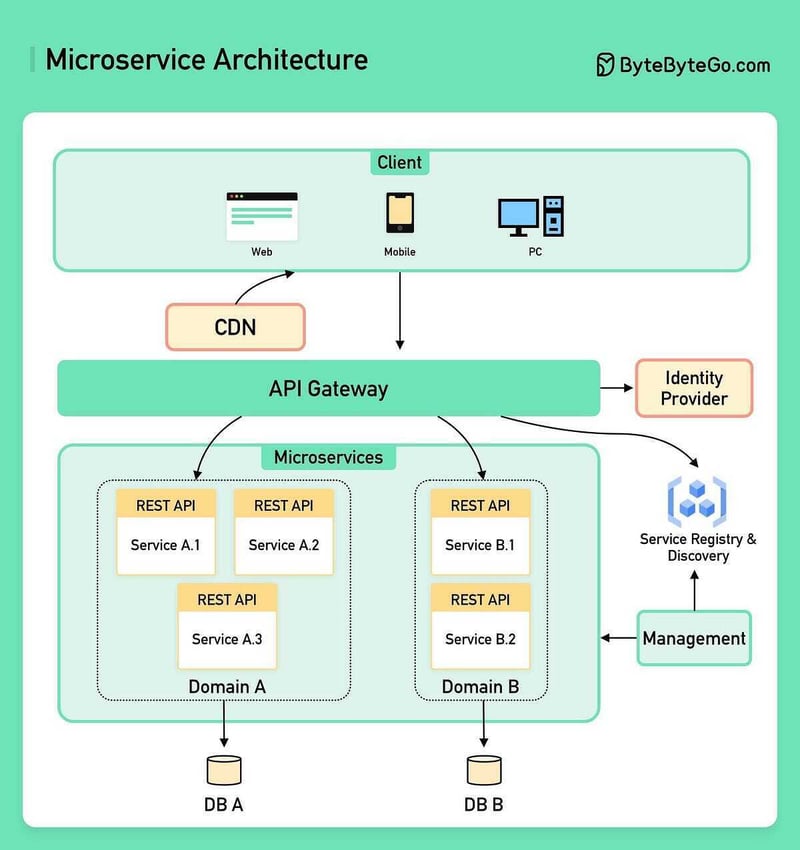
# Micro services vs Load Balancers

* Microservices are a different (micro-) application each. Each with its own application logic and database.
* Load Balancers are usually used to distribute client requests to a cluster of instances of the same application.

That means: You can also use a load balancer to distribute requests for a microservice that is deployed in a cluster with many instances. But a load balancer can also be used to distribute requests to many instances of a large monolithic application (as opposed to micro).



# 19 Microservices Patterns for System Design Interviews

<https://dev.to/somadevtoo/19-microservices-patterns-for-system-design-interviews-3o39>

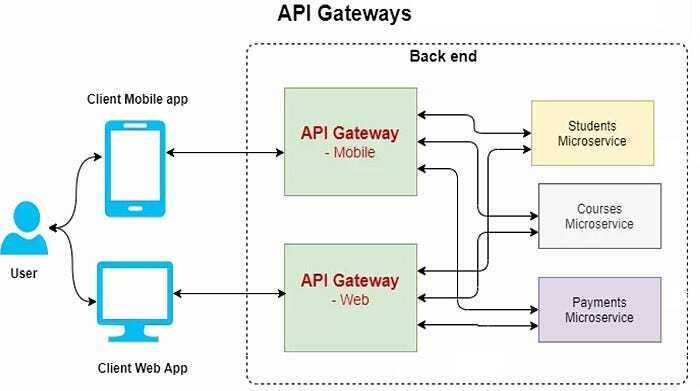
# API Gateway vs Load Balancer

An API gateway acts as a middleware component that sits between clients and backend services, providing a centralized entry point for accessing various endpoints and functionalities. Its primary function is to facilitate communication, security, and management of APIs. Here are key aspects of API gateways:

