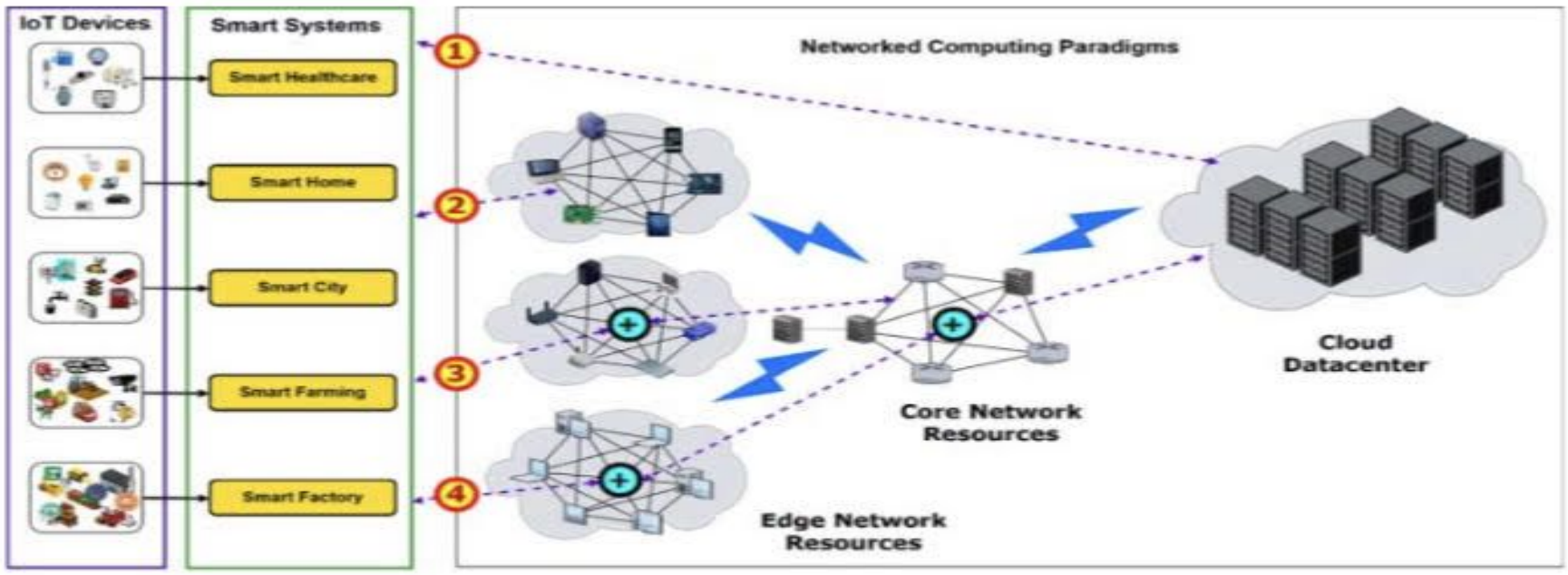
Chapter 2.Edge Computing Infrastructure

1. Edge computing architectures and components: Requirements and views for Edge architecture,
2. Edge Computing Reference Architecture, critical elements for Edge architecture, Challenges for Edge application Development.
3. Setting up Edge computing environments: development tools, python libraries.
4. Edge computing platforms and frameworks: AWS IoT Greengrass, Azure IoT Edge, Google Cloud IoT Edge, IBM Edge Application Manager, KubeEdge.
5. Virtualization and containerization for edge computing: Introduction to Virtualization.
6. Introduction to containerization
7. Advantages of Virtualization and Containerization in Edge Computing.
8. Resource Efficiency, Faster Time to Market.

9. Self-Learning Topics: Apache Edgent, Eclipse ioFog-Student discussion

Edge computing architectures and components: Requirements and views for Edge architecture

The notion of having computational resources near the data sources may seem not new. Particularly, the term—edge computing appeared in 2004 to illustrate a system that distributes program methods and the corresponding data to the network edge towards enhancing performance and efficiency .Similarly, the notion of having virtualization technology-based computing resources within the Wi-Fi subnet was introduced in 2009. However, the real industrial interest in extending computational resources to the edge network only started after the introduction of fog computing for IoT. Prior to that, applying utility cloud at the edge network was more or less a research topic in academia without explicit definition or architecture and with minor industrial involvement



Smart Systems enabled by different computing paradigms:
1 Cloud Computing, 2 Edge Computing, and 3 & 4 Fog Computing
+ - Indicates capability further enhanced by the next level resources in the network.

Figure 1.1 IoT applications and environments with supporting computing paradigms.

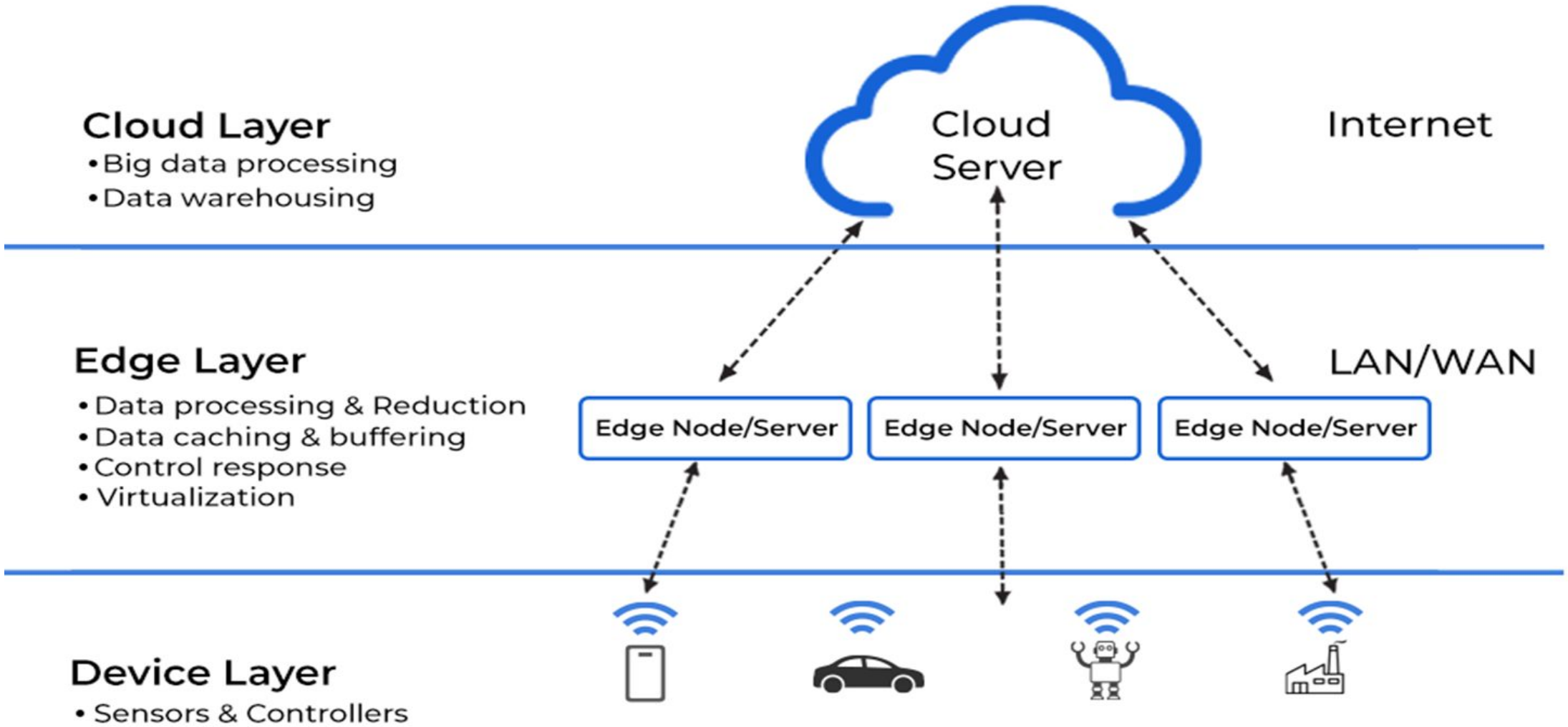
The generic edge computing paradigm distributes certain tasks to the IoT devices or the co-located computers within the same subnet of the IoT devices. Such tasks can be data classification , filtering, or signal converting

The decision of where the system should assign the tasks among the resources across different tiers depends on efficiency and adaptability.

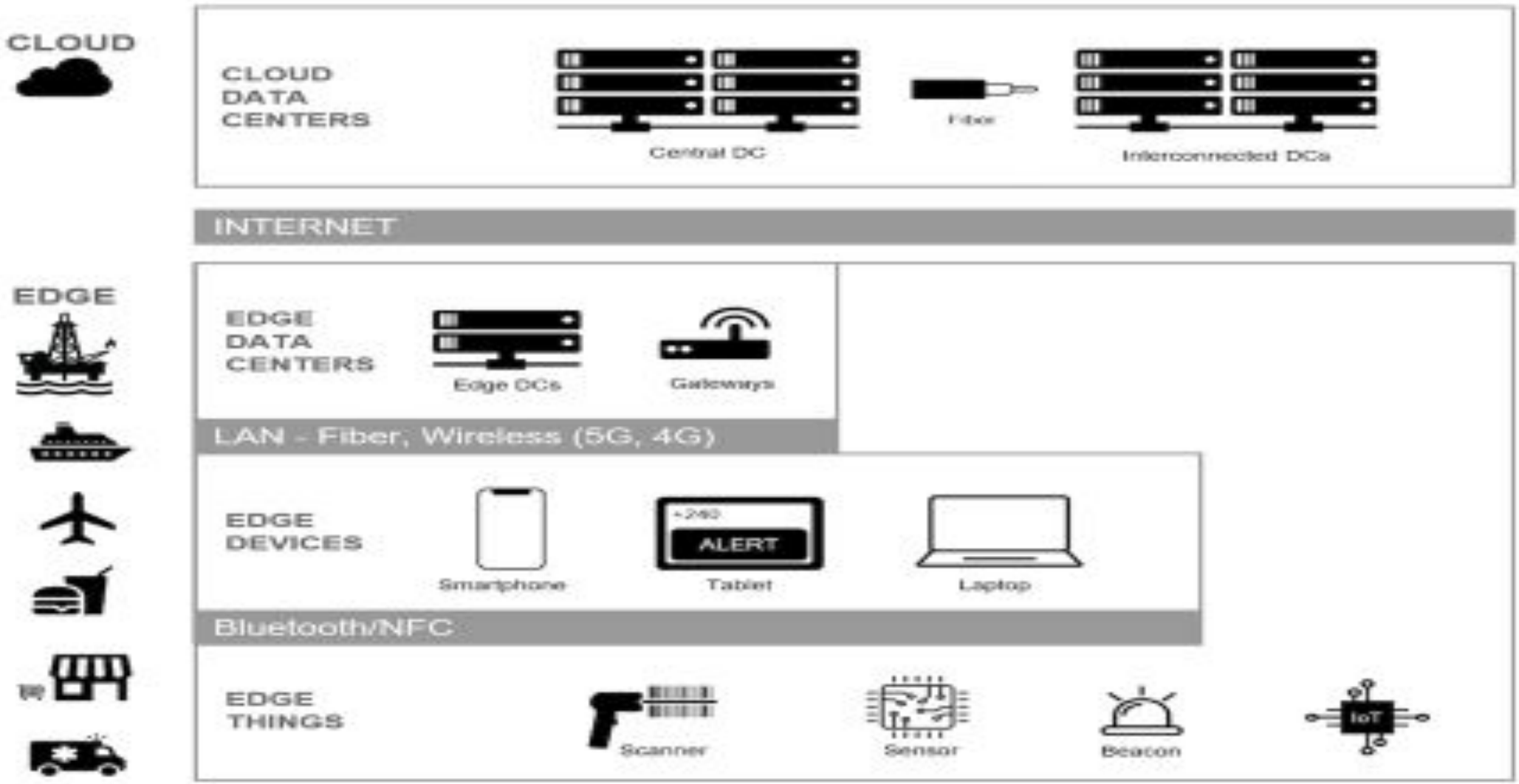
For example, smart systems may need to assign certain decision-making tasks to the edge devices in order to provide timely notification about the situation, such as the patient's condition in the smart healthcare, the security state of the smart home, the traffic condition of the smart city, the water supply condition of smart farming, or the production line operation condition of a smart factory

TOOLBOX™

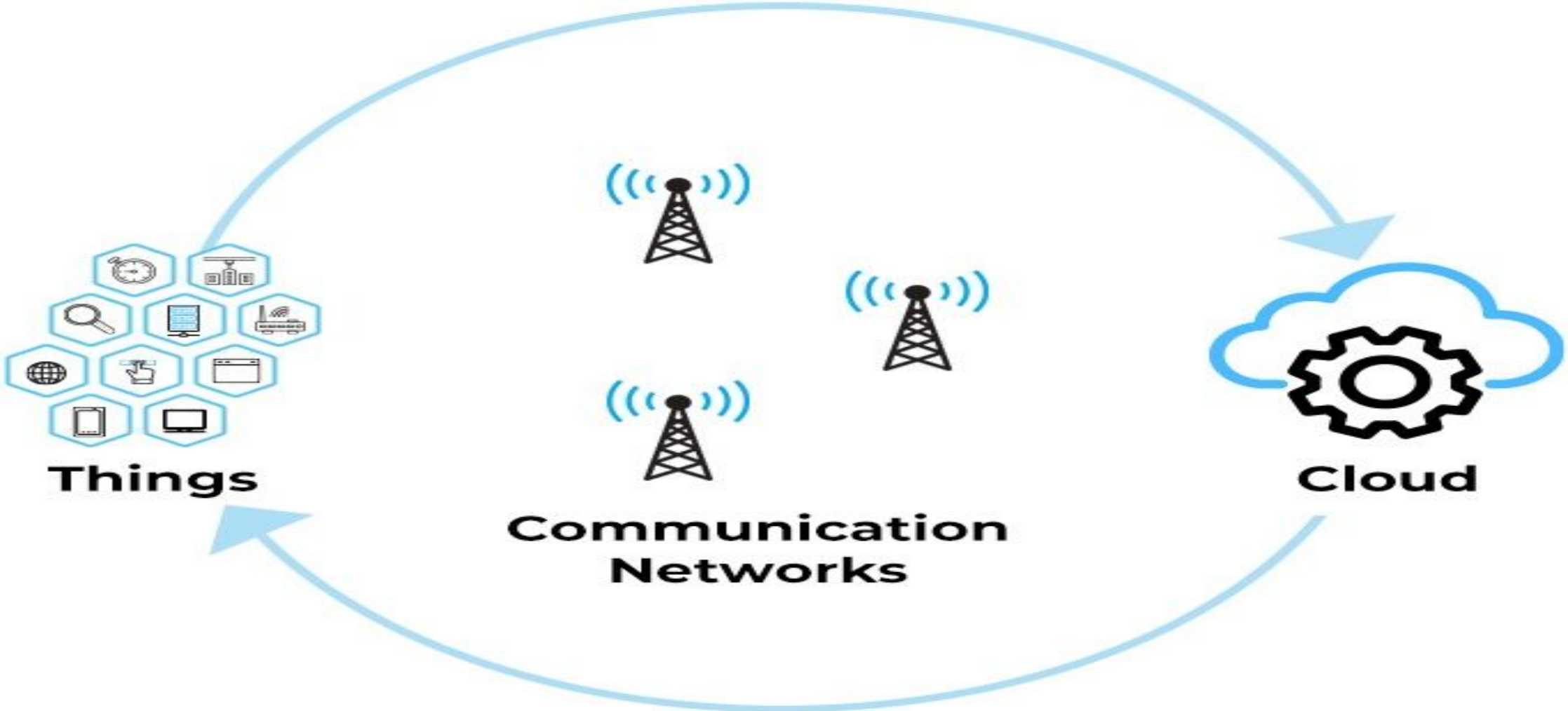
EDGE COMPUTING ARCHITECTURE



Edge computing architectures and components: Requirements and views for Edge architecture



