

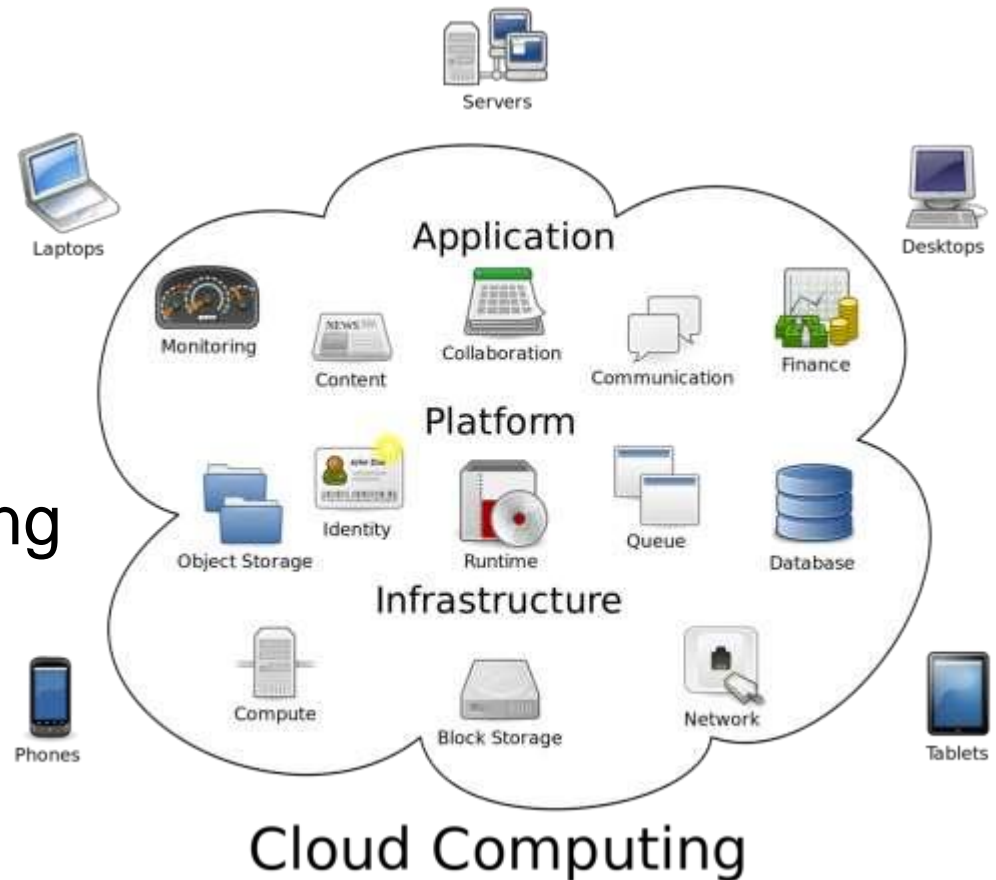
# Cloud Computing

## UNIT 5

Data Collection, Storage and Computing  
Using a Cloud Platform  
for IOT Applications/Services

# Introduction

- Sharing of resources over a network
- Virtually infinite computing resources
- Scalable on-demand
- Pay as you go

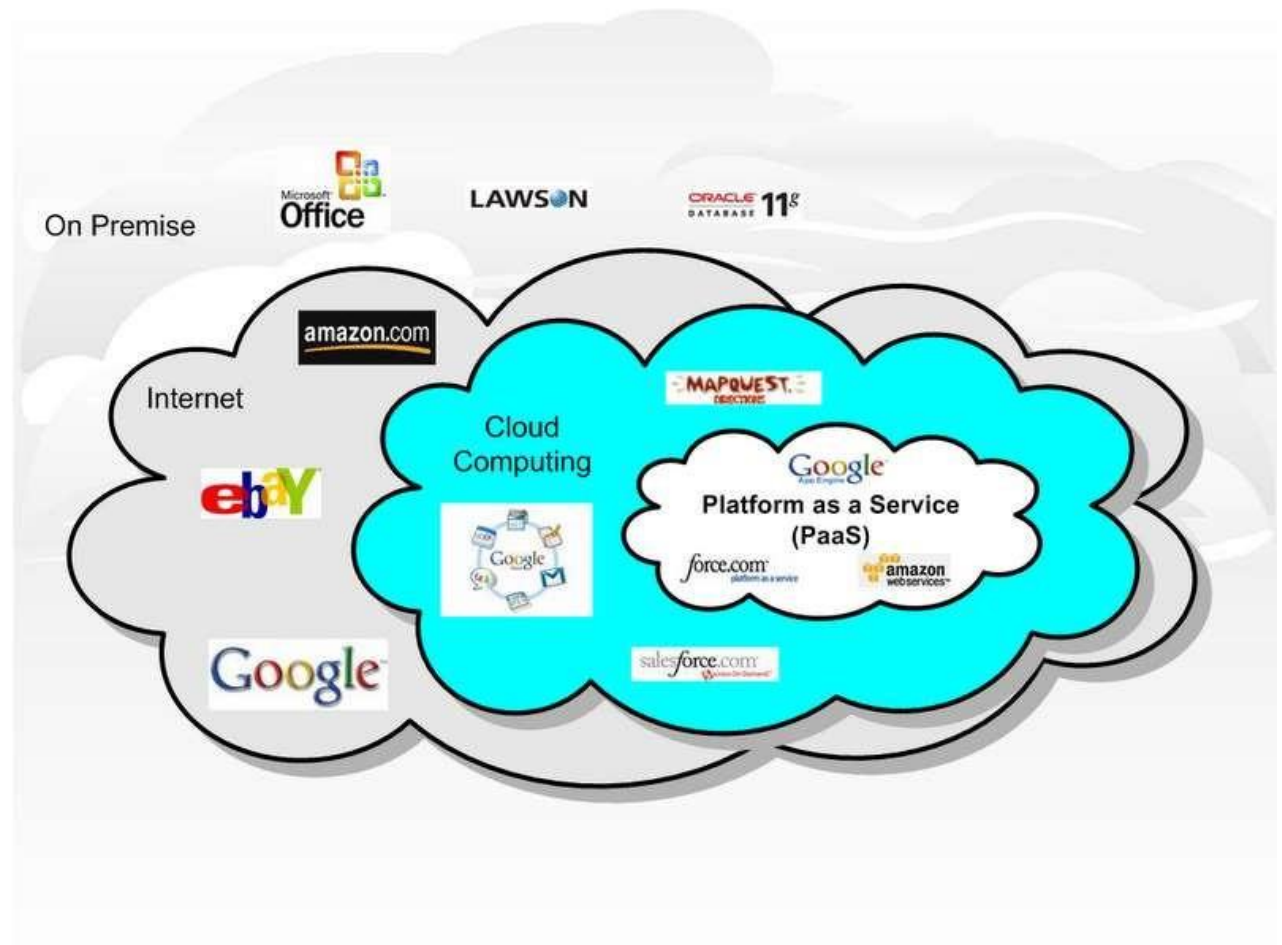


# Definition of Cloud Computing

- **"Clouds are vast resource pools with on-demand resource allocation.**  
– *Jan Pritzker*
- **“The analogy that finally made sense to them is what I will call 'cloud dining.’”**  
– *Omar Sultan*
- **"Cloud computing is ... the user-friendly version of grid computing.”**  
– *Trevor Doerksen*

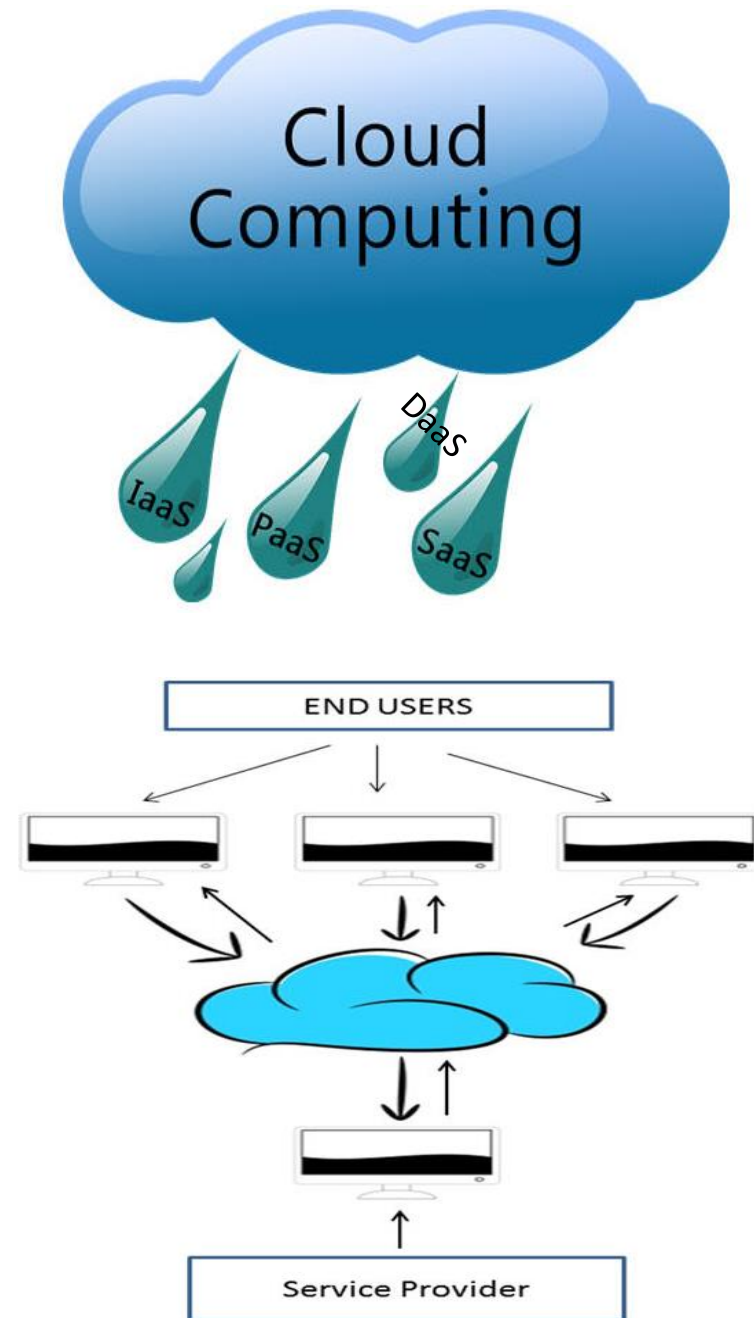
# Applications

- Academic
- Enterprise



# Cloud Computing

- **Cloud Computing** is defined as storing and accessing of data and computing services over the internet. It doesn't store any data on your personal computer. It is the on-demand availability of computer services like servers, data storage, networking, databases, software, analytics, and intelligence etc. The main purpose of cloud computing is to give access to data centers to many users. Users can also access data from a remote server.
- **Examples of Cloud Computing Services:** AWS, Azure, Google Cloud, Xively, Nimbits



