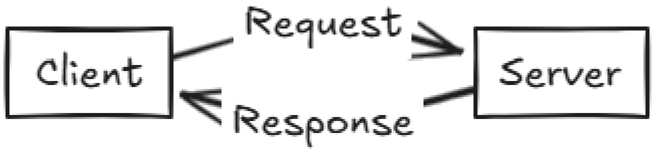
FUNDAMENTALS OF SYSTEM DESIGN

Tool:
https://cbt.apporto.com/
https://github.com/MohammedKhan3/SYSRTEM_DESIGN-WORKSHOP/tree/main

CONTENT:

CLIENT-SERVER MODEL:

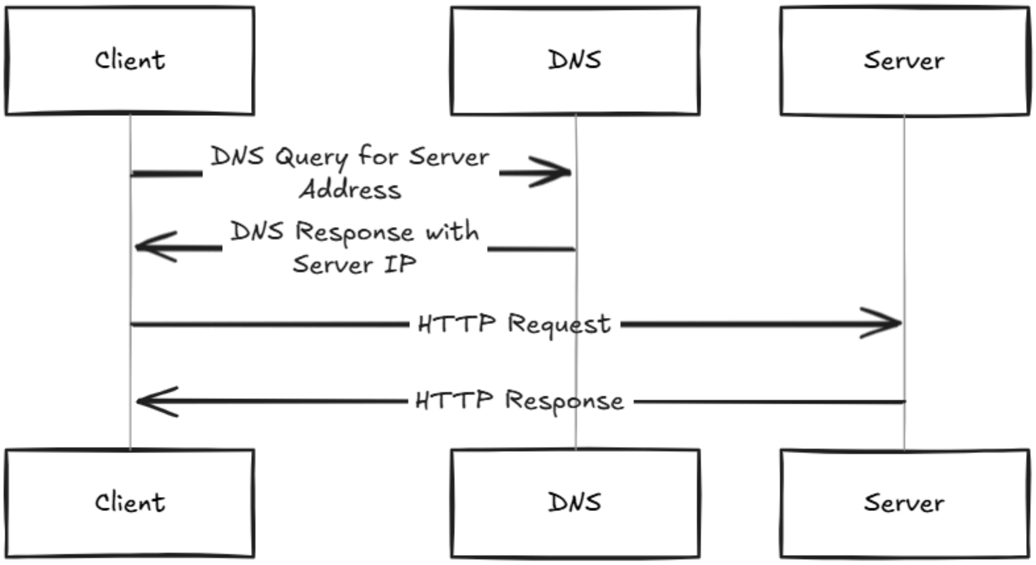


Client : A Machine or Process that requests data or services from a server.
Server : A Machine or Process that provides data or service for a client, usually by listening for incoming network calls.

Domain Name System- It describes the entities and protocols involved in the translation from domain names to IP addresses. Typically, machines make DNS queries to a well known entity which is responsible for the IP address of the requested domain name in the response.

Example:

DNS (Domain Name System) is like the phonebook of the internet. It translates human-readable domain names (like google.com) into IP addresses (like 142.250.182.14) that computers use to identify each other on the network.



Let's open git bash and type nslookup google.com

Output should look like this :

```
$ nslookup google.com
```

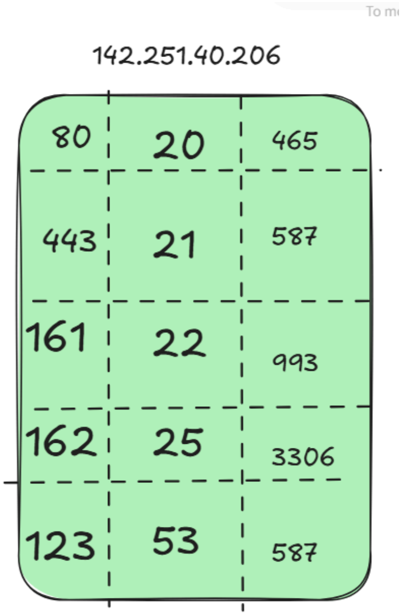
```
Server:  UnKnown
Address:  172.16.0.4

Non-authoritative answer:
Name:  google.com
Addresses:  2607:f8b0:4004:c08::8ab
          2607:f8b0:4004:c08::71
          2607:f8b0:4004:c08::66  —------> IPV6
          2607:f8b0:4004:c08::8b
          142.251.40.206—---> IPV4
```

Research on IPV6 VS IP4?

Ports: In order for multiple programs to listen for new network connections on the same machine without colliding, they have to pick a port to listen on. A port is an integer between 0 and 65,535.

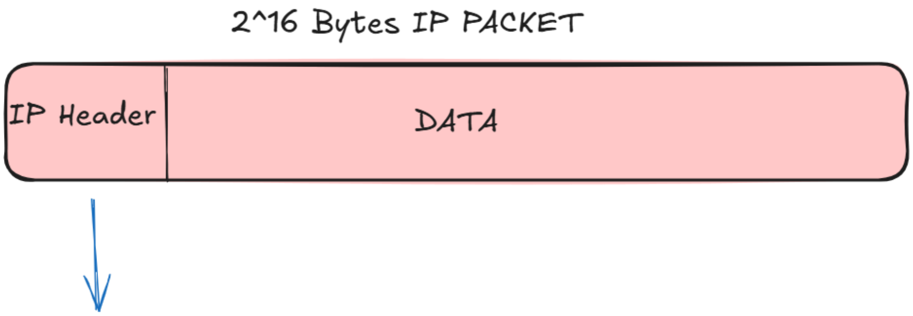
- Ports from 0-1023 are reserved for system ports also called well-known ports and shouldn't be used by user-level processes.
- 22: Secure Shell
- 53: DNS lookup
- 80: HTTP
- 443:HTTPS



80	20	465
443	21	587
161	22	993
162	25	3306
123	53	587

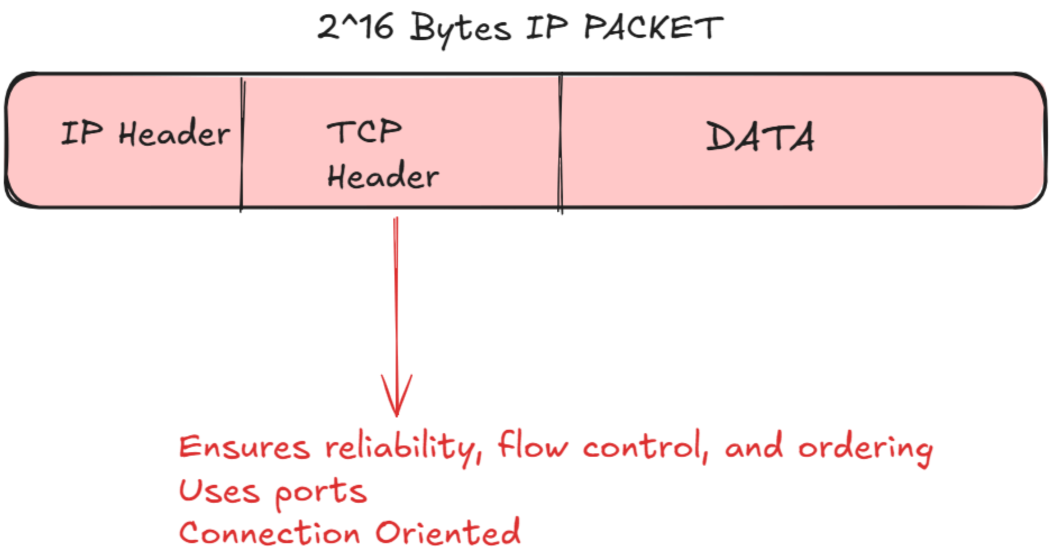
ZOOMING IN INTO REQUEST AND RESPONSE:

IP: Internet Protocol. This network protocol outlines how almost all machine to machine communications should happen in the world.



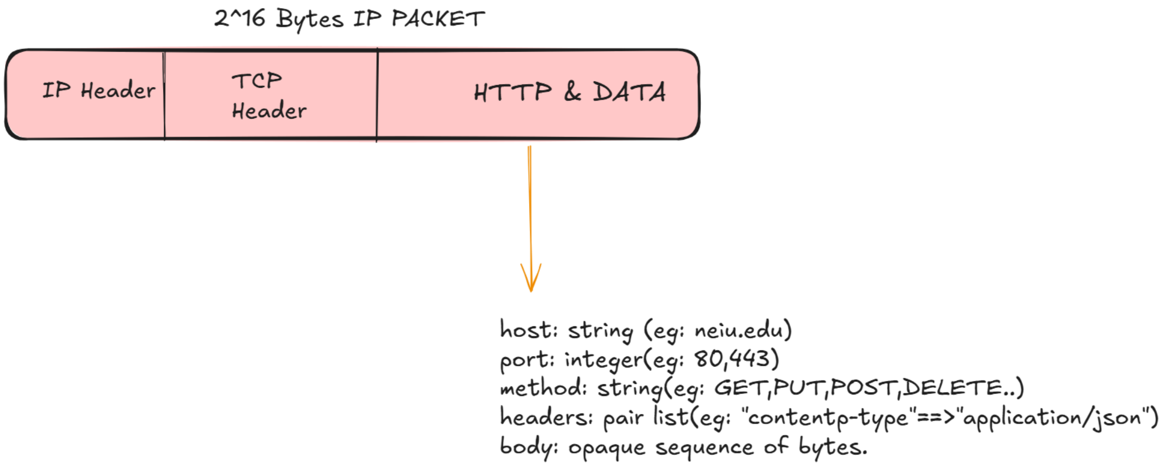
Contains useful information:
Directs packets to their destination using source and destination IPs.
IP Versions.
Total size of packet.
Header is 0-16 bytes.

TCP: Transmission Control Protocol is built on top of IP. Allows for ordered,reliable data delivery between machines over the public internet by creating connections.



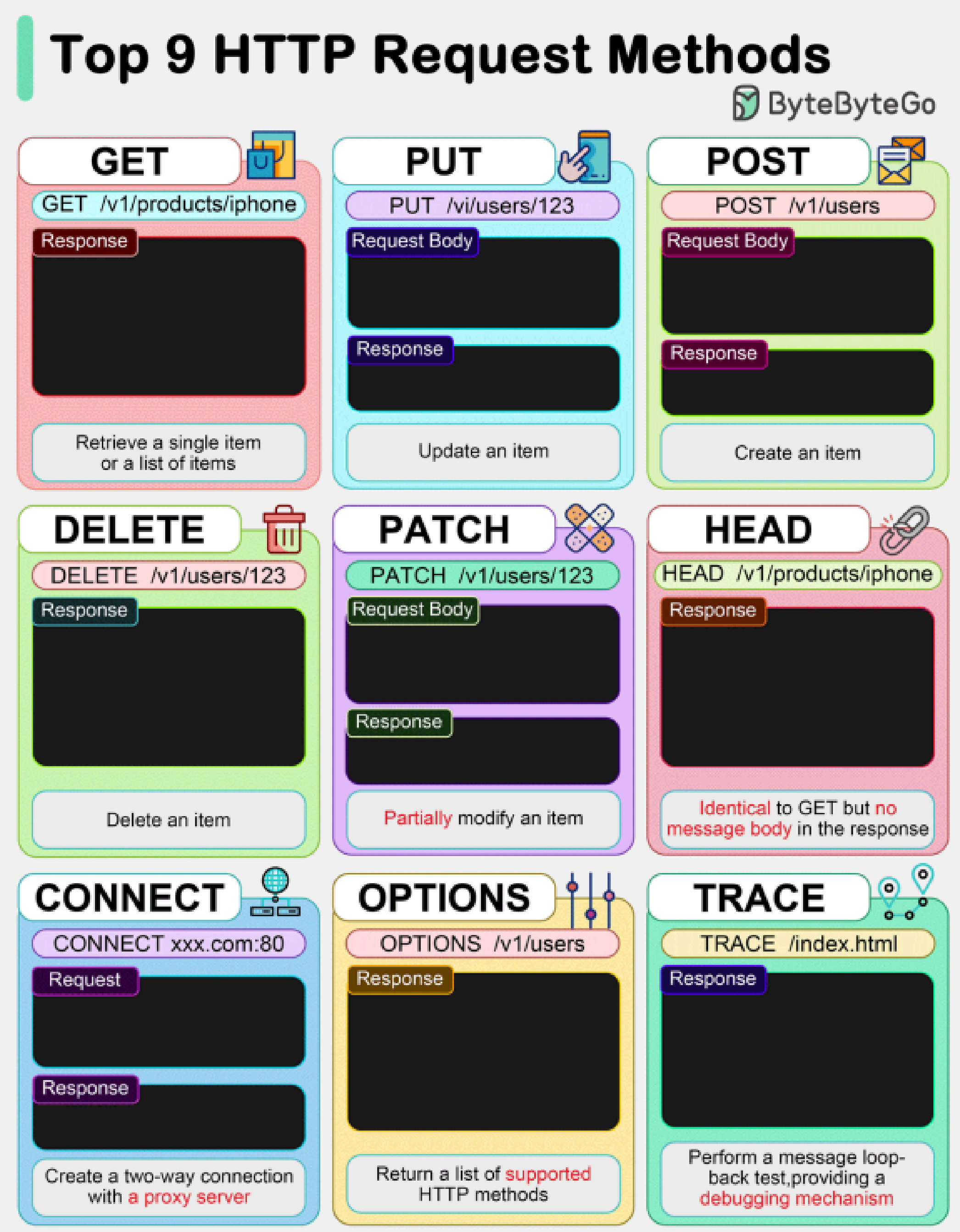
Ensures reliability, flow control, and ordering
Uses ports
Connection Oriented

HTTP: HyperText Transfer protocol is a very common network protocol implemented on top of TCP. Clients make HTTP requests and the server responds with a response.



host: string (eg: neiu.edu)
port: integer(eg: 80,443)
method: string(eg: GET,PUT,POST,DELETE..)
headers: pair list(eg: "contentp-type"==>"application/json")
body: opaque sequence of bytes.

