**API GATEWAY:**

An API Gateway is an API management tool that sits between a client and collection of backend services.

To overcome the Difficulties for clients and to access the information they need in a fast and secure fashion, API gateway shall be used.

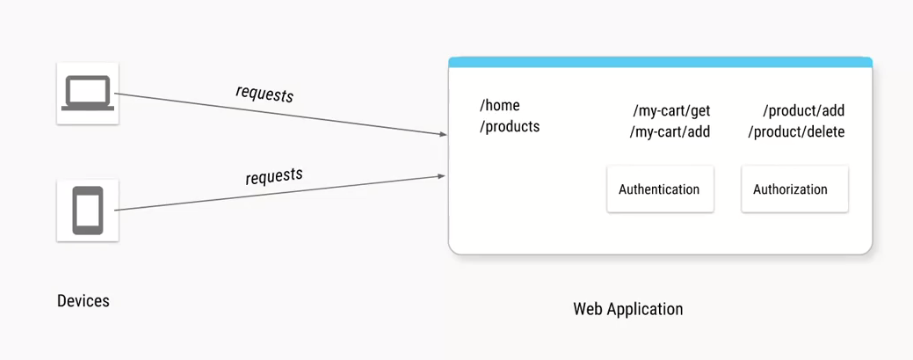
**Advantages:**

It provides optimal API for each client.

Reduce the number of requests.

Reduce Load on microservices

Let’s see how API Gateway enhance application performance



**Example:**

If we create a e-commerce web application, asusual, request will be pass through multiple device(cell/lap etc).

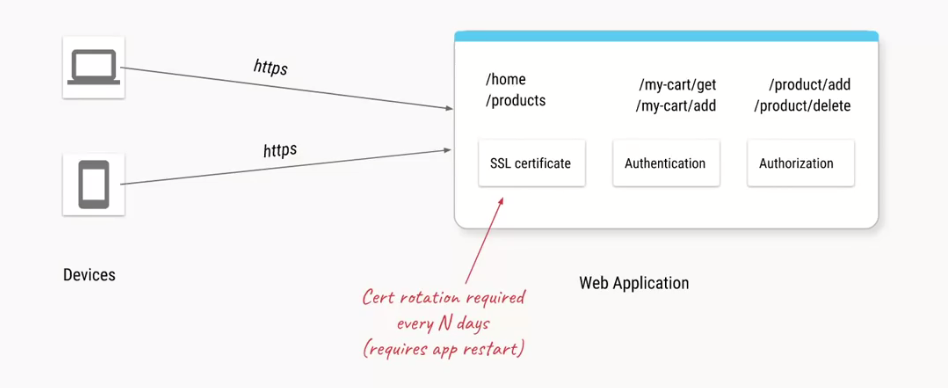
So result is if it’s a static page (/home, /products)then not must of authentication or authorization, it will fetch and shows the data.

If it’s some pages which need user credentials(/my-cart/get , /my-cart/add), or Authorizaion of Admin user(/product/add ,/product/delete)

So without API Gateway the process will be like this..

Lets see how API Gateway enchancing performance of an application, and the list of API Gateway’s feature

