Cloud Management

CLOUD SERVICES LIKE AWS, AZURE is not fully responsible for the CLOUD SECURITY. Their main concern is to deploy commands, rules and management but not make them. If there is compromise on data, they are not responsible for the damage.

BEST PRACTICES and TERMS

1. Best practice to always do **Single Sign On**

**Single sign**-on (**SSO**) is a technology which combines several different application **login** screens into one. With **SSO**, a user only has to enter their **login** credentials (username, password, etc.) one time on a **single** page to access all of their SaaS applications.

2. Always do **Automation**

**Automation** is the use of electronics and **computer**-controlled devices to assume control of processes. The aim of **automation** is to boost efficiency and reliability. ... According to PC Magazine, **automation** by definition is: “Replacing manual operations with electronics and **computer**-controlled devices.

3. **Netting**

In general, **netting** is used to describe connecting **computers**, businesses, schools, or people together through networking or the Internet. ... When referring to cables, **netting** is a term sometimes used to describe networking cable.

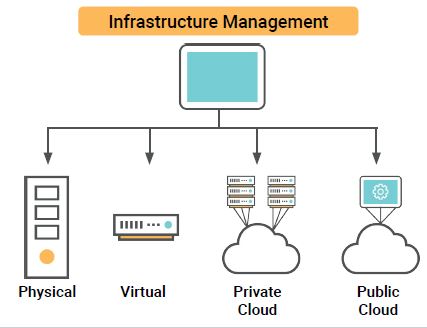
4. PCI IAC (infrastructure as code)

<https://docs.microsoft.com/en-us/microsoft-365/compliance/offering-pci-dss?view=o365-worldwide>

5. Shared Responsibility Model - Amazon Web Services (AWS)

<https://aws.amazon.com/compliance/shared-responsibility-model/#:~:text=Security%20and%20Compliance%20is%20a%20shared%20responsibility%20between%20AWS%20and%20the%20customer.&text=The%20customer%20assumes%20responsibility%20and,AWS%20provided%20security%20group%20firewall>

6. **Infrastructure Management**



**Containers**, **IaC**, and **Provisioners**

**Containers**, **infrastructure as code**, and **provisioners** are new technologies that provide powerful solutions to some of the most difficult problems of infrastructure management.

A. **Containers** (DOCKER is the most used and trusted Container, AWS and AZURE also have Container Services)

-Containers can be thought of as “lightweight VMs.”

-Containers are smaller than VMs (megabytes rather than gigabytes) and require fewer CPU resources.

-Containers is easy to destroy and deploy as needed. If the container is compromised, just destroy it and create a new one.

**Virtual machines** and **containers** differ in several ways, but the primary difference is that containers provide a way to virtualize an OS so that multiple workloads can run on a single OS instance. With VMs, the hardware is being virtualized to run multiple OS instances.

B. **Provisioners (A Tool used for AUTOMATION)**

-**Provisioners are tools** that automatically configure VMs or containers for you.

-Instead of manually logging into a machine and issuing commands like **apt get**, or editing configuration files yourself, you can use a provisioner to do this automatically.

C. **IAC**  <https://www.edx.org/course/infrastructure-as-code> (Term used for Cloud coding in Linux)

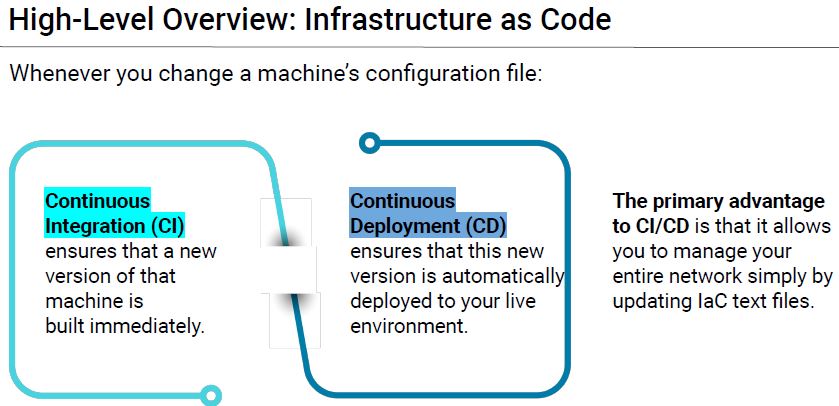
**Infrastructure as code (IaC)** is the idea that the configurations for all of the VMs, containers, and networks in your deployment should be defined in text files, which you can use with provisioners to automatically recreate machines and networks whenever necessary.