When you use **HTTP** or **HTTPS**, you're interacting with web services that follow certain **API methods** (also called **HTTP methods** or **verbs**). These methods define the actions you want to perform on the server. The most commonly used HTTP methods are:



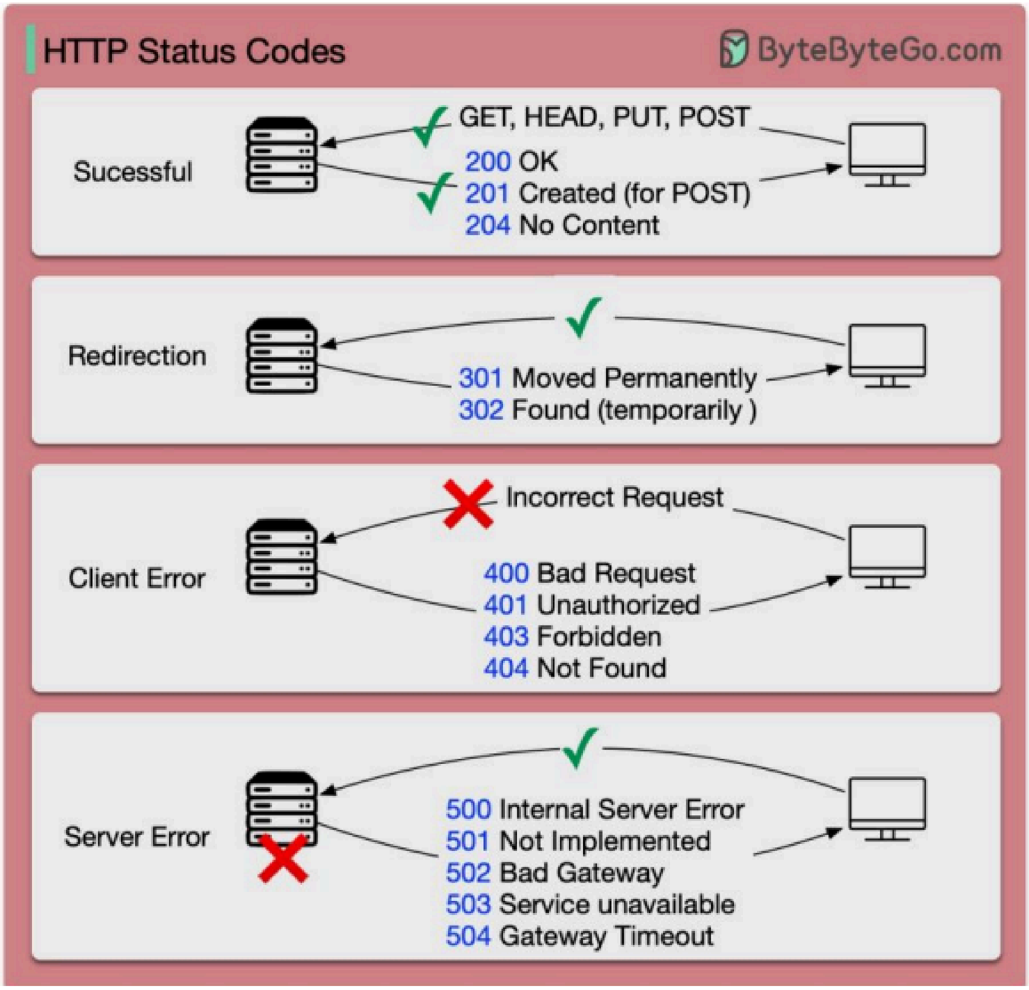
Let’s head To :

https://hoppscotch.io/
(FREE API TESTING TOOL)
&&
https://apipheny.io/free-api/

My Favorite:

https://official-joke-api.appspot.com/random_joke

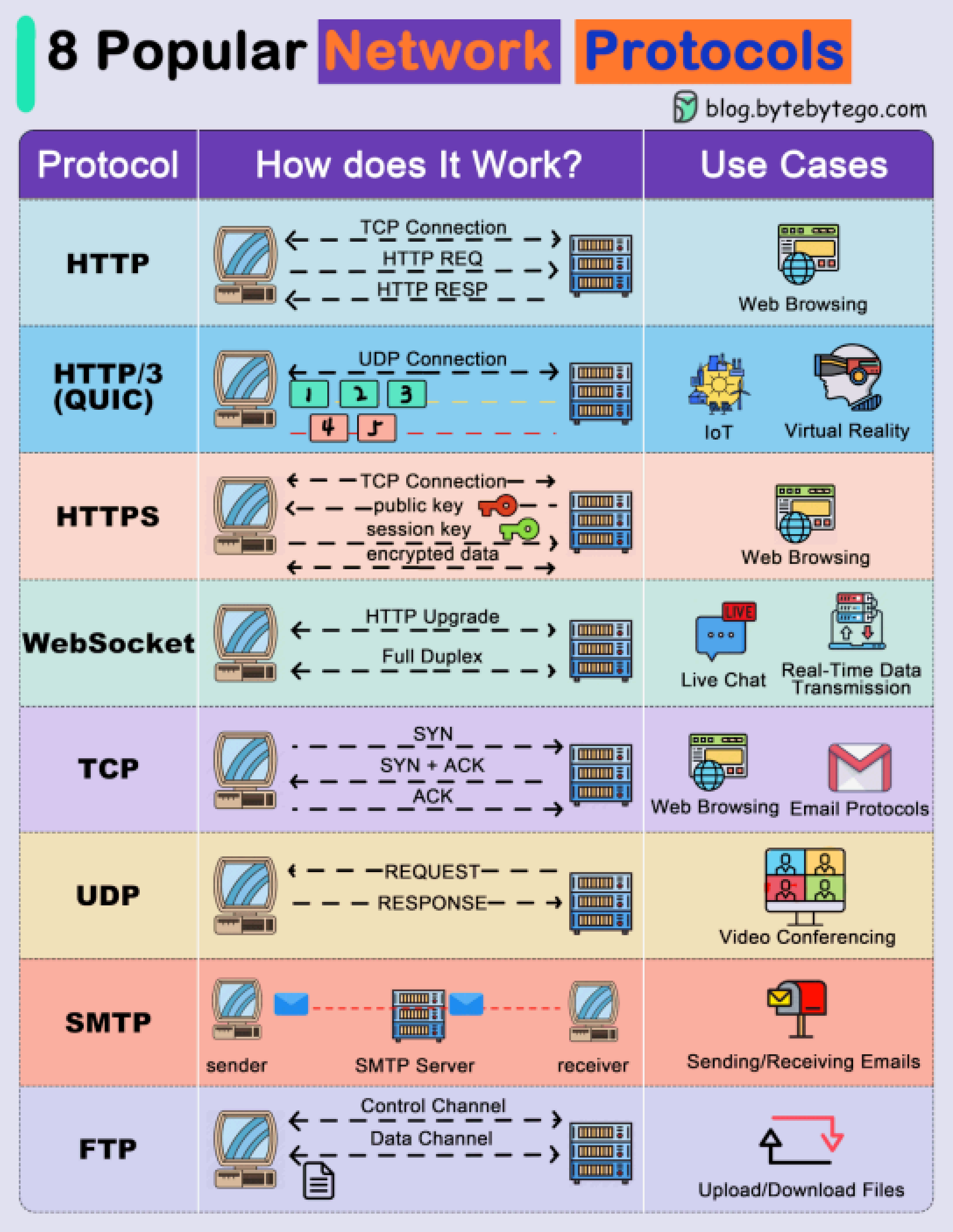
Some Common API Status Codes:



More Protocols:

HTTP (HyperText Transfer Protocol):

HTTP is a protocol for fetching resources such as HTML documents. It is the foundation of any data exchange on the Web and it is a client-server protocol.



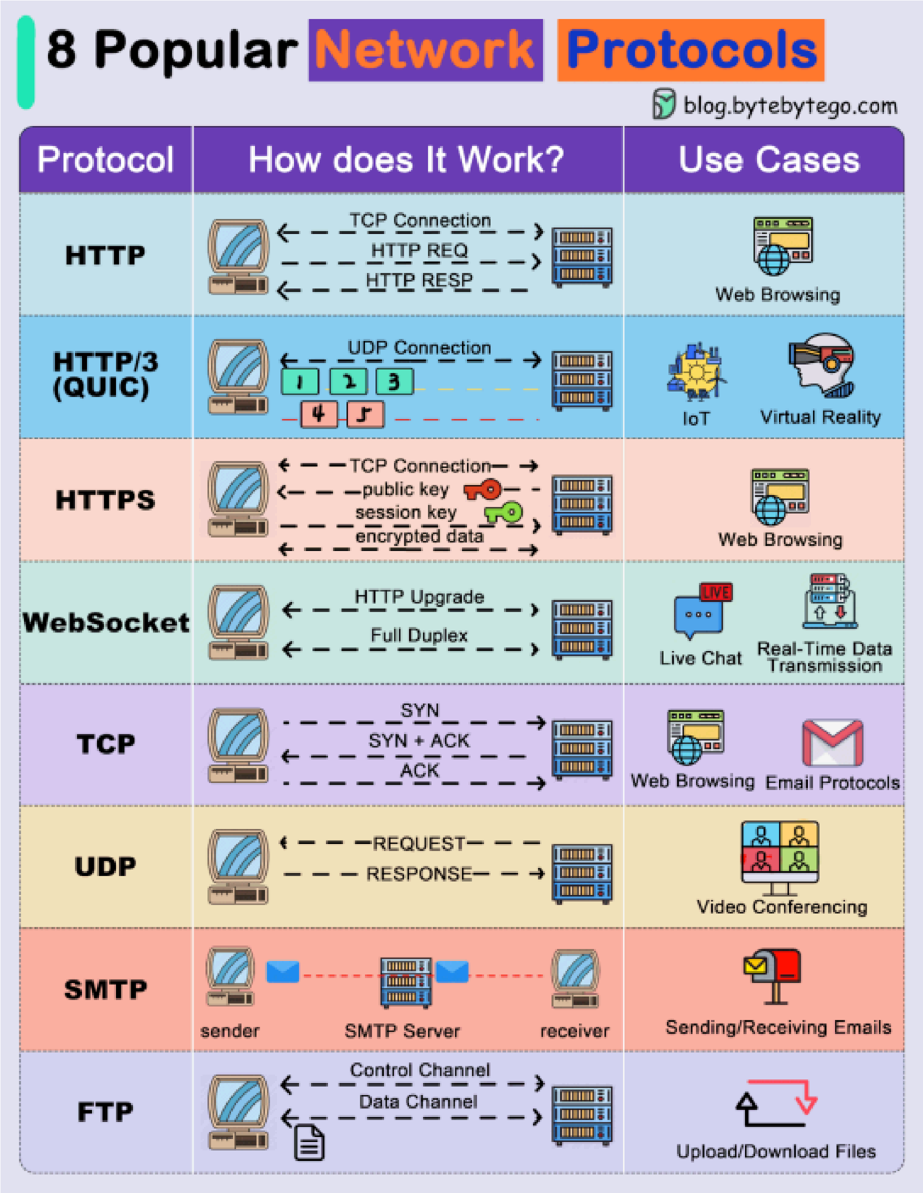
8 Popular Network Protocols

blog.bytebytego.com

Protocol	How does It Work?	Use Cases
HTTP	TCP Connection HTTP REQ HTTP RESP	Web Browsing
HTTP/3 (QUIC)	UDP Connection 1 2 3 4 5	IoT Virtual Reality
HTTPS	TCP Connection public key session key encrypted data	Web Browsing
WebSocket	HTTP Upgrade Full Duplex	Live Chat Real-Time Data Transmission
TCP	SYN SYN + ACK ACK	Web Browsing Email Protocols
UDP	REQUEST RESPONSE	Video Conferencing
SMTP	sender SMTP Server receiver	Sending/Receiving Emails
FTP	Control Channel Data Channel	Upload/Download Files

HTTP/3 :

HTTP/3 is the next major revision of the HTTP. It runs on QUIC, a new transport protocol designed for mobile-heavy internet usage. It relies on UDP instead of TCP, which enables faster web page responsiveness. VR applications demand more bandwidth to render intricate details of a virtual scene and will likely benefit from migrating to HTTP/3 powered by QUIC.



8 Popular Network Protocols

blog.bytebytego.com

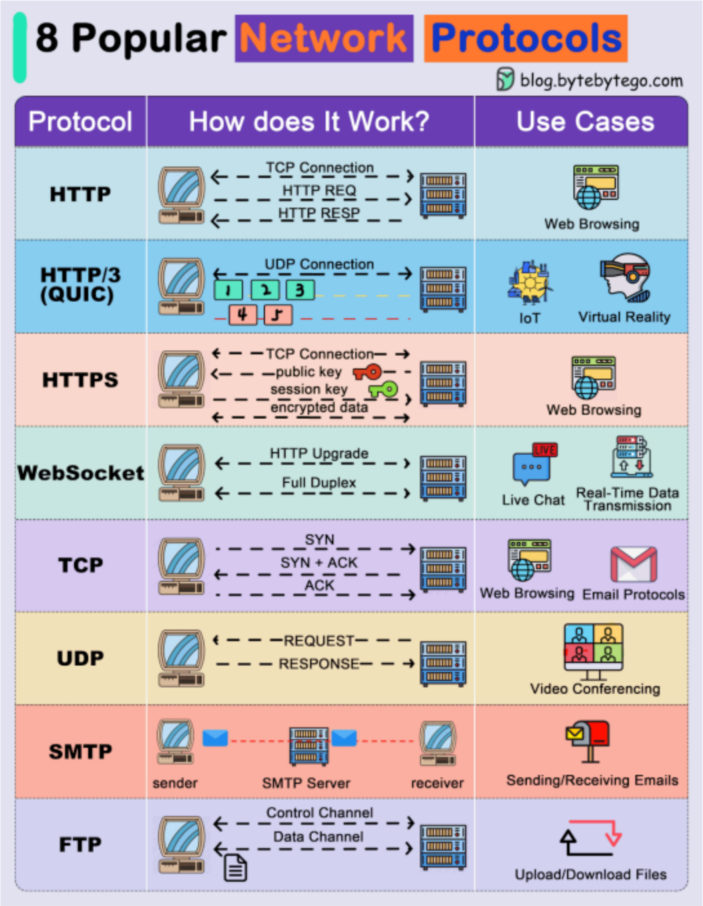
Protocol	How does It Work?	Use Cases
HTTP	TCP Connection HTTP REQ HTTP RESP	Web Browsing
HTTP/3 (QUIC)	UDP Connection 1 2 3 4 5	IoT Virtual Reality
HTTPS	TCP Connection public key session key encrypted data	Web Browsing
WebSocket	HTTP Upgrade Full Duplex	LIVE Live Chat Real-Time Data Transmission
TCP	SYN SYN + ACK ACK	Web Browsing Email Protocols
UDP	REQUEST RESPONSE	Video Conferencing
SMTP	sender SMTP Server receiver	Sending/Receiving Emails
FTP	Control Channel Data Channel	Upload/Download Files

HTTPS (HyperText Transfer Protocol Secure):

HTTPS extends HTTP and uses encryption for secure communications.

WebSocket:

WebSocket is a protocol that provides full-duplex communications over TCP. Clients establish WebSockets to receive real-time updates from the back-end services. Unlike REST, which always “pulls” data, WebSocket enables data to be “pushed”. Applications, like online gaming, stock trading, and messaging apps leverage WebSocket for real-time communication.



