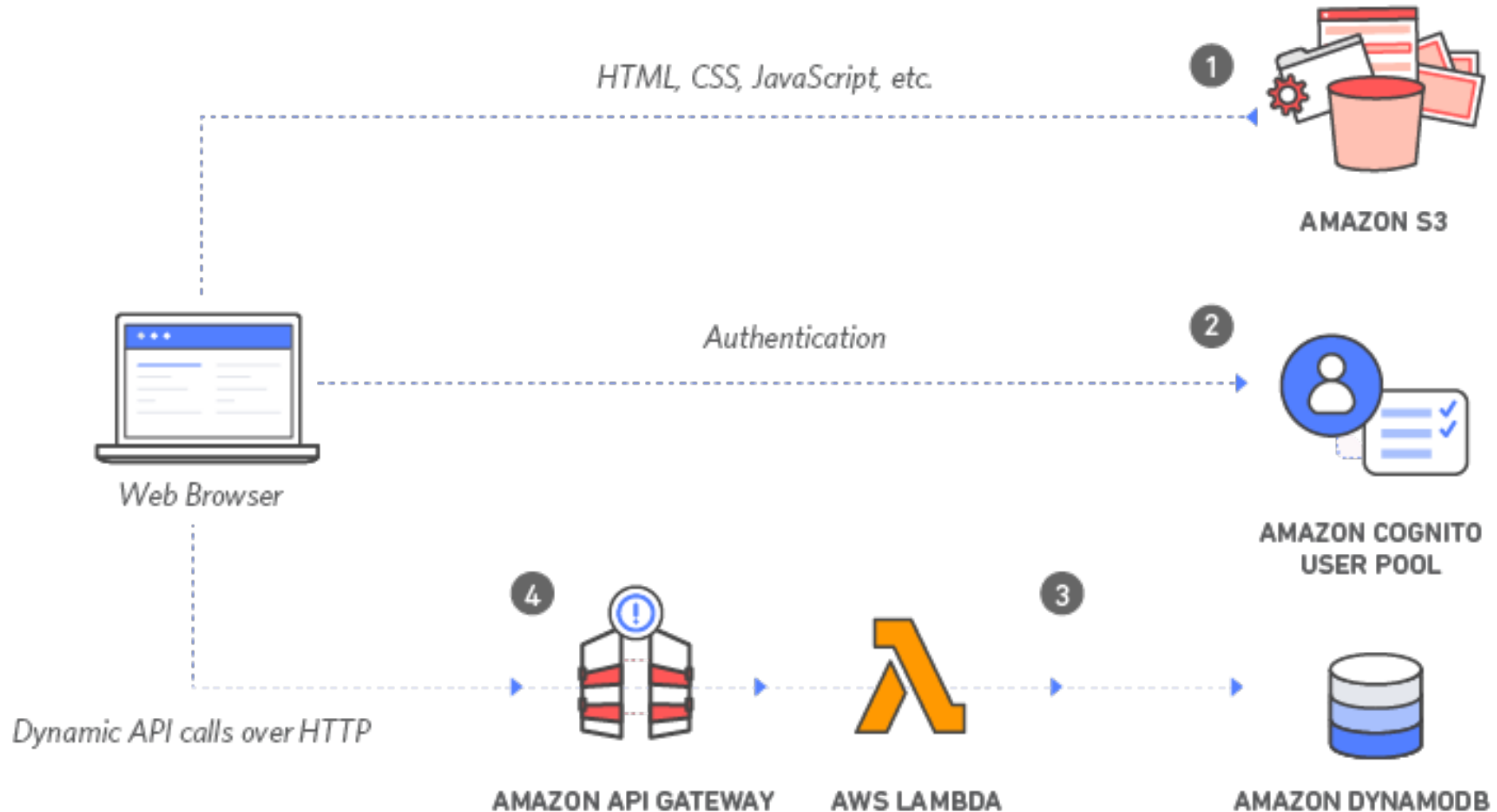
AWS Experience: Beginner

Time to complete: 2 hours

Cost to complete: Each service used in this architecture is eligible for the AWS Free Tier. If you are outside the usage limits of the Free Tier, completing this tutorial will cost you less than $0.25*.

Application Architecture

Build a Serverless Web Application
with Amazon S3, AWS Lambda, Amazon API Gateway, Amazon DynamoDB, and Amazon Cognito

HTML, CSS, JavaScript, etc.