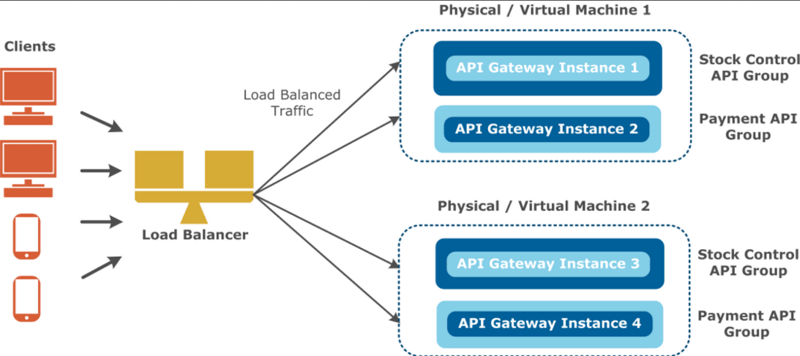
1. **API Management**  
   API gateways offer features for managing APIs, including authentication, authorization, rate limiting, and caching. They serve as a control point for enforcing security policies and access control measures.
2. **Protocol Transformation**  
   API gateways can handle protocol translation, allowing clients to communicate with backend services using different protocols or message formats. This capability enhances interoperability in heterogeneous environments.
3. **Routing and Versioning**  
   With an API gateway, requests can be routed to the appropriate backend service based on predefined rules and configurations.

Additionally, versioning support enables the coexistence of multiple API versions, ensuring backward compatibility and smooth migrations.

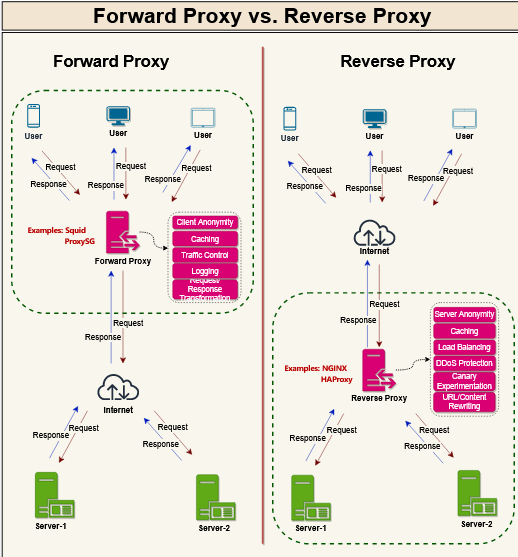
1. **Analytics and Monitoring** API gateways provide insights into API usage, performance metrics, and error tracking. This visibility enables operators to monitor the health of APIs, identify bottlenecks, and optimize resource utilization.



A load balancer acts as a traffic distributor, evenly distributing incoming requests across multiple backend servers or instances to optimize resource utilization, improve availability, and enhance performance.



# Forward Proxy (Client side) vs Reverse Proxy (Server side)



# Nginx

Extracted from [here](https://www.youtube.com/watch?v=iInUBOVeBCc&list=WL&index=43&t=137s)

It can act as:

* Web server
* Proxy server (proxy means something that sits in between)

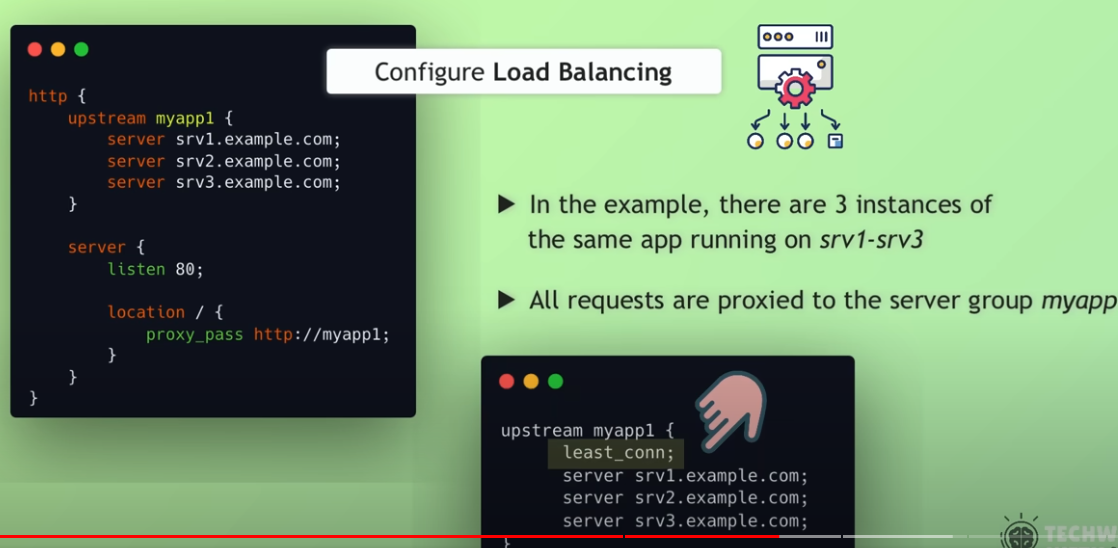
Proxy server can have many properties:

