



# **CLOUD SECURITY FUNDAMENTALS V2**

## **Lab 3: Container Network and Security**

**Document Version: 2022-12-22**

Copyright © 2022 Network Development Group, Inc.  
[www.netdevgroup.com](http://www.netdevgroup.com)

NETLAB+ is a registered trademark of Network Development Group, Inc.

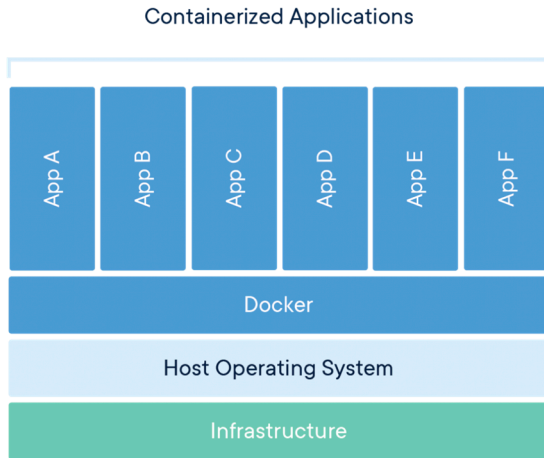
Palo Alto Networks and the Palo Alto Networks logo are trademarks or registered trademarks of Palo Alto Networks, Inc.

## Contents

Introduction .....	3
Objective .....	3
Lab Topology .....	4
Lab Settings .....	5
1 Protecting Sensitive Data .....	6
1.0 Load Lab Configuration .....	6
1.1 Create a Docker Container Network .....	11
1.2 Pull a Container Image and Run a Docker Container .....	15
1.3 Map the Host Port to the Running Web Container and Access the Container using the Web Browser .....	21

## Introduction

In this lab, you will use a Docker Container for Network Security.

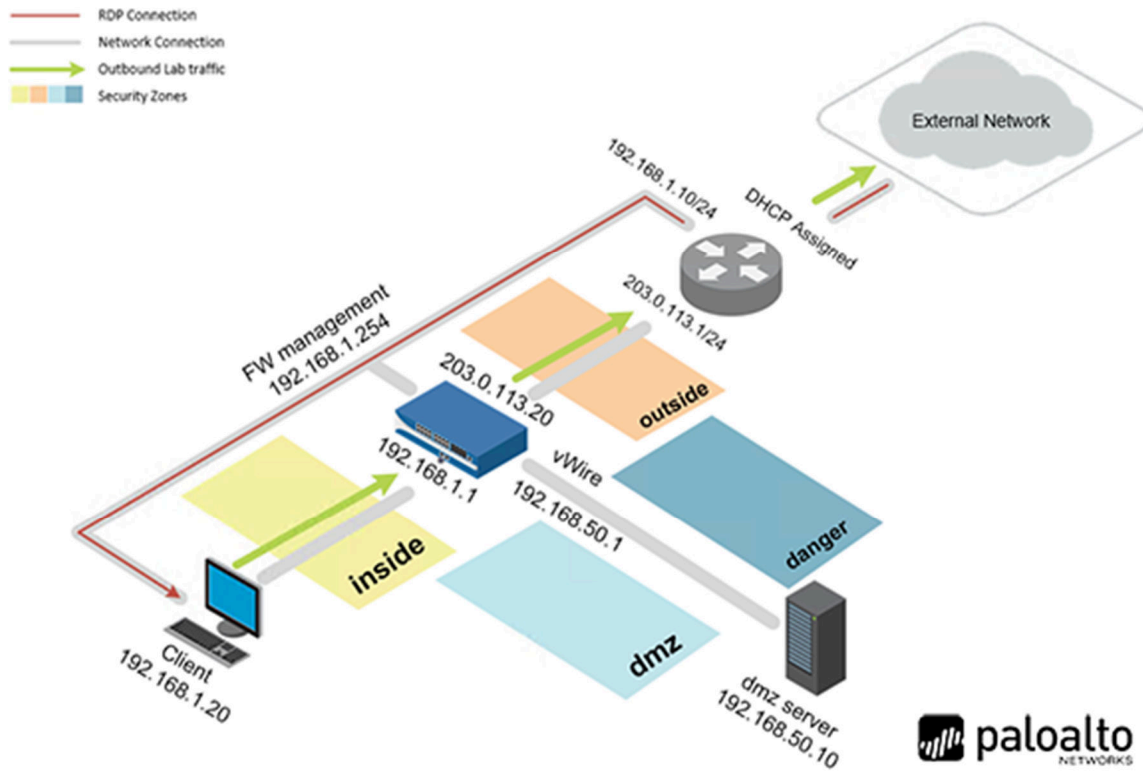


## Objective

In this lab, you will perform the following tasks:

- Create a Docker bridge network
- Pull Container images from Docker public repository
- Run containers in detached and interactive mode on created Docker bridge network
- Test container network connectivity
- Run Web nginx Docker container in detached mode and assign host ports to the container
- Test container Web application

## Lab Topology



## Lab Settings

The information in the table below will be needed in order to complete the lab. The task sections below provide details on the use of this information.

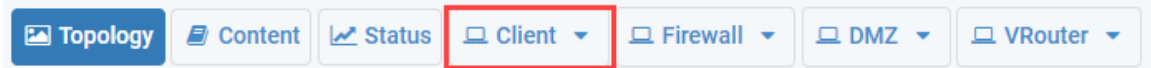
Virtual Machine	IP Address	Account (if needed)	Password (if needed)
Client	192.168.1.20	lab-user	Pal0Alt0!
DMZ	192.168.50.10	root	Pal0Alt0!
Firewall	192.168.1.254	admin	Pal0Alt0!

## 1 Protecting Sensitive Data

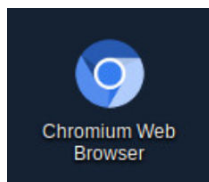
### 1.0 Load Lab Configuration

In this section, you will load the Firewall configuration file.

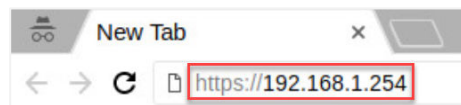
1. Click on the **Client** tab to access the Client PC.



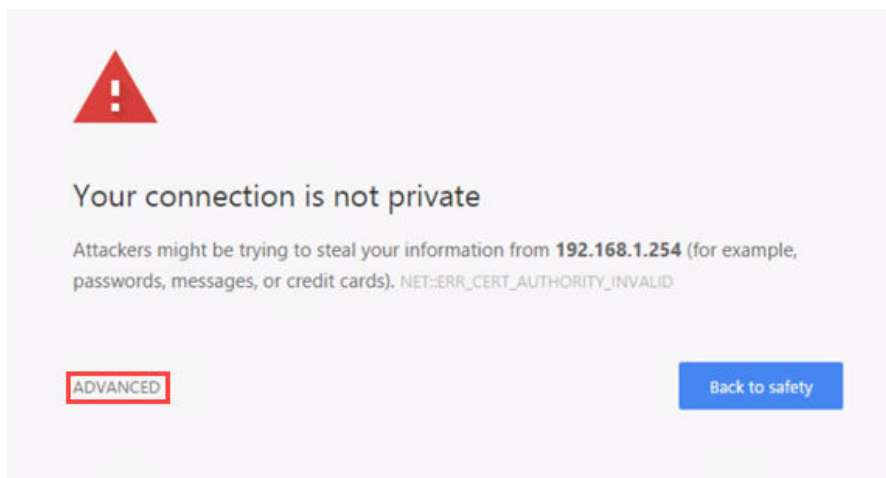
2. Log in to the Client PC as username `lab-user`, password `Pa10Alt0!`.
3. Double-click the **Chromium Web Browser** icon located on the Desktop.