① **AMAZON S3**

Authentication

② **AMAZON COGNITO USER POOL**

Web Browser

④

③

Dynamic API calls over HTTP

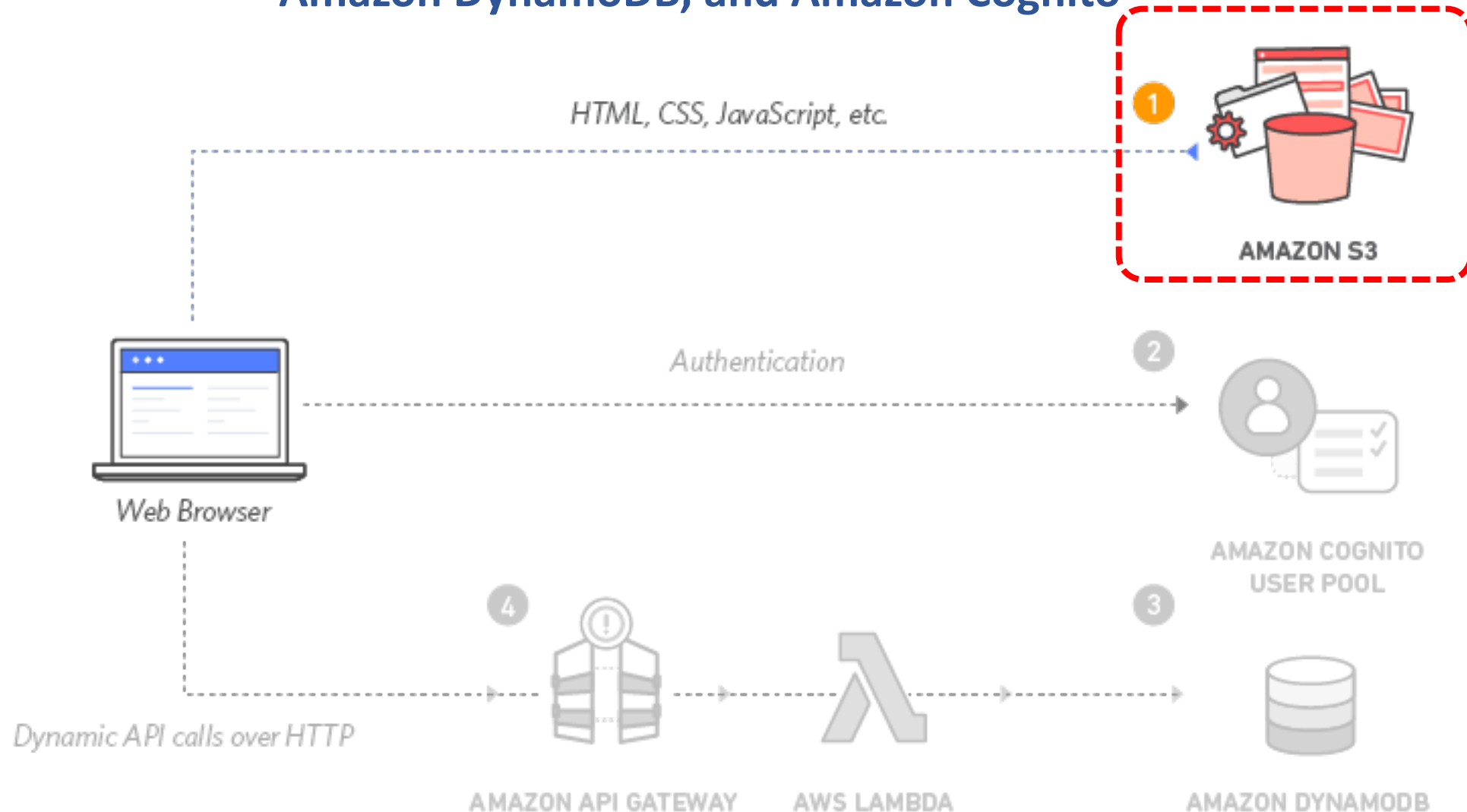**AMAZON API GATEWAY**   **AWS LAMBDA**   **AMAZON DYNAMODB**

Build a Serverless Web Application

with Amazon S3, AWS Lambda, Amazon API Gateway, Amazon DynamoDB, and Amazon Cognito

1

Build a Serverless Web Application

with Amazon S3, AWS Lambda, Amazon API Gateway, Amazon DynamoDB, and Amazon Cognito

1

HTML, CSS, JavaScript, etc.

