What is Cloud Application Development?

It is a software solution with components that are stored and executed on the cloud, and these apps can be accessed with an internet-connected device. It helps businesses by employing only the needed resources and also enables faster time to market.

Google docs, Microsoft 365, Gmail, and Zoom are some common examples of cloud-based applications.

Ansible, Chef, Cloudability, Informatica, Kubernetes, Lacework, and Puppet are some of the best cloud computing tools.

What Is Cloud-Based Apps?

Any cloud app consists of data and processing logic (code). The last one is quite resource-consuming and needs ample space to be run. Here we have two ways to go: storing the data on a physical server or load it all to the digital cloud. And now let us speculate a little.

Of course, we can buy a physical server and store all the data there. But looking into the perspective, will we be able to handle 10000 users' requests and scale data to different services? Well, that might be too tough a challenge. So, if you are planning on a scalable business, a cloud provider might be a better option.

How Does a Cloud-Based App Work?

While the user interacts with an app via their web browser or mobile, the data processing is handled by remote servers that have enough power to do it. In the case of cloud-based apps, these servers are cloud storages.

Thus, a user's phone or PC serves only as the "facade" or input device, while the burden of processing lies on the cloud. A digital storage significantly offloads the devices' processor and helps the app run fast.

Cloud-based development is especially relevant for apps that provide real-time interaction between users. For example, users need to share their live location and communicate in real-time in on-demand delivery and movement apps. Cloud solution is one of the best solutions to allow for such real-time processing.

Benefits Of Developing Cloud-Based Apps

You might have already thought about cloud-based storage's perks to the apps' users. All the benefits come in comparison with hardware servers. So let's look at them together!

Smart Spending

When you start using a conventional hardware server, you have to pay for onsite hardware and capital expenses. It can cost a pretty penny, especially if your business scales fast. Unlike hardware servers, cloud servers can be added as needed. Such solutions are applicable on demand, pay as you need.

Scalable Solution

Using cloud services allows you to scale up your product any time you want. All you need to do is buy additional space in the cloud and call it a day. In the case of hardware services, this is hardly possible. You need to pay for the extra server setup for each user individually, which can take loads of time and effort.

Opportunity For User Research

Cloud storage is not only a great place for data processing. It helps to gather knowledge about what features are useful for the user and what are not. With cloud storage, you have access to information about the user's experience with the app. This gives you an excellent opportunity to track the app's deficiencies and improve the product with the next version.

Support On the Provider

While using cloud storage, you do not have to worry about supporting the server's infrastructure. Bugs, errors, back sets and other server stuff – all these are handled by the cloud provider. You can focus on the maintenance of the product.

How To Develop a Cloud-Based App In 6 Steps?

Talking about the process of cloud-based app development, I would highlight 6 steps.

Step 1. Market Research

The first step you do in any process is exploring the status quo. By that I mean figuring out what's going on in the market and niche you are about to step in. Here is what can be done at this stage:

1. Explore the trends: what's going on globally?
2. Look at your competitors: how can you be different from them?
3. What about your users: who are they? What's their problem?
4. Finally, what's your value? How can you solve the problem for your users?

The answer to the final question will be your hypothesis, which you will have to validate.

Step 2. Finding Your Team of Developers

Obviously, you cannot work the whole way up all by yourself. You need a team of developers who will help you at least with the technical aspects of your project. Moreover, things get even more complicated when it comes to startup development. Not only do you need a team of polished programmers, but also experts in product development. There should be a team of people who will find your product's positioning and its technical implementation.

The team for cloud-based app development should match these requirements:

1. Focused on the product, not only programming;
2. Be familiar with cloud providers and know the difference between them;
3. Be flexible and forward-thinking to plan the app's architecture well.

Here Is What You Can Do in Your Pursuit of The Perfect Team:

1. Review ratings and reviews on Clutch, Wadline, GoodFirm, VenturePact;
2. Check freelancing websites, like Upwork, Toptal, Guru;
3. Look for company's websites, blogs, and articles, like this one;
4. Check companies' backgrounds and cases.