# Cloud Computing Necessity

- With increase in computer and Mobile user's, data storage has become a **priority** in all fields. Large and small scale businesses today thrive on their **data** & they spent a huge amount of money to maintain this data. It requires a strong IT support and a storage hub. Not all businesses can afford high cost of in-house IT infrastructure and back up support services. For them **Cloud Computing is a cheaper solution**. And provides efficiency in storing data, computation and less maintenance cost.
- Cloud computing decreases the hardware and software demand from the user's side. The only thing that user must be able to run is the cloud computing systems interface software, which can be as simple as Web browser, and the Cloud network takes care of the rest. Some of the popular cloud services we have used or we are still using are mail services like **gmail**, **hotmail** or **yahoo** etc.

# INTRODUCTION TO DATA COLLECTION AND STORAGE

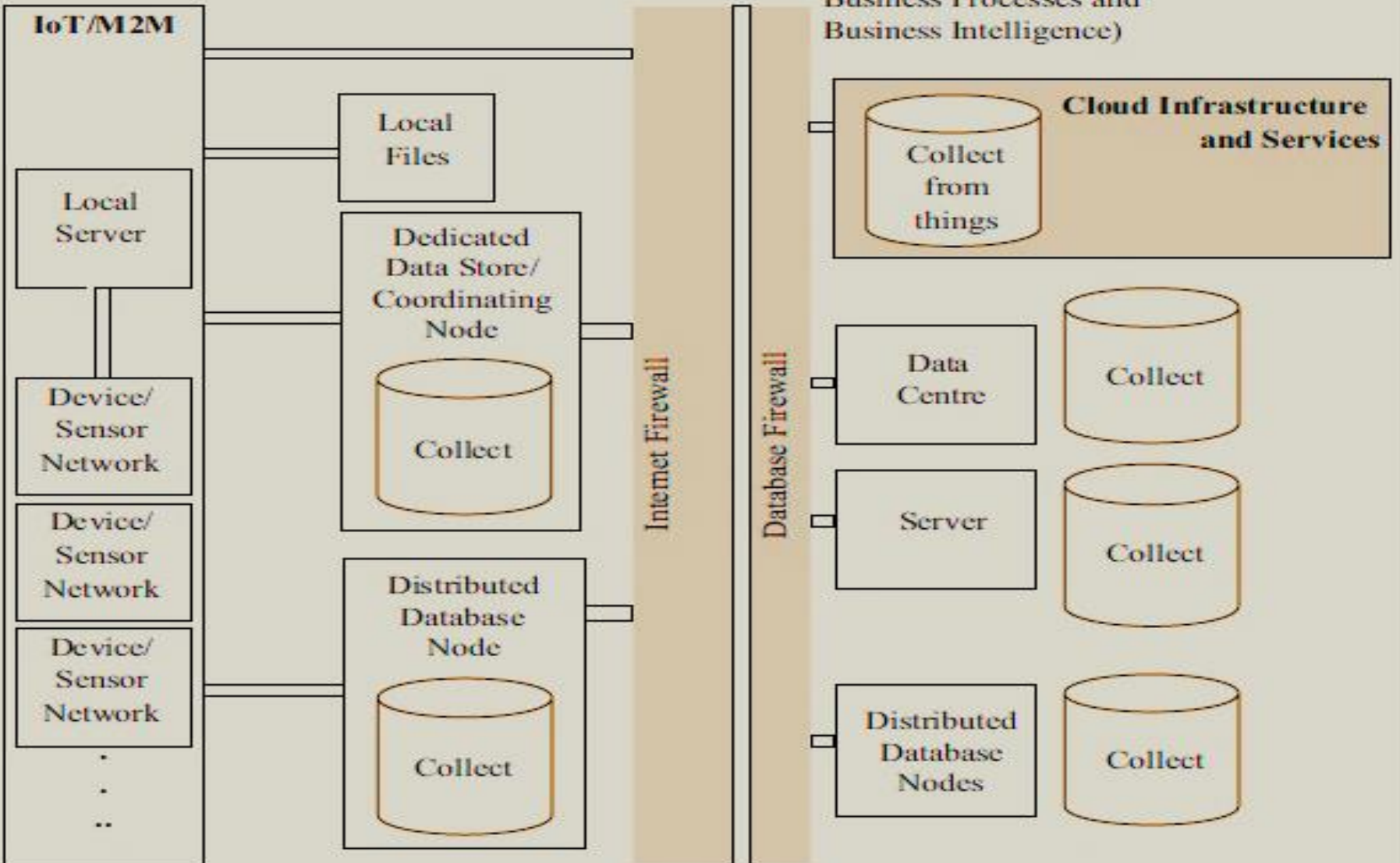
- A few conventional methods for data collection and storage are as follows: Communicating
  1. Saving devices' data at a local server for the device nodes.
  2. The devices' data in the files locally on removable media, such as micro SD cards and computer hard disks
  3. The data and results of computations in a dedicated data store or coordinating node locally
  4. Data at a local node, which is a part of a distributed DBMS
  5. At a remote node in the distributed DBMS
  6. On the Internet and saving at a data store in a web or enterprise server
  7. Communicating on the Internet and saving at data center for an enterprise
  8. Cloud is a new generation method for data collection, storage and computing.

# CLOUD COMPUTING PARADIGM FOR DATA COLLECTION, STORAGE AND COMPUTING

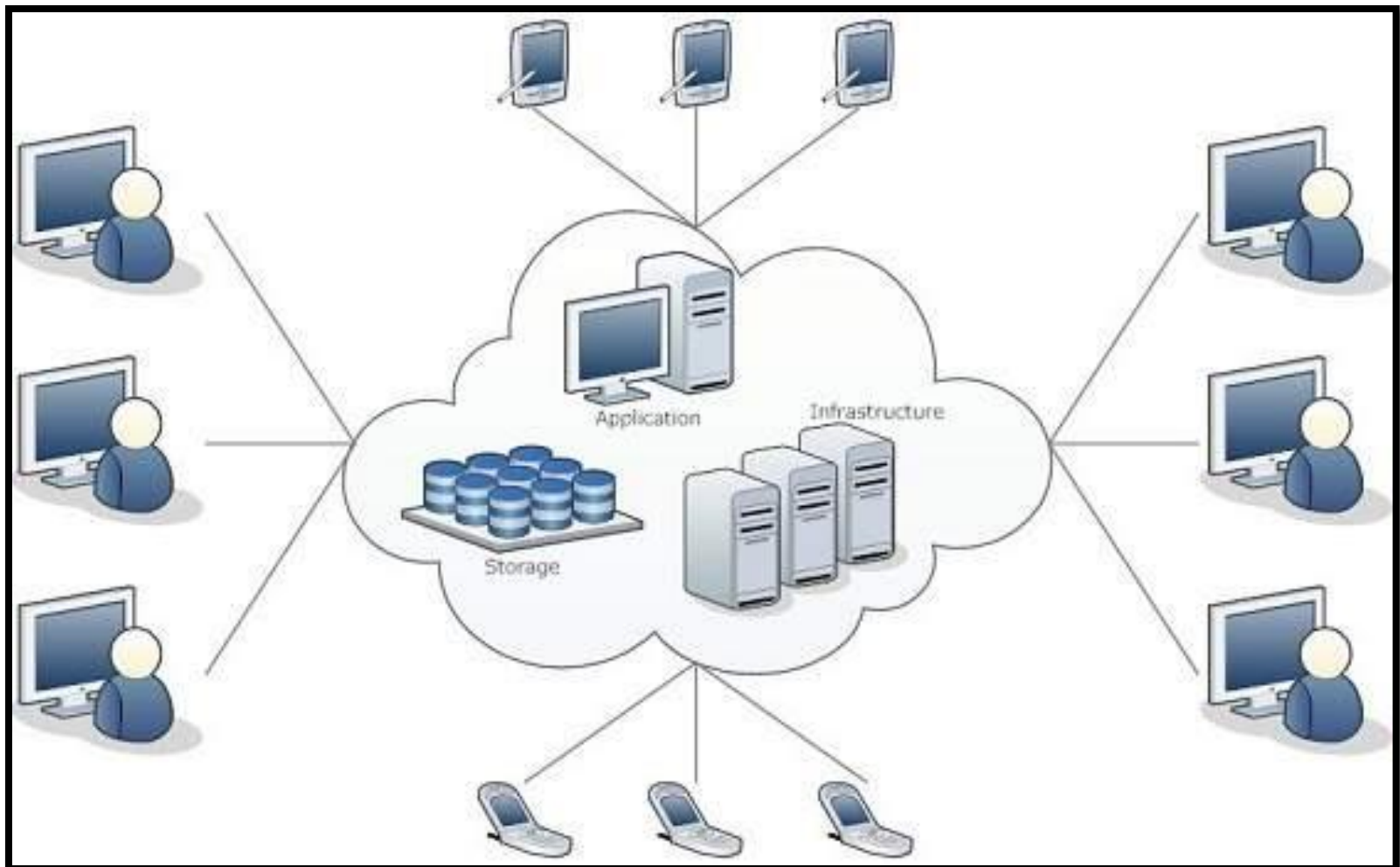
- Different methods of data collection, storage and computing in fig (i) Devices or sensor networks data collection at the device web server, (ii) Local files, (iii) Dedicated data store at coordinating node, (iii) Local node in a distributed DBMS, (iv) Internet-connected data centre, (v) Internet-connected server, (vi) Internet-connected distributed DBMS nodes, and (vii) Cloud infrastructure and services.
- Cloud computing paradigm is a great evolution in Information and Communications Tech. (ICT) - The new paradigm uses XAAS at the Internet connected clouds for collection, storage and computing.



