CLOUD COMPUTING (Week-1)

USAMA MUSHARAF

LECTURER (Department of Software Engineering)

FAST-NUCES PESHAWAR

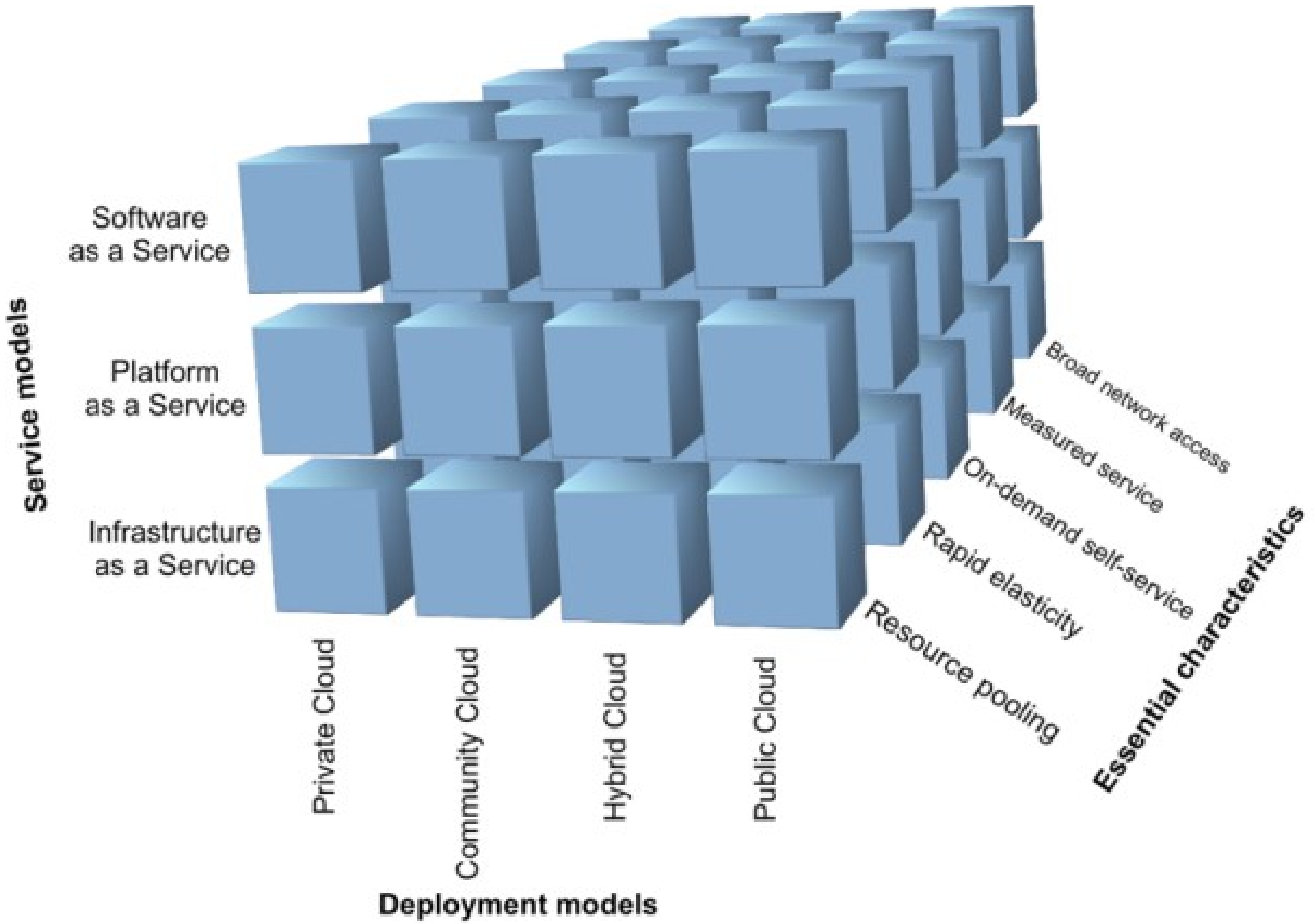
CLOUD COMPUTING

WHAT IS CLOUD COMPUTING

Cloud computing is a model for enabling convenient, *on-demand network access to a shared pool of configurable computing resources* (e.g., networks, servers, storage, applications, and services) that can be *rapidly provisioned* and released with *minimal management* effort or service provider interaction.

This cloud model promotes availability and is composed of *five essential characteristics, three service models, and four deployment models."*

WHAT IS CLOUD COMPUTING



ESSENTIAL CHARACTERISTICS

ESSENTIAL CHARACTERISTICS

On-demand self-service

- a consumer can unilaterally provision computing capabilities without human interaction with the service provider
- computing capabilities

  – server time, network storage, number of servers etc.

ESSENTIAL CHARACTERISTICS

Broad network access

- capabilities are
  - – available over the network
  - – accessed through standard mechanisms

ESSENTIAL CHARACTERISTICS

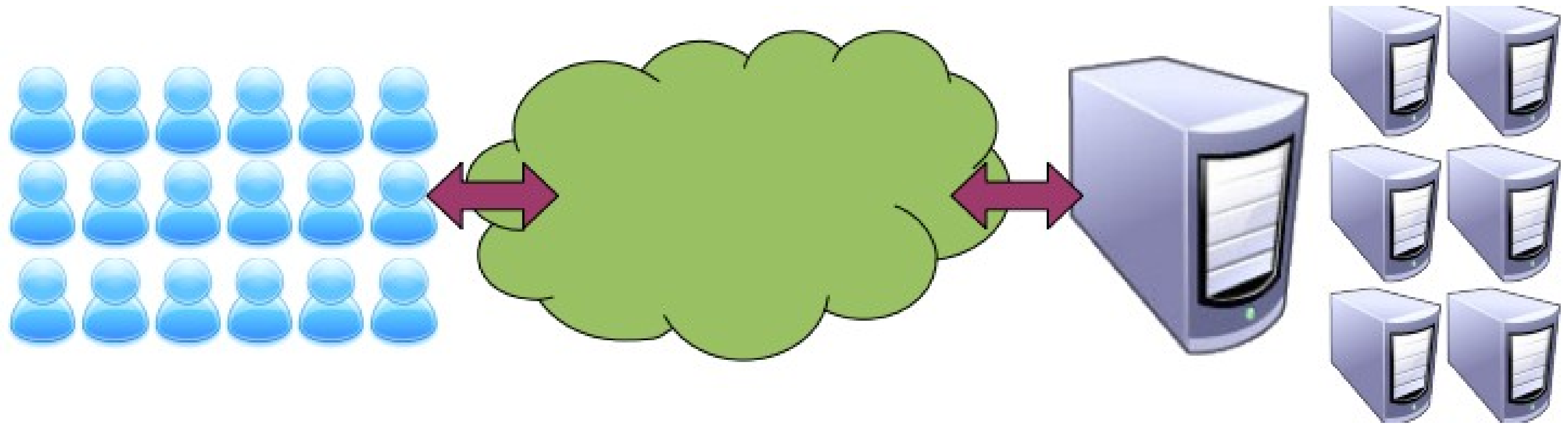
Multi-tenancy / Resource pooling

- provider’s computing resources are pooled to serve multiple consumers
- computing resources
  - – storage, processing, memory, network bandwidth and virtual machines
- location independence
  - – no control over the exact location of the resources
- has major implications
  - – performance, scalability, security

ESSENTIAL CHARACTERISTICS

Rapid elasticity

- capabilities can be rapidly and elastically provisioned
- unlimited virtual resources



ESSENTIAL CHARACTERISTICS

Measured service

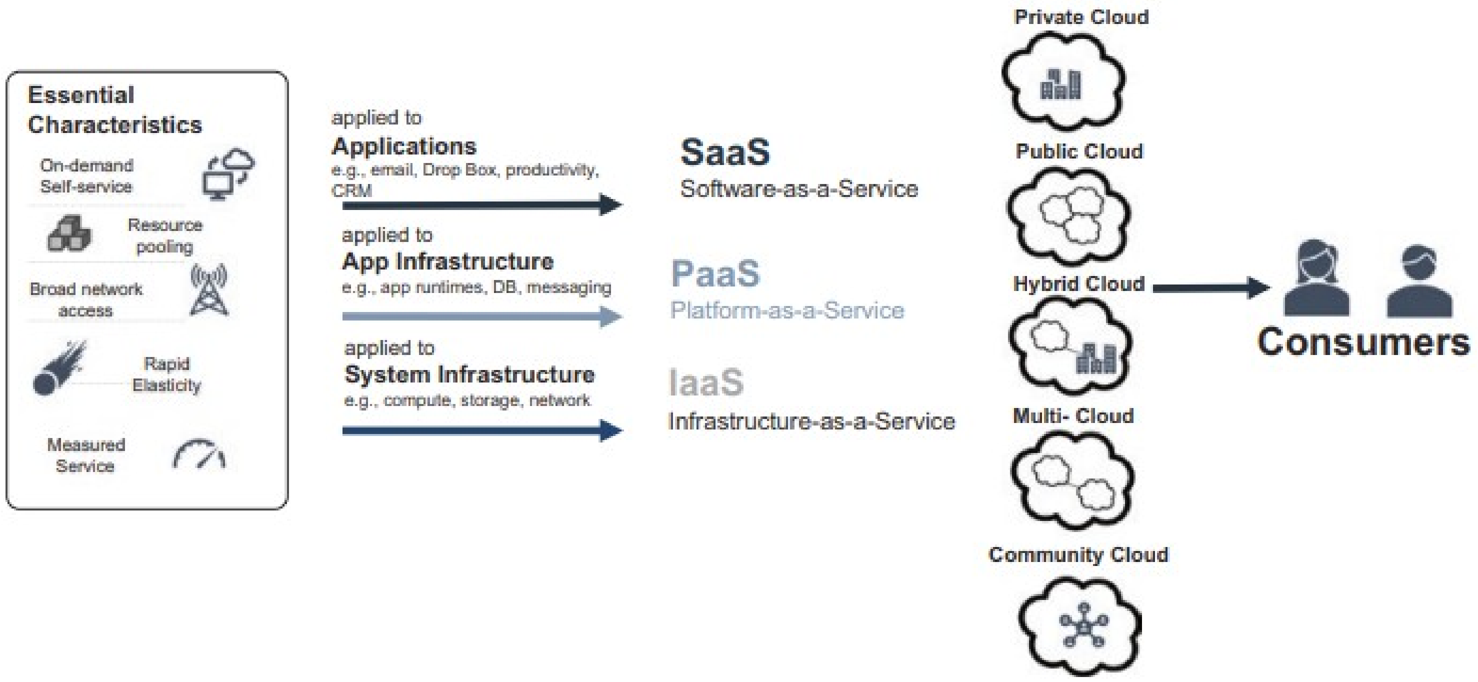
- metering capability of service/resource abstractions
  - storage
  - processing
  - bandwidth
  - active user accounts

CLOUD SERVICES

UNDERSTANDING CLOUD SERVICES

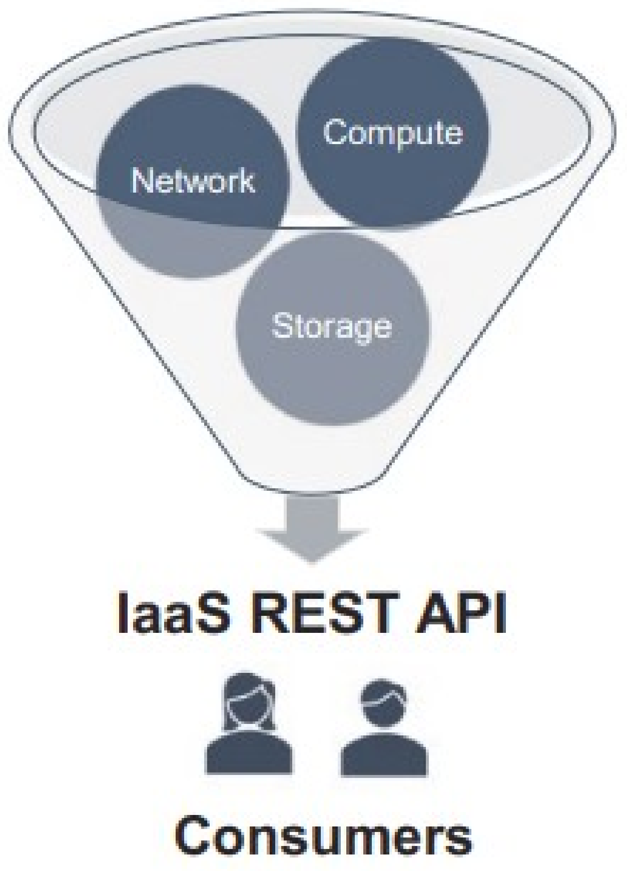
Cloud computing provides various services delivered on-demand to the customers over the Internet.