Step 3. Make A List of Product Requirements

Now it is time to list down the features of your cloud-based app that would form your MVP. However, you cannot get these requirements out of nowhere. The features of your app should derive from the product scope that you and your product managers will make before the project starts. Read more about how to define requirements for your product in our article Functional Vs. Non-Functional Requirements: Why Are Both Important?

Once you have the requirements, you know which specific need your product should cover. Now make a list of product features that will match this need or needs. These should be basic features that most concisely reflect the idea of your product and take the least time for development.

Step 4. Tech Research & Choosing the Provider

One of the most important steps in developing cloud-based applications is choosing the cloud provider. Several popular cloud providers on the market vary in price, services, the territory of service:

1. Amazon Web Service (AWS);
2. Google Cloud Provider (GCP);
3. Azure

Many popular cloud services provide ready-to-go solutions, which can become a part of your app. For example, this is authorization in AWS. At the same time, cloud providers try to keep up with global tech trends, like Big Data, IoT, or machine learning.

So how to choose the right provider for your product? I would suggest looking at such criteria:

1. **Security:** does the provider deliver the necessary level of data protection?
2. **Scalability:** does the provider allow you to scale up your services?
3. **Price:** is the storage affordable enough for you to use it?
4. **Data-processing speed:** doesn't it slow down your app?

Step 5. Launch MVP And Run Tests

Now it is time to get down to the development stage. Your development team will work on the technical implementation of the first version of your product. With this basic version, you will be able to test your idea with users and see if:

1. Users like your idea;
2. Users are ready to use your implementation of this idea.

Gather users' feedback, improve your product in the following iteration and launch again!

Step 6. Iterate And Maintain Your App

After the app's launch, you should check how it works, fix bugs, and make updates. Sometimes you will need to create custom updates for particular users or companies. You can also add new features, scale the app, enter new markets, etc.

Before Building a Cloud-Based App, Consider These

Cloud-based app development is a specific area of development that requires the whole team's unique approach. Here are some moments that should be considered before plunging into the development of cloud-based apps.

Data Security

The issue of security is striking when working with databases. For cloud-based applications, this issue is especially acute. For example, it is common for government-related apps not to be allowed to feature a cloud platform because of security concerns.

Luckily, there are numerous robust measures that you can take to ensure security for your cloud storage. Here are some of them:

1. **Using a private cloud**. You can create a cloud, which will only include your services, that will be available only to your users. Any exit outwards will only be accessible via gateways;
2. **Defining access rules**. You can limit the access of people to specific services. For example, people with a premium subscription have access to the service;
3. **Firewalls.** These are the "barriers" that you put between your internal network and the traffic from external sources. Usually, such barriers work according to the rules that you initially set;
4. **Alarm system**. Due notification is essential in case some unsanctioned action has already taken place. In such a case, every minute matters. The faster you get this notice, the more minor financial injury you will bear.

Architecture Arrangement

When it comes to cloud-based apps, the issue of app architecture comes to the top. In particular, the app architecture should have information on classifying the data for a cloud provider. Building an architecture means arranging communication between the services and data and organizing these services into full-fledged apps.

Cashing Data

Does all the data live in the cloud? Well, not particularly. Sometimes, the app's data can be partially stored on the device itself while "cashed" via its processor. Thanks to this data, the app can partially run offline. But once the user gets online again, the app will immediately move the data collected offline to the cloud.

Think of Scalability

Scalability is also an important thing to consider while developing a cloud-based app. You should always forecast the moment when the traffic on your product increases. If you cannot scale at this point, the app won't handle such traffic surges, and you will lose potential customers.

If you want to build an app, you will most likely develop a cloud-based one. Though challenging in terms of security, architecture, and scalability, this process still gives you more opportunities. Cloud solutions focus on business processes while leaving all the maintenance issues to the provider. Moreover, using a cloud is cheaper and faster.