**Without API Gateway:**



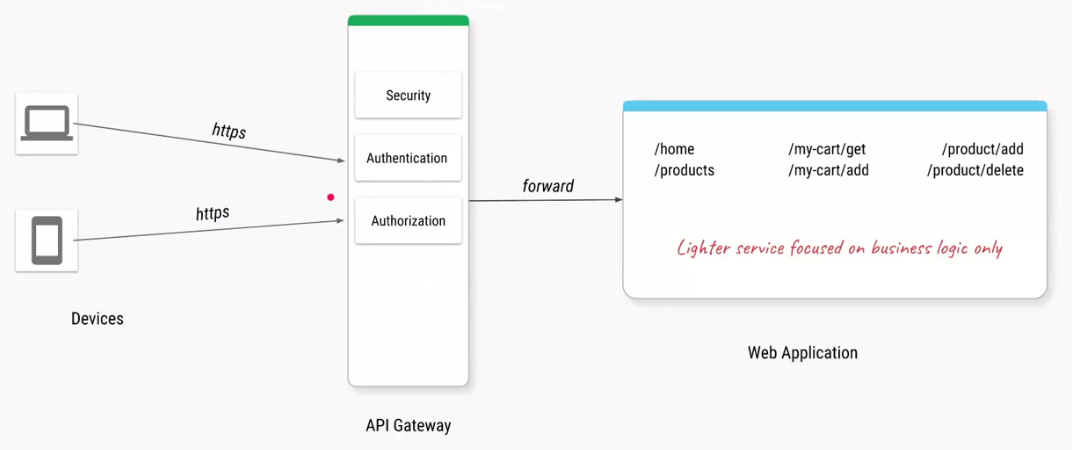
Basically our web application have 3 added components(SSL,Authentication,Authorization) in addition to all the business logics.

For Security we will use https, so for that we need SSL certificate(for best practise we should change every 30 or 60 days)

**We want to try to separate out these things which are not business logics related into a separate component. That component is called API Gateway. Reason for the separation Is to enhance the performance of the application, and in upcoming we can see why we are segregating? What can we segregate? And Feature of API Gateway.**

**With API Gateway:**

API Gateway is the component which acts as an entry point for our API’S. It reduce the number of requests



So every time the client make request, it will first goes to the API Gateway, the API Gateway wil ensure the https(Security) using SSL certificate, ensure the user is Authenticated(Authentication)and ensure the user has right role if the URL requested for a product change(product/add)

If the above 3 condition is satisfy then that request forward to web application.

**So we can say as API Gateway will act as a guard for any API Request which is coming to our application**

**So it will protect our API from bad actors(malfunction)**

These are the **1st feature** of API Gateway.

**Feature 1** – **Separate out cross cutting concerns**

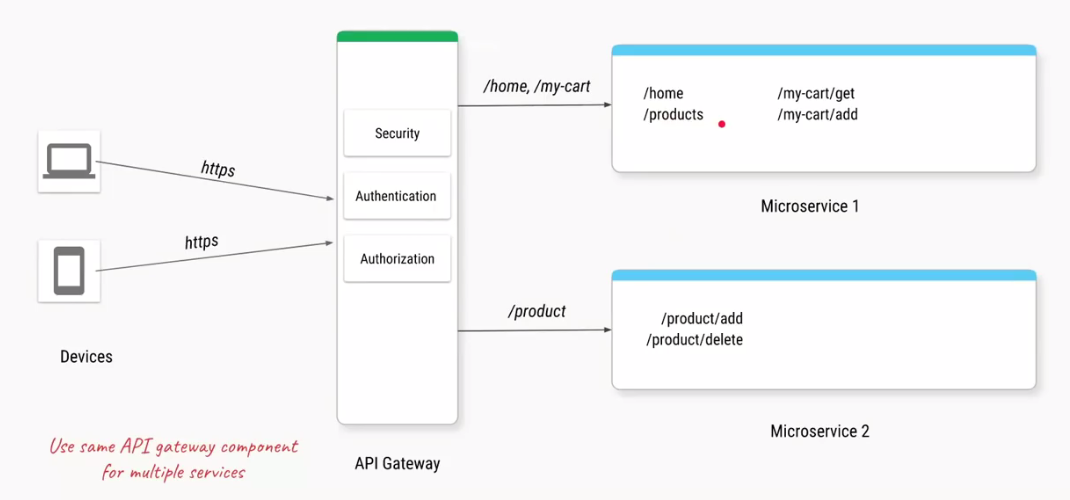
\* Authentication

\* Authorization

\* SSL Termination

\* DDoS protection / Throttling

Okay for performance we are converting our monolithic application to some microservices(simple POC available for Microservices).