**IoT/M2M Local device network**  
data collection and storage



Cloud Computing refers to **manipulating, configuring, and accessing** the hardware and software **resources** remotely. It offers online **data storage, infrastructure, and application**.



# key terms and their meanings

- **Resource** refers to one that can be read (used), written (created or changed) or executed (processed). A path specification is also a resource. The resource is atomic (not-further divisible) information, which is usable during computations. A resource may have multiple instances or just a single instance. The data point, pointer, data, object, data store or method can also be a resource.
- **System resource** refers to an operating system (OS), memory, network, server, software or application.
- **Environment** refers to an environment for programming, program execution or both. For example, cloud9 online provides an open programming environment for BeagleBone board for the development of IoT devices; Windows environment for programming and execution of applications; Google App Engine environment for creation and execution of web applications in Python or Java.
- **Platform** denotes the basic hardware, operating system and network, and is used for software applications or services over which programs can be run or developed. A platform may provide a browser and APIs which can be used as a base on which other applications can be run or developed.

- **Edge computing** is a type of computing that pushes the frontier of **computing applications, data and services away from centralized nodes to IoT data generating nodes, that means at logical extremes of the network**. IoT device nodes are pushed by events, triggers, alerts, messages and data is collected for enrichment, storage and computations from the remote centralized database nodes. Pushing the computations from centralized nodes enables the usage of resources at device nodes, which could be a requirement in case of low power lossy networks. The processing can also be classified as edge computing at local cloud, grid or mesh computing. The nodes may be mobile or of a wireless sensor network or cooperative distributed in peer-to-peer and ad-hoc networks.
- **Distributed computing** refers to **computing and usage of resources which are distributed at multiple computing environments over the Internet**. The resources are logically-related, which means communicating among themselves using message passing and transparency concepts, and are cooperating with each other, movable without affecting the computations and can be considered as one computing system (location independent).
- **Service is a software which provides the capabilities and logically grouped and encapsulated functionalities**. A service is called by an application for utilizing the capabilities. A service has a description and discovery methods, such as advertisement for direct use or through a service broker. The service binds to Service Level Agreement (SLA) between service (provider end point) and application (end point). One service can also use another service.

- **Web Service**, according to the W3C definition, is an application identified by a URI, described and discovered using the XML based Web-Service Description Language (WSDL). A web service interacts with other services and applications using XML messages and exchanges the objects using Internet protocols.
- **Service-oriented architecture** consists of components which are implemented as independent services which can be dynamically bonded and orchestrated, and which possess loosely coupled configurations, while the communication between them uses messages. Orchestrating means a process which predefines an order of calling the services (in sequences and in parallel) and the data and message exchanges.
- **Web computing** refers to computing using resources at computing environment of web server(s) or web services over the Internet.
- **Grid computing** refers to computing using the pooled interconnected grid of computing resources and environments in place of web servers.
- **Utility computing** refers to computing using focus on service levels with optimum amount of resources allotted when required and takes the help of pooled resources and environments for hosting applications. The applications utilize the services.

- **Cloud computing** refers to computing using a collection of services available over the Internet that deliver computational functionality on the infrastructure of a service provider for connected systems and enables distributed grid and utility computing.
- **Key Performance Indicator (KPI)** refers to a set of values, usually consisting of one or more raw monitored values including minimum, average and maximum values specifying the scale. A service is expected to be fast, reliable and secure.
- **Localization** means cloud computing content usage is monitored by determining localization of the QoS level and KPIs.
- **Seamless cloud computing** means during computing the content usages and computations continue without any break when the service usage moves to a location with similar QoS level and KPIs. For example, continue using same cloud platform when developer of software shifts.
- **Elasticity** denotes that an application can deploy local as well as remote applications or services and release them after the application usage. The user incurs the costs as per the usages and KPIs.
- **Measurability** (of a resource or service) is something which can be measured for controlling or monitoring and enables report of the delivery of resource or service.

- **Homogeneity** of different computing nodes in a cluster or clusters refers to integration with the kernel providing the automatic migration of the processes from one to other homogeneous nodes. System software on each computing node should ensure same storage representation and same results of processing.
- **Resilient computing** refers to the ability of offering and maintaining the accepted QoS and KPIs in presence of the identified challenges, defined and appropriate resilience metrics, and protecting the service.
- **Scalability** in cloud services refers to the ability using which an application can deploy smaller local resources as well as remotely distributed servers and resources, and then increase or decrease the usage, while incurring the cost as per the usage on increasing scales.
- **Maintainability** in cloud services refers to the storage, applications, computing infrastructure, services, data centres and servers maintenances which are responsibilities of the remotely connected cloud services with no costs to the user.
- **XAAS** is a software architectural concept that enables deployment and development of applications, and offers services using *web* and *SOA*. A computing paradigm is to integrate complex applications and services and use XAAS concept for deploying a cloud platform.
- **Multitenant** cloud model refers to accessibility to a cloud platform and computing environment by multiple users who pay as per the agreed QoS and KPIs, which are defined at separate SLAs with each user. Resource pooling is done by the users but each user pays separately



# Cloud Computing Paradigm

- Cloud computing means a collection of services available over the Internet. Cloud delivers the computational functionality. Cloud computing deploys infrastructure of a cloud-service provider. The infrastructure deploys on a utility or grid computing or web- services environment that includes network, system, grid of computers or servers or data centers.
- Just as we—users of electricity—do not need to know about the source and underlying infrastructure for electricity supply service, similarly, a user of computing service or application need not know how the infrastructure deploys or the details of the computing environment. Just as the user does not need to know Intel processor inside a computer, similarly, the user uses the data, computing and intelligence in the cloud, as part of the services. Similarly, the services are used as a utility at the cloud.
- ***Cloud Platform Services*** , Cloud platform offers the following:
  - Infrastructure for large data storage of devices, RFIDs, industrial plant machines, automobiles and device networks
  - Computing capabilities, such as analytics, IDE (Integrated Development Environment)
  - Collaborative computing and data store sharing



# Cloud Platform Usages

- Cloud platform usages are for connecting devices, data, APIs, applications and services, persons, enterprises, businesses and XAAS.
- The following Equation describes a simple conceptual framework of the Internet Cloud:
- Internet Cloud + Clients = User applications and services with 'no boundaries and no walls'
- An application or service executes on a platform which includes the operating system (OS), hardware and network. Multiple applications may initially be designed to run on diversified platforms (OSs, hardware and networks). Applications and services need to integrate them on a common platform and running environment.
- Cloud storage and computing environment offers a *virtualized environment*, which refers to a running environment made to appear as one to all applications and services, but in fact physically two or more running environments and platforms may be present.

## *Cloud Computing Features and Advantages*

- Essential features of cloud storage and computing are:
  - On demand self-service to users for the provision of storage, computing servers, software delivery and server time
  - Resource pooling in multi-tenant model
  - Broad network accessibility in virtualized environment to heterogeneous users, clients, systems and devices
  - Elasticity
  - Scalability, Maintainability, Homogeneity, Virtualization
  - Interconnectivity platform with virtualized environment for enterprises and provisioning of in-between Service Level Agreements (SLAs)
  - Resilient computing
  - Advanced security
  - Low cost