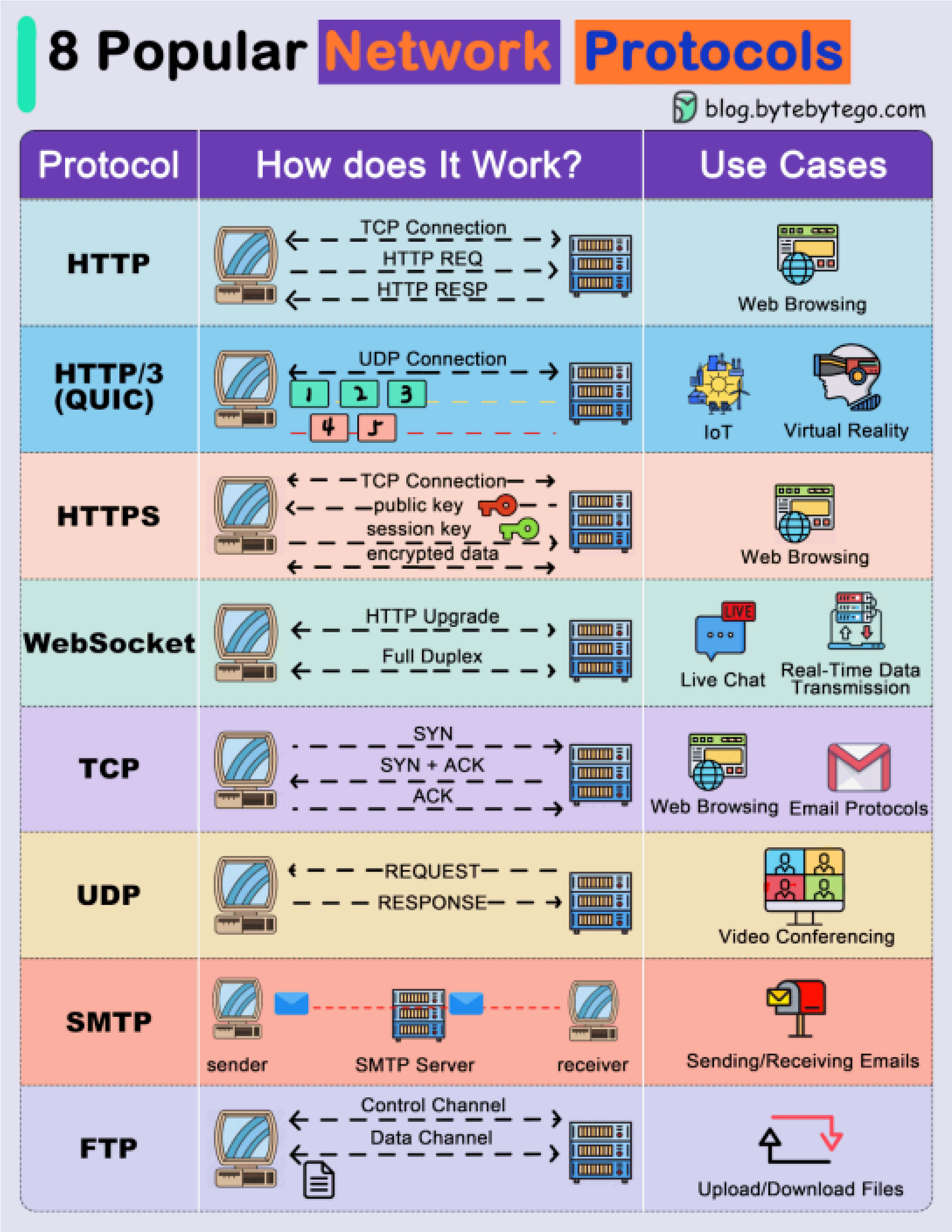
8 Popular Network Protocols

blog.bytebytego.com

Protocol	How does It Work?	Use Cases
HTTP	TCP Connection HTTP REQ HTTP RESP	Web Browsing
HTTP/3 (QUIC)	UDP Connection 1 2 3 4 5	IoT Virtual Reality
HTTPS	TCP Connection public key session key encrypted data	Web Browsing
WebSocket	HTTP Upgrade Full Duplex	LIVE Live Chat Real-Time Data Transmission
TCP	SYN SYN + ACK ACK	Web Browsing Email Protocols
UDP	REQUEST RESPONSE	Video Conferencing
SMTP	sender SMTP Server receiver	Sending/Receiving Emails
FTP	Control Channel Data Channel	Upload/Download Files

TCP (Transmission Control Protocol)

TCP is designed to send packets across the internet and ensure the successful delivery of data and messages over networks. Many application-layer protocols are built on top of TCP.



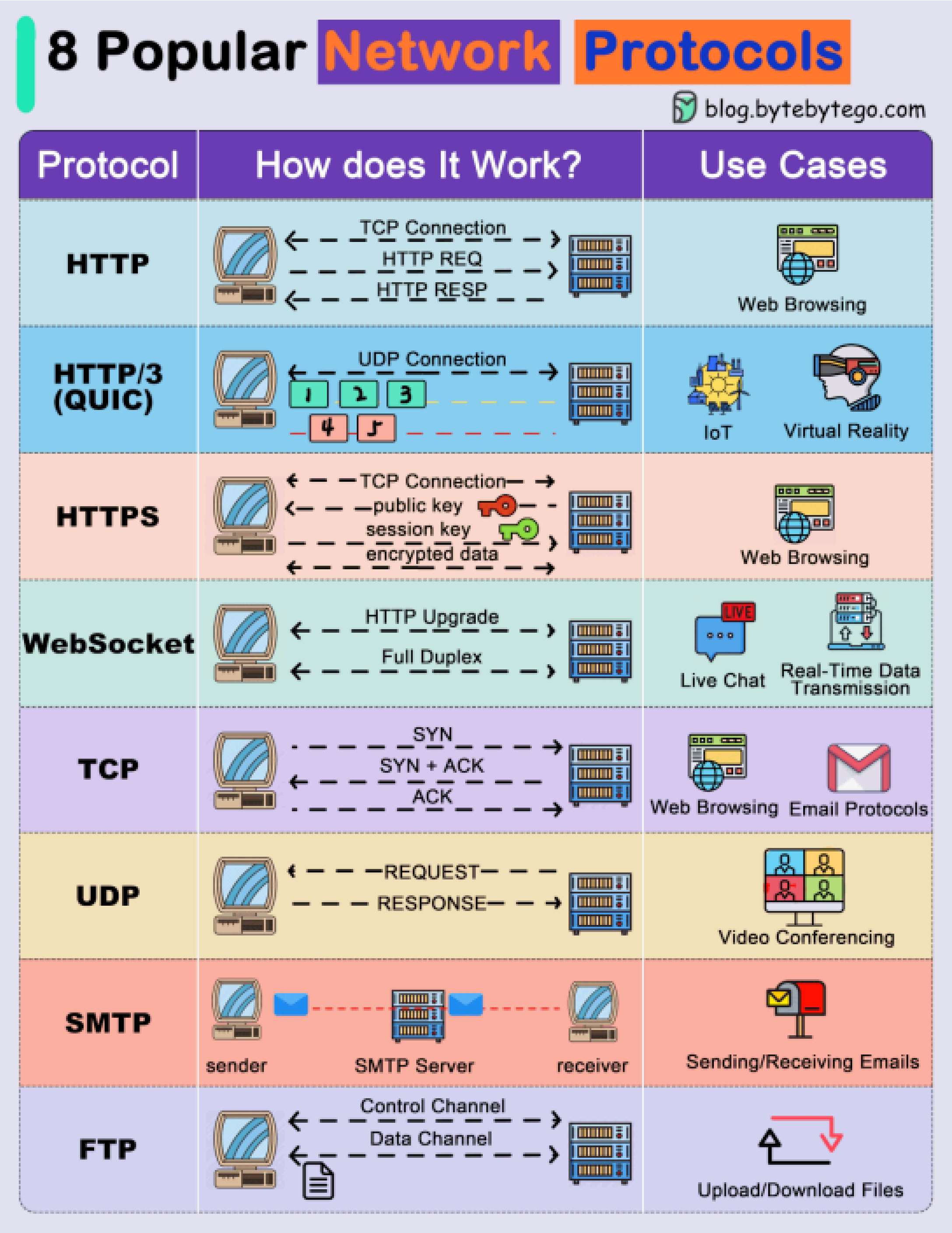
8 Popular Network Protocols

blog.bytebytego.com

Protocol	How does It Work?	Use Cases
HTTP	TCP Connection HTTP REQ HTTP RESP	Web Browsing
HTTP/3 (QUIC)	UDP Connection 1 2 3 4 5	IoT Virtual Reality
HTTPS	TCP Connection public key session key encrypted data	Web Browsing
WebSocket	HTTP Upgrade Full Duplex	LIVE Live Chat Real-Time Data Transmission
TCP	SYN SYN + ACK ACK	Web Browsing Email Protocols
UDP	REQUEST RESPONSE	Video Conferencing
SMTP	sender SMTP Server receiver	Sending/Receiving Emails
FTP	Control Channel Data Channel	Upload/Download Files

UDP (User Datagram Protocol)

UDP sends packets directly to a target computer, without establishing a connection first. UDP is commonly used in time-sensitive communications where occasionally dropping packets is better than waiting. Voice and video traffic are often sent using this protocol.



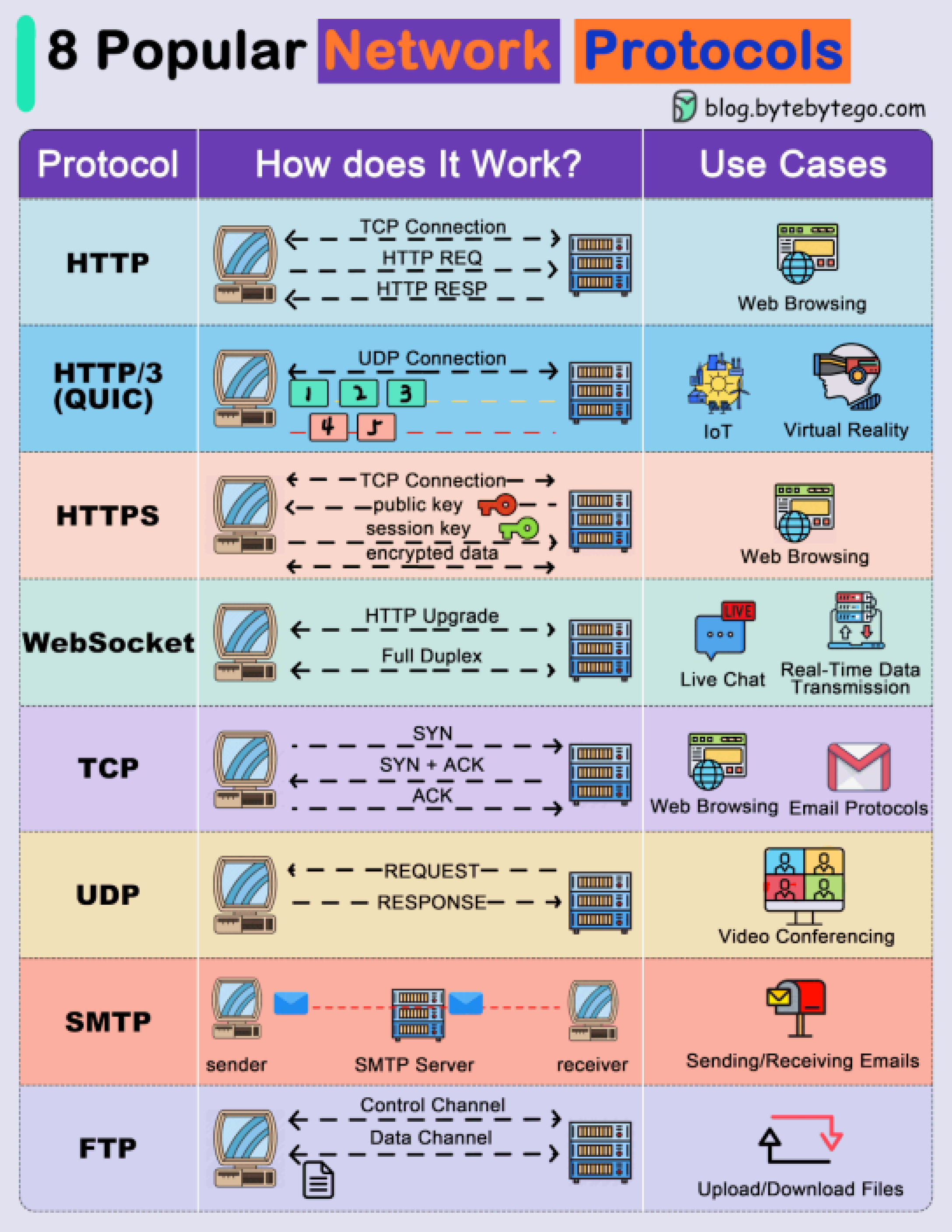
8 Popular Network Protocols

blog.bytebytego.com

Protocol	How does It Work?	Use Cases
HTTP	TCP Connection HTTP REQ HTTP RESP	Web Browsing
HTTP/3 (QUIC)	UDP Connection 1 2 3 4 5	IoT Virtual Reality
HTTPS	TCP Connection public key session key encrypted data	Web Browsing
WebSocket	HTTP Upgrade Full Duplex	Live Chat Real-Time Data Transmission
TCP	SYN SYN + ACK ACK	Web Browsing Email Protocols
UDP	REQUEST RESPONSE	Video Conferencing
SMTP	sender SMTP Server receiver	Sending/Receiving Emails
FTP	Control Channel Data Channel	Upload/Download Files

SMTP (Simple Mail Transfer Protocol) :

SMTP is a standard protocol to transfer electronic mail from one user to another.



8 Popular Network Protocols

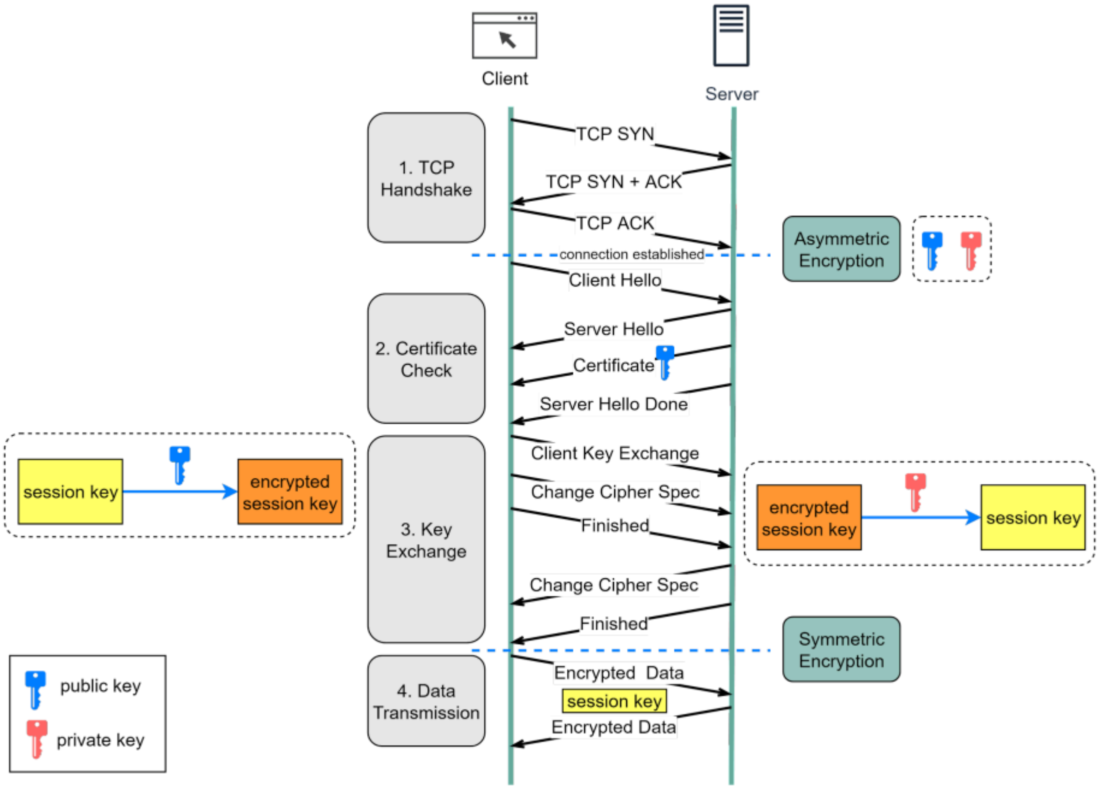
blog.bytebytego.com

Protocol	How does It Work?	Use Cases
HTTP	TCP Connection HTTP REQ HTTP RESP	Web Browsing
HTTP/3 (QUIC)	UDP Connection 1 2 3 4 5	IoT Virtual Reality
HTTPS	TCP Connection public key session key encrypted data	Web Browsing
WebSocket	HTTP Upgrade Full Duplex	Live Chat Real-Time Data Transmission
TCP	SYN SYN + ACK ACK	Web Browsing Email Protocols
UDP	REQUEST RESPONSE	Video Conferencing
SMTP	sender SMTP Server receiver	Sending/Receiving Emails
FTP	Control Channel Data Channel	Upload/Download Files

FTP (File Transfer Protocol)

FTP is used to transfer computer files between client and server. It has separate connections for the control channel and data channel.
More Information On API Protocols :
(Get pdf from here -
https://github.com/MohammedKhan3/SYSRTEM_DESIGN-WORKSHOP/tree/main)

How does HTTPS in simple language:



Step 1 - The client (browser) and the server establish a TCP connection.
Step 2 - The client sends a “client hello” to the server. The message contains a set of necessary encryption algorithms (cipher suites) and the latest TLS version it can support. The server

responds with a “server hello” so the browser knows whether it can support the algorithms and TLS version. The server then sends the SSL certificate to the client. The certificate contains the public key, host name, expiry dates, etc. The client validates the certificate.
Step 3 - After validating the SSL certificate, the client generates a session key and encrypts it using the public key. The server receives the encrypted session key and decrypts it with the private key.