AMAZON S3

Static Web Hosting

Amazon S3 hosts static web resources including HTML, CSS, JavaScript, and image files which are loaded in the user's browser.
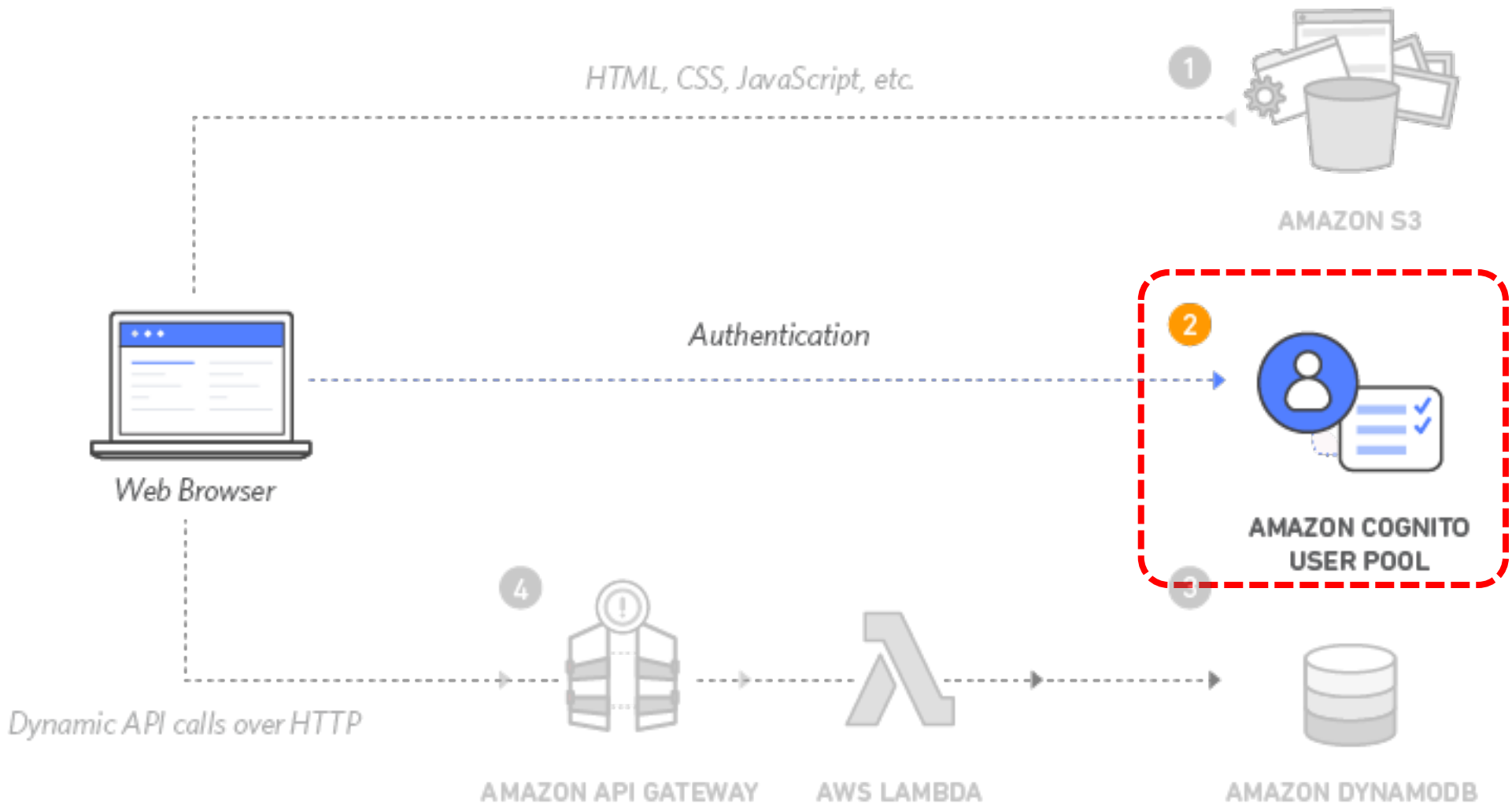
2

AMAZON COGNITO
USER POOL

3

AMAZON DYNAMODB

Build a Serverless Web Application

with Amazon S3, AWS Lambda, Amazon API Gateway, Amazon DynamoDB, and Amazon Cognito

2

HTML, CSS, JavaScript, etc.