These services are designed to provide easy, affordable access to applications and resources, without the need for internal infrastructure or hardware.

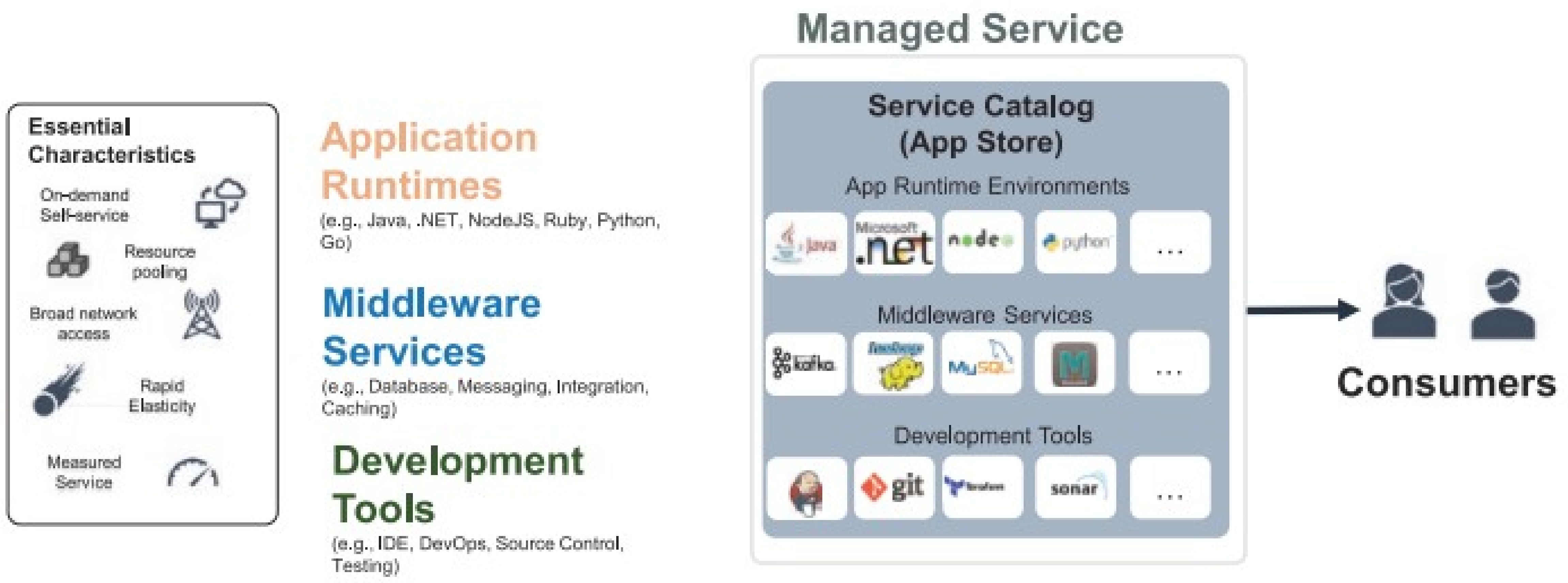


INFRASTRUCTURE AS A SERVICE

IaaS is a form of cloud computing that delivers fundamental compute, network, and storage resources to consumers on-demand over the Internet and on a pay-as-you-go basis.



PLATFORM AS A SERVICE

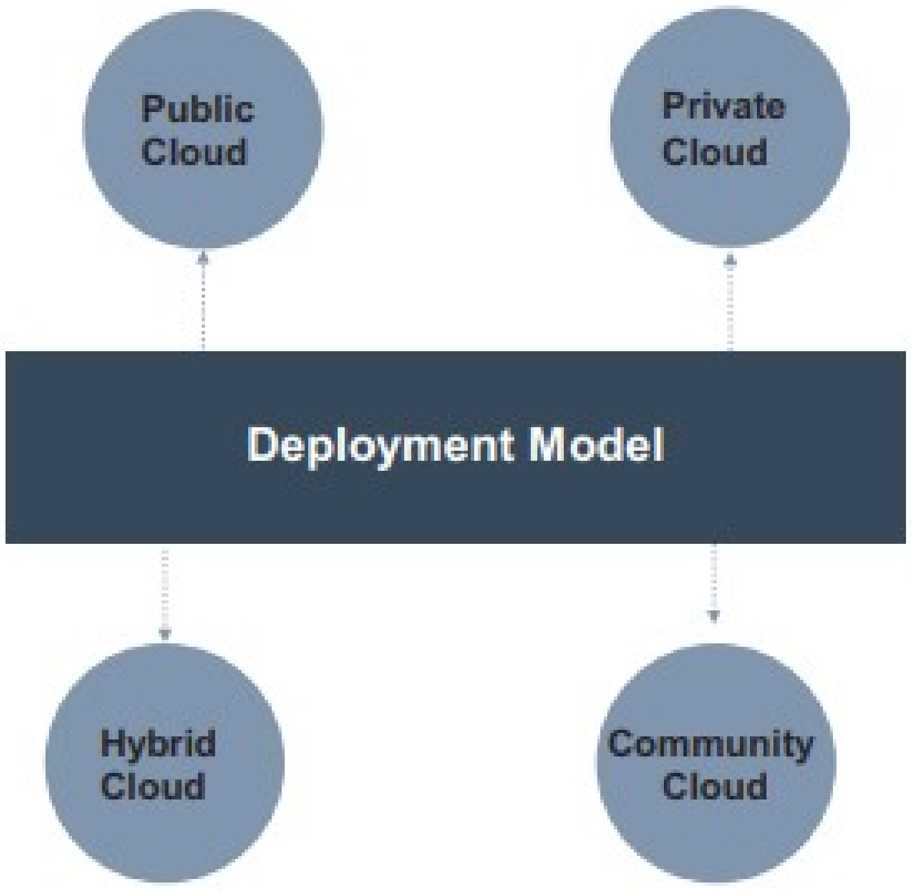


SOFTWARE AS A SERVICE

Software as a service (SaaS) is a software distribution model in which a cloud provider hosts applications and makes them available to end users over the internet.

CLOUD DEPLOYMENT MODEL

CLOUD COMPUTING DEPLOYMENT MODELS



PUBLIC CLOUD

In public cloud computing, the use of computing resources and software is based on the subscription model.

The resources are provisioned for open use by the public and various organizations.

PRIVATE OR ON-PREMISES CLOUD

In private cloud computing, the infrastructure is used by a single organization.

Such an infrastructure is managed within an organization or dedicated infrastructure in the cloud.

HYBRID CLOUD

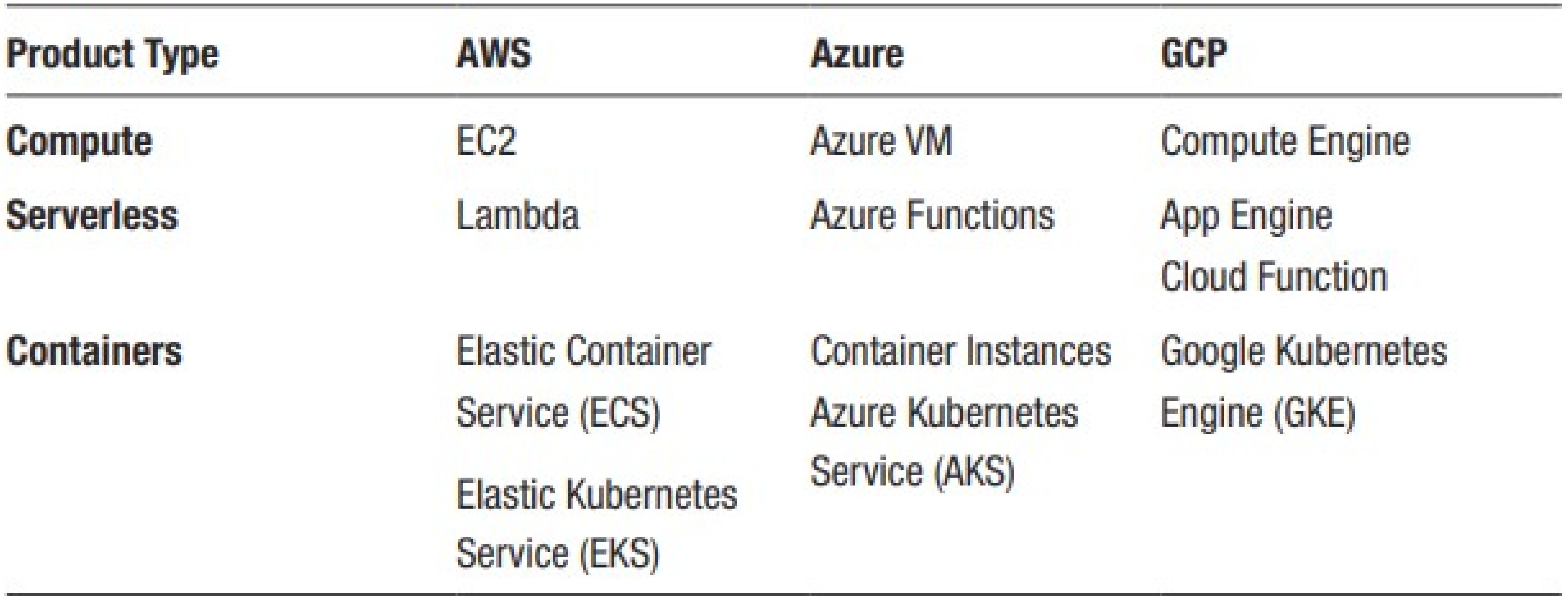
A hybrid model is a combination or composition of two or more distinct cloud infrastructures, such as private, public.

COMMUNITY CLOUD

In community cloud computing, multiple organizations share computing resources that are part of a community.

