

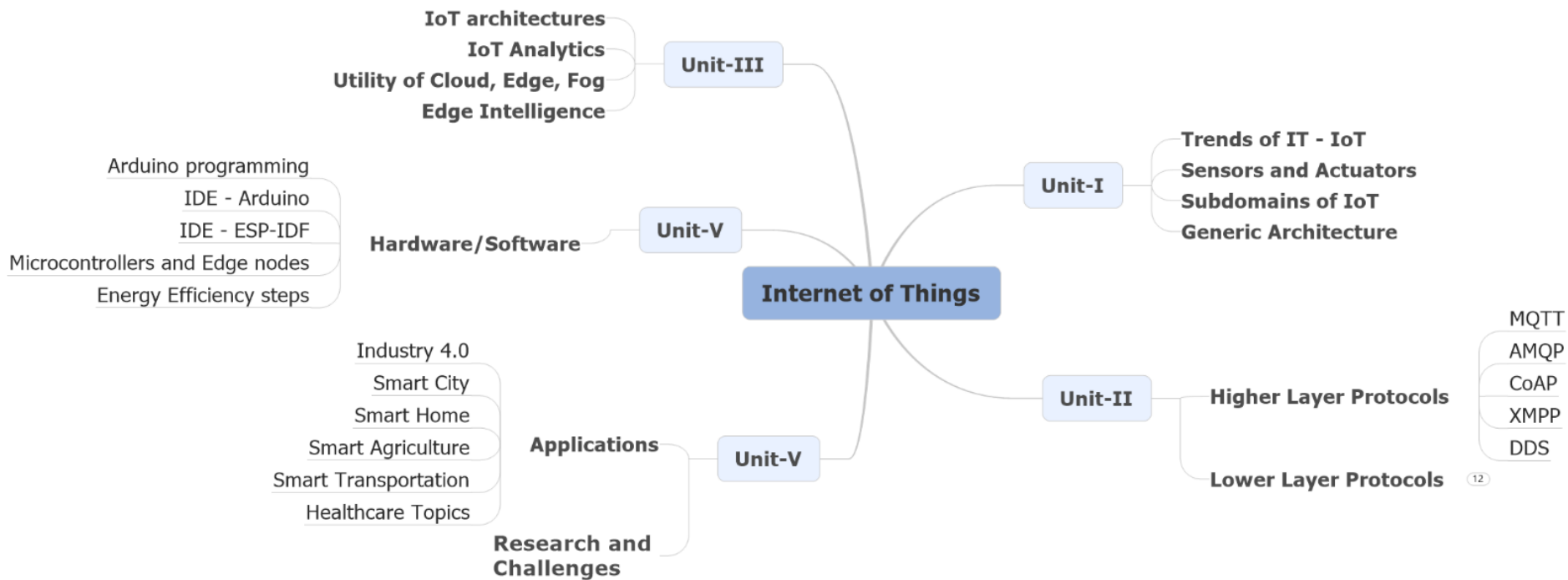
# Selected Topics in Computer Architecture, Computer Networks, and Distributed Systems (Internet of Things) (IN3450)

---

Dr. Shajulin Benedict  
[shajulin@iiitkottayam.ac.in](mailto:shajulin@iiitkottayam.ac.in)  
[shajulinbenedict@mytum.de](mailto:shajulinbenedict@mytum.de)

Prof. Dr. Michael Gerndt  
[gerndt@in.tum.de](mailto:gerndt@in.tum.de)

# Syllabus



# IoT Architectures

---

- In general, an architecture is an organization of components that bring forth the objectives set for an application.
- It is a programmers' view of an IoT system or ecosystem or modules.
- The architecture, in IoT, is designed in several approaches...

# IoT Architectures

---

Generic Architecture

Layered Architecture

Functional Modular IoT Architecture

Enterprise Information Architecture

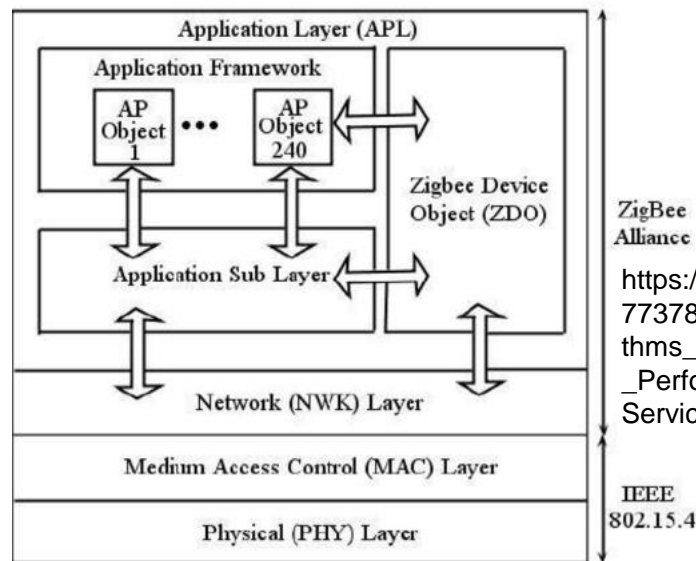
Industrial IoT Architecture

Scalable IoT Architecture

Name-specific IoT architectures

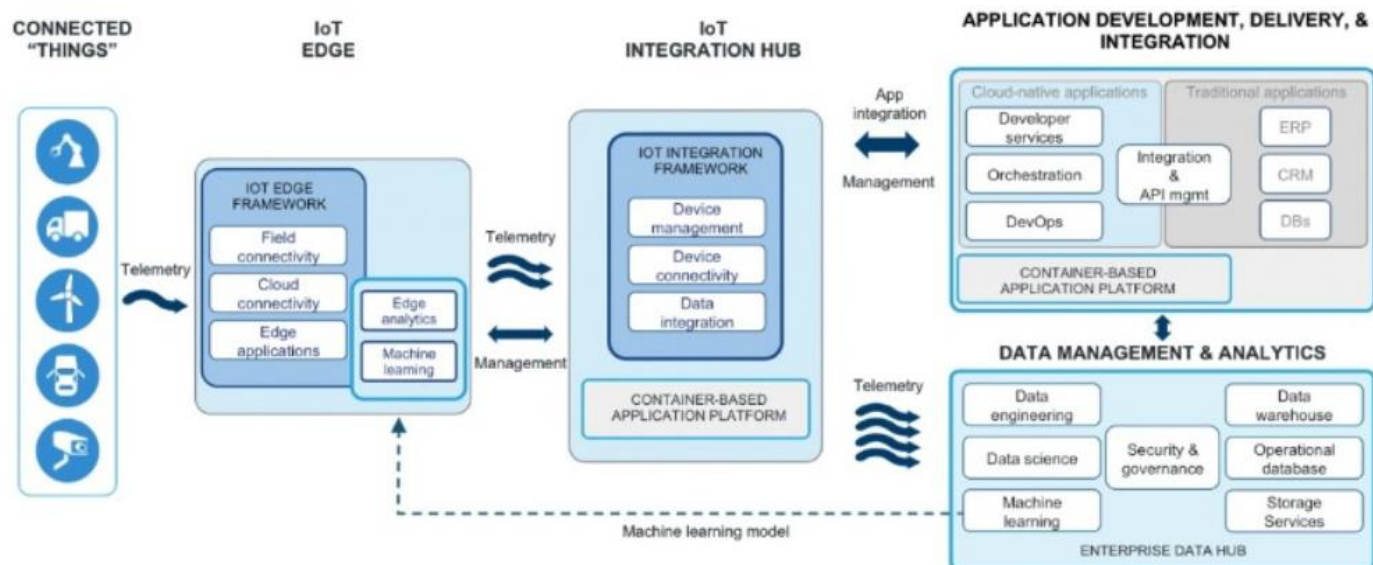
# Layered Architecture

- Here, an IoT architecture is defined in terms of networking layers.
- For e.g., in the zigbee context, a layered architecture is represented as follows



# Functional Modular Architecture

- In this type of architecture, an IoT application is represented as functional modules.
- Mostly, loosely coupled modules that provide opportunities to extend further...
- They are often designed based on open standards.



[www.eurotech.com/en/news/eurotech-red-hat-cloudera-iot-architecture](http://www.eurotech.com/en/news/eurotech-red-hat-cloudera-iot-architecture)

[www.sbenedictglobal.com](http://www.sbenedictglobal.com)

# Enterprise Information Architecture

---

- This architecture focuses on information derived from IoT systems.
- Usually enterprise architects apply agile methods that improve organization's vision.
- Information architects aim to find the best possible information to withstand in the market (business).
  - i.e., we need to frame decisions based on data rather than from opinions.
- Typical successful business-oriented architectures have included
  - Hadoop – for BigData;
  - IoT analytics – for machine intelligence

# Use-cases

---

- Agriculture

- Cost of farm production and optimization
- Yield analysis
- Agricultural goods commodity pricing and trading
- Hadoop/IoT
  - Analysis and optimization of plowing patterns, fertilization, readiness for harvesting, moisture content

- Automotive manufacturing

- Cost and quality of supply chain analysis
- Warranty analysis
- Sales and marketing analysis,
- Human capital management
- Hadoop/IoT
  - Analysis of customer sentiment and analysis of connected vehicles
  - Need for service and service scheduling
  - Driving history



# Use-cases

---

- Banking

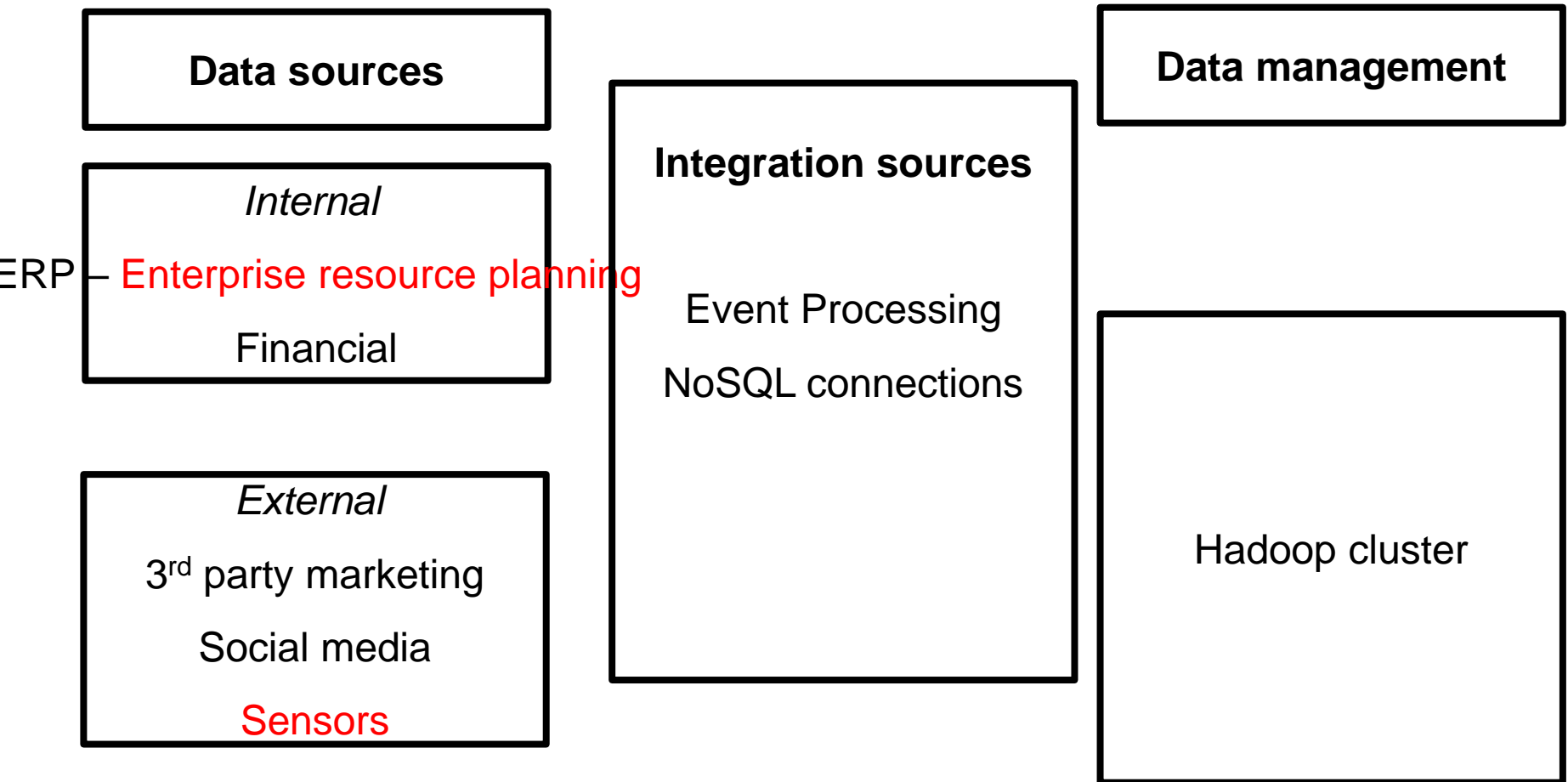
- Single view of customers across financial offering channels
- Financial analysis
- Fraud detection
- Credit worthiness
- Real estate management and optimization

- Similarly,

- Media and entertainment
- Retail
- Oil and gas
- Industrial manufacturing
- Transportation and logistics

## E.g., Enterprise Information Architecture

---



Source: Robert Stackowiak, Art Licht, Venu Mantha and Louis Nagode, “BIG data and the internet of things”, apress publishers, 2015.

# Industrial IoT Architecture

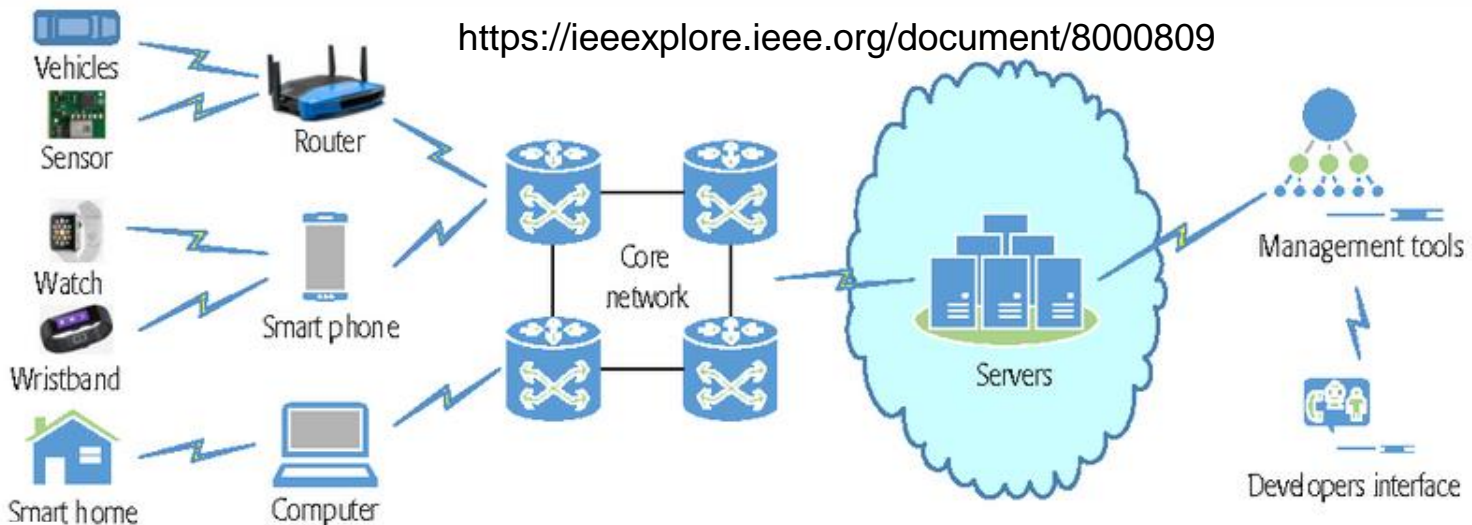
- This architecture focuses on industries/machines.
- Important characteristics of these architectures include
  - Automation – both locally and globally.
  - Support for robust user-interface that supports industrial workers.
  - Support for standards relevant to Industry 4.0.