1 AMAZON S3

Authentication

Web Browser

2 AMAZON COGNITO USER POOL

4

Dynamic API calls over HTTP

3

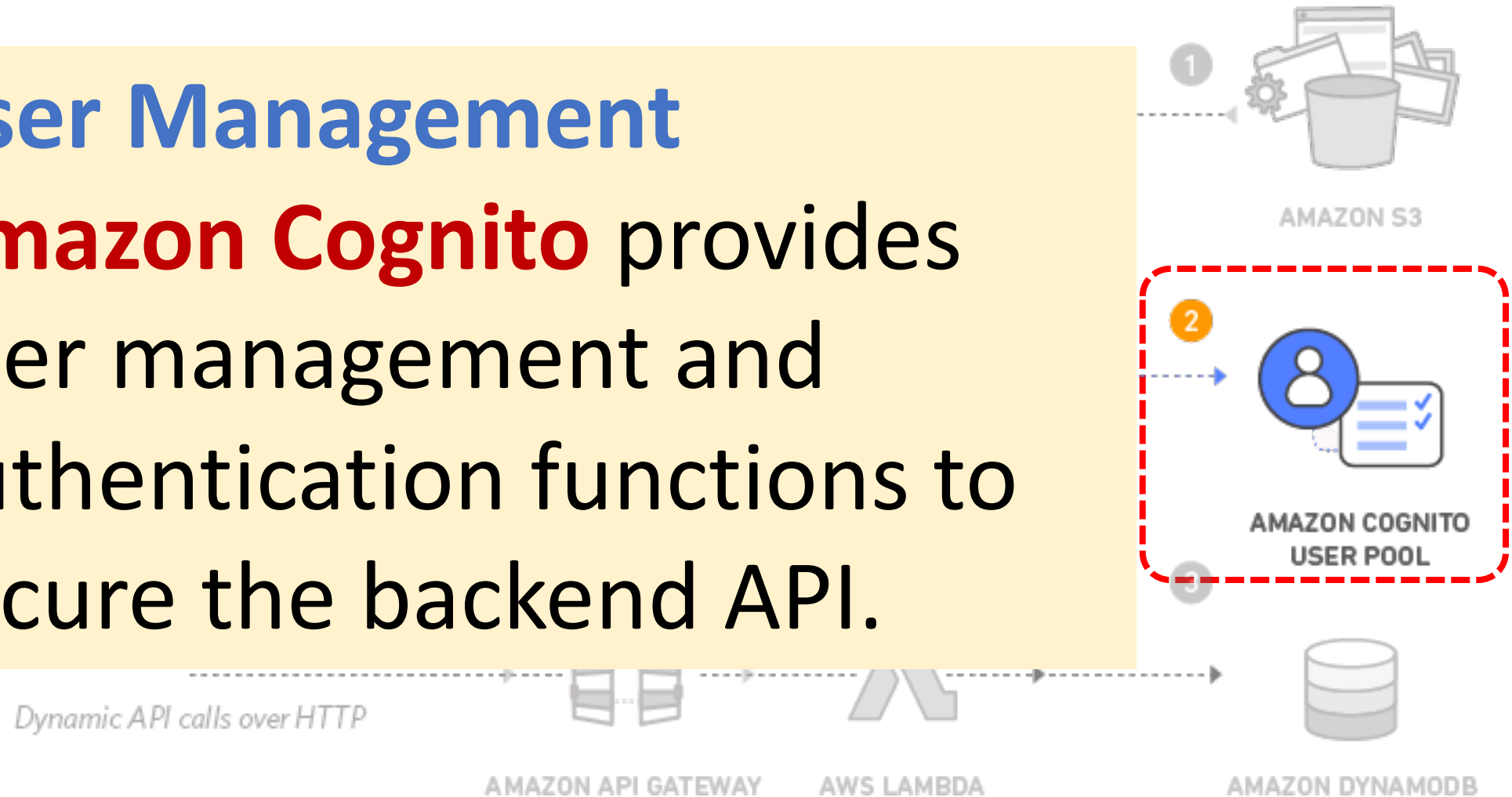AMAZON API GATEWAY   AWS LAMBDA   AMAZON DYNAMODB

Build a Serverless Web Application

with Amazon S3, AWS Lambda, Amazon API Gateway, Amazon DynamoDB, and Amazon Cognito

