# CLOUD PLATFORMS:

Although the term “cloud” often gives cloud computing a somewhat mystical connotation, in reality, it isn’t all that different from traditional computing architecture. Cloud computing still relies upon the same physical server hardware that forms the backbone of any computer network. The difference is that cloud architecture makes the processing power and storage capacity of that hardware available over the internet. This allows cloud providers to use servers distributed around the world to form a consolidated, powerful computing network that can be accessed from any internet connection.

Cloud computing has become a common term over the last decade, but the service sometimes creates confusion. With all the new cloud options and the phrase “as a service” seemingly tacked onto everything imaginable, it’s helpful to take a step back and look at the differences between the main types of cloud deployment and the different types of cloud computing services.

# Types of Cloud Deployment

Cloud deployment describes the way a cloud platform is implemented, how it’s hosted, and who has access to it. All cloud computing deployments operate on the same principle by virtualizing the computing power of servers into segmented, software-driven applications that provide processing and storage capabilities.

## Public Cloud

Some public cloud examples include those offered by Amazon, Microsoft, or Google. These companies provide both services and infrastructure, which are shared by all customers. Public clouds typically have massive amounts of available space, which translates into easy scalability. A public cloud is often recommended for software development and collaborative projects. Companies can design their applications to be portable, so that a project that’s tested in the public cloud can be moved to the private cloud for production. Most cloud providers package their computing resources as part of a service. Public cloud examples range from access to a completely virtualized infrastructure that provides little more than raw processing power and storage (Infrastructure as a Service, or IaaS) to specialized software programs that are easy to implement and use (Software as a Service, or SaaS).

The great advantage of a public cloud is its versatility and “pay as you go” structure that allows customers to provision more capacity on demand. On the downside, the essential infrastructure and operating system of the public cloud remain under full control of the cloud provider. Customers may continue to use the platform under the terms and conditions laid out by the provider, but they may have difficulty repatriating their assets if they want to change providers. Should the provider go out of business or make significant changes to the platform, customers could be forced to make significant infrastructure changes on short notice.

## Private Cloud

Private clouds usually reside behind a firewall and are utilized by a single organization. A completely on-premises cloud may be the preferred solution for businesses with very tight regulatory requirements, though private clouds implemented through a colocation provider are gaining in popularity. Authorized users can access, utilize, and store data in the private cloud from anywhere, just like they could with a public cloud. The difference is that no one else can access or utilize those computing resources. Private cloud solutions offer both security and control, but these benefits come at a cost. The company that owns the cloud is responsible for both software and infrastructure, making this a less economical model than the public cloud.

The additional control offered by a private cloud makes it easier to restrict access to valuable assets and ensures that a company will be able to move its data and applications where it wants, whenever it wants. Furthermore, since the private cloud isn’t controlled by an outside vendor, there’s no risk of sudden changes disrupting the company’s entire infrastructure. A private cloud solution will also not be affected by a public cloud provider’s system downtime. But private clouds also lack the versatility of public clouds. They can only be expanded by adding more physical compute and storage capacity, making it difficult to scale operations quickly should the business need arise.

## Hybrid Cloud

Hybrid clouds combine public clouds with private clouds. They are designed to allow the two platforms to interact seamlessly, with data and applications moving seamlessly from one to the other.

The primary advantage of a hybrid cloud model is its ability to provide the scalable computing power of a public cloud with the security and control of a private cloud. Data can be stored safely behind the firewalls and encryption protocols of the private cloud, then moved securely into a public cloud environment when needed. This is especially helpful in the age of big data analytics, when industries like healthcare must adhere to strict data privacy regulations while also using sophisticated algorithms powered by artificial intelligence (AI) to derive actionable insights from huge masses of unstructured data.

There are two commonly used types of hybrid cloud architecture. Cloudbursting uses a private cloud as its primary cloud, storing data and housing proprietary applications in a secure environment. When service demands increase, however, the private cloud’s infrastructure may not have the capacity to keep up. That’s where the public cloud comes in. A cloudbursting model uses the public cloud’s computing resources to supplement the private cloud, allowing the company to handle increased traffic without having to purchase new servers or other infrastructure.

