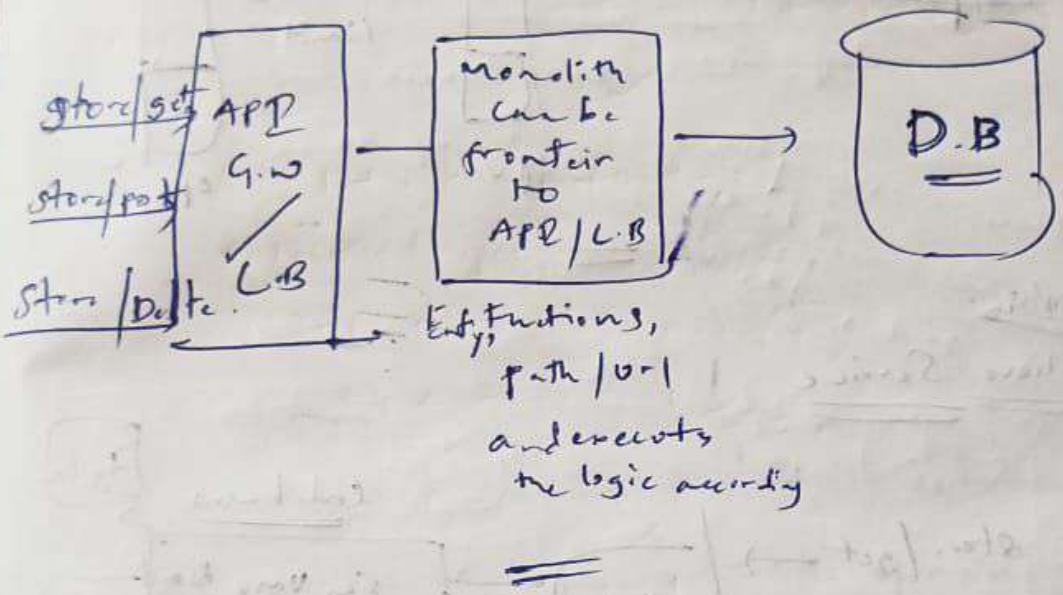
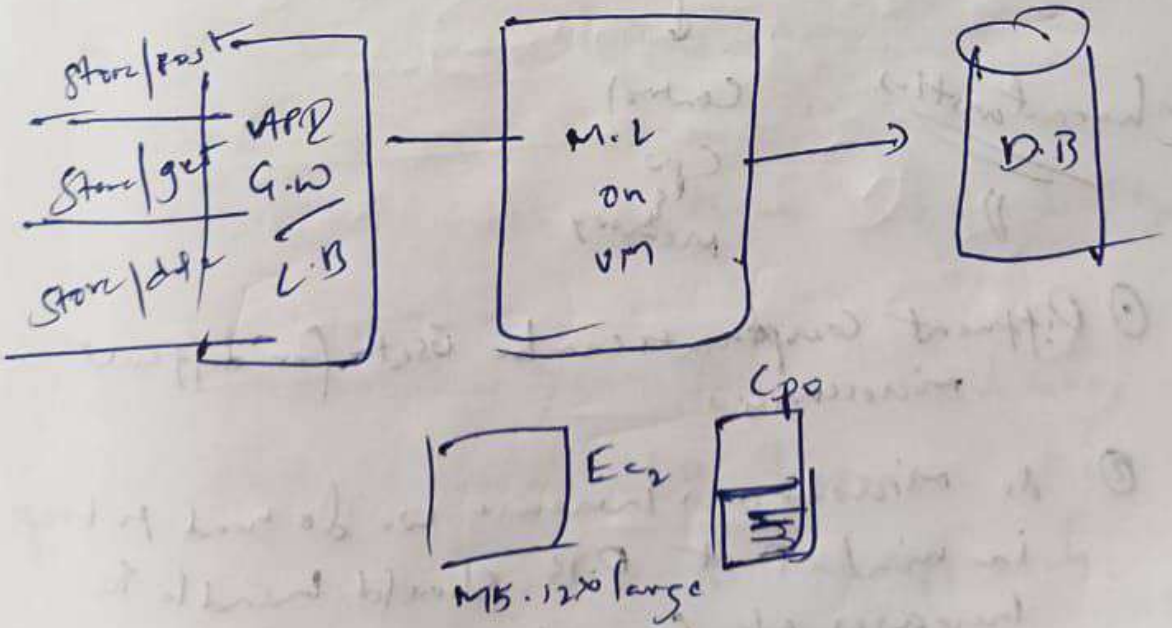
Can you use API with Monolith ?