2 User Management

Amazon Cognito provides user management and authentication functions to secure the backend API.
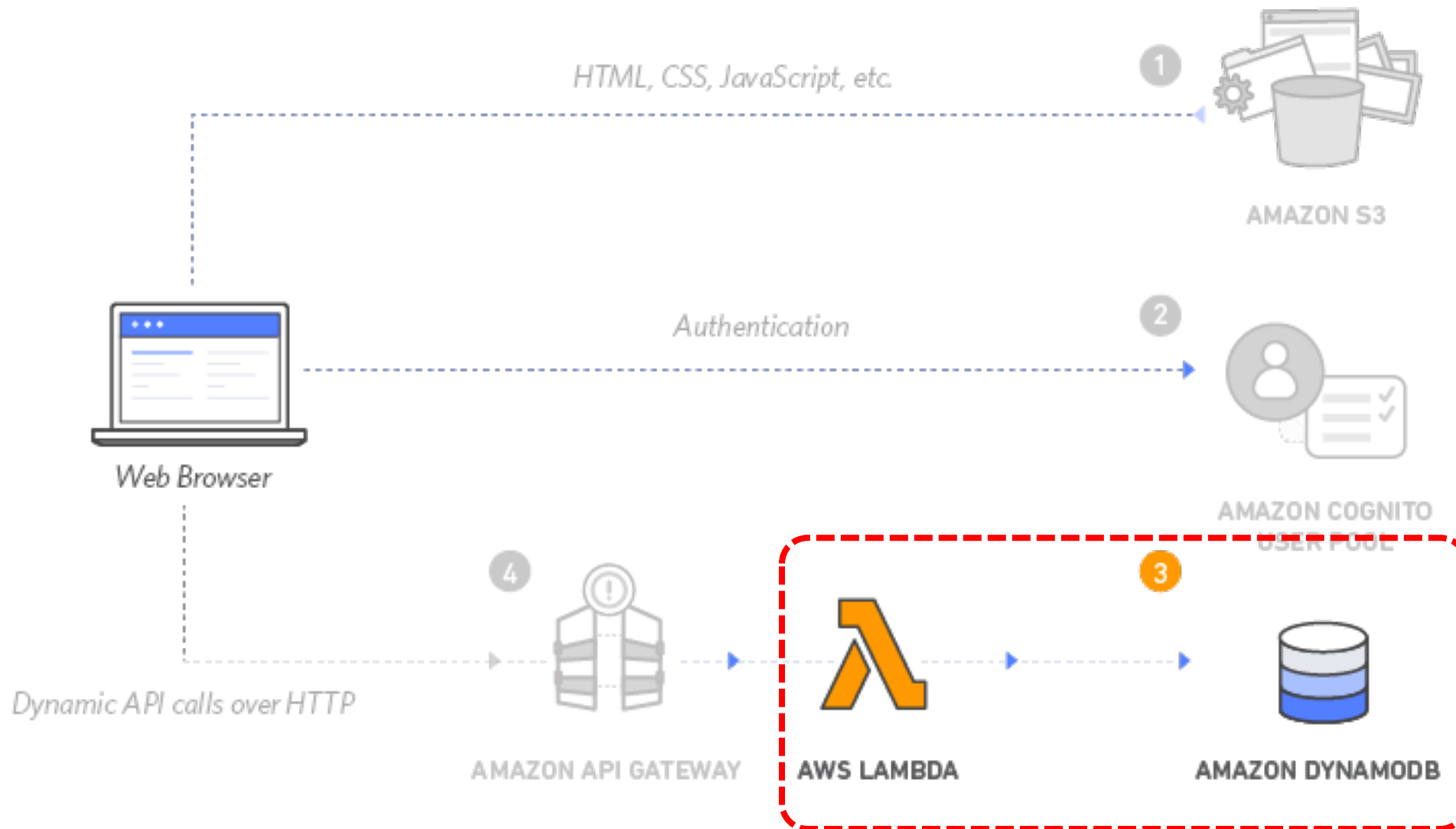
AMAZON S3

AMAZON COGNITO USER POOL

Dynamic API calls over HTTP

AMAZON API GATEWAY

AWS LAMBDA

AMAZON DYNAMODB

Build a Serverless Web Application

with Amazon S3, AWS Lambda, Amazon API Gateway, Amazon DynamoDB, and Amazon Cognito

3

HTML, CSS, JavaScript, etc.

