Question 1:

*"Describe how docker can be used to solve the increasing interoperability issues associated with a growing number of software products and versions used in a cloud services. Give specific examples comparing Docker to Traditional Virtual Machines. "*

**Word Count: 841**

Since it has become difficult to manage the storage of software in hardware that is always evolving, cloud services have become highly favourable for software management. As software is in the form of code and many APIs, the amount of software that sits inside a cloud is the real issue. Software has dependencies or interdependencies whereby the software cannot work if any of the component is missing. Even a little modification can break the software. A good solution to this is isolating pieces of software from each other. Docker is one method of doing this. Docker is basically a ‘shipping container’ for source code. It is used to encapsulate different pieces of software in a container, thereby separating software of an application but it also allows these different pieces of software to interact with each other. This is a great method for application deployment and any modifications to the software doesn’t break the application at all.

Docker is composed of an Engine which is a portable runtime environment that runs on top of Linux. It also has a Hub which is a cloud based service and defines the images and containers. Mainly a container is firstly defined, pushed to the Hub and then it is pulled down elsewhere. The containers are then run. Decoupling is known as the way of building small lightweight containers and holding different parts of software of an application. The pieces of application software are isolated in many different containers, and hence are lightweight and can easily be pulled down and assembled very quickly. This helps to eliminate inconsistency between different environments e.g. development environment, production environment, testing environment etc. Docker is also advantageous in making the lifecycle more efficient and repeatable. Since the application is now isolated into containers, the shipping weight may only be in megabytes or kilobytes in comparison to gigabytes. The application now is low cost and low weight and can be shipped to different destinations.

When Docker is compared to traditional virtual machines, Docker is the true winner. Virtual machines have a server which runs the operating system. If you want to create multiple copies of an application running on a server, you need to fully replicate the OS and all the libraries. This will create a full instance of the running system and all in all, it is similar to creating a full-fledged PC. Docker on the other hand shares the libraries of an application with other containers wherever they may be. Since the containers are lightweight and can instances run inside the containers, the size will obviously be the different factor when comparing the multiple copies produced by a virtual machine and the containers that allow instances to run inside them. Virtual machines needs a full stack to run anything. Docker only needs the different containers interacting with each other to do this. With virtual machines deployment size becomes an issue with the applications reaching the size of 10 gigabytes in comparison to 10 megabytes when using Docker.

Containers share the operating system and hence are smaller than virtual machines. Docker only uses a single server with hundreds of containers that can run a device. When comparing this with virtual machines, the virtual machines need at least 3 or 4 full stacks to make this happen. Docker also has memory and when using it, the start time doesn’t take long in comparison to virtual machines. It starts up quickly and operates and hence is efficient when it comes to time. It can share the binaries we create with multiple instances of the same container. This in turn makes multiple copies for the container and the user. Virtual machine needs to make copies of virtual machines. Virtual machines are composed of an operating system which sits on top of binaries and which in turn is on top of the application. When modifications are made, this will require a full replication of the whole stack and the copies need to be movies in order for them to even work. Docker on the other hand shares the libraries on top of the application. If we want to create another instance of the application, we can simply copy the application only instead of the whole stack like the virtual machines.

If Docker makes modifications, the difference between the first version and the next version of the application is available and hence it does not require a full on replication like a virtual machine. Docker helps us to get away from rigid stacks that are interdependent. Modifying even a small amount of software most likely causes a ripple effect and the many different software on different platforms are affected. In order to fix these, these software individual need to be called back and the modifications implemented and then shipped out. Docker allows us to make changes and does not break the software. Docker is the ultimate solution when it comes to shipping a completed application as low weight and low cost. It also is the best way to run applications rather than using virtual machines.