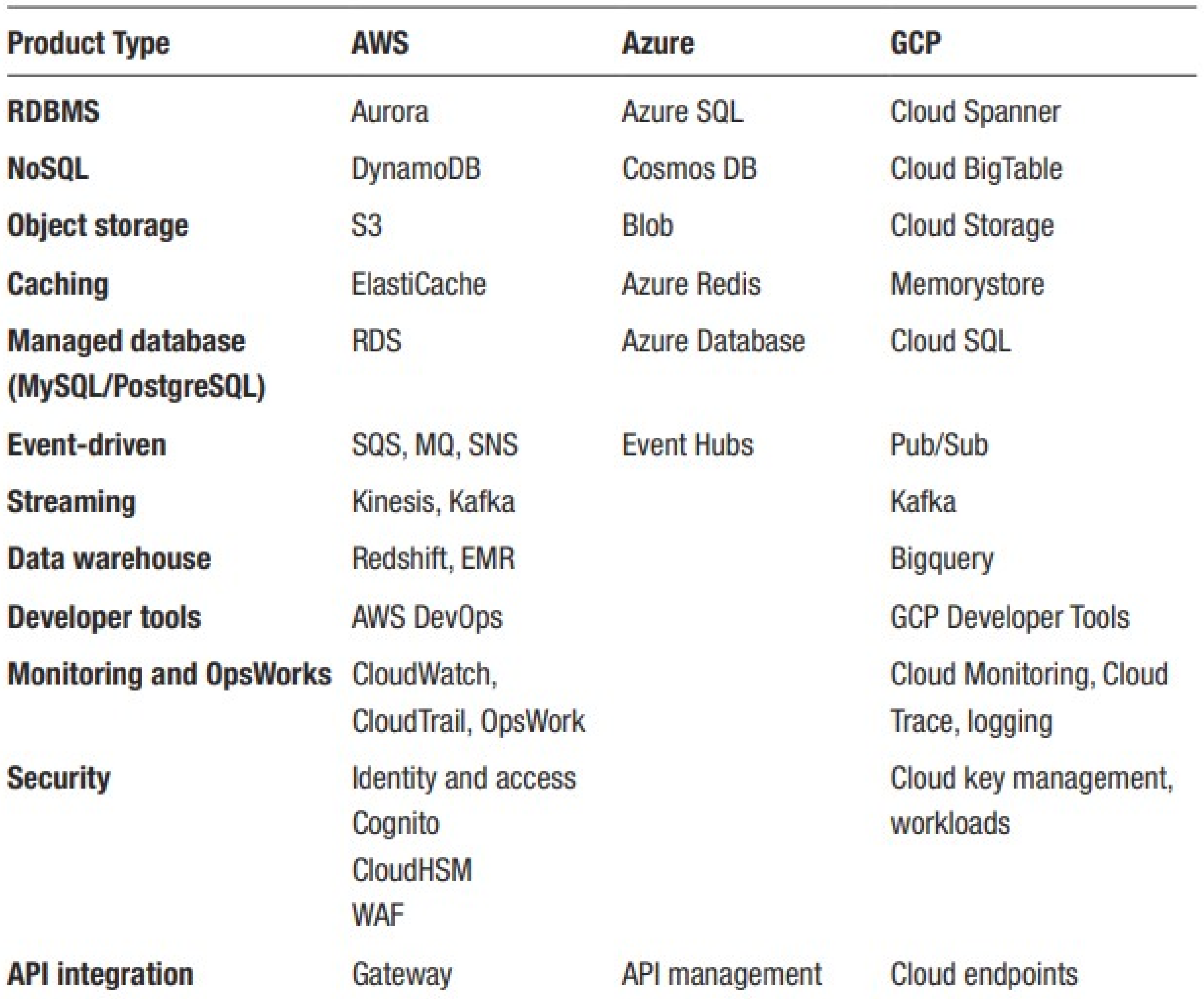
The community must have a shared concern, for example, shared policies, SLAs, shared security requirements, etc.

CLOUD SERVICES



Product Type	AWS	Azure	GCP
Compute	EC2	Azure VM	Compute Engine
Serverless	Lambda	Azure Functions	App Engine Cloud Function
Containers	Elastic Container Service (ECS) Elastic Kubernetes Service (EKS)	Container Instances Azure Kubernetes Service (AKS)	Google Kubernetes Engine (GKE)

CLOUD SERVICES



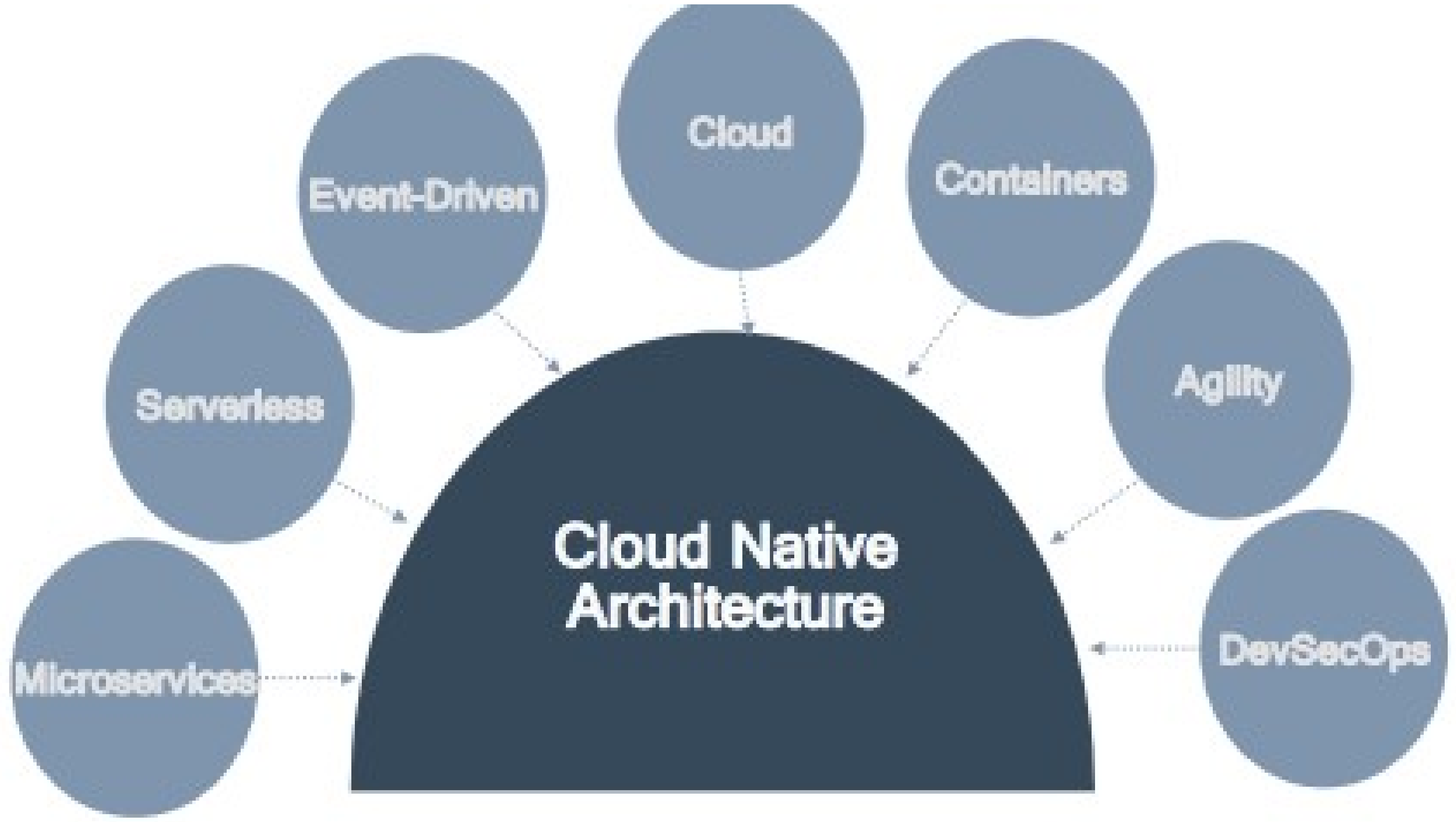
Product Type	AWS	Azure	GCP
RDBMS	Aurora	Azure SQL	Cloud Spanner
NoSQL	DynamoDB	Cosmos DB	Cloud BigTable
Object storage	S3	Blob	Cloud Storage
Caching	ElastiCache	Azure Redis	Memorystore
Managed database (MySQL/PostgreSQL)	RDS	Azure Database	Cloud SQL
Event-driven	SQS, MQ, SNS	Event Hubs	Pub/Sub
Streaming	Kinesis, Kafka		Kafka
Data warehouse	Redshift, EMR		Bigquery
Developer tools	AWS DevOps		GCP Developer Tools
Monitoring and OpsWorks	CloudWatch, CloudTrail, OpsWork		Cloud Monitoring, Cloud Trace, logging
Security	Identity and access Cognito CloudHSM WAF		Cloud key management, workloads
API integration	Gateway	API management	Cloud endpoints

CLOUD NATIVE COMPUTING

WHAT IS CLOUD NATIVE COMPUTING?

A modern way of developing application using some of the latest state of the art methods, technologies, and tools.

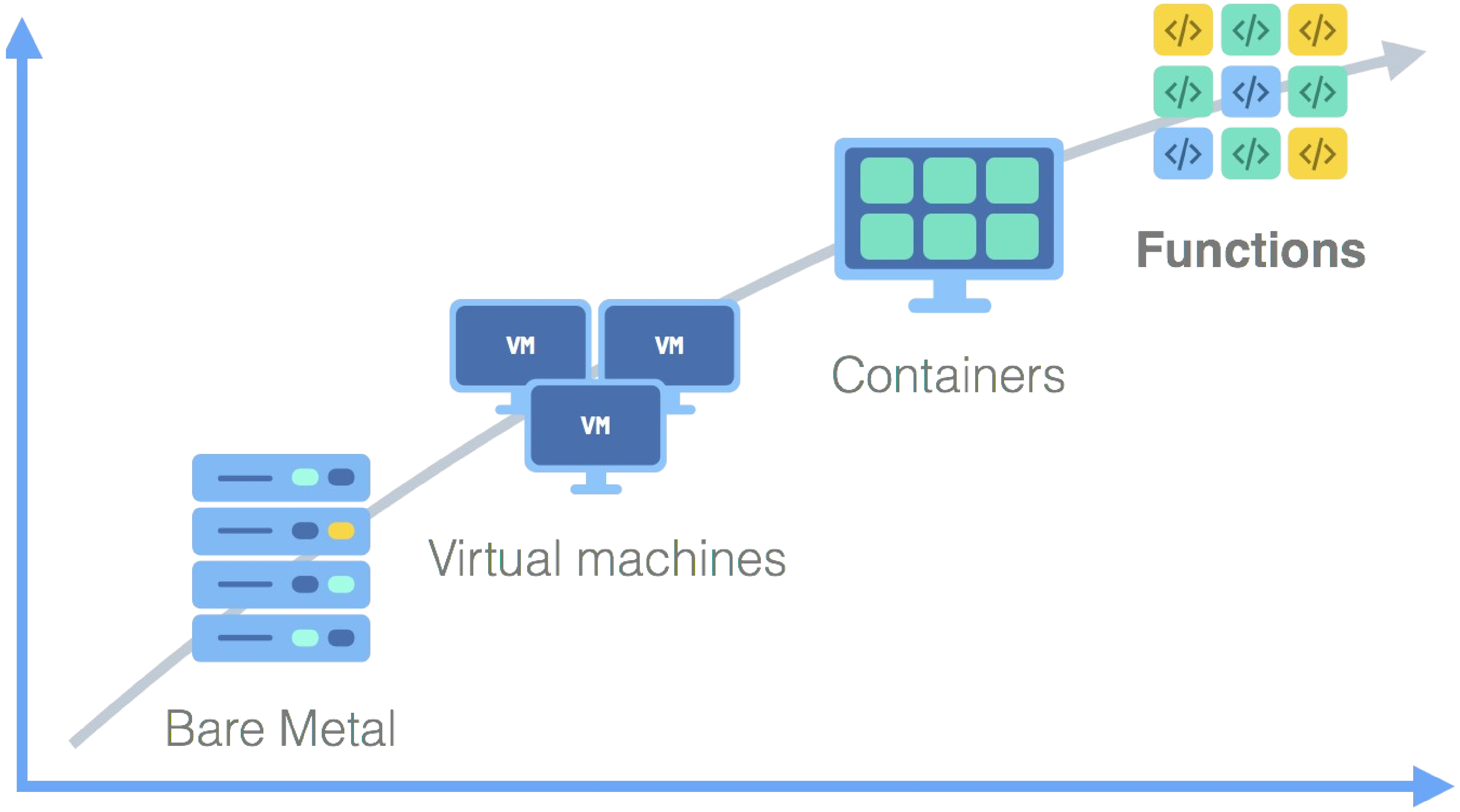
ELEMENTS OF CLOUD NATIVE COMPUTING?