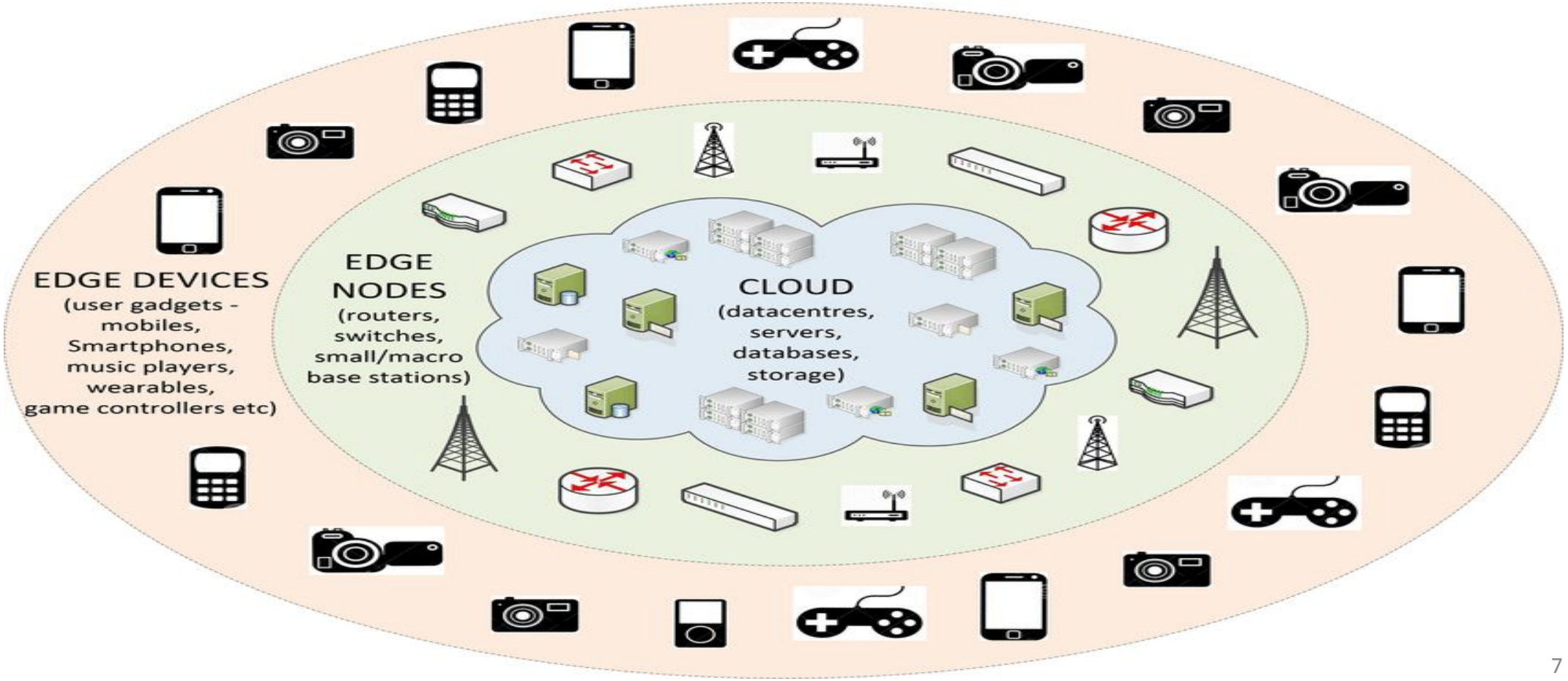
COMPONENTS OF EDGE COMPUTING



Fundamentally, in order to interconnect the physical entities to the Internet, the system will utilize various **front-end devices such as wired or wireless sensors, actuators, and readers** to interact with them. Further, the **front-end devices have the Internet connectivity via the mediate gateway nodes such as Internet modems, routers, switches, cellular base stations**

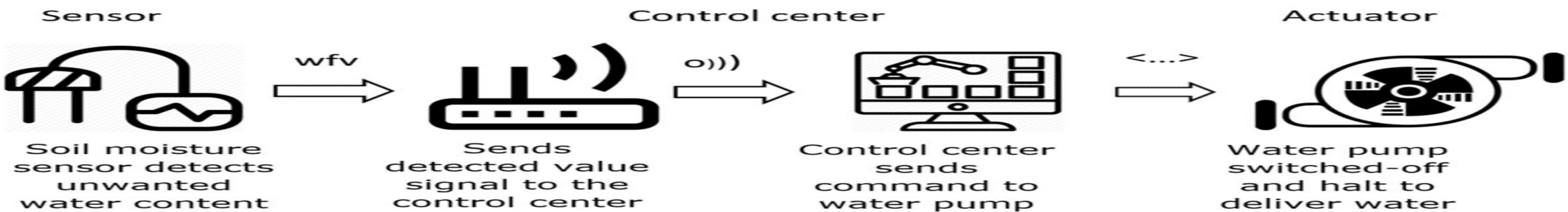
Edge computing relies on Edge devices, servers, routers, switches, and nodes

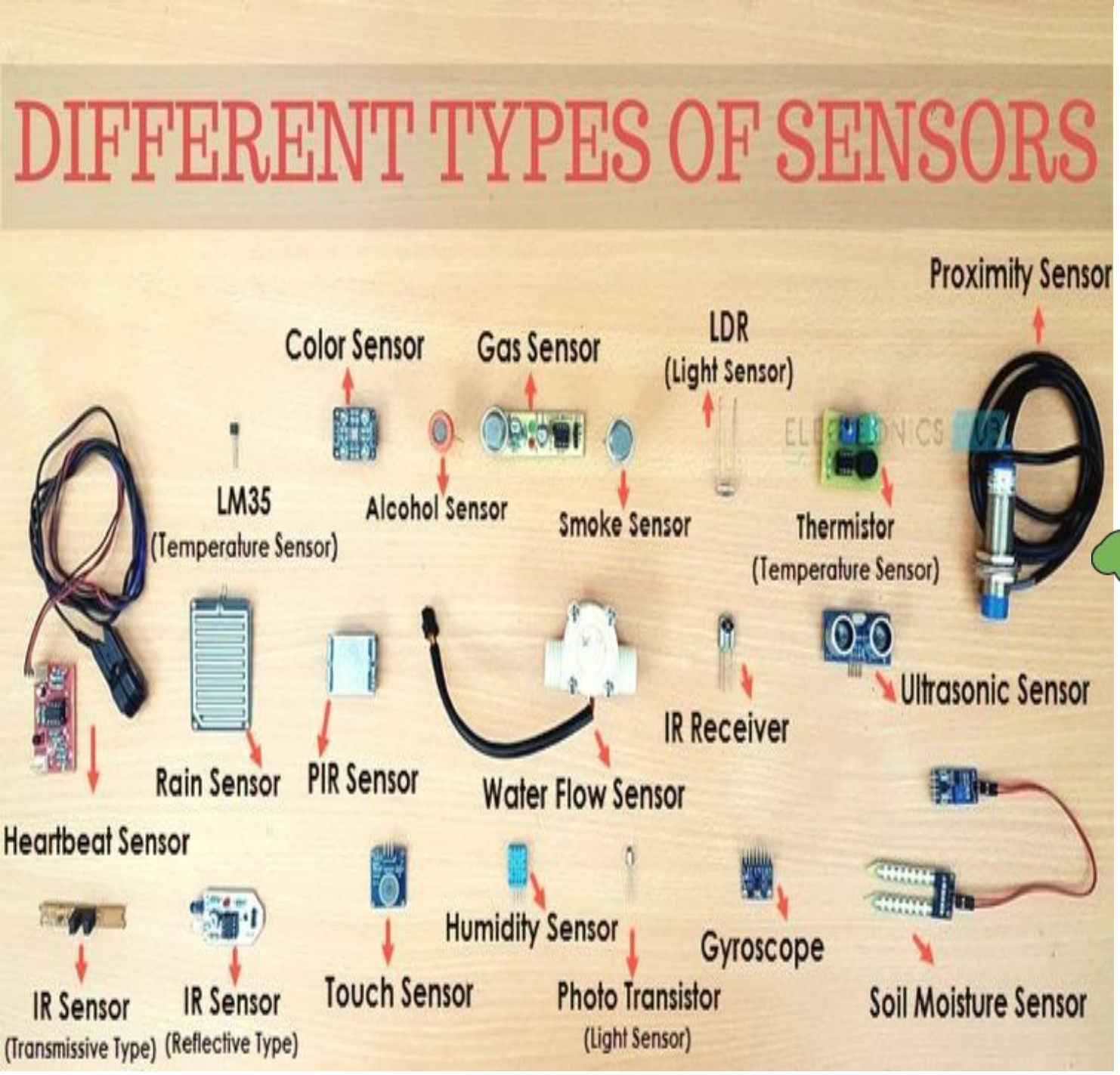
An edge device is any piece of hardware that controls data flow at the boundary between two networks. Edge devices fulfill a variety of roles, depending on what type of device they are, but they essentially serve as network entry -- or exit -- points.
An edge device is a device that provides an entry point into enterprise or service provider core networks. Examples include routers, routing switches, integrated access devices (IADs), multiplexers, and a variety of metropolitan area network (MAN) and wide area network (WAN) access devices.

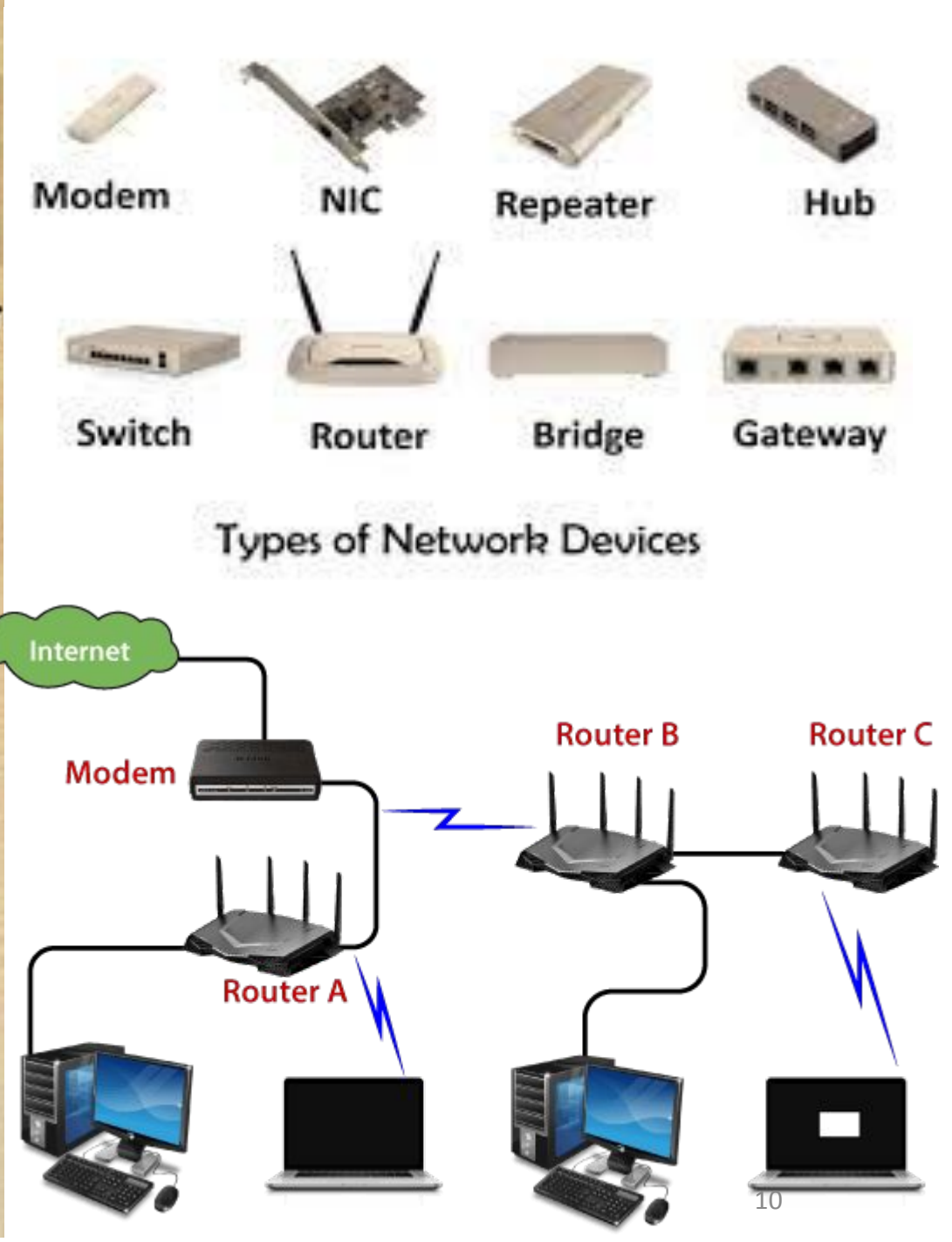


https://www.google.com/imgres?imgurl=https%3A%2F%2Fwww.altoros.com%2Fblog%2Fwp-content%2Fuploads%2F2022%2F04%2FKubeEdge-Kubernetes-Edge-Computing-Cloud-native-v2.gif&tbnid=5jf3dCyHdzxL-M&vet=12ahUKEwiO04Pkz6mAAxWrzaACHU6pDZgQMygZegUIARCfAg..i&imgrefurl=https%3A%2F%2Fwww.altoros.com%2Fblog%2Fkubeedge-monitoring-edge-devices-at-the-worlds-longest-sea-bridge%2F&docid=InbGwo7IosLJvM&w=1280&h=720&q=edge%20devices&ved=2ahUKEwiO04Pkz6mAAxWrzaACHU6pDZgQMygZegUIARCfAghttps://www.google.com/imgres?imgurl=https%3A%2F%2Fwww.altoros.com%2Fblog%2Fwp-content%2Fuploads%2F2022%2F04%2FKubeEdge-Kubernetes-Edge-Computing-Cloud-native-v2.gif&tbnid=5jf3dCyHdzxL-M&vet=12ahUKEwiO04Pkz6mAAxWrzaACHU6pDZgQMygZegUIARCfAg..i&imgrefurl=https%3A%2F%2Fwww.altoros.com%2Fblog%2Fkubeedge-monitoring-edge-devices-at-the-worlds-longest-sea-bridge%2F&docid=InbGwo7IosLJvM&w=1280&h=720&q=edge%20devices&ved=2ahUKEwiO04Pkz6mAAxWrzaACHU6pDZgQMygZegUIARCfAg

