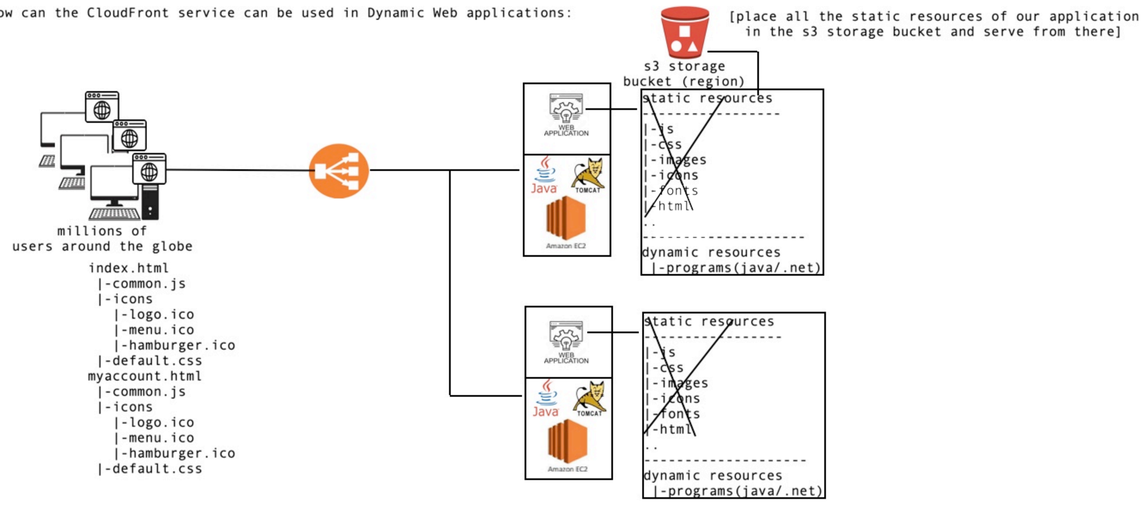
Cloud Front (service)

There are 2 types of Web Applications are there
1. static webapplication = the static webapplications are deployed and served over the static web servers, wherein for all the requests that are coming-in the static webserver returns or dispatches the same response to the clients. (These are not interactive)

2. dynamic webapplication = the dynamic web applications are deployed and served over the dynamic web servers. These applications comprises of both static/dynamic (programs) resources. these are interactive applications wherein the enduser can send the data as an input aspart of the request, so that the dynamic webserver receives the request identifies the resource.
- if it is an static resource, it dispatches to the client, without performing an operation
- if it is an dynamic resource, it executes the relevant program by passing the user data and returns the output of the program as a response back to the client.

#1. Let us understand how can the CloudFront service can be used in Dynamic Web applications:



In the above deployment architecture of an dynamic web application:
Both the static and dynamic resources are packaged into the application and is deployed, hence the dynamic webserver will be involved in serving the static resources, even though there is nothing to do for the dynamic webserver in serving the static resources which would lead us to several problems as described here:
1. across the instances of the application, the static resources are being duplicated and is loaded aspart of the dynamic web server, thus resulting in wastage of the system resource.
2. The users/webbrowser aspart of rendering an web page might parse and send requests in downloading several static resources that are referenced in the page to the web server.
(in HTML page we might be referring, css, js, images, icons, fonts etc) which will be resolved while rendering the in the web browser, so that the browser will sends requests to the webserver asking for downloading them inorder to render the web page.
so from this we can understand a request for resource to the server would potential leads to 10+x requests in downloading the static resources to the webserver this could lead to several problems
2.1 the traffic between the client and the webserver increases which could lead to network latency in serving the requests
2.2 there could be a limit on the number of connections the dynamic webserver can handle, as the dynamic webserver is serving the static resources, it might quickly reach the maximum limit, which leads to queuing up the actual requests that are supposed to be served by the dynamic web server
2.3 I/O bandwidth in serving the content will be limted and leads to poor performance in serving the web pages/content.