<https://www.iiot-world.com/industrial-iiot/connected-industry/iiic-industrial-iiot-reference-architecture/>

# Scalable IoT Architecture

- These architectures emphasize on the scalability metric of IoT applications.
- In this lieu, the architectures incorporate scalability-based compute components in a hierarchical fashion.
- They also include management tools to look into the failures, if any.



# Name-Specific IoT Architectures

---

- Here, IoT architectures are designed based on some specific names.
- Many commercial IoT platforms/solutions fall under this category.
- Example:
  - Oracle IoT architecture,
  - EasyIoT architecture,
  - IoTAlliance architecture,
  - AWS IoT architecture

# Others

---

- Reference IoT architectures
  - End-to-end IoT architectures
  - Security-Oriented architectures
- 
- And, so forth (limitless – as research grows!)

# Architecture and platforms – How to choose?

---

## Connectivity

- How well the connection is?

## Method of connectivity.

- Cellular, wifi, ...

## Type of service

- End to end,
- Warranty-based,

## Geographic coverage

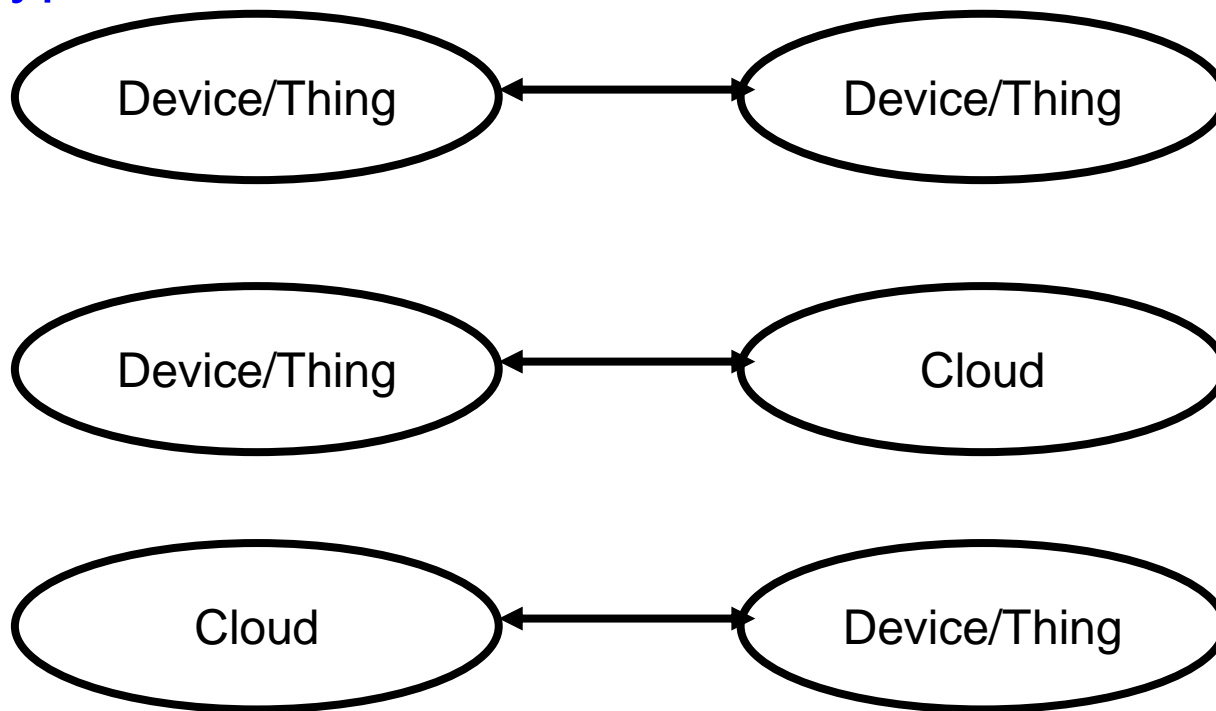
## Security/privacy

## Device management

# AWS IoT

---

- **AWS IoT** provides connectivity, analytics, and manageability (as pay-as-you-go model) of devices.
- It is designed to know the state of things/devices...
- 3 Types of connections...





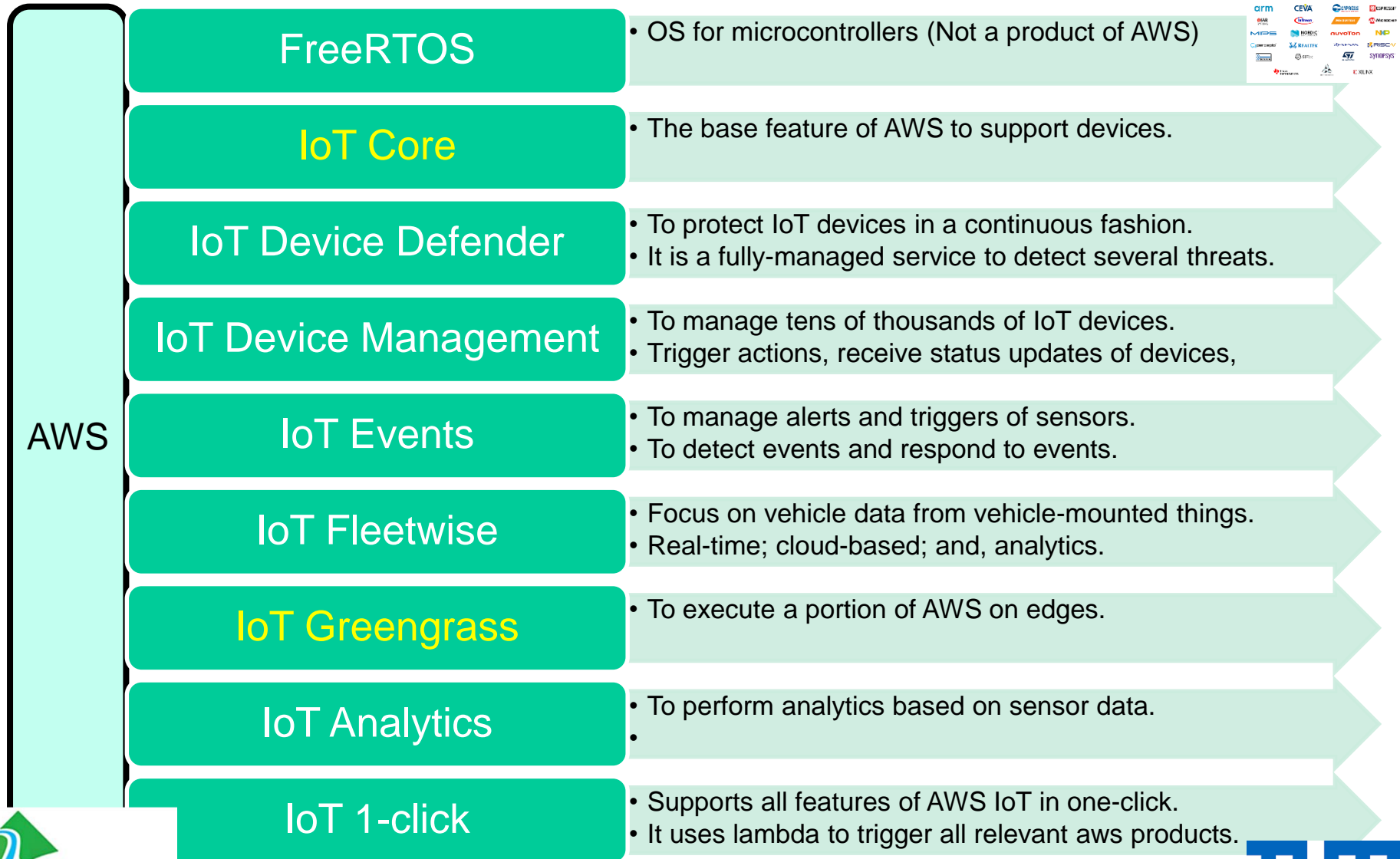
# AWS – IoT

---

- AWS IoT provides secure and bi-directional communication between IoT-enabled connected devices.
- Example devices include
  - Sensors/actuators,
  - embedded micro-controllers,
  - smart appliances and
  - AWS Cloud.

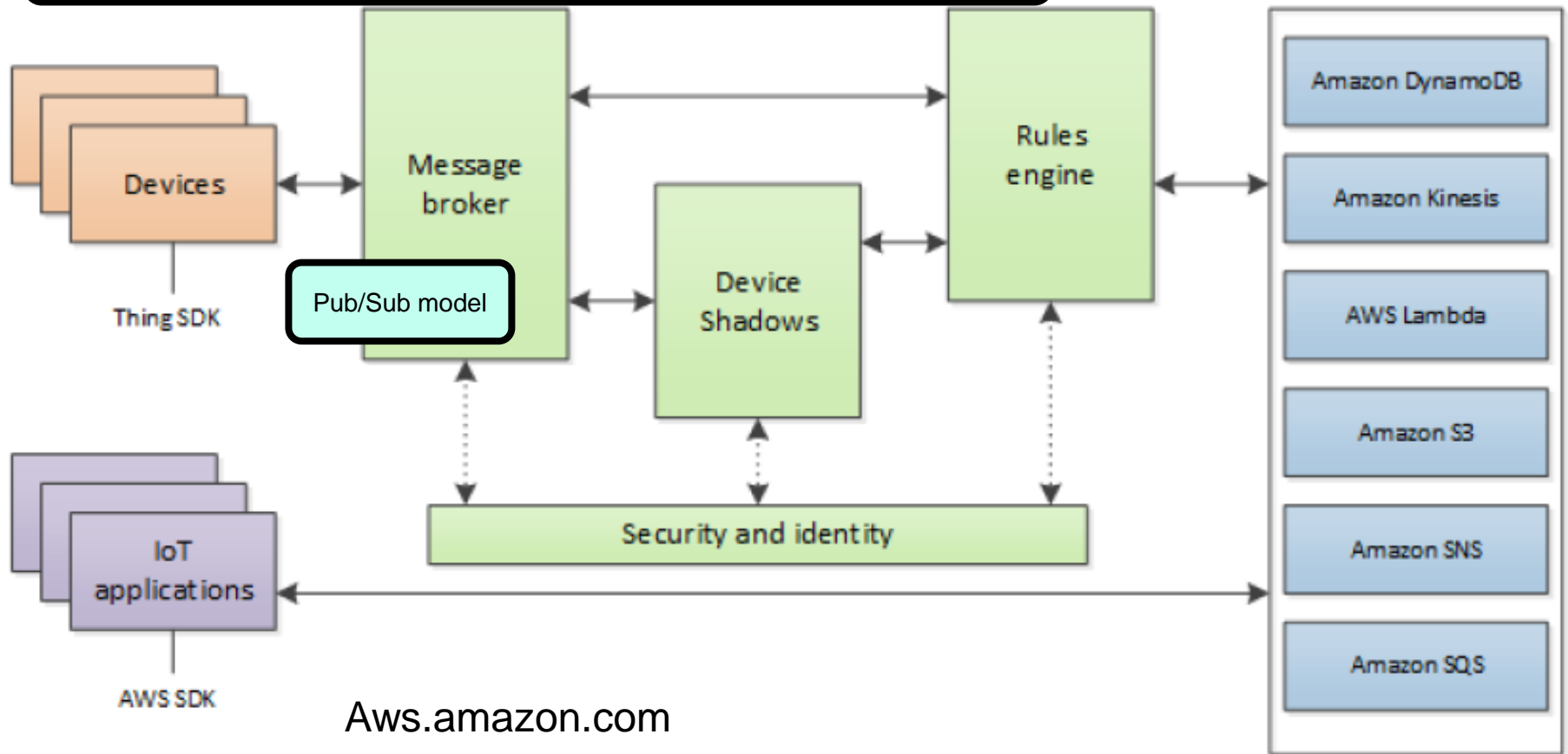
(Discussions are based on the AWS-IoT official site)

# AWS IoT Products



# AWS-IoT Core -- Architecture

Billions of IoT devices; Route trillions of messages



# AWS IoT Architecture

---

- The state of each device connected to AWS IoT is stored in a Device Shadow.
- The Device Shadow is an AWS service.
- The service enables
  - devices to communicate with applications and
  - applications to communicate with devices.
  - (even when the devices are not available).
- Rules can be created that define one or more actions to perform based on the data in a message.
  - E.g., augment or filter sensor data.
  - E.g., Send notifications to users or apps.
- We can create, delete, and modify rules of AWS IoT.

# AWS IoT Architecture

---

- AWS IoT makes connections with the other AWS products, such as, DynamoDB, Lambda, EC2, and so forth.
- We could create table or invoke a Lambda function.
- Rules and policies are utilized in AWS architecture to filter messages and perform actions.
- Based on the incoming messages from sensors, the rules engine triggers the action using the selected properties.
- Rules also contain an IAM role that grants AWS IoT permission to the AWS resources used to perform the action.

# Device/Apps Accessing Methods

---

## AWS CLI

- It utilizes AWS commands on machines to connect to devices or applications.

## AWS IoT API

- It helps to build IoT applications using HTTP or HTTPS requests.
- These APIs create and manage things, certificates, rules, and policies.

## AWS SDKs

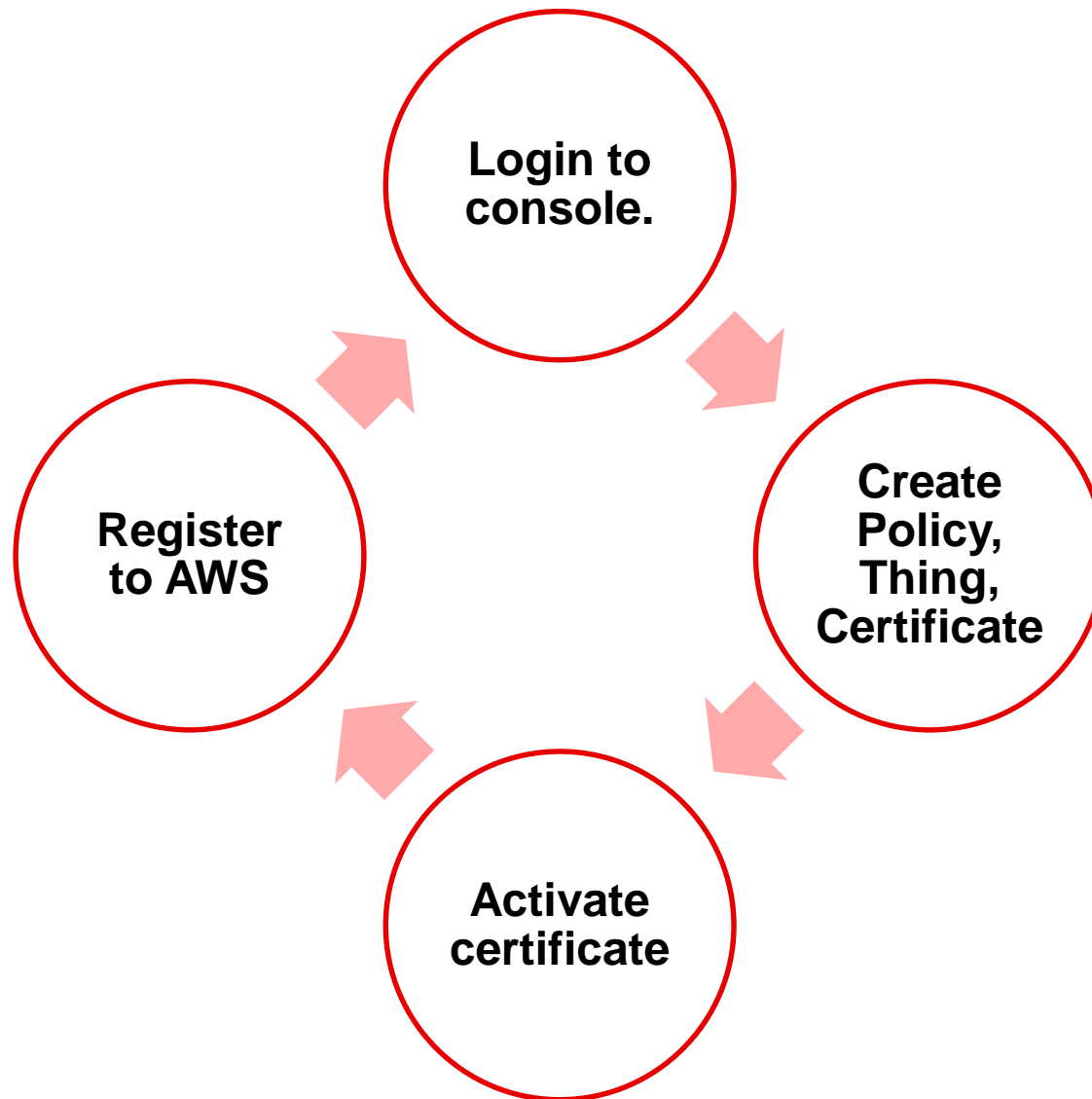
- It helps to build IoT applications using language-specific APIs.
- These languages uses HTTP APIs.