* Load balancing: (means to balance the load across all instances of a web server)
* Caching
* One entrypoint (security)
  + Encrypted communication
* Compression (mostly used with video streaming)
  + Segmentation (sending video in chunks)

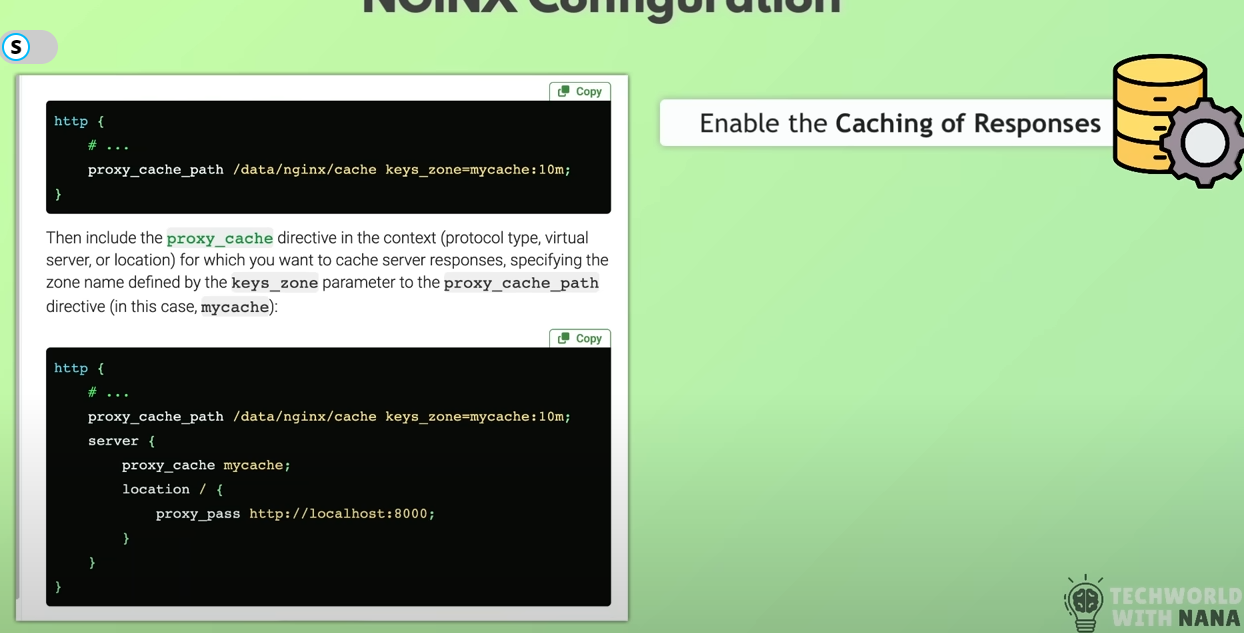
The main file is nginx.config

Here you can create the code comprising of directives and blocks, and configs whether it’s a web server or proxy server.