Absolutely, yes

API is have nothing to deal with monolith or Microservices



Scaling of Monolith : Issues



Generally there are lots of techniques to optimize the reading from the DB like replica catching etc.

(7) Another advantage is as Microservices have their own backends and are independent with each other these can be controlled by different teams

Python
NodeJS
Go

which is so polyglot

Advantages / Characteristics of Microservices

1) The core characteristic of Microservices they can be, Microservices is independent to each other and they can scale irrespective to each other.

(7) You can have different Governance, Security guidelines and they can be deployed each of them independently.

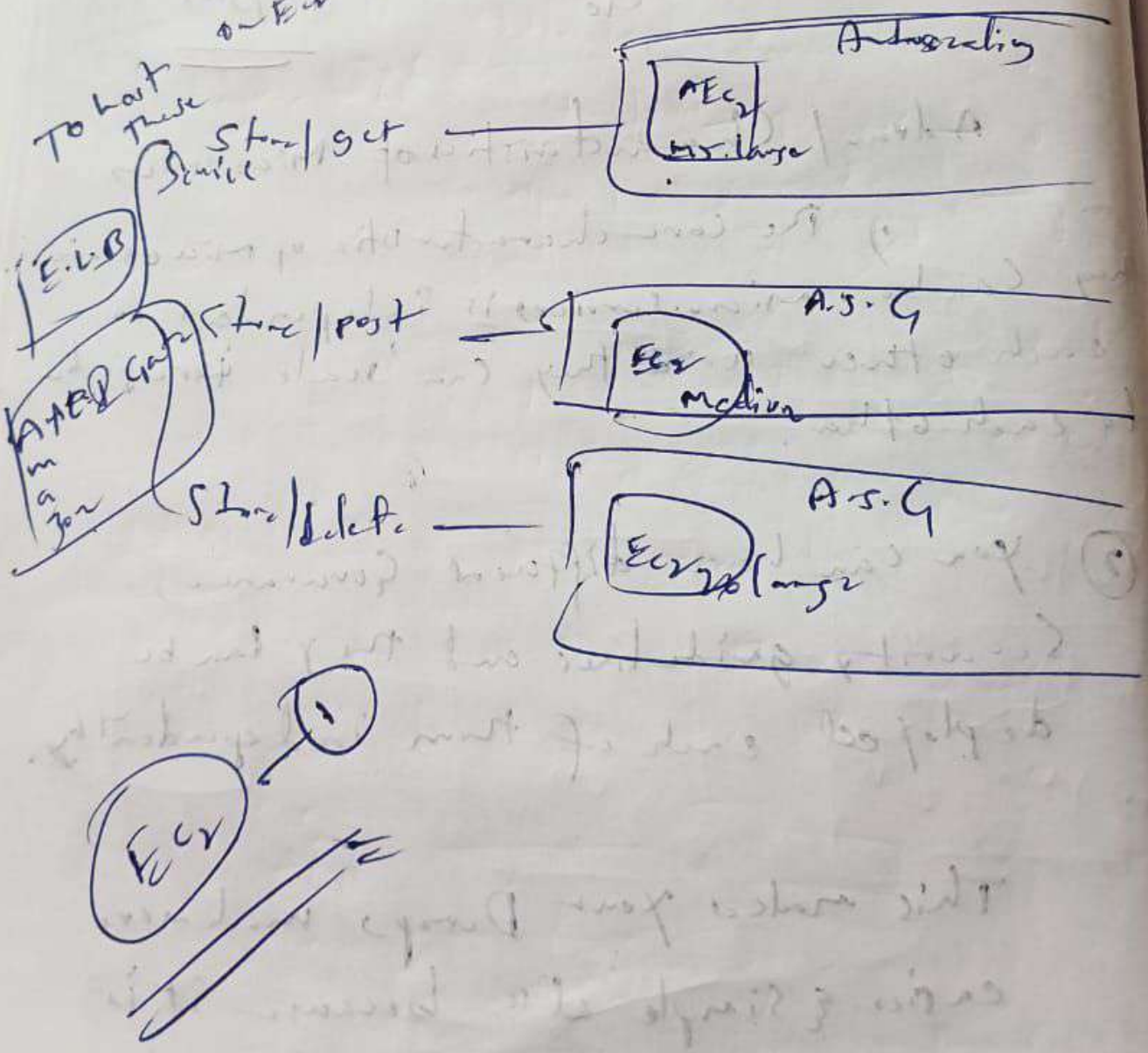
This makes your Devops much more easier & simple also because it is no longer all or nothing approach.