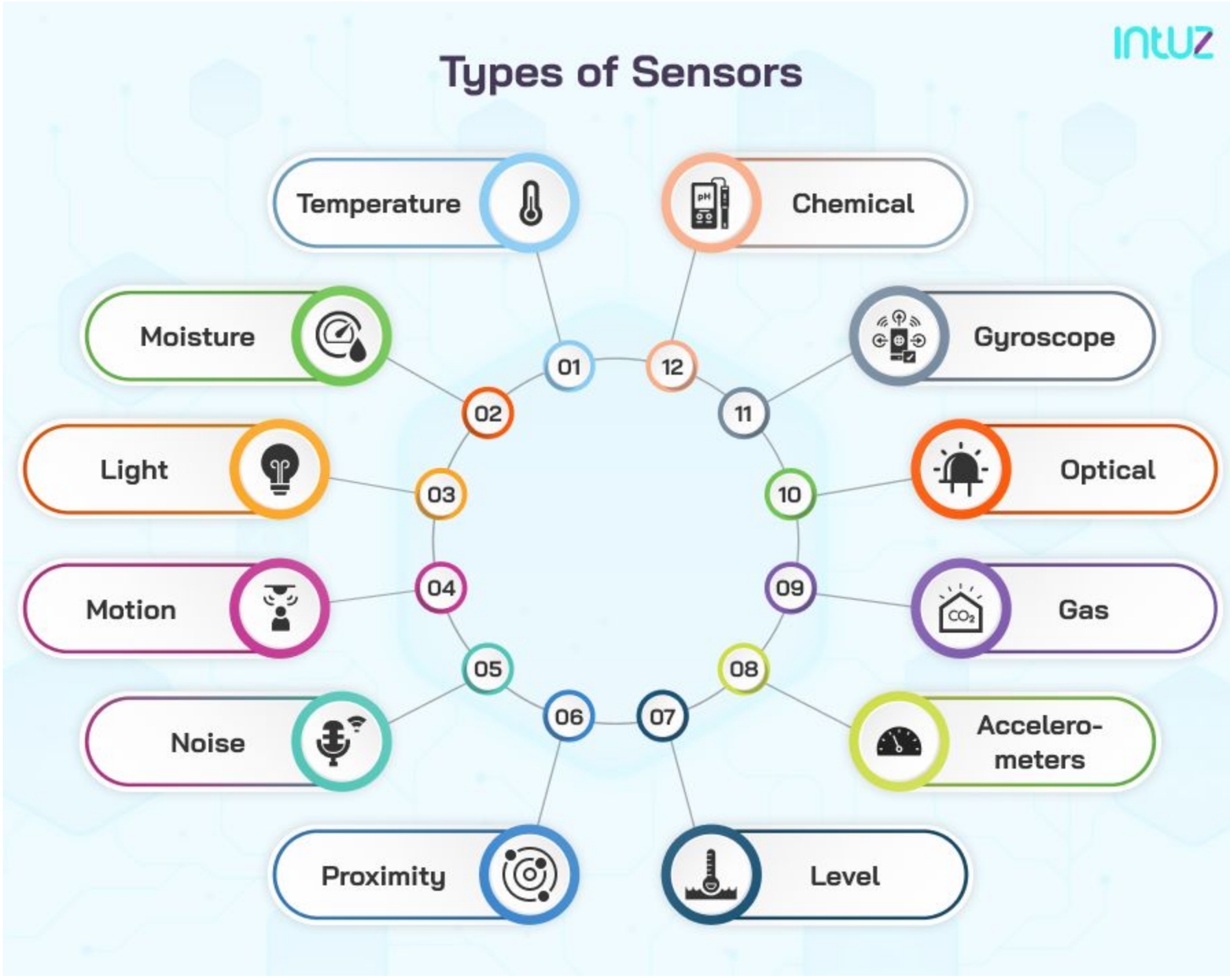
Sensor to **Actuator** Flow

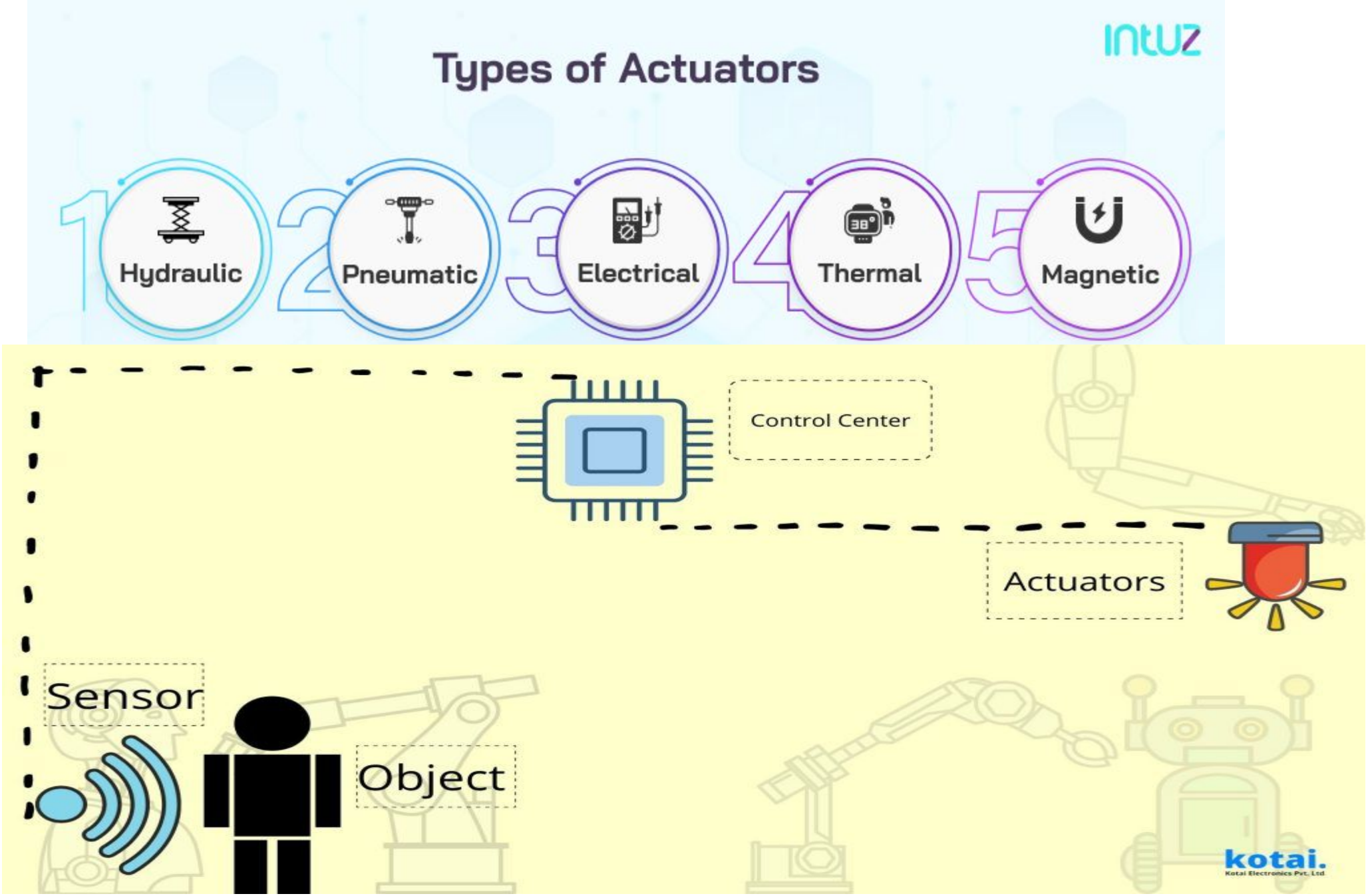



DIFFERENT TYPES OF SENSORS
Proximity Sensor
Color Sensor
Gas Sensor
LDR
(Light Sensor)
LM35
(Temperature Sensor)
Alcohol Sensor
Smoke Sensor
Thermistor
(Temperature Sensor)
Ultrasonic Sensor
IR Receiver
Rain Sensor
PIR Sensor
Water Flow Sensor
Heartbeat Sensor
Humidity Sensor
Gyroscope
IR Sensor
(Transmissive Type)
IR Sensor
(Reflective Type)
Touch Sensor
Photo Transistor
(Light Sensor)
Soil Moisture Sensor

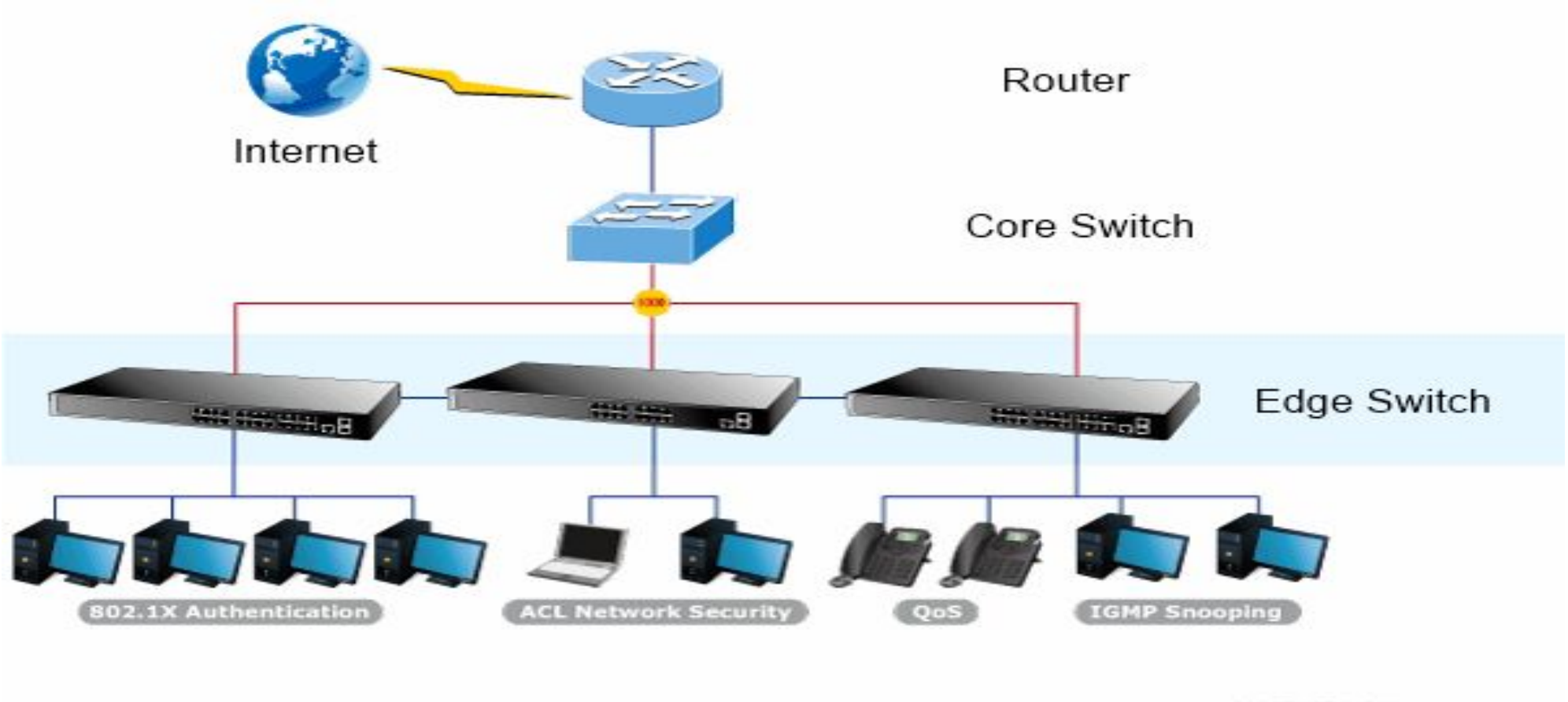

Modem
NIC
Repeater
Hub
Switch
Router
Bridge
Gateway
Types of Network Devices
Internet
Modem
Router A
Router B
Router C

Types of Sensors




Types of Actuators
1
Hydraulic
2
Pneumatic
3
Electrical
4
Thermal
5
Magnetic
Control Center
Actuators
Sensor
Object

As the name indicates, an edge switch is a switch located at the meeting point of two networks. These switches connect end-user local area networks (LANs) to Internet service provider (ISP) networks.



https://www.ibm.com/cloud/architecture/architectures/edge-computing/reference-architecture/

Opportunities and Challenges

Additional opportunities and challenges concern the out-of-box experience, open platforms, and system management.

1 Out-of-Box Experience

Industrial marketing research forecasts that the market value of FEC hardware components will reach $7,659 million by the year 2022 [10], which indicates that more FEC-ready equipment such as routers, switches, IP gateway, or hubs will be available in the market. Further, it is foreseeable that many of these products will feature the out-of-box experience(OOBE) in two **forms—OOBE-based equipment and OOBE-based software**.

1.1 OOBE-Based Equipment

OOBE-based equipment represents that the product vendors have integrated the FEC runtime platform with their products such as routers, switches, or other gateway devices in which the consumers who purposed the equipment can easily configure and deploy FEC applications on the equipment via certain user interfaces, which is similar to the commercial router products that have graphical user interfaces for users to configure customized settings.

1.2 OOBE-Based Software

This is similar to the experience of Microsoft Windows in which the users who own FEC-compatible devices can purchase and install OOBE-based FEC software on their devices toward enabling FEC runtime environment and the SCANC mechanisms without any extra low-level configuration. The OOBE-based FEC faces challenges in defining standardization for software and hardware. First, OOBE-based equipment raises a question for the vendors as to what FEC platform and related software packages should be included in their products. Second, OOBE-based software raises a question for the vendors regarding compatibility. Specifically, users may have devices in heterogeneous specification and processing units (e.g. x86, ARM etc.) in which the vendor may need to provide a version for each type of hardware.Moreover, developing and maintaining such an OOBE-based software can be extremely costly unless a corresponding common specification or standard for hardware exists.

Addressing the Challenges in Federating Edge Resources

Edge computing is rapidly evolving to alleviate latency, bandwidth, and quality-of-service (QoS) concerns of cloud-based applications as billions of 'things' are integrated to the Internet.

Current research has primarily focused on decentralizing resources away from centralized cloud data centres to the edge of the network and making use of them for improving application performance. Typically, edge resources are configured in an **ad hoc** manner and an application or a collection of applications may privately make use of them. These resources are not publicly available, for example, like cloud resources. **Additionally, edge resources are not evenly distributed but are sporadic in their geographic distribution.**

ad hoc, private, and sporadic edge deployments are less useful in transforming the global Internet. The benefits of using the edge should be equally accessible to both the developing and developed world for ensuring computational fairness and for connecting billions of devices to the Internet. However, there is minimal discourse on how edge deployments can be brought to bear in a global context – federating them across multiple geographic regions to create a global edge-based fabric that decentralizes data center computation. **This, of course, is currently impractical, not only because of technical challenges but also because it is shrouded by social, legal, and geopolitical issues.**

Figure 2.1 Networking and management challenges in federating edge resources.