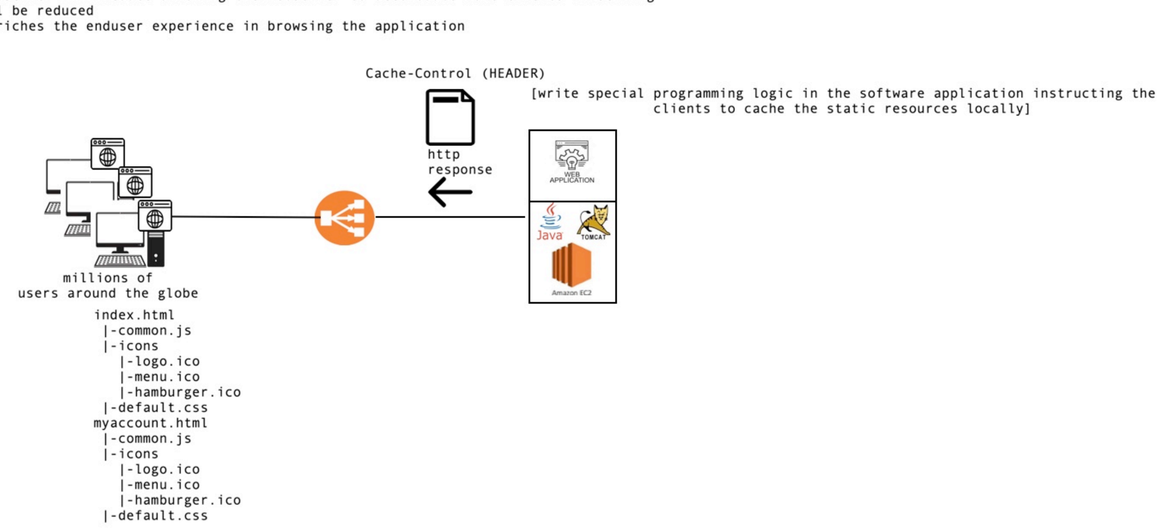
Above all would accumulate and leads to the delay in loading the web pages at the client that would hamper the enduser experience in accessing the application.

How to solve the above problem in distributing the static resources of the application to the clients/browsers?
We can use Client-Side caching mechanism or technic. The application developers aspart of their application has to write programming logic in sending special instructions called “Cache-Control” Headers asking the client/browsers to cache the responses for certain interval of time locally and reuse it for subsequent requests.

For eg.. when the browser sends the request for an home page (home.html), the server responds to the request by sending home.html page, while the browser parses and renders the page, it might send bunch of requests to the webserver asking to download the static resources like js, css, icons, fonts, images etc. upon receiving the request for these static resources, the application that is running on the webserver along with serving these resources will include an response header aspart of the response called “Cache-Control” asking the client to cache these resources.

Now the webbrowser takes the response and sees the Cache-Control header and saves these js, icons, images, css etc locally on the client-machine, so that the subsequent requests referencing these resources will be using the local cached copies of them rather than sending the request to the server/application

advantages:-
1. The traffic between the client and webserver reduces greatly thus reducing the network latency in serving the requests
2. i/o connections to the webserver will reduce allowing the webserver to accomodate more clients in serving
3. bandwidth consumption will be reduced
4. better page loads that enriches the enduser experience in browsing the application



By looking at client-side caching mechanism, we can think off the problems in serving the static has been resolved. but when it comes to web applications, there are millions of users around the globe accessing our application. per each new user, the request for accessing the static resources will be recieved by the webserver and the webserver is responsible for serving these static resource, hence we landup into the same problem again. even the intensity of the problem is less, but still the problem persist.
1. different clients are located at different locations geographically, for eg.. the webserver has been hosted in asian pacific region and the client is accessing the application from united states, so the client request has to travel all the way from US to the asia pacific region in resolving and serving the static resources that increases the network latency in accessing the webpages.
2. considerably the load on the webserver in serving the static resources are still high

How to solve these problems?
To solve this problems the people has introduced CDN proxies or intermediatories.
The CDN Proxies sits between the client and the webserver, that takes care of caching and distributing the static resources to the users across the world.