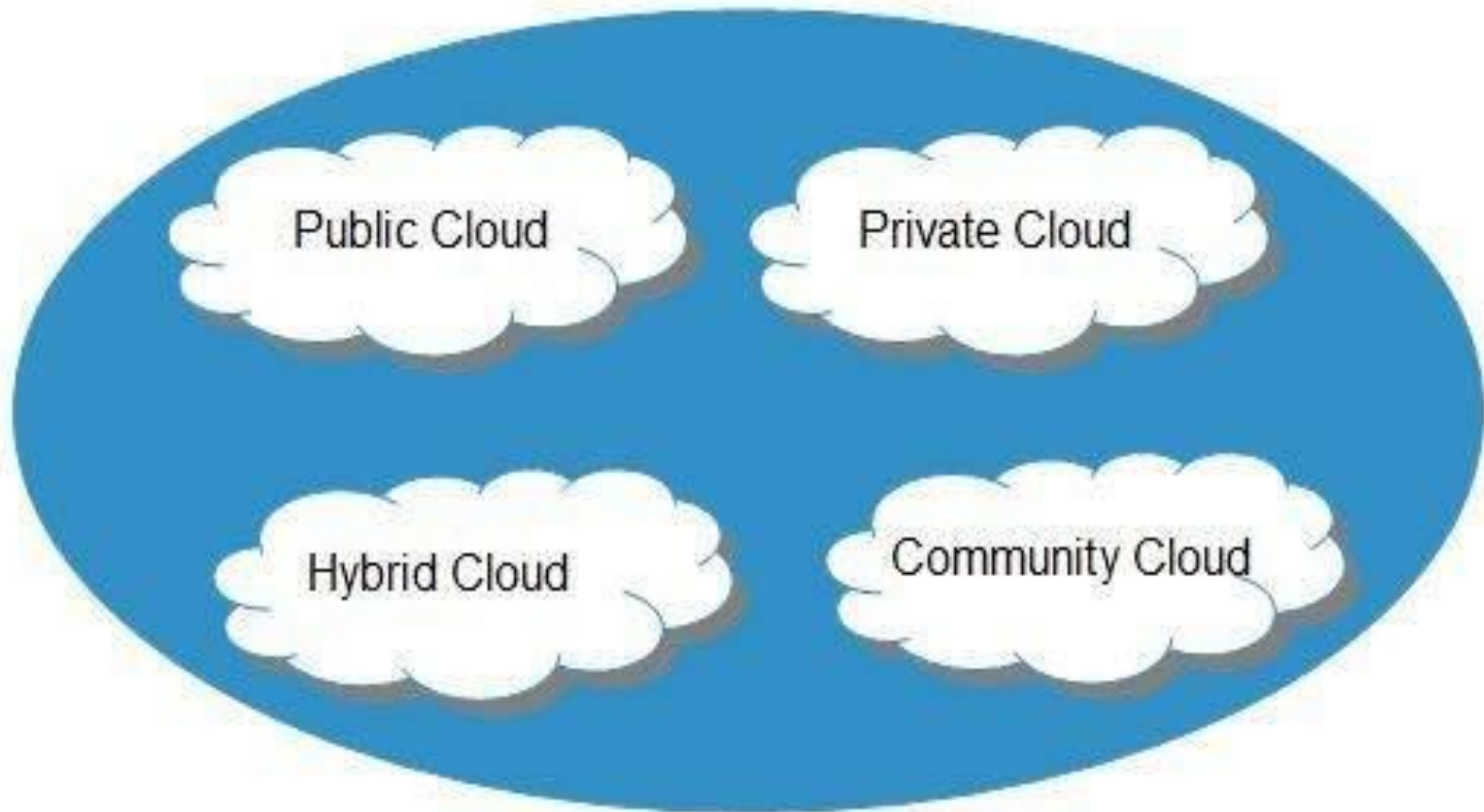
# ***Cloud Computing Concerns***

- Concerns in usage of cloud computing are:
  - Requirement of a constant high-speed Internet connection
  - Limitations of the services available
  - Possible data loss
  - Non delivery as per defined SLA specified performance
  - Different APIs and protocols used at different clouds
  - Security in multi-tenant environment needs high trust and low risks
  - Loss of users' control

# Basic Concepts

- There are certain services and models working behind the scene making the cloud computing feasible and accessible to end users. Following are the working models for cloud computing:
  - Deployment Models
  - Service Models

# Deployment Models



# Cloud Deployment Models

|                 |
|-----------------|
| Private Cloud   |
| Community Cloud |
| Public Cloud    |
| Hybrid Cloud    |

- Four cloud deployment models: **public, private, community or hybrid**
- **Public cloud**: This model is provisioned by educational institutions, industries, government institutions or businesses or enterprises and is open for public use.
- **Private cloud**: This model is exclusive for use by institutions, industries, businesses or enterprises and is meant for private use in the organization by the employees and associated users only.
- **Community cloud**: This model is exclusive for use by a community formed by institutions, industries, businesses or enterprises, and for use within the community organization, employees and associated users. The community specifies security and compliance considerations.
- **Hybrid cloud**: A set of two or more distinct clouds (public, private or community) with distinct data stores and applications that bind between them to deploy the proprietary or standard technology.
- Cloud platform architecture is a virtualized network architecture consisting of a cluster of connected servers over the data centers and Service Level Agreements (SLAs) between them.

# Service Models

- Cloud computing is based on service models. These are categorized into **Three** basic service models which are -
- Infrastructure-as-a-Service (IaaS)
- Platform-as-a-Service (PaaS)
- Software-as-a-Service (SaaS)
- Data-as-a-Service (DaaS)
- **Anything-as-a-Service (XaaS)** is yet another service model, which includes Network-as-a-Service, Business-as-a-Service, Identity-as-a-Service, Database-as-a-Service or Strategy-as-a-Service.

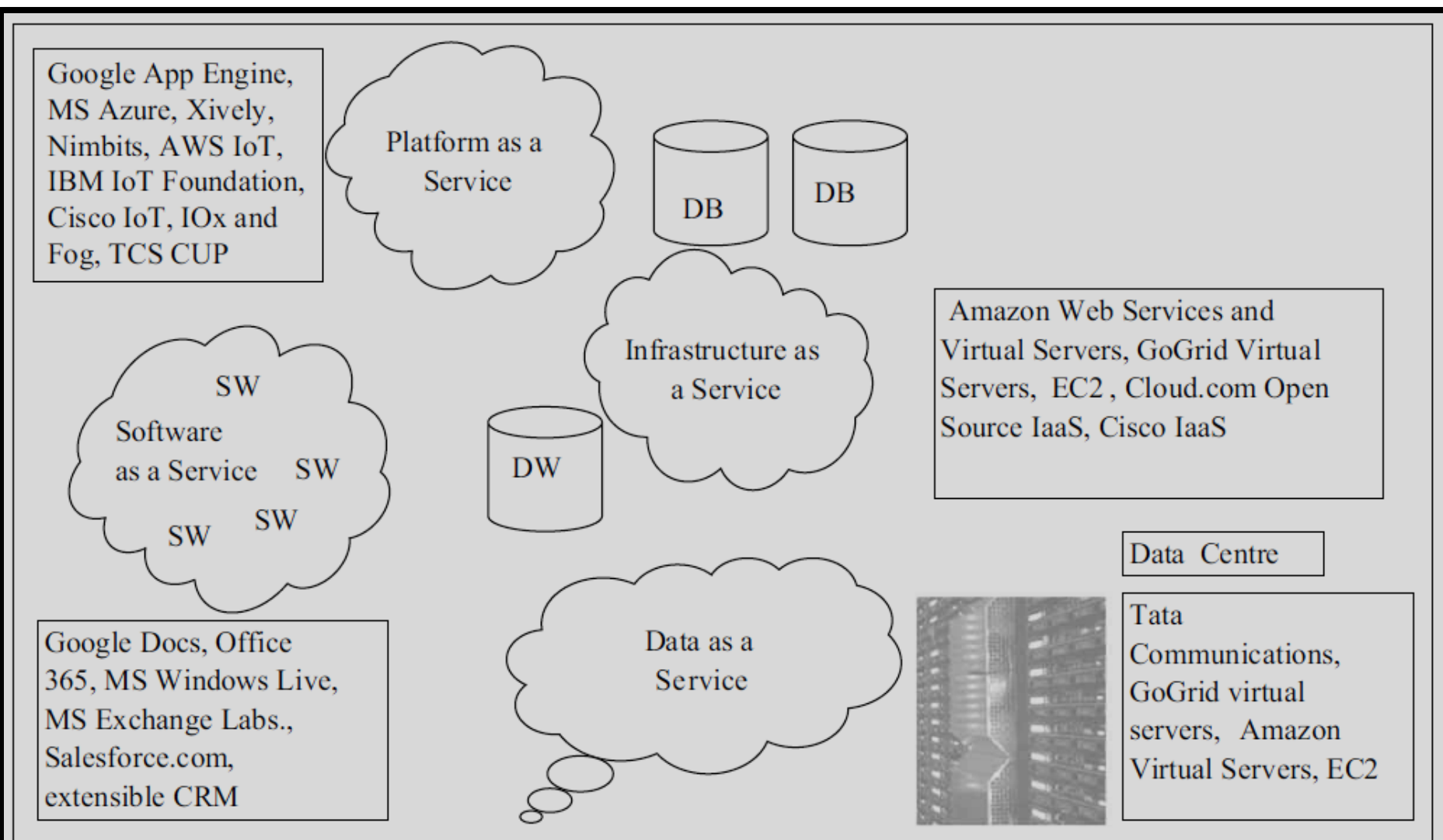
# EVERYTHING AS A SERVICE AND CLOUD SERVICE MODELS

- Cloud connects the devices, data, applications, services
- Describe cloud computing service models in a software architectural concept, persons and business. Cloud services can be considered as distribution service—a service for linking the resources (computing functions, data store, processing functions, everything as a service (XAAS) networks, servers and applications) and for provision of coordinating between the resources.
- Cloud computing can be considered by a simple equation:
- **Cloud Computing = SaaS + Paas + IaaS + DaaS**