The second type of hybrid cloud model also runs most applications and houses data in a private cloud environment, but outsources non-critical applications to a public cloud provider. This arrangement is common for organizations that need to access specialized development tools (like Adobe Creative Cloud), basic productivity software (like Microsoft Office 365), or CRM platforms (like Salesforce). Multi-cloud architecture is often deployed here, incorporating multiple cloud service providers to meet a variety of unique organizational needs.

# 3 Different Types of Cloud Computing Services

All public cloud computing services are built upon the same conceptual framework of remote infrastructure powered by servers housed in a data center. Since there are so many similarities between them, it’s helpful to think of cloud computing as a pyramid comprised of three layers. Each layer is more specialized than the one below it, but it’s built upon the same basic structure. The lower layers are much broader, representing their versatility, customizability, and wide range of applications, while the upper layers are narrower because they’re purpose-built for a specific task.

## Infrastructure as a Service (IaaS)

As the foundation of the cloud computing pyramid, IaaS is the most comprehensive and flexible type of cloud service available. Essentially, it provides a completely virtualized computing infrastructure that is provisioned and managed over the internet. An IaaS provider manages the physical end of the infrastructure (servers, data storage space, etc) in a data center, but allows customers to fully customize those virtualized resources to suit their specific needs. With IaaS, the customer can purchase, install, configure, and manage any software they need to use, including things like operating systems, middleware, applications, business analytics, and development tools. Highly scalable, companies only pay for the infrastructure they use, allowing them to scale their computing needs as needed without having to build out additional capacity.

IaaS eliminates the capital expense of building up in-house infrastructure. It’s a great option for small companies and startups that don’t have the resources to purchase the hardware and software needed to create their own network internally. It also takes the day-to-day burdens of managing computing infrastructure off the hands of IT departments, freeing them to focus on core business drivers instead of troubleshooting. Since the IaaS provider continuously updates their system with the latest software and update patches, it’s easier to get new programs and applications up and running. IaaS provides the latest in security protections and usually offers services like disaster recovery to go along with their uptime reliability SLAs.

Examples of IaaS: Microsoft Azure, Amazon Web Services (AWS), Cisco Metacloud, Google Compute Engine (GCE)

## Platform as a Service (PaaS)

Situated a bit higher up the cloud computing pyramid is PaaS. Whereas IaaS delivers all the tools available through the cloud and leaves it to customers to build whatever suits their needs, PaaS is a bit more specialized. Rather than pure infrastructure, PaaS provides the framework needed to build, test, deploy, manage, and update software products. It utilizes the same basic infrastructure as IaaS, but it also includes the operating systems, middleware, development tools, and database management systems needed to create software applications.

PaaS is extremely helpful for any company that develops software and web-based applications. Many of the tools needed to develop for multiple platforms (computers, mobile devices, browsers, etc) can be quite expensive. By using PaaS, customers can access the development tools they need, when they need them, without having to purchase them outright. Since the platform is accessible over the internet, remote development teams can all access the same assets to speed up product development. Most PaaS tools provide extensive pre-coded applications built into the platform, which can greatly reduce coding time and help companies get their products to market faster.

Examples of PaaS: AWS Elastic Beanstalk, Apache Stratos, Google App Engine, Microsoft Azure

## Software as a Service (SaaS)

For most people, SaaS is the most familiar form of cloud computing. Situated at the top of the pyramid, SaaS is a fully-developed software solution ready for purchase and use over the internet on a subscription basis. The SaaS provider manages the infrastructure, operating systems, middleware, and data necessary to deliver the program, ensuring that the software is available whenever and wherever customers need it. Many SaaS applications run directly through web browsers, eliminating the need for downloads or installations. This greatly reduces software management issues for internal IT teams and allows companies to streamline their operations with hybrid and multi-cloud deployments.

SaaS applications allow companies to get up and running very quickly as well as scale operations rapidly. There’s no need to purchase or deploy the hardware and software used to deliver their business services. Even sophisticated enterprise-level applications, such as customer relationship management (CRM) or enterprise resource planning (ERP) programs, can be easily accessed by the smallest organizations, providing them with tools that allow them to grow their businesses more effectively than ever.