Every computer or process in a network can act as a server or client in a client-server architecture. Powerful computers, known as client servers, are used just to manage network traffic, disk drives, and printers. Clients use their workstations or personal computers to run their apps. The servers' main function is to supply resources like equipment, files, and processing power. In this article, you will learn about what is client server architecture and other aspects of it.

But let's first examine the nature of the client and server before learning more about improved client-server architecture.

- **Client:** Any computer that makes a request to the server is a client. For instance, when we visit a website, we ask for the page from its domain. So, in this case, we play the client.
- **Server:** On the other hand, the server is the computer created to fulfill client requests. In the same scenario as earlier, the client queries the server for the web page, which is subsequently returned to them by the server.

What is Client Server Architecture?

A computing model known as client/server architecture places several components into clearly defined roles that allow them to communicate. The majority of the resources and services for client consumption are hosted, delivered, and managed by the server. In a shared resources architecture, a central server is connected to one or more client computers via a network or the internet.

Client/server architecture is also known as a networking computer model or client/server network since all requests and services are supplied through a network. Due to the fact that each component is working on its own, it is regarded as a type of distributed cloud computing system.

What is a client in client server architecture? In a client-server architecture, there are many clients (remote processors) who request and receive service from a centralized server (host computer). Here is a working example of client/server connections. The user or client enters the URL while using a browser to typically view a server-side website. The DNS server searches for the IP address of the web server and provides it to the browser. The server, acting as the producer, sends the files in response to an HTTP or HTTPS request made by the browser. As the consumer, the client receives them and often sends follow-up inquiries.

Components of Client Server Architecture

A few components are necessary for each architecture to function. The client-server architecture is also built on the three interrelated components that are stated below.

1. Workstations or Client

Their operating systems do not have administration or security policies, centralized databases, or shared software. They have localized variations of programs and regulations that can be used to apply them independently. Because they are not required to handle requests from numerous computers or store data from many computers, workstations also have fewer technological requirements than servers in the areas of memory, hard drive space, and CPU performance.

2. Servers

A server serves as a central repository for network files, programs, databases, and regulations, independent of the role it plays. It simplifies maintenance and backup because it is not dependent on individual user setups but can be universally and uniformly deployed across the network.

3. Networking devices

Each networking device in the client-server architecture operates and has its own set of attributes.

- Hubs are used to connect to a server with different workstations.
- Repeaters are used to transmit data from one device to another.
- Bridges are employed for segmenting isolated networks.

What's the Purpose and How Does it Work?

The purpose of a client server architecture is to speed up data transmission while also protecting the data sent. It is a smart solution for businesses seeking faster and more secure data transfer.

How does it work and interacts with servers?

In the client-server architecture, when a client computer sends a data request to the server via the internet, the server accepts the request and returns the data packets requested to the client. The architecture interacts with the servers through the following steps.

- After the user enters the URL of the website or file, the Browser sends requests to the DNS (DOMAIN NAME SYSTEM) Server.
- The DNS Server searches for the WEB Server's address.
- The DNS Server then replies with the WEB Server's IP address.
- The Browser sends an HTTP/HTTPS request to the WEB Server's IP address (provided by the DNS server).
- The Server sends the necessary files for the website.
- The Browser then renders the files and displays the website. The rendering is accomplished with the assistance of a DOM (Document Object Model) interpreter, CSS interpreter, and JS Engine, which is collectively referred to as JIT or (Just in Time) Compilers.

Client-Server Model

We now know that client-server architecture consists of two components: one that offers services and the other that uses those services.

Learn how the browser communicates with the server to gain a greater understanding of the procedure.

- After the user types the unified resource location (URL) of the website or file, the browser sends a request to the domain name system (DNS) server.
- The DNS server responds with the IP address of the web server when asked for the address of the server.
- Following receiving a response from the DNS server, the browser sends an HTTP or HTTPS request to the IP address of the web server, which was provided by the DNS server.
- The server then sends the essential files for the website.
- After rendering the files, the browser shows the webpage.

Types of Client Server Architecture

The following are the different types of client server architecture:

1-tier architecture

In a 1-tier design, every user interface environment, client/server installation setting, data logic system, and marketing logic system are present on the same system. These services are trustworthy, but handling the jobs they assign for replication of the full operation is highly challenging because they contain all the data in multiple variances.