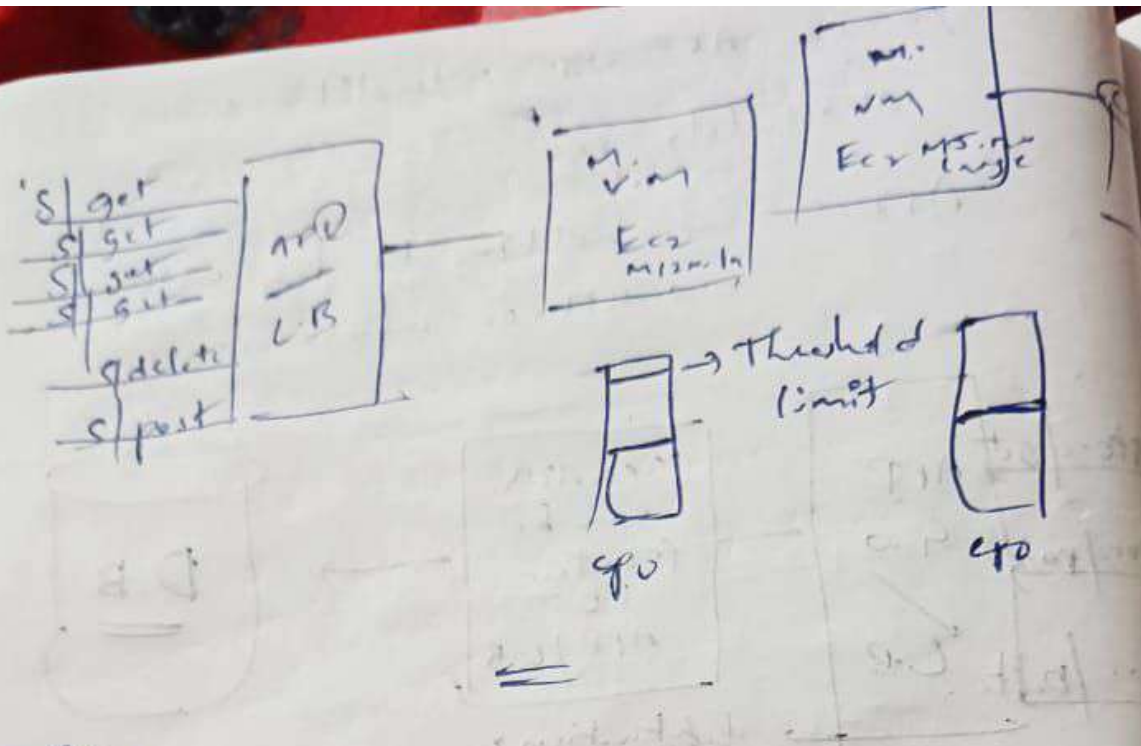
You can test this MS independent to each other and they should have different functionalities.