## AWS IoT Device SDKs

- It helps to build applications that run on devices.
- It can send messages to and receive messages from AWS IoT.

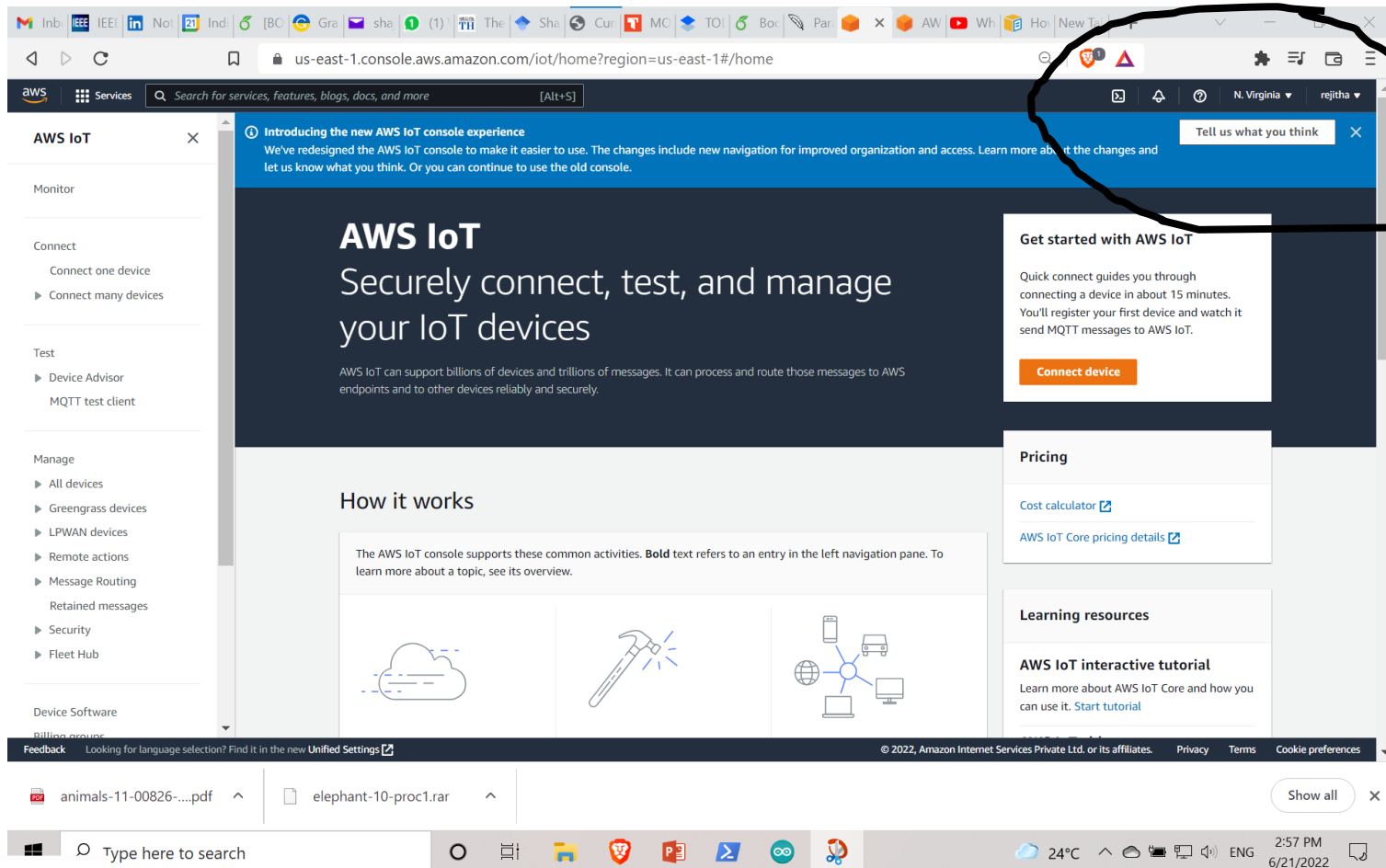
# Register ESP-32 on AWS IoT

---



# Register ESP32 on AWS IoT

- Step 1 & 2: Register and login to AWS IoT console.





# Register ESP-32 on AWS IoT

---

- Create Policies.

- On the Policies page of AWS, choose Create a Policy.
- It is a JSON document. It contains three components:
  - Effect – to specify whether action is allowed or not.
  - Action – to specify the action to allow or deny a device.
  - Resource – to specify the resource under consideration.
- For e.g., while registering ESP-32 on AWS IOT core...
  - On the Create a Policy page:
    - a. Enter a name for the policy (for example, esp32-policy).
    - b. For Action, enter iot:\*. (connecting any device).
    - c. For Resource ARN, (Amazon Resource Names), enter \*.
    - d. Under Effect, choose Allow, and then choose Create.

# Policies -- Screenshots

The screenshot displays the AWS IoT console interface for the 'esp-policy'. The left sidebar shows navigation options like 'Test', 'Manage', and 'Security'. The main content area shows the policy details, including the Policy ARN, Active version (3), Created date, and Last updated date. Below this, the 'Active version: 3' section shows a table of policy actions and resources. The 'All versions (3)' section is also visible at the bottom.

**Policy Details:**

Policy ARN	Active version	Created	Last updated
arn:aws:iot:us-east-1:123456789012:policy/esp-policy	3	June 07, 2022, 21:18:30 (UTC+0200)	June 07, 2022, 21:18:30 (UTC+0200)

**Active version: 3**

Policy effect	Policy action	Policy resource
Allow	iot:Connect	
Allow	iot:Publish	
Allow	iot:Subscribe	
Allow	iot:Receive	

**All versions (3)**

The active and previous versions of this policy. Only one version can be active. A policy can have no more than 5 versions. To update a policy with 5 versions, you must first delete one.

Buttons: Delete, Set as active, Edit version, View JSON

# Create Policy

esp-policy

A policy name is an alphanumeric string that can also contain period (.), comma (,), hyphen(-), underscore (\_), plus sign (+), equal sign (=), and at sign (@) characters, but no spaces.

Tags - optional

Policy statements | Policy examples

**Policy document** [Info](#)

An AWS IoT policy contains one or more policy statements. Each policy statement contains actions, resources, and an effect that grants or denies the actions by the resources.

Policy effect	Policy action	Policy resource	
Allow	iot:Connect	arn:aws:iot:us-east-1:123456789012:device/*	Remove
Allow	iot:Publish	arn:aws:iot:us-east-1:123456789012:device/*	Remove
Allow	iot:Subscribe	arn:aws:iot:us-east-1:123456789012:device/*	Remove
Allow	iot:Receive	arn:aws:iot:us-east-1:123456789012:device/*	Remove

[Add new statement](#)

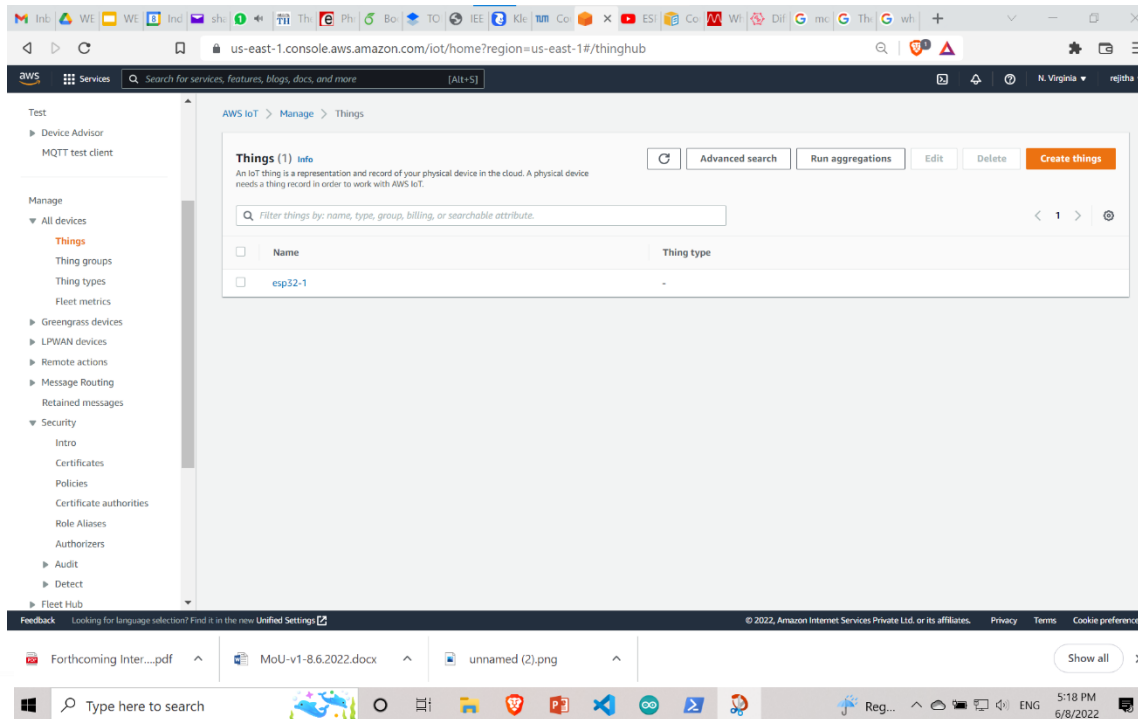
[Builder](#) [JSON](#)

[Cancel](#) [Create](#)

# Register ESP32 on AWS IoT

- Create Thing

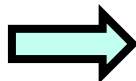
- On AWS console's **navigation pane**, choose Manage, and then choose Things. Choose Create.
- On the Creating AWS **IoT things page**, choose Create a single thing.
- On the Add your device to the **device registry page**, enter esp32-1, and then choose Next.



# NOTES: Go to objects/things

In Europe, devices are named as OBJECTS

In the US, devices are named as things



# Create/Activate Certificates

---

- Create Certificate

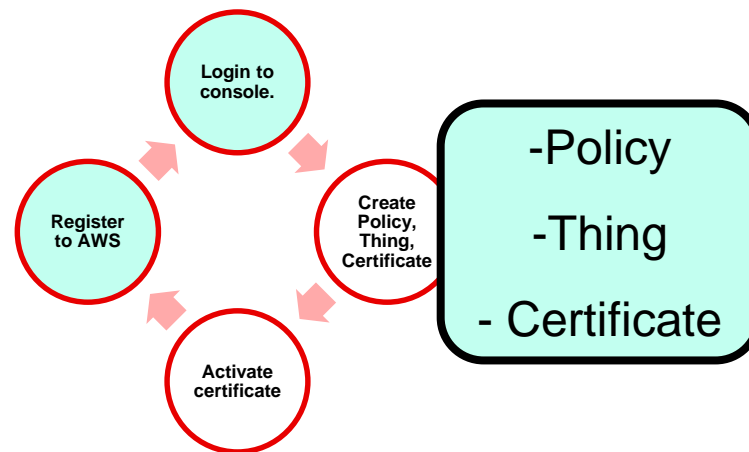
- In the next step, we need to create certificates and download them.
- This step is important to create secure connections between devices and AWS services.

- Activate Certificates

- Choose *Activate* button to activate the certificate.

- Attach Certificates to policies.

- Next, attach the policy created earlier with the thing.



# Snapshots: Create Certificate

The screenshot shows the AWS IoT console interface in a web browser. The browser's address bar displays the URL: `https://eu-west-1.console.aws.amazon.com/iot/home?region=eu-west-1#/create/single-provision`. The console header includes the AWS logo, a 'Services' menu, a search bar, and the user's name 'rejitha'. The breadcrumb navigation path is: **AWS IoT** > **Administer** > **Objects** > **Creating Objects** > **Create a single object**.

The left sidebar shows the progress of the setup steps:

- Step 1: Specify object properties
- Step 2 - optional: **Configure the device certificate**
- Step 3 - optional: Attach policies to certificate

The main content area is titled **Configure the device certificate - optional** with a link to [about](#). Below the title, a paragraph explains: "A device needs a certificate to connect to AWS IoT. You can choose to enroll a certificate for your device now, or you can create and enroll a certificate for your device at a later time. Your device cannot connect to AWS IoT until it has an active certificate with a corresponding policy."

The **Device Certificate** section contains four radio button options:

- ☒ **Automatic creation of a new certificate (recommended)**  
Generate a certificate, public key, and private key using the AWS IoT CA.
- ☐ **Use my certificate**  
Use a certificate that is signed by your own CA.
- ☐ **Upload CSR**  
Register your CA and use your own certificates on one or more devices.
- ☐ **Skip creating a certificate at this time**

The footer of the console includes a 'Feedback' link, a language selection prompt, copyright information for 2022, and links for 'Privacy', 'Conditions', and 'Cookie settings'. The Windows taskbar at the bottom shows the search bar, system tray icons, and the date/time (8:05 PM, 6/7/2022).

# Snapshots: Create Thing and Download Certificates

The screenshot shows the AWS IoT console interface. A modal window titled "Downloading certificates and keys" is open. The modal contains the following sections:

- Download certificate and key files to install on your device so that it can connect to AWS.**
- Device Certificate**  
You can activate the certificate now or later. The certificate must be activated before a device can use it to connect to AWS IoT.  
There is a text input field for the "Device Certificate" name, which is highlighted with a red box. To its right are buttons for "Disable Certificate" and "Download".
- Key files**  
Keys are unique to this certificate and cannot be downloaded after you leave this page. Download them now and save them in a safe place.  
Below this is a warning message: "This is the only time you can download the key files for this certificate." with a warning icon.
- Public key file**  
There is a text input field for the "Public key file" name, which is highlighted with a red box. To its right is a "Download" button.
- Private key file**  
There is a text input field for the "Private key file" name, which is highlighted with a red box. To its right is a "Download" button.

The background shows the AWS IoT console navigation pane with "Step 1: Specify object properties", "Step 2 - optional: Configure the device certificate", and "Step 3 - optional: Attach policies to certificate". The browser address bar shows the URL: <https://eu-west-1.console.aws.amazon.com/iot/home?region=eu-west-1#/create/single-provision>.



