FRONTEND

Authentication/JWT

### What is JWT Authentication?

### JSON Web Token (JWT) is a JSON encoded representation of a claim(s) that can be transferred between two parties. The claim is digitally signed by the issuer of the token, and the party receiving this token can later use this digital signature to prove the ownership on the claim.

JWTs can be broken down into three parts: header, payload, and signature. Each part is separated from the other by dot (.), and will follow the below structure:

**Header.Payload.Signature**

### HEADER

### The information contained in the header describes the algorithm used to generate the signature. The decoded version of the header from the above example looks like:

{

“alg”: “HS256”,

“typ”: “JWT”

}

HS256 is the hashing algorithm HMAC SHA-256 used to generate the signature in the above example.

### PAYLOAD

All the claims within JWT authentication are stored in this part. Claims are used to provide authentication to the party receiving the token. For example, a server can set a claim saying ‘isAdmin: true’ and issue it to an administrative user upon successfully logging into the application. The admin user can now send this token in every consequent request he/she sends to the server to prove their identity.

The decoded version of the payload from the JWT example provided above looks like:

{

“sub”: “1234567890”,

“name”: “John Doe”,

“iat”: 1516239022

}

The ‘name’ field is used to identify the user to whom the token was issued to. The ‘sub’ and ‘iat’ are examples of registered claims and are short for ‘subject’ and ‘issued at’.

SIGNATURE

The signature part of a JWT is derived from the header and payload fields. The steps involved in creating this signature are described below:

1. Combine the base64url encoded representations of header and payload with a dot (.)

base64UrlEncode(header) + “.” + base64UrlEncode(payload)

2. Hash the above data with a secret-key only known to the server issuing the token. The hashing algorithm is the one described inside the header.

hash\_value = hash([base64UrlEncode(header) + “.” + base64UrlEncode(payload)], secret-key)

3. Base64Url encode the hash value obtained from the step above

Signature = base64UrlEncode(hash\_value)

Is JWT safe for authentication?

JWTs can be used as an authentication mechanism that does not require a database. The server can avoid using a database because **the data store in the JWT sent to the client is safe**.

**React.js**

## What is React?

**React is a JavaScript library created by Facebook.**

**React is a tool for building UI components.**

**What is react JS used for?**

**React. js is an open-source JavaScript library that is used for building user interfaces specifically for single-page applications. It's used for handling the view layer for web and mobile apps. React also allows us to create reusable UI components**

## How does React Work?

## **React creates a VIRTUAL DOM in memory.**

## **Instead of manipulating the browser's DOM directly, React creates a virtual DOM in memory, where it does all the necessary manipulating, before making the changes in the browser DOM.**

**Is React JS frontend or backend?**

React. js/React is an **open-source frontend framework** that is based on JavaScript, developed by Facebook, and best known for its virtual DOM feature. With React, we recommend Express. js/Express as a backend service.

**Who uses ReactJS?**

Who uses React? 9943 companies reportedly use React in their tech stacks, including **Uber, Airbnb, and Facebook**.

**Do backend developers use React?**

Far not every web app built with React needs a backend. And those that do, can have different programming languages server-side. What backend to use with React? **Python, PHP, Ruby, or Java** – you can add any of them to your project tech stack.

LOCAL STORAGE

**What is localStorage?**

Local storage **allows developers to store and retrieve data in the browser**. The data stored in local storage will not expire. This means the data will persist even if the tab or the browser window is closed.

**When should you use localStorage?**

If you're building a static site (like a single page app, for instance), using something like local storage means **your web pages can run independently of any web server**. They don't need any backend language or logic to store data in the browser: they can just do it as they please.

**How do I set up localStorage?**

**To use localStorage in your web applications, there are five methods to choose from:**

1. setItem() : Add key and value to localStorage.
2. getItem() : This is how you get items from localStorage.
3. removeItem() : Remove an item by key from localStorage.
4. clear() : Clear all localStorage.

**How much data can we store in localStorage?**

Data stored using local storage isn't sent back to the server (unlike cookies, ). All the data stays on the client-side, thus there is a defined limitation regarding the length of the values, and we can currently store from **2 MB to 10 MB size** of data depending upon the browser we use. Syntax: localStorage.

**Does LocalStorage have limits?**

LocalStorage should be avoided because it is synchronous and will block the main thread. It is **limited to about 5MB** and can contain only strings. LocalStorage is not accessible from web workers or service workers.