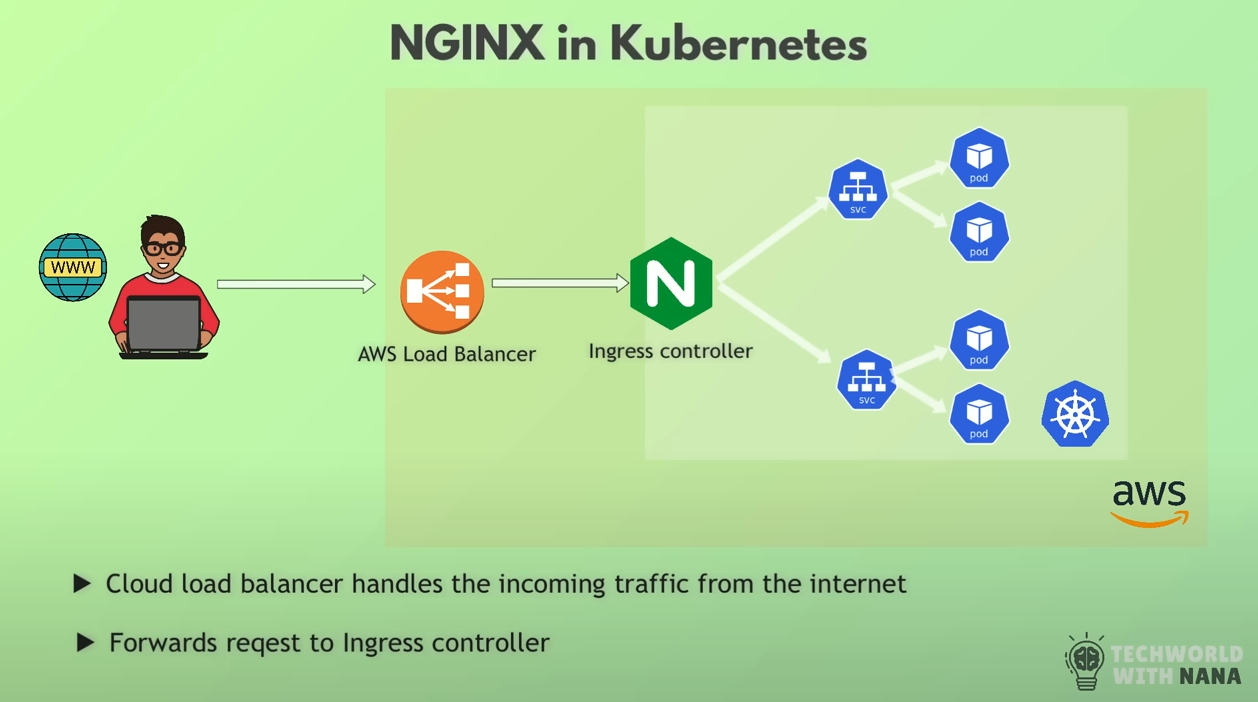


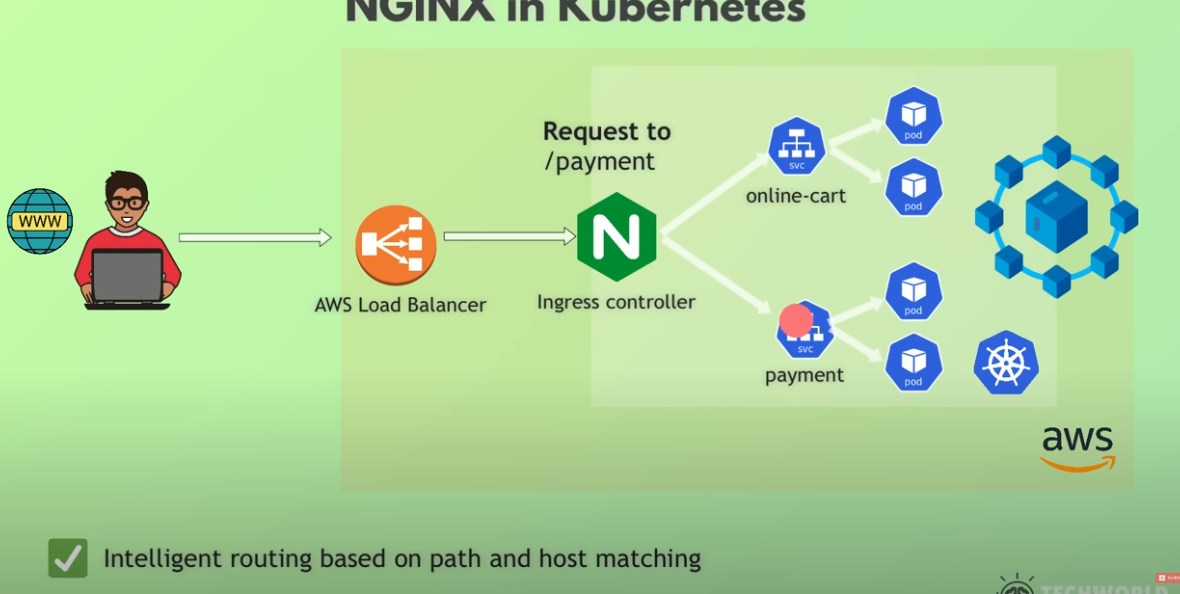
Here least\_conn is the load balancing algorithm (default is round-robin)



## Nginx for kubernetes

Nginx is implemented in kubernetes via K8s Ingress Controller. Ingress is a specialized load balancer for managing ingress traffic in Kubernetes. Kubernetes is same as EC2 but for docker images, instead of booting and configging the OS it directly run the images.





Microservice based architecture.