# Utilize things using aws commands

---

AWS commands could be utilized to connect with things and AWS services:

i) **Create thing**

```
aws iot create-thing --thing-name "esp-32" --attribute-payload  
"{\"attributes\":{\"xxxxxx\":\"xx\", \"xxxxxx\":\"xx\"}}"
```

ii) **List things**

```
aws iot list-things
```

iii) **Search things**

```
aws iot describe-thing --thing-name "esp-32"
```

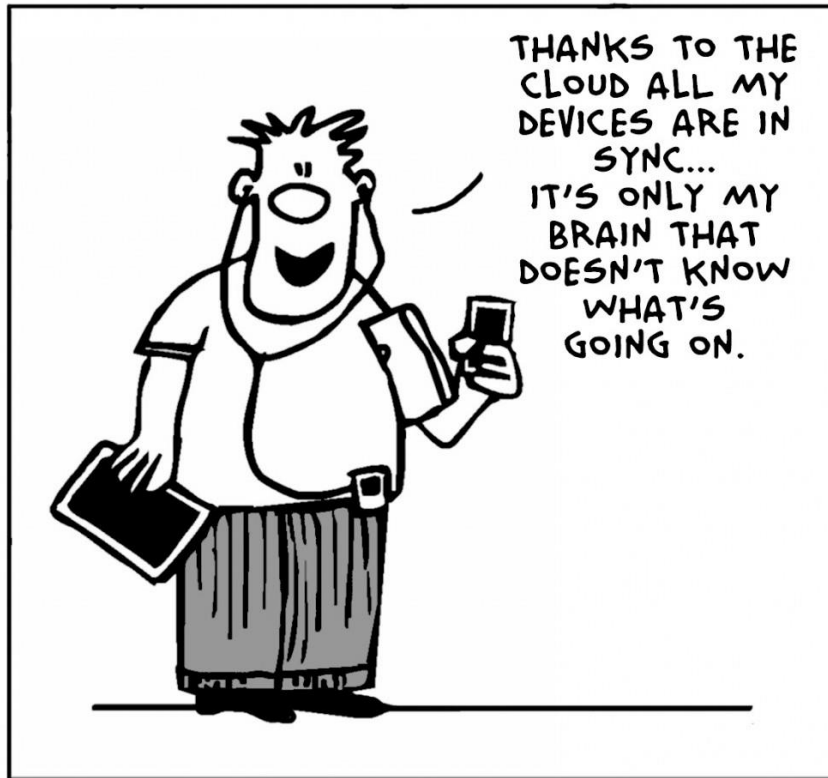
iv) **Update thing**

```
aws iot update-thing --thing-name "esp-32" --attribute-payload  
"{\"attributes\":{\"xxxxxx\":\"xx\", \"xxxxxx\":\"xx\"}}"
```

---

# Utility of Cloud, Fog, Edge

# Cloud Computing - Introduction



<https://s-media-cache-ak0.pinimg.com>

# What is Cloud Computing? - Definition

---

- According to NIST

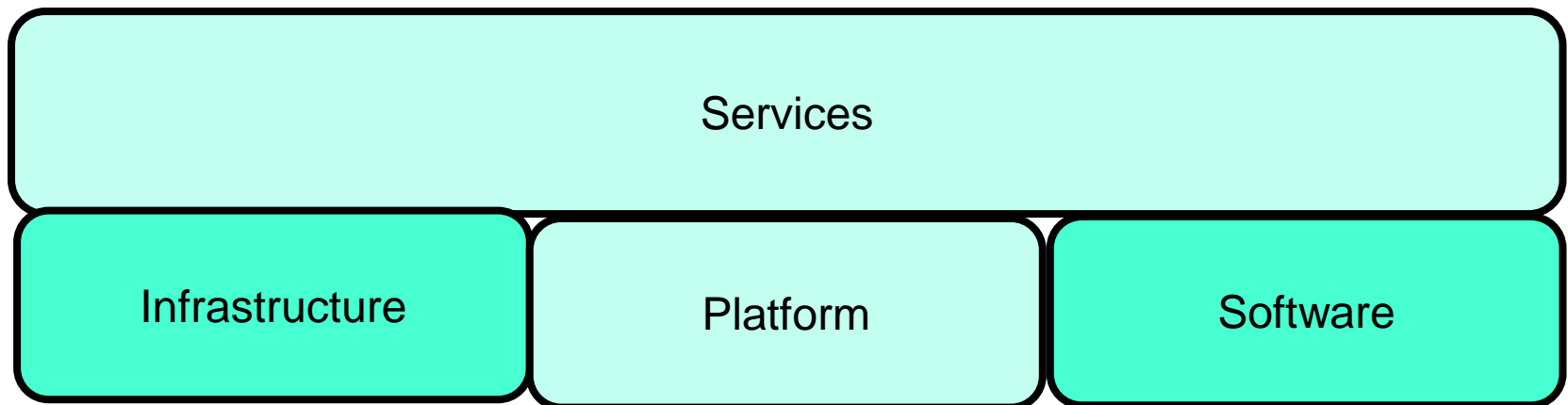
- *Cloud Computing is an approach to computing pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.*

- According to Buyya

- *Cloud is a parallel and distributed computing system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one established through **negotiation** between the service providers and consumers.*

# Cloud Computing and Services

---



# Cloud Deployment Models

---



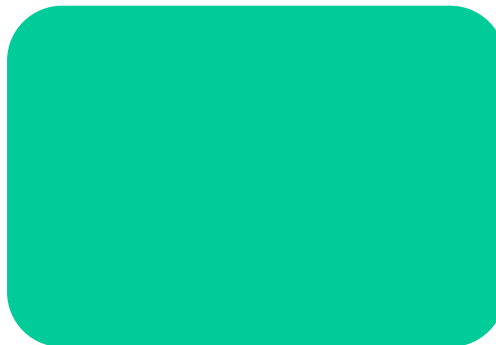
Public Cloud



Private Cloud



Community  
Cloud



Hybrid Cloud

# Virtualization

---

- Virtualized resources are often utilized in IoT solutions.
- What is meant by virtualization?
  - A computer architecture.
- Why and How -- virtualization?
  - Improve the usage <5%
  - VMM (Hypervisor)
  - Business tactics!!!

# Virtualization Techniques

---

- Full virtualization
- Para virtualization
- Hypervisors
  - Hosted hypervisor
  - Bare-metal hypervisors
- IoT Network virtualization



# Docker

---

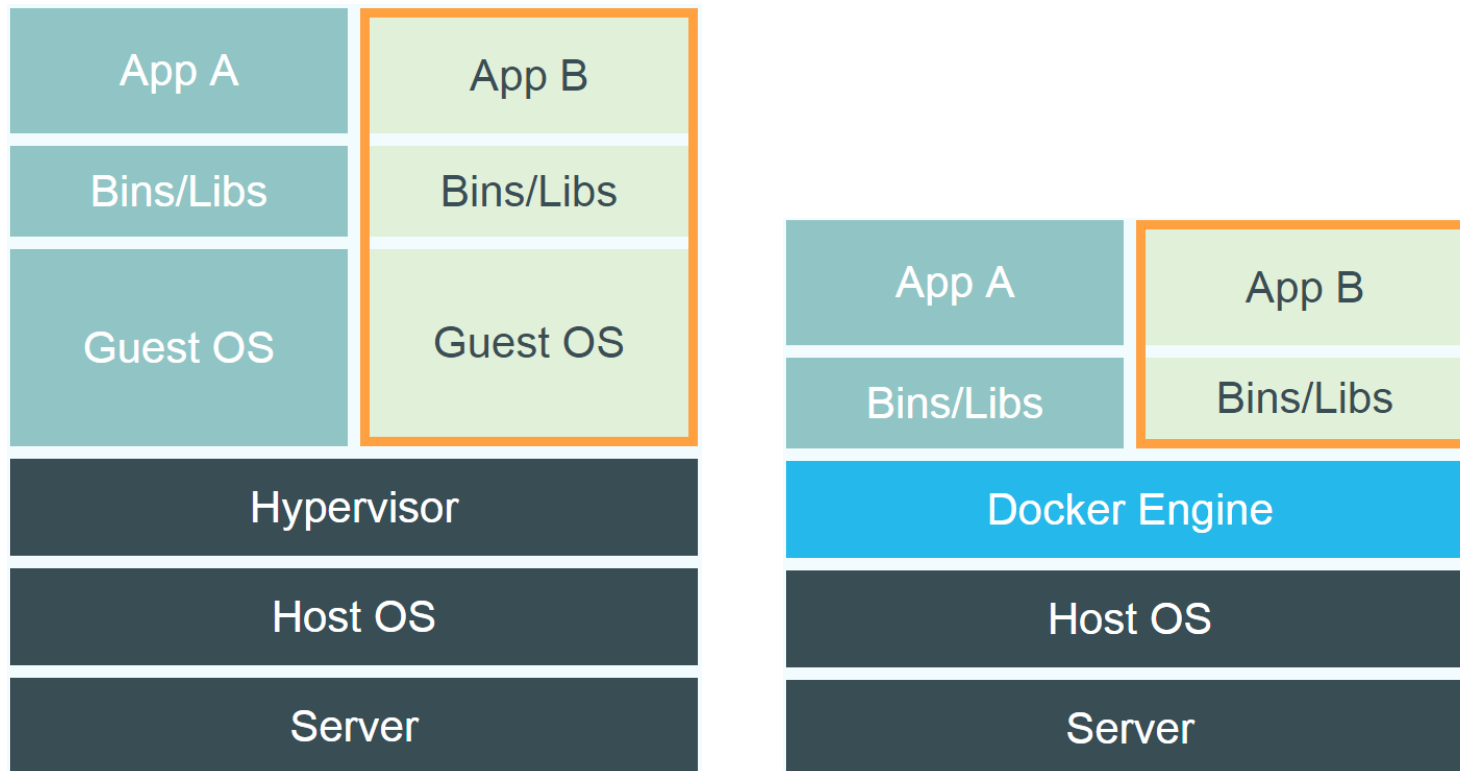
- Docker utilizes the container technology (cgroups and namespaces)
  - It easily **ports** containers
  - It **replicates** containers across environments.
- Thus, it reduces the time between writing code and producing them.
- It removes unnecessary configurational hurdles of applications.
- *Since 2014...*



# Docker vs. Virtual Machines

---

- Problems with VM – size, memory, integration

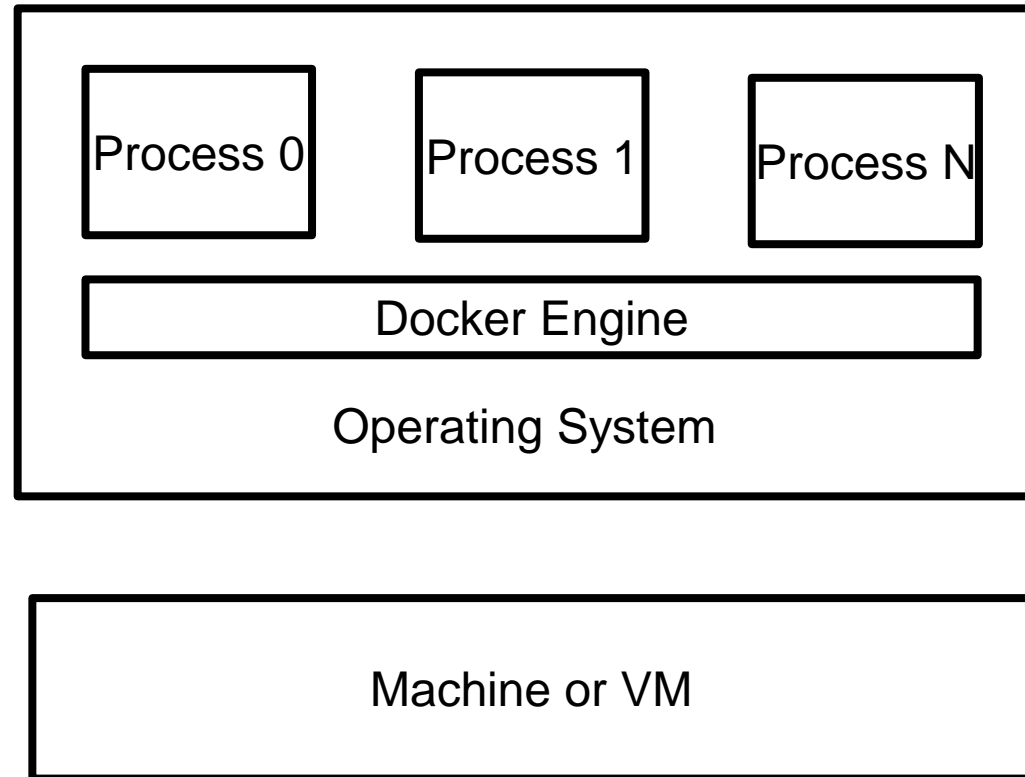


It utilizes union filesystem –

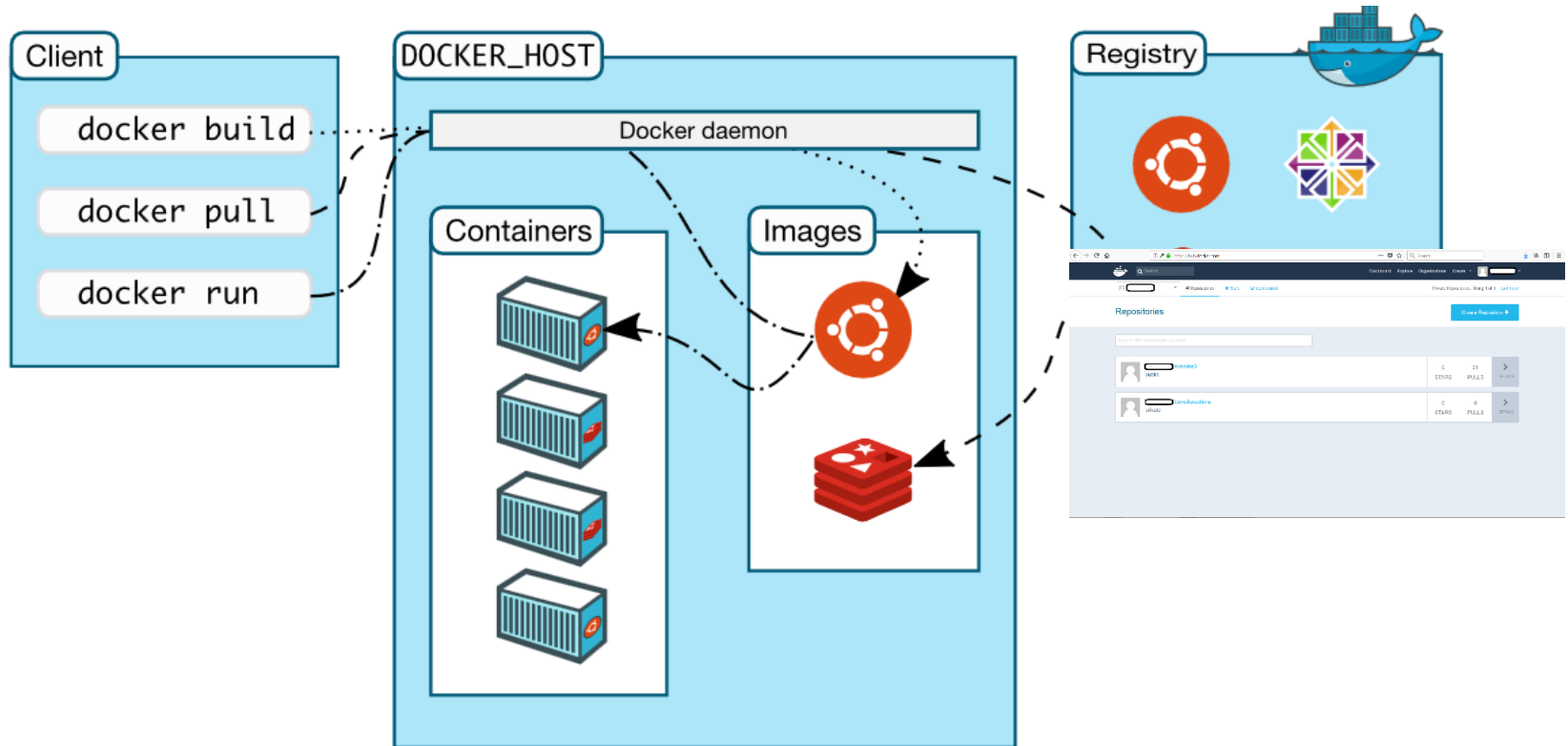
*Unionfs is a **filesystem** service for Linux, FreeBSD and NetBSD which implements a **union** mount for other **file systems**. It allows **files** and directories of separate **file systems**, known as branches, to be transparently overlaid, forming a single coherent **file system**. (wiki...)*

# Docker Containers -- Operations

- Docker **cached** the layers the **first time** of building them.
- For eg. ...
  - Golang || Apache
- For the initial install, Ubuntu is cached or golang;
- For the second build, only apache or mysql is built rather than initiating the build process from Ubuntu base!!!
- Thus, the deployment is faster using dockers.