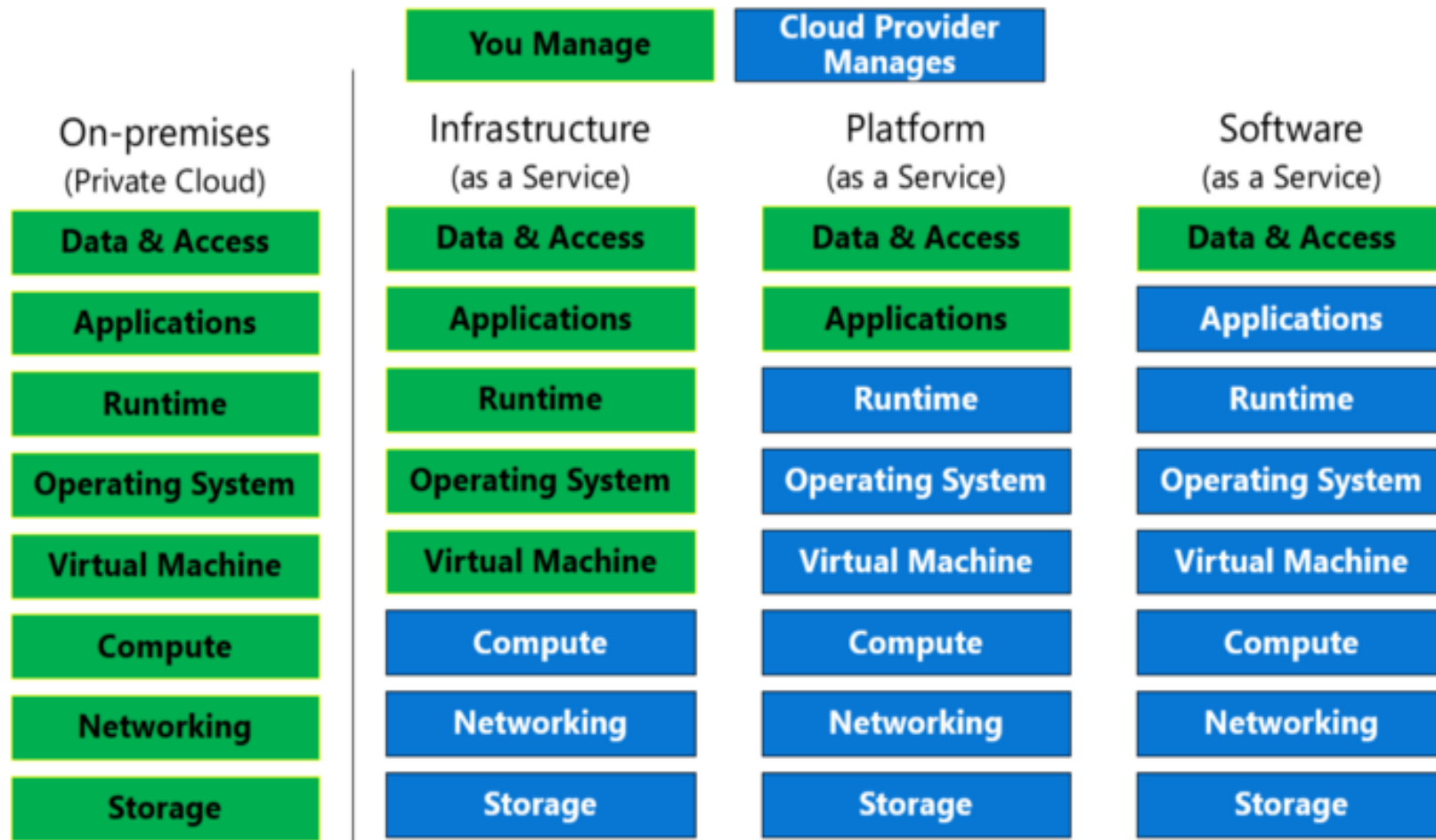
- **SaaS means Software as a Service.** The software is made available to an application or service on demand. SaaS is a service model where the applications or services deploy and host at the cloud, and are made available through the Internet on demand by the service user. The software control, maintenance, updation to new version and infrastructure, platform and resource requirements are the responsibilities of the cloud service provider.
- **PaaS means Platform as a Service.** The platform is made available to a developer of an application on demand. PaaS is a service model where the applications and services develop and execute using the platform (for computing, data store and distribution services) which is made available through the Internet on demand for the developer of the applications. The platform, network, resources, maintenance, updation and security as per the developers' requirements are the responsibilities of the cloud service provider.

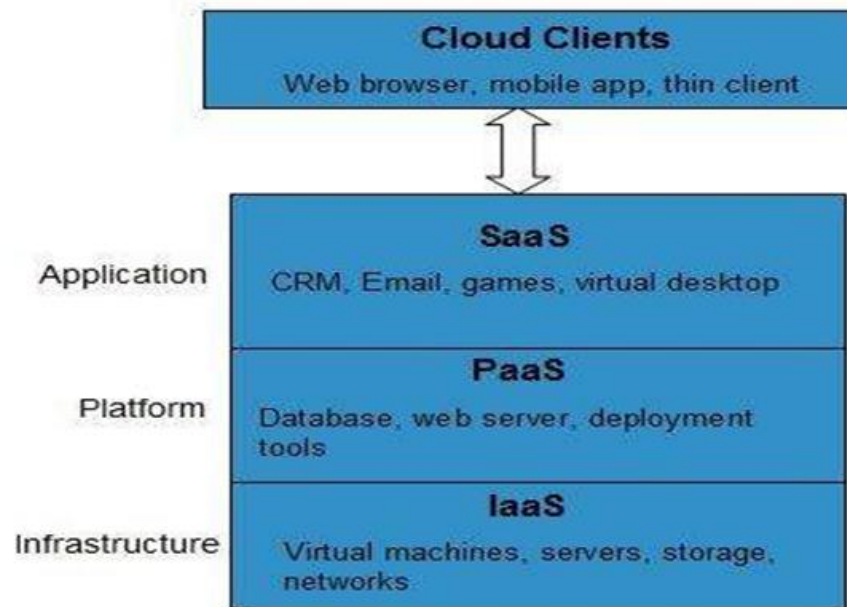
- **IaaS means Infrastructure as a Service.** The infrastructure (data stores, servers, data centres and network) is made available to a user or developer of application on demand. Developer installs the OS image, data store and application and controls them at the infrastructure. IaaS is a service model where the applications develop or use the infrastructure which is made available through the Internet on demand on rent (pay as per use in multi-tenancy model) by a developer or user. IaaS computing systems, network and security are the responsibilities of the cloud service provider.
- **DaaS means Data as a Service.** Data at a data center is made available to a user or developer of application on demand. DaaS is a service model where the data store or data warehouse is made available through the Internet on demand on rent (pay as per use in multi tenancy model) to an enterprise. The data center management, 24×7 power, control, network, maintenance, scale up, data replicating and mirror nodes and systems as well as physical security are the responsibilities of the data center service provider.



**Figure 6.2** PaaS, SaaS, IaaS and DaaS Cloud Service model

[DW—Data Warehouses; DB—Databases; EC2—Elastic Computing Cloud; SW—Software; MS—Microsoft; CRM—Customer Customer Relations Management]

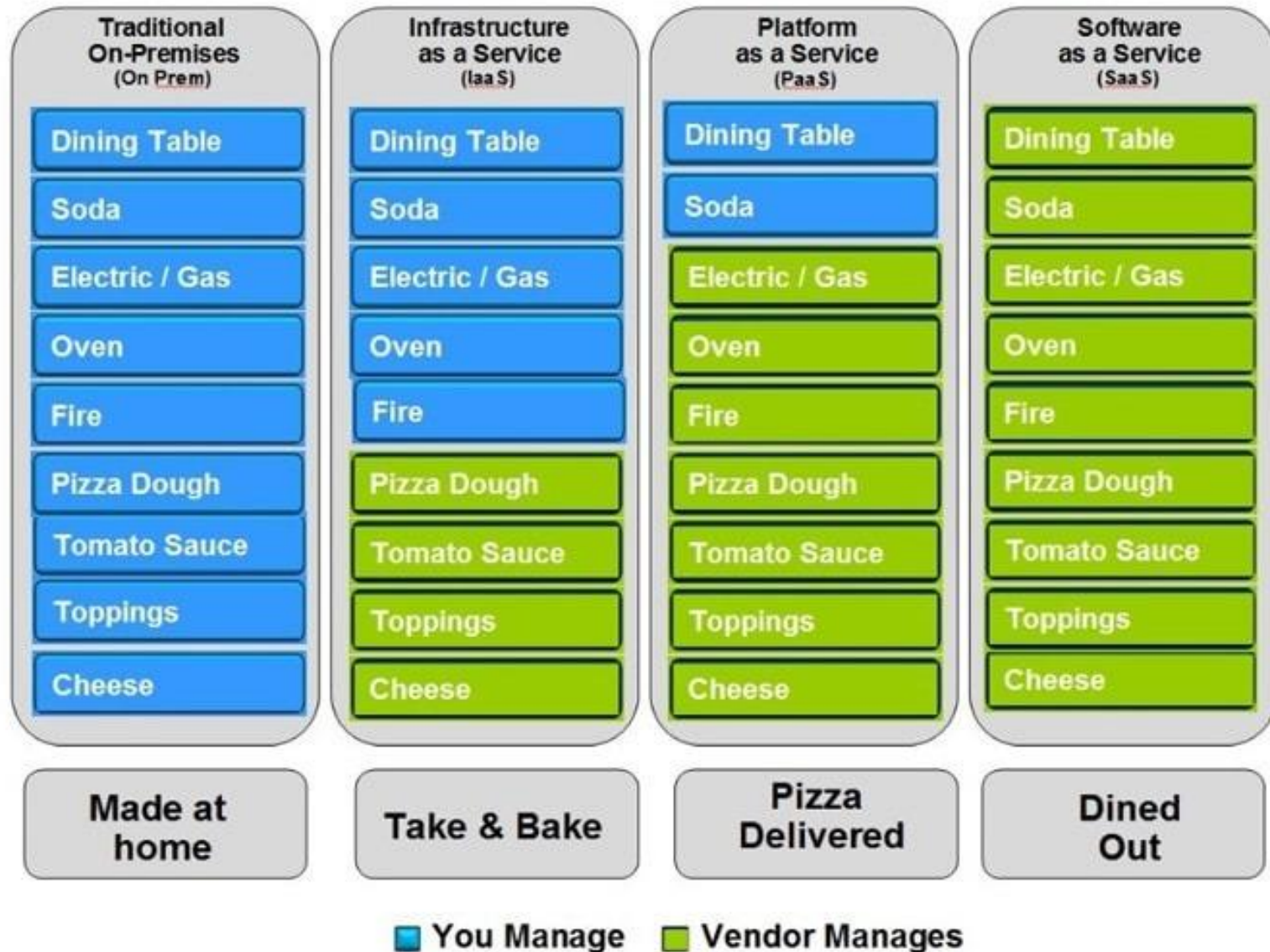




### Examples of Azure Service Types(IaaS, PaaS, SaaS)

| IaaS                   | PaaS                    | SaaS                 |
|------------------------|-------------------------|----------------------|
| Azure virtual machines | Azure App Service       | Outlook email        |
| Azure Storage accounts | Azure SQL databases     | Calendar             |
|                        | Azure Cosmos DB         | Microsoft Office 365 |
|                        | Azure Synapse Analytics |                      |

# Pizza as a Service



# Virtualization

- Virtualization is a partitioning of single physical server into multiple logical servers. Once the physical server is divided, each logical server behaves like a physical server and can run an operating system and applications independently. Many popular companies' like **VMware** and **Microsoft** provide virtualization services, where instead of using your personal PC for storage and computation, you use their virtual server. They are fast, cost-effective and less time consuming.
- Virtualization is mainly used for three main purposes
  - 1) Network Virtualization**
  - 2) Server Virtualization**
  - 3) Storage Virtualization**

# Virtualization

- **Network Virtualization:** It is a method of combining the available resources in a network by splitting up the available bandwidth into channels, each of which is independent from the others and each channel is independent of others and can be assigned to a specific server or device in real time.
- **Storage Virtualization:** It is the pooling of physical storage from multiple network storage devices into what appears to be a single storage device that is managed from a central console. Storage virtualization is commonly used in storage area networks (SANs).
- **Server Virtualization:** Server virtualization is the masking of server resources like processors, RAM, operating system etc, from server users. The intention of server virtualization is to increase the resource sharing and reduce the burden and complexity of computation from users.