SOFTWARE-DEFINED NETWORKING (SDN)

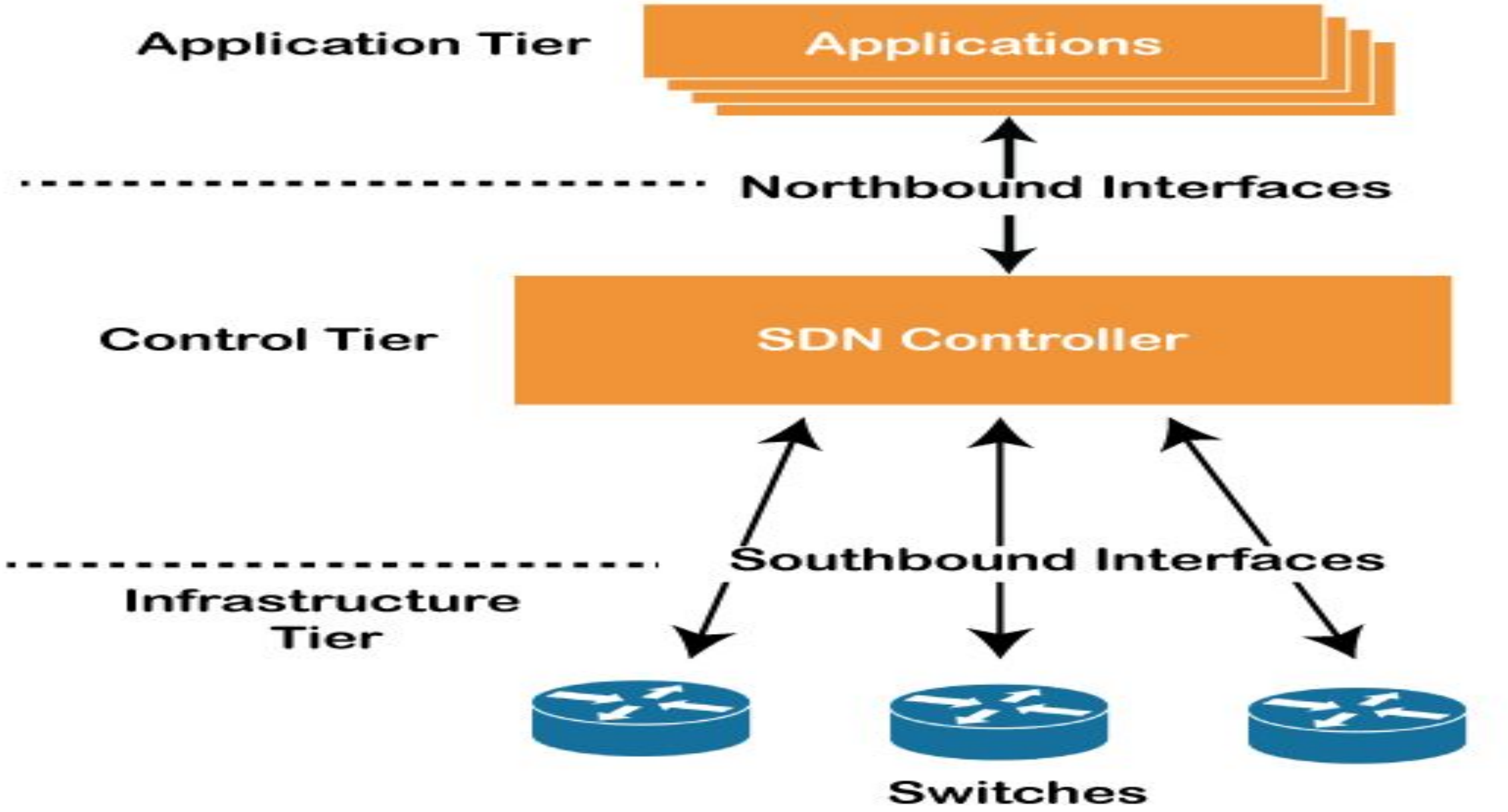


Table 2.1 Network challenges, their causes and potential solutions in federating edge resources.

Networking challenge	Why does it occur?	What is required?
User mobility	Keeping track of different mobility patterns	Mechanisms for application layer handover
QoS in a dynamic environment	Latency-intolerant services, dynamic state of the network	Reactive behavior of the network
Achieving a service-centric model	Enormous number of services with replications	Network mechanisms focusing on "what" instead of "where"
Ensuring reliability and service mobility	Devices and nodes joining the network (or leaving)	Frequent topology update, monitoring the servers and services
Managing multiple administrative domains	Heterogeneity, separate internal operations and characteristics, different service providers	Logically centralized, physically distributed control plane, vendor independency, global synchronization

1)A Service-Centric Model

The first challenge is in achieving a service-centric model on the edge. The traditional host-centric model follows the ‘server in a given geographic location’ model, which is restrictive in a number of ways. For example, simply transferring a virtual machine (VM) image from one location to another can be difficult. However, in global edge deployments, the focus needs to be on ‘what’ rather than ‘where’ so that services can be requested without prior knowledge of its geographic location . In this model, services may have a unique identifier, may be replicated in multiple regions, and may be coordinated. However, this is not a trivial task, given the current design of the Internet and protocol stacks, which do not facilitate global coordination of services.

2) Reliability and Service Mobility

The second challenge is in ensuring reliability and service mobility. User devices and edge nodes may connect and disconnect from the Internet instantly. This could potentially result in an unreliable environment. A casual end-user device will be expecting seamless service perhaps via a plug-and-play functionality to obtain services from the edge, but an unreliable network could result in latencies. The challenge here will be to mitigate this and create a reliable

environment that supports the edge. One mechanism to implement reliability is by either replicating services or by facilitating migration (considered in the management challenge) of services from one node to another. The key challenge here is to keep the overheads to a minimum so that the QoS of an application is not affected in any way.

3 Multiple Administrative Domains

The third challenge is in managing multiple administrative domains. The net-work infrastructure will need to be able to keep track of recent status of the network, edge servers and services deployed over them. When a collection of end-user devices requires a service at the edge, first the potential edge host will need to be determined. The most feasible edge node will then be chosen as the resource for the execution. There are two alternate scenarios that need to be considered for this operation: (i) the server is nearest to the end-users; or (ii) the potential server resides in another geographic region. Independent of the scenario, the network should forward the request to the server and return the response to the end user. During this progress, the data packets may travel across several distinct domains with multiple transport technologies. The challenge here is, given this heterogeneity, user experience must not be compromised and the technical details may need to be concealed from the user device.

Table 2.2 Management challenges, the need for addressing them, and potential solutions in federating edge resources.

Management challenge	Why should it be addressed?	What is required?
Discovery of edge nodes	To select resources when they are geographically spread and loosely coupled	Lightweight protocols and handshaking
Deployment of service and applications	Provide isolation for multiple services and applications	Monitoring and benchmarking mechanisms in real-time
Migrating services	User mobility, workload balancing	Low overhead virtualization
Load balancing	To avoid heavy subscription on individual nodes	Auto-scaling mechanisms

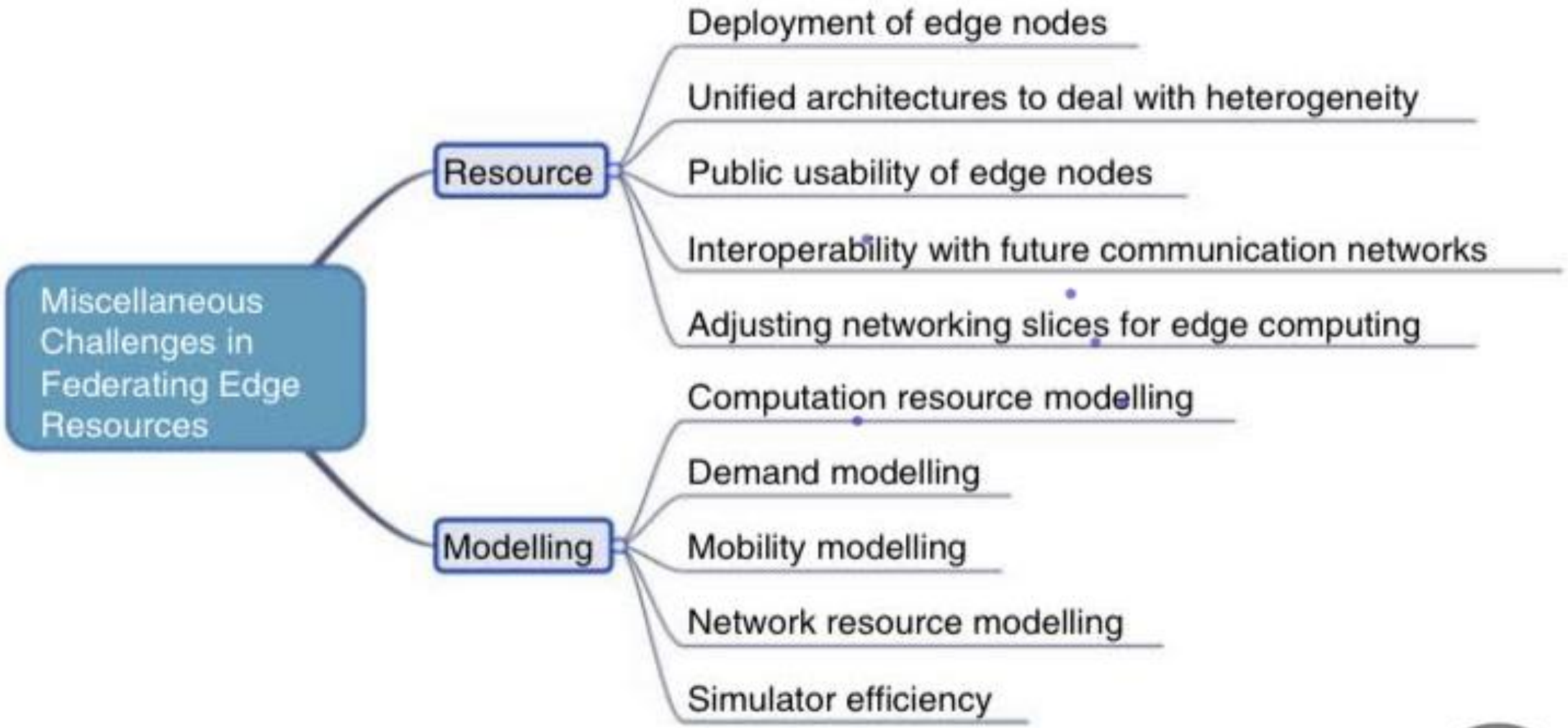


Figure 2.3 Resource and modeling challenges in federating edge resources.

How is Edge Intelligence different?

Edge Intelligence is a technology that is created by incorporating AI functionalities into Edge Computing. Even though Edge Computing has made data processing much more efficient, in order to keep up with the demand, there is a need to improve it. For this purpose, AI is essential due to its ability to quickly analyse huge volumes of data and provide valuable insights. This helps to improve the quality of the decision-making process.

There are **4 fundamental components** for edge intelligence. They are **edge caching, edge training, edge inference, and edge offloading.**

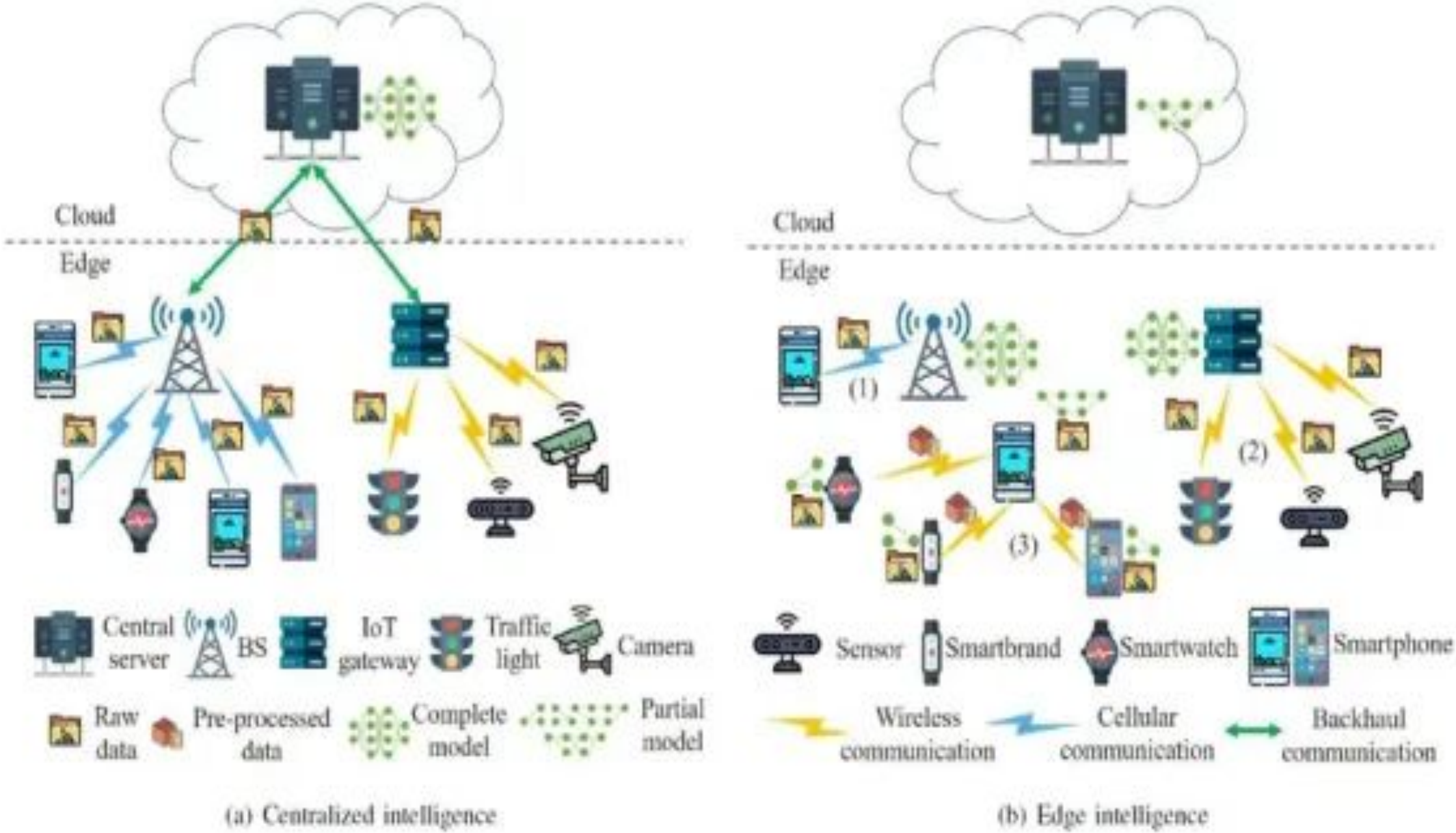
Edge Caching: It basically refers to a distributed data system proximity to end users, which collects and stores data generated from edge devices as well as data received from the internet to support intelligent tasks for users at the edge. Data is distributed at the edge itself. In edge caching, this collected data is used as input for intelligent applications, and results are sent back to where data is cached.