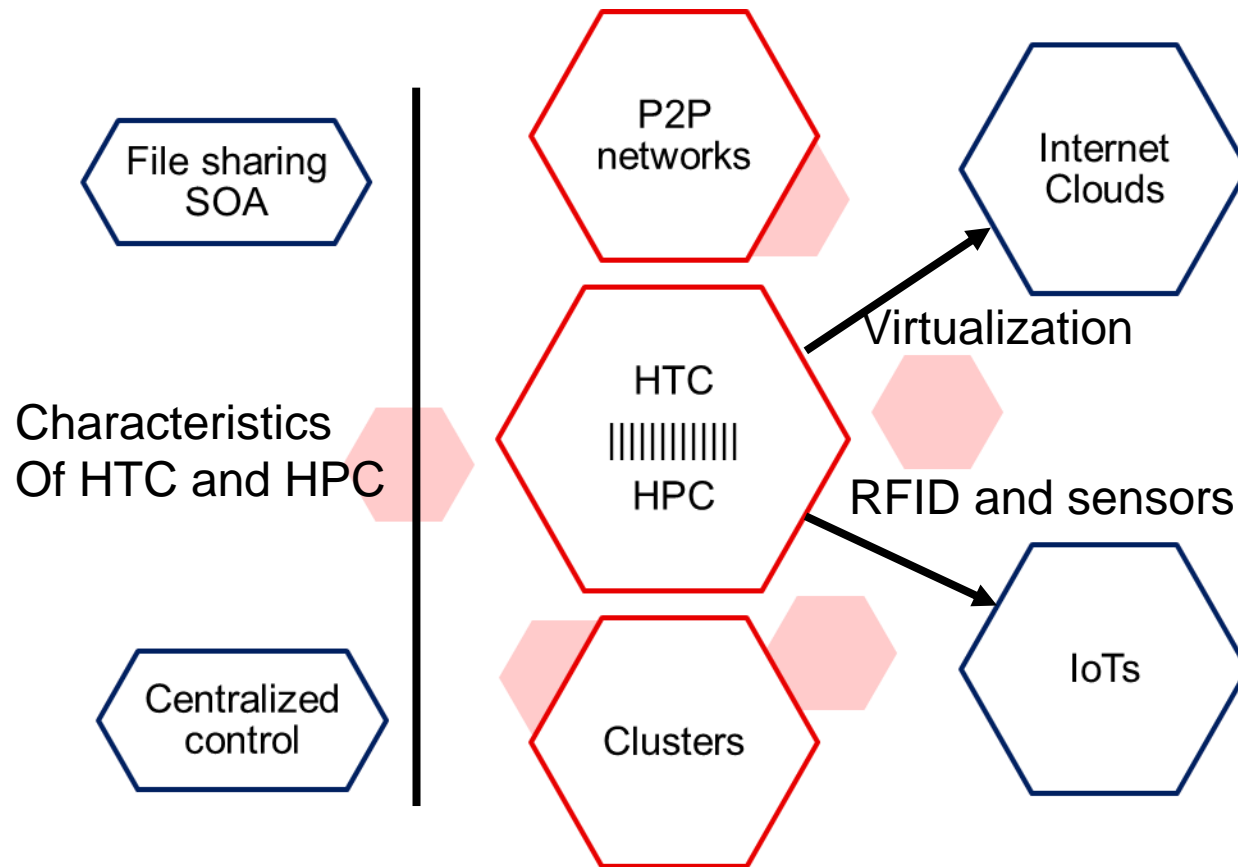
# Docker Architecture



- Docker client talks to the docker daemon.
- The docker daemon does the following:
  - Building, running, distributing docker containers.
- The docker registry stores docker images
  - Either in local registry or in public registries (such as, docker hub or docker cloud)

# Evolution of HTC and HPC over Internet

---



HPC -> Supercomputers are replaced by clusters of computers

HTC -> P2P networks, clouds, and webservice platforms are more focused on HTC's.

# Fog computing

---

- It is a computing platform for performing IoT analytics.
- Fog computing utilizes decentralized servers in between network core and network edge for **data processing**
- Fog computing serves nearer to edge nodes or sensor nodes.
- Should we replace Cloud Computing in IoT environments?
  - No,

# Fog Computing

---

- Fog nodes or clusters are typically deployed anywhere nearer to a service station.
- For e.g.,
  - on a factory floor,
  - In a smart city,
  - in a smart vehicle, and so forth.
- Any cluster with computing, storage, and network connectivity can be served as a Fog Cluster.

# Fog computing

---

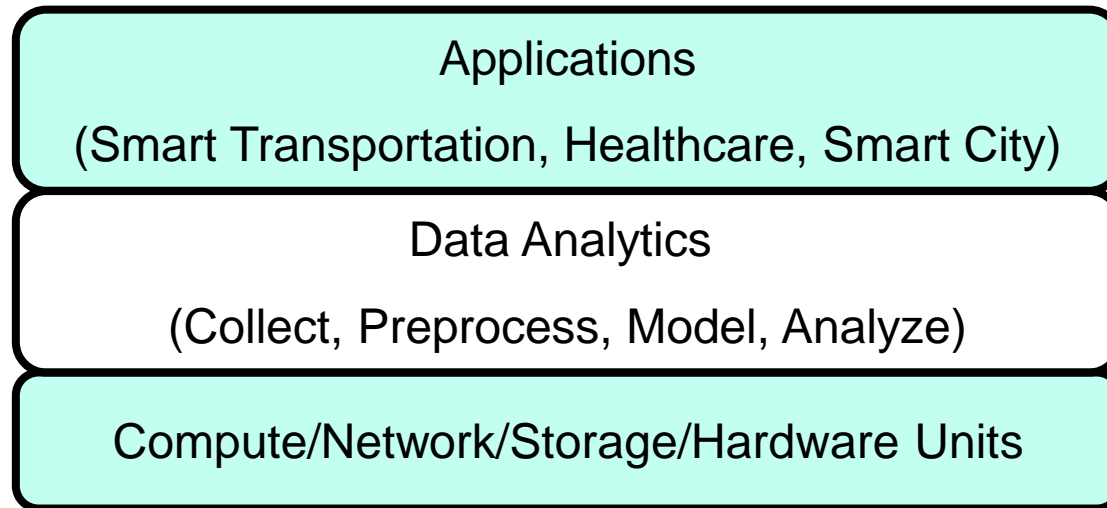
- Why cloud should be freed from computations?
  - To handle latency-sensitive applications....
  - To deal with efficient resource allocations...
  - To handle a large volume of BIG data... (and, so forth).
  - For e.g., a Jet engine fitted with 5,000 sensors can generate up to 10 GB of data per second.
- Consider applications such as
  - Smart Traffic Light System
    - Automatically activates traffic lights based on traffic.
    - Detects the presence of pedestrians and their frequency in accessing roads.
    - Goals...
      - (1) requires real-time reaction,
      - (2) near-real time, and
      - (3) relates to the collection and analysis of global data over long periods.



# Fog Software Architecture

---

- Fog architectures has to handle a large volume of data.
- Next, make decisions based on learning algorithms.



# Fog Software Architecture

---

- Fog nodes are **heterogeneous** in nature.
- Fog architecture should facilitate seamless **resource management** across diverse set of platforms..
- A fog platform should host different applications belonging to various sectors (healthcare, smart city, smart transportation, and so forth).

# Edge computing

---

- It enables connected devices to process data closer to where data is generated.
- The computation happens either in the sensor or sensor connected devices.
- They are designed specific to certain applications at a time.
- These devices are power constrained devices.
  - Eg. Raspberry pi or similar devices.
- Edge computing is preferred
  - when data processing should be much faster than expected.
  - Low bandwidth between sensors and cloud/fog nodes.
  - These nodes function similar to caches in computing applications!!!

# Edge Applications

---

- For eg. In autonomous vehicles (sensors should process immediately and near to the sensor).
- For e.g., For producing augmented reality in IIoT environments.
- Predictive Maintenance and Metaverse examples.

# Applications – Edge computing

---

- Transportation
- Healthcare
- Manufacturing
- Agriculture and smart farms
- Energy & grid control
- Any others...

## Advantages of Edge

- Real-time processing of larger data
- Lower costs or cheaper components are involved
- Network traffic is reduced.
- Increased application efficiency (more IoT applications can be processed in a network).

# Resiliency Challenges of Edges

---

- English Meaning: the capacity to recover quickly from difficulties; toughness.
- Edge applications require low latency and high bandwidth. However, we could not attain resiliency...
  - Scalability Issues Occur:
    - Connecting more number of devices to edge node could lead to scalability issues.
    - Note, scalability is possible in cloud scenario. But, it fails in edge.
  - Failover Issues:
    - Due to non-availability of resources, resiliency challenge occur.
  - Failing to meet real-time deadlines

# Authentication and Physical Challenges

---

- Security challenges need to be addressed in Edge nodes.
- For instances, the security solutions should be scalable and decentralized...
  - i.e., Scalable Authentication
    - For e.g., attackers can probe or install malicious software on IoT devices.
    - As a result, any cryptographic keys stored on the device are subject to tampering and eavesdropping.
  - i.e., Decentralized Security
    - Often, edge network will get disconnected.
    - Obviously, the security mechanisms in the edge devices should be autonomous. They should receive updated policies whenever the security issues are predicted.

# Multi-tenant Issues

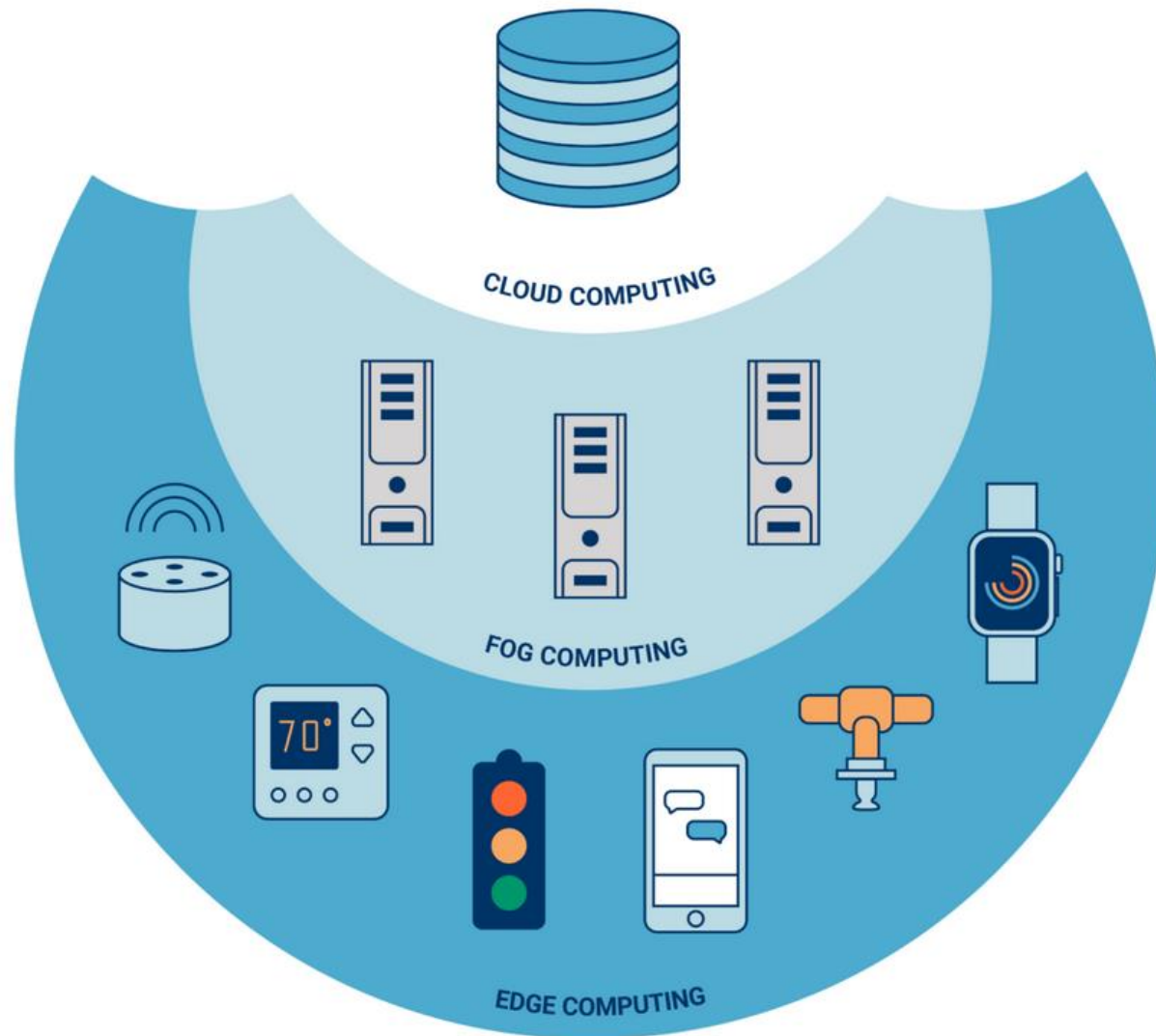
---

- An edge device, much like a computing node in the cloud, will need to support multiple tenants.
- Clouds perform this by virtualization combined with pay-per-use public billing system.
- Edge computers will need a similar billing system to properly manage resources in a congested system.
- But, how to exactly measure the hardware properties or energy utilization of running services? How to audit the services properly? – Challenges.



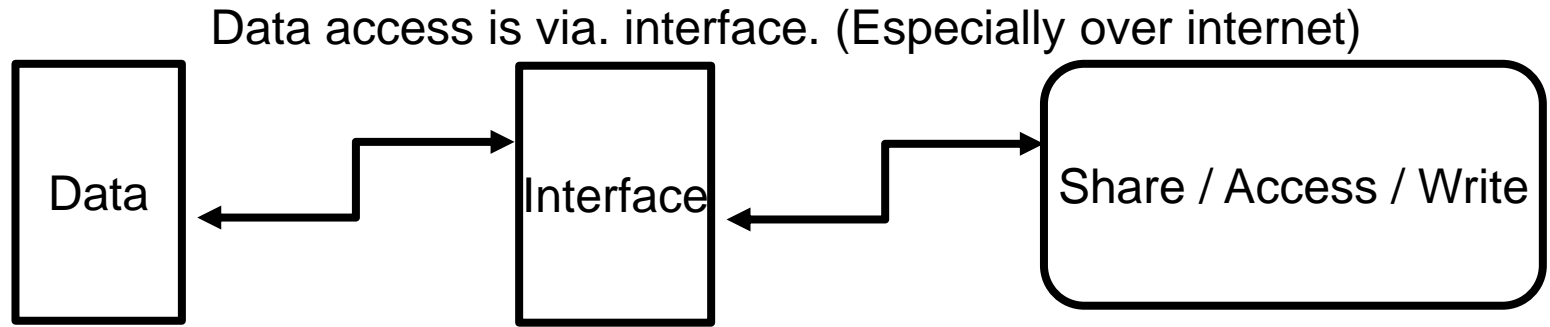
# Cloud, Fog, and Edge computing

---



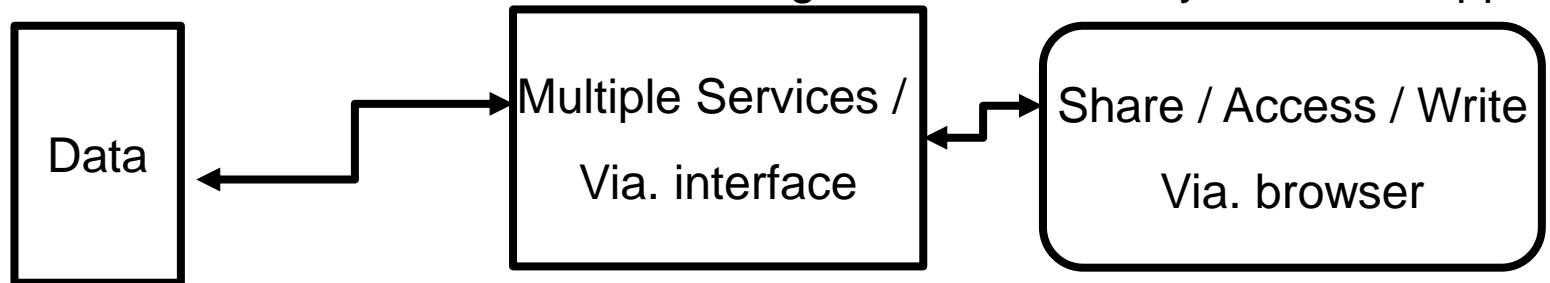
# Monolithic System

---



- Example case: Ecommerce application

Multiple services – Order service, Accounting service, inventory service, shipping



- Advantages: simple to develop, test, deploy.