Deploying Micro Services In Aws :-

We have one service answer to Every thing Ec2

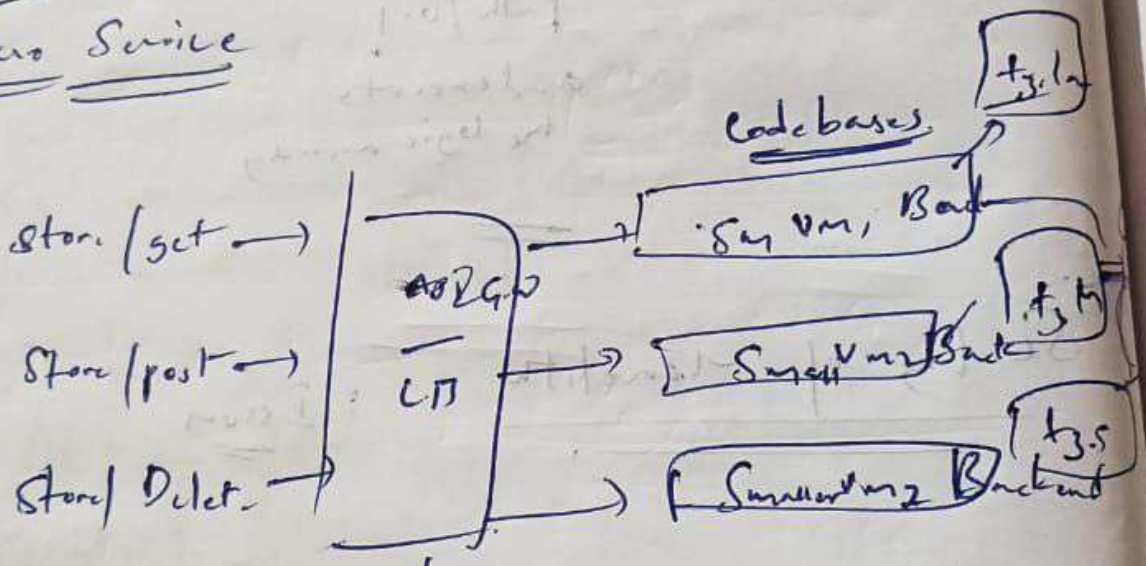
There is misconception MS cannot be run on Ec2 (This wrong)





APIGW

Micro Service



Characteristics

Control Cpu & memory

① Different components can be used for different microservices

② As Microservices increase we do need to keep in mind that D.B should handle the increases of connections.

kubernetes part 6

pod: Smallest deployable unit

1) Elastic L.B
2) Amazon API gateway
3) Application Load Balancer
4) Ingress ALB.

A. Elastic Kubernetes Service

Ingress ALB
- store/cart → Service A → Pod
- store/post → Service B → pod
- store/order → Service C → pod

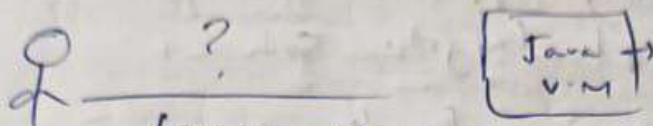
Since it independent with each other & written different programming language

we can mix & match all these services

ALB
- S/S
- S/P
- S/get

Container

Load Balancer



How you urge / access EC2 or invoke this application

Ways

1) Each Amazon EC2 has IP address. So you can directly invoke this IP address to invoke the application

2) But there are some caveats to it?

① What if this EC2 goes down and another EC2 comes up with different IP address

② How will you dicover the New IP address

③ What if Application Scales and now instead of one EC2 you have 3 Any EC2 what if even more EC2's keep getting added

④ How will you discover these EC2's as well as invoke the application and ensure that traffic is balanced b/w these EC2's

That's where load Balancer comes to the picture

For modern API Services we have for modern application development we can use different AWS Lambda for each microservices backend

Remember Lambda Scales
autoscaling group
or
automatically.

