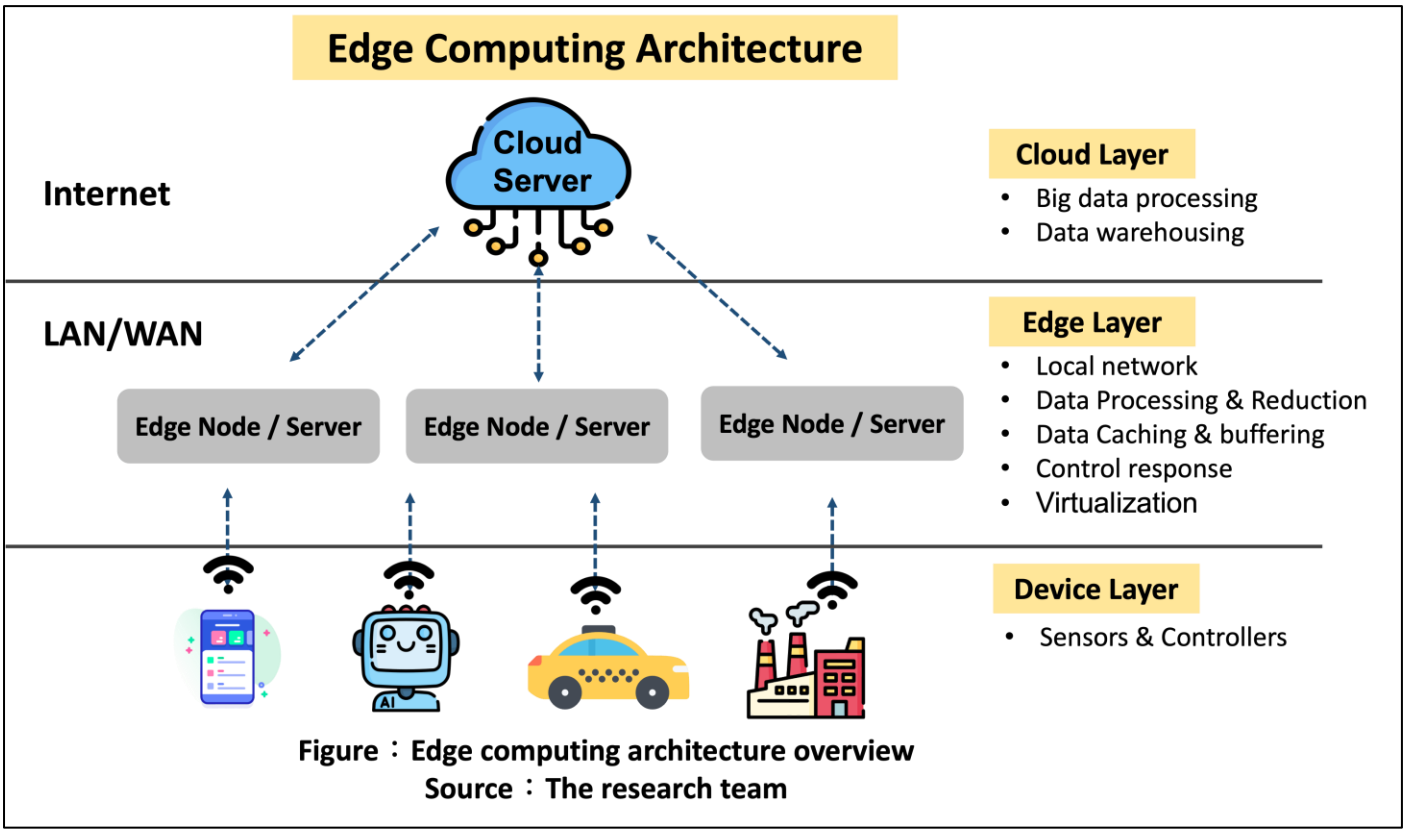
Chap 02

Edge Computing Infrastructure

Edge Computing Architecture Overview:

A typical edge computing architecture can be divided into three layers: The cloud layer, or the layer that is responsible for processing and storing all data; The edge layer, or the layer that handles the data processing near real time; And the device layer, or the layer that is in charge of detecting and performing simple processing. Let's demystify the three layers in the edge computing architecture in the following paragraphs.