4. In the *Chromium* address field, type `https://192.168.1.254` and press **Enter**.

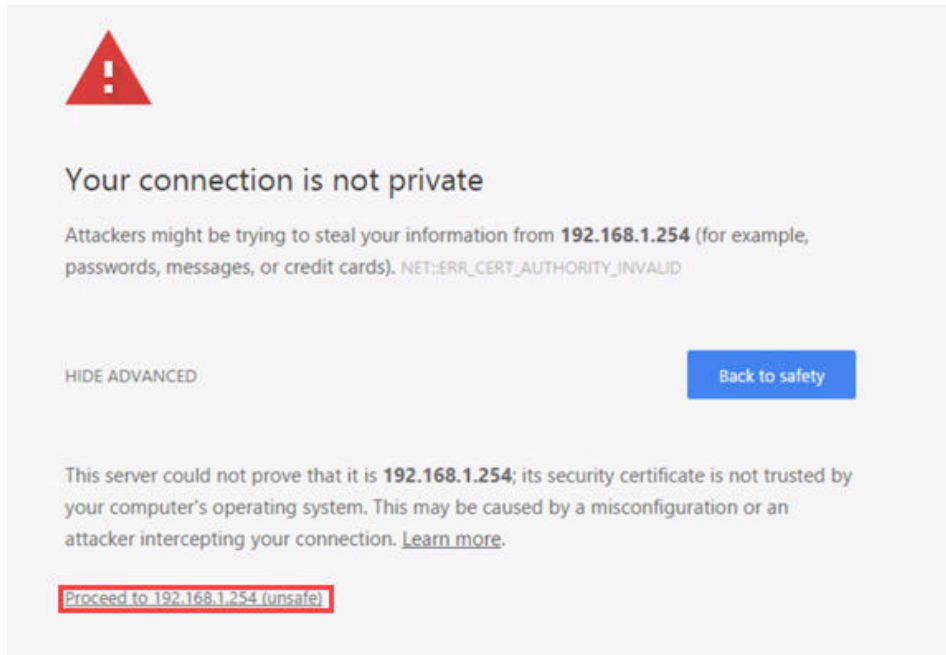


5. You will see a “Your connection is not private” message. Click on the **ADVANCED** link.



If you experience the “Unable to connect” or “502 Bad Gateway” message while attempting to connect to the specified IP above, please wait an additional 1-3 minutes for the Firewall to fully initialize. Refresh the page to continue.

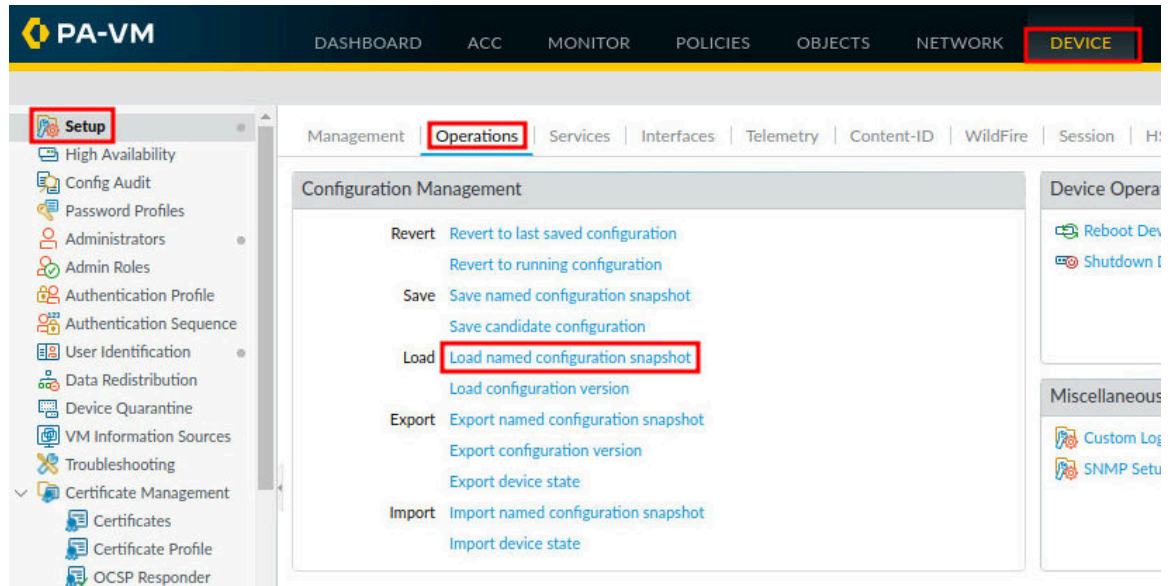
- Click on **Proceed to 192.168.1.254 (unsafe)**.