For example, one software package can be used for presentation, business, and data access layers. The local machine stores all of the data. There are certain programs that control all three tiers, such as the MP3 player and Microsoft Office, but these programs are categorized as 1-tier apps.

2-tier architecture

The optimum client/server environment is provided by a two-tier design, which enables the user interface to be stored on the client system and all databases to be saved on the server computer. It is necessary to preserve database logic and business logic regardless of whether they are kept on the server or the user's end. The architecture is known as "fat client thin server" when these logics are stored on the client side, but we refer to it as "thin client fat server" when these logics are handled by the server.

The quickest rate is reached because client-server devices are in direct communication with one another when the client fires the order. This strategy has a few advantages, such as the best performance, simplicity in 2-tier application design, user satisfaction with the response from this architecture, and a homogeneous environment.

3-tier architecture

Middleware is required in this three-tiered design because if the client machine makes the request to the server machine, the middle layer will first receive it before passing it on to the server. Therefore, the middle layer first receives the server's response before passing it on to the client machine. Both business and data logic is kept on the middleware. Middleware is used to increase flexibility and provide top performance.

A three-tier architecture is separated into three layers: the presentation layer (Client Tier), the application layer (Business Tier), and the database layer (Data Tier). Presentation layer management is handled by the client machine, Application layer management is handled by the server machine, and Database layer management is handled by the database layer.

N-tier architecture

The approach is a scaled version with three levels. The strategy is often referred to as "Multi-tier architecture." Each function, including display, application processing, and data management functionalities, can be placed in this design as a separate layer.

Characteristics of Client-server Architecture

The following are the characteristics of client-server architecture:

- A mechanism of requests and responses powers the architecture. The client sends a request to the server, and the server returns data in response to the information requested.
- The architecture uses a common contact protocol so that devices can simply communicate with one another. Every data transport protocol is available at the application layer.
- A server may only be able to handle a limited number of client requests at once. It uses a method targeting priority to respond to each and every query.
- By assaulting the server with duplicate requests, denial of service attacks makes it difficult for it to respond to legitimate client requests.
- Scalability is a crucial feature of client-server systems. They can be resized either horizontally or vertically. Adding or removing client workstations while barely affecting performance is known as horizontal scaling. Vertical scaling refers to upgrading to a more powerful server.
- The environment is frequently multivendor and heterogeneous. Client and server operating systems and hardware platforms typically differ from one another. A well-defined set of common application program interfaces (APIs) and remote procedure calls (RPCs) is used by client and server processes to communicate with one another.

Difference Between System Architecture and Server Architecture

System architecture is a conceptual model that specifies the structure and behaviour of a system, which is the primary distinction between system architecture and software architecture. Server architecture, in contrast, is a high-level structure that specifies the responses to meet technical and business goals while maximizing the software's quality.

System architecture incorporates both software and hardware parts and is utilized to enable the designing of such a composite system. Server architecture, on the other hand, considers a variety of aspects, including human dynamics, business strategy, quality aspects, design, and IT architecture, among others.

Advantages and Disadvantages of Client-Server Architecture

Advantages

- The server contains all of the necessary data. As a result, it is simple to secure the data and offer authentication and authorization.
- It is not necessary for the server to be close to the clients. The information can still be accessed effectively.
- Customers have limited significance in this architecture and require less management on a level administration server.
- It offers service integration, which allows all of your clients to access corporate data using their terminals and removes all pointless log-in rights.

- Data recovery is simpler with a client-server model than it is with decentralized server models.

Disadvantages

- The server could get overwhelmed if every client makes a data request at once. This could cause network congestion.
- None of the client's requests can be satisfied if the server malfunctions for any reason. This causes the client-server network to fail.
- Setting up and maintaining a system is expensive.
- Need a well-qualified staff to maintain the server.

What is a Distributed Cloud?