① AMAZON S3

Web Browser

Authentication

② AMAZON COGNITO USER POOL

④

Dynamic API calls over HTTP
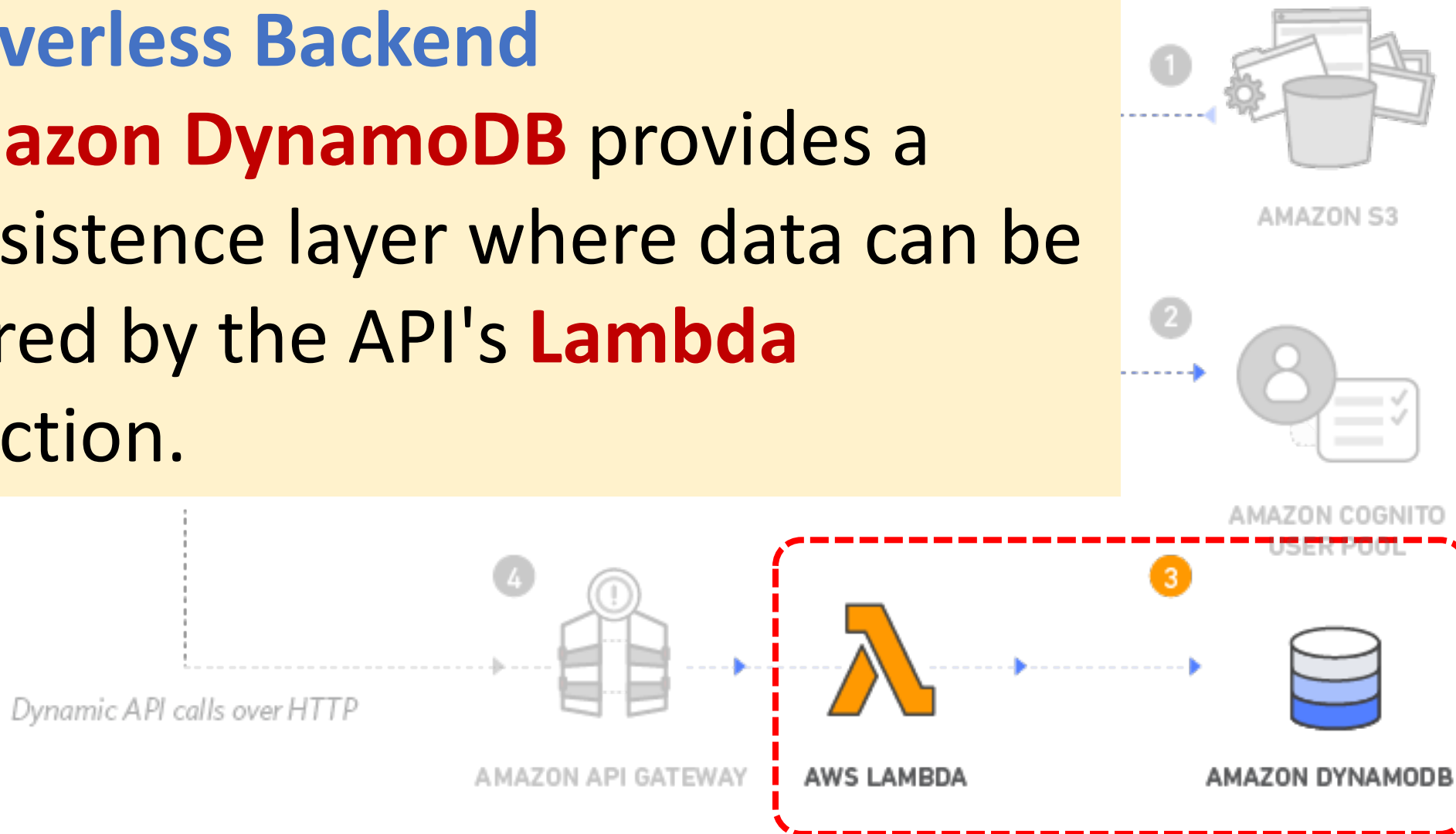
AMAZON API GATEWAY

③ AWS LAMBDA

AMAZON DYNAMODB

Build a Serverless Web Application

with Amazon S3, AWS Lambda, Amazon API Gateway, Amazon DynamoDB, and Amazon Cognito