Figure：Edge computing architecture overview
Source：The research team

Cloud Layer

Although edge computing was introduced to address network congestion and latency problems commonly found in cloud computing, cloud computing in fact still plays an important role in the entire edge computing architecture. We can say that cloud computing and edge computing complement one another. Through the edge layer described in the next section, the entire system determines if data needs to be processed in the cloud layer. If that is the case, edge servers will pass data to the cloud layer for complex processing. On the other hand, edge servers will also pass a part or critical data to the cloud layer for storage and comprehensive analysis. This also demonstrates the integration between both the cloud and edge layers.

Edge Layer

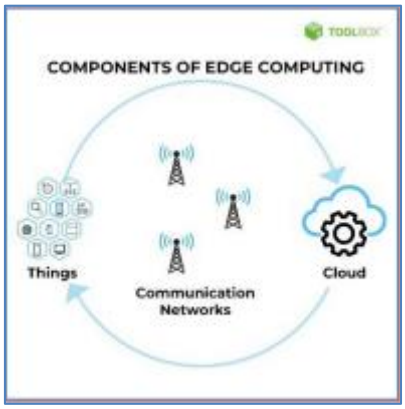
This layer mainly consists of edge servers, and when compared to the cloud layer, the edge layer contains edge servers that are larger in quantity and more vastly deployed. Therefore, through distributed edge computing, the edge layer can process data that is closer to the data source and address latency problems found in cloud computing. The edge layer can be considered the core in the entire edge computing architecture. After data from the device layer is analyzed and processed in the edge layer, data is transmitted to the cloud layer for subsequent processing and analysis. Data which cannot be processed in the edge layer can be sent to and analyzed in the cloud layer to ensure data integrity.

Device Layer

Amongst the three layers, the device layer contains the most devices. Ranging from devices that are as small as our mobile phones or computers to ones that are as large as buses and factories, these devices are all examples of components in the device layer. Through their sensors, devices in the device layer collect and capture data used to help products achieve the purposes they are designed for. Equipment in a hospital collecting vital signs of patients and autonomous vehicles capturing data of other nearby vehicles are all such examples. Although components in the cloud and edge layers possess better computing power, the devices in the device layer can still perform data analyses, processing and storage tasks which require negligible computing power, as well as process data closest to the data source in almost real-time.

Discuss the different components used in Edge architecture:

Edge architecture encompasses various components that work together to enable efficient and responsive processing of data at or near the edge of the network. These components contribute to the overall functionality, scalability, and effectiveness of edge computing solutions. Here are the different components used in edge architecture:



- Thing
- Communication
- Cloud

Edge Devices:

- These are the IoT devices, sensors, actuators, and other hardware units that generate and collect data at the edge of the network.
- Edge devices can range from simple sensors to more powerful devices with processing capabilities.
- They play a crucial role in data acquisition and initial data processing.

Edge Nodes:

- Edge nodes are computational units that perform data processing, analytics, and decision-making at the edge.
- They are responsible for executing tasks that require local processing, reducing the need to send data to centralized clouds.
- Edge nodes may include gateways, edge servers, routers, and more.

Edge Gateways:

- These devices bridge the gap between edge devices and the cloud by aggregating data from multiple devices and transmitting it to the cloud or higher-tier edge nodes.
- Edge gateways often include protocol translation, data preprocessing, and communication management capabilities.

Networking Components:

- Networking infrastructure, including switches, routers, and communication protocols, connects edge devices and nodes within the edge architecture.

Big Data Storage Solutions:

- Local data storage solutions, including databases and file systems, store data generated by edge devices for further processing or analysis.

Edge Clouds:

- Edge clouds are localized cloud infrastructures deployed closer to the edge devices.
- They provide additional processing and storage capacity and can host applications, services, and data repositories for edge-specific requirements.

Fog Nodes:

- Fog nodes are intermediate computing entities positioned between edge devices and the cloud.
- They handle more complex processing tasks compared to edge devices, improving efficiency and reducing the load on the central cloud.

Data Analytics Engines:

- These engines perform data analysis and extraction of insights from the data collected at the edge.
- Analytics engines are responsible for turning raw data into actionable information for real-time decision-making.

Management and Orchestration Tools:

- Tools for provisioning, monitoring, and managing edge resources.
- These tools ensure efficient utilization, updates, and maintenance of edge components.