Step 4 - Now that both the client and the server hold the same session key (symmetric encryption), the encrypted data is transmitted in a secure bi-directional channel.

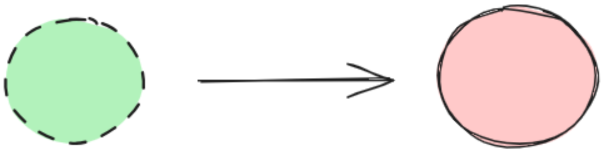
Let's check google’s SSL certificate:

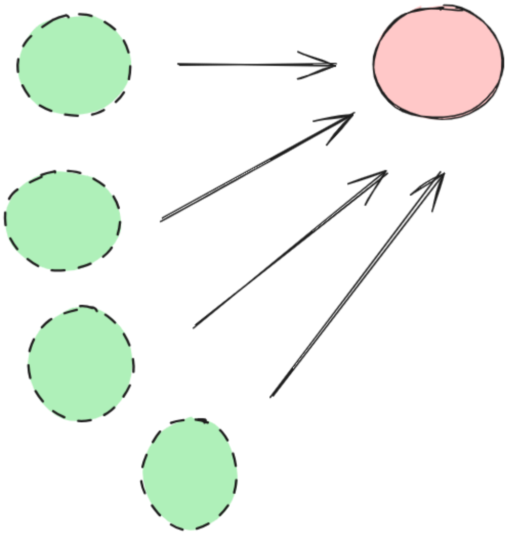
Open git bash and run:

```
$ openssl s_client -servername google.com -connect google.com:443
```

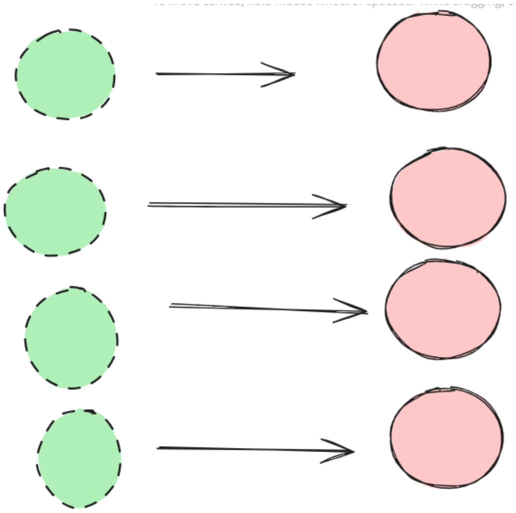
—-----------------Enough on Networks—----------------------------.

HORIZONTAL SCALING:





Increasing the number of servers based on the number of clients is a form of **horizontal scaling.**

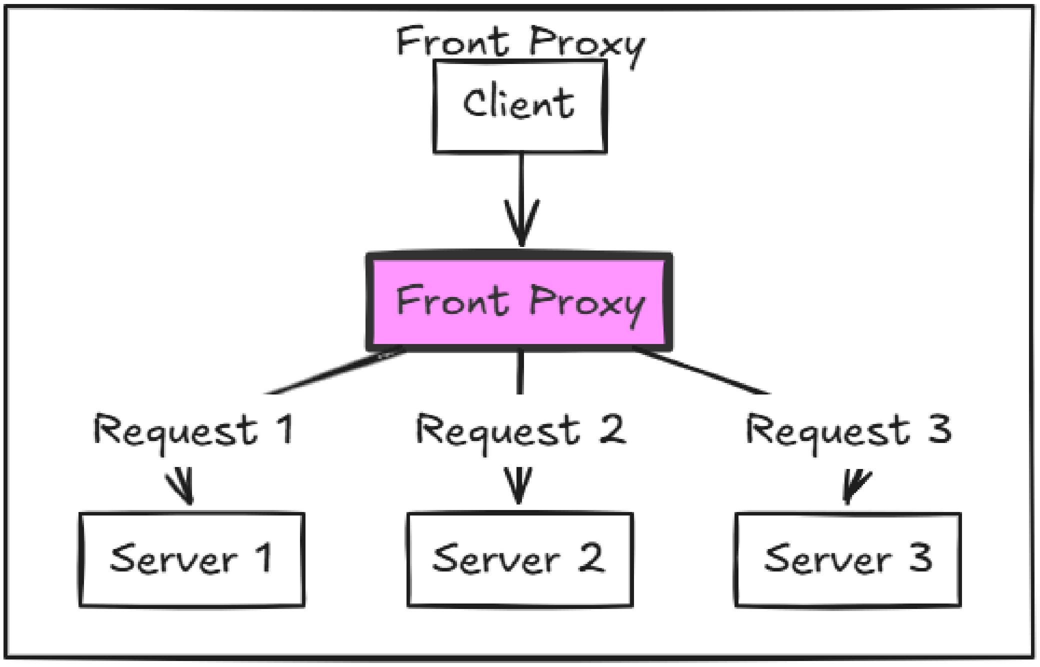


PROXIES:

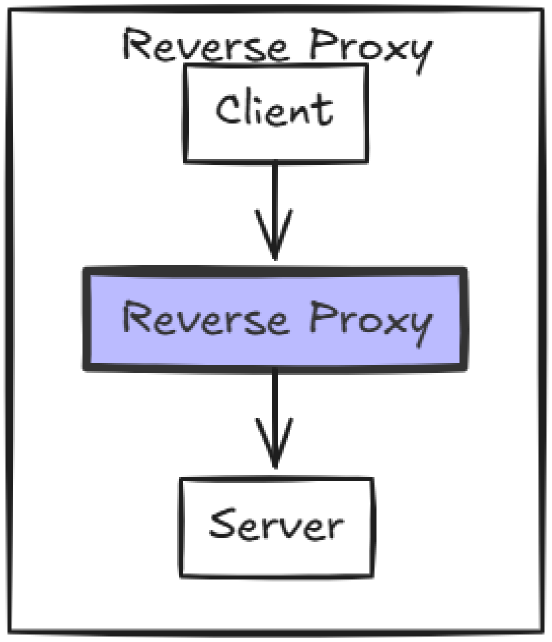
Forward Proxy: A server that sits between client and server and acts on behalf of the client, typically used to mask the client's identity.

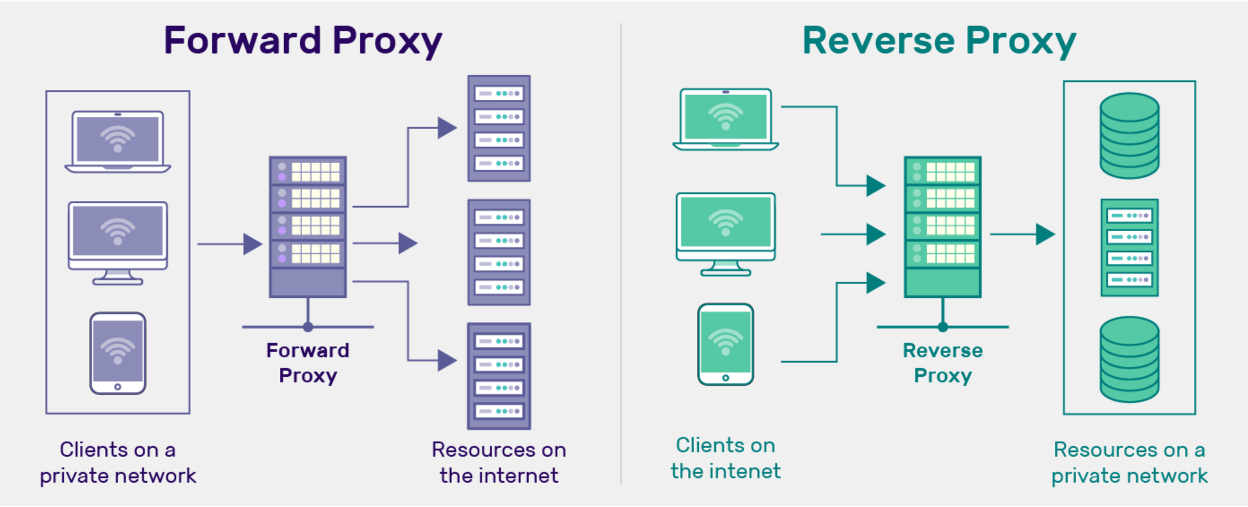
- Protect clients
- Avoid browsing restrictions

- Block access to certain content



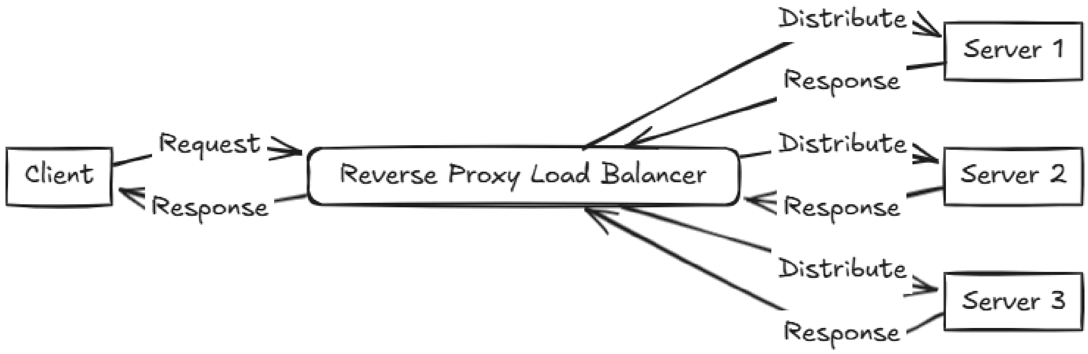
Reverse Proxy: A server that sits between clients and server and acts on behalf of the servers, typically used for load balancing or caching.




Forward Proxy
Forward Proxy
Clients on a private network
Resources on the internet
Reverse Proxy
Reverse Proxy
Clients on the intenet
Resources on a private network

LOAD BALANCER:

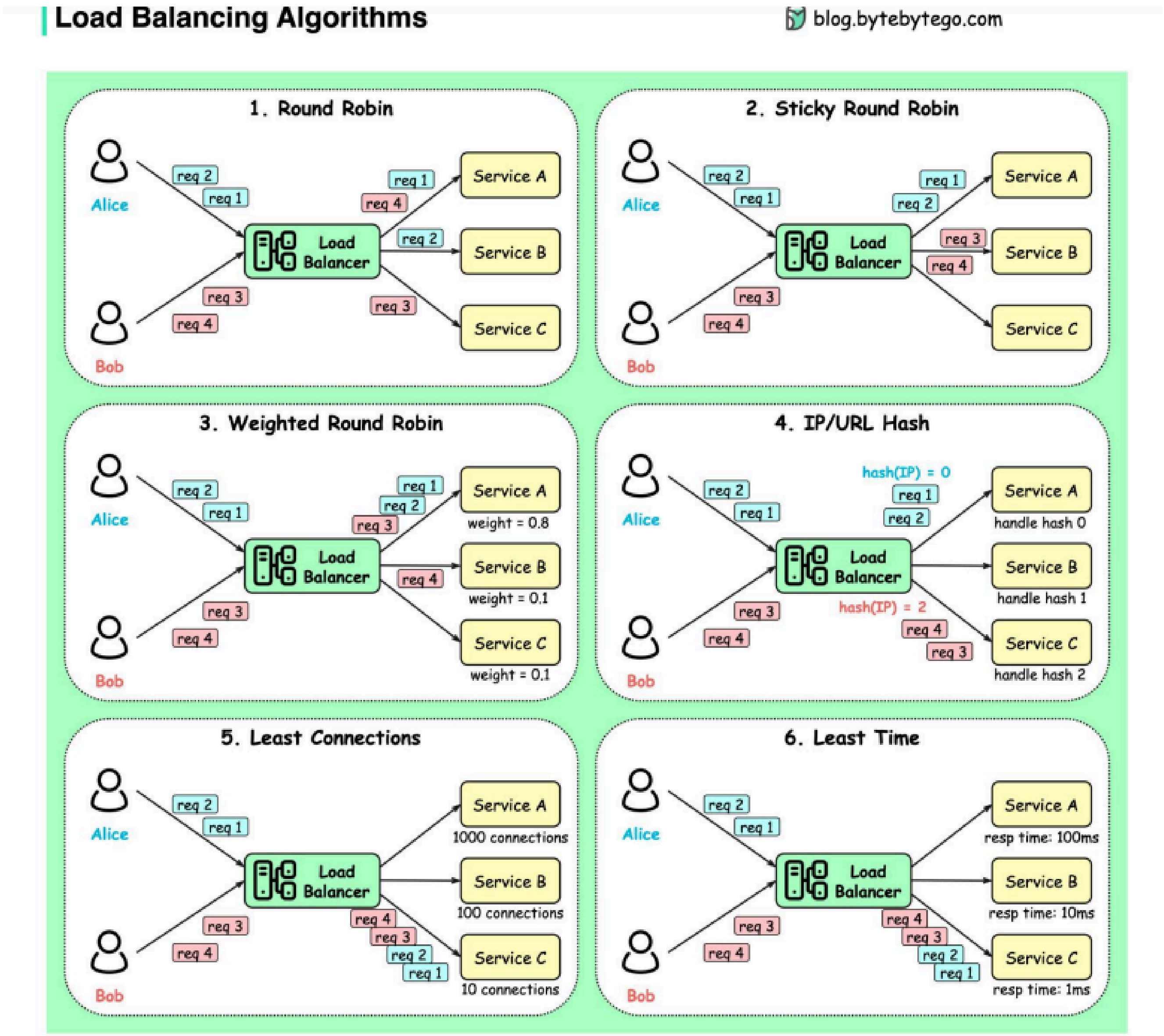
Load Balancer: Is a type of reverse proxy that distributes the traffic across servers. Load balancers can be found in many parts of a system, from the DNS layer all the way to the database layer.



Nginx - A very popular web server that is often used as a reverse proxy and load balancer.

Server-Selection Strategy - How load balancer chooses servers when distributing traffic amongst multiple servers.

Top 6 Load Balancing Algorithms



Static Algorithms:

1. Round robin The client requests are sent to different service instances in sequential order. The services are usually required to be stateless.
2. Sticky round-robin This is an improvement of the round-robin algorithm. If Alice's first request goes to service A, the following requests go to service A as well.
3. Weighted round-robin The admin can specify the weight for each service. The ones with a higher weight handle more

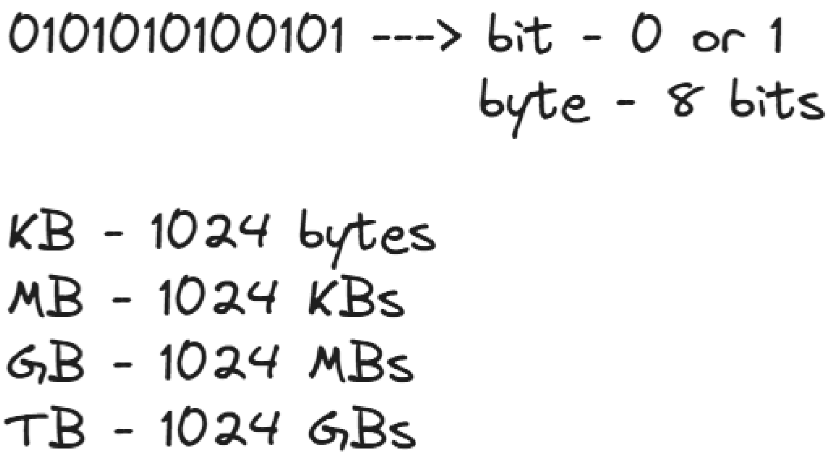
requests than others.