And Lambda can be fronted by E.L.B
or Amazon API Gateway

ELB → Store/get → AWS Lambda ——— (1)

Store/post → AWS Lambda

API Gateway → Store/Update → AWS Lambda

Lambda scales automatically.

Another choice is Containers ——— (2)
Popular choice

ELB → Store/get → Am Elastic kubernetes Service

Amazon API Gateway → Store/post → Amazon Elastic Container Service.

Store/delete

LB with kubernetes pods

Similarly kubernetes pods load Balancer can distribute traffic across multiple pods and in this case

LB acts as ingress / service.

LB with Lambda

Similary even if your application is running in lambda load balancer can distribute traffic to Any lambda (those as well)

What application is exposed to website that you already have such as

Low no-store-low:

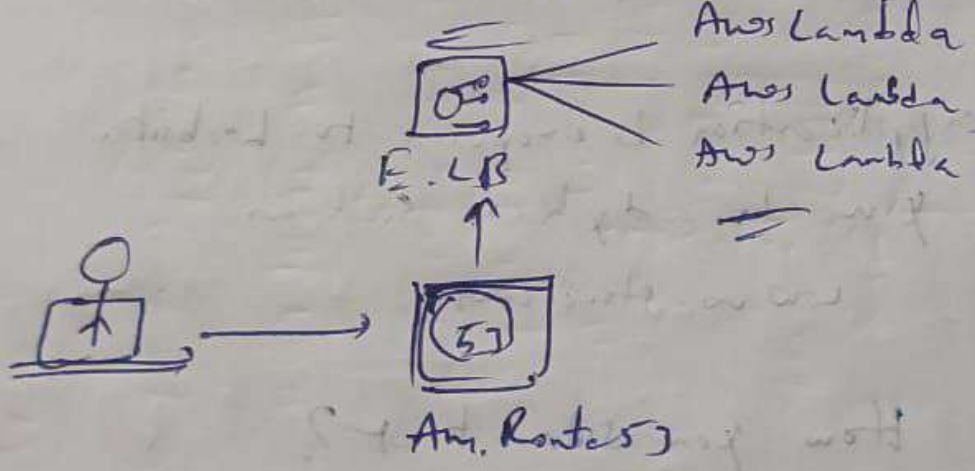
How you solve that?

E.L.B gives you DNS. Will you can use

ELB with a DNS services such as Amazon Route53.

Whenever user invokes www.store.com
The traffic first goes to Amazon Route 53
and from it then ↓

User type URL
&
DNS Resolution starts
↓
Resolution queries authorize by DNS
↓
DNS return ELB point
↓
Browser send HTTP Request to
↓ ELB
ELB routes traffic to backend
Servers



Load Balancer automatically distributes incoming application traffic across multiple targets, such as

- EC2's
- Containers
- IP addresses etc

Each Elastic L.B has a unique DNS name so that your user can invoke this application using these DNS world.

If the traffic goes up and your backend needs to scale up and let's say it adds another EC2.

L.B automatically discover this new EC2 to Instance and start distributing traffic to it.

It also tracks the health of all EC2 instances.

Even if one EC2 goes down and it will start distributing traffic to the rest of the healthy backend targets.