Challenges for Edge application Development:

Developing applications for edge computing comes with a unique set of challenges due to the distributed and resource-constrained nature of edge devices. Addressing these challenges is crucial to ensure the success and efficiency of edge applications. Here are some of the key challenges for edge application development:

Resource Constraints: Edge devices often have limited computational power, memory, and storage. Developers need to optimize their applications to run efficiently on these constrained resources.

Data Management: Edge applications generate and process large volumes of data. Managing data locally, ensuring data consistency, and minimizing data transfer to the cloud are complex tasks.

Security: Edge devices are susceptible to physical tampering and may lack robust security features. Developers must implement strong security measures to protect data and applications.

Real-Time Processing: Many edge applications require real-time or low-latency processing. Achieving sub-millisecond response times can be demanding and necessitates careful design.

Scalability: Designing applications that can scale to accommodate a growing number of edge devices and users is challenging. Load balancing and efficient task distribution are key concerns.

Deployment and Management: Deploying and managing edge applications across a distributed network can be challenging. Tools for remote monitoring, updates, and troubleshooting are necessary.

Data Privacy and Compliance: Edge devices may process sensitive data, and compliance with data privacy regulations is critical. Developers must implement data encryption, access controls, and auditing.

Power Efficiency: Many edge devices run on battery power or have limited energy sources. Optimizing applications for power efficiency is essential for extended device lifetimes.

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

Setting up an edge computing environment for development involves selecting the right tools, libraries, and platforms to build and test edge applications. Python is a popular language for edge development due to its versatility and extensive libraries. Here are some development tools and Python libraries for setting up an edge computing environment:

Development Tools:

1. Docker: Docker is a containerization platform that allows you to package applications and their dependencies into containers, which can be deployed consistently across edge devices.
2. Kubernetes: Kubernetes is an open-source container orchestration platform that can manage the deployment, scaling, and management of containers in edge environments.
3. Visual Studio Code: A popular code editor with a range of extensions and plugins that support edge development and debugging.
4. PlatformIO: An open-source ecosystem for IoT development, PlatformIO supports various embedded platforms and frameworks, making it suitable for edge computing.
5. EdgeX Foundry: An open-source framework for building IoT edge computing platforms. It provides a collection of microservices and libraries for IoT applications.
6. Edge development kits: Depending on your target edge platform, use development kits provided by manufacturers or communities for specific hardware.
7. Version Control (e.g., Git): Version control tools are essential for managing and collaborating on edge application code.

Python Libraries for Edge Development:

1. Numpy: For numerical and scientific computing, often required in edge applications involving data analytics.
2. Pandas: A powerful data manipulation and analysis library for handling and processing data at the edge.
3. Scikit-Learn: For machine learning and data mining, especially useful for creating predictive models in edge applications.
4. TensorFlow and PyTorch: Popular deep learning libraries for training and deploying machine learning models on edge devices.

5. MQTT Libraries (e.g., Paho-MQTT): For implementing the MQTT protocol in Python, which is commonly used for IoT data communication.
6. ZeroMQ: A high-performance asynchronous messaging library that can be used for building distributed edge applications.
7. OpenCV: For computer vision and image processing tasks, often used in edge applications involving cameras and visual data.
8. Twisted: An event-driven networking engine for building network applications that may be required in edge scenarios.
9. Requests: A simple and intuitive HTTP library for sending HTTP requests to external services or cloud platforms.
10. Redis-Py: A Python client library for Redis, a popular in-memory data store that can be used for caching and real-time data processing at the edge.
11. FastAPI: A modern, fast (high-performance) web framework for building APIs that may be used in edge applications to expose services.
12. Zerynth: A Python development platform for edge and IoT devices, offering libraries and tools for hardware and cloud integrations.
13. CircuitPython: A beginner-friendly, open-source Python programming environment for microcontrollers that can be useful in edge IoT scenarios.

Edge computing platforms and frameworks:

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.

“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 edge computing framework provides a structured approach for designing, developing, and deploying applications at the edge of a network, enabling efficient processing of data closer to the data source.

Benefits of a framework:

- 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 Benefits of a framework.

Types of frameworks:

IoT Frameworks: IoT frameworks provide tools for developing and deploying Internet of Things (IoT) solutions. They often include device management, communication protocols, and data processing components.

Examples: AWS IoT, Azure IoT, EdgeX Foundry.