## IoT/M2M

Device/  
Sensor  
Network

Device/  
Sensor  
Network

Device/  
Sensor  
Network

## IoT/M2M Local Data

Communication  
Gateway

Internet Firewall

Database Firewall

Collect + Storage for  
the applications,  
services, enterprise and  
business processes  
and Intelligence

Collect  
from  
things

Cloud  
Infrastructure  
and Services

User  
Application

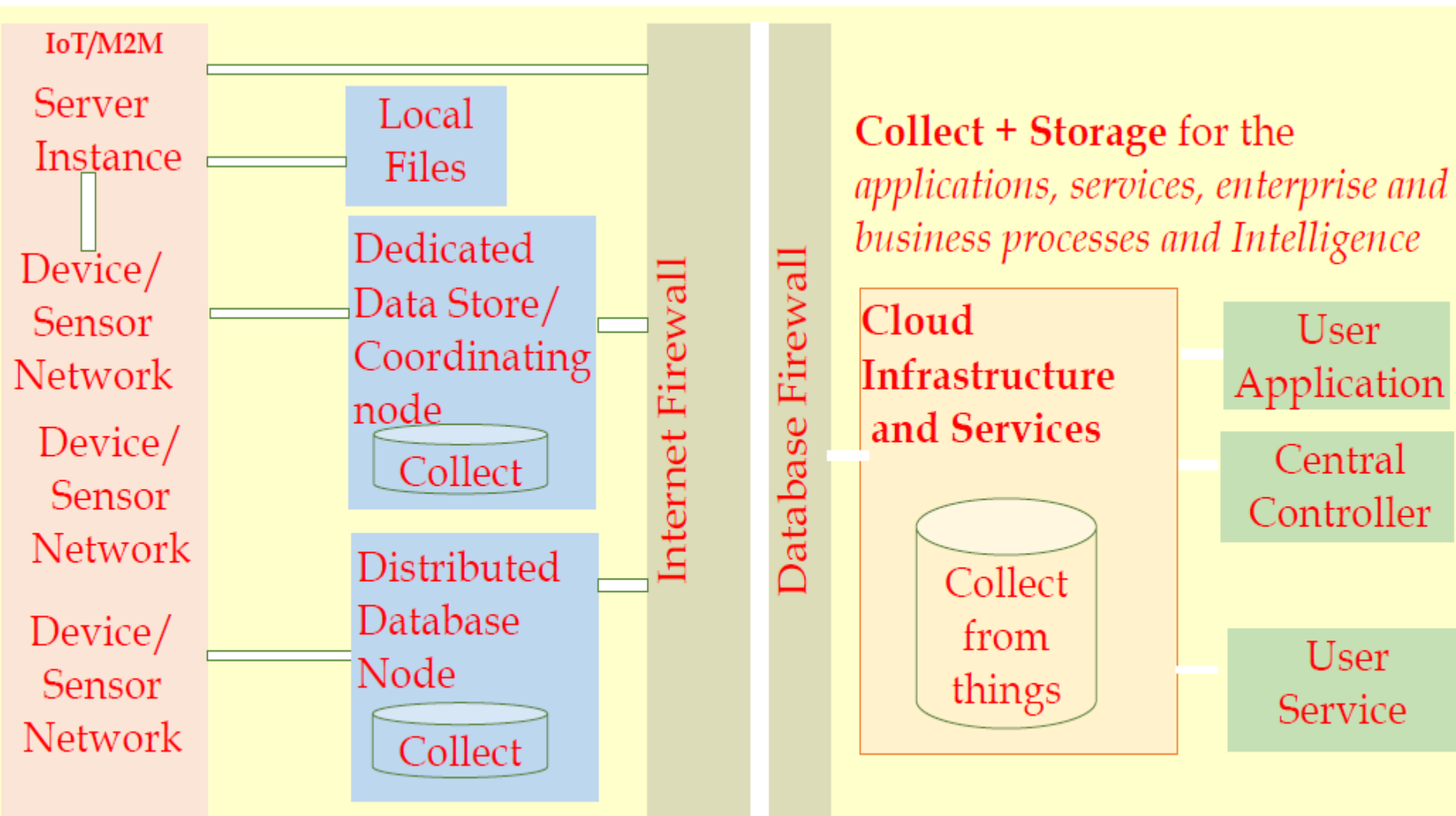
User  
Service

User  
Application

• Devices/Sensor Networks Data Collection at the Cloud Infrastructure and Services

## IoT CLOUD-BASED SERVICES USING THE XIVELY, NIMBITS AND OTHER PLATFORMS

- A user is an application or service. The user obtains responses or feeds from the application or service provides for data collection, data points, messages and calculation objects. The service also provisions for the (Pachube/COSM) and Nimbits generation and communication of alerts, triggers and feeds to the user. A server can be deployed at the edges (device nodes) which communicates the feeds to the cloud service.
- Cloud service deploys a server instance at the IoT sensor network
- A data point generates a new data each time a new sensed value (data) is recorded. A feed means a set of data points or objects or data streams or messages which generate and communicate after application of rules for filter, calculation, compaction, fusion, compression, analysis or aggregation.



Devices/Sensor Networks Data Collection at the Devices-network Web Server and at the Cloud infrastructure

# IoT Cloud-based Data Collection, Storage, Computing using Xively

- **Xively** (formerly known as **Cosm** and **Pachube**) is an Internet of Things (IoT) platform owned by **Google**. Xively offers product companies a way to connect products, manage connected devices and the data they produce, and integrate that data into other systems. It is pronounced "**zively**" (rhymes with lively).
- **Pachube** is a platform for data capture in real-time over the Internet. **Cosm** is a changed domain name, where using a concept of console, one can monitor the feeds. **Xively** is the latest domain name. Xively is an **open source platform** for Arduino which is an open-source prototyping platform that provides connectivity with web deploying Internet.
- Xively is a commercial **PaaS** for the IoT/M2M. It is used as a data aggregator and data mining website often integrated into the Web of Things. Xively is an IoT PaaS for services and business services. The platform supports the REST, WebSockets and MQTT protocols and connects the devices to Xively Cloud Services. There are native SDKs for Android, Arduino, ARM mbed, Java, PHP, Ruby and Python languages. Developers can use the workflow of prototyping, deployment and management through the tools provided by Xively.

# Xively PaaS services offers the following features

- It enables services, business services platform which connects the products, including collaboration products, Rescue, Boldchat, join.me, and operations to the Internet.
- Data collection in real-time over the Internet.
- Data visualization for data of connected sensors to IoT devices.
- Graphical plots of collected data.
- It generates alerts.
- Access to historical data.
- It supports Java, Python and Ruby, and Android platform.
- It generates feeds which can be real-world objects of own or others.
- It supports the ARM mBedTM-based, Arduino-based and other hardware-platform- based IoT devices, and HTTP-based APIs which are easy to implement on device hardware acting as clients to Xively web services, and connect to the web service and send data.
- It supports REST.

- Xively APIs enable interface with Python, HTML5, HTML5 server, tornado, webSocket, webSocket Server and WebSockets and interface with an RPC (Remote Procedure Call).
- Devices get an online presence. For example, an Arduino climate logging client that can be accessed via a browser or mobile using Xively. Arduino is an open-source prototyping platform which is based on ATmega microcontroller for embedded applications and IoT/ M2M. Another platform is mBed™. It is based on ARM Cortex-microprocessors. The mBed™ is also for embedded applications and IoT/M2M. Xively is an open source platform that enables IoT devices or sensors network to connect the sensor data to the web. Xively provides services for logging, sharing and displaying sensor data of all kinds using an HTTP based API.
- Xively is based on the concept of users, feeds, data streams, data points and triggers. A feed is typically a single location (e.g. a house), and data streams are of individual sensors associated with that location (for example, ambient lights, temperatures, power consumption).

- ***Pull or Push (Automatic or Manual Feed) Methods for IoT Devices Data :*** Xively provides two modes for data capture, viz. a pull method (automatic feed type) where data is collected from an http server, and a push method (manual feed type) where data is written to Xively using an http client.
- ***Data Formats and Structures:*** Number of data formats and structures enable interaction, data collection and services with Xively. The support exists for JSON, XML and CSV (a tabular, spreadsheet, excel, data numbers and text with a comma-separated values in file).
- ***Private and Public Data Access:*** A free account supports up to 10 sensor feeds updated in near real time and the data is stored for up to 3 months. Interactive graphs provided by the service can be embedded onto the mobile giving the application access to data anywhere. The application can even use other user's data feeds as inputs.
- ***Data streams, Data points and Triggers:*** Xively enables data streams, data points and triggers. Data stream means continuous sensed data flow over the Internet. Data points mean data values, whereas trigger means action on a state change. For example, ambient light sensing below a threshold value near a group of streetlights or data point reaching at threshold, maximum or minimum or set point.