4. Hash This algorithm applies a hash function on the incoming requests' IP or URL. The requests are routed to relevant instances based on the hash function result.

Dynamic Algorithms

1. Least connections A new request is sent to the service instance with the least concurrent connections.
2. Least response time A new request is sent to the service instance with the fastest response time.

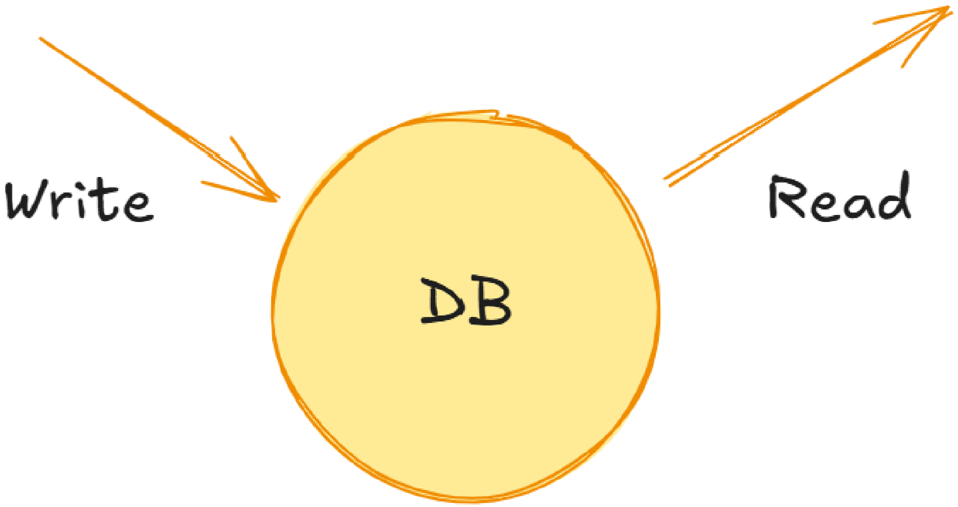
STORAGE:

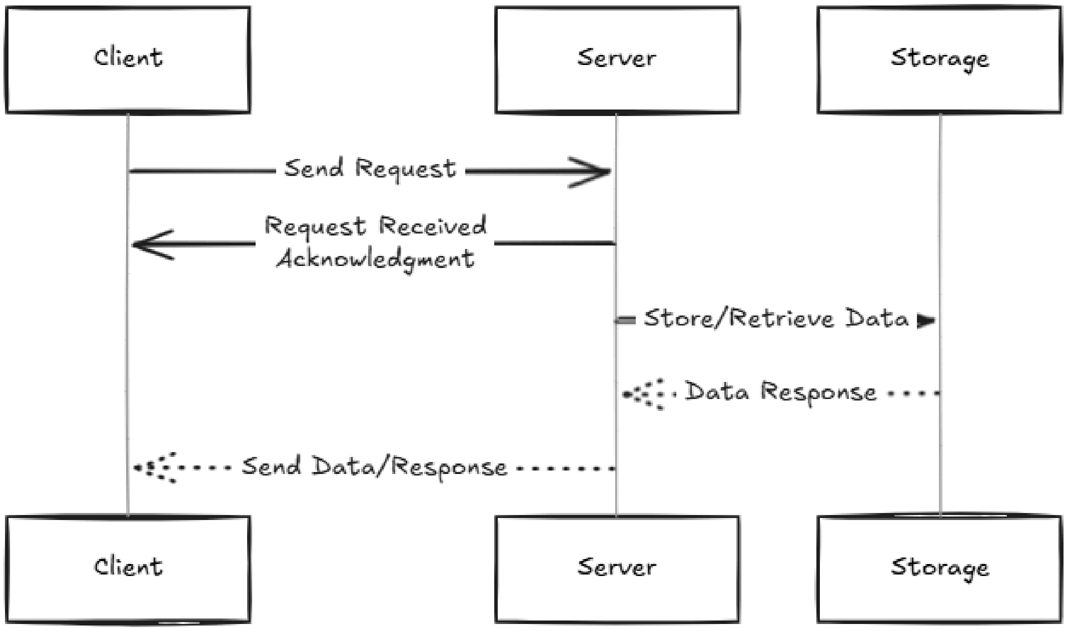


0101010100101 ---> bit - 0 or 1
byte - 8 bits

KB - 1024 bytes
MB - 1024 KBs
GB - 1024 MBs
TB - 1024 GBs

Databases are programs that either use disk or memory to do 2 core things: record data and query data. In general they are themselves servers that are long loved and interact with the rest of your application through network calls, with the help of protocols.

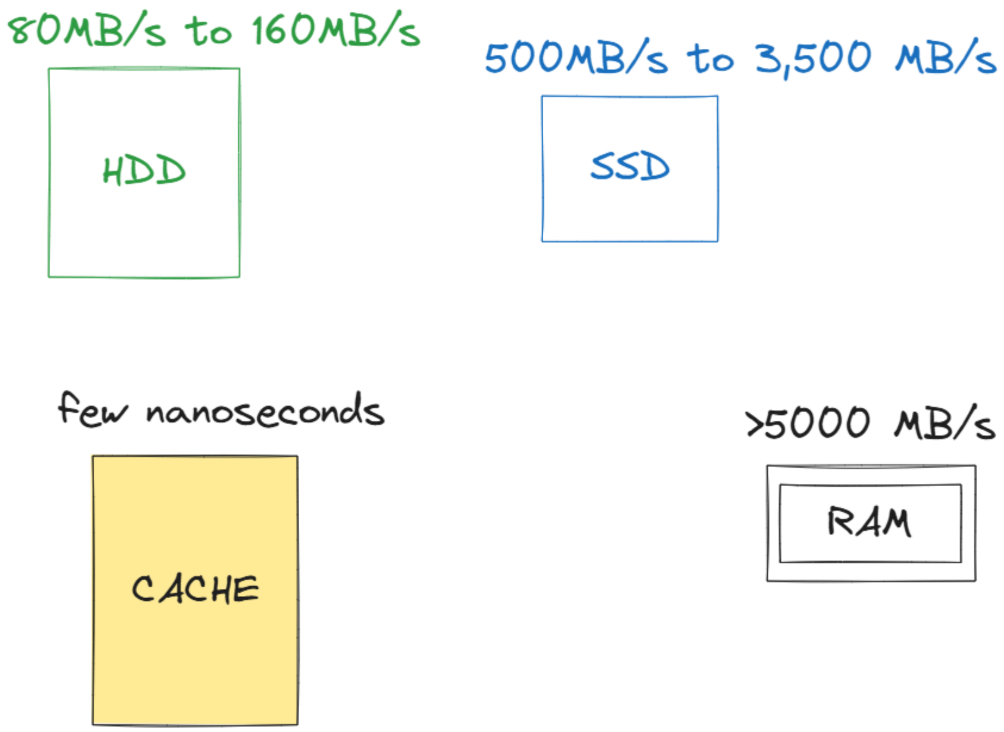




Some databases only keep records in memory, and the users of such databases are aware of the fact that those records may be lost forever if the machine or process dies.

Since data is stored in memory, read and write operations happen almost instantly.

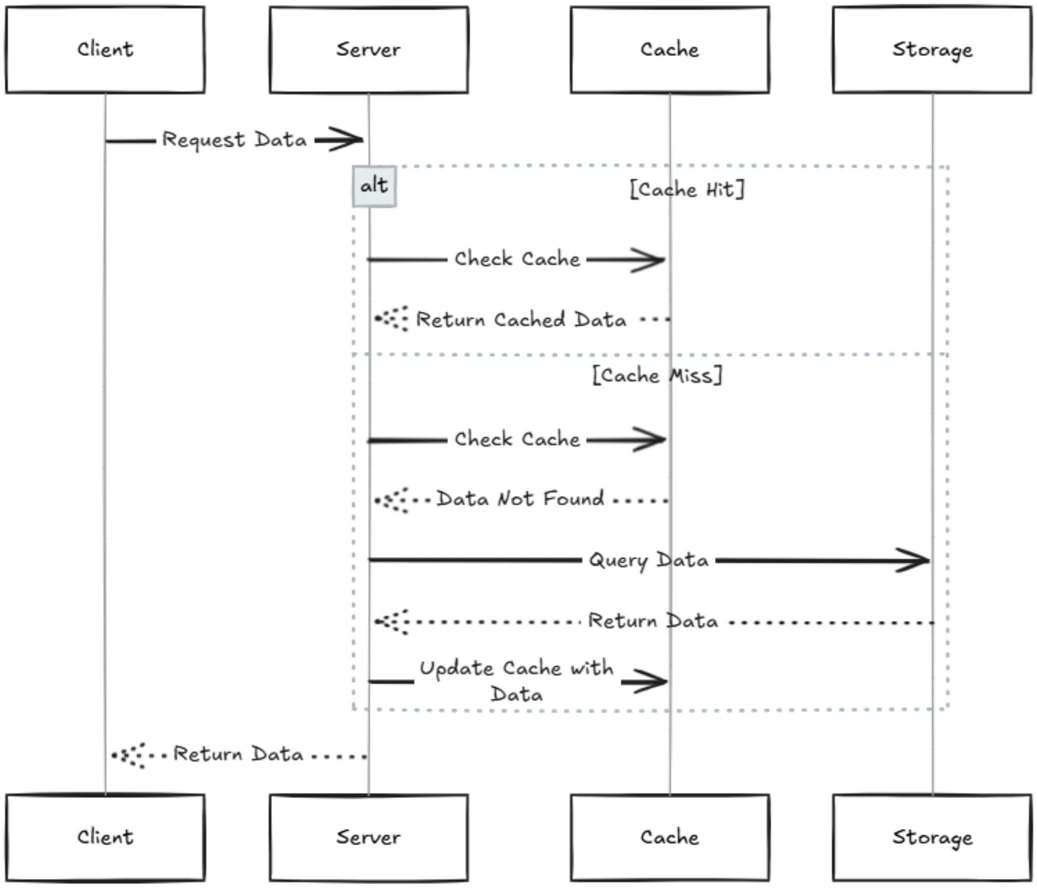
Disk refers to either Hard disk drive(HDD) or Solid state drive(SSD). Data written to disk will persist through power failures and general machine crashes. SSD is faster than HDD. SSD are more expensive than HDDs.



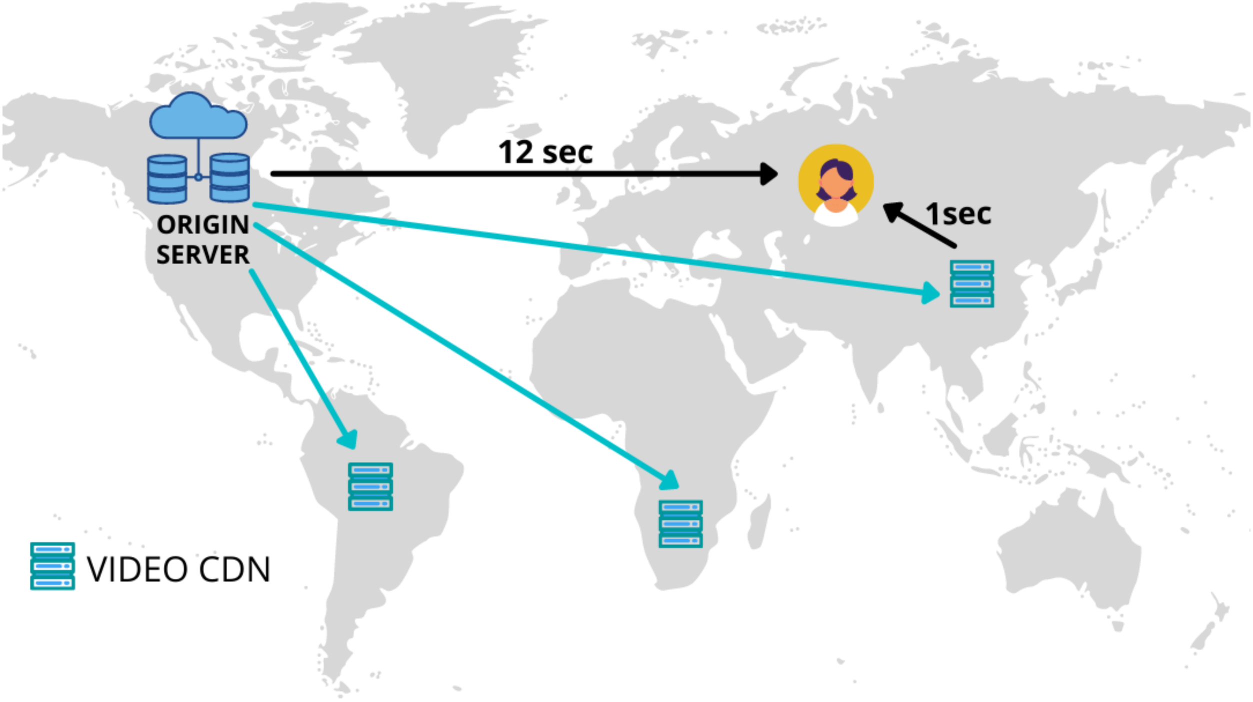
CACHE : A piece of hardware or software that stores data, typically meant to retrieve the data faster than otherwise.

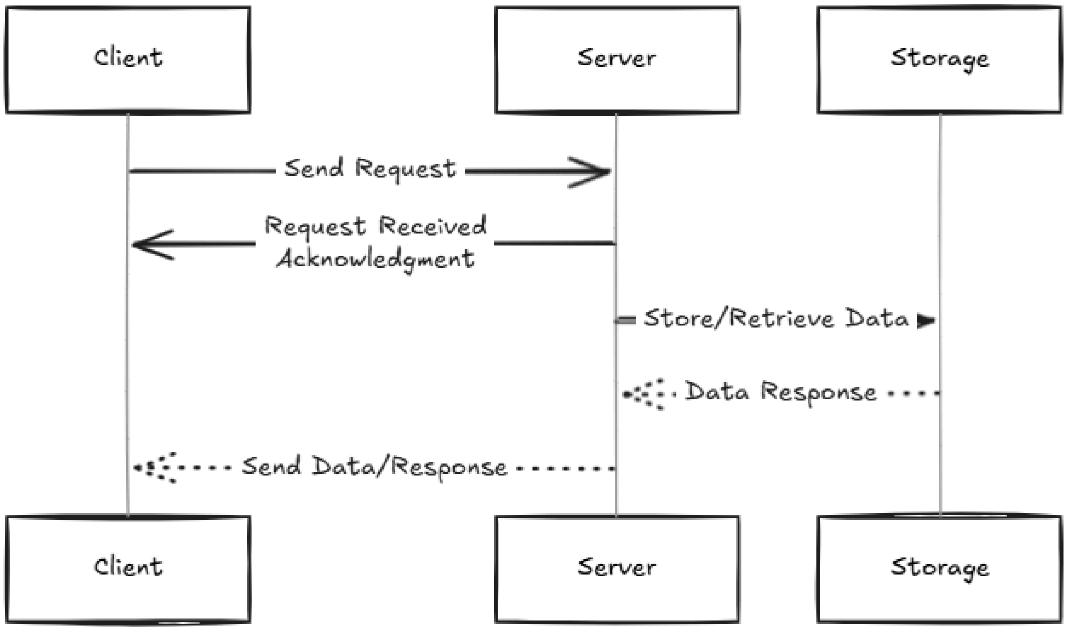
Often used to store responses to network requests as well as results of computationally-long operations.