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.

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.

Testing Frameworks: Testing frameworks provide tools for automated testing of software. They assist in writing, executing, and managing unit tests, integration tests, and more.

Examples: JUnit (Java), pytest (Python), Jasmine (JavaScript).

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.

Game Development Frameworks: Game development frameworks offer tools and resources for creating video games. They include rendering engines, physics libraries, and game logic components.

Examples: Unity, Unreal Engine.

How to choose an IoT framework?

Define Your Requirements: Clearly outline your project's requirements, including the types of devices you'll be using, the scale of your deployment, data processing needs, security requirements, and integration with other systems.

Identify Use Cases: Determine the specific use cases and applications your IoT solution will address. Different frameworks excel in different use cases, such as industrial IoT, smart cities, healthcare, or consumer applications.

Evaluate Features: Research and compare the features offered by different IoT frameworks. Look for features such as device management, data processing, analytics, security, edge computing, and cloud integration.

Scalability and Performance: Consider the scalability and performance capabilities of the framework. Will it be able to handle the growing number of devices and data streams as your project expands?

Compatibility and Integration: Check if the framework is compatible with the devices and technologies you plan to use. Also, assess its ability to integrate with existing systems and tools.

Security: Security is paramount in IoT deployments. Ensure the framework offers robust security features, including encryption, authentication, and secure communication protocols.

Data Privacy and Compliance: If your project involves sensitive data, ensure the framework aligns with data privacy regulations relevant to your industry and region.

Cost Considerations: Evaluate the pricing model of the framework. Some frameworks offer open-source options, while others may have licensing fees based on usage.

Explain Edge computing platforms:

- Edge computing platforms are software frameworks and environments designed to enable the development, deployment, and management of applications at the edge of a network.
- These platforms provide a set of tools, services, and resources that help developers create efficient and responsive applications that process data locally on edge devices.
- An IoT platform is a platform, including software and hardware, used to manage internet-connected devices and the networks controlling them.
- 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.
- IoT platforms help reduce those operations' energy consumption, improve safety and security, and enable real-time analytics.

The 4 Types of IoT Platforms:

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.

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.

IoT Application Enablement Platforms

IoT Application Enablement Platforms create and deploy applications that attachment 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.

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.

Here are some notable edge computing platforms:

AWS IoT Greengrass:

- Platform by Amazon Web Services (AWS) that enables local computing, messaging, and data caching for edge devices.
- Allows devices to process data locally even when not connected to the cloud, improving responsiveness.

Azure IoT Edge:

- Microsoft Azure's offering that extends cloud capabilities to edge devices.
- Supports running containerized workloads, machine learning models, and Azure services at the edge.

Google Cloud IoT Edge:

- Google Cloud's platform for extending cloud capabilities to edge devices.
- Offers container support and integration with Google Cloud services.

EdgeX Foundry:

- An open-source, vendor-neutral project that provides a framework for building interoperable edge computing systems.
- Offers a collection of microservices for common edge computing tasks.

Balena:

- Platform for deploying and managing containerized applications on edge devices.
- Streamlines application deployment, updates, and management for edge scenarios.

OpenFog:

- Consortium focused on advancing fog and edge computing technologies.
- Provides reference architectures and frameworks for building scalable and secure edge solutions.

KubeEdge:

- Open-source project that extends Kubernetes to the edge, enabling containerized application deployment.
- Offers features like local data processing, device management, and real-time data synchronization.

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.

Types of Virtualizations:

- Hardware Virtualization.
- Operating system Virtualization.
- Server Virtualization.
- 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.

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

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.

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.
- VMs have longer boot times compared to containers, which can impact responsiveness in edge applications requiring quick startup.

⚜Containerization

Containerization is a technology that allows you to package an application and its dependencies, including libraries and runtime environment, into a single unit called a container. Containers are lightweight and portable, as they share the host OS kernel while maintaining isolation from each other

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.

Advantages of Containerization in Edge Computing:

1. Lightweight Isolation: Containers offer lightweight isolation, allowing multiple containers to share the same OS kernel without the overhead of separate OS instances.
2. Resource Efficiency: Containers consume fewer resources than VMs, making them suitable for edge environments with limited resources.
3. Fast Startup: Containers have faster startup times compared to VMs, making them suitable for latency-sensitive applications.
4. Portability: Containers are highly portable across different platforms, enabling consistent application deployment across edge devices.
5. Microservices Architecture: Containers are well-suited for microservices architecture, allowing applications to be broken down into smaller components for easier development and management.