In the above pic we can see as,

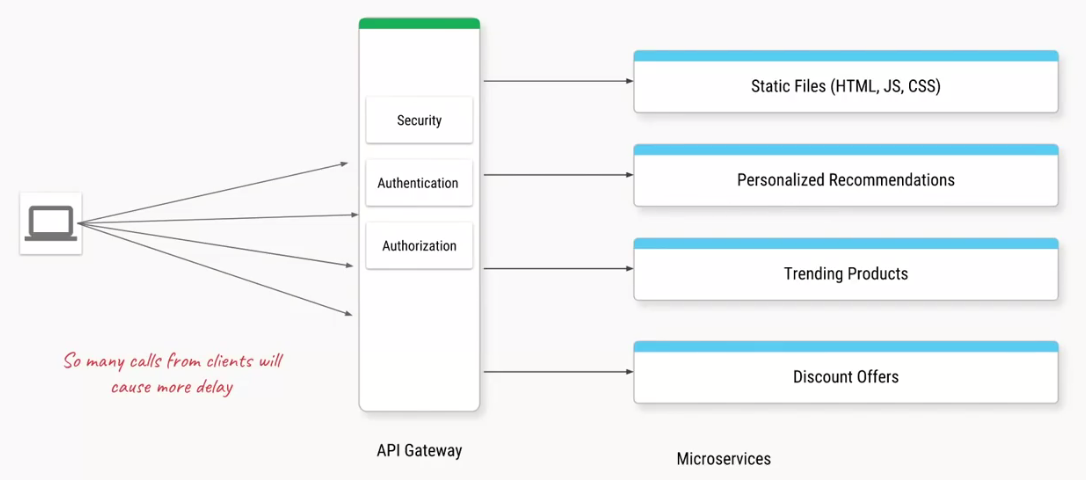
If there are any URL hits from the clients from /home (or) /my-cart then it should goes to Microservice 1,

Incase if it is /product it will redirect to Microservice 2.

**Feature 2** **- Separate and consolidating cross cutting concerns across microservices**

Day by day passing client may ask new requirement to add new features.

Ex, Personalized Recommendation for Logged in user, Trending products and Discount offers.

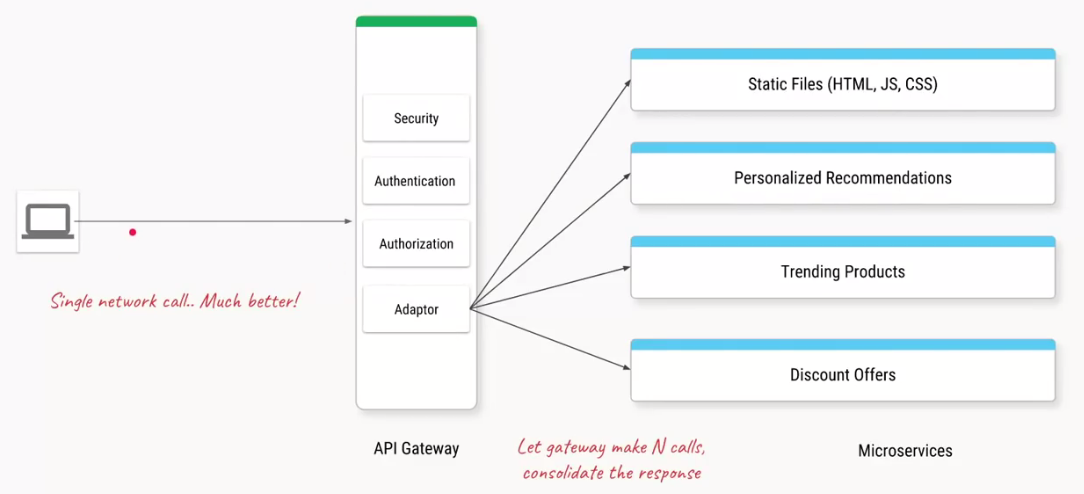


Okay so what? Just Imagine all existing and additional features are provided by single microservices.