The primary benefit to **IaC** is that everyone can see exactly how the network is configured by reading text files. These can be easily version controlled with tools like Git, Apple Time Machine, or Microsoft OneDrive.

**Continuous Integration/ Continuous Deployment (CI/CD)** is the concept of automaticallyupdating machines on your networkwhenever your **IaC** files change.

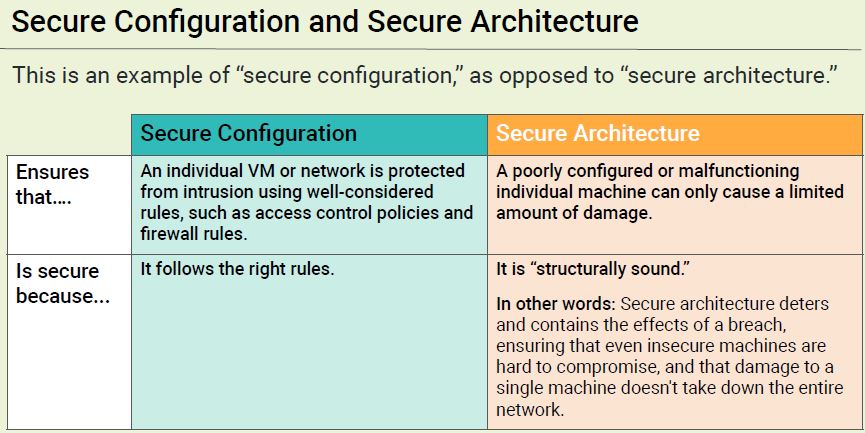
<https://azure.microsoft.com/en-ca/services/devops/>

<https://www.jenkins.io/>



7. **FOUNDATIONS OF NETWORK ARCHITECTURE**

\*\*Our virtual machine is not accessible because our security group firewall is blocking all traffic to it. This is to ensure that no one can possibly gain access to the VM.



* The best practice is to control policies not to expose the jumpbox or the gateway to the internet. The best example is to create policy that the user or client cannot connect to Facebook or Gmail or Yahoomail in the workstation otherwise it is secured otherwise the gateway will be exposed it means the architecture of the whole network is exposed.

8. NETWORK REDUNDANCY

**Network redundancy** is a process through which additional or alternate instances of **network** devices, equipment and communication mediums are installed within **network** infrastructure. It is a method for ensuring **network** availability in case of a **network** device or path failure and unavailability.

**-The best EXAMPLE** is if the 1st server is COMPROMISED the other server takes over.

**Redundancy vs. Budget**

Creating redundant systems for everything may be advisable from an engineering

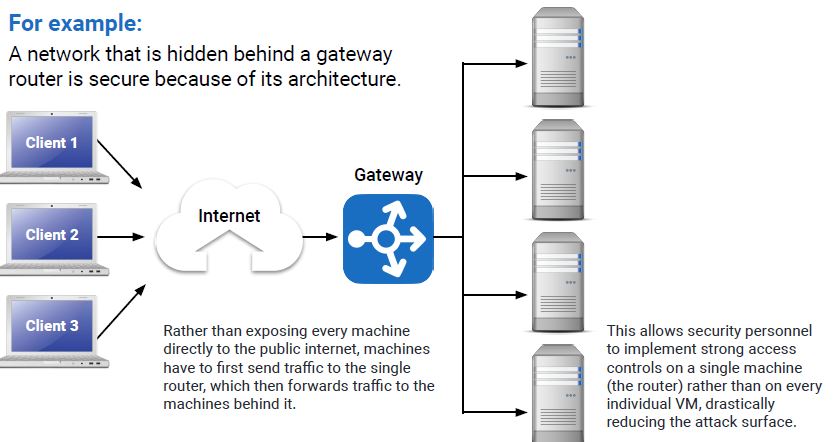
standpoint, but not every administrator will have the budget to do so.