A **distributed cloud** is an architecture where multiple clouds are used to meet compliance needs, performance requirements, or support edge computing while being centrally managed from the public cloud provider.

In essence, a distributed cloud service is a public cloud that runs in multiple locations, including

- The public cloud provider's infrastructure
- On-premises at end customer locations in the data center or at the edge
- In another cloud provider's data center
- On third party or colocation center hardware

Although there are multiple locations and geographies involved, all of the cloud services are managed as on from a single control plane that handles the differences and inconsistencies in such a hybrid, multi-cloud environment.

This distribution of services enables an organization to meet very specific requirements for response time and performance, regulatory or governance compliance mandate, or other demand requiring cloud infrastructure to be located anywhere other than the cloud provider's typical availability zones.

- Reduced latency
- Lowered or eliminated network congestion
- Guaranteed quality of service (QoS) for mission critical application and mobile users

What are the benefits of distributed cloud?

There are many benefits of a distributed cloud architecture. Gartner points to these as noteworthy:

- **Increased compliance.** Distributed by nature, workloads, and data can be located where they must be to meet regulatory demands.
- **Increased uptime.** Since cloud services can reside on local subnets, they can be isolated – even untethered from the main cloud – when needed to ensure they are isolated from a crashed system to provide redundancy.
- **Scalability:** Adding VMs or nodes as needed enables not only rapid scalability but also improves the overall availability of the cloud system as a whole.
- **Flexibility:** Distributed clouds simplify the installation, deployment, and debugging of new services.
- **Faster processing.** Distributed systems can be faster by leveraging compute of multiple systems for a given task. Also, the distributed cloud enables more responsive communications for specific regions.
- **Performance.** Unlike centralized computer network clusters, the distributed cloud can provide higher performance and better cost performance.

How does a distributed cloud work?

In a distributed cloud, services are located or 'distributed' to specific locations to reduce latency and these services enjoy a single, consistent control place across public and private cloud environments. Gartner states that organizations can see major performance gains by lowering latency and reducing the can deliver major improvements in performance due to the elimination of latency issues, reducing overall risk of outage or control plane inefficiencies.

A distributed cloud takes not just an application but the entire computing stack and distributes it to the locations where it is needed, whether public cloud provider, on-premises, or in third party colocation facility. The consuming cloud customer sees this distributed infrastructure as a single cloud entity, and the cloud provider manages all of the elements of the distributed cloud as a whole from a single control plane.

The public cloud provider continues to be responsible for all cloud operations, including security, availability, updates, and governance of the entire distributed infrastructure. To paraphrase Gartner, distributed cloud fixes what hybrid cloud and multi-cloud breaks.

What are use cases for distributed clouds?

Distributed clouds offer a broad range of applications, from smart edge computing to simplifying the management of multi-cloud environments and hybrid deployments. Common use cases include:

Distributed cloud and edge computing support everything from simplified multi-cloud management to improved scalability and development velocity to the deployment of state-of-the-art automation and decision support applications and functionality.

- **Edge/IoT.** With new uses for video inference and facial recognition being developed daily, IoT is using AI and machine learning (ML) for improving automobile manufacture, analyzing medical imaging, to smart buildings and smart cities that find the shortest route to parking and turn off the heating after the last employee has left for the evening. Many of these applications would be stymied if data had to traverse from the edge back to the cloud or a data center for analysis and processing.
- **Content optimization.** Distributed clouds can effectively become a content delivery network (CDN), which can improve the streaming experience or reduce web page load time latency, offering the best possible user experience for a broad range of applications.
- **Scaling on demand.** Distributed clouds enable expansion to existing locations without the need to build out additional infrastructure. As needs grow, the cloud footprint can grow along seamlessly, supporting the organization's changing needs.
- **Single pane of glass management.** Adopting a distributed cloud approach enhances visibility into a hybrid, multi-cloud deployment, including the ability to manage all the infrastructure as a single cloud from a single console with a single set of tools.
- **Meet compliance mandates.** Local, federal, and international data privacy regulations can mandate where a user's personal information must be stored and whether that information can travel outside that jurisdiction. In those cases where the data can't be moved to the public cloud provider, the public cloud provider can effectively be moved to the data, ensuring not only that governance and regulatory mandates are met but that data will be processed as efficiently and with the minimum amount of latency.