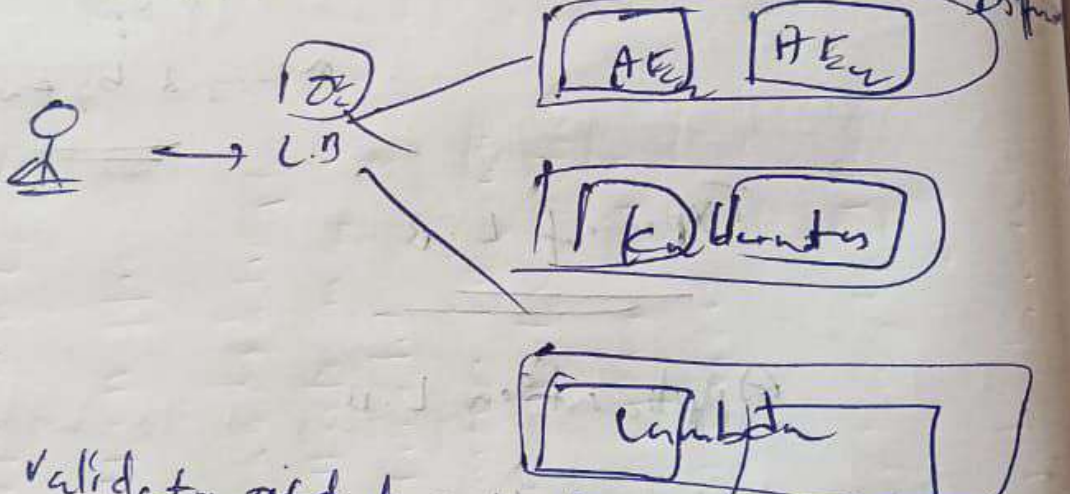
We can distribute the traffic b/w backends with paths to the target groups.

Properties of L.B

① Operates on OSI Layer 7

② Route traffic based on url path

③ For different url, the backend doesn't need to be same. Ec2 or Lambda



④ Validate and terminates SSL

Network L.B

- Operates on OSI Layer 4
- Route traffic based on protocol and port of incoming traffic

N.L.B cannot send traffic to backend based on different URL because URL path is Application layer property

- SSL passthrough by default

N.L.B can pass through & terminate as well

ALB or NLB ? What to use

1. N/LB handles spiky traffic better
2. NLB exposes static Ip address
3. • ALB need Global accelerator
   Influenced by choices we have
   - API Gateway Rest API integration with backend like ECS / Kubernetes etc
   Integrate with private link
   ingress
   - N/B Support EC2 instance and Ip address in Backend target groups. It does not support lambda
   ALB Support Lambda as B.E
   me couple of steps but
   LB/NLB supported as B.E

Properties of Load Balancer

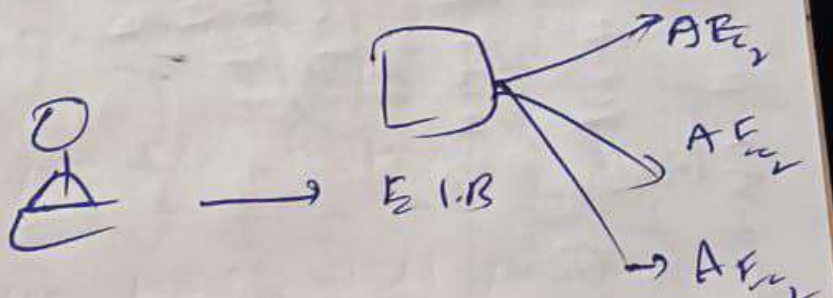
- Automatically distribute incoming traffic across multiple targets
- Monitors health of targets
- Integrates with CSL
- Elastic - traffic goes up LB also will also scale up.
  Managed by AWS

Types of L.B

Application L.B

Network L.B

ALB :



Let's say on instance website www.store.com. I want different instances and backend for different operation.

Let www.store.com/get/post/delete like this.