● If Amazon was only running on one server, and the server went down, it would likely lose

millions of dollars during downtime.

● It wouldn’t make sense for a smaller web company to invest in the infrastructure needed to

support a site like Amazon. The company would likely go out of business due to unsustainable operational costs.

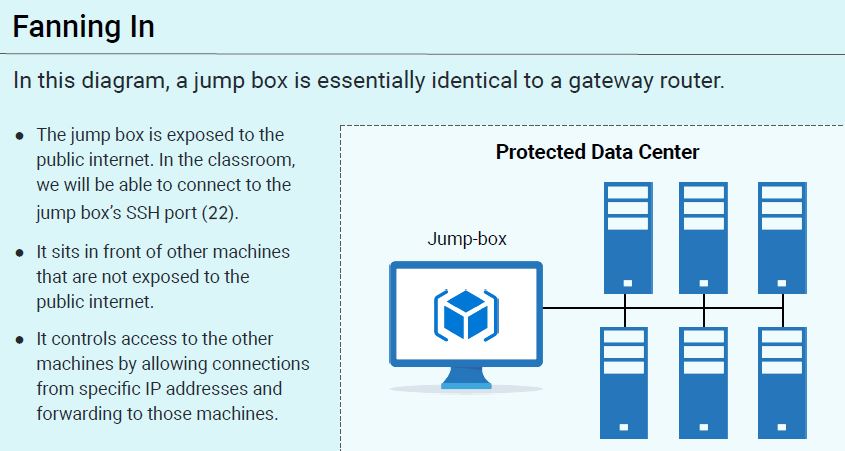


9. JUMP BOX ADMINISTRATION

A. **Fanning In**

-Placing a gateway router between VMs on a network forces all traffic through

a single node. **Securing and monitoring** this single node is called fanning in.



**BEST PRACTICE ON SECURING NETWORK IN CONNECTING TO INTERNET**

While this architecture is secure enough, we can and should further harden setups by:

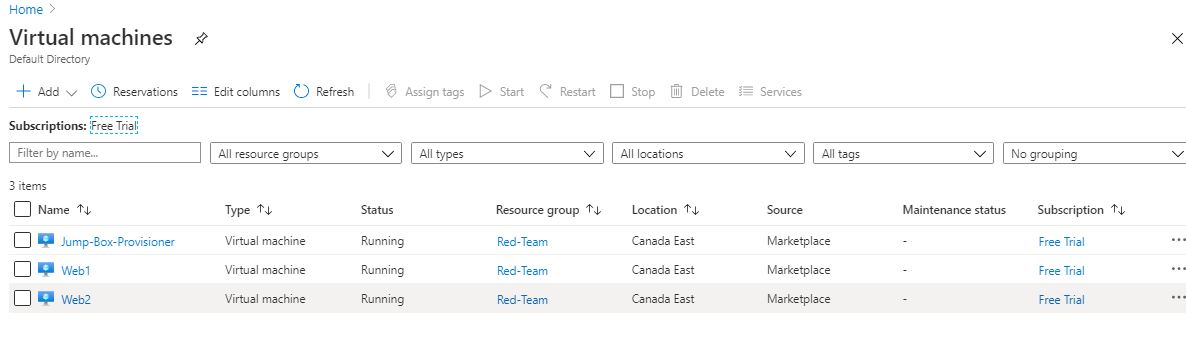
1. The jump box is exposed to the public internet. In the classroom, we will be able to connect to the jump box’s SSH port (22).
2. It sits in front of other machines that are not exposed to the public internet.
3. It controls access to the other machines by allowing connections from specific IP addresses and forwarding to those machines.
4. Limiting the number of machines that our jump box can access.
5. Locking the root account and limiting **sudo** access of the admin account on the jump box.
6. Implementing log monitoring on the jump box.
7. Implementing two-factor authentication for SSH login to the jump box.
8. Implementing a host firewall (UFW or IPtables) on the jump box.
9. Limiting jump box network access with a virtual private network (VPN).

