Interview Question

What are the security benefits of defining cloud infrastructure as code?

Infrastructure as Code (IaC) uses machine-readable scripts and code to deploy resources to On-demand cloud computing platforms like Azure and AWS instead of relying on manual configurations. IaC can be used to configure just a single machine, or it can setup and configure an entire environment including Virtual Networks, Network Security Groups, Virtual Machines, containers, and logging services. The main security benefit of IaC is the ability to deliver a consistent and stable environment quickly with scalability. Once the code is written, it only takes a few minutes to adjust to suit specific needs. Infrastructure as Code also prevents a lot of human error as once the code has been successfully tested all a human has to do is run the command to begin the deployment.

A common tool used for IaC is Ansible. I have created several playbooks that configure two different types of virtual machines. I setup web servers that are configured with a docker container for DVWA. I also configured an Ubuntu server with a docker ELK container to monitor the traffic and statistics of the two web servers. These servers were all setup to automatically start the docker service and the containers, even after a reboot. Running IaC code consists of entering the appropriate commands for the tool you are using. Once the configurations have been completed, testing is performed by either connecting to the resources or navigating to specific sites like those on an ELK stack server. To test my configurations, I first connected to the virtual machines using SSH and checked to make sure the docker containers configured with DVWA were running using the command “sudo docker ps”. This showed that the containers were started. I was then able to navigate to the appropriate URL (http://{Public\_IP\_Address}/setup.php) which allowed me to view the setup page for my DVWA web servers. For the ELK stack, I also connected via SSH and ensured the container was started. Once I verified it was up and running, I was able to navigate to the Kibana app page using the public IP address with /app/Kibana.

Another example of a tool that can be used with IaC is Terraform. I have limited experience with it, but this tool can be used to setup entire Resource groups within services like Azure and AWS all from your command line, no need for a separate machine to run the docker container for Ansible.

While the initial creation of the code (called playbooks in Ansible) can be a lengthy task, the time saving benefit of having code to configure any number of servers the exact same way more than makes up for the front-end workload. The configuration also happens much faster than a human could perform the same tasks using a GUI.

One of the largest disadvantages of using IaC is the level of knowledge and understanding you must learn/have to write code efficiently and securely to be used to configure resources. Improperly written IaC could lead to unauthorized access. This would be extremely bad with IaC as all other resources configured in the same manner would be vulnerable. Graphical User Interfaces take less knowledge to begin using, are simple and intuitive. Most of the On-Demand cloud infrastructure services provide tutorials and how-to documents to properly setup resources using a GUI. Microsoft’s Azure has a very robust set of easy-to-understand documents that greatly assist in the configuration. These GUIs are not always easy to use or very clear though, especially without an understanding of what configuration settings mean. This can lead to improperly configured resources which could result in failed deployments at best or at worst, a malicious actor gaining entry to the system.