**When is it appropriate to use containers in cloud deployments, and what are the security benefits of doing so?**

**Provide a Concrete Example Scenario**

In Project 1, we used Azure portal to setup a secure cloud network. We created and launched 4 virtual machines, 1 jump-box VM to ensure the network had a secure SSH connection, 2 web VM’s and an Elk-Server VM. Docker containers were created with Ansible so that all VM’s could run separately using their own libraries and configuration files, but use the same OS and could be automated and managed leading to flexibility, auditability, and ubiquity. The Elk-Server VM was created as a collection of three open-source products — Elasticsearch, Logstash, and Kibana, to provide centralized logging in order to identify problems with the servers or applications of the 2 web VM’s and the jump-box VM. It allowed the user to search all the logs in a single place. It also helped to find issues in multiple VM’s by connecting logs during a specific time frame.

**Explain the Solution Requirements**

This was an appropriate use for the containers because in a class setting I needed something more lightweight than a VM as their images are measured in megabytes rather than gigabytes. The containers require fewer IT resources to deploy, run, and manage and they can be spun up in milliseconds. A single system can host many more containers as compared to VMs, making it more ideal in this scenario. The security benefits that I expected from using the containers were:

1. Transparency. It’s easy to look inside a container image, or the Dockerfile on which it is based, to understand what runs inside the container.
2. Modularity. If you use containers to host a microservices application, you can link security problems to specific microservices. This makes it easier to find and resolve vulnerabilities without disrupting the entire application.
3. Smaller attack surfaces. With containers, you only need to secure the host, the Docker daemon (which is much smaller than a virtual operating system) and the application running inside the container.
4. Easy updates.  The ability to update an application quickly, with minimal disruption to end users, is critical for addressing security vulnerabilities efficiently.
5. Environment parity. Containers’ ability to provide environment parity—meaning a consistent type of application environment no matter which type of operating system the application is hosted on, or whether it is running in testing or in production—is a key selling point of Docker. Environment parity is valuable because it helps ensure that a containerized application that is deemed secure before production will remain secure in production because the environment variables do not change.

**Explain the Solutions Details**

In Project 1, I configured the VM’s to be able to run containers by downloading Docker w/Ansible using the jump-box VM. Download the ansible.config configuration file and copy it to Ansible host file at /etc/ansible/hosts. For ansible.cfg edit, cd into /etc/ansible/ directory and nano ansible.cfg. Once inside the nano file, locate remote\_user and change remote\_user = sysadmin. Assign username and SSH Public Key for Web1, Web2, ELK Virtual Machine in Azure GUI

Ensure Web1 / Web2 / ELK Server’s SSH’s Public Key’s are updated by copying the sysadmin SSH Key : copy id\_rsa.pub from the ansible control node in .ssh/ directory. To get the SSH Key run this command: ~/.ssh# ssh-keygen ~/.ssh# cat id\_rsa.pub.

Create a playbook.yml file and copy the contents from that file into newly created files, installelk.yml, filebeat-config.yml, and metricbeat-config.yml. Once those files are created, configure the files to meet the parameters needed. Once the config files are created, run the ansible-playbook installelk.yml file. Download filebeat and metricbeat playbooks using curl -L -O > Ansible/filebeat/metricbeat.yml commands and update the config files using nano commands. Once this updated, check to see if the playbooks run by using you both localhost ipaddress/setup.php for ansible, and ipaddress :5601/app/kibana#/home to test Elf stack.

**Identify Advantages/Disadvantages of the Solution**

In order to have created a secure cloud network without containers, I would have had to utilize separate VM’s which would have required me to create their own operating systems to operate off of. Each VM also would have had their own binaries, libraries and applications that it services. The advantages to doing it without containers are: 1) Security; 2) Reliability; 3) ISA Structure; 4) Multiple O/S; and 5) Malware identification. The disadvantages are: 1) Cost; 2) Performance; 3) Efficiency; 4) Complexity; and 5) Infections.