* THE BEST EXAMPLE of NETWORK MONITORING is **KIBANA**

Kibana is an open source data visualization dashboard for Elasticsearch. It provides visualization capabilities on top of the content indexed on an Elasticsearch cluster. Users can create bar, line and scatter plots, or pie charts and maps on top of large volumes of data.

ACTIVITY 1 : (SETUP AZURE JUMPBOX PROVISIONER)

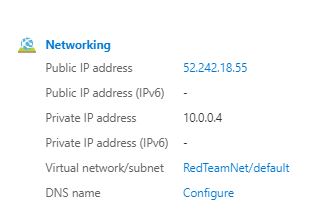
Step 1. Connect and run the Virtual Machines



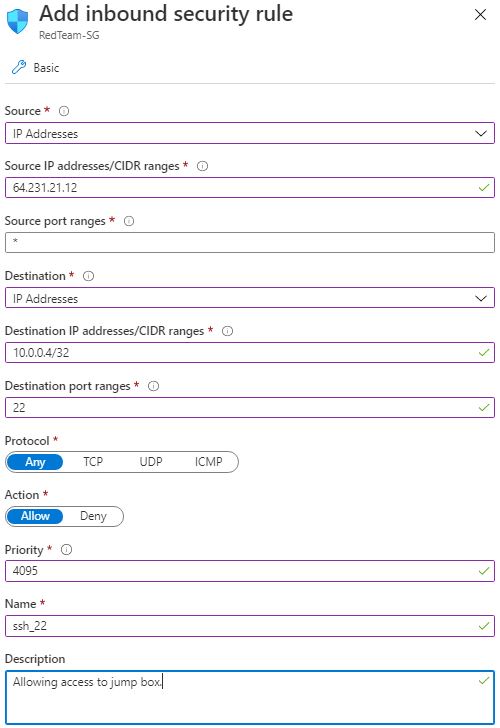
Step 2. Create firewall and connection on jumpbox Provisioner

PRIVATE IP 10.0.0.4 (AZURE > Jump-Box-Provisioner > Private IP)

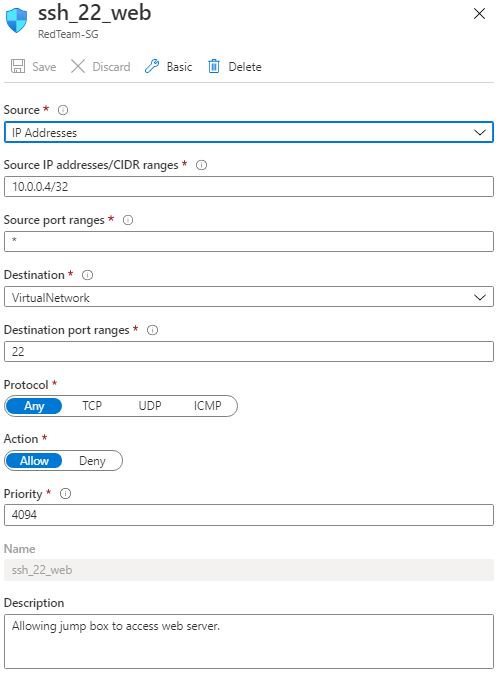
My Public IPv4 is: 64.231.21.12 (Go to Google.com > search https://www.whatismyip.com/)



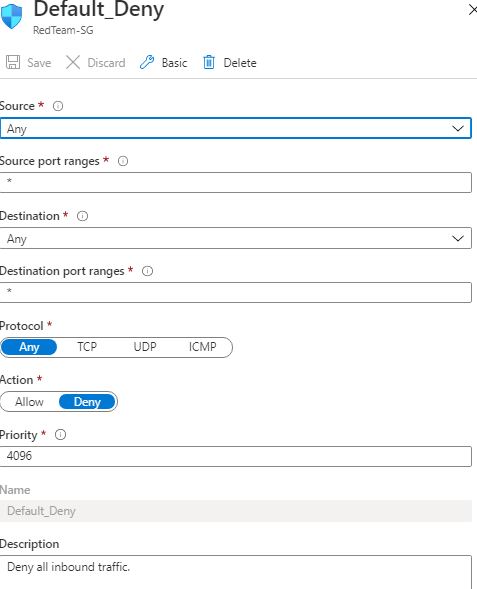
FIREWALL 1: Port 22 Priority 4095 (Allow client access to jump box.)