# Challenges

---

- Challenges of such monolithic service system:
  - What happens if a new developer modifies the code??
    - He might have lesser knowledge on the existing programs or the associated programming language.
    - Hence, it takes too much time to delivery...
    - Not scalable
  - Software bugs lead to a standstill !!!
    - In feb2016, Japanese Hitomi spacecraft broke off due to s/w failures.
  - Software update requires the monolith system to rerun or reboot the entire application (all dependent services).
  - Most of the services are often loaded in containers these days.
    - The monolithic service system might have to load a large volume of programs on these containers!!! Hence, poor performance.
    - Migration issues due to heavy memory footprints.

# Introduction to microservices

---

- Microservices are distributed applications consisting of smaller independent (isolated) service modules.
- The service modules can be implemented in independent programming languages (node.js or go or java, python, or c++)
  - – leading to a flexible development scenario.
- Finding out a software bug could be much easier now (ie. Less time is required).
- Continuous integration of new developments is possible.
- In succinct,
  - Time to the market for services is much lesser than monolithic solutions.
  - Engineers and s/w developers could enjoy autonomy to develop, ship, and iterate s/w improvements.

# Microservices - Characteristics

---

## Flexibility

- Continuous integration of s/w

## Modularity

- Each service modules contribute to overall system

## Fine grain

- Each service should be as small as possible (but independent)

## Decentralized

- All services are decentralized.

Typically, several services are built behind a loadbalancer  
(or, service discovery platforms).

They utilize OS-level containers rather than heavy weight VMs.

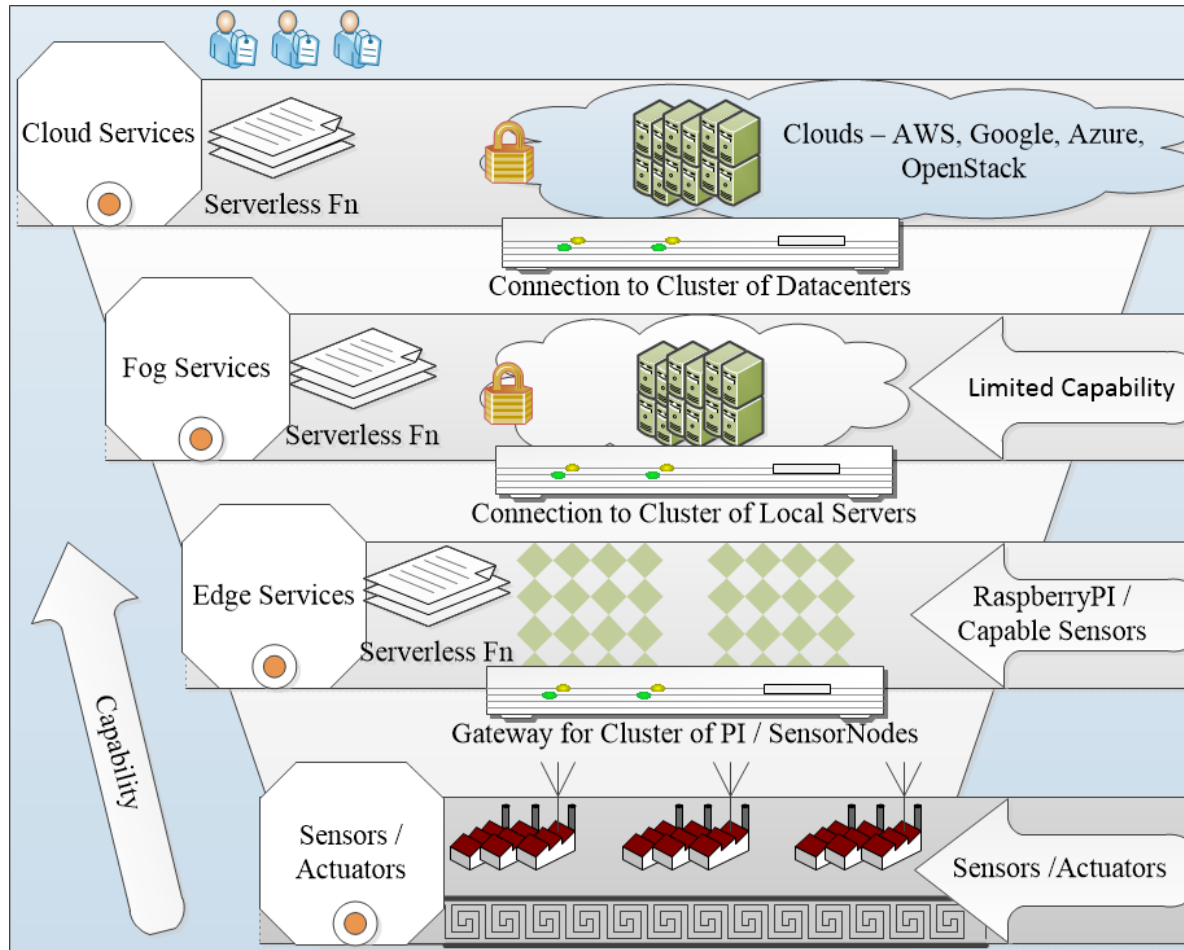
This architecture is made for DevOps.

# Serverless Computing

---

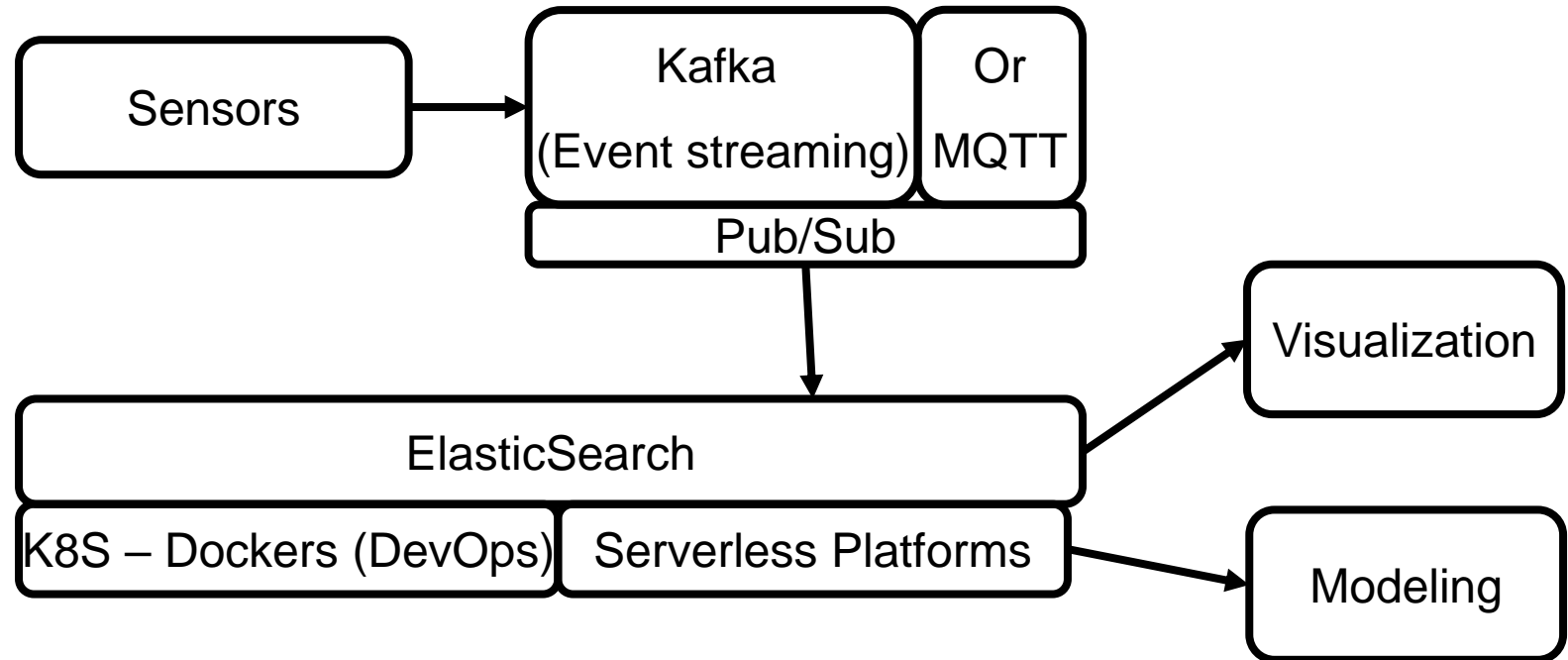
- It is an execution model of Clouds (next step to microservices). It is named as **NoOps**.
- Here, servers are hidden;
- In this approach, the server-side application is reduced to functions.
- These functions are hosted on cloud infrastructures.
- They are executed on parallel containers of clouds.
- Hence, these containers could be scaled and monitored separately...
- Who develops serverless functions?
  - Application developer
    - They want their code to behave as serverless...
- It is also named as **Function-as-a-Service**.

# Serverless IoT – Use Case



# Example IoT Serverless Platform

---





# Serverless Platforms

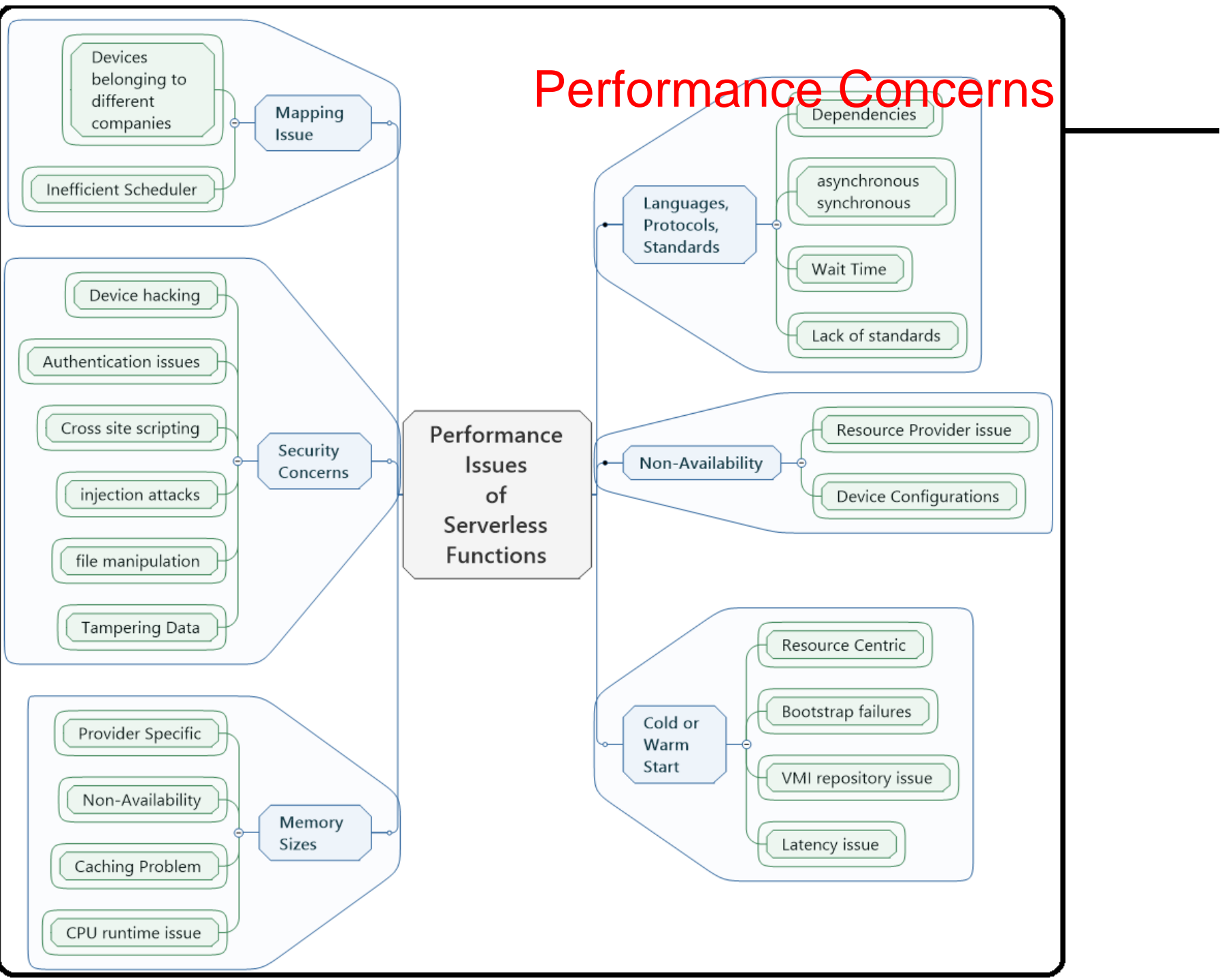
---

- Functions are executed on containers or server instances.
- Certain Platforms are available:
  - kubeless functions, iron functions, Fission, Fn platform, and so forth, are executed on Kubernetes;
  - OpenWhisk functions are executed on IBM Apache servers;
  - AWS Lambda and Sparta are executed on AWS instances;
  - Google Cloud Functions are accomplished using Google Cloud Run; and
  - Microsoft Azure functions are executed on Azure.

---

# **Performance Concerns and Tools...**

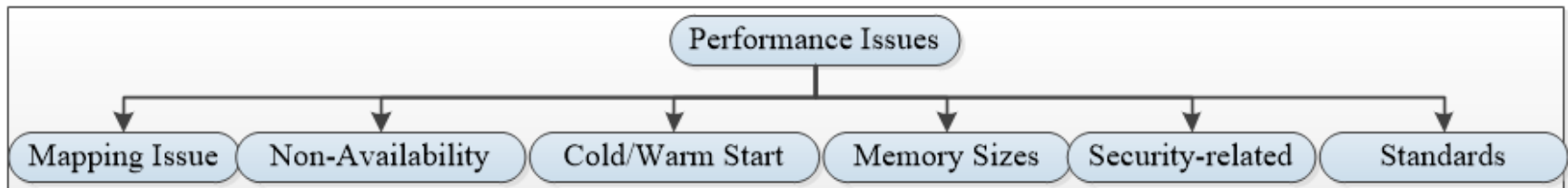
# Performance Concerns



# Performance Monitoring Mechanisms

---

- Mapping Issue
  - Protocols and Rules of different heterogeneous IoT Sensors
- Non-Availability Issue
  - Containers, Memory, and Network availability
  - Providers have to provide them.
- Cold – Warm start Issue
  - Cold start has high overhead than Warm start



# Performance Monitoring Mechanisms

---

- Impact of Memory Sizes

- Several providers offer only limited memory sizes for executing functions.
- For. E.g., 128MB to 3008 MB free for functions.

- Security and Standards

- Lack of proper standards or protocols for serverless functions at the IoT domain

# Performance Monitoring Mechanisms

---

## Tracing Approach

- It is carried out by sniffing into the log or error messages of serverless functions in a distributed fashion.