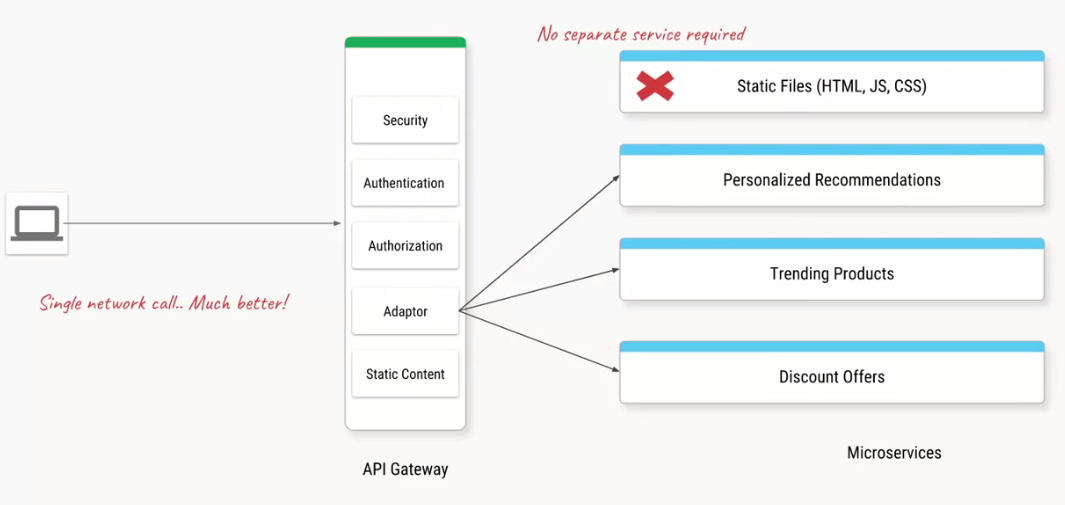
In above image we can see 4 http call to ensure home page rendered correctly. If it goes on then the service list will be increase and http call will increase to render single home page.

So its better, to pass only one http request and have some component in api gateway called **adapter**, which make 4 calls on behave of the client(lap/phone),it consolidate the response from the services, and send back a single response to the client showed in below image.

**So the response time to render the home page is much faster than before.**



Okay how We can simplify the above process again for better approach,



If it is a static files with no business logics, then using it in microservices is not recommended. Because it serve same file again and again so to avoid that, we have a component within the API Gateway, where if there are any request for those static files the api gateway itself return the response, so we are reducing microservice here.

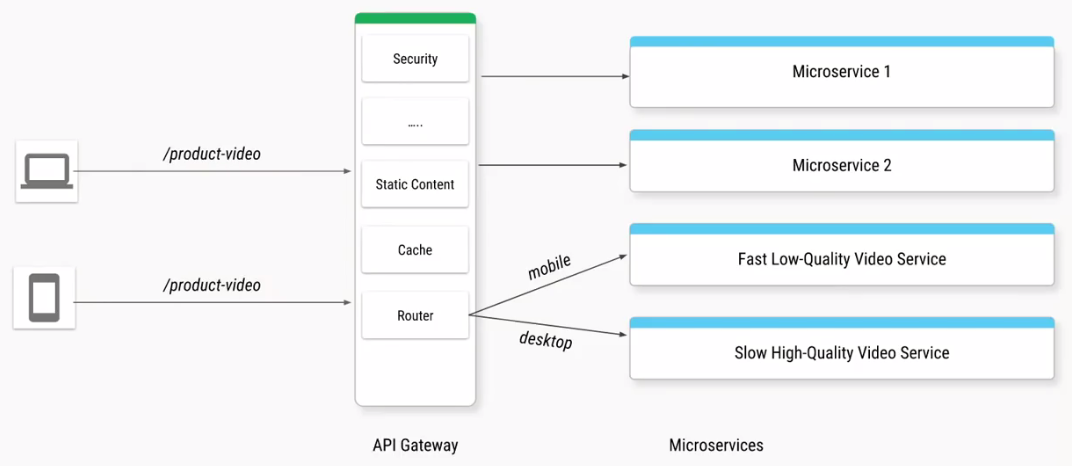
Lets see we have a service to fetch trending products, just imagine the trend product price are recalculated for every 1 hour. So if it 50 calls in one hour, it will always forwarded by the API Gateway and the same response come from microservice again and again.