What is the difference between cloud and distributed cloud?

- **Traditional Cloud Computing** is the delivery of IT resources and services on demand, including servers, storage, and databases, to name a few. These services are typically provided over the public internet or private network connection from one of many hyperscale cloud providers. Cloud services can be categorized as public cloud, private cloud (including on-premises data centers), hybrid cloud (the combination of public and private), and multi-cloud (including multiple public cloud providers).
- **Distributed Cloud Computing** discards the categories of public, private, hybrid, and multi-cloud. The distributed cloud presents to the user organization as a single cloud platform, but in reality, it is comprised of multiple components that can include 'all of the above – public cloud elements from the primary provider and one or more of its competitors, private cloud or enterprise data center, and third-party colocation partner. These varied elements are all managed as one by the primary cloud provider and consumed as one by the ultimate customer.

What are the challenges of distributed cloud?

Managing an enterprise using a multi-site cloud deployment has its challenges, including:

- **Bandwidth.** A broadly dispersed multi-cloud environment may have different connectivity models for each location. Moving more computing to the edge can stress existing broadband connections and require upgrading or adapting to meet increased demand for throughput.
- **Security.** Securing a distributed cloud presents new challenges for both cloud providers and end users, as resources can be scattered across the globe and can be collocated with other enterprise servers and storage resources.
- **Data protection.** Backup and business continuity plans for dispersed data resources may require a redesign of backup and recovery strategies to ensure data stays in the geographies it is supposed to.

Cloud Application

What are Cloud Native Applications and Traditional Applications?

An application created specifically for a cloud computing architecture is called cloud-native. These applications are built to take advantage of the inherent benefits of a cloud computing software delivery model. They are hosted and run in the cloud. A native app is a software created specifically for a platform or gadget.

Traditional application development is an older method of creating apps now referred to as legacy applications. They operate in a client-server or mainframe environment and have a monolithic architecture.

Traditional vs. Cloud Native Applications

Traditional Applications are the ones that were created 10 to 20 years ago. These basic applications run on a mainframe environment or do have a client/server environment. Most people in information technology(IT) usually refer to these applications only as "really old" or ancient." They are old workhorses that have yet to be retired. Expanding the Application based on these technologies usually brings a lot of hassle. Plenty of new hardware has to be purchased to upgrade or scale up to the next level in data storage and support services.

Cloud Computing is a term that has become quite popular in recent times, regardless of the fact if you are working in IT or not. We are moving away from the world of inactive, flat, and sluggish worlds of traditional Applications to a world that is faster, more active, and versatile in its approach. The concept of Cloud Native Applications started making its impression in the last ten years or so. Still, it brought the necessary change in the process and the ideas, which is why the Traditional vs. Cloud Native Applications comparison came into existence.

Why is Everyone Inclined Toward Cloud Native Applications?

Organizations is shifting from Traditional application methodologies to the cloud-native mindset. Cloud-native is the name for an approach for building applications and services which are specific to the cloud environment. It represents the characteristics of the apps and services. The cloud approach helps Businesses to reach their full potential. The Cloud-native trend is growing so fast that it is reshaping the core idea of how companies tackle application development projects. Research indicates that organizations is adopting cloud-native application development techniques, and by the year 2020, almost 32% of new enterprise applications will be developed as cloud-native projects.

Traditional vs. Cloud Native Applications:

1. Development Time

Traditional Applications usually take a lot of time to be built, especially when compared to their Cloud-native counterpart. They are usually developed and released as one big package. Whereas, Cloud-native experts conform to a framework that is designed to maximize flexibility.

2. Operating System Dependency

The other main comparison is in terms of OS. The traditional Application architecture allows for the dependency between the operating system and the Application. This dependency is the reason which makes migration and scalability an entirely complex and challenging issue. Whereas the architecture for the Application based on cloud-native is designed to permit developers to use the platforms as a means to abstract away the dependencies. The main motive behind this is to let the teams focus on what matters while comparing Traditional vs. Cloud Native Applications.