Disadvantages of Containerization in Edge Computing:

1. Less Isolation: Containers provide weaker isolation compared to VMs, which might be a concern for certain edge applications requiring stronger isolation.
2. Compatibility: Containers share the host OS kernel, which can lead to compatibility issues if an application requires specific OS dependencies.

What are the different opportunities in Edge computing?

Edge computing presents a wide range of opportunities across various industries and use cases due to its ability to process data closer to the source, reduce latency, and enable real-time decision-making. Here are some different opportunities in edge computing

1. **IoT and Industrial Automation**: Edge computing is integral to IoT and industrial automation, allowing real-time monitoring and control of devices and machinery on the factory floor, improving efficiency and reducing downtime.
2. **Smart Cities**: Edge computing enables smart city applications such as smart traffic management, waste management, and energy optimization by processing data from sensors and devices deployed throughout the city.
3. **Healthcare**: In healthcare, edge computing supports remote patient monitoring, real-time analysis of medical data, and rapid response to critical patient conditions.
4. **Retail**: Edge computing enhances the retail experience by enabling personalized recommendations, inventory management, and real-time analysis of customer behaviour in stores.
5. **Transportation**: Edge computing facilitates autonomous vehicles, real-time traffic management, and predictive maintenance for transportation fleets.
6. **Energy Management:** Edge computing optimizes energy consumption in buildings and factories by processing data from energy sensors and devices, leading to cost savings and improved sustainability.
7. **Agriculture**: In precision agriculture, edge computing processes data from sensors and drones to optimize irrigation, crop monitoring, and pest control.
8. **Gaming**: Edge computing enhances cloud gaming by reducing latency, enabling real-time interactions, and delivering a smoother gaming experience.
9. **Smart Homes**: In smart homes, edge computing supports home automation, security systems, and energy management, processing data from smart devices within the home.

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 Que
Provider	Amazon Web Services	Microsoft	Google Cloud Platform
Launch Year	2006	2010	2011
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

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

Aspect	Virtualization	Containerization
Isolation	Heavyweight virtualization with full OS	Lightweight isolation at the application level
Resource Overhead	Higher resource consumption (CPU, memory)	Lower resource consumption
Boot Time	Longer boot time for virtual machines	Faster startup time
Performance	Slightly reduced performance due to overhead	Better performance due to lower overhead
Isolation Efficiency	Strong isolation between VMs	Less strong isolation between containers
Resource Utilization	May lead to underutilization of resources	Efficient utilization of resources
Scaling	Slower scaling due to larger footprint	Faster scaling due to smaller footprint

Aspect	Virtualization	Containerization
Dependency	VMs require guest OS for each instance	Containers share host OS kernel
Image Size	Larger image sizes	Smaller image sizes

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 Que
Provider	Amazon Web Services	Microsoft	Google Cloud Platform
Launch Year	2006	2010	2011
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
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