Target type — Application level — Http status code — N/w and Tcp connection

Health checks — Application level (Http status code) — N/w level Tcp connection

Simple Telugu;

ALB = Web master
Browser lo URL open chesthey, HTTP/HTTPS request ni read chesi, path or header batti correct servers ki pampistadi

NLB: Highway traffic cop NLB ki content thoramu ledu, yevarraina, connect ithey fast ga tcp level lo forward chestadi

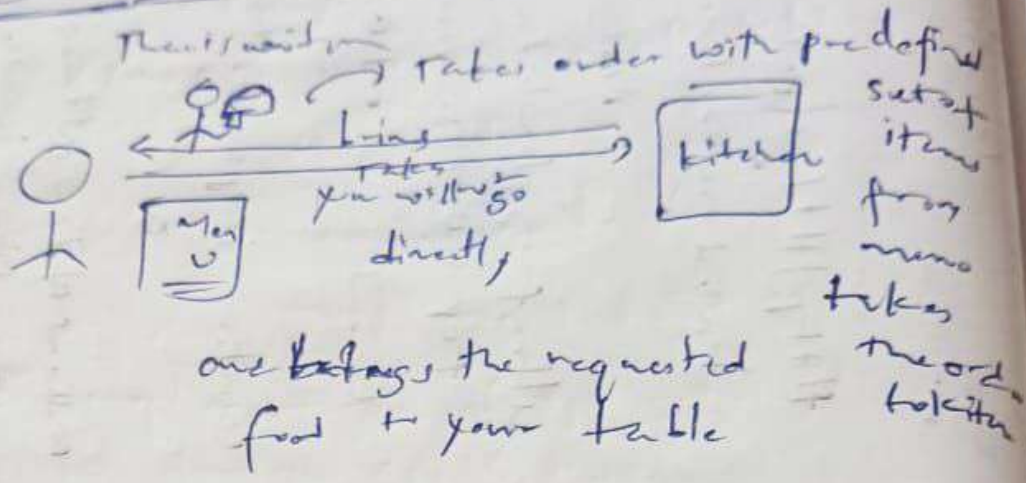
Http/Https/path based routing karali → ALB

Ultrafast, low latency tcp & UDP service karali NLB

Content inspection karali ALB

Millions of Low latency connection, handle cheyyali (gaming, real time ity) NLB

API & why do we need them?



Here waiter acts as a interface to the you & kitchen. So waiter is API in this case

Backend → Bank / airline booking / tinder etc → with AWS EC2

waiter → API

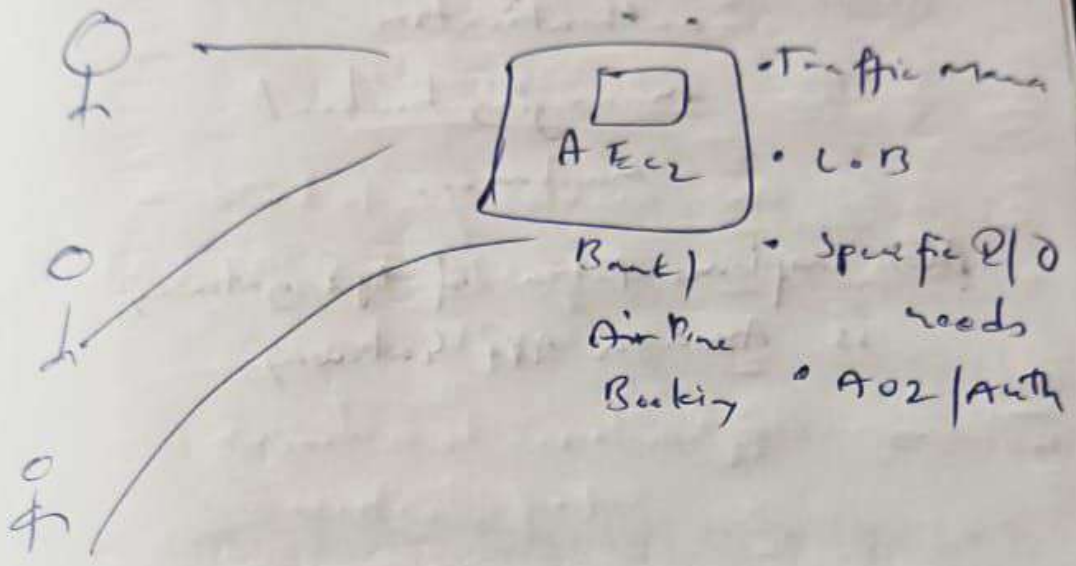
Menu → with predefined input fields

What is the use of API? How do you skip API and directly Backend in realtime?

