**Topology Choice**

There are a large amount of services that could accomplish the tasks that were asked, but there were a few reasons I chose the ones I did.

For the webserver, I chose to use apache. I’ve had a history of using apache, so it was quite easy for me to setup and configure. Give that we only had to serve up a static web page, there wasn’t much configuration required anyways. If there was more to be done however, apache offers an easy configuration file for adding new modules.

For the load balancer, I chose nginx. Before this I had never set up a load balancer, so after doing some research, most of my findings pointed me towards nginx. After installing it I only had to add 4 lines in the configuration file and I had a functional load balancer. Given that this was a free option and easy to configure I deemed it a valid pick.

For the caching server, I originally tried my hand with a squid caching proxy. I found thought that it was quite confusing, and wasn’t helpful it determining what was causing it not to cache. So after some trouble I decided to use my go back to nginx load and configure it as a caching server as well. It turns out it was quite easy to configure it to cache as well, and it provided a lot of different options for optimizing the performance of the server.



I chose this layout for my servers because I felt it made the most sense. Putting the load balancer in front of the web servers allows the load balancer to direct the traffic before it hits the web servers so it can balance the load. And putting the caching server in front of the load balancer allows for an even greater reduction of traffic. Assuming both web servers are serving the same content, the server will cache that content and serve it up to the clients without having to even hit the load balancer. An alternative configuration would be to put a caching server in front of each web server and have the load balancer be the first point of contact. This could be helpful if the web servers are serving different content, but if not, you’re essentially having 2 server do the same job, which will waste resources and time.

All of these technologies are extremely helpful in any scale environment. The webservers are the ones that serve your content, can’t do much without them. The load balancer helps ensure that you aren’t putting one web server under too much stress, allowing for more consistent delivery of your content. The caching server allows for faster delivery of your content. It will save some of your content that is frequently request and serve it up on your behalf, reducing the number of hops taken for the client to get its information. With all of these technologies together it ensures a fast and consistent environment.

In terms of our assignment, we are simply serving up a static web page. There’s not much that needs to be cached here, essentially the entire page would just be cached since it will rarely change. Which would result in them just becoming our web server, a job our original web server was doing. Not much to gain from that. In terms of a load balancer, it could potentially be helpful, regardless of what content you’re serving up, your server can only handle so many requests. By implementing a load balancer it will alleviate the amount of requests one server will get. Which is helpful in any size environment, with any kind of content.