BENEFITS OF CLOUD NATIVE COMPUTING

- Agility
- Speed
- Devops
- Efficient resource consumption
- On demand infrastructure
- Reusability
- Portability
- Scalability

Evolution of Servers



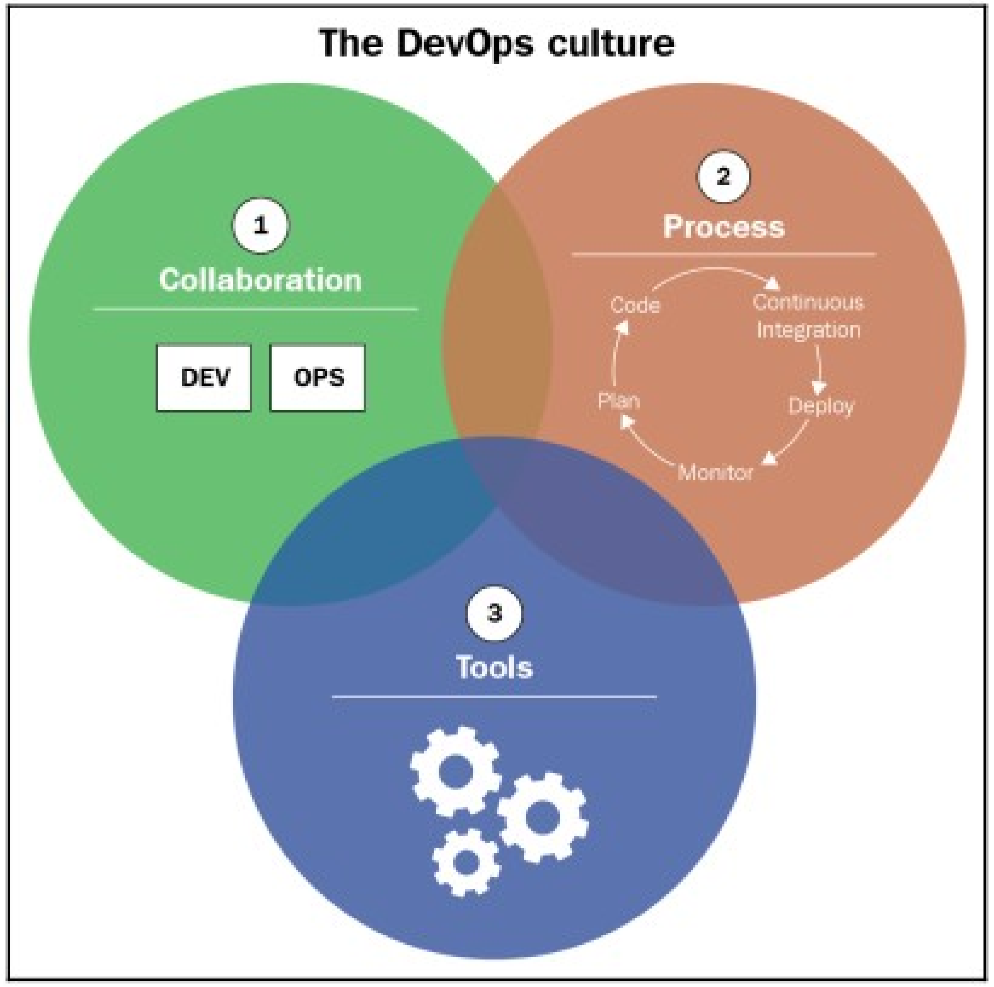
AGILITY

- Predictable cost and schedule
- Focuses on business value
- Focuses on end users
- Stakeholder engagement and early feedback
- Faster time to market and early predictable delivery
- Reduced risk

DEVOPS

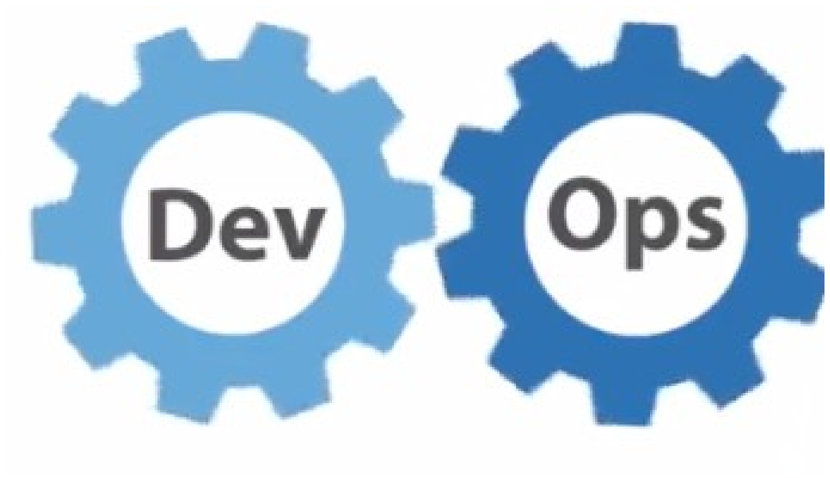
The DevOps movement is based on three axes:

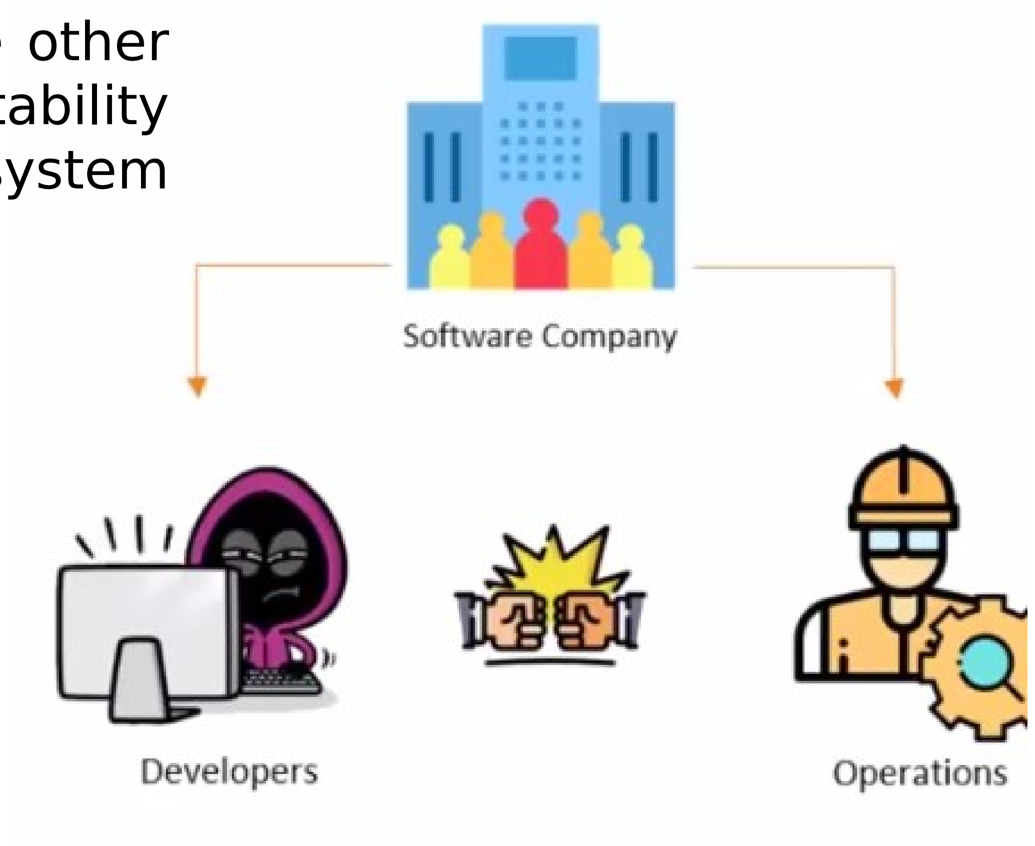
- The culture of collaboration
- Processes
- Tools



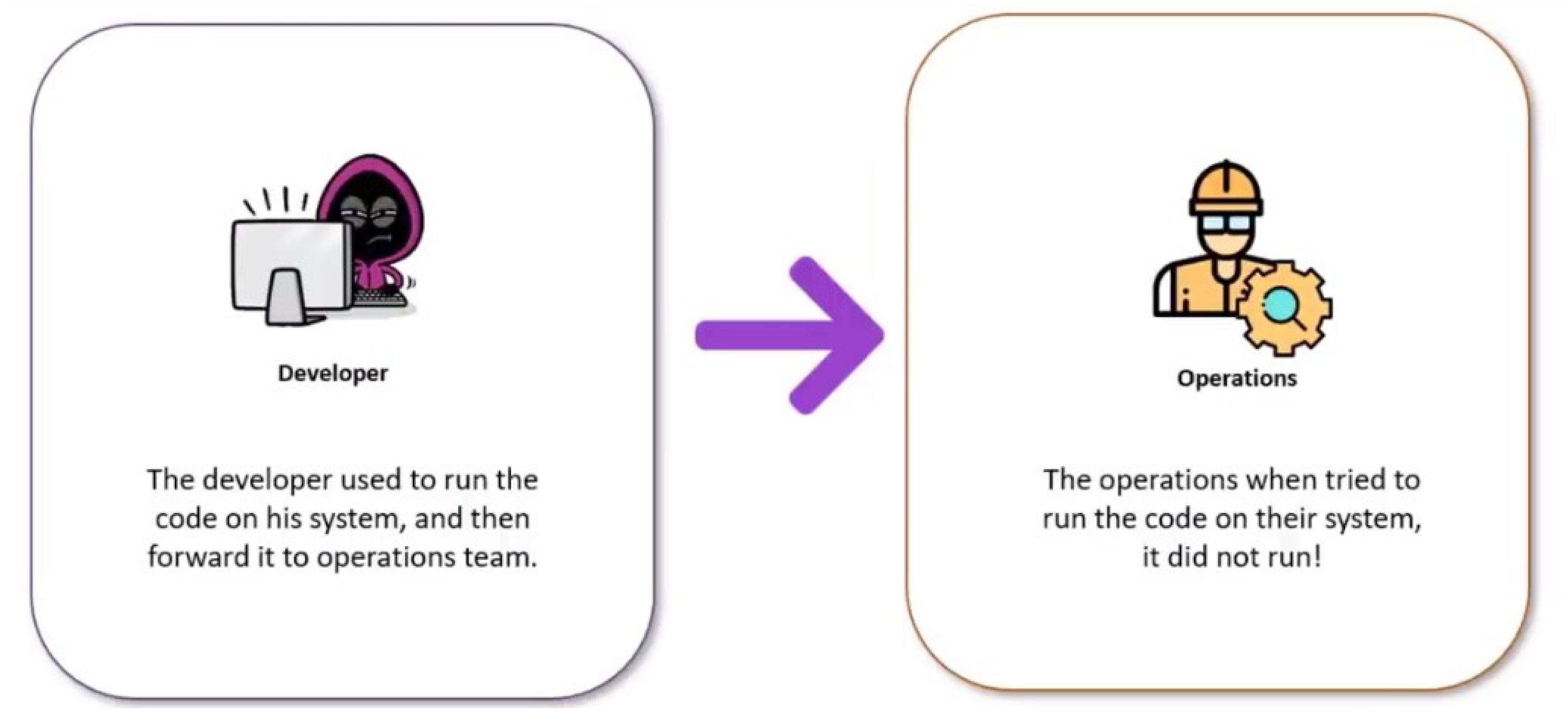
DEVOPS

DevOps culture is a set of practices that reduce the barriers between developers, who want to innovate and deliver faster, on the one side and, on the other side, operations, who want to guarantee the stability of production systems and the quality of the system changes they make.





WHY DEVOPS


Developer
The developer used to run the code on his system, and then forward it to operations team.
Operations
The operations when tried to run the code on their system, it did not run!

WHY DEVOPS

This led to a lot of back and forth between the developer and the operations team, hence impacted efficiency.