Edge Training: It refers to a distributed learning procedure. This model is trained to learn the optimal values for all the weights and bias, or the hidden patterns based on the training set cached at the edge.

Edge Inference: This is the stage where a trained model is used to infer the testing instance on edge devices. So, this allows the edge device to provide actionable intelligence using AI technologies

Edge Offloading: It is a distributed computing paradigm that provides a computing interface for edge caching, edge training, and edge inference. Edge devices that do not have enough resources for specific applications can offload some tasks to edge servers or other edge devices.

Edge Intelligence – Architecture



In the traditional model, all the edge devices first upload the data to a central server for performing intelligent tasks such as model training, inference, etc. The central server is usually located in the remote cloud. The results after processing are then sent back to the edge devices

In the EI model, the intelligent tasks are done by the edge servers/devices themselves. Only a very small/negligible amount of data needs to be uploaded to the cloud compared to the traditional models. The different functions are distributed among the different edge devices, which work together to complete the task.

Setting up Edge computing environments: development tools, python libraries.

Edge computing is a distributed computing framework that brings enterprise applications closer to data sources such as IoT devices or local edge servers. This proximity to data at its source can deliver strong business benefits, including faster insights, improved response times and better bandwidth availability.

Azure IoT Edge Dev Tool command-line tool (CLI). This tool is preferred for development.
Azure IoT Edge tools for Visual Studio Code extension. The extension is in maintenance mode.

What is Python development environment?
Python integrated development environments, or Python IDEs, are software platforms that provide programmers and developers with a comprehensive set of tools for software development in a single product, specifically in the Python programming language.

How many types of Python libraries are there?
Some common libraries are OpenCV, Apache Spark, TensorFlow, NumPy, etc.

How many libraries are in Python? There are over 137,000 Python libraries available today. These libraries can be helpful in creating applications in machine learning, data science, data manipulation, data visualization, etc

Edge computing platforms and frameworks: AWS IoT Greengrass, Azure IoT Edge, Google Cloud IoT Edge, IBM Edge Application Manager, KubeEdge

The Internet of Things platform is a collection of components that enables the following: Deployment of applications for monitoring, managing, and controlling connected devices. the framework supports customizations for a specific domain like medical, manufacturing...etc

What is edge computing framework?
Edge computing is a distributed computing framework that brings enterprise applications closer to data sources such as IoT devices or local edge servers. This proximity to data at its source can deliver strong business benefits, including faster insights, improved response times and better bandwidth availability.

An IoT Edge module is a container with executable code. You can deploy one or more modules to an IoT Edge device. Modules perform specific tasks like ingesting data from sensors, cleaning and analyzing data, or sending messages to an IoT hub. For more information, see Understand Azure IoT Edge modules.

When developing IoT Edge modules, it's important to understand the difference between the development machine and the target IoT Edge device where the module deploys. The container that you build to hold your module code must match the operating system (OS) of the target device. For example, the most common scenario is someone developing a module on a Windows computer intending to target a Linux device running IoT Edge. In that case, the container operating system would be Linux. As you go through this tutorial, keep in mind the difference between the development machine OS and the container OS.

IoT Platforms

Enterprises are increasingly connecting a broad variety and number of IoT endpoints to access data from and better manage physical assets that are relevant to their business. Typical IoT-enabled business objectives include traditional benefits, such as improved asset optimization, as well as new business opportunities and revenue models, such as subscribed-to services (versus owned assets). **An IoT platform is an on-premises software suite or a cloud service (IoT platform as a service [PaaS]) that monitors and may manage and control various types of endpoints, often via applications business units deploy on the platform**. The IoT platform usually provides (or provisions) Web-scale infrastructure capabilities to support basic and advanced IoT solutions and digital business operations.

An IoT platform is a **platform, including software and hardwar**e, used to manage internet-connected devices and the networks controlling them. It provides the infrastructure for connecting smart devices in a simple and secure way, as easy as any SaaS website you're used to using every day.

These platforms handle tasks like provisioning software, device management, secure data storage, and analytics. For a company that's just adopting IoT, it enables rapid deployment and iteration. And in a more mature IoT operation, these platforms save hours of work by making maintenance a one-click job.

IoT platforms help reduce those operations' energy consumption, improve safety and security, and enable real-time analytics. Data workflows spanning different departments can be built on a user-friendly interface, so there's no need for in-house developer talent. What's more, once your services are in customers' hands, an IoT platform can provide detailed insights into user behavior, enabling more targeted marketing and product development.

The 4 Types of IoT Platforms

1. IoT Connectivity Platforms. An IoT Connectivity Platform is used to manage and monitor the communication protocols that connect devices across WiFi, bluetooth, and mobile internet. ...
2. IoT Device Management Platforms. ...
3. IoT Application Enablement Platforms. ...
4. IoT Analytics Platforms.

The 4 Types of IoT Platforms:Not all IoT platforms are alike. Depending on where you are as a business, you might need to use one or all of them to get the results you desire. Let's go over the four types of IoT platforms and what they can do for you.

1. IoT Connectivity Platforms

An IoT Connectivity Platform is used to manage and monitor the communication protocols that connect devices across WiFi, bluetooth, and mobile internet. These platforms provide a user-friendly interface for the provisioning and management of devices across whichever networks you need to use in the moment.Running connected devices on a platform like this helps organizations reduce operating costs by making their networks more efficient and more stable. This reduces the cost of deploying these systems into new locations and maintaining them over time.

2. IoT Device Management Platforms

IoT Device Management Platforms provide tools for large organizations to monitor, troubleshoot, and update connected devices remotely. These platforms can handle the secure provisioning, configuration, and tracking of thousands of connected devices in real-time.Device management platforms also provide support for over-the-air software updates. These platforms allow organizations to keep their devices up-to-date, secure, and compliant with industry standards without the need for IT staff to spend hours upgrading every system on-site.

3. IoT Application Enablement Platforms

IoT Application Enablement Platforms create and deploy applications that leverage IoT data, whether they're smart home devices or industrial control systems.They also allow organizations to quickly develop scalable, secure, and feature-rich applications that are ready to be integrated with a wide range of IoT platforms, such as HomeKit or Google Cloud Platform, and gather the usage data they need to improve their operation.

4. IoT Analytics Platforms

IoT Analytics Platforms help organizations gain insight into the data generated by their connected devices.
Similar to something like Google Analytics, these platforms make it easy to perform in-depth analysis of the data gathered from connected devices, helping organizations to unlock the full potential of the IoT data.
With so much data coming from different devices and customers running different software versions, data might need a lot of complex formatting to fit on one unified database.

What is an IoT framework?

An IoT framework provides you with a solid foundation to build your application. When you use a framework, you get access to a range of features and capabilities for the development of IoT Solutions, applications, and smart connected products. Since you do not have to develop your product from scratch, you can build future-proof solutions at an accelerated pace, which both improves your time-to-market and helps build a competitive edge.

There are both open-source and proprietary IoT frameworks out there. However, open-source doesn't necessarily mean that you'll get everything for free - it's a misconception. Open-source usually indicates that the source code is available for everyone to use and improve. On the other hand, tech giants, such as Amazon Web Services, Google Cloud, and Cisco, offer proprietary, paid IoT Platforms.

Proprietary IoT frameworks:

Amazon Web Services IoT

AWS Internet of Things falls under the category of proprietary frameworks. It provides a broad and deep range of services and solutions to help IoT teams build intelligent IoT solutions with Artificial Intelligence (AI) and Machine Learning (ML) capabilities. With AWS IoT, an IoT team can easily connect and manage billions of devices and safeguard their device data with preventative mechanisms like encryption and access control.

IoT application development companies use AWS IoT offerings to complete complex IoT projects and streamline data analysis for industrial, commercial, customer, and automotive workloads. Additionally, the platform provides a highly secure and elastic infrastructure for IoT teams to scale easily and reliably.

Azure IoT

Azure IoT helps businesses build, deploy, and manage IoT applications at scale. IoT development agencies across the world tap into Azure's IoT capabilities to develop IoT applications and solutions for manufacturing, retail, energy, healthcare, and logistics. The cloud services platform provides development teams with intelligent edge-to-cloud technologies and an ecosystem of thousands of partners to enable businesses to solve industry-specific business challenges.

Cisco IoT Solutions

Cisco's end-to-end IoT solutions are applied across industries such as manufacturing, utilities, smart cities, ports and terminals, rail, roadways, etc. IoT teams use Cisco's IoT solutions to connect, manage, and scale assets, applications, and data in real-time to drive business results.

Frameworks are tools that help developers build apps, websites and digital systems. Since they function as blueprints for these different projects, frameworks can help programmers, developers and coders conduct tasks more effectively.

If you're interested in a career in software development, you may benefit from understanding frameworks and how they compare to other software development tools, such as libraries and programming languages.

What is a framework?

A framework is a foundation for developing software applications. Software engineers and developers use a framework as a template to create websites and applications. Developers do this by adding code to a framework, then personalizing it for their specific purpose.

A framework can combine multiple resources, such as an image or document file, to create a package unique to a project. Even after an application is complete, coders can revise the framework of an application to add new features or edit existing components.

Benefits of a framework

Frameworks are helpful tools for programmers and developers to use during the creation of websites and other applications. Here are some of the benefits a framework can offer:

- Saving software developers time and energy
- Providing a basic outline for coders to follow
- Allowing coders to focus on tasks more specific to their project
- Creating clean and adaptable code
- Reducing costs by shortening the amount of time a developer spends programming the application

Types of Frameworks

There are several types of frameworks. Developers define each by either the framework's function or the main coding language they add to it. Here are some of the most popular types of frameworks:

1. Web app framework

Developers use web app frameworks when designing a website. A web app framework allows a software engineer's creations to function well on the internet, and they usually have a higher rate of usability, making them inclusive to users. Websites require frequent updates and changes and developers and coders benefit from using web app frameworks, as they're easy to adjust.

2. Mobile app framework

A mobile app framework provides a general structure for developers to add onto to create an application for mobile devices, such as smartphones. These frameworks are often open-source, and developers can use a variety of coding languages to create them. While the mobile app framework is often similar to a web app framework, this framework allows software developers to format the application specifically for easy use on a smartphone or tablet.

3. Technology framework

Software engineers generally use technology frameworks for business purposes. This framework allows them to establish information technology (IT) systems within a company's database. Uses for a technology framework can include security measures for discrete data, management tools and common applications.

4. Enterprise architecture framework

An enterprise architecture framework provides a blueprint for complex IT systems within a company. There are four major types of enterprise architecture frameworks, which developers choose based on the framework's fit for their specific project. These framework subtypes include the Zachman, Federal, Gartman and the Open Group enterprise framework (TOGAF). Developers most commonly use the latter.

5. Database framework

Software developers use database frameworks to manage a variety of database engines. They also help developers perform database management tasks quickly and without spending time and effort on additional programming.

Developers use the database framework to help analyze, sort and find data within a certain collection. Database frameworks can also act as a starting structure for software developers to create functions and commands that other people can use to perform the same database management tasks without coding knowledge.

6. Testing framework

Software developers use testing frameworks as guidelines for creating test cases. This helps supply developers with tools and practices that allow them to complete quality assurance tasks. Testing frameworks remind developers what to test for and how to complete tests, and they ensure accuracy and efficiency in quality assurance practices.

Characteristics of an effective framework

Because a variety of frameworks exist, it's helpful to understand the qualities of a good framework. This can save you time when selecting the right framework for your project. Here are some characteristics of an effective framework:

- Documentation: It's a good idea to choose a framework that uses well-documented code. If it provides implementation training, this can also save you time.
- Community engagement: Successful frameworks have active user support, so seek out a well-established framework with an active user base.
- Functionality: Different frameworks have different purposes, so consider why you need one and what you need to help your team accomplish. These considerations ensure you don't select a framework that's too complex or has too many features.

Library vs. framework

The difference between a library and a framework is that a framework requires coding to function. Similarly, code requires libraries to operate. Some developers work with both libraries and frameworks, while others choose between the two.

Generally, the main way developers distinguish the two is by their primary purposes. Libraries can help to solve specific issues or to create unique applications and functions. Frameworks provide a general blueprint for an application and are reusable for several projects. Here are some other key differences between the two:

Programmer control and customization

While the programmer controls both a framework and a library, libraries allow developers to complete their work more independently, as they're a resource to reference for a specific reason. Frameworks automatically perform some functions that a programmer normally would. Though developers can change or add different aspects to a framework, libraries are completely unique for a project, so developers can customize them entirely.

Extensibility

Because developers build frameworks with the intention of using them for more than one project, frameworks are often generic. This allows frameworks to have more extensibility, which helps developers when they want to add to the framework for new projects. Software engineers build libraries for more specific functions. Because of this, there's less of a need for extensibility.

Programming language vs. framework

While developers use both programming languages and frameworks to develop software, they serve different purposes. You build a framework for one specific purpose, but you can build many things with a programming language.

Additionally, a programming language helps construct each framework. Consider, too, that while you don't always require a framework to develop software, you always use a programming language to develop software.

How to choose an IoT framework?

Device management is your top priority when selecting a framework to develop and deploy your IoT applications. However, in today's competitive landscape, there are numerous other factors that you need to consider to start building on a framework that helps you achieve your project goals.

A future-proof IoT platform: Invest in a framework that is secure, scalable, and can evolve with the market. IoT is still in an early phase, and a lot of new technologies are likely to impact its ecosystem. That's why you should consider a platform that is continuously updating itself in line with the new standards and protocols that emerge in the IoT landscape, whether they impact the system or cloud environment, security stack, or end-point applications.

Security: This aspect of IoT development is crucial for the success of your project. Thoroughly analyze the security aspects of any IoT platform. Choosing a secure framework will ensure that your confidential or third-party data and information is safe and secure, and that breaches and other forms of cyber attacks do not ruin your business reputation.

Time-to-value: The ideal framework should simplify and accelerate your development and deployment process. Look for platforms that provide end-to-end solutions so you do not have to build your PoC using one framework and develop and deploy on another. Using a platform that provides full-suite IoT capabilities often leads to faster time to market and more secure and reliable products.

Pricing: It's one of the critical considerations in the selection of an IoT framework. Understand the pricing models of all potential IoT platforms. Speak with their teams. Look for hidden costs (if there are any) so no surprises await you when you are about to close a deal or start building a project.

Don't overcommit to one scenario: The success of an IoT project depends on the tech stack and resources you use to connect, control, and secure your devices and applications. Cloud is a powerful technology, but you may also need to run some workloads across edge and on-premises. So, it is essential not to overcommit to one scenario and have the bandwidth to use modern architecture to reach a wider customer base while not compromising on data security and compliance.

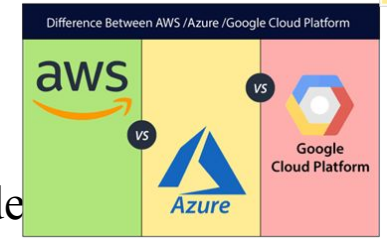
Difference between AWS, Azure, and Google Cloud Platform : However, each platform has its strengths and areas of focus. AWS is known for its wide range of services, Azure has a strong focus on building and deploying applications, and GCP is known for its machine learning and data analytics offerings

Amazon Web Services (AWS)

Amazon Web Services (AWS) is a cloud computing platform which was introduced in 2002. It offers a wide range of cloud services such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

AWS provides the largest community with millions of active customers as well as thousands of partners globally. Most of the organizations use AWS to expand their business by moving their IT management to the AWS.