3 **Serverless Backend**

Amazon DynamoDB provides a persistence layer where data can be stored by the API's **Lambda** function.
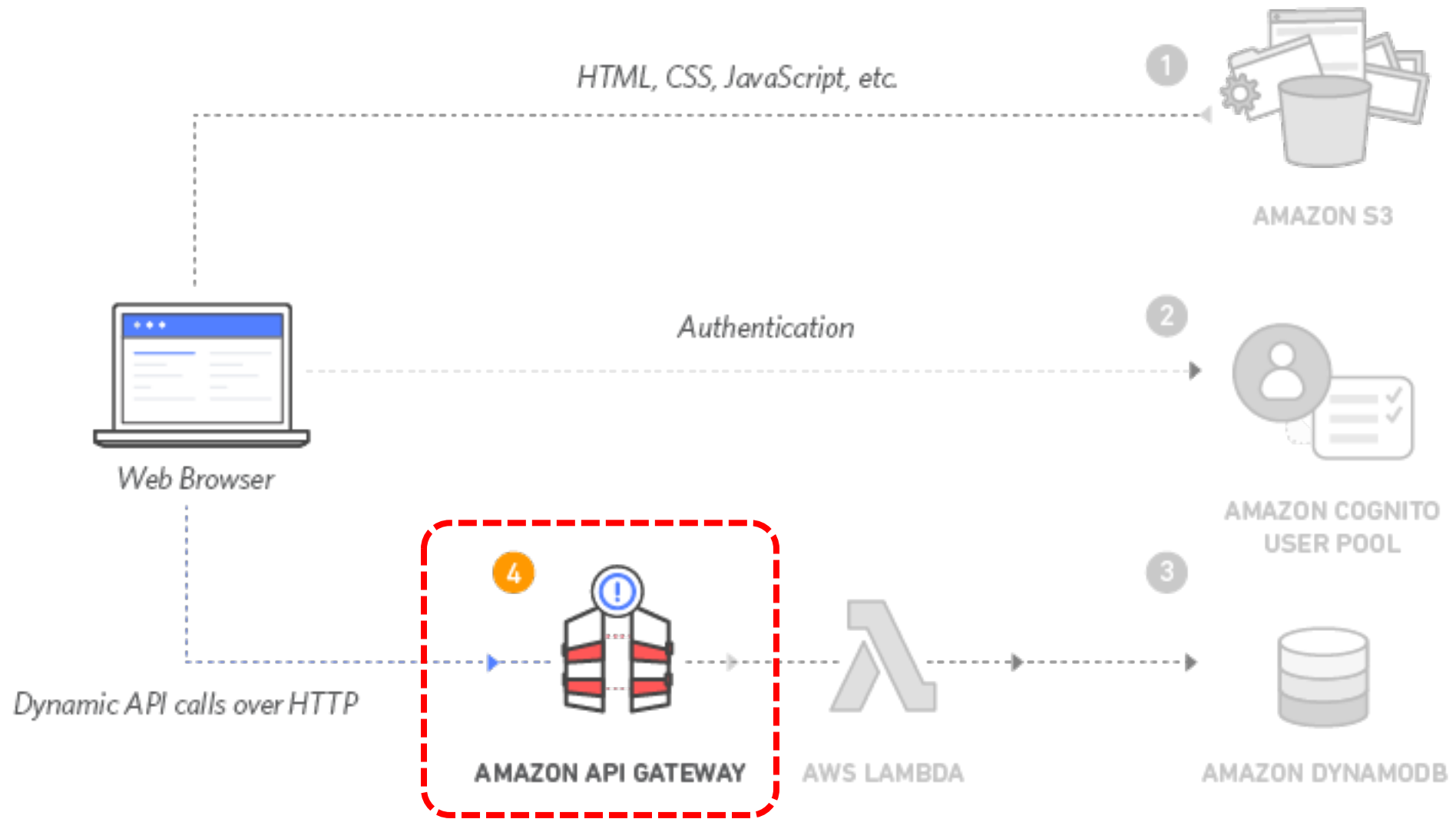
AMAZON S3

AMAZON COGNITO USER POOL

Dynamic API calls over HTTP

AMAZON API GATEWAY

AWS LAMBDA

AMAZON DYNAMODB

Build a Serverless Web Application

with Amazon S3, AWS Lambda, Amazon API Gateway, Amazon DynamoDB, and Amazon Cognito

4

HTML, CSS, JavaScript, etc.