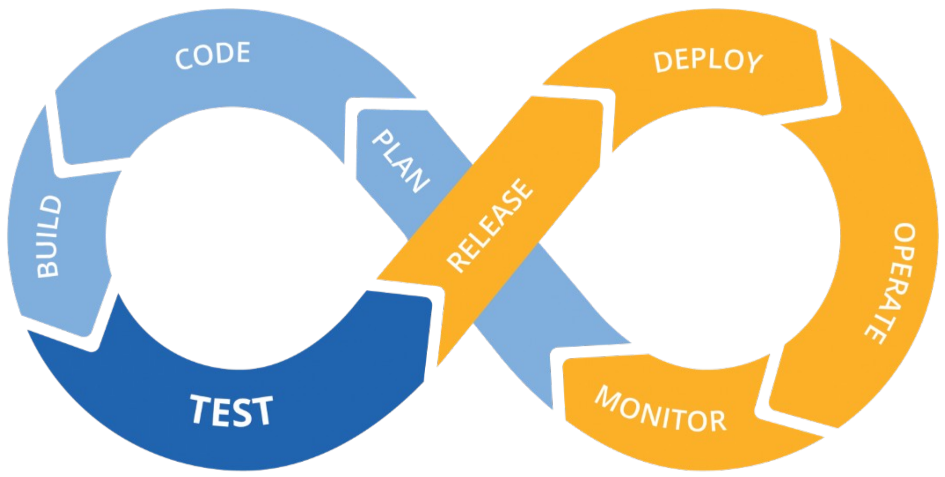
WHY DEVOPS

This problem was solved using Devops!

DEVOPS

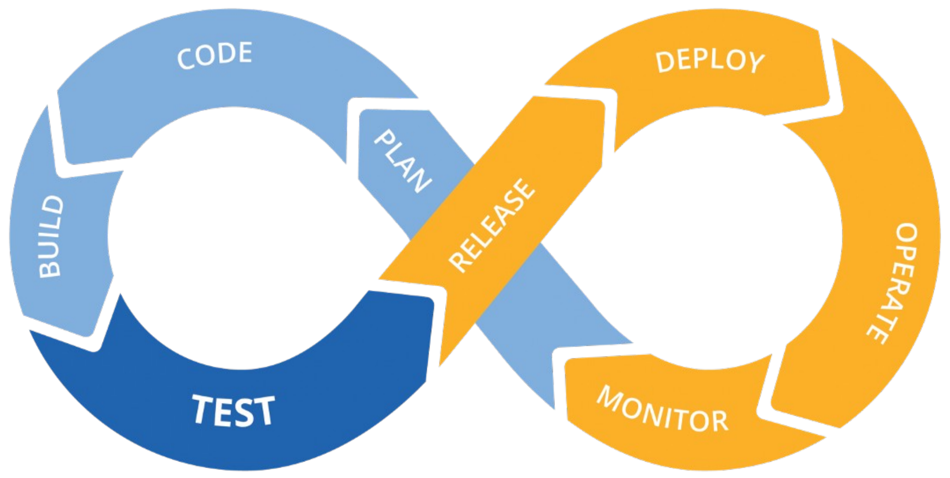
DevOps assimilates development and operations teams to improve the collaboration process.

A DevOps Engineer will work with IT developers to facilitate better coordination among operations, development, and testing functions by automating and streamlining the integration and deployment processes.



DEVOPS

DevOps culture is also the extension of agile processes (scrum, XP, and so on), which make it possible to reduce delivery times and already involve developers and business teams, but are often hindered because of the non-inclusion of Ops in the same teams.



1-CULTURE OF COLLABORATION

This is the very essence of DevOps—the fact that teams are no longer separated by specialization (one team of developers, one team of Ops, one team of testers, and so on), but, on the contrary, these people are brought together by making multidisciplinary teams that have the same objective: to deliver added value to the product as quickly as possible.

2-PROCESSES

To expect rapid deployment, these teams must follow development processes from agile methodologies with iterative phases that allow for better functionality quality and rapid feedback.

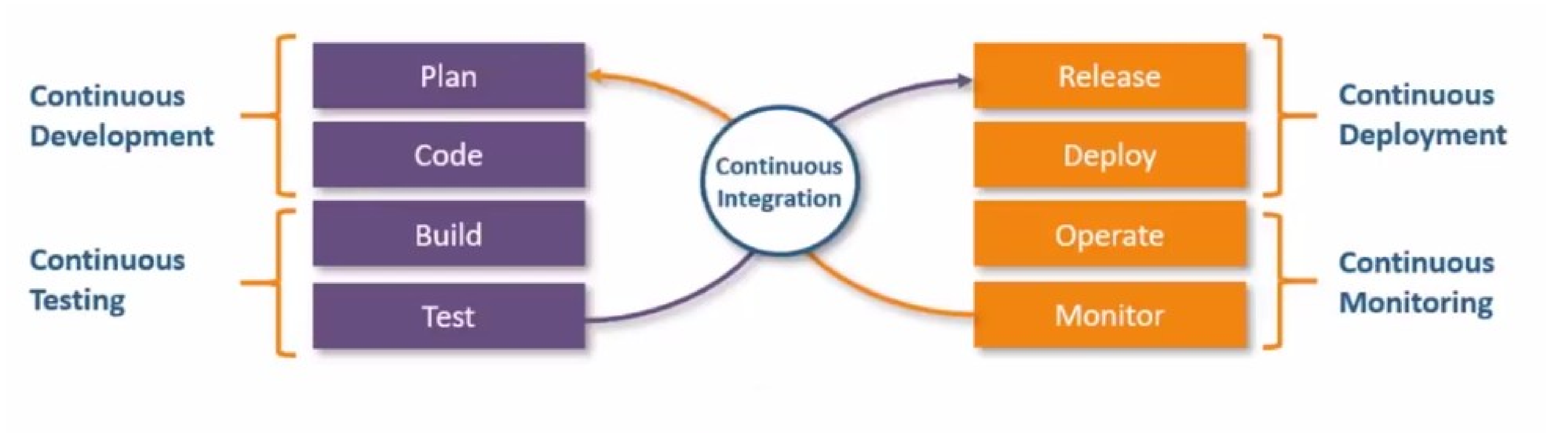
These processes should not only be integrated into the development workflow with continuous integration but also into the deployment workflow with continuous delivery and deployment.

The DevOps process is divided into several phases:

- The planning and prioritization of functionalities
- Development
- Continuous integration and delivery
- Continuous deployment
- Continuous monitoring

HOW DEVOPS WORK

The DevOps life cycle divides the SDLC into the following stages.



3-TOOLS

The choice of tools and products used by teams is very important in DevOps.

Indeed, when teams were separated into Dev and Ops, each team used their specific tools—deployment tools for developers and infrastructure tools for Ops—which further widened communication gaps.

SET OF TOOLS IN DEVOPS CULTURE

SET OF TOOLS IN DEVOPS CULTURE

GitHub

Docker

Kubernetes

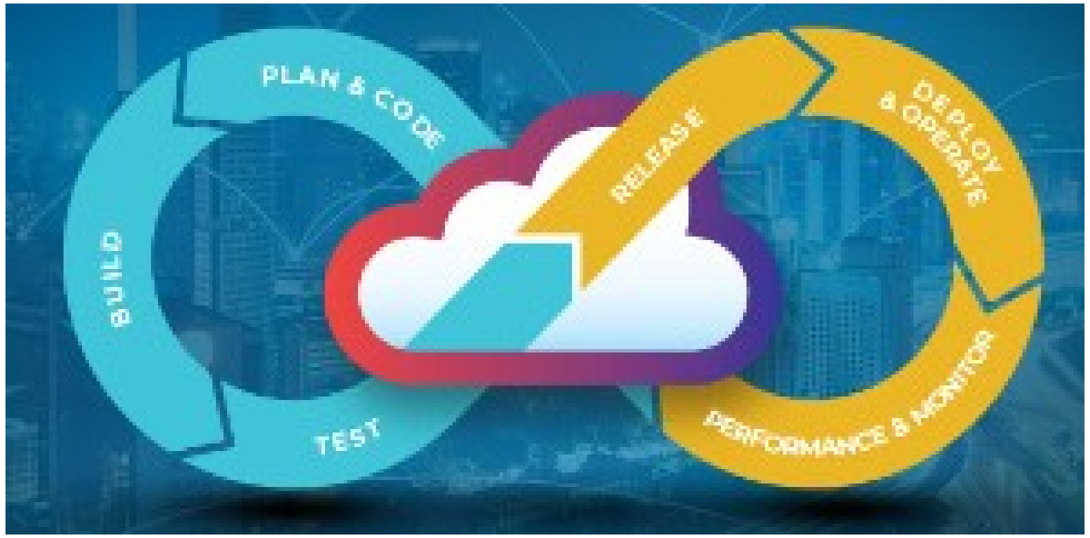
Jenkins

Terraform

WHAT IS CLOUD DEVOPS?