API gateway has a feature called **response cache**, where u can give it to URL, and a threshold time for which it needs to cache the response. So if the first time client makes the call, it will forward that to microservice, it will get the response and cache it for one hour, and if next time client send the same request, instead of going back to the service, it will return response from cache itself.

That is,

**Feature 3** – **Replacing multiple client call with single API call also some feature of reverse proxy : Serving static content, Caching responses.**

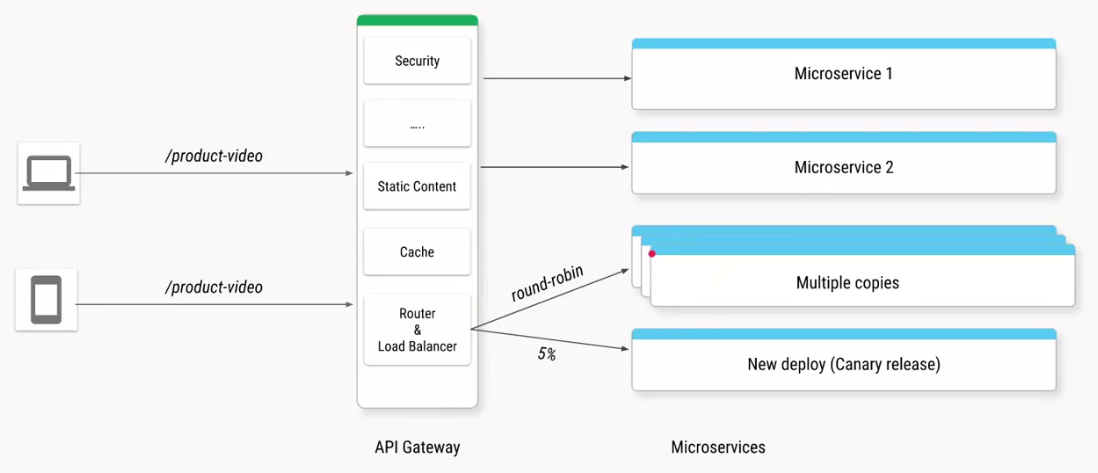
Lets see another feature, before that Imagine we have exposed a URL /product-video so will have 2 services in the backend, once is for Low quality video and another for high quality video, so in API Gateway we can configure that as Router



Lets say one of our service become more popular, so obviously it getting many API hits, to that service, and we make multiple copies of that service,

In this case API Gateway also take the responsible for the, Load balancer, where the first Api call to this service is made to the first copy and second call is made to the second copy and goes on, in a round robin fashion, and if we have a new version , of the same service deployed,

Then the Api gateway we can just configure it, so that it sends 5% of traffic to the new service while the rest of 95 percent will still go to the same old version of the service, shown below,



