Shalitha Suranga Follow

Jul 20, 2021 · 6 min read ·

10 Must-Know Concepts of Modern Web Architecture

Widely used generic cloud computing components in modern web-based software systems

Photo by Science in HD on Unsplash.

After Tim Berners-Lee invented the WWW (World Wide Web) in 1989, almost all types of physical services started moving to the cloud ecosystem. Before the rapid evolution of the internet, there were remotely connected desktop applications. Users had to

After that, websites and web applications started growing rapidly. Earlier, we had static web pages with Web 1.0. But now, we have highly dynamic, user-friendly, and interactive web applications thanks to Web 2.0 and Web 3.0. Web application architecture also evolved from the simple monolithic client-server model to highly scalable microservices in order to handle massive traffic coming from internet users.

Every modern web application architecture has common cloud computing theoretical components that every developer should know. Cloud service providers may promote these components by using various names, but the underlying theoretical cloud computing component remains the same.

I researched some modern tech giants' different web application architectures (Netflix, Medium, Airbnb, and Facebook) and found the following building blocks.

1. Server

A server refers to a computer that provides a service (or multiple services) over a private network or internet. Other devices known as clients can connect the server to obtain the provided service via different network ports. Servers are typically named based on the service they provide. For example, if a server accepts HTTP requests from port 80 and serves web pages to clients, the particular server is known as a web server. Likewise, we have file servers, mail servers, authentication servers, database servers, email servers, application servers, etc.

Nowadays, virtual servers are more popular than bare-metal servers. Cloud service providers create virtual machines on top of their physical hardware by using hypervisors and virtualization software. For example, several popular cloud providers use the Xen type-1 hypervisor to create secure virtual machines.

2. Client

The client is the device that connects with servers to consume its services or resources.

naming, clients are also named based on what service they consume. We have email clients, web clients, database clients, and online chat clients.

In the client-server model, techniques like authentication help to expose a specific service to only specific clients. There are two key client types: thin clients and thick clients. A thin client depends entirely on a server like a normal single-page application (SPA). On the other hand, a thick client doesn't depend on a server. It's similar to a standalone app that persists data on a server.

3. Microservice

A monolithic client-server architecture-based system brings several drawbacks. Monolithic systems' failures affect the entire system, and maintenance work can be problematic. The microservice pattern motivates developers to decompose large monolithic systems into small services known as microservices. A microservice refers to a loosely coupled and isolated service that is responsible for a particular process.

If you implement a user management system, your microservices may work on user registration, report generation, and payment processes. In real-world web-based software systems, most microservices are RESTful APIs running inside a virtual machine or container.

4. Cloud Function

Microservices can be a bit heavy and a bit hard to maintain when the complexity of the code grows. Also, infrastructure costs could soar because microservices typically wait all the time for client connectivity on virtual machines or containers. The serverless concept introduced a way to decompose large systems into smaller maintainable functions known as serverless functions (aka cloud functions).

A cloud function gets activated when there are incoming requests. Otherwise, the particular cloud function will enter hibernation mode. A cloud function's lifecycle is

5. Load Balancer

Load balancing refers to distributing the incoming workload to different computing units. In web architecture, a load balancer is a component that directs web traffic into different servers based on availability. There are two main types of load balancers: hardware-based load balancers (HLB) and software-based load balancers. Software-based load balancers are popular nowadays because HLBs are expensive and need physical servers.

Now almost all cloud service providers integrate load balancers into their services stack with the well-known *as-a-service* model.

6. API Gateway

A particular web application can have multiple APIs, and every API needs to be protected from overuse via rate limiting. An API gateway provides a single point of access to multiple APIs and other services. The API gateway's behavior is similar to that of the load balancer, but it directs every client request to mapped services. API gateways typically come as a part of an API management solution. An API management solution offers a GUI to manage APIs inside a protected admin dashboard.

API gateways offer various integrated services like API monitoring, rate limiting, API monetization, and API versioning. API gateways are often used with RESTful APIs, but API gateways may support SOAP, GraphQL, gRPC, and TCP protocols.

7. Message Queue

In modern web architecture, we have decomposed and manageable components known as microservices. Microservices use either RESTful APIs or message queues for communicating with each other. Message queues build an asynchronous messaging channel between two microservices by following the pub-sub architecture. Message handling the error. Also, the malfunctioned REST message is discarded and stored nowhere. On the other hand, message queues persist every message.

Therefore, if the consumer fails when the producer sends the message, the consumer will pick up the particular message when it gets restarted again. These advantages motivate web architects to select message queues for scenarios where transaction reliability is crucial.

8. CDN

A Content Delivery Network (CDN) refers to geographically distributed servers that cache static content to improve web applications' performance, availability, and security. A web application typically consists of resources like images, videos, stylesheets, and JavaScript files. Whenever a user visits your web application, the particular user's browser will load those resources from your server. If a user who lives far away from the physical server's location visits your web application, the page loading time will increase. CDNs cache static content in many servers around the world and serve content fast to geographically closer clients.

Moreover, CDNs can mitigate Denial-of-service (DDoS) attacks because CDN caching servers act as a proxy to your original server.

9. Database

A database is a component that stores various types of data records. There are several database types: SQL, Cloud, Key-Value, Graph, and Document databases. Database servers allow clients to interact with the database via specific communication protocols. For example, the MySQL database server uses the MySQL protocol. Web architects need to decide on a database type based on real-world requirements.

For example, if there is a need to store many user sessions based on a unique ID, a key-value-type database is a good fit.

10. Services

Web applications require different services such as authentication, emailing, logging, monitoring, machine learning, payment, etc. Also, web applications need development, architectural, and deployment-related services such as repository hosting, continuous integration/delivery (CI/CD), database, application hosting, CDN, searching/indexing, etc. Many open source frameworks fulfill the core requirements of these services. However, we need infrastructure to install and consume those open source services.

Nowadays, the many as-a-service model-based companies and startups offer almost all these cloud services for web development. Also, several big tech companies created their own development ecosystems by aggregating many cloud services.

Thanks to Athif Shaffy and Anupam Chugh