## Sandboxing Approach

- Here, an isolated and secure environment will be created to study the performance impact of the newer services (e.g., predictive maintenance in IIoT)

## Modeling / Prediction Approach

- modeling of the end-to-end execution of serverless functions (to study the performance impacts)

## Language-Inherent or Instrumentation Approach

- It is done by injecting a few predefined functions or notations into the serverless functions before their executions

# Performance Analysis Tools


---



Amazon X-Ray



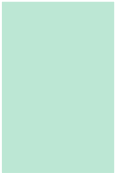
Google  
StackDriver



ApplicationInsights



CloudWatch



Prometheus  
(OpenWhisk),



Thundra



AppDynamics



Wavefront



Epsagon

# Performance Analysis Tools

1. Amazon X-Ray
2. Google StackDriver
3. ApplicationInsights
4. CloudWatch
5. Prometheus
6. Thundra
7. AppDynamics
8. WaveFront
9. Epsagon

Description	1	2	3	4	5	6	7	8	9
Commercial	1	1	1	1		1	1	1	1
Opensource					1				
FreeTrial Version	1	1	1	1	1		1	1	1
Logs		1		1				1	
Plugin			1						
Tracing	1				1	1	1	1	1
Library		1			1	1			1
Daemon / Agent			1				1		
Service	1			1				1	1
Function	1	1	1		1	1		1	
Task		1	1						
Application				1		1	1		1



# Challenges of Tools / Research Directions

---

The serverless functions are often short-lived. Hence, it is difficult to reproduce the same fault of serverless functions.

Large volumes of performance data are possible (visualization issues).

PA Tools face vendor-lockin problems.

Integration issues.

Difficult to precisely identify the energy consumption issues.

# AWS IoT Greengrass

---

- AWS IoT Greengrass is a software that extends cloud capabilities to local devices.
- Everything functions as services.
- Use case 1:
  - Data is closer to the sources.
  - Data management feature.
  - An edge computing service.
- Use case 2:
  - AWS IoT Greengrass developers can use AWS products to develop their products without the help of clouds (avoids high latency).
  - Later, it could be deployed on millions of devices across the globe.

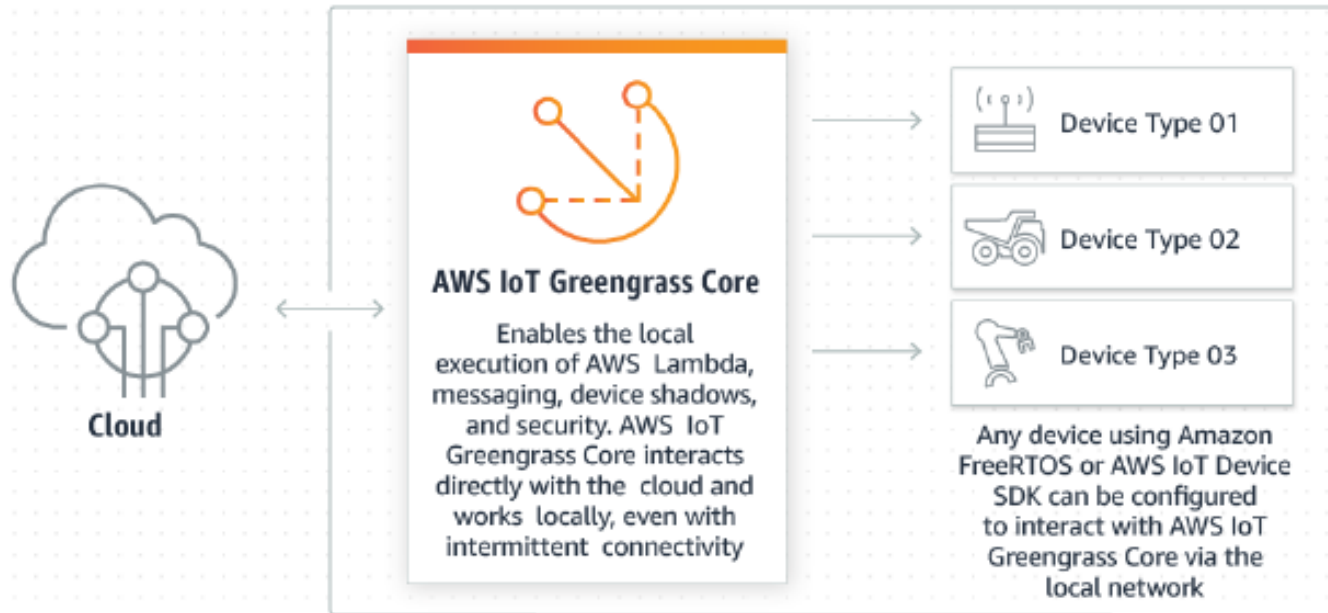
# AWS IoT Greengrass

---

- Use case 3:
  - Learning inferences at edge nodes.
  - Video captures and analytics.
- Use case 4:
  - Note: Security is offered from edge nodes to cloud services.
  - With the inclusion of greengrass, a more robust security features could be added in these nodes.

# AWS IoT Greengrass

Aws website

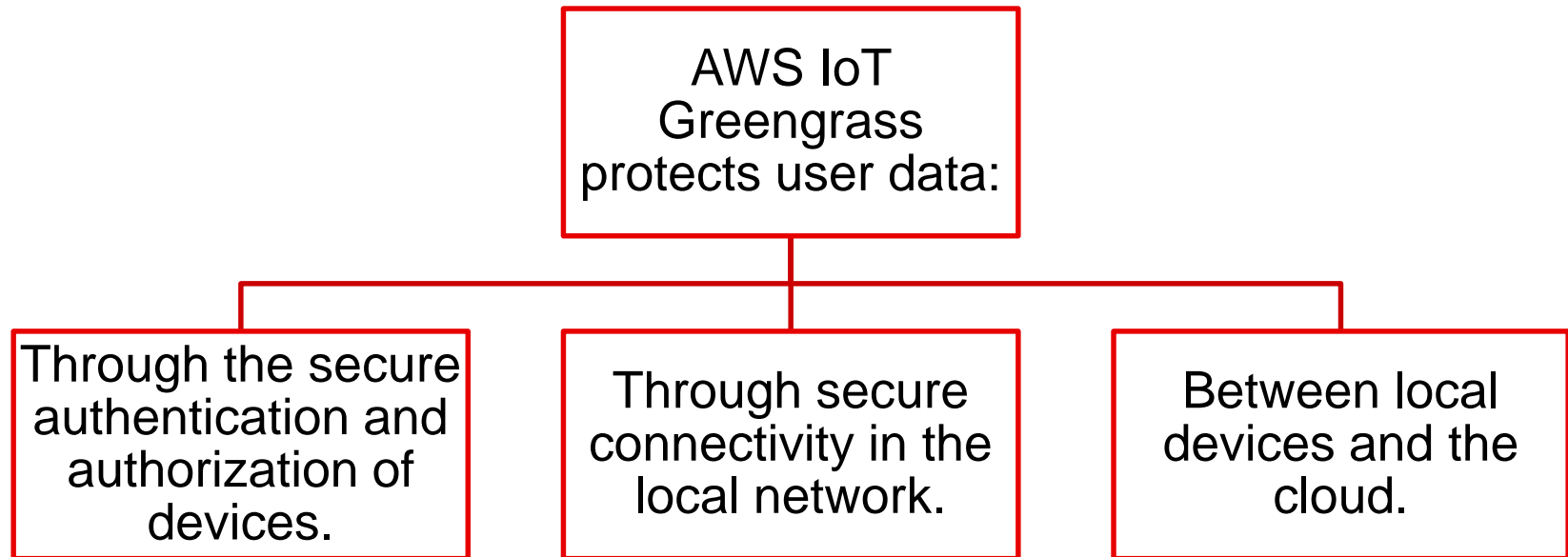


<https://www.youtube.com/watch?v=ynY7vQtcQUw>

# AWS IoT Greengrass

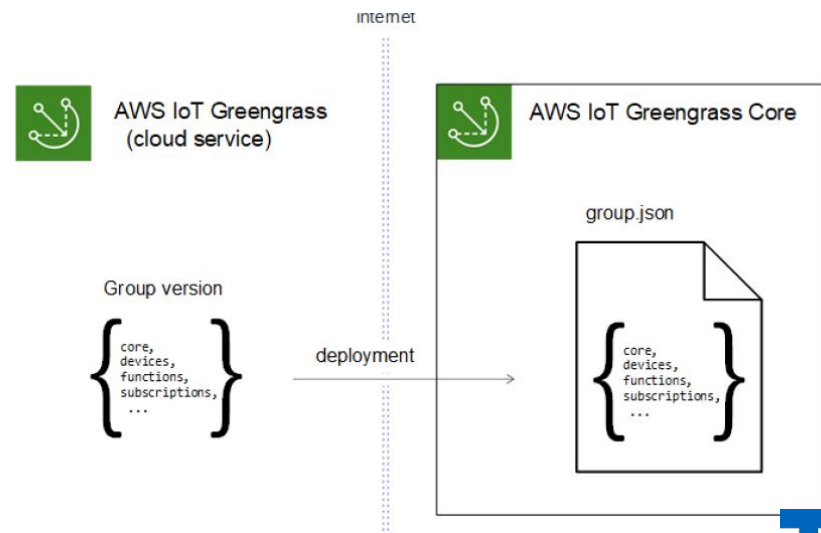
---

Greengrass continues to run even if there is no cloud connectivity or internet  
– A Green Setup



# AWS IoT Greengrass groups

- AWS IoT Greengrass groups are utilized to **organize entities** of edge environments.
- Additionally, groups regulate how members of the group can communicate with one another and the AWS Cloud.
- For instance,
  - only the devices in the group can use the local MQTT server for communication.



# AWS IoT Greengrass SDKs

---

- AWS IoT Greengrass provides AWS IoT Greengrass Core SDKs. (opensource)
- Languages include...
  - • AWS IoT Greengrass Core SDK for Java
  - • AWS IoT Greengrass Core SDK for Node.js
  - • AWS IoT Greengrass Core SDK for Python
  - • AWS IoT Greengrass Core SDK for C
- Using **AWS SDKs**, devices could easily connect with AWS services, including Amazon S3, DynamoDB, AWS EC2, AWS IoT, and so forth.
- Lambda functions are executed on AWS IoT Greengrass core unit.

# AWS IoT Greengrass

---

- Migrating Cloud-Based Lambda Functions:
  - AWS SDK programming model is utilized in IoT Greengrass Core.
  - The SDK easily ports Lambda functions that are developed for the cloud to Lambda functions that **run on an AWS IoT Greengrass core**. (AWS documentation)
- Controlling execution of greengrass lambda functions
  - AWS IoT Greengrass provides cloud-based management of Greengrass Lambda functions.



# Monitoring AWS IoT greengrass

---

- Monitoring is an important part of maintaining the reliability, availability, and performance of AWS IoT Greengrass and your AWS solutions
- Monitoring goals:
  - ❖ What are your monitoring goals?
  - ❖ Which resources will you monitor?
  - ❖ How often will you monitor these resources?
  - ❖ Which monitoring tools will you use?
  - ❖ Who will perform the monitoring tasks?
  - ❖ Who should be notified when something goes wrong?
- ❖ 3 most common monitoring tools
  - ❖ Amazon CloudWatch logs
  - ❖ AWS cloudtrail Log
  - ❖ AWS Eventbridge (provides notifications about state changes)

# AWS IoT Greengrass – Use cases

---

- Machine learning inference
- Vehicular automation

# Steps – AWS Greengrass on Raspberry

---

- Setup a raspberry pi as AWS IoT greengrass core unit.
- Setup raspberrypi
  - Download Raspbian Buster OS.
  - Use Etcher SD card writing tool to flash the OS to SD card.
  - Install OS and launch a working raspberry pi device.
  - Ensure that we are able to connect this device using SSH.

# Steps – AWS Greengrass on Raspberry

---

- AWS IoT Greengrass core
  - Step 1: Create a greengrass group
  - Step 2: Setup the access policies
  - Step 3: Download security credentials (.tar.gz)
  - Step 4: Download greengrass core software components (based on your device).
    - E.g., Armv7l Linux version for raspberry pi
  - Step 5: Start greengrassd on RaspberryPi unit.
  - Step 6: Create lambda functions (for triggering events).

---

# IOT ANALYTICS

# IoT Analytics

---

- Data analytics is the process of
  - deriving knowledge from data,
  - generating value like actionable insights from them.
- To discuss –
  - Categorization of analytic approaches

# IoT analytics – Application Domains

---

Several  
applications

- Smart Healthcare
- Smart Home
- Smart logistics
- Smart manufacturing
- Smart agriculture
- Smart Energy
- Smart Buildings

# Advantages of Analytics in IoT

---

- Analytics add value to IoT applications.
- It enables automation-based systems.
- Learning/Inferencing could be wider than a human-based approach.
- There is no need for human interventions in most cases.



# Modeling Basics

---

- In general, a model is a quantifying system that mimics a real-life situation in terms of mathematical expressions;
- It could be used for predicting future events or for understanding the current situations.
- The three most important variables used in a modeling process are
  - i) dependent variable or response variable or outcome variable,
  - ii) independent variable,
  - and iii) the other variable.

# Modeling Basics

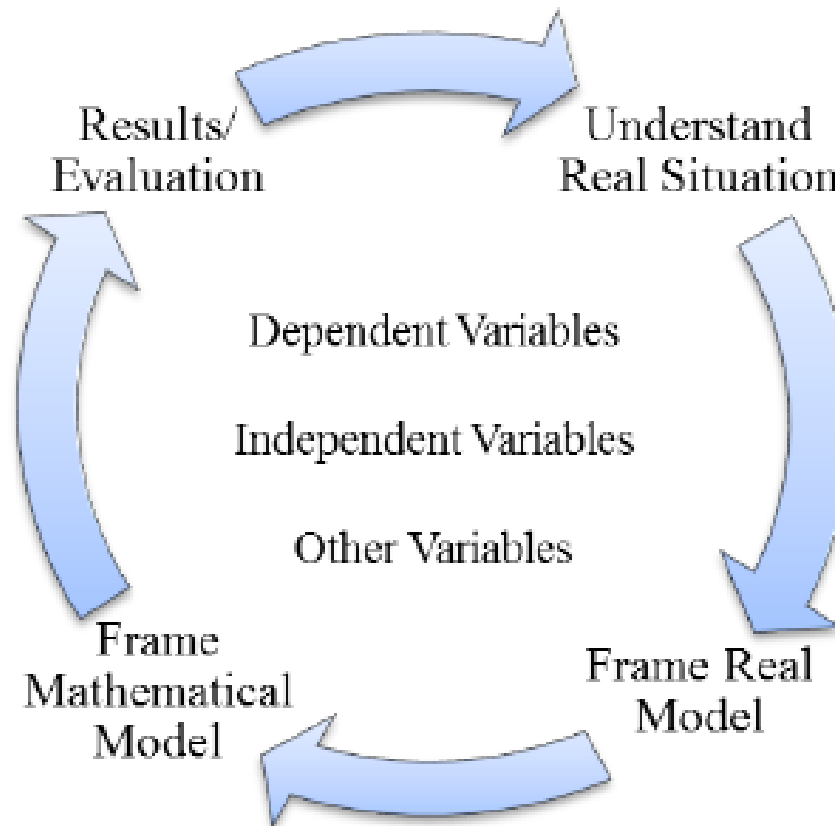
---

- A dependent variable could be realized as a variable which is required to get an answer after prediction;
- An independent variable is a variable that directly influences the dependent variable during the process of prediction;
- The other variables, e.g., coefficient variables or weights, are the variables which do nothing with real experiments. However, these variables could heavily influence the modeling outcome in terms of prediction results or goodness of fit.

