Plagiarism Scan Report

Summary	
Report Genrated Date	04 Apr, 2018
Plagiarism Status	**100% Unique**
Total Words	747
Total Characters	4795
Any Ignore Url Used	

Content Checked For Plagiarism:

Heard of Docker?

Before understanding what Docker is, it is important for one to understand briefly the concept of Containerization.
Suppose if you're a student and you want to run your application, that you built on your system, on your professor's system. This involves packaging your application with the supporting OS, dependencies, plugins, libraries, executable files, configuration files and what not, to ensure proper functioning of your application on somebody else's system.
Wonder if there exists an easier way to deploy your applications without having to work this much?! Well, basically this is what Containerization helps us in.
According to the Wikipedia definition, "Operating-System-Level Virtualisation or Containerization refers to an Operating System feature in which the kernel allows the existence of multiple isolated user-space instances.These instances are Containers that may look like real computers from the point of view of programs running in them. However, a computer program running on an ordinary person's computer's Operating System is actually different from programs running inside a container."

Could understand this Wikipedia definition at once? If yes, then I must say you are simply a genius. And if not, don't worry. In the next few paragraphs, I'll try break this definition.
Usually the kernel (where the OS runs) allows association with a single user-space instance(where user processes run). This means all the user-processes run in the same area, accessing all the resources of the system and accessing the system OS. But we may want different user-space instances to isolate our applications and dependencies into self-contained units that can run anywhere.
In order to do so, one way is to have various user-space instances each having its own separate kernel and the associated libraries. Then we'll also need a hypersupervisor that act as a communication bridge between the separte kernels and the host machine's kernel. Since the units contain their own separate kernels, it would make the units heavy. So, we have another method to create lightweight multiple user-space instances. Instead of having to box each unit with its libraries and kernel, we make the kernel of the host machine available for all the individual units. This will not only make these units(or containers) light weighted but also will use fewer resources and hence their creation would be much faster and simpler.
Containers include only runtime components, necessary to run the desired application. One can think of a program running on OS as a program running inside a box having "see through" walls. Such a program can see all resources of that computer. Now imagine the

programs running inside a box(container) having “opaque walls”. These programs have the capacity to just see and use the container’s contents and devices assigned to it and believes them to be ALL that is available. The “opaque walls” do not allow programs to access or see other resources of that computer.
The former box having “see-through” walls can be considered as a computer program running on an ordinary person’s computer’s Operating System. And the programs running inside the container i.e box having “opaque walls”consider the container like a mini computer inside a computer.

Now try reading the Wikipedia definition again!
Docker is the most common application containerization technology. It has redefined the standards of Containerization. Released in 2014, it gained popularity due to its ease of use. It has made it easier to create, deploy and execute applications using containers. You can dockerize you application and launch it within minutes! People majorly used it for deployment, but now-a-days its use for development purposes has increased.
Docker not only makes it easier to quickly build and test portable applications, but also to scale them. You can break you applications’ functionality into individual containers like for example you can have the front-end of your application in one container and the back-end in one. Docker easily links up the containers to create your application. Now if the traffic increases and you want to scale it, you can easily replicate the front-end containers multiple times without having to replicate the back-end container.
There are times when you have to work on multiple versions of the same application(like when building new application on the latest version but maintaining an older one on the previous version). But who wants to install 2–3 versions of same application. Docker is for your rescue! Docker lets you run applications without having to actually install them.
You can even run Google Chrome inside a container! Yes, Docker captures the most basic uses too.