FIREWALL 2: Port 22 Priority 4094 (ALLOW JUMP BOX TO ACCESS WEB SERVER)

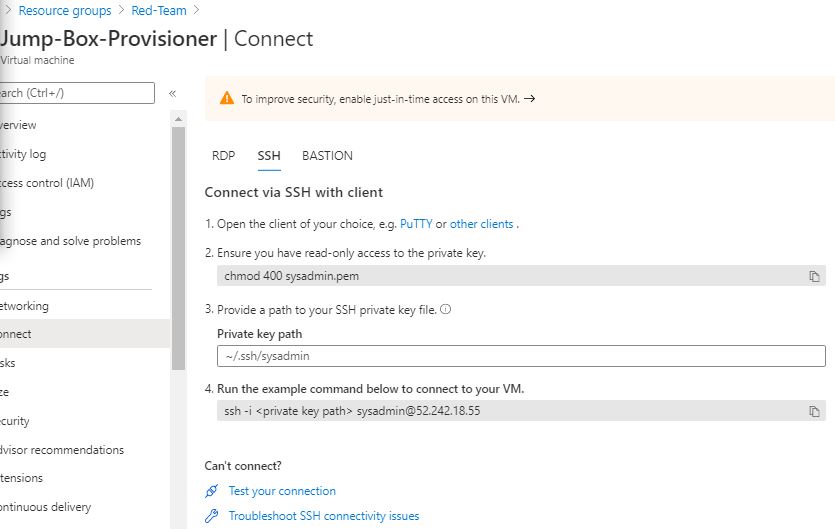


FIREWALL 3: Deny all Access Priority 4096



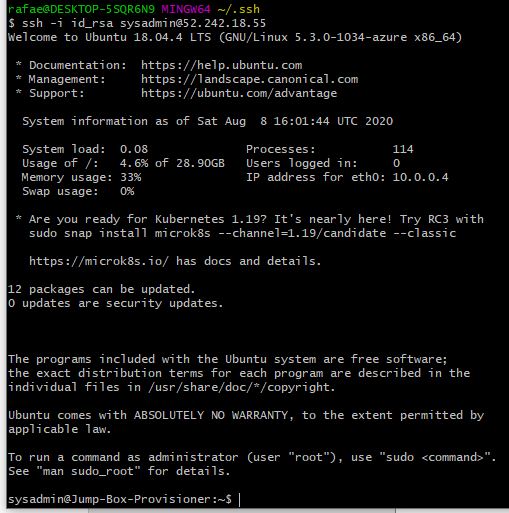
Step 3. Connect the the JUMP-BOX-PROVISIONER to the SSH CLIENT