# Xively Cloud : Dashboard

**Device Name**

Smart Plant

**Device Description** optional

Plant Health monitor

**Privacy** You own your data, we help you share it. [more info](#)

☐ **Private Device**  
You use API keys to choose if and how you share a device's data.

☒ **Public Device**  
You agree to share a device's data under the [CC0 1.0 Universal license](#). The Device's data is indexed by major search engines, and its Feed page is publicly viewable.

**To link a new  
device**

Xively Cloud has become a Google Cloud now



IoT device can send data to a channel

using Xively APIs

For each channel, you can create one or more **triggers**.

**Triggers** are used for integration with third party applications

Xively devices can have one or more **channels**

Each channel enables **bi directional** communication between IoT devices and Xively Cloud.

The screenshot shows the Xively web interface for an 'Atmel Gateway' device. The top navigation bar includes links for PLATFORM, SERVICES, ECOSYSTEM, NEWS & EVENTS, DEV CENTER, and WEB TOOLS. The main content area is divided into several sections:

- Device Status:** Shows 'Activated' with a 'Deactivate' button and a 'Deploy' button.
- Device Information:** Lists fields like Product ID, Product Secret, Serial Number, and Activation Code.
- Feed Information:** Displays the Feed ID (106526775), Feed URL, and API Endpoint.
- Add Channels to your Device!** A section with a blue arrow pointing down and a '+ Add Channel' button.
- Request Log:** A section titled 'Waiting for requests' with a 'Pause' button.
- API Keys:** A section showing an 'Auto-generated Atmel Gateway device key for feed 106526775' and a long alphanumeric key string.

New device Details

A **trigger Specification** includes a channel to which the trigger corresponds, trigger condition and an **HTTP POST URL** to which the request is sent when trigger fires.

# Xively Cloud : Python Program for sending data to a Xively Cloud

Background: temperature monitoring using Raspberry Pi and the measured data is sent to Xively cloud. Raspberry Pi runs a controller program that reads the sensor data(e.g.DHT11) every few seconds and sends data to the cloud.

A Python library, needed to execute the code

```
import time
import datetime
import requests
import xively
```

```
from random import randint
global temp_datastream
```

```
#Initialize Xively Feed
```

```
FEED_ID = "YOURFEEDID" #enter your device's <FEED_ID>
API_KEY = "YOURAPIKEY" #enter authenticated <API Key>
```

```
api = xively.XivelyAPIClient(API_KEY)
```

```
#function to read temperature
```

```
def readTempSensor():
    #Return random value
    return randint(20,30)
```

```
#Controller Main function
```

```
def runController():
    global temp_datastream
    temperature = readTempSensor()
    temp_datastream.current_value = temperature
    temp_datastream.at = datetime.datetime.utcnow()

    print("Updating Xively feed with Temperature : %s" %temperature)
    try:
        temp_datastream.update()
    except requests.HTTPError as e:
        print("HTTPError(0) : 1".format(e.errno, e.strerror))
```

Feed object is created by providing API key and Feed ID. Then a channel named **temperature** is created

```
#Function to get existing or
#Create new xively data stream for temperature
```

```
def get_tempdatastream(feed):
```

```
    try:
```

```
        datastream = feed.datastreams.get ("temperature")
        return datastream
```

```
    except:
```

```
        datastream = feed.datastreams.create("temperature",tags="temperature")
        return datastream
```

```
#Controller setup function
```

```
def setupController():
```

```
    global temp_datastream
    feed = api.feeds.get(FEED_ID)
    feed.location.lat="30.733315"
    feed.location.lon="76.779418"
    feed.tags="Weather"
    feed.update()
```

```
    temp_datastream = get_tempdatastream(feed)
    temp_datastream.max_value = None
    temp_datastream.min_value = None
```

```
setupController()
```

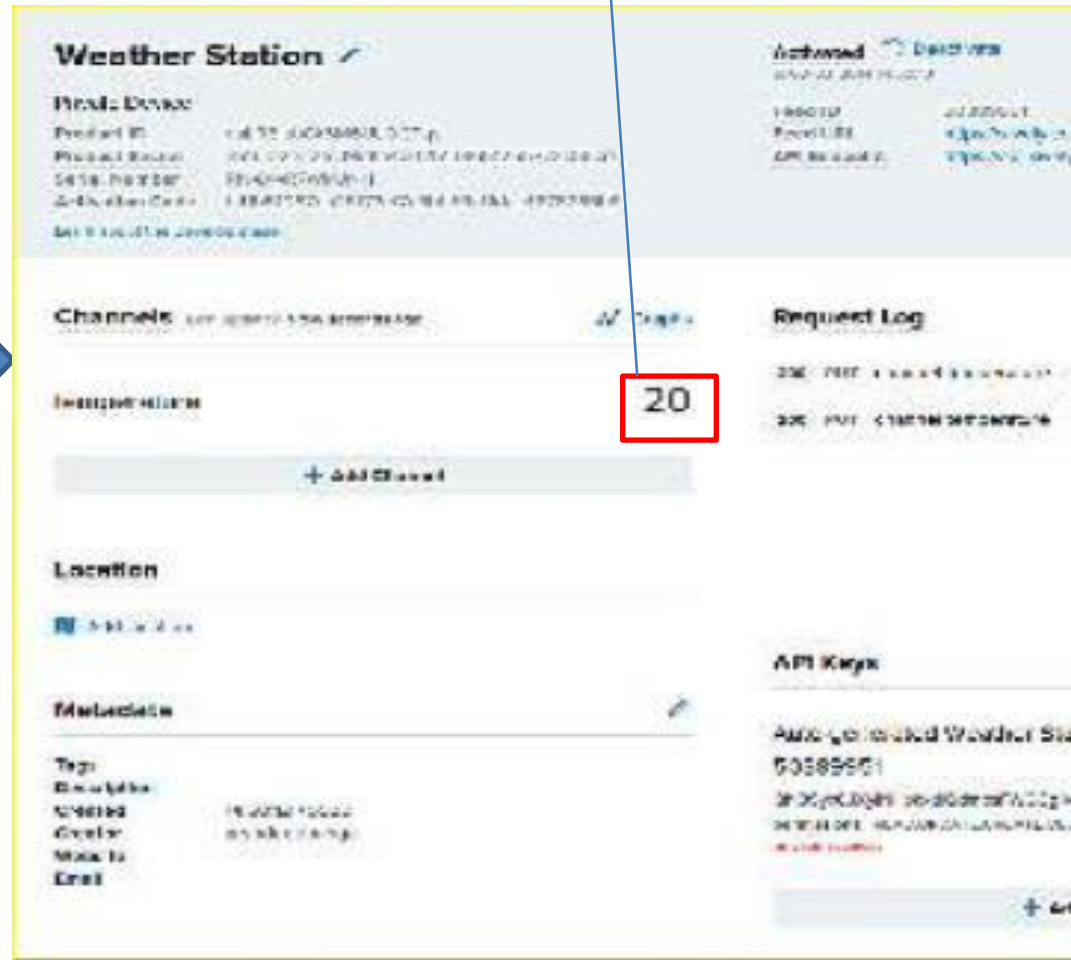
```
while True:
```

```
    runController()
    time.sleep(10)
```

Temperature data is sent to temperature c the **runcontroller()** function every 10second Xively dashboard.

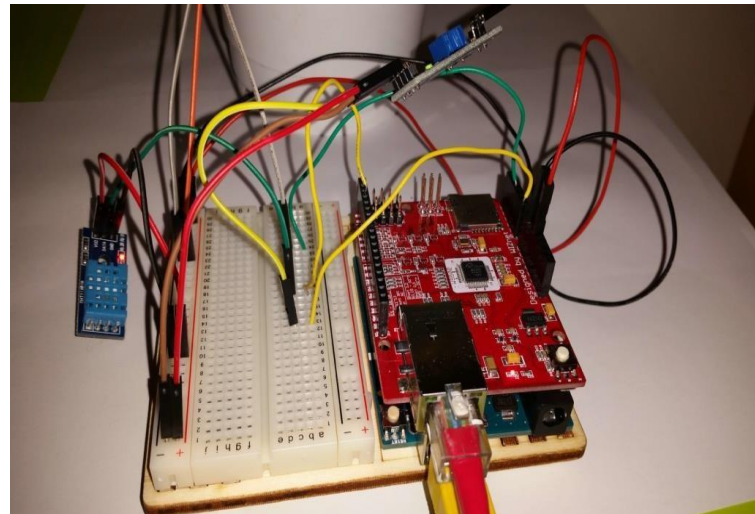
# Xively Cloud : Python Program for sending data to a Xively Cloud

Temperature read Data is updated as 20



The screenshot displays the Xively Weather Station interface. The 'Channels' section shows a temperature reading of 20, which is highlighted with a red box. A blue arrow points from the text 'Temperature read Data is updated as 20' to this value. The interface also includes sections for 'Location', 'Metadata', and 'API Keys'.