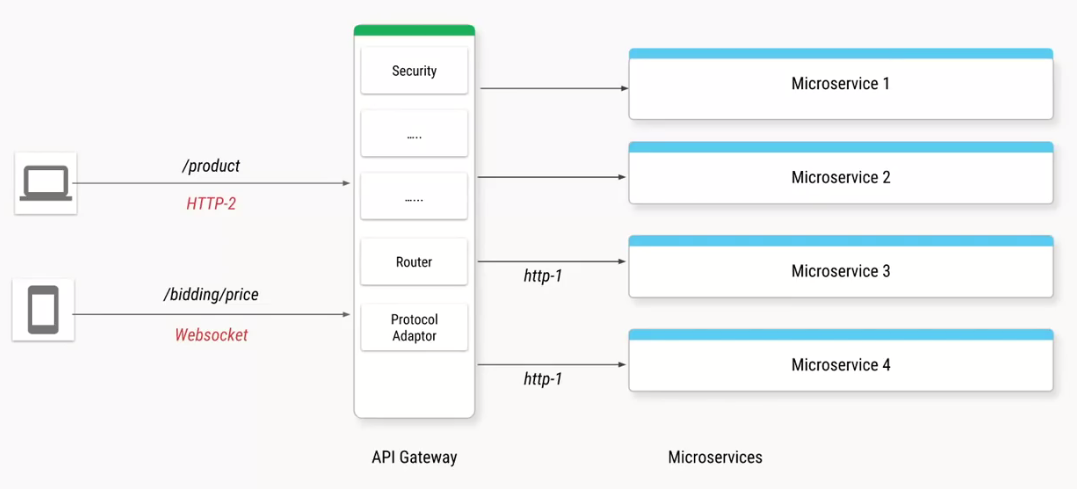
Its **Feature no 4** – **Routing to the right service,and routing to the different copies of the service, based on whather you are doing the balancing, or a canary release even A/B testing**

Canary Release – It is a software development strategy in which a new version of an API is deployed for testing purposes, and the base version remains deployed as a production release for normal operations on the same stage

A/B testing – Also known as split testing or bucket testing, and it is a method of comparing 2 versions of a webpage or app against each other to determine which one perform better

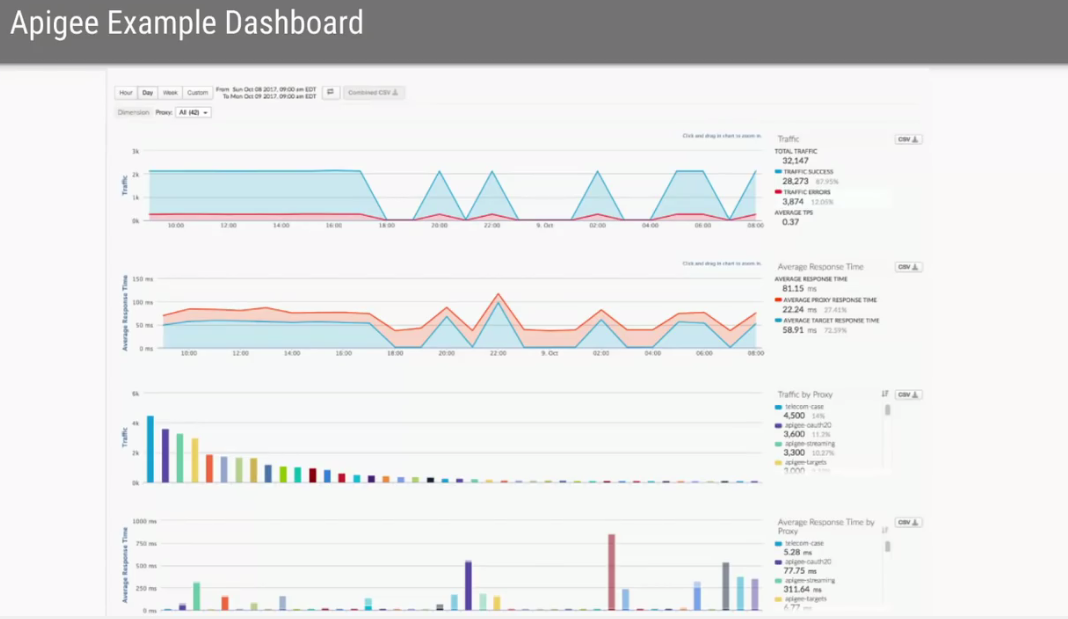
Next feature of API Gateway deals with,

**Feature no 5** - **Protocol**,



Want to take advantage of new protocols like web sockets or we want to use new version of http which is http-2 so even if our backend services is not ready, not compatible with http-2 or web socket API Gateway can take the responsibility of converting new protocol to an older protocol.