A. Jump-Box-Provisioner > Connect > SSH and Copy the command in Step 4 below

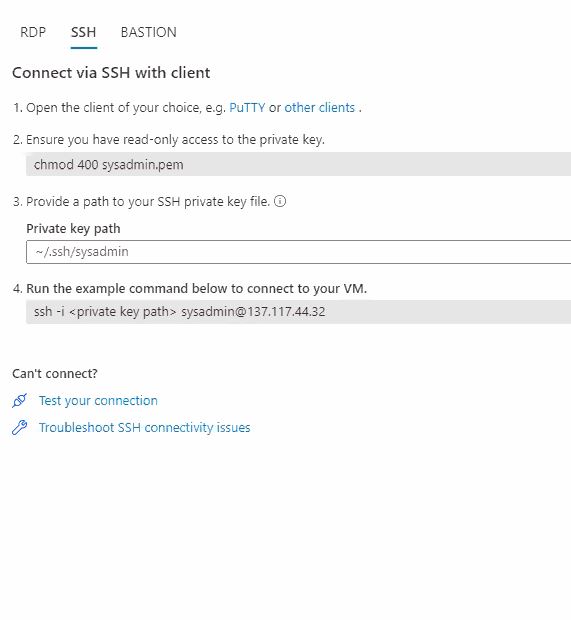
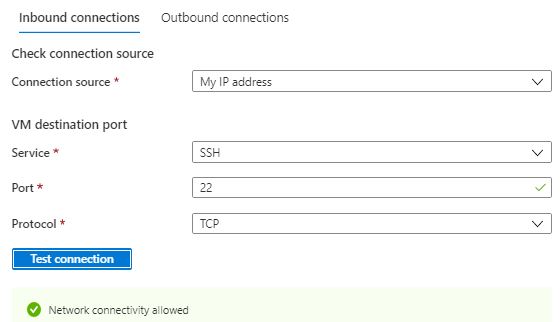


B. Run this command in Git Bash (Linux Based)

‘cd ~/.ssh’

‘ssh -I id\_rsa sysadmin@52.242.18.55’

C. Test the Connection of the JUMP BOX



END OF ACTIVITY : CREATE A PROVISIONER to the JUMPBOX

B. JUMP Box Administration > Containers

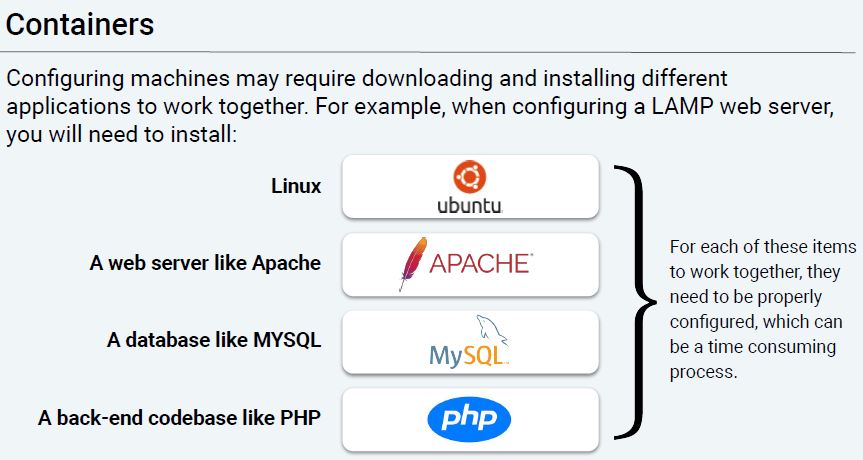
AZURE DOCKER (CONTAINER) SECURITY

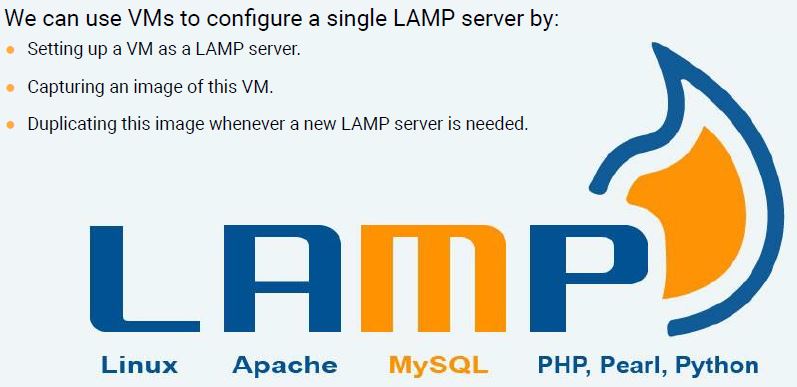
-is a best example of a container

CONTAINER VS VM

-A container contain 1 service while a VM have multiple containers

-If 1 container fails or became compromised, just destroy it and create a new one





* + Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels.
  + A Docker is a like an IMAGE file saved on the CLOUD that contains services ready to be deployed.