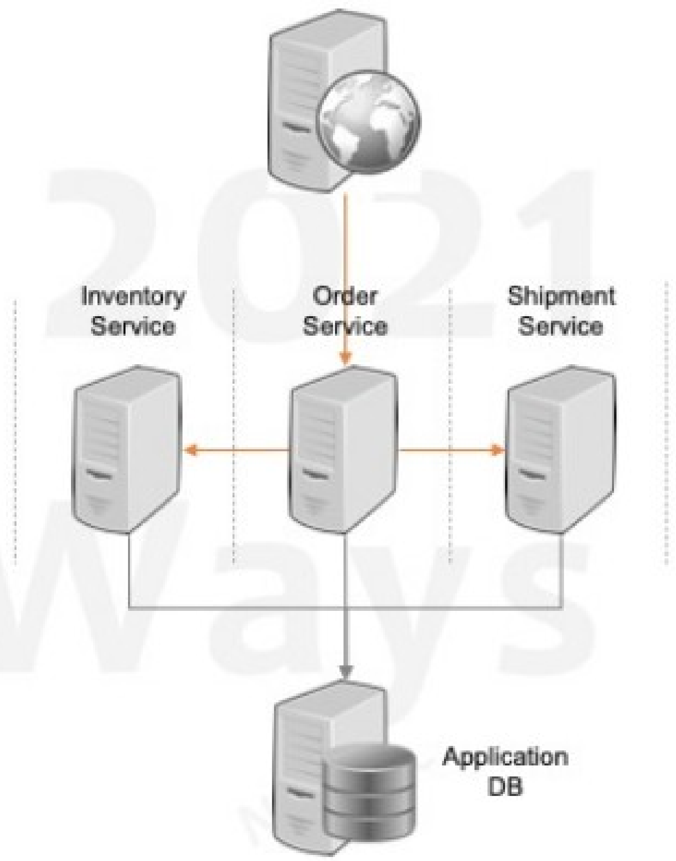
CLOUD DEVOPS

Merging Cloud + DevOps

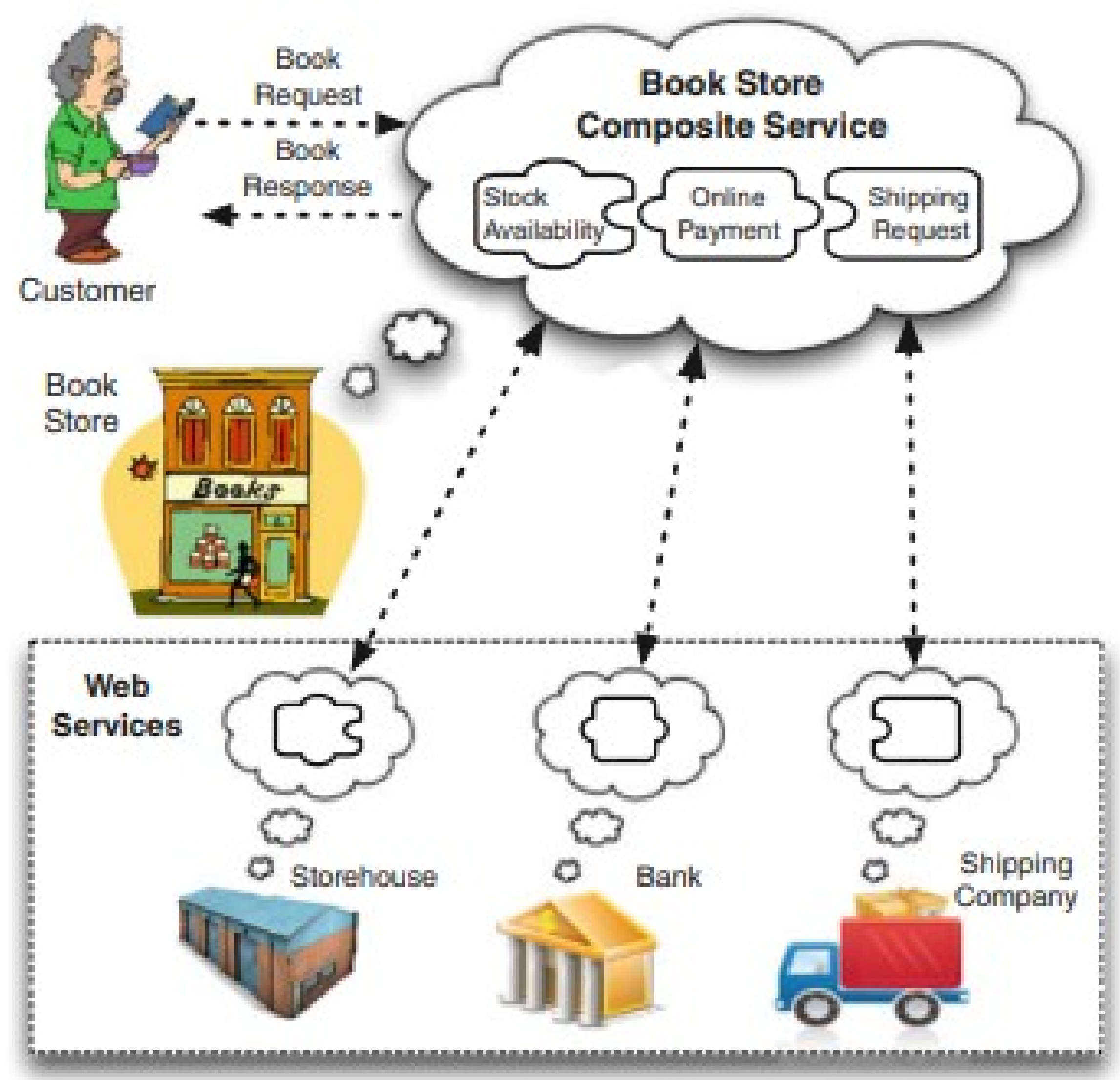


Service Oriented Architecture

- Independent
  - Each service can have its own technology stack, libraries, frameworks etc.
  - Each service can be scaled independently and differently
- Not Independent
  - Common interface schema
    - XML schema
  - Common database schema
    - RDBMS schema
- Issues
  - Service development may be independent but not deployment
  - Single database has scalability limitations



Service Oriented Architecture

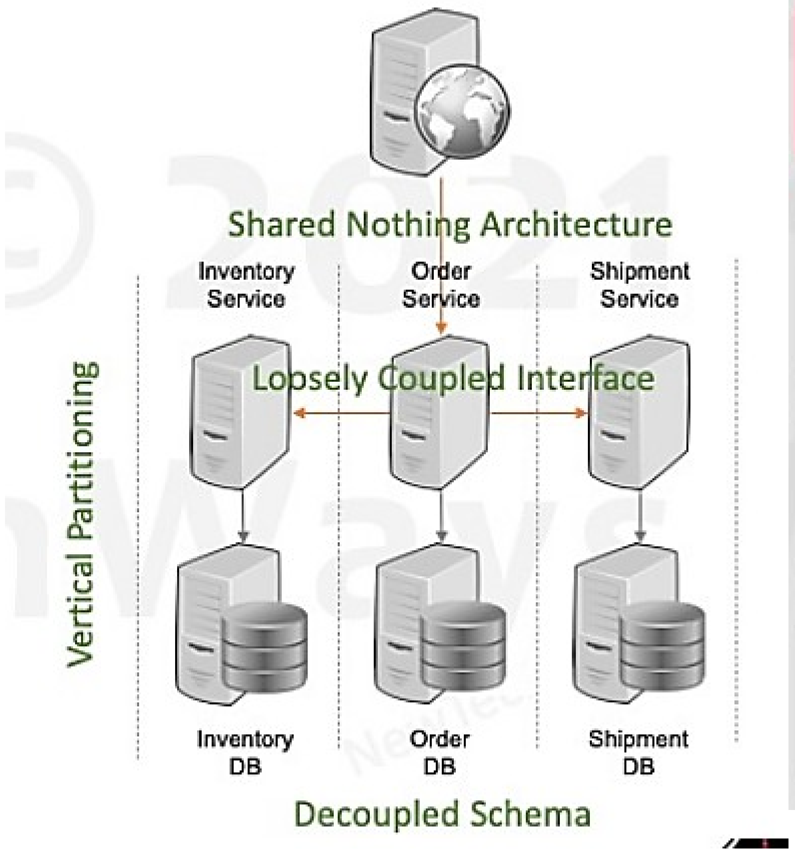


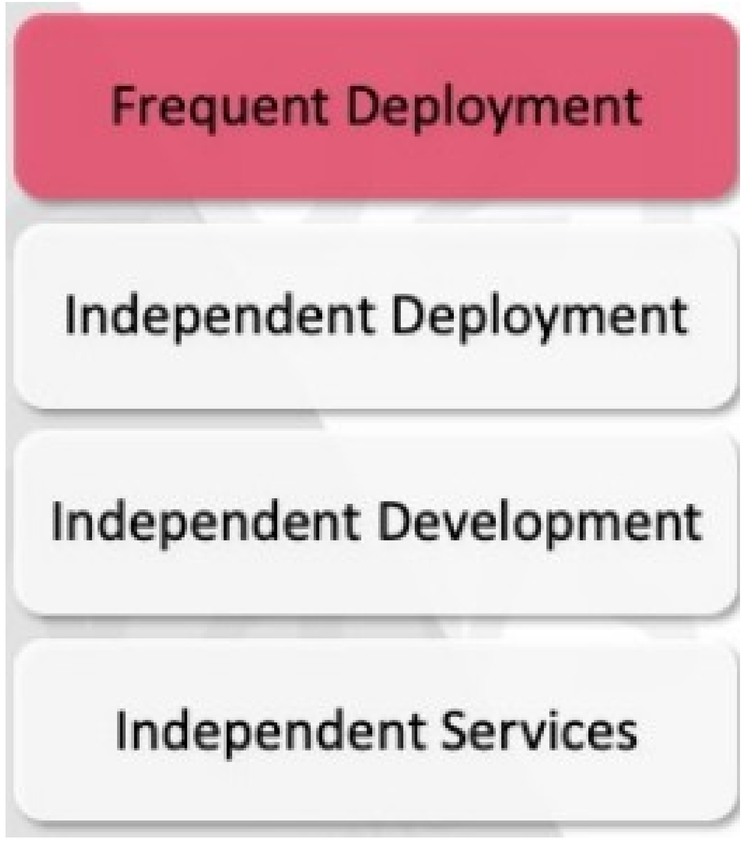
An example of Web service composition

MICROSERVICES

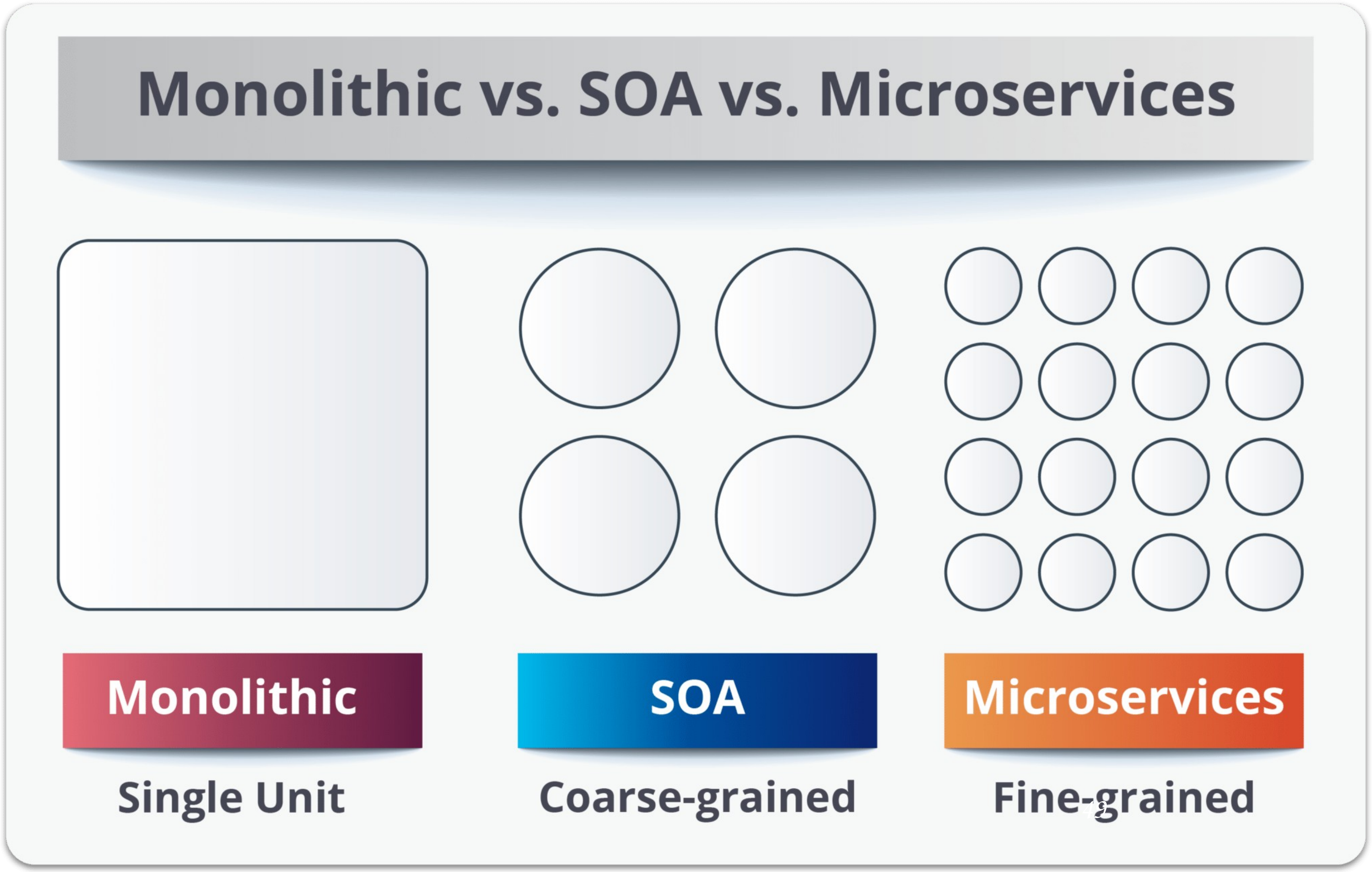
MICROSERVICES

- Microservices are small, individually deployable services performing different operations.
- Variant of SOA