Flexibility, security, scalability, and better performance are some important features of AWS.



Microsoft Azure

Microsoft Azure is also called as Windows Azure. It is a worldwide cloud platform which is used for building, de managing services. It supports multiple programming languages such as Java, Nodejs, C, and C#. The advantage of using Microsoft Azure is that it allows us to a wide variety of services without arranging and purchasing additional hardware components.

Microsoft Azure provides several computing services, including servers, storage, databases, software, networking, and analytics over the Internet.

Google Cloud Platform (GCP)

Google Cloud Platform (GCP) is introduced by Google in 2011. It allows us to use Google's products such as Google search engine, Gmail, YouTube, etc. Most of the companies use this platform to easily build, move, and deploy applications on the cloud. It allows us to access these applications using a high-speed internet connection. The advantage of GCP is that it supports various databases such as SQL, MYSQL, Oracle, Sam, and more.

Google Cloud Platform (GCP) provides various cloud computing services, including computing, data analytics, data storage, and machine learning.

Parameter	AWS	Azure	Google Cloud Platform
App Testing	It uses device farm	It uses DevTest labs	It uses Cloud Test labs.
API Management	Amazon API gateway	Azure API gateway	Cloud endpoints.
Kubernetes Management	EKS	Kubernetes service	Kubernetes engine
Git Repositories	AWS source repositories	Azure source repositories	Cloud source repositories.
Data warehouse	Redshift	SQL warehouse	Big Query
Object Storage	S3	Block Blobs and files	Google cloud storage.
Relational DB	RDS	Relational DBs	Google Cloud SQL
Block Storage	EBS	Page Blobs	Persistent disks
Marketplace	AWS	Azure	G suite
File Storage	EFS	Azure Files	ZFS and Avere
Media Services	Amazon Elastic transcoder	Azure media services	Cloud video intelligence API
Virtual network	VPC	VNet	Subnet
Pricing	Per hour	Per minute	Per minute
Maximum processors in VM	128	128	96
Maximum memory in VM (GiB)	3904	3800	1433
Catching	ElasticCache	RedisCache	CloudCDN
Load Balancing Configuration	Elastic Load Balancing	Load Balancer Application Gateway	Cloud Load Balancing
Global Content Delivery Networks	CloudFront	Content Delivery Network	Cloud Interconnect

Azure IoT Edge modules are implemented as containers, so IoT Edge needs a container engine to launch them. Microsoft provides a container engine, moby-engine, to fulfill this requirement. This container engine is based on the Moby open-source project. Docker CE and Docker EE are other popular container engines. They're also based on the Moby open-source project and are compatible with Azure IoT Edge. Microsoft provides best effort support for systems using those container engines; however, Microsoft can't ship fixes for issues in them. For this reason, Microsoft recommends using moby-engine on production systems.

Azure IoT Edge is an Internet of Things (IoT) service that builds on top of IoT Hub. This service is ideal for solutions that need to analyze data on edge nodes or devices, instead of in the cloud

Azure IoT Edge supported platforms

Get support
If you experience problems while using the Azure IoT Edge service, there are several ways to seek support. Try one of the following channels for support:

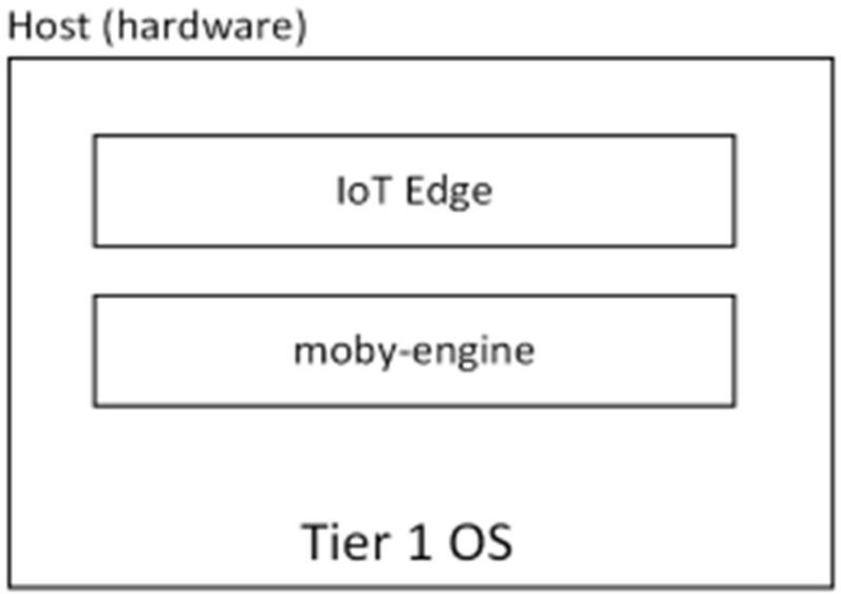
Reporting bugs - Most development that goes into the Azure IoT Edge product happens in the IoT Edge open-source project. Bugs can be reported on the issues page of the project. Bugs related to Azure IoT Edge for Linux on Windows can be reported on the iot edge-eflow issues page. Fixes rapidly make their way from the projects in to product updates.

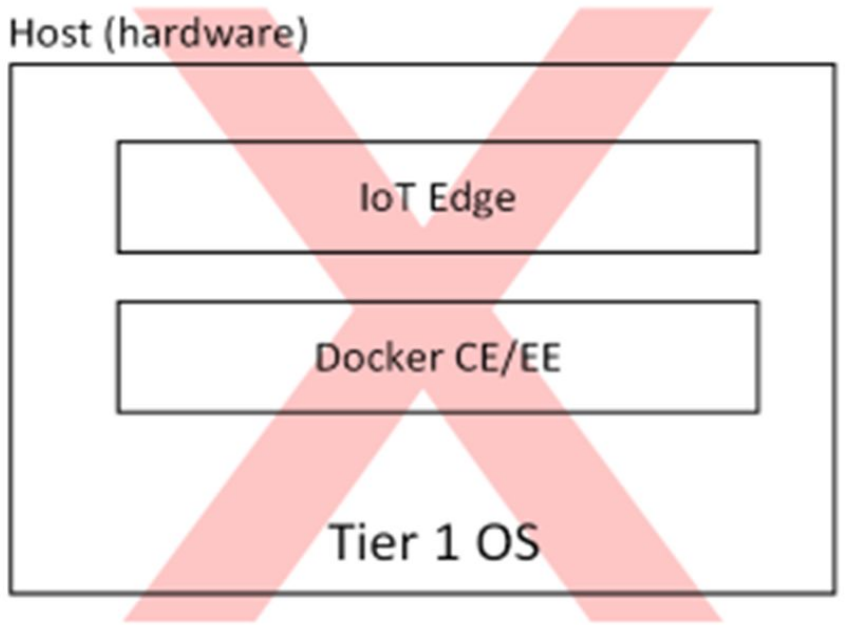
Microsoft Customer Support team - Users who have a support plan can engage the Microsoft Customer Support team by creating a support ticket directly from the Azure portal.

Feature requests - The Azure IoT Edge product tracks feature requests via the product's Azure feedback community.

Container engines

Azure IoT Edge modules are implemented as containers, so IoT Edge needs a container engine to launch them. Microsoft provides a container engine, moby-engine, to fulfill this requirement. This container engine is based on the Moby open-source project. Docker CE and Docker EE are other popular container engines. They're also based on the Moby open-source project and are compatible with Azure IoT Edge. Microsoft provides best effort support for systems using those container engines; however, Microsoft can't ship fixes for issues in them. For this reason, Microsoft recommends using moby-engine on production systems.





- This configuration works; however it should only be used when using your Windows machine as a dev box for a solution that will be run on Linux

Operating systems

Azure IoT Edge runs on most operating systems that can run containers; however, not all of these systems are equally supported. Operating systems are grouped into tiers that represent the level of support users can expect.

Tier 1 systems are supported. For tier 1 systems, Microsoft:
has this operating system in automated tests
provides installation packages for them
Tier 2 systems are compatible with Azure IoT Edge and can be used relatively easily. For tier 2 systems:
Microsoft has done informal testing on the platforms or knows of a partner successfully running Azure IoT Edge on the platform
Installation packages for other platforms may work on these platforms

Tier 1

The systems listed in the following tables are supported by Microsoft, either generally available or in public preview, and are tested with each new release.

Linux containers

Modules built as Linux containers can be deployed to either Linux or Windows devices. For Linux devices, the IoT Edge runtime is installed directly on the host device. For Windows devices, a Linux virtual machine prebuilt with the IoT Edge runtime runs on the host device.

IoT Edge for Linux on Windows is the recommended way to run IoT Edge on Windows devices.

Windows containers

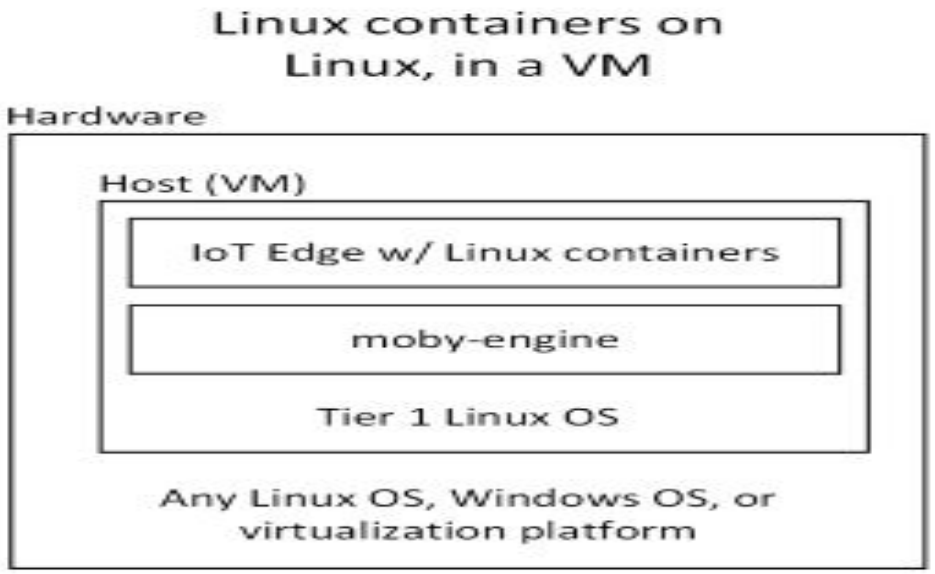
We no longer support Windows containers. IoT Edge for Linux on Windows is the recommended way to run IoT Edge on Windows devices.

Tier 2

The systems listed in the following table are considered compatible with Azure IoT Edge, but aren't actively tested or maintained by Microsoft.

Virtual Machines

Azure IoT Edge can be run in virtual machines, such as an Azure Virtual Machine. Using a virtual machine as an IoT Edge device is common when customers want to augment existing infrastructure with edge intelligence. The family of the host VM OS must match the family of the guest OS used inside a module's container. This requirement is the same as when Azure IoT Edge is run directly on a device. Azure IoT Edge is agnostic of the underlying virtualization technology and works in VMs powered by platforms like Hyper-V and vSphere.



Minimum system requirements

Azure IoT Edge runs great on devices as small as a Raspberry Pi3 to server grade hardware. Choosing the right hardware for your scenario depends on the workloads that you want to run. Making the final device decision can be complicated; however, you can easily start prototyping a solution on traditional laptops or desktops.

Experience while prototyping will help guide your final device selection. Questions you should consider include:

- How many modules are in your workload?
- How many layers do your modules' containers share?
- In what language are your modules written?
- How much data will your modules be processing?
- Do your modules need any specialized hardware for accelerating their workloads?
- What are the desired performance characteristics of your solution?
- What is your hardware budget?

AWS IoT Greengrass – Software for edge computing

AWS IoT Greengrass is software that lets customers run local compute, messaging, data caching, sync, and ML inference capabilities for connected devices, allowing connected devices to operate even with intermittent connectivity to the cloud. After the device reconnects, AWS IoT Greengrass synchronizes the data on the device with AWS IoT Core, providing constant functionality regardless of connectivity. AWS IoT Greengrass seamlessly extends AWS to devices so they can act locally on the data they generate, while still using the cloud for management, analytics, and durable storage.

Security capabilities
AWS IoT Greengrass authenticates and encrypts device data for both local and cloud communications, and data is never exchanged between devices and the cloud without proven identity. The service uses security and access management similar to what customers are familiar with in AWS IoT Core, with mutual device authentication and authorization, and secure connectivity to the cloud.

More specifically, AWS IoT Greengrass uses X.509 certificates, managed subscriptions, AWS IoT policies, and AWS Identity and Access Management (IAM) policies and roles to ensure that AWS IoT Greengrass applications are secure. AWS IoT devices require an AWS IoT thing, a device certificate, and an AWS IoT policy to connect to the AWS IoT Greengrass service. This allows AWS IoT Greengrass core devices to securely connect to the AWS IoT cloud service. It also allows the AWS IoT Greengrass cloud service to deploy configuration information, AWS Lambda functions, and managed subscriptions to AWS IoT Greengrass core devices. In addition, AWS IoT Greengrass provides hardware root of trust private key storage for edge devices.

Other important security capabilities of AWS IoT Greengrass are monitoring and logging. For example, core software in the service can write logs to Amazon CloudWatch (which also functions for AWS IoT Core) and to the local file system of customers' core devices. Logging is configured at the group level and all AWS IoT Greengrass log entries include a time stamp, log level, and information about the event. AWS IoT Greengrass is integrated with AWS CloudTrail—a service that provides a record of actions taken by a user, role, or an AWS service in AWS IoT Greengrass—and if activated by the customer, it captures application programming interface (API) calls for AWS IoT Greengrass as events. This includes calls from the AWS IoT Greengrass console and code calls to the AWS IoT Greengrass API operations. For example, customers can create a trail and calls can enable continuous delivery of AWS CloudTrail events to an Amazon Simple Storage Service (Amazon S3) bucket, including events for AWS IoT Greengrass. If customers don’t want to create a trail, they can view the most recent events in the AWS CloudTrail console in event history. This information can be used to do a number of things, such as determining when a request was made to AWS IoT Greengrass and the IP address from which the request was made.