ACTIVITY 2: DOCKER SERVICE (Continuation from ACTIVITY 1)

A. PERFORM BELOW IN SEQUENCE to INSTALL AND ENABLE DOCKER SERVICE

sudo apt-get update

sudo apt install [docker.io](https://slack-redir.net/link?url=http%3A%2F%2Fdocker.io)

sudo service docker start

systemctl status docker

B. DOWNLOAD AN OPERATING SYSTEM IN DOCKER HUB <https://hub.docker.com/>

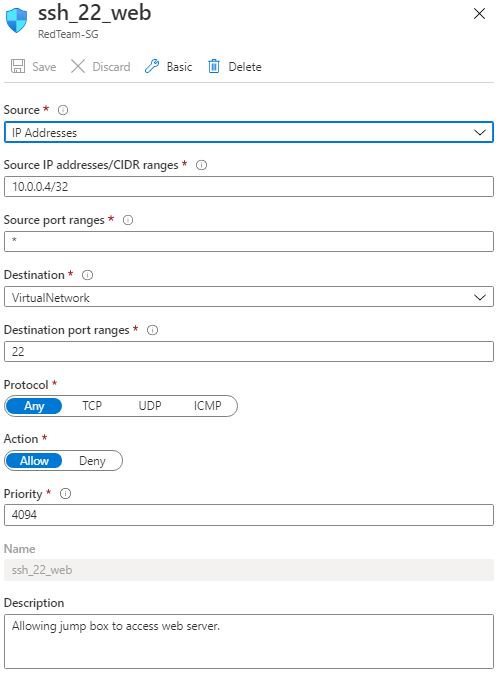
sudo docker pull cyberxsecurity/ansible

C. RUN and AUTOMATE DOCKER WITH ANSIBLE

**ANSIBLE** IS THE WAY TO AUTOMATE **DOCKER** IN YOUR ENVIRONMENT. ANSIBLE ENABLES YOU TO OPERATIONALIZE YOUR DOCKER CONTAINER BUILD AND DEPLOYMENT PROCESS IN WAYS THAT YOU'RE LIKELY DOING MANUALLY TODAY, OR NOT DOING AT ALL.

IN AZURE PORTAL > GET THE JUMPBOX PRIVATE IP ADDRESS: 10.0.0.4

CREATE ssh\_22\_web INBOUND SECURITY (THIS IS ALREADY CREATED in ACTIVIY 1 just to show the best practice step in Jump Box Administration



ACTIVITY 2 SUMMARY

Steps to get into Web 2 server using Ansible Container (DOCKER)

IN SEQUENCE :

CONNECT TO THE JUMP-BOX-PROVISIONER VM (This jump box provisioner have 3 different firewalls, this is where we control all the services and security zones of all the other VMs)

JUMP-BOX-ROVISIONER PUBLIC IP: [40.86.250.146](https://portal.azure.com/)

Web2 Public IP : [40.86.250.146](https://portal.azure.com/)

Web2 Private IP : 10.0.0.6

KEY : id\_rsa , id\_rsa\_pub

‘cd ~/.ssh’

‘ssh -i id\_rsa [sysadmin@52.242.18.55](mailto:sysadmin@52.242.18.55)’

sysadmin@Jump-Box-Provisioner:~$

NEXT STEP is to create and attach the container the whole provision.

sudo apt-get update - this will update all packages

sudo apt install [docker.io](https://slack-redir.net/link?url=http%3A%2F%2Fdocker.io) - install docker application

sudo service docker start - start the docker application

systemctl status docker - status of the docker application

sudo docker pull cyberxsecurity/ansible - download the docker file

sudo docker run -ti cyberxsecurity/ansible bash – run and create a docker bash

then exit to exit on the container

sudo docker start <image-name> - starts the image specified  
sudo docker ps -a - list all active/inactive containers

sudo docker attach <image-name> - effectively sshing into the ansible

container

\*\*\*After sshing to the container, assign the password for Web2 VM (Container VM)