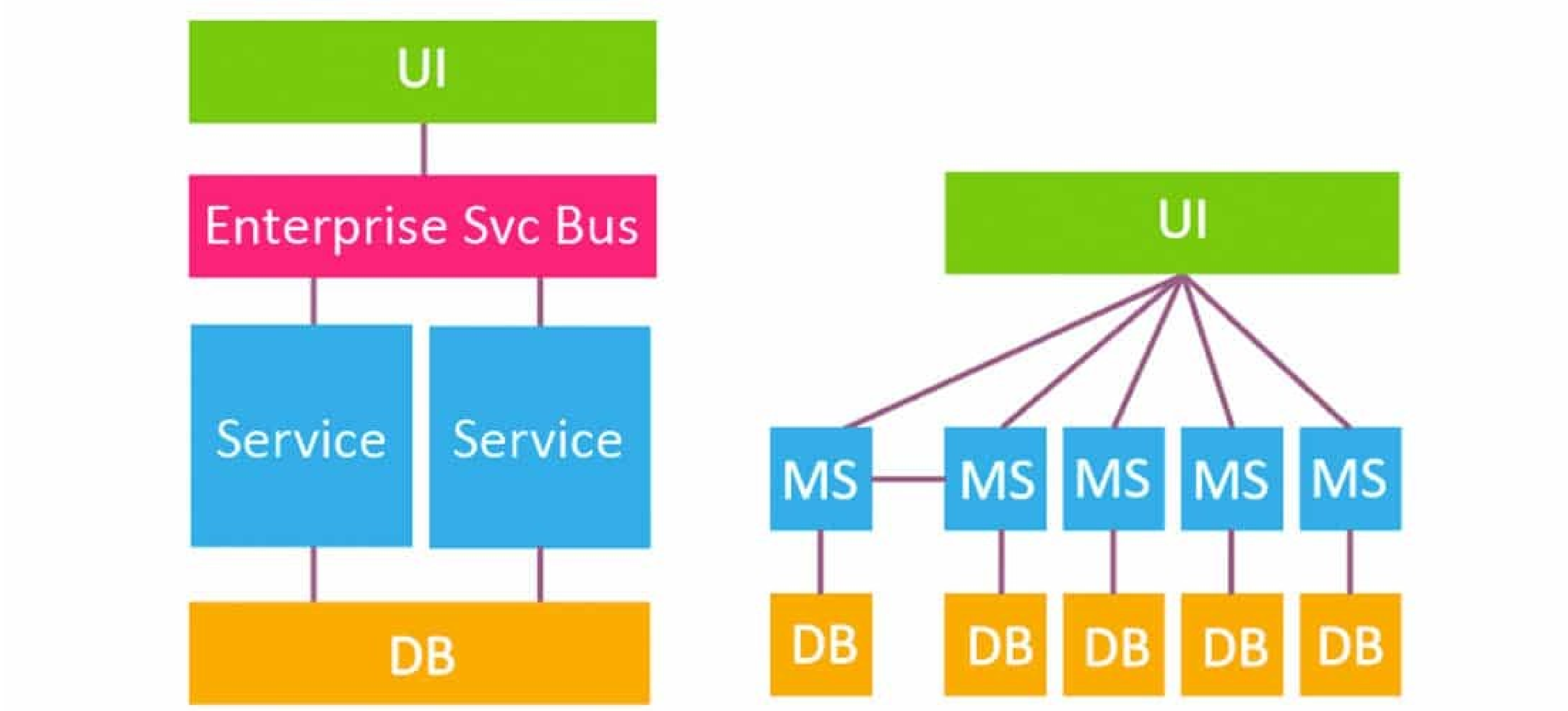




SOA VS MICROSERVICES



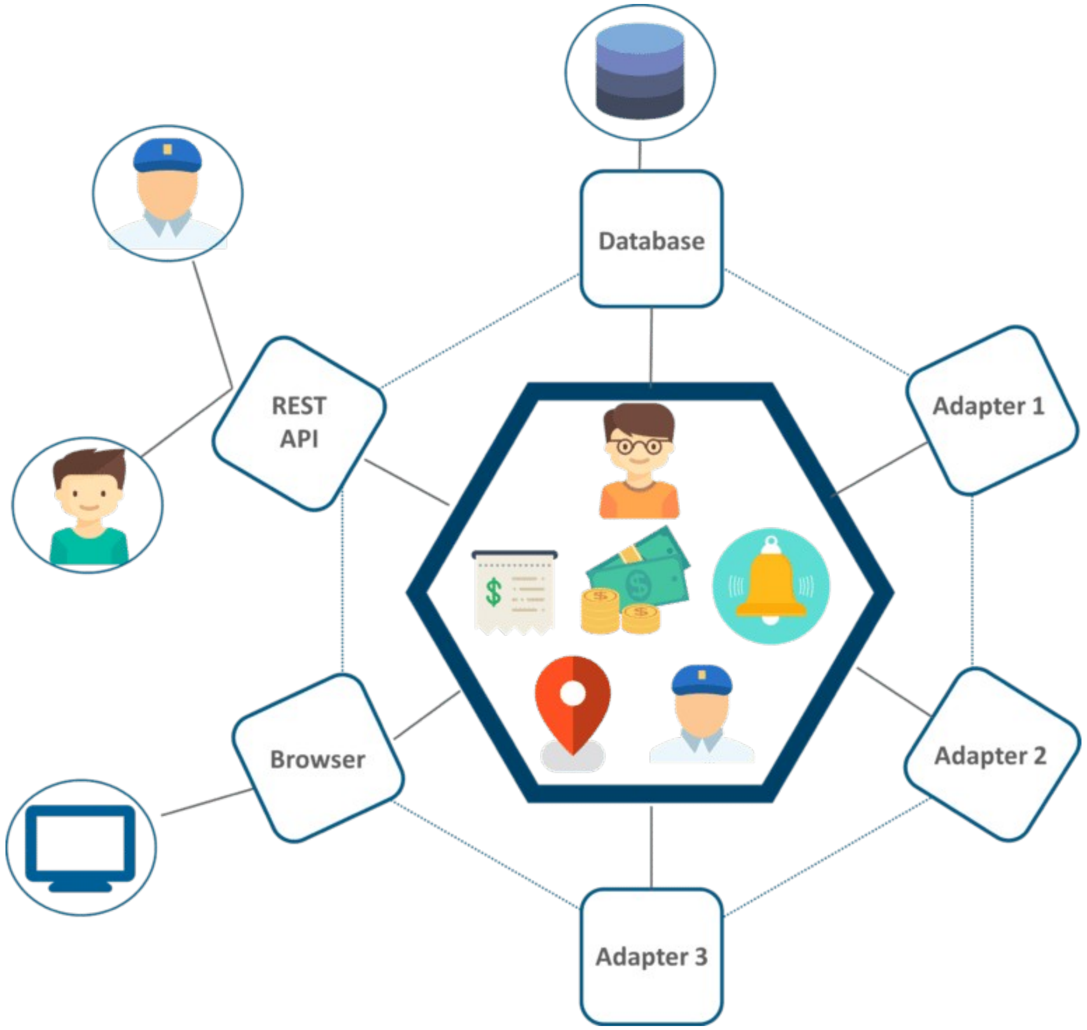
SOA VS MICROSERVICES



UBER'S PREVIOUS ARCHITECTURE

- A REST API is present with which the passenger and driver connect.
- Three different adapters are used with API within them, to perform actions such as billing, payments, sending emails/messages that we see when we book a cab.
- A MySQL database to store all their data.

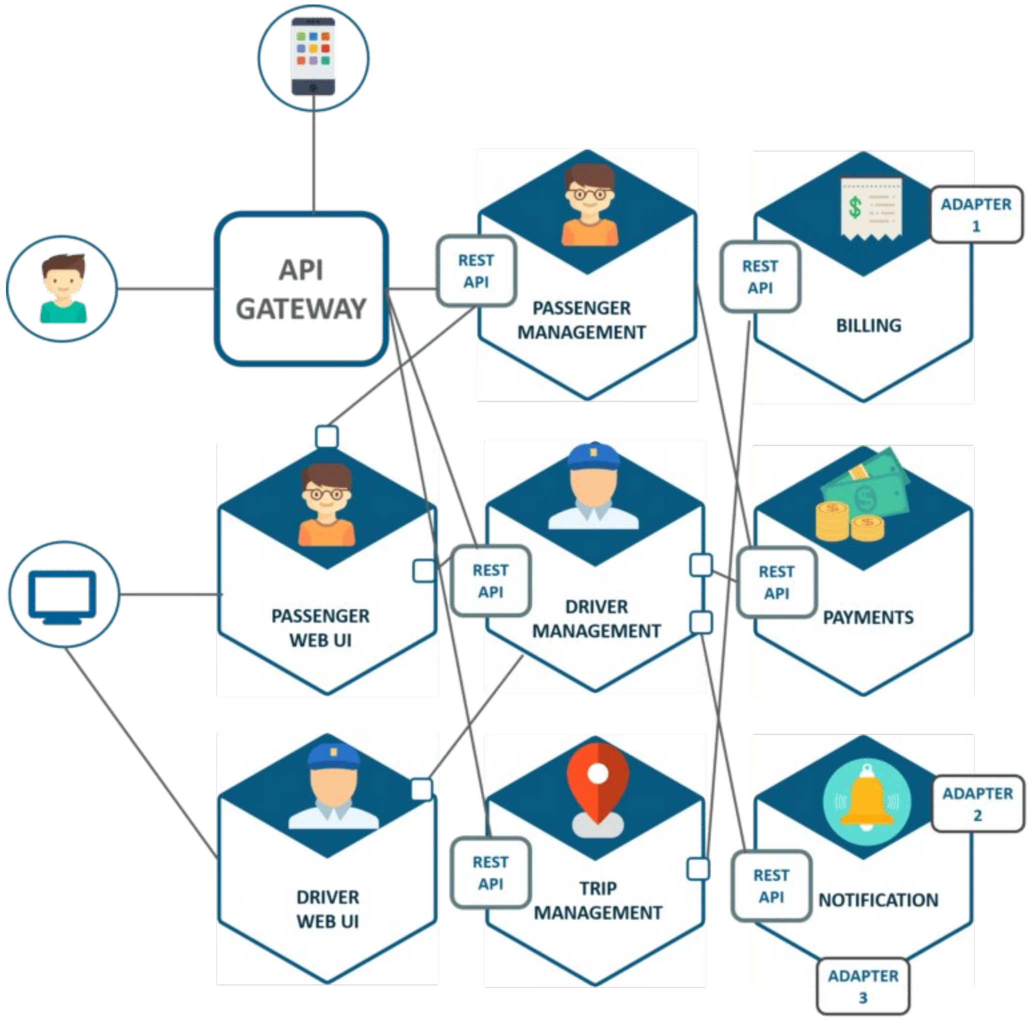
All the features such as passenger management, billing, notification features, payments, trip management, and driver management were composed within a single framework.



PROBLEM IN UBER’S ARCHITECTURE

- All the features had to be re-built, deployed and tested again and again to update a single feature.
- Fixing bugs became extremely difficult in a single repository as developers had to change the code again and again.
- Scaling the features simultaneously with the introduction of new features was quite tough to be handled together.

SOLUTION IS MICROSERVICES ARCHITECTURE



SOLUTION

- The units are individual separate deployable units performing separate functionalities.
- For Example: If you want to change anything in the billing Microservices, then you just have to deploy only billing Microservices and don’t have to deploy the others.
- All the features were now scaled individually i.e. The interdependency between each and every feature was removed.

DECOMPOSITION OF MICROSERVICES

There are some Prerequisite of decomposition of microservices.

Services must be cohesive.

- A service should implement a small set of strongly related functions.

Services must be loosely coupled

- Each service as an API that encapsulates its implementation.