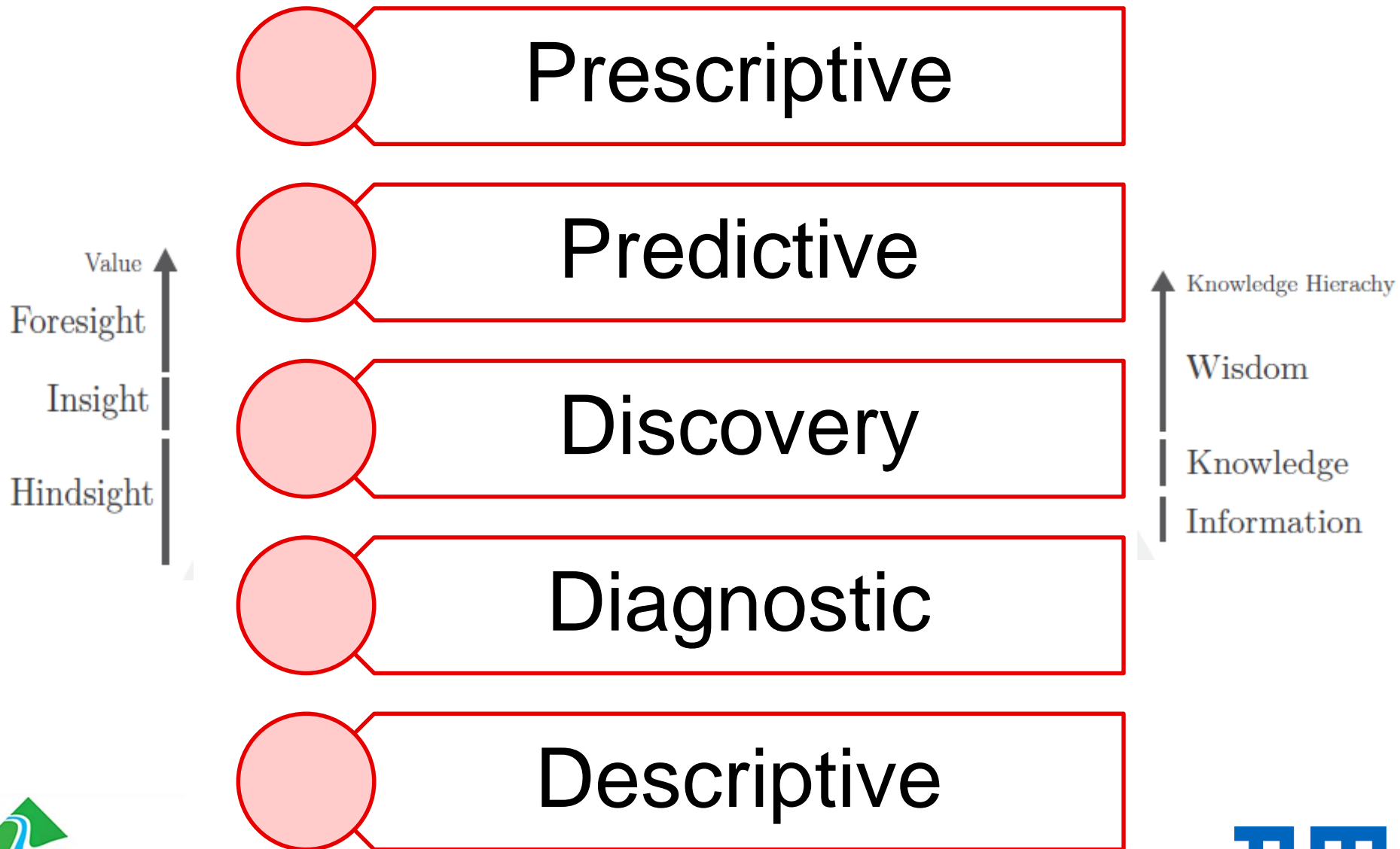
# Model Development

---

- A model development process has four step-wise activities as shown in Figure:



# Categories of Analytics Capabilities



# Descriptive Analytics

---

- Descriptive *analytics* is based on the past activities or incidents.
  - It enables us to understand the reasons behind past success or failure.
- It helps us to answer the question “What happened?”
- For eg. We could check the humidity value and describe the reason for it.

# Diagnostic Analytics

---

- It is the process of understanding why something has happened.
- One step deeper than descriptive analytics
- I.e. It attempts to find the root cause of the problem.

# Discovery in analytics

---

- Discovery in analytics deals with diving into the search space of analytic applications.
- i.e., Diving into the available big data.
- Discovery attempts to answer the question of **what happened** that we don't know about !!!
- Difference of this approach compared to the previous types of analytics is that the analysis is so deep that it tries to find some **novel reasons/suggestions**.

# Predictive analytics

---

- The previous three approaches offered only hindsight or insights of problems or situations.
- Predictive analytics and prescriptive analytics offer foresight information ( i.e., future values too).
- Predictive analytics answers the question **what is likely to happen**.
- This is when historical data is **combined with rules, algorithms**, and occasionally external data to determine the probable future outcome of an event or the likelihood of a situation occurring.
  - [https://www.wikiwand.com/en/Prescriptive\\_analytics](https://www.wikiwand.com/en/Prescriptive_analytics)



# Prescriptive analytics

---

- Prescriptive analytics not only anticipates what will happen and when it will happen, but also **why it will happen**. (Wikipedia)
- Further, prescriptive analytics **suggests decision options** on how to take advantage of a future opportunity or mitigate a future risk and shows the implication of each decision options.
- Prescriptive analytics ingests hybrid data, a combination of structured (numbers, categories) and unstructured data (videos, images, sounds, texts), and business rules to predict what lies ahead and to prescribe how to take advantage of this predicted future without compromising other priorities. (Wikipedia)

# Specific Types of Analytics

---

- Visual Analytics

- It combines interactive visualizations with data analytics techniques.
- It is applied for a better understanding, reasoning and decision making of large and complex datasets.
- For eg. What is the percentage of male and female in India?

- Data Mining

- It is a part of the Knowledge discovery from data.
- Here, the interesting patterns and knowledge are discovered from large amounts of data.
- For eg.
  - Multi-dimensional data summary,
  - Association and correlation,
  - classification,
  - clustering,
  - pattern discovery,
  - anomaly detection. [www.sbenedictglobal.com](http://www.sbenedictglobal.com)

# Specific types of Analytics

---

- Content Analytics

- Content analytics is a broad area in which analytical techniques are applied to digital content.

- Text Analytics

- Text analytics is the derivation of high quality information from unstructured text.

- Video analytics

- It is about the use of specific software and hardware to analyze captured video.
- To detect objects, events, behavior, etc in real time.

# Specific types of Analytics

---

- Trend analytics

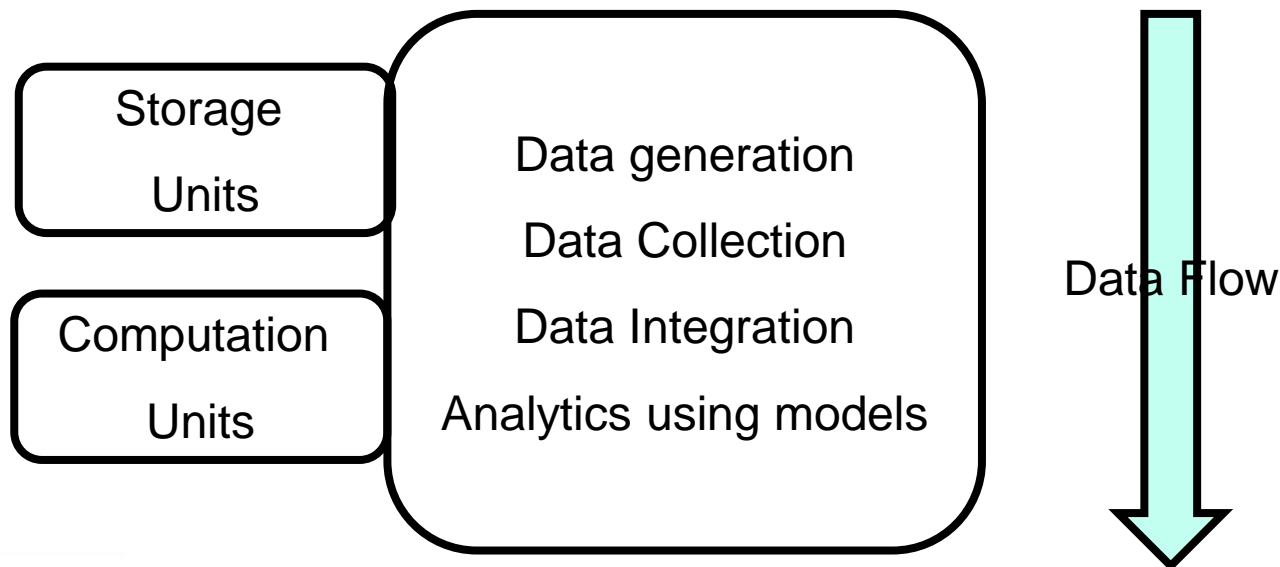
- It is about looking into the data and events across time.
- Time-series information.

- Business analytics

- It is the practice of using an organizations' data to gain insights through analytical techniques that can better inform business decisions and optimize business processes.

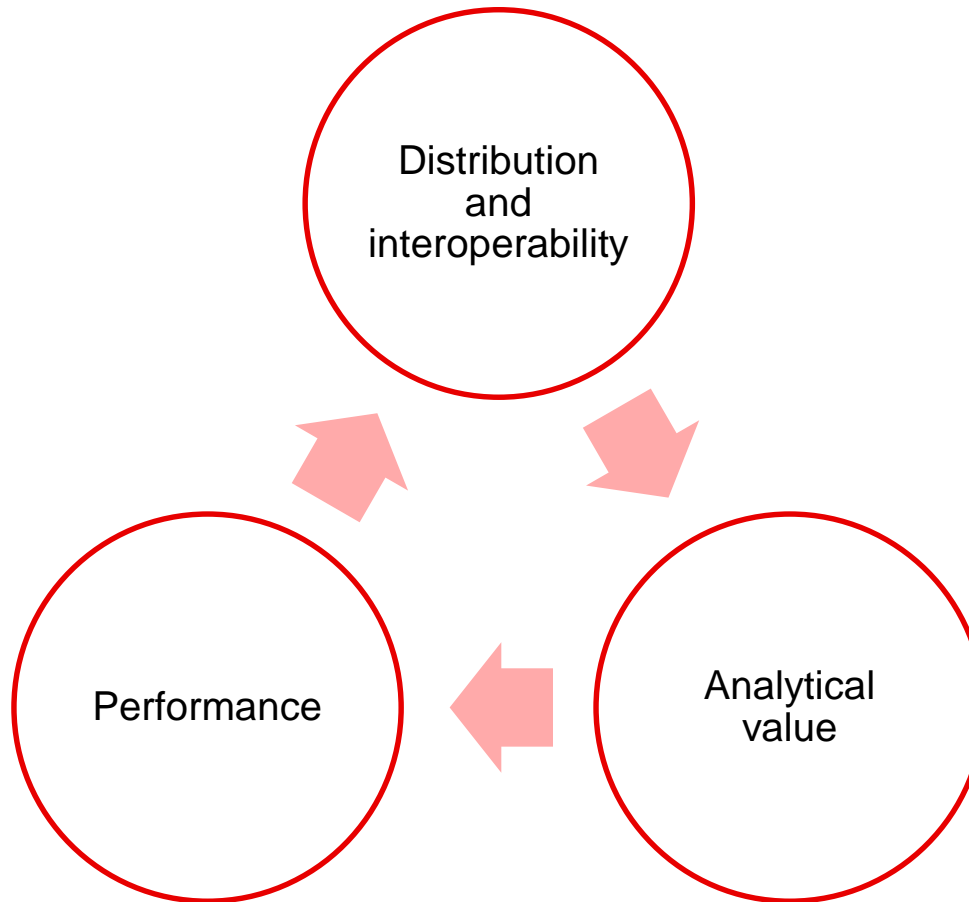
# Data flow process in IoT Analytic Architectures

- Step 1: Data generation and collection from sensors or storage units.
- Step 2: Data aggregation and integration with storage and compute units (Hadoop or cloud or local).
- Step 3: IoT data analytics.



# Analytics -- Challenges

---



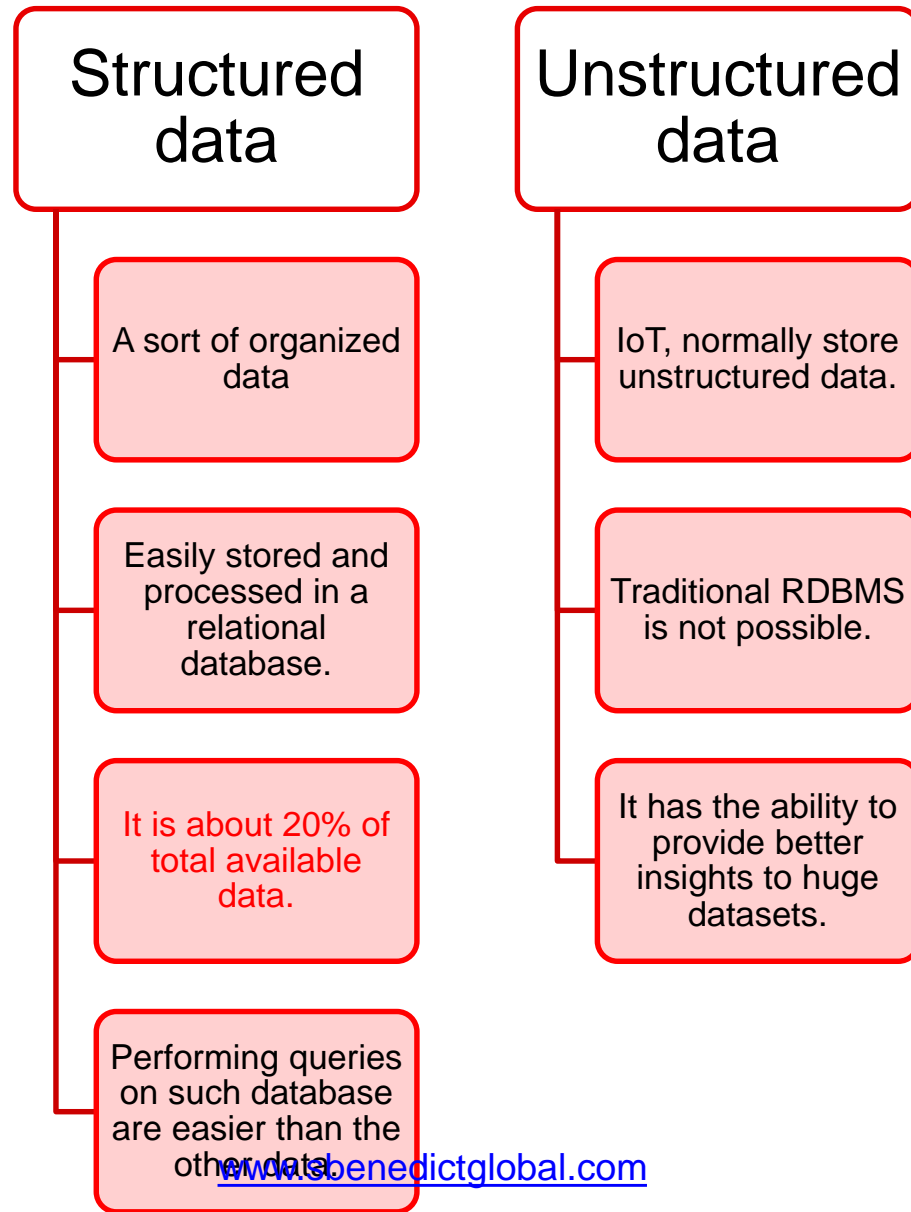
# Analytics - Challenges

---

- Tens of thousands of data emerge in IoT system.
  - It is a BIG DATA problem.
  - How to handle the data?
  - How to collect the data?
  - How to analyze the data?
- 
- What are the programming models available for handling data?

# Types of Data

---





# Edge Analytics

---

- Here, data is analyzed in the edge devices.
- Mobile devices, wearables, cameras, and the other connected devices are participants for edge analytics.
- Also, microcontrollers or tiny microprocessors.
- Characteristics:
  - Speed
  - Reduced communication cost
  - React locally by its own decisions
  - Collaborate with nearer devices.

# Machine Learning on Edge

---

- ML on edge is carried out by several algorithms such as Deep Learning (Neural Networks), SVM, Random Forests, and so forth.
- Here, the decentralized ML is undertaken by
  - i) transferring the model
  - li) transferring the data
  - lii) transferring the model parameters
  - Iv) hybrid approaches.
- Typically, this learning process is performed in a hierarchical learning fashion.
- One subset of the learning process that utilizes edge nodes is termed as Federated learning.

Shajulin Benedict, Energy Efficient Aspects of Federated Learning -- Mechanisms and Opportunities, in icSoftComp2020, in Soft Computing and its Applications, Springer-CCIS series, DOI: [https://doi.org/10.1007/978-981-16-0708-0\\_4](https://doi.org/10.1007/978-981-16-0708-0_4), 2021.