① AMAZON S3

Web Browser

Authentication

② AMAZON COGNITO USER POOL

④ AMAZON API GATEWAY

Dynamic API calls over HTTP
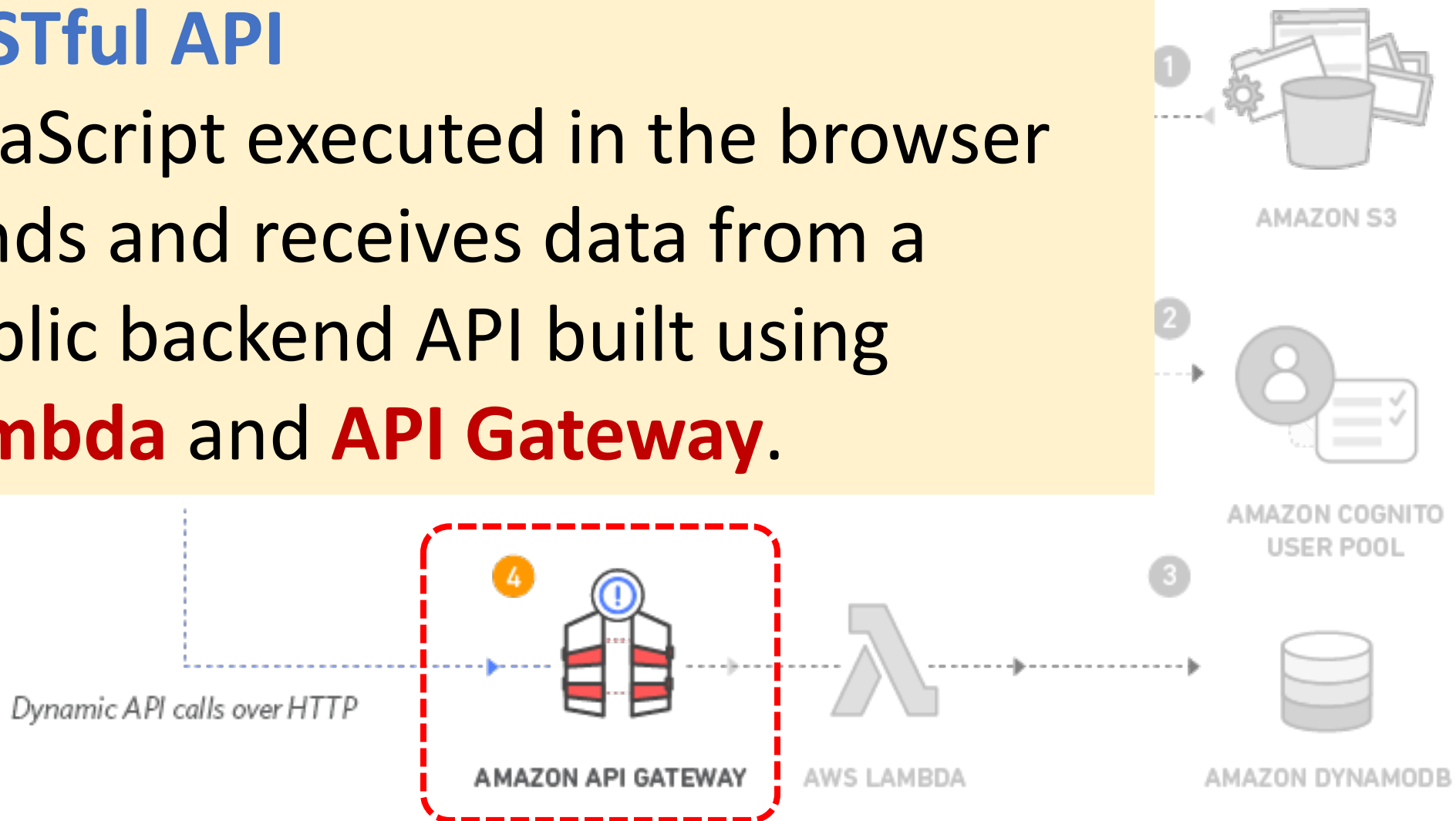
③

AWS LAMBDA

AMAZON DYNAMODB

Build a Serverless Web Application

with Amazon S3, AWS Lambda, Amazon API Gateway, Amazon DynamoDB, and Amazon Cognito

4

RESTful API

JavaScript executed in the browser sends and receives data from a public backend API built using **Lambda** and **API Gateway**.

5

Terminate resources

Resource Cleanup

You will terminate an **Amazon S3** bucket, an **Amazon Cognito** User Pool, an **AWS Lambda** function, an **IAM** role, a **DynamoDB** table, a **REST API**, and a **CloudWatch** Log.

It is a best practice to delete resources you are no longer using to avoid unwanted charges.

Summary

- **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.**

Summary

- **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.

Summary

- **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.**

Summary

- **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.

Summary

- **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.

References

- Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

- Ian Sommerville (2015), Software Engineering, 10th Edition, Pearson.

- Titus Winters, Tom Manshreck, and Hyrum Wright (2020), Software Engineering at Google: Lessons Learned from Programming Over Time, O'Reilly Media.

- Project Management Institute (2021), A Guide to the Project Management Body of Knowledge (PMBOK Guide) – Seventh Edition and The Standard for Project Management, PMI.

- Project Management Institute (2017), A Guide to the Project Management Body of Knowledge (PMBOK Guide), Sixth Edition, Project Management Institute.

- Project Management Institute (2017), Agile Practice Guide, Project Management Institute.