Data in cache can become stale if the main source of truth for that data gets updated.


Client
Server
Cache
Storage
Request Data
alt
[Cache Hit]
Check Cache
Return Cached Data
[Cache Miss]
Check Cache
Data Not Found
Query Data
Return Data
Update Cache with Data
Return Data
Client
Server
Cache
Storage

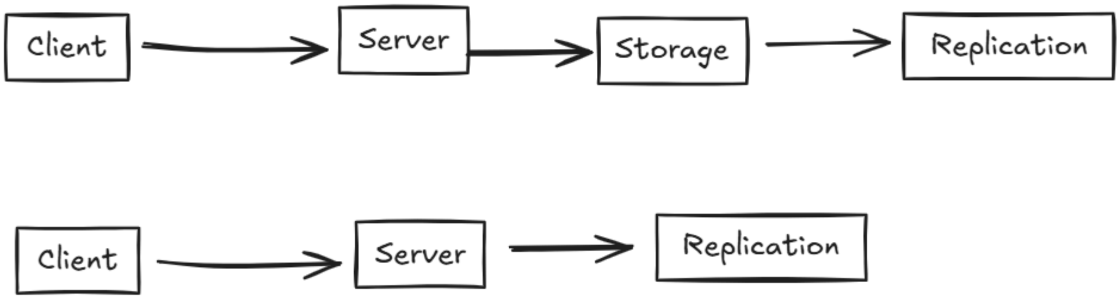
CDN: Content Delivery Network. A CDN is a third party service that acts like a cache for your servers. A CDN has servers all around the world, meaning that the latency to CDN's server will almost always be far better than the latency to your server. Also referred to as PoPs. Two of the most popular CDNs are Cloudflare and Google cloud CDN.

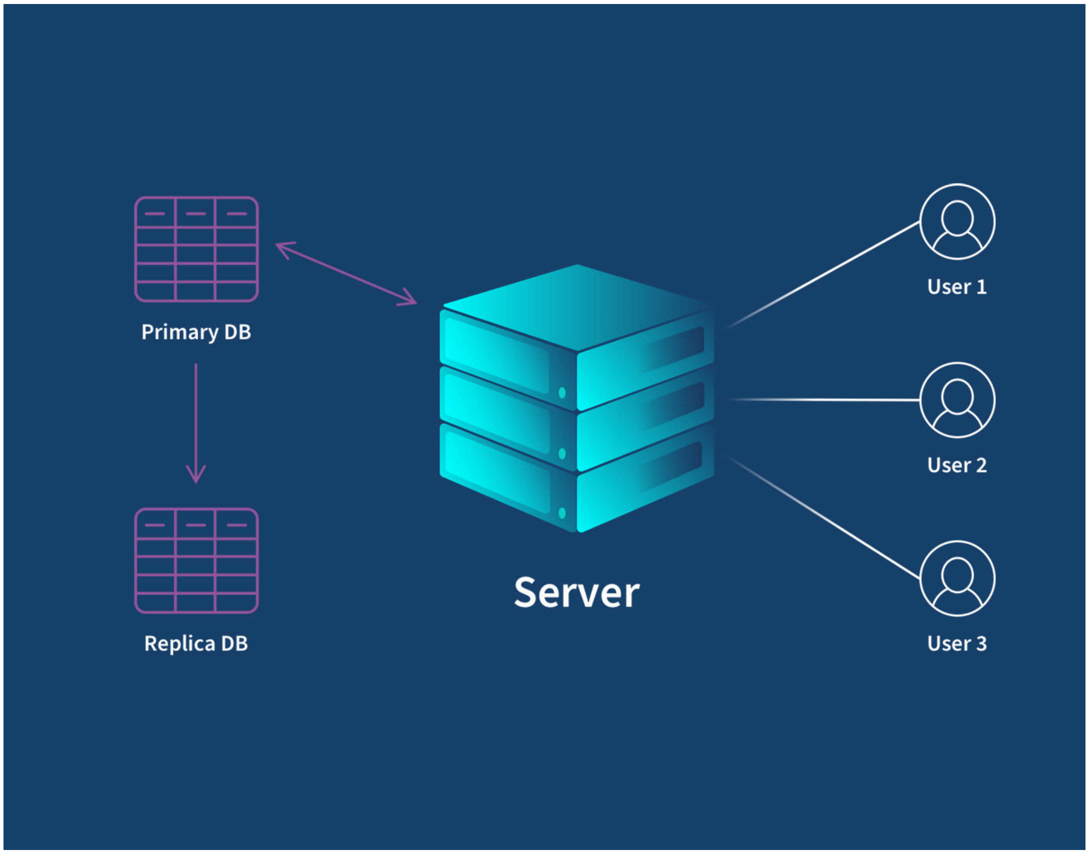




Replication is the act of duplicating the data from one database server to another. This is sometimes used to increase redundancy of your system and tolerate regional failures of instances.

Replication can also be used to bring data closer to your clients.

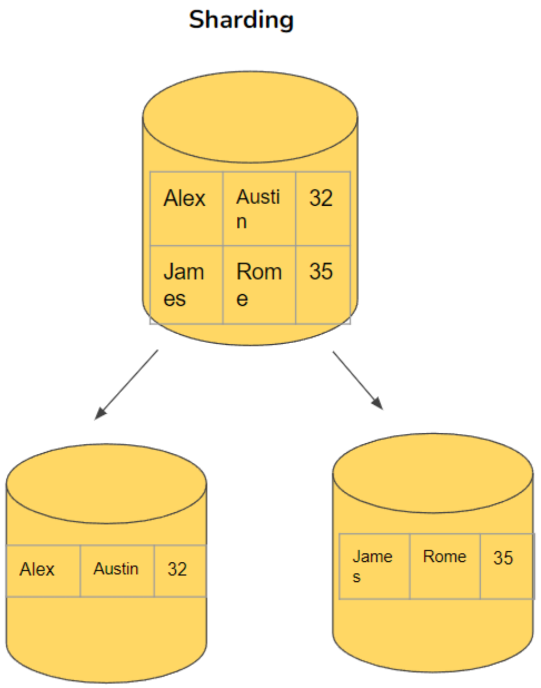


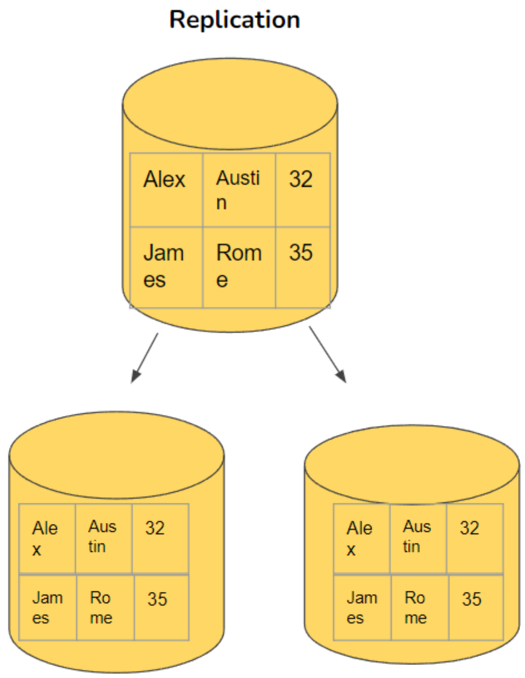


Sharding - also known as data partitioning, is the act of splitting a database into two or more pieces called shards and is typically done to increase throughput of your database.

Example:

You can perform sharding based on the client's region.

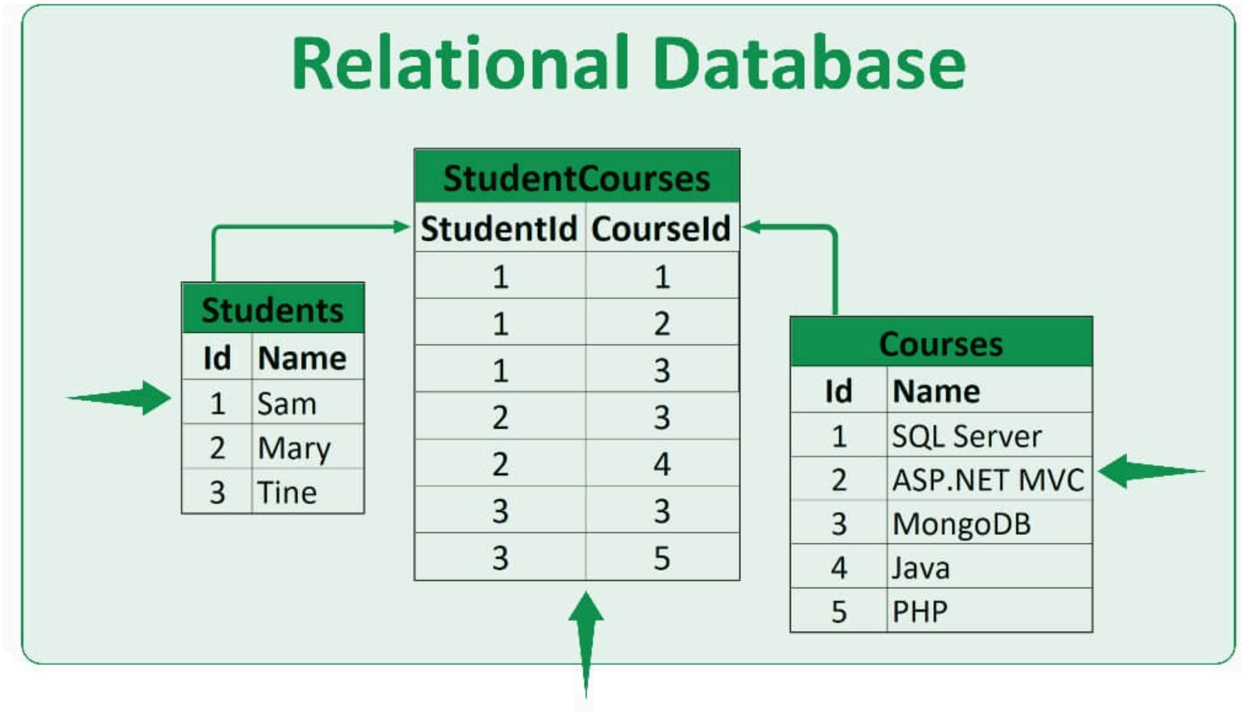




Relational databases:

A type of structured database in which data is stored in the following tabular format.

Often supports powerful queries using SQL.



Students

Id	Name
1	Sam
2	Mary
3	Tine

StudentCourses

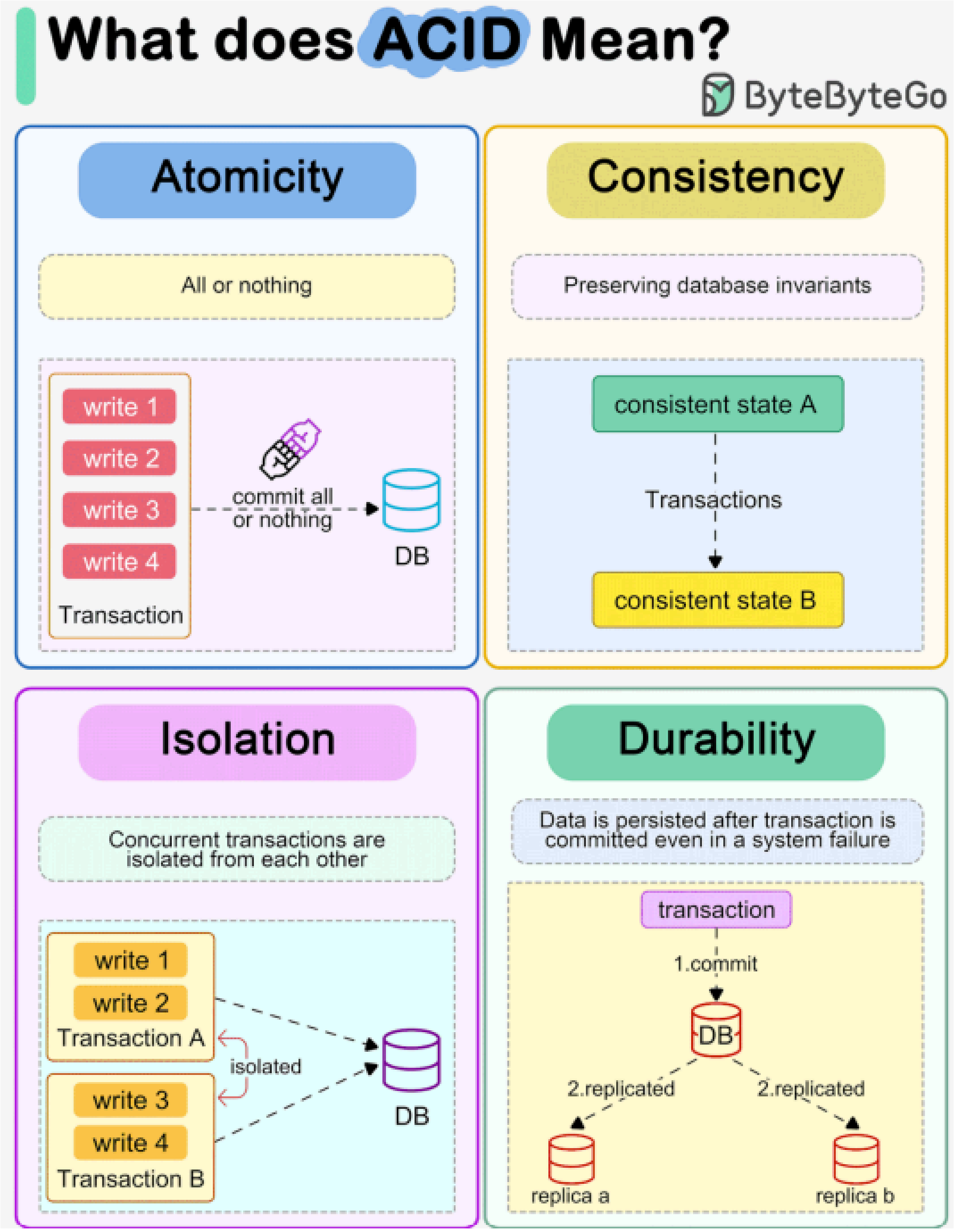
StudentId	CourseId
1	1
1	2
1	3
2	3
2	4
3	3
3	5

Courses

Id	Name
1	SQL Server
2	ASP.NET MVC
3	MongoDB
4	Java
5	PHP

ACID Transaction:

Atomicity – ensures that each transaction is treated as a single unit, which either completes entirely or does not happen at all.
Consistency – ensures that a transaction brings the database from one valid state to another.
Isolation – ensures that the operations in a transaction are isolated from other transactions until it is completed.
Durability – ensures that once a transaction is committed, it will remain committed even in the case of a system failure.