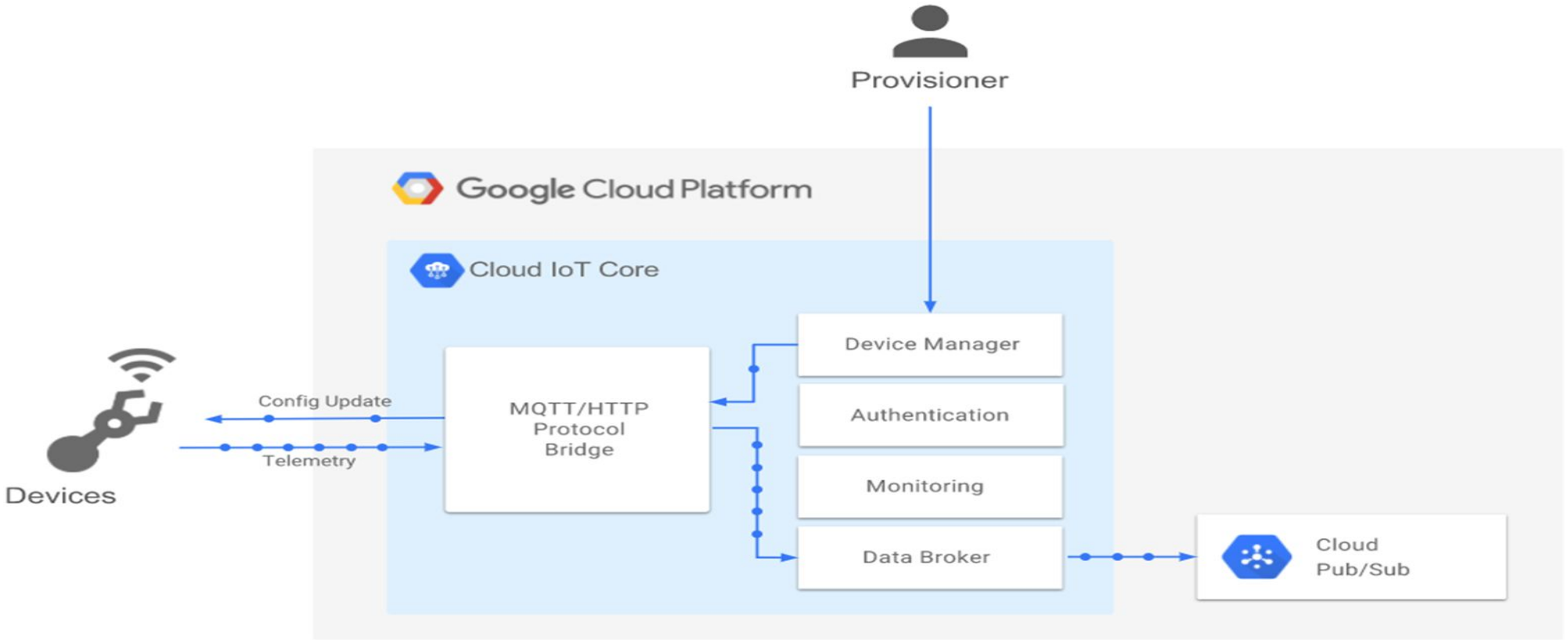
What is Google IoT cloud?

Google Cloud Internet of Things (IoT) Core is a fully managed service for securely connecting and managing IoT devices, from a few to millions. Ingest data from connected devices and build rich applications that integrate with the other big data services of Google Cloud Platform.

Components
The main components of Cloud IoT Core are the device manager and the protocol bridges:

A device manager for registering devices with the service, so you can then monitor and configure them
Two protocol bridges (MQTT and HTTP) that devices can use to connect to Google Cloud Platform
Device telemetry data is forwarded to a Cloud Pub/Sub topic, which can then be used to trigger Cloud Functions. You can also perform streaming analysis with Cloud Dataflow or custom analysis with your own subscribers.

The following diagram summarizes the service components and the flow of data:



How do you manage and promote security across thousands of edge servers and hundreds of thousands of edge devices cost-effectively? The answer: IBM Edge Computing® Manager, an intelligent and flexible application that provides autonomous management for edge computing. A single administrator can manage the scale, variability and rate of change of application environments across endpoints simultaneously.

1. **IAAS: Infrastructure As A Service (IAAS)** is means of delivering computing infrastructure as on-demand services. It is one of the three fundamental cloud service models. The user purchases servers, software data center space, or network equipment and rent those resources through a fully outsourced, on-demand service model. It allows dynamic scaling and the resources are distributed as a service. It generally includes multiple-user on a single piece of hardware. A good example of IaaS is AWS EC2.

It totally depends upon the customer to choose its resources wisely and as per need. Also, it provides billing management too.

2. **PAAS: Platform As A Service (PAAS)** is a cloud delivery model for applications composed of services managed by a third party. It provides elastic scaling of your application which allows developers to build applications and services over the internet and the deployment models include public, private and hybrid. E.g. AWS Elastic Beanstalk.

Basically, it is a service where a third-party provider provides both software and hardware tools to the cloud computing. The tools which are provided are used by developers. PAAS is also known as Application PAAS. It helps us to organize and maintain useful applications and services. It has a well-equipped management system and is less expensive compared to IAAS.

3. **SAAS: Software As A Service (SAAS)** allows users to run existing online applications and it is a model software that is deployed as a hosting service and is accessed over Output Rephrased/Re-written Text the internet or software delivery model during which software and its associated data are hosted centrally and accessed using their client, usually an online browser over the web. SAAS services are used for the development and deployment of modern applications. Google Workspace — formerly known as Google G Suite — is one of the most popular SaaS-based infrastructure services.
It allows software and its functions to be accessed from anywhere with good internet connection device and a browser. An application is hosted centrally and also provides access to multiple users across various locations via the internet.

https://youtu.be/36zducUX16w

Basis Of	IAAS	PAAS	SAAS
Stands for	Infrastructure as a service.	Platform as a service.	Software as a service.
Uses	IAAS is used by network architects.	PAAS is used by developers.	SAAS is used by the end user.
Model	It is a service model that provides virtualized computing resources over the internet.	It is a cloud computing model that delivers tools that are used for the development of applications.	It is a service model in cloud computing that hosts software to make it available to clients.
Technical understanding.	It requires technical knowledge.	Some knowledge is required for the basic setup.	There is no requirement about technicalities company handles everything.
Popularity	It is popular among developers and researchers.	It is popular among developers who focus on the development of apps and scripts.	It is popular among consumers and companies, such as file sharing, email, and networking.
Percentage rise	It has around a 12% increment.	It has around 32% increment.	It has about a 27 % rise in the cloud computing model.

Usage	Used by the skilled developer to develop unique applications.	Used by mid-level developers to build applications.	Used among the users of entertainment.
Cloud services.	Amazon Web Services, sun, vCloud Express.	Facebook, and Google search engine.	MS Office web, Facebook and Google Apps.

Enterprise services.	AWS virtual private cloud.	Microsoft Azure.	IBM cloud analysis.
Outsourced cloud services.	Salesforce	Force.com, Gigaspaces.	AWS, Terremark
User Controls	Operating System, Runtime, Middleware, and Application data	Data of the application	Nothing

Others	It is highly scalable and flexible.	It is highly scalable to suit the different businesses according to resources.	It is highly scalable to suit the small, mid and enterprise level business

Virtualization

Virtualization is the "creation of a virtual (rather than actual) version of something, such as a server, a desktop, a storage device, an operating system or network resources".

In other words, Virtualization is a technique, which allows to share a single physical instance of a resource or an application among multiple customers and organizations. It does by assigning a logical name to a physical storage and providing a pointer to that physical resource when demanded.

What is the concept behind the Virtualization?
Creation of a virtual machine over existing operating system and hardware is known as Hardware Virtualization. A Virtual machine provides an environment that is logically separated from the underlying hardware.

The machine on which the virtual machine is going to create is known as Host Machine and that virtual machine is referred as a Guest Machine

Types of Virtualization:

1. Hardware Virtualization.
2. Operating system Virtualization.
3. Server Virtualization.
4. Storage Virtualization.

1) Hardware Virtualization:

When the virtual machine software or virtual machine manager (VMM) is directly installed on the hardware system is known as hardware virtualization.

The main job of hypervisor is to control and monitoring the processor, memory and other hardware resources.

After virtualization of hardware system we can install different operating system on it and run different applications on those OS.

Usage:

Hardware virtualization is mainly done for the server platforms, because controlling virtual machines is much easier than controlling a physical server.

2) Operating System Virtualization:

When the virtual machine software or virtual machine manager (VMM) is installed on the Host operating system instead of directly on the hardware system is known as operating system virtualization.

Usage:

Operating System Virtualization is mainly used for testing the applications on different platforms of OS.

3) Server Virtualization:
When the virtual machine software or virtual machine manager (VMM) is directly installed on the Server system is known as server virtualization.

Usage:
Server virtualization is done because a single physical server can be divided into multiple servers on the demand basis and for balancing the load.

4) Storage Virtualization:
Storage virtualization is the process of grouping the physical storage from multiple network storage devices so that it looks like a single storage device.

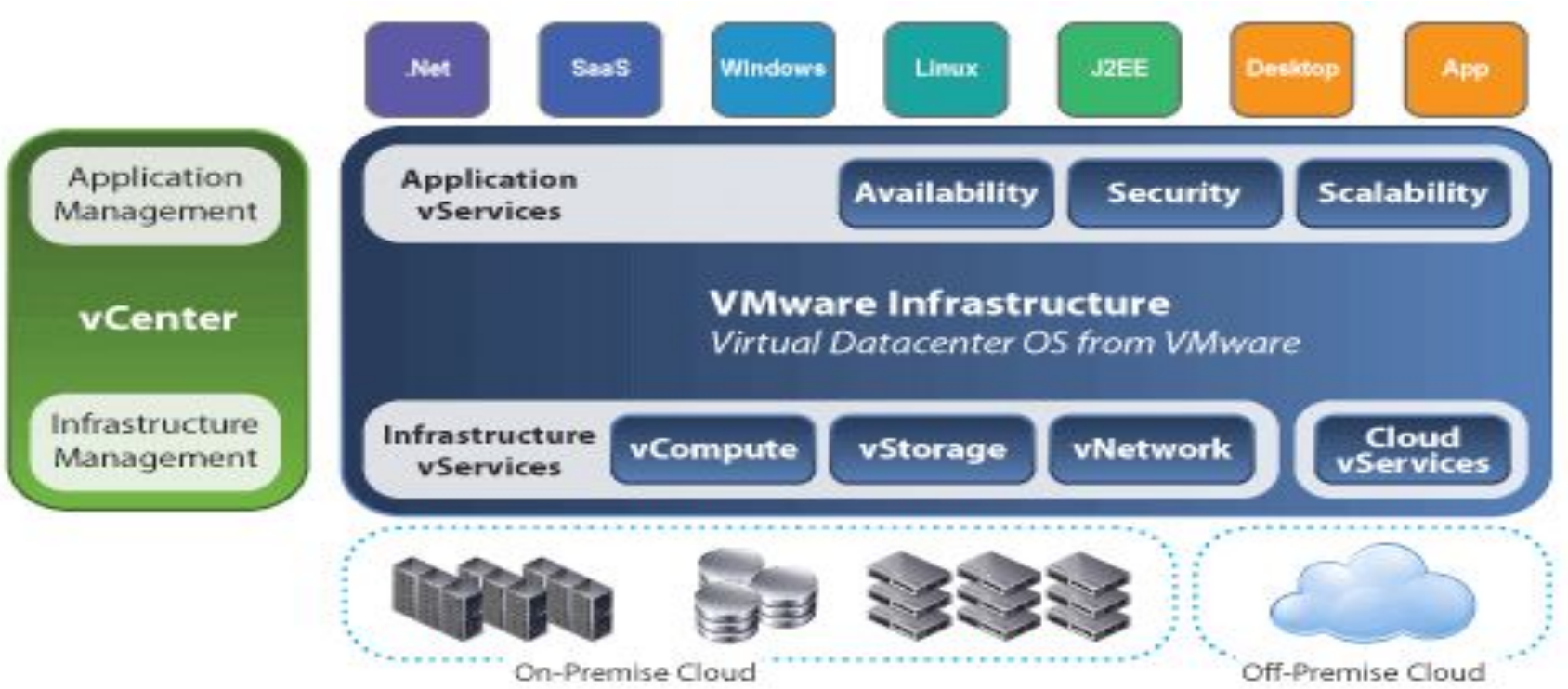
Storage virtualization is also implemented by using software applications.

Usage:
Storage virtualization is mainly done for back-up and recovery purposes.

Virtualization plays a very important role in the cloud computing technology, normally in the cloud computing, users share the data present in the clouds like application etc,but actually with the help of virtualization users shares the Infrastructure.

The main usage of Virtualization Technology is to provide the applications with the standard versions to their cloud users, suppose if the next version of that application is released, then cloud provider has to provide the latest version to their cloud users and practically it is not possible because it is more expensive.

To overcome this problem we use basically virtualization technology, By using virtualization, all severs and the software application which are required by other cloud providers are maintained by the third party people, and the cloud providers has to pay the money on monthly or annual basis.



Data Virtualization

Data virtualization is the process of retrieve data from various resources without knowing its type and physical location where it is stored. It collects heterogeneous data from different resources and allows data users across the organization to access this data according to their work requirements. This heterogeneous data can be accessed using any application such as web portals, web services, E-commerce, Software as a Service (SaaS), and mobile application.

We can use Data Virtualization in the field of data integration, business intelligence, and cloud computing.

Advantages of Data Virtualization

There are the following advantages of data virtualization -

- It allows users to access the data without worrying about where it resides on the memory.
- It offers better customer satisfaction, retention, and revenue growth.
- It provides various security mechanism that allows users to safely store their personal and professional information.
- It reduces costs by removing data replication.
- It provides a user-friendly interface to develop customized views.
- It provides various simple and fast deployment resources.
- It increases business user efficiency by providing data in real-time.
- It is used to perform tasks such as data integration, business integration, Service-Oriented Architecture (SOA) data services, and enterprise search.

Disadvantages of Data Virtualization

- It creates availability issues, because availability is maintained by third-party providers.
- It required a high implementation cost.
- It creates the availability and scalability issues.
- Although it saves time during the implementation phase of virtualization but it consumes more time to generate the appropriate result.

Uses of Data Virtualization

There are the following uses of Data Virtualization -

1. Analyze performance

Data virtualization is used to analyze the performance of the organization compared to previous years.

2. Search and discover interrelated data

Data Virtualization (DV) provides a mechanism to easily search the data which is similar and internally related to each other.

3. Agile Business Intelligence

It is one of the most common uses of Data Virtualization. It is used in agile reporting, real-time dashboards that require timely aggregation, analyse and present the relevant data from multiple resources. Both individuals and managers use this to monitor performance, which helps to make daily operational decision processes such as sales, support, finance, logistics, legal, and compliance

4. Data Management

Data virtualization provides a secure centralized layer to search, discover, and govern the unified data and its relationships.

Industries that use Data Virtualization

- Communication & Technology

  In Communication & Technology industry, data virtualization is used to increase revenue per customer, create a real-time ODS for marketing, manage customers, improve customer insights, and optimize customer care, etc.

- Finance

  In the field of finance, DV is used to improve trade reconciliation, empowering data democracy, addressing data complexity, and managing fixed-risk income.

- Government

  In the government sector, DV is used for protecting the environment.

- Healthcare

  Data virtualization plays a very important role in the field of healthcare. In healthcare, DV helps to improve patient care, drive new product innovation, accelerating M&A synergies, and provide a more efficient claims analysis.

- Manufacturing

  In manufacturing industry, data virtualization is used to optimize a global supply chain, optimize factories, and improve IT assets utilization.

Data Virtualization Tools

There are the following Data Virtualization tools -

1. Red Hat JBoss data virtualization

Red Hat virtualization is the best choice for developers and those who are using micro services and containers. It is written in Java.