3. Release Pace

The Cloud-native approach also ensures a faster release pace for an update means a company responds to the user need faster, eventually increasing users' retention. This is only possible because a specific bug can be located or traced to a particular microservice within a specific container. Once that bug is fixed hence will be updated quickly. After all, it is not an entirely new version of the Application. It is just some new lines of code. The amount of time spent searching for the problem's source and then working on a fix is significantly less than the traditional approach, which eventually saves time and money.

4. Transition

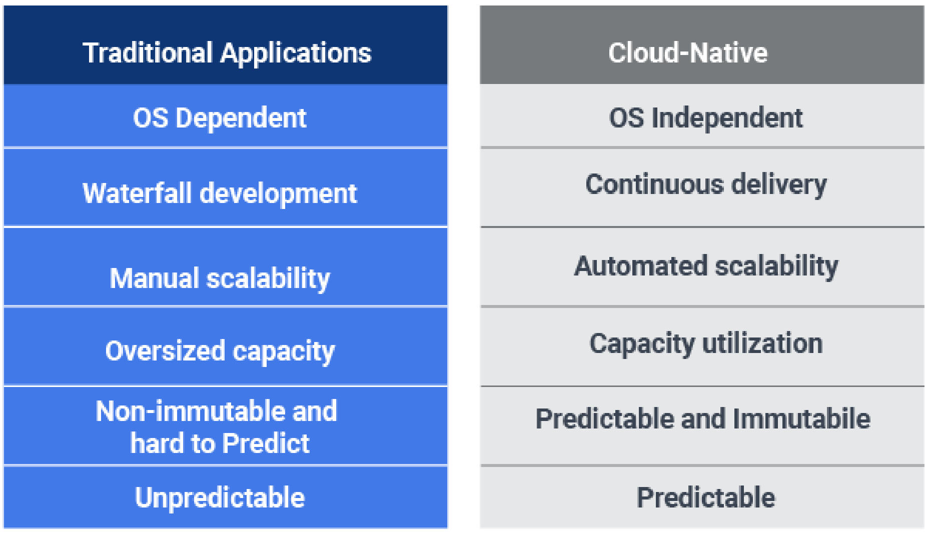
The other main difference is the way development works in both cases. In traditional applications, the developers will provide the finished application code to the operations team, and they will run that in production, which slows things down. The Priorities of the organizational structure in traditional applications take priority over customer value. In contrast, the cloud native follows a different approach as it combines people, processes, and tools. It provides a swift and smooth transition toward transferring finished application code into production.

4. Cost

Cloud-native is cost-effective, and you only pay for what you use, unlike traditional Applications where you have to set up the data storage and all the services. This is despite the amount of work being done as the downtime is low, which means the work performance is improved. The profit is increased in the longer run.

5. Security

The security of data is a significant concern. The cloud might seem less secure than storing data at local storage as anyone with access to the server can view and use the stored data and the Application on the cloud. But it is one of the primary reasons why Cloud-Native Security is famous as the ability of the data to be accessed from anywhere across the globe makes it easy to use.



Traditional Applications	Cloud-Native
OS Dependent	OS Independent
Waterfall development	Continuous delivery
Manual scalability	Automated scalability
Oversized capacity	Capacity utilization
Non-immutable and hard to Predict	Predictable and Immutabile
Unpredictable	Predictable

Risks of Traditional App Development

Traditional App Development is an excellent way to develop applications. However, it comprises some serious risks that might impact your business severely. For example:

1. Hard-Coding

When an organization works on the automation process, there is always a risk of hard-coding some human errors into its essential infrastructure. Also, incorporating human operators slows down the process of diagnosing issues. The use of computer automation nullifies these challenges.