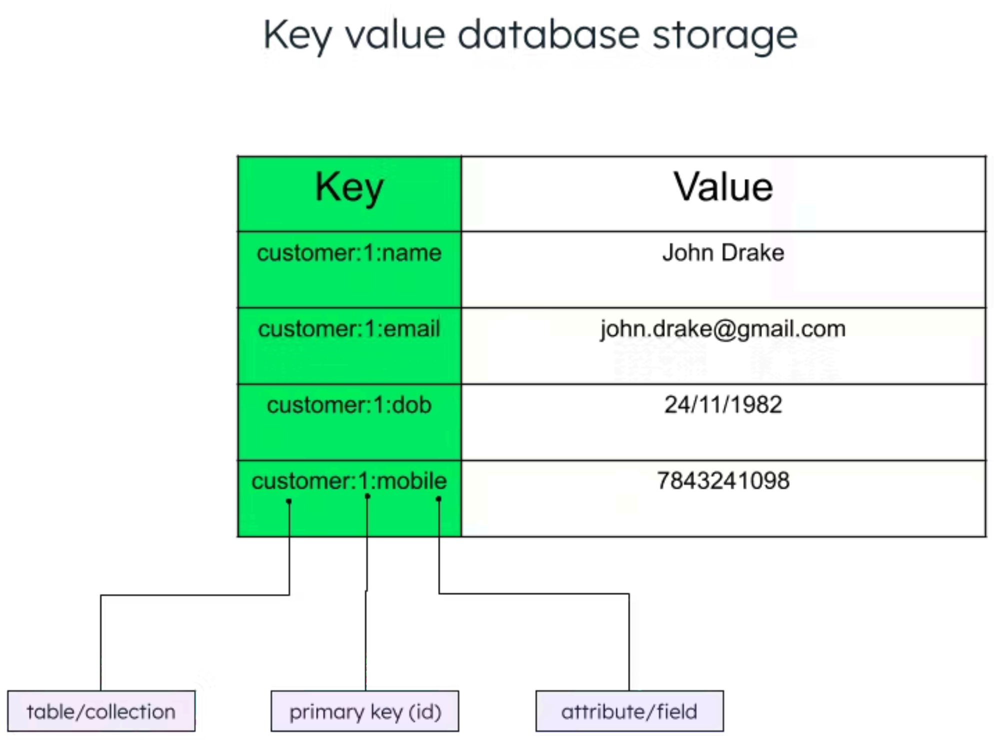


Non-Relational DB:

In contrast with relational databases, a type of database that is free imposed, tabular like structure.

Key-Value Store:

Often used for caching.



Key value database storage

Key	Value
customer:1:name	John Drake
customer:1:email	john.drake@gmail.com
customer:1:dob	24/11/1982
customer:1:mobile	7843241098

Blob Storage:

Usually people use this to store things like large binaries, database snapshots, images, videos or other static assets that a website might have.

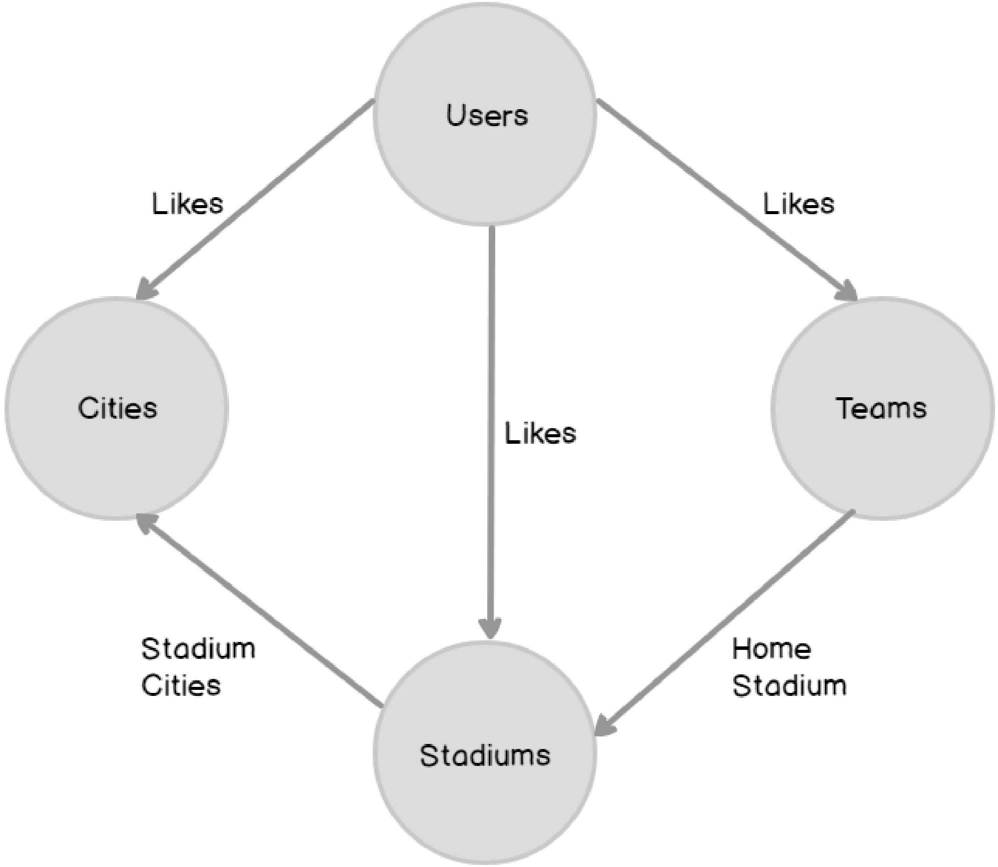
Example: GCS or S3

Time Series Database:

A TSDB is a special kind of database optimized for storing and analyzing time-indexed data.

Graph Database:

A type of database that stores data following the graph data model.



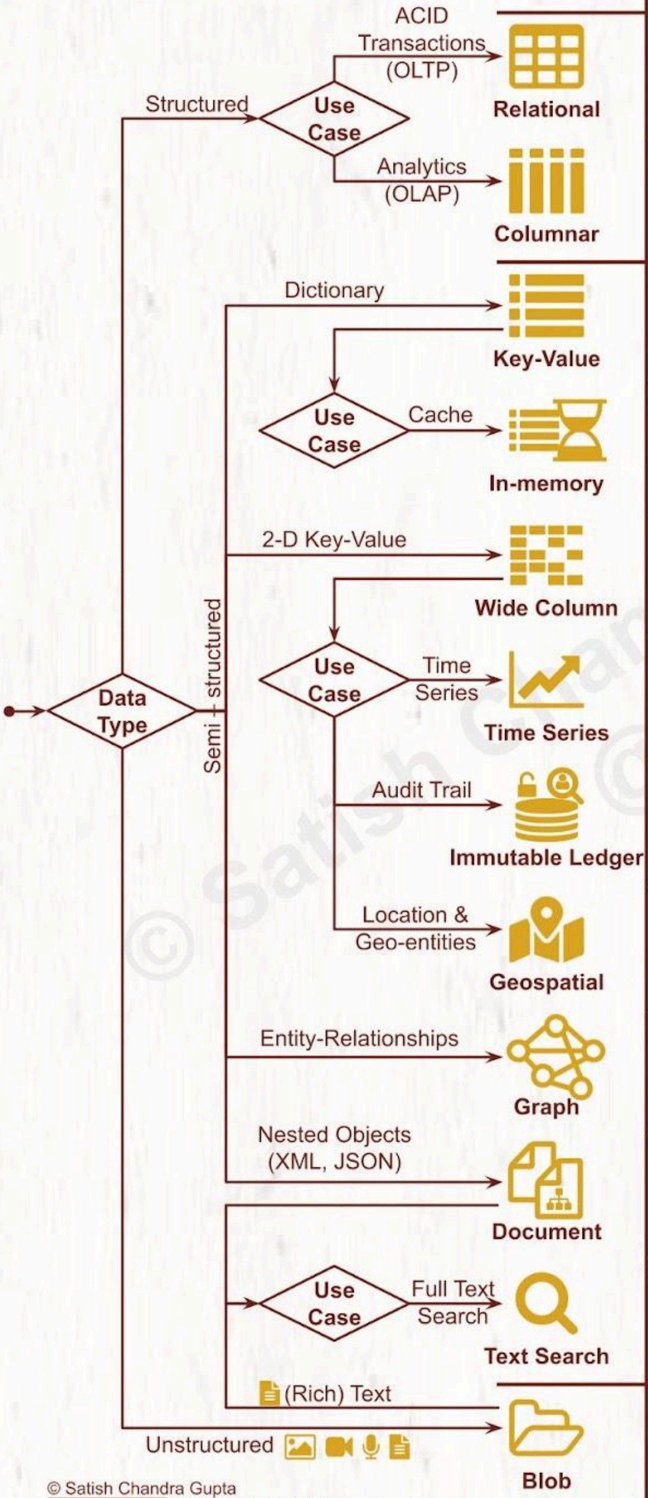
Vector Database:

A vector database is a type of database optimized for storing, indexing, and searching high-dimensional vector data. These are commonly used in machine learning, AI, and recommendation systems where vector embeddings represent complex data like text, images, and audio.

Example: PineCone

SQL vs. NoSQL: Cheatsheet for AWS, Azure, and Google Cloud

scgupta.link/datastores



Data Type

Structured → Use Case → ACID Transactions (OLTP) → Relational

Use Case → Analytics (OLAP) → Columnar

Semi-structured → Dictionary → Key-Value

Key-Value → Use Case → Cache → In-memory

Semi-structured → 2-D Key-Value → Wide Column

Wide Column → Use Case → Time Series → Time Series

Use Case → Audit Trail → Immutable Ledger

Use Case → Location & Geo-entities → Geospatial

Semi-structured → Entity-Relationships → Graph

Semi-structured → Nested Objects (XML, JSON) → Document

Document → Use Case → Full Text Search → Text Search

(Rich) Text → Blob

Unstructured → Blob

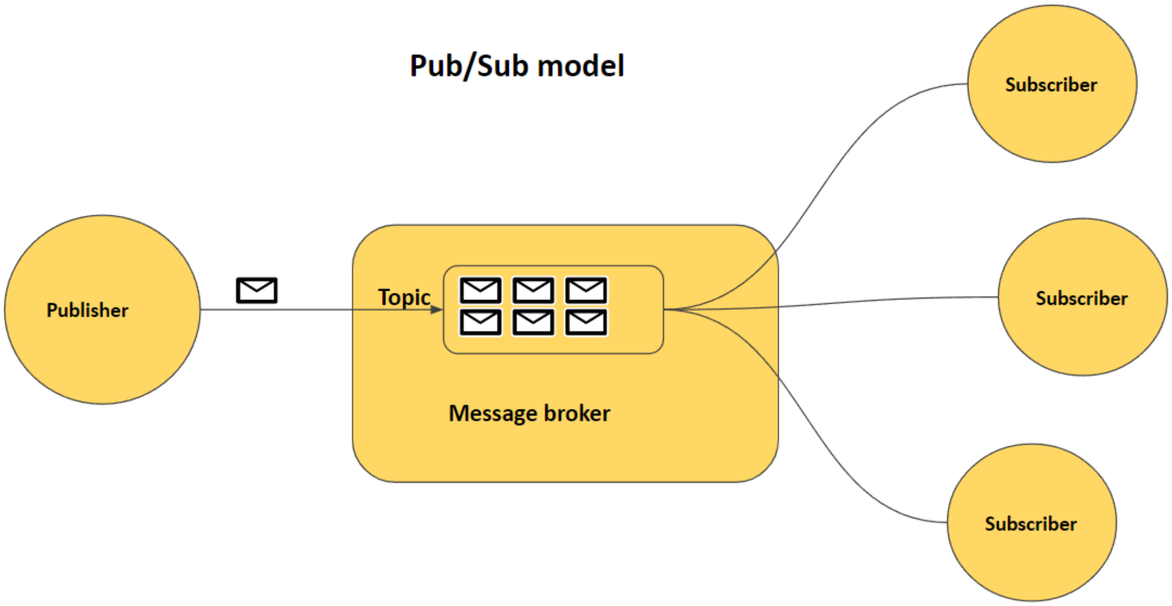
	aws	Azure	Google Cloud	Cloud Agnostic
Relational	RDS, Aurora	Azure SQL Database	Cloud SQL, Cloud Spanner	SQL Server, Oracle, DB2, MySQL, PostgreSQL
Columnar	RedShift	Azure Synapse	BigQuery	Snowflake, ClickHouse, Druid, Pinot, Databricks
Key-Value	DynamoDB	Cosmos DB	BigTable	Redis, ScyllaDB, Ignite
In-memory	ElastiCache	Azure Cache for Redis	Memory-store	Redis, Memcached, Hazelcast, Ignite
Wide Column	Keyspaces	Cosmos DB	BigTable	HBase, Cassandra, ScyllaDB
Time Series	Timestream	Cosmos DB	BigTable, BigQuery	OpenTSDB, InfluxDB, ScyllaDB
Immutable Ledger	Quantum Ledger Database (QLDB)	Azure SQL Database Ledger	✗	Hyperledger Fabric
Geospatial	Keyspaces	Cosmos DB	BigTable, BigQuery	Solr, PostGIS, MongoDB (GeoJSON)
Graph	Neptune	Cosmos DB	JanusGraph + BigTable	OrientDB, Neo4J, Giraph
Document	Document DB	Cosmos DB	Firestore	MongoDB, Couchbase, Solr
Text Search	Open-Search, Cloud-Search	Cognitive Search	Search APIs on Datastores	Elastic-Search, Solr, Elassandra
Blob	S3	Blob Storage	Cloud Storage	HDFS, MinIO

NoSQL

scgupta.me
twitter.com/scgupta
linkedin.com/in/scgupta

—------------Storage done—-------------------------------

IS CLIENT-SERVER THE ONLY MODEL? NO.
We also have a Publisher and subscriber model.



—---------------XXX—--------------------------------------

Before we move on to Microservices and Monolithic Services:

CI/CD:

Let's look at high level systems design and CI/CD Pipeline.