**DNS SERVER**

## **What is DNS?**

**The Domain Name System (DNS) is the phonebook of the Internet. Humans access information online through domain names, like nytimes.com or espn.com. Web browsers interact through Internet Protocol (IP) addresses. DNS translates domain names to** [**IP addresses**](https://www.cloudflare.com/learning/dns/glossary/what-is-my-ip-address/) **so browsers can load Internet resources.**

**Each device connected to the Internet has a unique IP address which other machines use to find the device. DNS servers eliminate the need for humans to memorize IP addresses such as 192.168.1.1 (in IPv4), or more complex newer alphanumeric IP addresses such as 2400:cb00:2048:1::c629:d7a2 (in IPv6).**

## **How does DNS work?**

**The process of DNS resolution involves converting a hostname (such as www.example.com) into a computer-friendly IP address (such as 192.168.1.1). An IP address is given to each device on the Internet, and that address is necessary to find the appropriate Internet device - like a street address is used to find a particular home. When a user wants to load a webpage, a translation must occur between what a user types into their web browser (example.com) and the machine-friendly address necessary to locate the example.com webpage.**

**How do I fix a DNS server problem?**

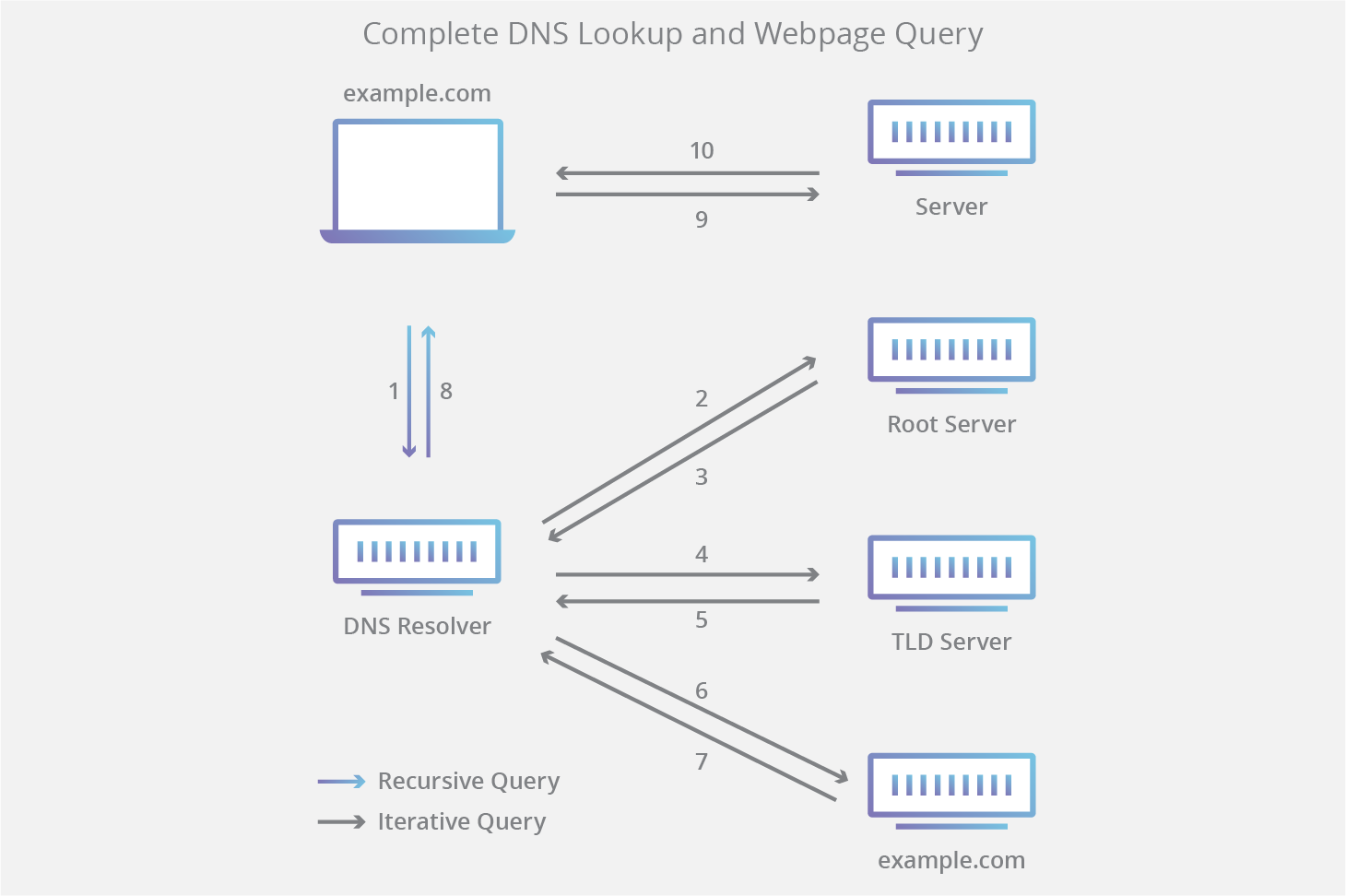
**Let's take a look at ten potential ways you can fix “DNS Server Not Responding” on Windows and Mac devices.**

1. **Switch to a Different Browser. ...**
2. **Start Your Computer in Safe Mode. ...**
3. **Temporarily Disable Your Antivirus Software and Firewall. ...**
4. **Disable Secondary Connections. ...**
5. **Disable the Windows Peer-to-Peer Feature. ...**
6. **Restart Your Router**

## **How do DNS servers resolve a DNS query?**

**typical DNS query without any** [**caching**](https://www.cloudflare.com/learning/cdn/what-is-caching/)**, there are four servers that work together to deliver an IP address to the client: recursive resolvers, root nameservers, TLD nameservers, and authoritative nameservers.**