2. Downtime

This might occur with human error and downtime. In the cloud, there is a fair amount of automation. Updates are deployed automatically without interfering with the deployed Application or the user base. Scalability, testing, as well as resource allocation can be automated. Cloud provides Greater Protection from failures. As all the microservices are isolated in containers means if something fails, it is only limited in its scope; the whole Application does not go down, which makes it more reliable.

3. Low Backup Capabilities

Traditional applications often offer low backup capabilities; a small mistake can lead to failure. In comparison, backups on the cloud are automated. The containers provide dynamic and high-quality virtualization on top of the VM-matching microservices. Clusters are placed across the VM to get their benefits in elastic scaling and recovery. The traditional Application can constantly be refactored by changing the code, not changing the site's infrastructure, and moving to the cloud infrastructure.

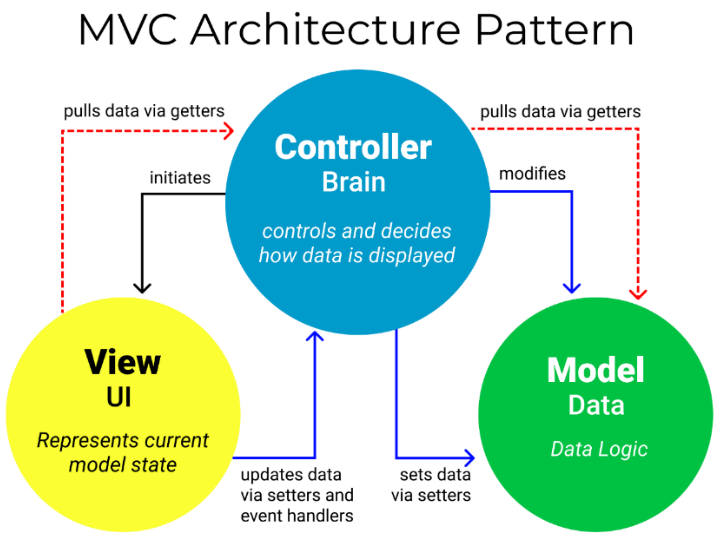
MVC

The Model View Controller Pattern – MVC Architecture and Frameworks

What is MVC?

MVC stands for model-view-controller. Here's what each of those components mean:

- **Model**: The backend that contains all the data logic
- **View**: The frontend or graphical user interface (GUI)
- **Controller**: The brains of the application that controls how data is displayed



Today the MVC pattern is used for modern web applications because it allows the application to be scalable, maintainable, and easy to expand.

Why Should You Use MVC?

The MVC pattern helps you break up the frontend and backend code into separate components. This way, it's much easier to manage and make changes to either side without them interfering with each other.

How to Use MVC

To better illustrate the MVC pattern, A web application that shows how these concepts all work.

My Car Clicker application
Here are some of the major differences in my app:

1. Multiple car models are listed
2. There are multiple click counters
3. It only displays the selected car

Now let's dive into these three components that make up the MVC architecture pattern.

Model (data)

The model's job is to simply manage the data. Whether the data is from a database, API, or a JSON object, the model is responsible for managing it.

In the Car Clicker application, the model object contains an array of car objects with all the information (data) needed for the app.

Views (UI)

The view's job is to decide what the user will see on their screen, and how.
The Car Clicker app has two views: carListView and CarView.
Both views have two critical functions that define what each view wants to initialize and render.

Controller (Brain)

The controller's responsibility is to pull, modify, and provide data to the user. Essentially, the controller is the link between the view and model.

Through getter and setter functions, the controller pulls data from the model and initializes the views.

If there are any updates from the views, it modifies the data with a setter function.

MVC Frameworks

JavaScript has grown in popularity, and it's taken over the backend in recent years. More and more full-blown JavaScript applications have opted for the MVC architecture pattern in one way or another.

Frameworks come and go, but what has been constant are the concepts borrowed from the MVC architecture pattern.

Some of the early frameworks that applied these concepts were **KnockoutJS**, **Django**, and **Ruby on Rails.**

Modern web applications are very complex, and making a change can sometimes be a big headache.

Managing the frontend and backend in smaller, separate components allows for the application to be scalable, maintainable, and easy to expand.