In Real world there are multiple users using apps Sometimes millions

	ALB	NLB
Layer	Layer 7 Application Layer	Layer 4 Transport layer
Traffic type	HTTP/HTTPS traffic	TCP/UDP/TLS level traffic
Routing Logic	Content based Routing (path, Hostname, Headers etc)	Fastest connection based routing, No deep request inception
Latency	Bit more (Because it checks content)	Ultra low latency (Millisecond level)
Use cases	For web apps, APIs & microservices	For high performance apps, real time TCP/UDP apps
SSL Termination	Ecs, Ecs, Lambda	Ecs and Ip address, Private links
Target type	Application level HTTP Status code	N/W level TCP connection

Beyond just invoking the program with some input parameter and getting output, there are a lot of other factors such as



If you want code all things in application program there are lot of overload because

① all this interface and cares have nothing to do with business logic that's running your application

② So coding all this into your application makes it difficult to test, debug, modify and deploy your code.

③ And inversely that a lot of open interfacing standards that you need to implement and implementing all these with their application code makes it very cumbersome and painful.

API Gateway

1) Can implement rate limiting, bursting for API's

2) Not possible to get static IP address for endpoint

3) Accepts HTTPS traffic

4) It is able to Request validation Request / response mapping

5) Able to handle Spiky traffic (default Rate limit 5k burst rate) Both traffic become superfast rate per second

Both integrate with AWS Web application Firewall (WAF)

Both can handle Spiky traffic

ALB:

No rate limiting, bursting capability

possible to get static IP address for LB endpoint (by using Global accelerator)

Both Accepts HTTP & HTTPS traffic

Not able to do Request validate Request / Response mapping.

5) Able to handle but some delay But we can pre-allocate LCU's to avoid delay.

① Able to integrate with Lambda from different regions with different accounts
Not Both would in diff account we would able to connect

② We can able to import/export Open API from ex openai, Swagger etc

③ Have extensive AD/A2 Integration - API key, IAM, Cognito user pool etc

④ Able to Cache Responses

⑤ Timeout limit 30 seconds

⑥ Integrates almost all AWS services

c ALB is a regional service

① no direct method to import/export for cross platform

② Integrates with any OIDC compliant IDP Cognito etc.

④ Not able to cache responses

⑤ Timeout limit 4000 seconds

⑥ Use Ecs lambda, IP address as backend.

Application Load Balancer vs API Gateway

Payara 3/3/25

ALB (App LB)

Automatically distributes incoming traffic across backend targets

Layer 7 load balancer

Infrastructure managed by AWS

highly available elastic

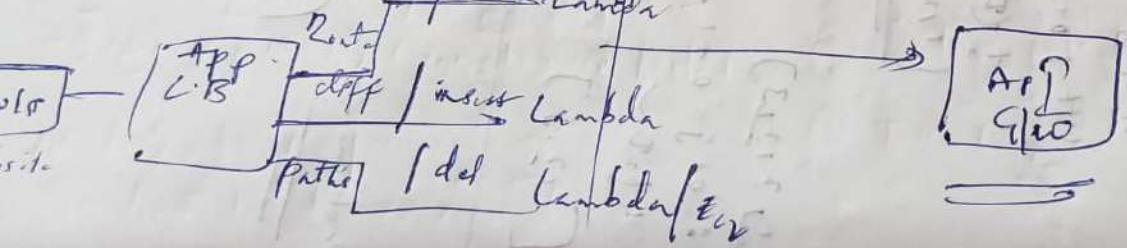
when traffic goes up, the LB needs to scale up as well. So to account etc the increase of rate of traffic

API (API G/W)

Fully managed and serverless API Service from AWS

Automatically scales up and down

Same Option



API Gateway → Both integrate ALB