Cloud Security: Containers

Before Docker and containers were used, you may have a team of developers all working on the same application where they would each need to install different services and software on their systems. They could each be running different operating systems, which would lead to different methods of installation as well as different interfaces for each. Each type of machine, might require different procedures of installation in order to set everything up directly on each machine. With all these different steps and configurations, there are many steps where something could go wrong. The same idea applies to cloud environments where different servers and machines with different operating systems or configurations would need to communicate and work with each other. Each installation and setup with multiple steps involved, have a very high chance that could lead to errors and inconsistencies, causing a risk to up-time as well as make it difficult to patch for security vulnerabilities.

With containers, the problem(s) are solved with software packaged with its own configurations and settings needed in their own isolated environments, with just one command to install. This makes the setup much easier and more efficient, eliminating the chance for errors in deployment. You can have certain applications run a newer version of a software or service on one container simultaneously with another container running an older more compatible version that is needed for other applications and services for either stability or security.

In the project, I set up an Ansible container on my Jump-Box-Provisioner VM to create an extra layer of security for accessing my Web Servers and then used Ansible container to help configure and install different software packages on each of the servers created with only one command. As I scaled on and added a third Web Server to my network set up, all I had to do was add that to the target list of my YAML playbook to also setup and install the same applications and packages with the settings needed for the new web server, while making sure to update my existing web servers to be the same.

The same container in the Jump-Box-Provisioner was then used when I had an ELK Server configured for deployment to use for my Elasticsearch Service along with the separate Docker containers for my Metricbeat and Filebeat services. Everything was easily tested along the way by accessing the applications through a web browser as soon as it was activated and running.

With all of the applications on the applications set up as containers, any issues that arise from the application(s) on the servers will be more or less contained to that specific container. If there are errors or problems with the container causing downtime, it can be easily removed and redeployed again from the Ansible server. This ensures quicker up-time for any services running that get compromised.

Alternatively, the same operations could have been done without containers by using only Virtual Machines, or VMs to run each of the instances/ applications. The network I set up only ran 1 container on each VM that was created for each server. The advantages with just 1 container is not as obvious, but if there were several containers running on each of the VM Servers set up, then different applications could be sharing the same VM resources, and removing/ setting up new ones would be easily managed by updating the playbook(s) on the Ansible container.

The advantage of running the services on VMs with dedicated purposes/applications rather than putting multiple containers onto the VM, would be that any attack or compromise to the VM would be limited to that particular VM. The disadvantages to this model are that it is more time consuming to set up and deploy, as well as cost additional resources in terms of hardware or financial costs.

However, the disadvantages to containers in general, is that they share the same kernel resource with physical or virtual host that they are running on. If one of the containers were compromised anyways with malware, the entire VM along with other containers could also likely be affected.