**The DNS recursor (also referred to as the DNS resolver) is a server that receives the query from the DNS client, and then interacts with other DNS servers to hunt down the correct IP. Once the resolver receives the request from the client, the resolver then actually behaves as a client itself, querying the other three types of DNS servers in search of the right IP.**

****

**First the resolver queries the root nameserver. The root server is the first step in translating (resolving) human-readable domain names into IP addresses. The root server then responds to the resolver with the address of a Top Level Domain (TLD) DNS server (such as .com or .net) that stores the information for its domains.**

**Next the resolver queries the TLD server. The TLD server responds with the IP address of the domain’s authoritative nameserver. The recursor then queries the authoritative nameserver, which will respond with the IP address of the origin server.**

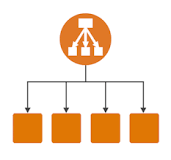
**The resolver will finally pass the origin server IP address back to the client. Using this IP address, the client can then initiate a query directly to the origin server, and the origin server will respond by sending website data that can be interpreted and displayed by the web browser.**

**Load Balancer**

**What is load balancer and how it works?**

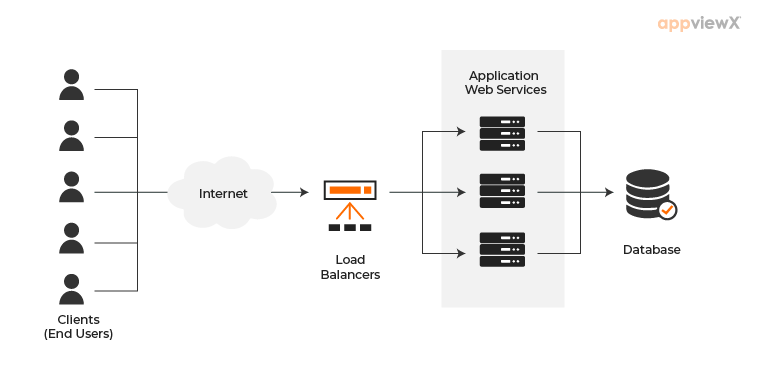
**Load balancing is defined as the methodical and efficient distribution of network or application traffic across multiple servers in a server farm. Each load balancer sits between client devices and backend servers, receiving and then distributing incoming requests to any available server capable of fulfilling them.**

**What is load balancer and its types?**

****

**Elastic Load Balancing supports the following types of load balancers: Application Load Balancers, Network Load Balancers, and Classic Load Balancers. Amazon ECS services can use either type of load balancer. ... Network Load Balancers and Classic Load Balancers are used to route TCP (or Layer 4) traffic.**

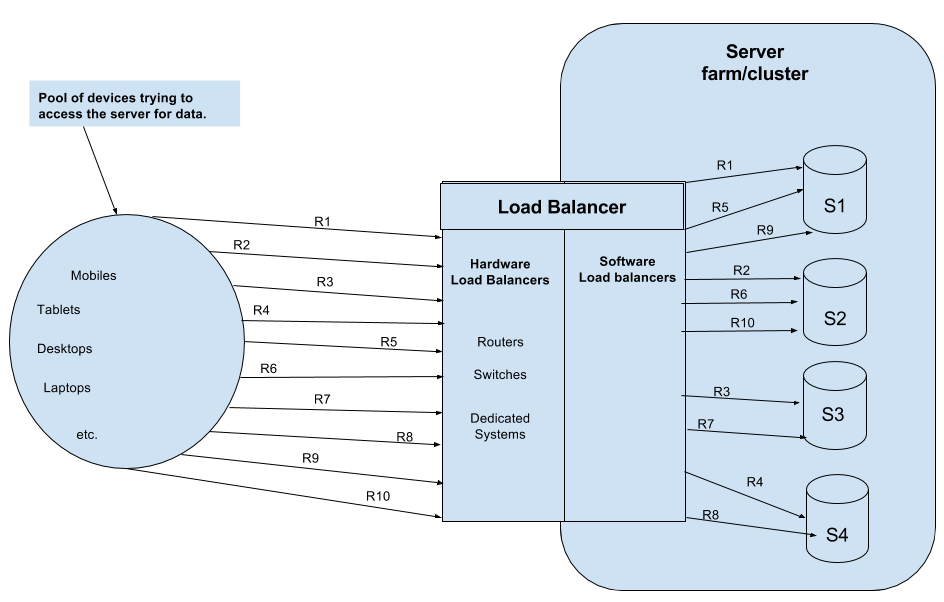
**What are the types of load balancer?**

****

**2.2 Types of Load Balancers – Based on Functions**

* **a.) Network Load Balancer / Layer 4 (L4) Load Balancer: ...**
* **b.) Application Load Balancer / Layer 7 (L7) Load Balancer: ...**
* **c.) Global Server Load Balancer/Multi-site Load Balancer: ...**
* **a.) Hardware Load Balancers: ...**

**What is a load balancer example?**

****

**The following are few examples of software load balancers: HAProxy – A TCP load balancer. NGINX – A http load balancer with SSL termination support. ... LVS – Linux virtual server offering layer 4 load balancing.**