2. TIBCO data virtualization

TIBCO helps administrators and users to create a data virtualization platform for accessing the multiple data sources and data sets. It provides a builtin transformation engine to combine non-relational and un-structured data sources.

3. Oracle data service integrator

It is a very popular and powerful data integrator tool which is mainly worked with Oracle products. It allows organizations to quickly develop and manage data services to access a single view of data.

4. SAS Federation Server

SAS Federation Server provides various technologies such as scalable, multi-user, and standards-based data access to access data from multiple data services. It mainly focuses on securing data.

5. Denodo

Denodo is one of the best data virtualization tools which allows organizations to minimize the network traffic load and improve response time for large data sets. It is suitable for both small as well as large organizations.

What is containerization?

Containerization is a software deployment process that bundles an application's code with all the files and libraries it needs to run on any infrastructure. Traditionally, to run any application on your computer, you had to install the version that matched your machine's operating system. For example, you needed to install the Windows version of a software package on a Windows machine. However, with containerization, you can create a single software package, or container, that runs on all types of devices and operating systems.

What are the benefits of containerization?

Developers use containerization to build and deploy modern applications because of the following advantages.

- Portability

  Software developers use containerization to deploy applications in multiple environments without rewriting the program code. They build an application once and deploy it on multiple operating systems. For example, they run the same containers on Linux and Windows operating systems. Developers also upgrade legacy application code to modern versions using containers for deployment.

- Scalability

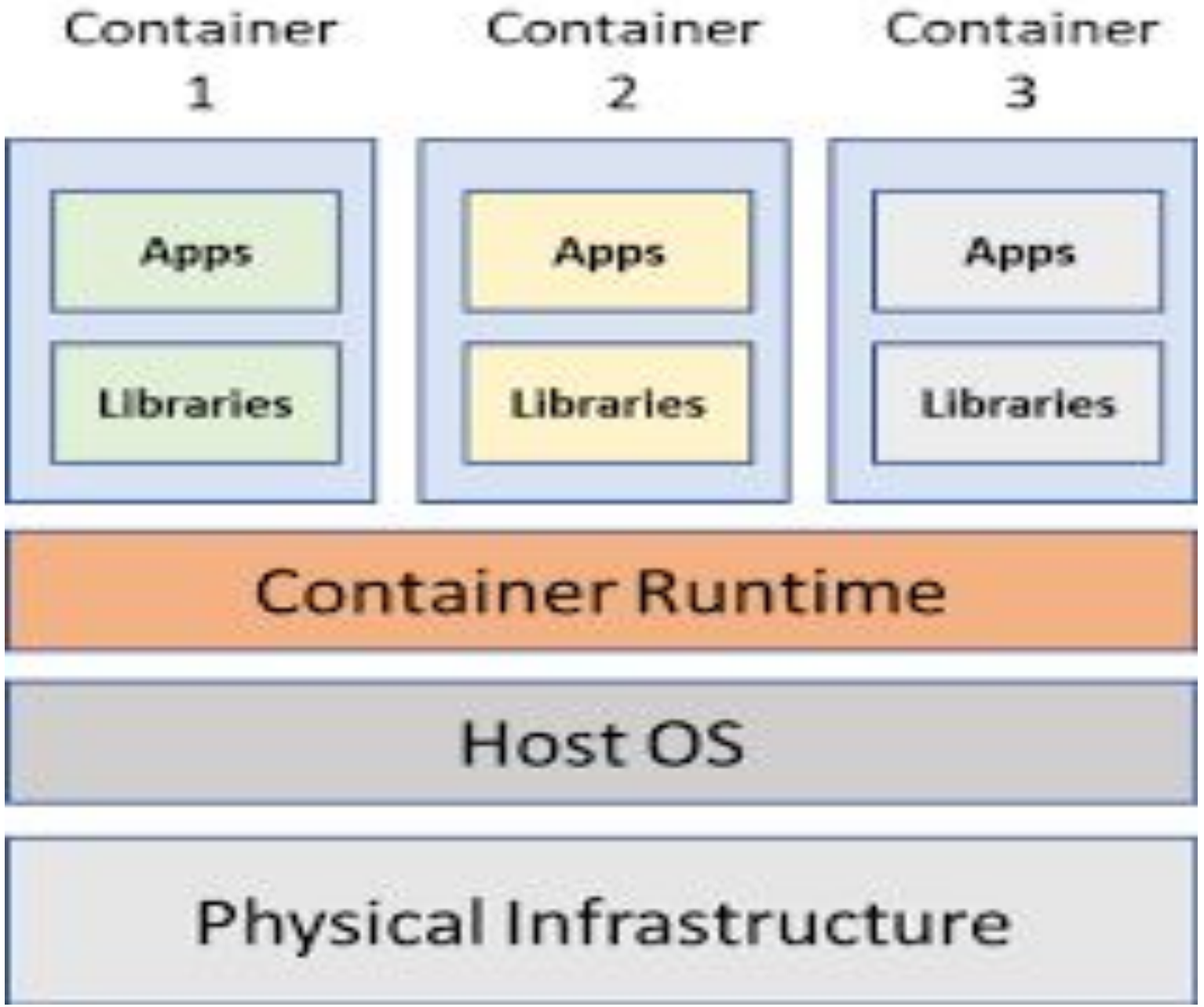
  Containers are lightweight software components that run efficiently. For example, a virtual machine can launch a containerized application faster because it doesn't need to boot an operating system. Therefore, software developers can easily add multiple containers for different applications on a single machine. The container cluster uses computing resources from the same shared operating system, but one container doesn't interfere with the operation of other containers.

- Fault tolerance

  Software development teams use containers to build fault-tolerant applications. They use multiple containers to run microservices on the cloud. Because containerized microservices operate in isolated user spaces, a single faulty container doesn't affect the other containers. This increases the resilience and availability of the application.

- Agility

  Containerized applications run in isolated computing environments. Software developers can troubleshoot and change the application code without interfering with the operating system, hardware, or other application services. They can shorten software release cycles and work on updates quickly with the container model.


Container 1
Container 2
Container 3
Apps
Libraries
Apps
Libraries
Apps
Libraries
Container Runtime
Host OS
Physical Infrastructure

How does containerization work?

Containerization involves building self-sufficient software packages that perform consistently, regardless of the machines they run on. Software developers create and deploy container images—that is, files that contain the necessary information to run a containerized application. Developers use containerization tools to build container images based on the Open Container Initiative (OCI) image specification. OCI is an open-source group that provides a standardized format for creating container images. Container images are read-only and cannot be altered by the computer system.

Container images are the top layer in a containerized system that consists of the following layers.

- Infrastructure

  Infrastructure is the hardware layer of the container model. It refers to the physical computer or bare-metal server that runs the containerized application.

- Operating system

  The second layer of the containerization architecture is the operating system. Linux is a popular operating system for containerization with on-premise computers. In cloud computing, developers use cloud services such as AWS EC2 to run containerized applications.

- Container engine

  The container engine, or container runtime, is a software program that creates containers based on the container images. It acts as an intermediary agent between the containers and the operating system, providing and managing resources that the application needs. For example, container engines can manage multiple containers on the same operating system by keeping them independent of the underlying infrastructure and each other.

- Application and dependencies

  The topmost layer of the containerization architecture is the application code and the other files it needs to run, such as library dependencies and related configuration files. This layer might also contain a light guest operating system that gets installed over the host operating system.

What are the types of container technology?

The following are some examples of popular technologies that developers use for containerization.

Docker

Docker, or Docker Engine, is a popular open-source container runtime that allows software developers to build, deploy, and test containerized applications on various platforms. Docker containers are self-contained packages of applications and related files that are created with the Docker framework.

Linux

Linux is an open-source operating system with built-in container technology. Linux containers are self-contained environments that allow multiple Linux-based applications to run on a single host machine. Software developers use Linux containers to deploy applications that write or read large amounts of data. Linux containers do not copy the entire operating system to their virtualized environment. Instead, the containers consist of necessary functionalities allocated in the Linux namespace.

Kubernetes

Kubernetes is a popular open-source container orchestrator that software developers use to deploy, scale, and manage a vast number of microservices. It has a declarative model that makes automating containers easier. The declarative model ensures that Kubernetes takes the appropriate action to fulfil the requirements based on the configuration files.

What are containerization use cases?

The following are some use cases of containerization.

- **Cloud migration**

  Cloud migration, or the lift-and-shift approach, is a software strategy that involves encapsulating legacy applications in containers and deploying them in a cloud computing environment. Organizations can modernize their applications without rewriting the entire software code.

- **Adoption of microservice architecture**

  Organizations seeking to build cloud applications with microservices require containerization technology. The microservice architecture is a software development approach that uses multiple, interdependent software components to deliver a functional application. Each microservice has a unique and specific function. A modern cloud application consists of multiple microservices. For example, a video streaming application might have microservices for data processing, user tracking, billing, and personalization. Containerization provides the software tool to pack microservices as deployable programs on different platforms.

- **IoT devices**

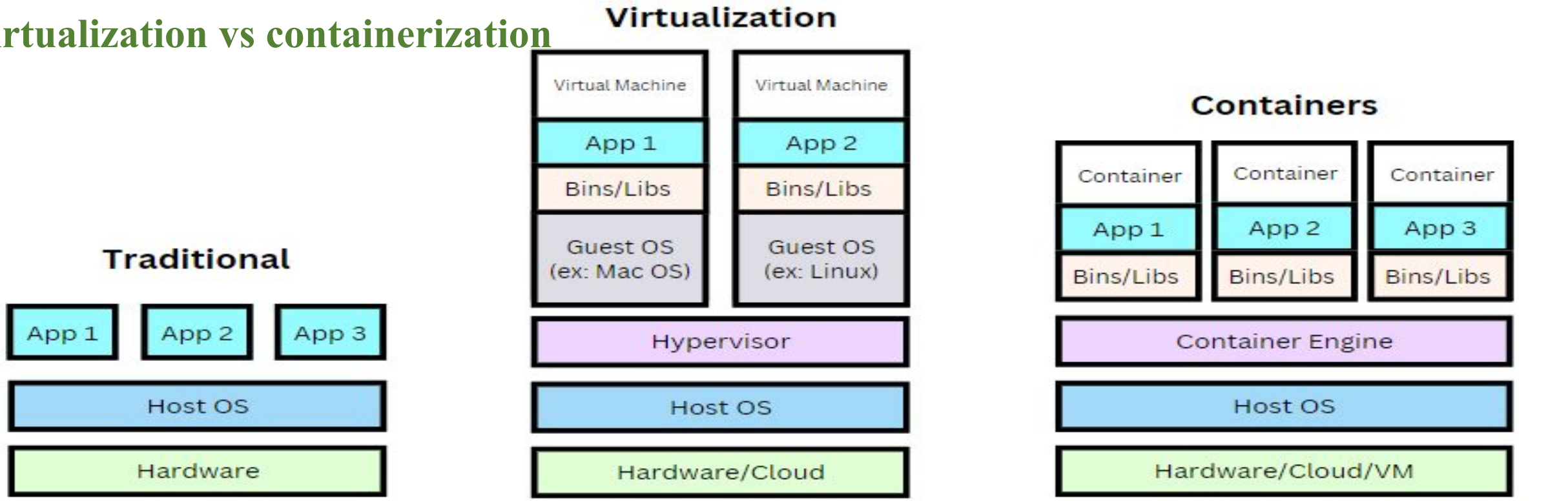
  Internet of Things (IoT) devices contain limited computing resources, making manual software updating a complex process. Containerization allows developers to deploy and update applications across IoT devices easily.

What is a virtual machine?
A virtual machine (VM) is a digital copy of the host machine's physical hardware and operating system. A host machine might have several VMs sharing its CPU, storage, and memory. A hypervisor, which is software that monitors VMs, allocates computing resources to all the VMs regardless of whether the applications use them.

Containerization compared to virtual machines

Containerization is a similar but improved concept of a VM. Instead of copying the hardware layer, containerization removes the operating system layer from the self-contained environment. This allows the application to run independently from the host operating system. Containerization prevents resource waste because applications are provided with the exact resources they need.

virtualization vs containerization



- No isolation of resources
- Overutilization by one app can crash the entire system
- Scaling issues
- Long Downtimes
- Expensive

- Better utilization of resources than traditional methods
- Applications are isolated

- OS images are heavy (GB) and have slow bootup process
- Applications are not portable
- Not Scalable
- Can get expensive

- Lightweight (Mbs) and quick bootup
- Containers are highly portable
- Not expensive. Highly scalable

- Applications are not fully isolated. Security is a concern.
- Management is critical as containers can be spun out at a rapid pace
- Skill shortage

Differences Between Virtualization and Containerization

At a technical level, both environments use similar properties while having different outcomes. Here are the primary differences between the two techniques.

1. Isolation

Virtualization results in a fully isolated OS and VM instance, while containerization isolates the host operating system machine and containers from one another. However, all containers are at risk if an attacker controls the host.

2. Different Operating Systems

Virtualization can host more than one complete operating system, each with its own kernel, whereas containerization runs all containers via user mode on one OS.

3. Guest Support

Virtualization allows for a range of operating systems to be used on the same server or machine. On the other hand, containerization is reliant on the host OS, meaning Linux containers cannot be run on Windows and vice-versa.

4. Deployment

Virtualization means each virtual machine has its own hypervisor. With containerization, either Docker is used to deploy an individual container, or Kubernetes is used to orchestrate (the planning or coordination of the elements of a situation to produce a desired effect) multiple containers across multiple systems.

5. Persistent Virtual Storage

Virtualization assigns a virtual hard disk (VHD) to each individual virtual machine, or a server message block (SMB) if shared storage is used across multiple servers. With containerization, the local hard disk is used for storage per node, with SMB for shared storage across multiple nodes.

6. Virtual Load Balancing

Virtualization means failover clusters are used to run VMs with load balancing support. Since containerization uses orchestration via Docker or Kubernetes to start and stop containers, it maximizes resource utilization. However, decommissioning for load balancing with containerization occurs when limits on available resources are reached.

7. Virtualized Networking

Virtualization uses virtual network adaptors (VNA) to facilitate networking, running through a master network interface card (NIC). With containerization, the VNA is split into multiple isolated views for lightweight network virtualization.

Resource Efficiency, Faster Time to Market

Edge computing also reduces energy consumption in networks, by reducing the total amount of data traversing the network. By running applications at the edge, data can be processed and stored nearer to the devices, rather than relying on data centres that are hundreds of miles away.

Edge computing environment provides an optimal solution for latency and lifetime of a device. Thus edge computing will be considered as the next step in developing applications in the IoT environment, maintaining the benefits. There are overheads that occur with computation at the edges and each of the edge nodes will directly or indirectly influence the battery life, i.e. energy consumption which is maintained by batteries. Apart from the regular constraints, edge computing needs to focus on privacy and data reduction methods, and is then compared with other computational systems.

https://www.techtarget.com/searchdatacenter/definition/edge-computing

https://iopscience.iop.org/book/edit/978-0-7503-4801-0/chapter/bk978-0-7503-4801-0ch16.epub

CONTENT OF ALL ABOVE SLIDES (From slide 15 -23) ARE TAKEN BY

FOG AND EDGE COMPUTING

BY:RAJKUMAR BUYYA


Thank
You