Examples of SaaS: Microsoft Office 365, Salesforce, Cisco WebEx, Google Apps

Cloud computing has transformed the way companies around the world do business in ways that many people don’t even realize. Understanding the different types of cloud computing and identifying which one is the right fit for a growing business is tremendously important. As cloud services continue to proliferate, they will surely provide new opportunities for companies looking to innovate and drive business results.

# HTTP AND MQTT REQUESTS:

# HTTP:

HTTP defines a set of **request methods** to indicate the desired action to be performed for a given resource. Although they can also be nouns, these request methods are sometimes referred to as *HTTP verbs*. Each of them implements a different semantic, but some common features are shared by a group of them: e.g. a request method can be [safe](https://developer.mozilla.org/en-US/docs/Glossary/safe), [idempotent](https://developer.mozilla.org/en-US/docs/Glossary/idempotent), or [cacheable](https://developer.mozilla.org/en-US/docs/Glossary/cacheable).

[**GET**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/GET)

The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.

[**HEAD**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/HEAD)

The HEAD method asks for a response identical to that of a GET request, but without the response body.

[**POST**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST)

The POST method is used to submit an entity to the specified resource, often causing a change in state or side effects on the server.

[**PUT**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PUT)

The PUT method replaces all current representations of the target resource with the request payload.

[**DELETE**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/DELETE)

The DELETE method deletes the specified resource.

[**CONNECT**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/CONNECT)

The CONNECT method establishes a tunnel to the server identified by the target resource.

[**OPTIONS**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/OPTIONS)

The OPTIONS method is used to describe the communication options for the target resource.

[**TRACE**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/TRACE)

The TRACE method performs a message loop-back test along the path to the target resource.

[**PATCH**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PATCH)

The PATCH method is used to apply partial modifications to a resource.

# MQTT:

MQTT is based on asynchronous messaging that follows the [**publish-subscribe paradigm**](https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe/): Senders and receivers are decoupled from one another in synchronicity, time, and space and one-to-many relationships are possible. It’s important to understand that the request-response pattern of MQTT functions and solves problems in a different way than synchronous, one-to-one based protocols like HTTP.  
An MQTT response usually doesn’t “answer” a “question” that the request presents. It is possible to implement a use case for MQTT in a way that is blocking and provides one-to-one messaging that responds with specific information based on parameters of the request. However, in most use cases, the request causes a specific action for the receiver and the response contains the result for this action.

## Response Topic

A response topic is an optional UTF-8 string that is available in any PUBLISH or CONNECT packet. In a CONNECT packet, the response topic refers to the WILL publish. If the response topic contains a value, the sender automatically identifies the corresponding PUBLISH as a request.  
The response topic field represents the topics on which the responses from the receivers of the message are expected. Both the actual topic of the initial PUBLISH (request) and the response topic can have one or more subscribers. It’s good practice for the sender of the original PUBLISH (request) to subscribe on the contained response topic before sending out the request.

## Correlation Data

Correlation data is optional binary data that follows the response topic. The sender of the request uses the data for identifying to which specific request a response that is received later relates. Response topics can be used without correlation data.  
Using the correlation makes it possible for the original sender of the request to handle asynchronous responses that can possibly be sent from multiple receivers. This data is irrelevant to the MQTT broker and only functions as a means to identify the relationship between sender and receiver.

## Response Information

In the spirit of enabling transparent implementation and better standardization, the MQTT 5 specification introduced the **Response Information** property. A client can request response information from the broker by setting a boolean field in the CONNECT.  
When this ***request response information*** is set to **true**, the broker can send an optional UTF-8 String field (***response information***) in the CONNACK packet to pass information about the response topics that are expected to be used.  
With this feature users can globally define a specific part of your topic tree on the broker, which can then be used by all clients that indicate they want to be using the request-response pattern at connection establishment.

## End-to-End Acknowledgement

MQTT ensures that the sender and receiver of messages are completely decoupled. The subscriber (receiver) that gets a message from the broker is in a separate message flow from the publisher (sender) that sends the message to the broker.