MICROSERVICES WITH DOCKERS AND KUBERNETES

CHARACTERISTICS OF MICROSERVICES

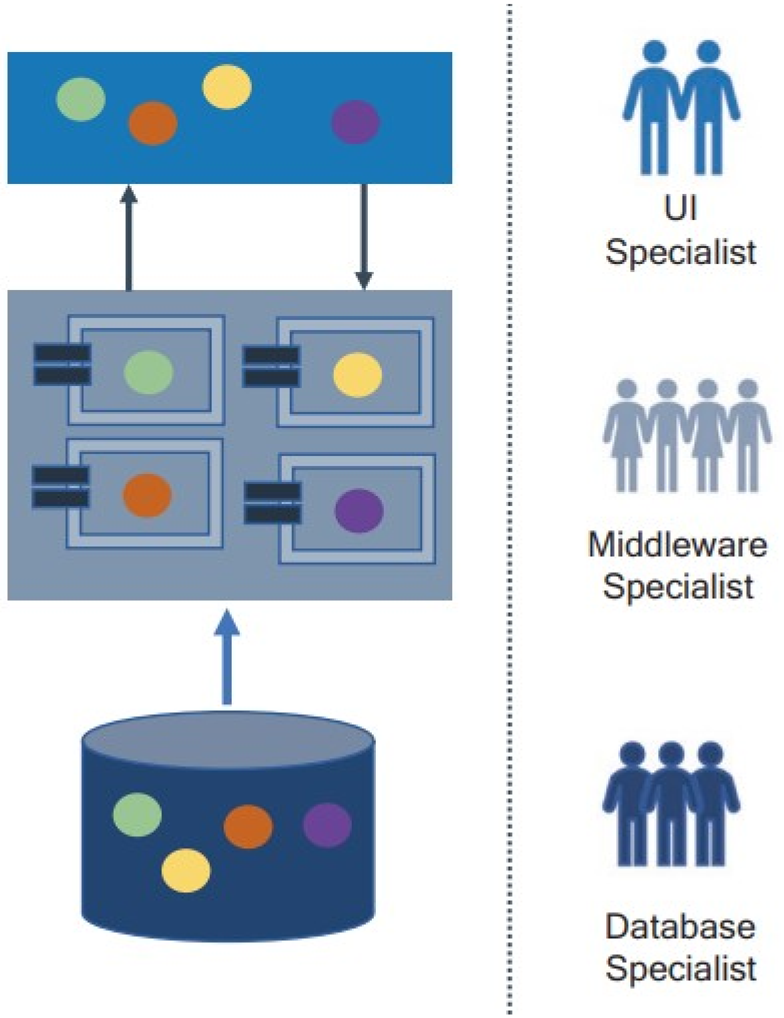
ORGANIZED AROUND BUSINESS CAPABILITIES

Before you start designing any microservices application, identifying and defining each microservice is important.

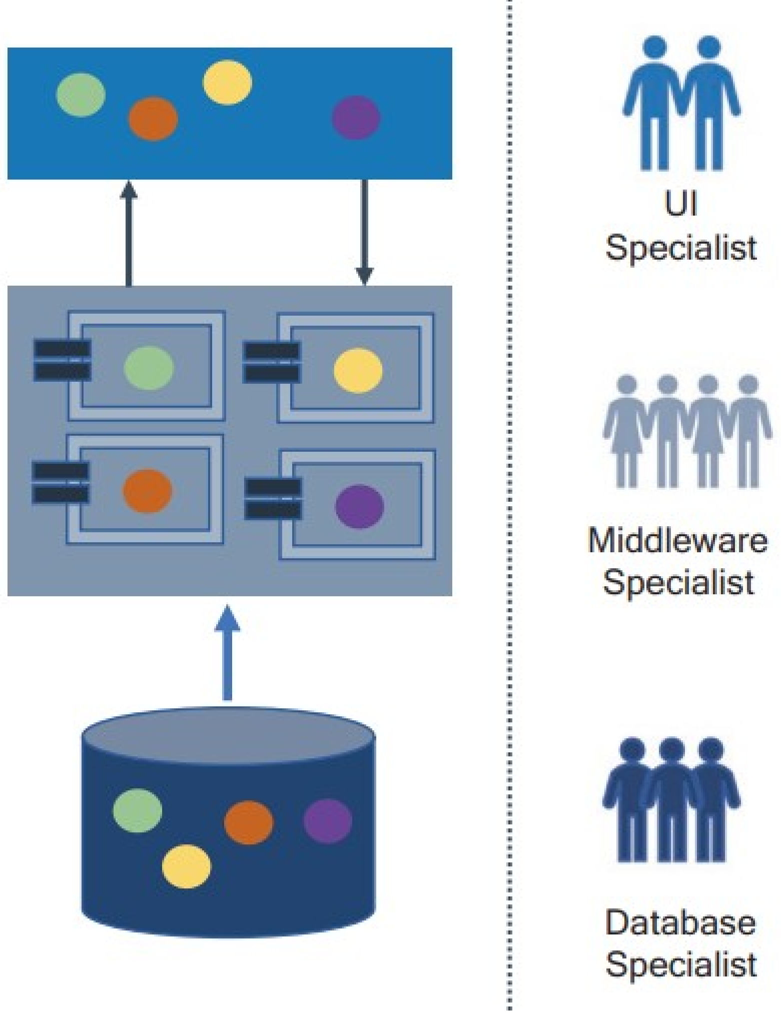
What is the boundary of the microservice?
What domain should it contain?
What are the events ?
Where will my microservices be deployed?

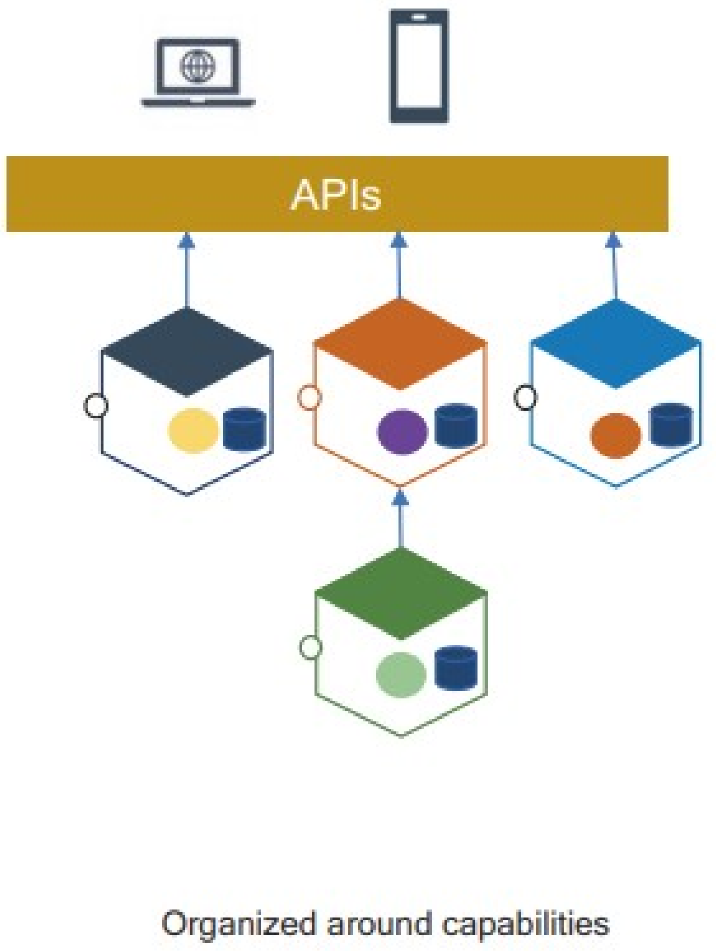
TRADITIONAL APPROACH
VS
ORGANIZED AROUND BUSINESS CAPABILITIES

In a traditional application development, as shown in Figure we designed around the *technological capabilities* such as the *user interface, databases, business logic,* etc., but we never had any discussion about the domain or contract or boundary.



TRADITIONAL APPROACH
VS
ORGANIZED AROUND BUSINESS CAPABILITIES





AUTONOMOUS

Microservices are a self-contained unit of functionality with loosely coupled dependencies across other services.

The services are separate entities and deployed in an isolated container environment.

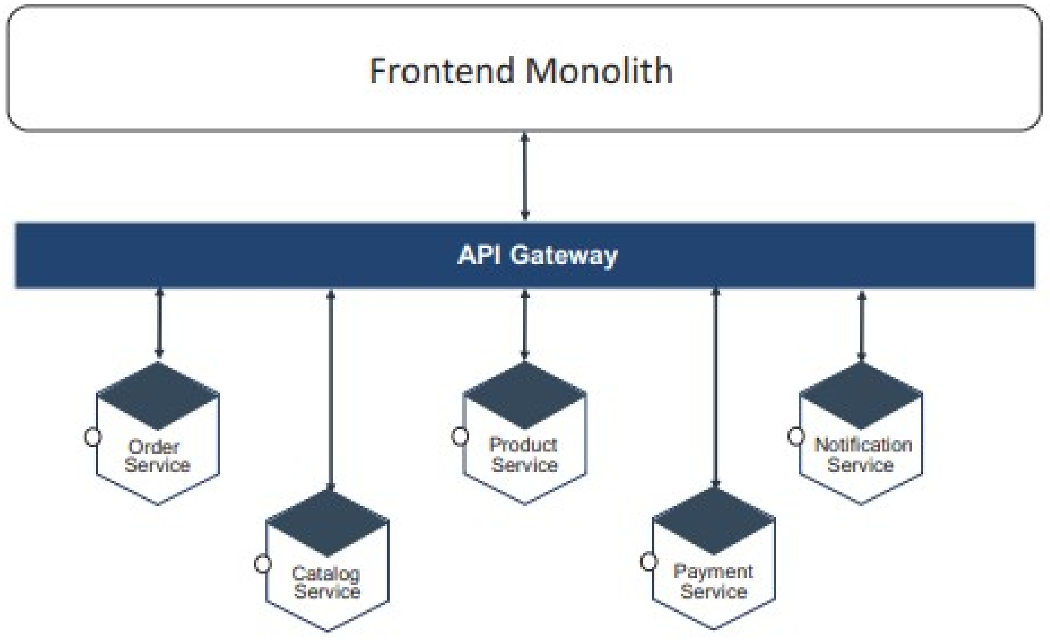
SMART END AND DUMB PIPES

The smart endpoints and dumb pipes simplify communication across microservices.

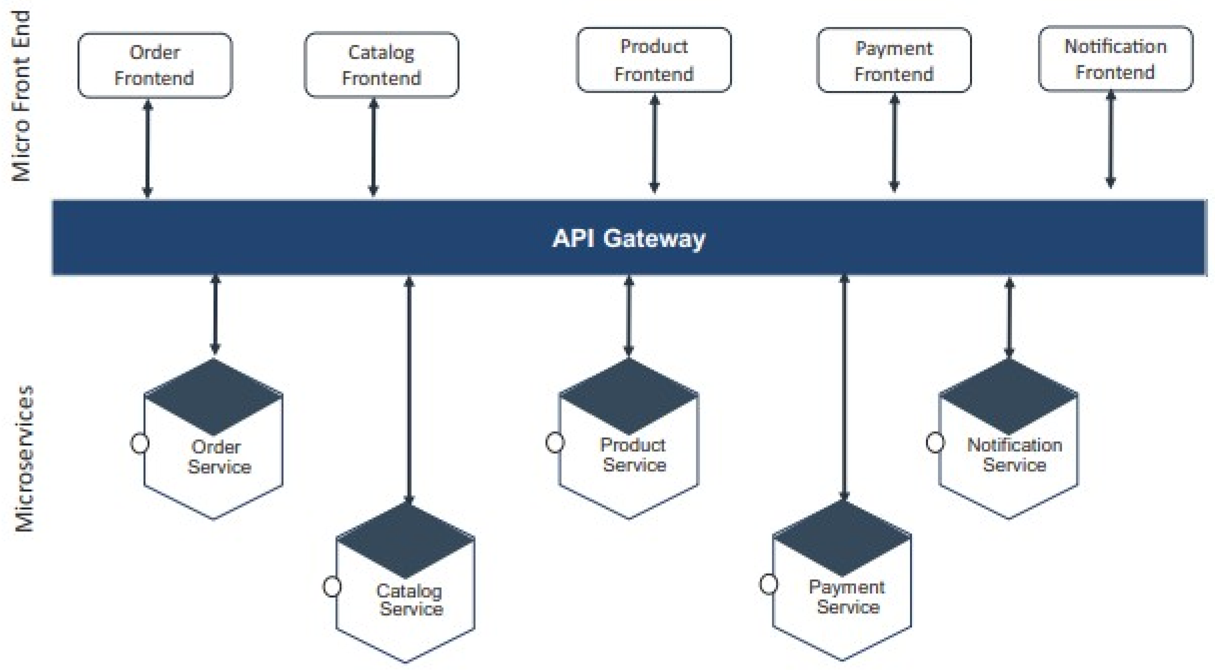
In microservices communication, we use two types of protocols: synchronous and asynchronous with request/response and publish/subscribe, respectively.

Endpoints are applications and the pipes are what connect them and allow them to communicate with each other.

MICROSERVICES AND USER INTERFACE: MICRO FRONT END



5-20. *Front-end monolithic with microservices*



HAVE A GOOD DAY!