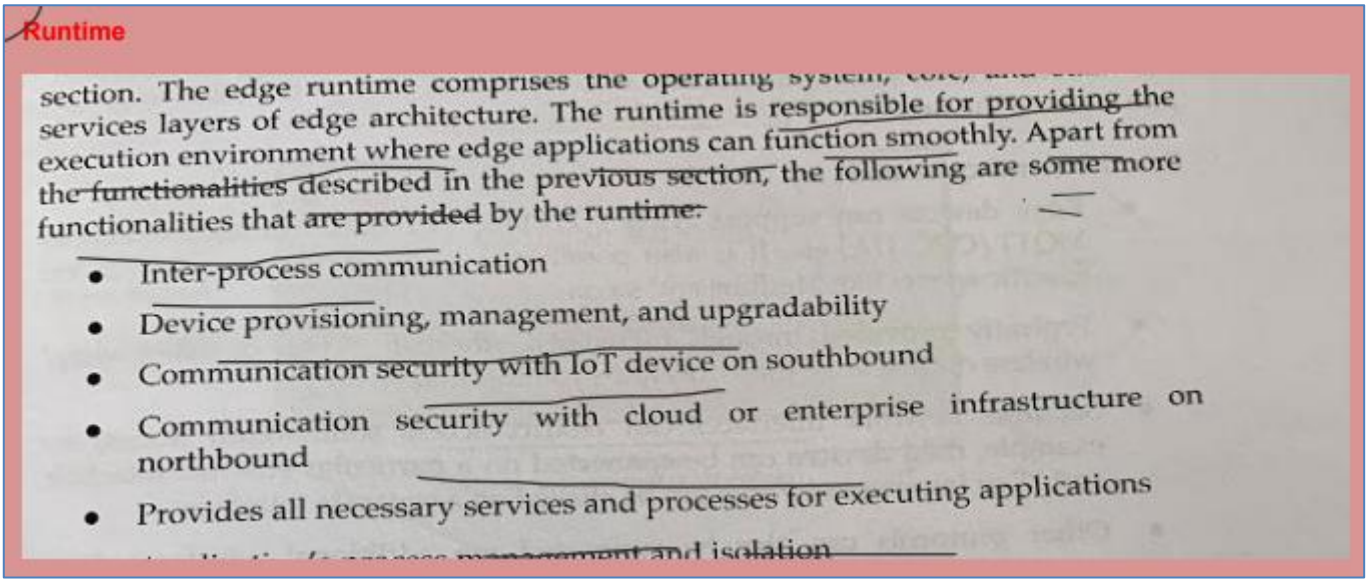
♻What is edge computing for efficient energy management:

Edge computing for efficient energy management refers to the use of edge computing technologies and principles to optimize the consumption and distribution of energy resources in various settings, such as buildings, factories, smart grids, and industrial processes. By processing energy-related data closer to the source and making real-time decisions at the edge, organizations can achieve better energy efficiency, reduce waste, and lower operational costs. Here's how edge computing contributes to efficient energy management:

1. **Real-time Data Processing:** Edge devices and sensors collect real-time data on energy consumption, generation, and distribution. Local processing at the edge enables quick analysis of energy data, allowing immediate response to changing conditions**.**
2. **Predictive Analytics**: Edge computing can employ machine learning algorithms to predict energy usage patterns, optimizing energy distribution and load balancing.
3. **Demand Response**: Edge computing enables rapid response to changes in energy demand and supply, allowing for automatic load shedding, peak shaving, and grid stability. Remote
4. **Monitoring and Control**: Edge devices monitor energy-intensive systems remotely and can automatically adjust settings to optimize efficiency. This is particularly useful in industries like manufacturing and agriculture.
5. **Lighting Control:** Edge computing can optimize lighting systems by adjusting brightness levels based on natural light conditions and occupancy.
6. **Load Balancing**: Edge devices can analyze energy consumption patterns and distribute loads efficiently to avoid overloading the grid during peak hours.
7. **Energy Storage Management**: Edge computing can manage energy storage systems like batteries, deciding when to charge or discharge based on demand and cost factors.

⚜Discuss about critical elements for Edge architecture:

Edge Devices: The foundation of edge architecture, these devices include sensors, IoT devices, gateways, and edge servers. They collect and generate data from the physical environment and act as the first point of data processing. **Connectivity and Communication:** A robust communication framework is vital for seamless data exchange between edge devices, the edge platform, and the cloud. Utilizes various protocols (e.g., MQTT, CoAP) and networking technologies to ensure reliable data exchange**.**


Runtime

section. The edge runtime comprises the operating system, core, and ... services layers of edge architecture. The runtime is responsible for providing the execution environment where edge applications can function smoothly. Apart from the functionalities described in the previous section, the following are some more functionalities that are provided by the runtime:

- Inter-process communication
- Device provisioning, management, and upgradability
- Communication security with IoT device on southbound
- Communication security with cloud or enterprise infrastructure on northbound
- Provides all necessary services and processes for executing applications

...management and isolation

Data Processing and Analytics(monitoring): Efficient data processing and real-time analytics capabilities are essential at the edge. This involves data transformation, filtering, aggregation, and analysis to derive valuable insights close to the data source.

Security and Privacy: Prioritizes security to protect devices, data, and communication channels from cyber threats. Incorporates encryption, authentication, access controls, and intrusion detection mechanisms.

Orchestration and Management: Orchestration tools manage application deployment, scaling, and lifecycle across diverse edge devices. Device management ensures remote monitoring, updates, diagnostics, and maintenance.