Web2 > Reset Password > Reset SSH Public Key – in this case we used id\_rsa\_pub

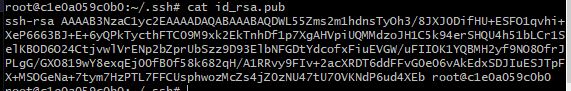
\*\*\* Create a ssh key inside the ansible

sudo docker start <image-name>

sudo docker ps -a

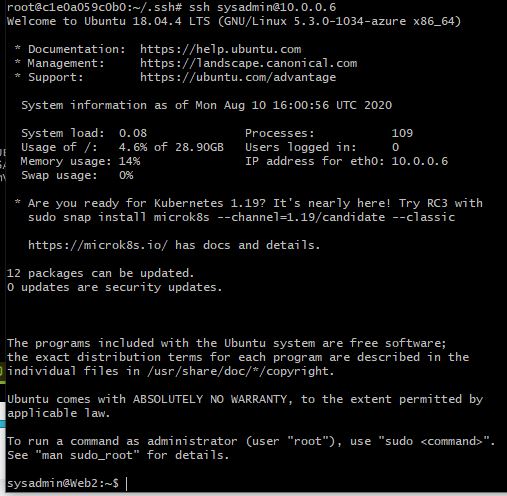
sudo docker attach <image-name> >>> to to root folder of the container





TEST CONNECT to WEB2 VM  
ssh sysadmin@<web\_2\_private\_ip> - this is to login to the web 2 server

Web2 Private IP : 10.0.0.6



CREATE A SIMPLE CONFIGURATION FOR THE ANSIBLE OR CONTAINER (FOR AUTOMATION)

exit (on Web2 and go back to ansible root#)

cd ~/.ssh

cd /etc/ansible/

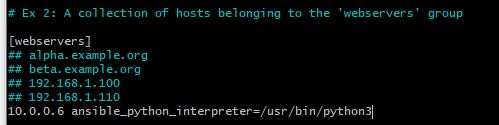
nano hosts



Once inside the hosts file > CTRL + W (to Search) > Enter webserver

In this case, we are enabling 10.0.0.6 to the WebServers

‘ansible\_python\_interpreter=/usr/bin/python3’ is an additional rule



Exit and save.

Create another configuration under ansible.cfg

nano ansible.cfg

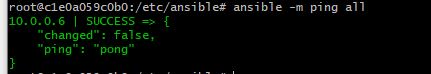
CTRL + W > enter remote\_user

remote\_user = sysadmin (This will enable a user “sysadmin” on the Container Server(Web2)



Exit and save.

Check the connection of ansible to the server. ansible m -ping all



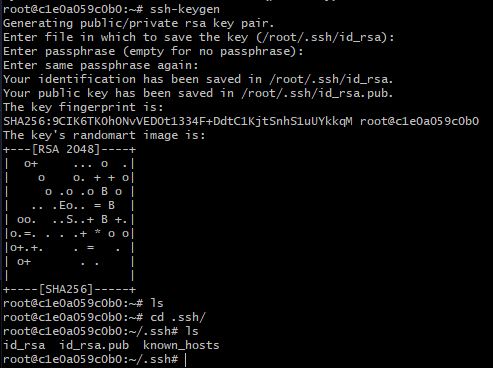
OTHER NOTE:

Create a ssh key

1. cd ~/.ssh

2. ssh-keygen <optional to input name of the keygen file>

For the Web2 ssh key, ssh first to the image then create a key from there.



Other way to attach the docker file and ssh execute as root

sysadmin@Jump-Box-Provisioner:~$ sudo docker exec -it <image name or container ID> bash

sysadmin@Jump-Box-Provisioner:~$ sudo docker exec -it <image name or container ID> /bin/bash

Where : ‘it’ means interactive terminal, this will not shutdown the image

It will make the image container persistent on exit

sysadmin@Jump-Box-Provisioner:~$ sudo docker attach -d <container>

Where : -d will run the container in the background consistently

END OF ACTIVITY 2

BEST PRACTICE FOR CLOUD MANAGEMENT :

<https://www.servicenow.com/content/dam/servicenow-assets/public/en-us/doc-type/resource-center/solution-brief/sb-cloud-management.pdf>