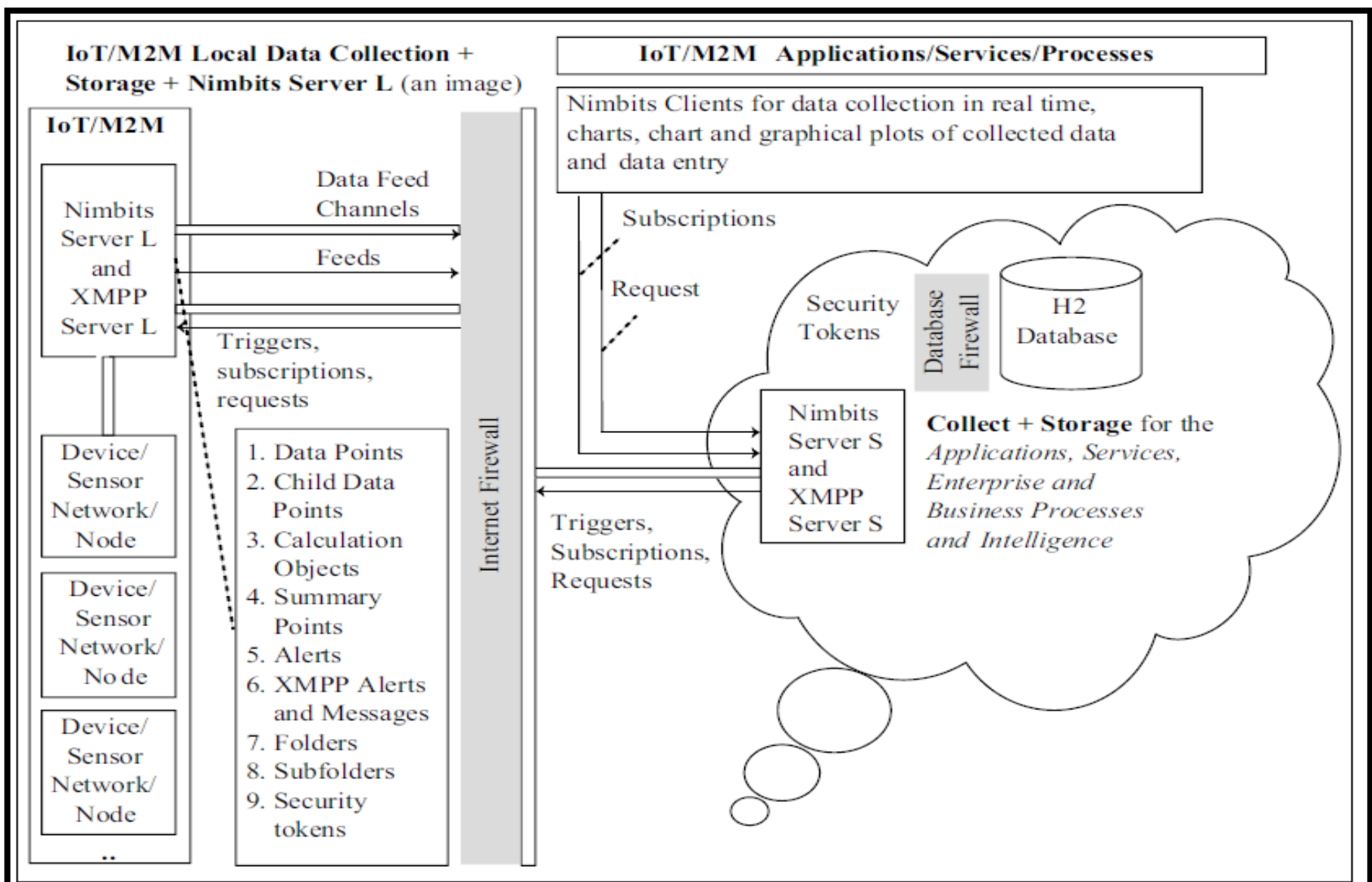
| Channel     | Value |
|-------------|-------|
| Temperature | 20    |



# IoT Cloud-based Data Collection, Storage and Computing Services Using **Nimbits**

- Nimbits enables IoT on an open source distributed cloud. Nimbits cloud PaaS deploys an instance of Nimbits Server at the device nodes. Nimbits functions as an M2M system data store, data collector and logger with access to historical data. Nimbits architecture is a cloud-based Google App Engine. Nimbits server is a class hierarchy `com.nimbits.server.system.ServerInfo` of `java.lang.Object`. Nimbits PaaS services offer the following features:
- Edge computing locally on embedded systems, built up of local applications. It runs the rules and pushes important data up to the cloud running when connected over Internet and an instance of Nimbits Server hosts at the device nodes which is then enabled.
- It supports multiple programming languages, including Arduino, new Arduino library, push functions from Arduino cloud, JavaScript, HTML or the Nimbits.io Java library.
- Nimbits server functions as a backend platform. Nimbits data point can relay data between the software systems, or hardware devices such as Arduino, using the cloud as a backend.
- An open source Java library called `nimbits.io` enables easy development of JAVA, web and Android solutions (Nimbits data, alerts, messages on mobile).
- It provides a rule engine for connecting sensors, persons and software to the cloud and one another. Rules can be for calculations, statistics, email alerts, xmpp messages, push notifications and more.

- It provides a data logging service and access, and stores the historical data points and data objects.
- Storage in any format that can be serialised into a string, such as JSON or XML.
- It filters the noise and important changes sent to another larger central instance.
- It processes a specific type of data and can store it.
- Time- or geo-stamping of the data.
- Nimbits clients provide over Internet, data collection in real time, charts, chart and graphical plots of collected data and data entry.
- Data visualisation for data of connected sensors to IoT devices.
- Supports the alerts subscription, generation and sending in real time over the Internet.
- It creates streams of data objects and stores them in a data point series.
- Data accessibility and monitoring from anywhere, and is used to shape the behaviour of connected devices and software.
- It supports the mBedTM, Arduino, Raspberry Pi based and other hardware platform based IoT devices.
- Web service APIs are easy to implement on device hardware acting as clients to Nimbits web services, and connect to the web service and send data.
- It deploys software on Google App Engine, any J2EE server on Amazon EC2 or on a Raspberry Pi.



**Figure 6.3** Connected devices, sensor nodes, network data points, Nimbits server, deployment at the device network nodes, and networked with the Nimbits Server (PaaS, SaaS and IaaS services) at cloud for applications and services.

# Data Points

- A data point means a collected value of a sensor in a group of sensors. Data points organize the data in a number of ways. For example, points can have child points (child points mean subpoints; for example, if light level is a data point then light on or off is a child point and light level above or below the threshold can be another child point.)
- Points can be in the folders. The folders can go as deep as like in a tree (Tree means a folder having several subfolders, a subfolder having several subfolders, till the leaf subfolder.)
- Any type of document can upload and organize them with the points. Files can be shared publicly or with the connections.
- A subscription data feed is a special point for each user that logs system messages, events, alerts from other points which are subscribed by a service and more.
- A user can create a data feed channel which shows the system events and messages that also shows data alerts which are subscribed to show up in the feed. The user can subscribe to the data point of other users also, and configure the subscription(s) to send messages to the feed. The user can observe the idle, high or low alerts here in real time. The user data feed is just another Nimbits data point.



# Using Advanced Features

- An application can create a connection to another Nimbits application or service. The application sends an invitation and if the invitee approves, then the application can see invitee's points and data in their tree (if they set the objects permission level to public or connection viewable).
- Nimbits 3.6.6 introduced H2 database engine. Nimbits 3.8.10 includes H2 database engine. H2 is Java SQL database. APIs are in pure Java, The main features of H2 are:
  - Very fast, open source, JDBC API
  - Embedded and server modes; in-memory databases
  - Encrypted database
  - ODBC driver
  - Full text search
  - Multi-version concurrency
  - Browser-based console application
  - Small footprint (around 1.5 MB jar file size)
- MySQL is not in pure Java and have no provisions for in-memory or encrypted databases. Footprint (DLL) is nearly 4 MB. (JAR means Java archive while DLL stands for dynamically linked library.)
- Security Tokens : Nimbits 3.9.6 provides security tokens in a new way.
- Breakthrough Performance and Data Integrity. Nimbits server 3.9.10 version launched in June 2015 provisions for the breakthrough performance and data integrity.
- Alerts : A filter means applying some rule to get new data for a data point. The filter item in the tree called "ah" is for XMPP alerts . A user application can now have many
- JIDs (Jabber IDs) for a single point—alerts and messages can be sent over XMPP using the custom JID of points.



# Jabbing

- Means pushing the alerts or messages down quickly or pushing repeatedly. Each type of alert or message is assigned a Jabber ID, called JID. Each JID consists of three main parts, viz. the node identifier (optional), domain identifier (required) and resource identifier (optional). A JID is written in notations as <JID>:= [<node>"@"<domain>["/"<resource>]. Subscriptions
- A user can create many subscriptions for a single point. It may subscribe to one of the points, other user, or anyone's public point to get the alerts. The user gets alerts when the point goes into an alarm state, or receives new data. Alarm state means reaching preset value. For example, a pressure boiler reaching critical pressure necessitating action or water reaching boiling point necessitating action.

Using Public Cloud IoT Platforms, Many cloud PaaS and SaaS platforms are now available for IoT. Table 6.1 gives the examples of cloud-based platforms.

| Cloud Platform | Features   |
|----------------|--|
| Spark          | Distributed, cloud-based IoT operating system and web-based IDE includes a command-line interface, support for multiple languages and libraries for working with many different IoT devices  |
| OpenIoT        | Open source middleware, enables communication with sensor clouds and enables cloud-based sensing as a service and developed use cases for smart agriculture, intelligent manufacturing, urban crowd sensing, smart living and smart campuses                                 |
| Device Hub     | Open source backbone for IoT, a cloud-based service that stores IoT-related data, provides data visualizations, allows web-page-based console to control IoT devices, enables developers to create applications such as tracking of vehicular data, monitoring weather data. |

# References

- [https://www.tutorialspoint.com/cloud\\_computing/cloud\\_computing\\_overview.htm](https://www.tutorialspoint.com/cloud_computing/cloud_computing_overview.htm)
- <https://azure.microsoft.com/en-in/overview/what-is-cloud-computing/>
- <https://www.w3schools.in/cloud-computing/cloud-computing/>