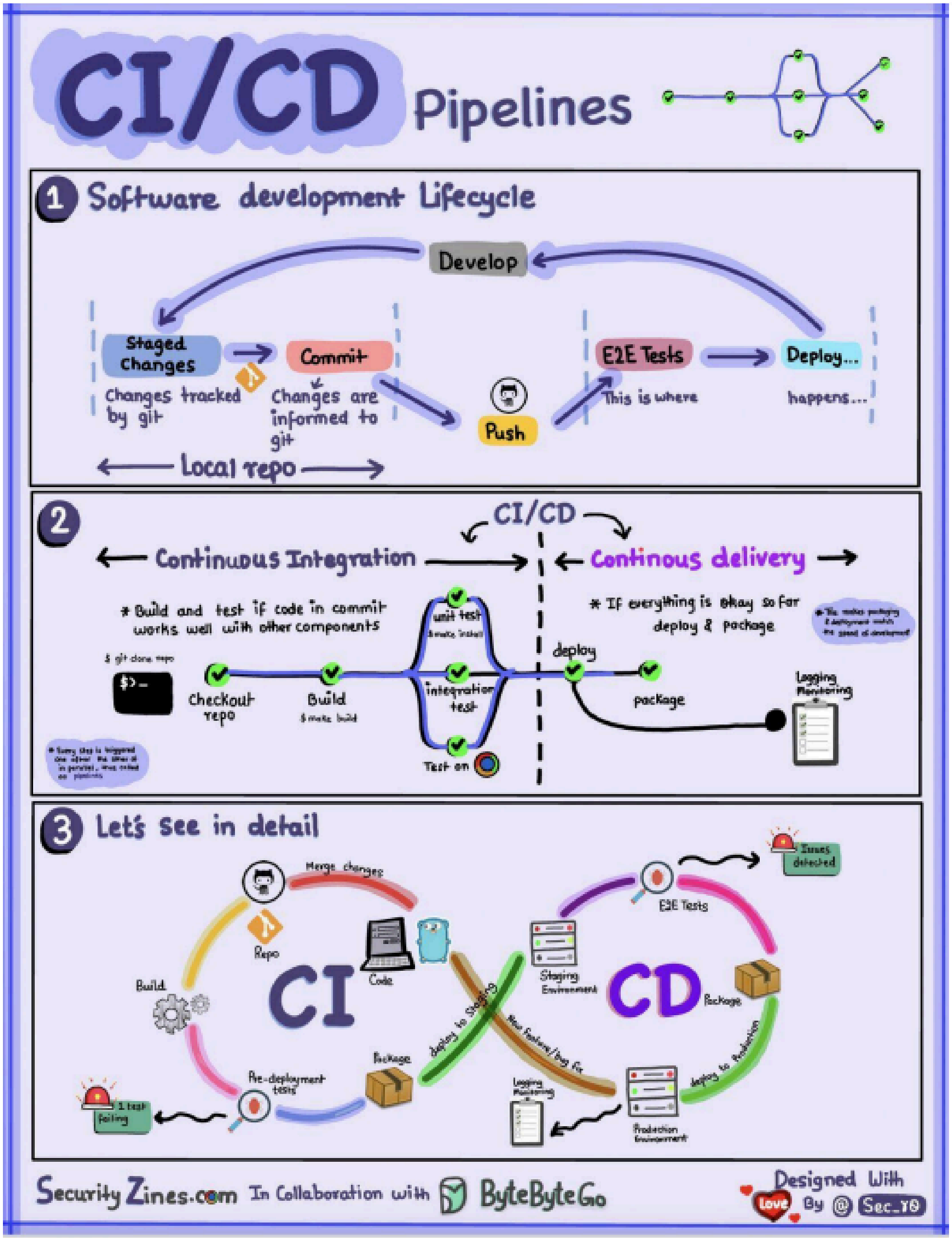
Section 1 - SDLC with CI/CD The software development life cycle (SDLC) consists of several key stages: development, testing, deployment, and maintenance. CI/CD automates and integrates these stages to enable faster, more reliable releases. When code is pushed to a git repository, it triggers an automated build and test process. End-to-end (e2e) test cases are run to validate the code. If tests pass, the code can be automatically deployed to staging/production. If issues are found, the code is sent back to development for bug fixing. This automation provides fast feedback to developers and reduces risk of bugs in production.

Section 2 - Difference between CI and CD- Continuous Integration (CI) automates the build, test, and merge process. It runs tests whenever code is committed to detect integration issues early. This encourages frequent code commits and rapid feedback.

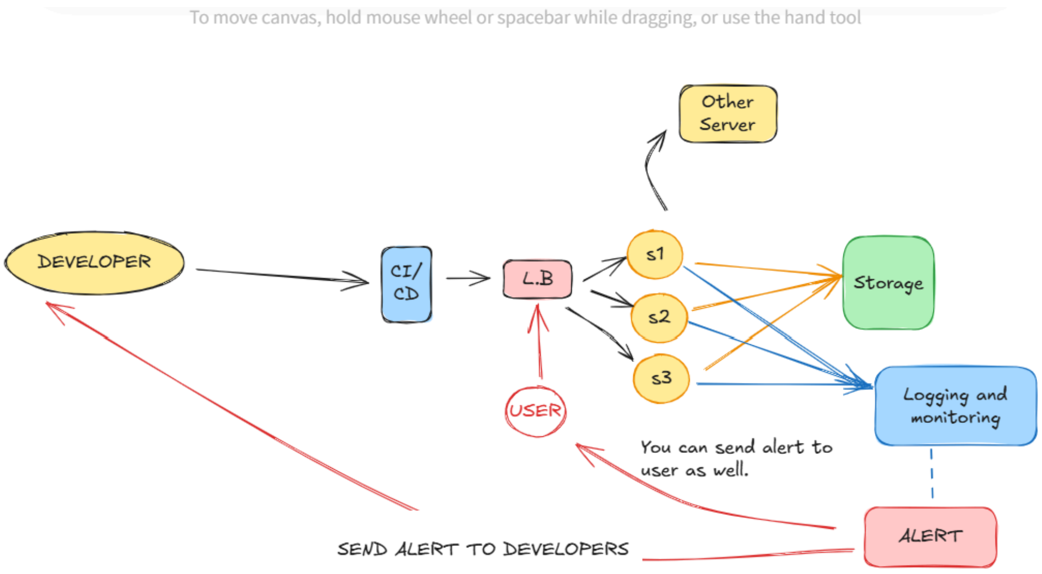
Continuous Delivery (CD) automates release processes like infrastructure changes and deployment. It ensures software can be released reliably at any time through automated workflows. CD may also automate the manual testing and approval steps required before production deployment.

Section 3 - CI/CD Pipeline A typical CI/CD pipeline has several connected stages:

-Developer commits code changes to source control
-CI server detects changes and triggers build
-Code is compiled, tested (unit, integration tests)
-Test results reported to developer
-On success, artifacts are deployed to staging environments
-Further testing may be done on staging before release
-CD system deploys approved changes to production

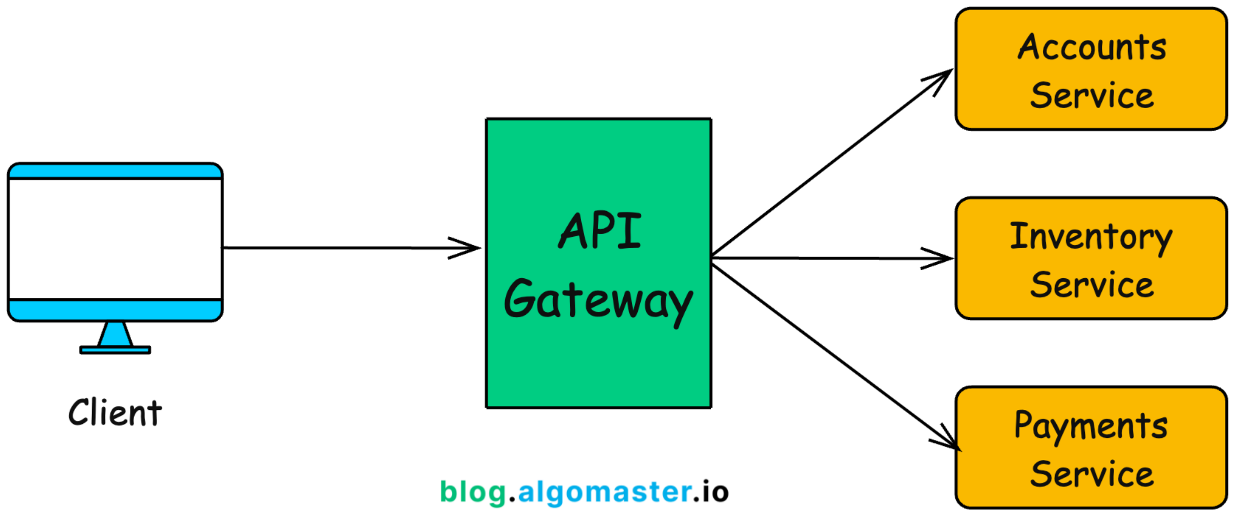

CI/CD Pipelines
1 Software development Lifecycle
Develop
Staged Changes
Commit
Changes tracked by git
Changes are informed to git
Push
E2E Tests
Deploy...
This is where
happens...
Local repo
2
CI/CD
Continuous Integration
Continous delivery
* Build and test if code in commit works well with other components
* If everything is okay so far deploy & package
* This makes packaging & deployment match the speed of development
$ git clone repo
$>_
Checkout repo
Build
$ make build
unit test
$ make install
integration test
Test on
deploy
package
Logging Monitoring
* Every step is triggered one after the other or in parallel, thus called as pipelines
3 Let's see in detail
Merge changes
Repo
Code
Build
CI
Package
Pre-deployment tests
1 test failing
deploy to Staging
New feature/bug fix
Staging Environment
E2E Tests
Issues detected
CD
Package
deploy to Production
Production Environment
Logging Monitoring
SecurityZines.com In Collaboration with ByteByteGo
Designed With Love By @ Sec_Y0

COMBINING EVERYTHING:


To move canvas, hold mouse wheel or spacebar while dragging, or use the hand tool
Other Server
DEVELOPER
CI/ CD
L.B
s1
s2
s3
Storage
Logging and monitoring
USER
You can send alert to user as well.
ALERT
SEND ALERT TO DEVELOPERS

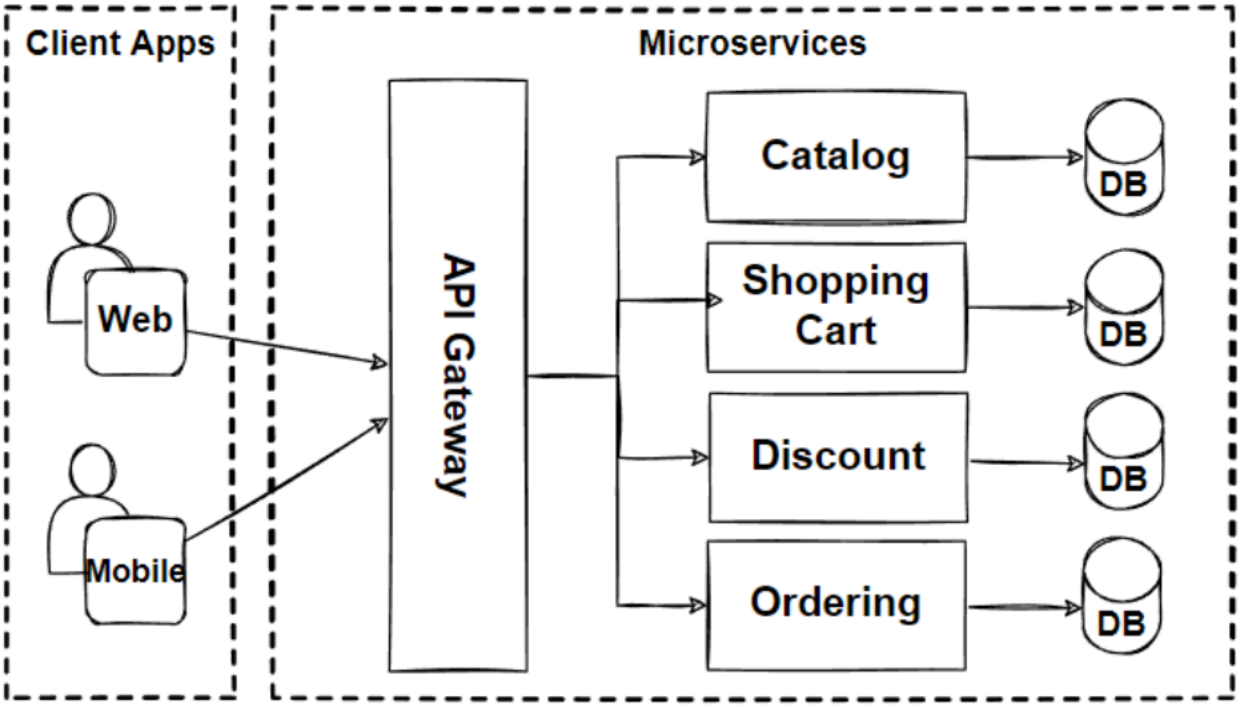
API GATEWAY:

An **API Gateway** is like a **traffic controller** for APIs. It sits between **clients** (like web apps, mobile apps, or other services) and **backend services**, managing and routing API requests efficiently.



Single Entry Point 🏠 → All API requests go through **one** place.
Security & Authentication 🔐 → Blocks unauthorized users (e.g., JWT, OAuth).
Load Balancing ⚖️ → Distributes requests across multiple backend servers.
Rate Limiting 🚦 → Prevents overuse (e.g., limits requests per second).
Caching ⚡ → Stores responses temporarily to reduce backend load.

Microservices:

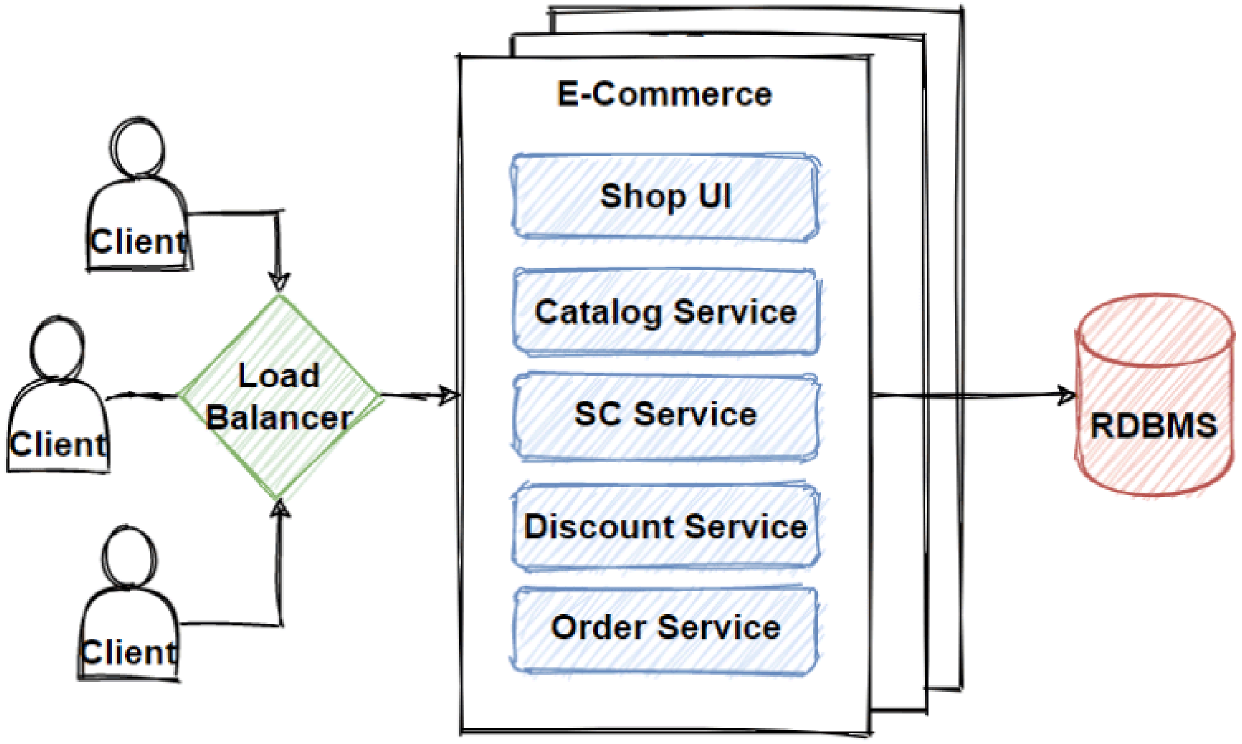


- Architecture where an app is built as a collection of small, independent services.
- Each service is deployed independently.
- Easily scalable – scale only the required service
- Different teams can work on different services using different technologies
- Failure in one service doesn't affect others.
- Can use different programming languages and databases for different services
- Example: Netflix, Amazon,Uber.

Monolith Service:

- A single unified application where all components are tightly coupled.
- Entire application is deployed as a single unit.
- Hard to scale – need to scale the entire application.
- Single team works on the entire application with a single tech stack.
- A bug or crash can bring down the entire application.
- Can be efficient for small applications but struggles with growth.
- Limited to a single tech stack.

Monolithic Architecture



—————————————————————END————————————————————

Sources:

Bytego, Byte. *BIG ARCHIVE: SYSTEM DESIGN 2023*. Publisher Name, 2023.

SystemsExpert

https://youtu.be/F2FmTdLtb_4?si=miLZkkrDiWzJl94o

https://algomaster.io/

https://www.geeksforgeeks.org/getting-started-with-system-design/