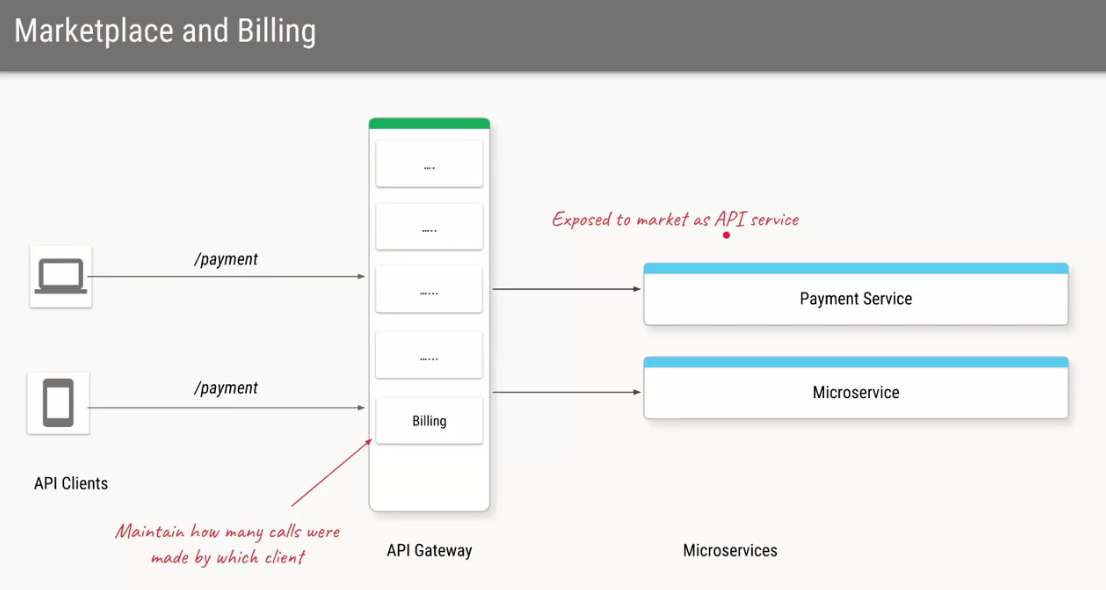
And of course if all these requests, are going through the single component API Gateway, it gives us an extra advantage, where we can monitor, lock all the requests and the response times where we can calculate the average redundancy.



And most of the API Gateway provide such dashboard, where using the charts, u can monitor how your API is performing behind the screens.ex – Apigee(Cloud service API gateway)

Lets say we develop a microservice, which is so reliable and show good, that we want to expose it, even for other clients, so we are actually creating an e-commerce app but our payment service is very good, we want to expose it as a payment gateway, so that all other websites also incorporates that.

So we want to expose the payment service as an Api service. So if we use the Api Gateway as cloud provider like AWS, we have to reach out API Service to the market place, any client can come in and start using an API service and based on how many calls, we make to a service we will charge them accordingly.



**Examples for API gateway service,**

**Self Managed (**If we are deploying our own application in virtual machines, or own data centers)

\* Apache

\* HAProxy

\* NGINX

\* Spring Cloud Gateway

**Cloud Services (**If we are deploying our application in the cloud)

\* AWS api gateway

\* Azure api gateway

\* Google cloud endpoints

\* Apigee

So overall we can say as if in an application, we are developing in Microservice architecture rather than Monolithic architecture and addition of, API Gateway it enhance the performance well.

These are the overall concept of API Gateway, based on which API Gateway service we are going to use in our Strive project, additional POC require to implement the respective gateway service in our microservice architecture.