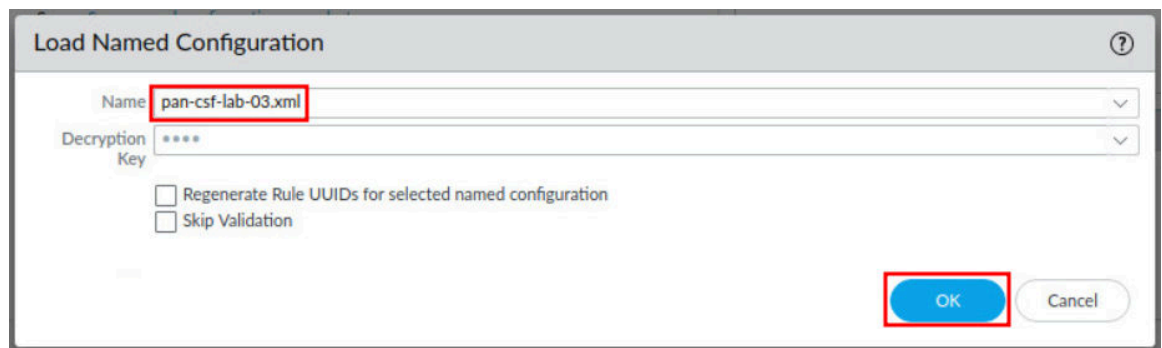
- Log in to the Firewall web interface as username admin, password Pal0Alt0!.



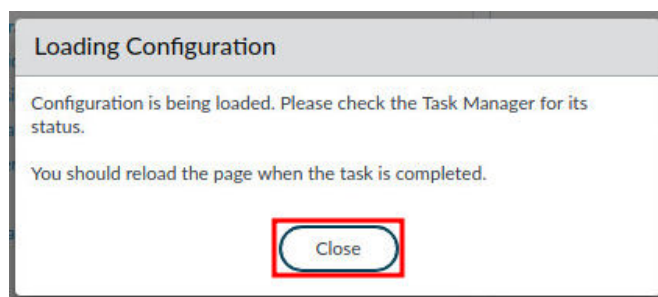
- In the web interface, navigate to **Device > Setup > Operations** and click on **Load named configuration snapshot** underneath the *Configuration Management* section.



- In the *Load Named Configuration* window, select **pan-csf-lab-03.xml** from the *Name* dropdown box and click **OK**.



- In the Loading Configuration window, a message will show *Configuration is being loaded. Please check the Task Manager for its status. You should reload the page when the task is completed.* Click **Close** to continue.

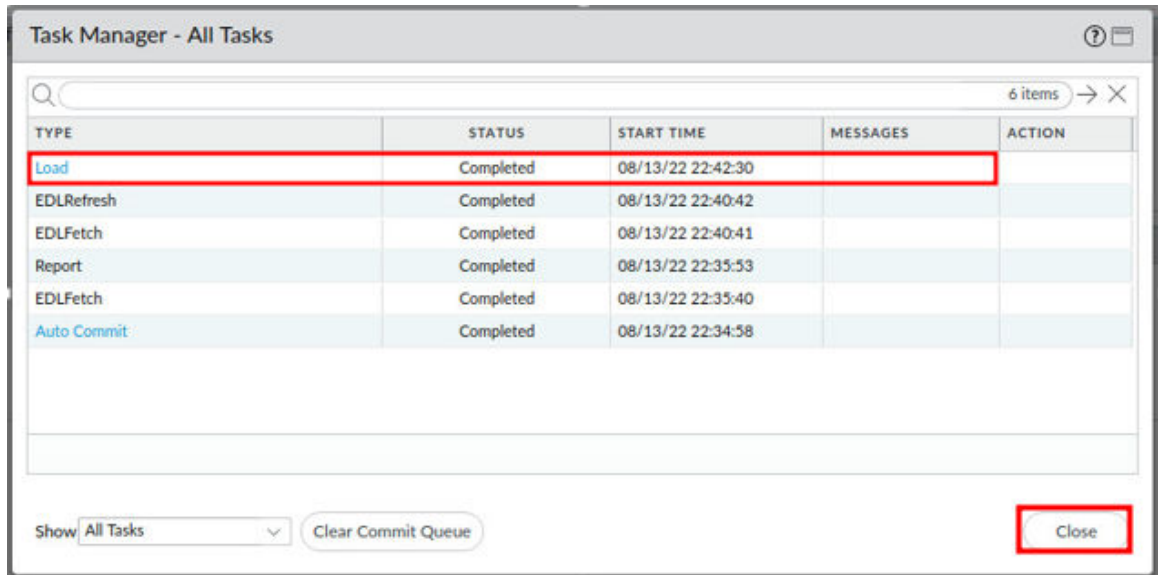




11. Click the **Tasks** icon located at the bottom-right of the web interface.



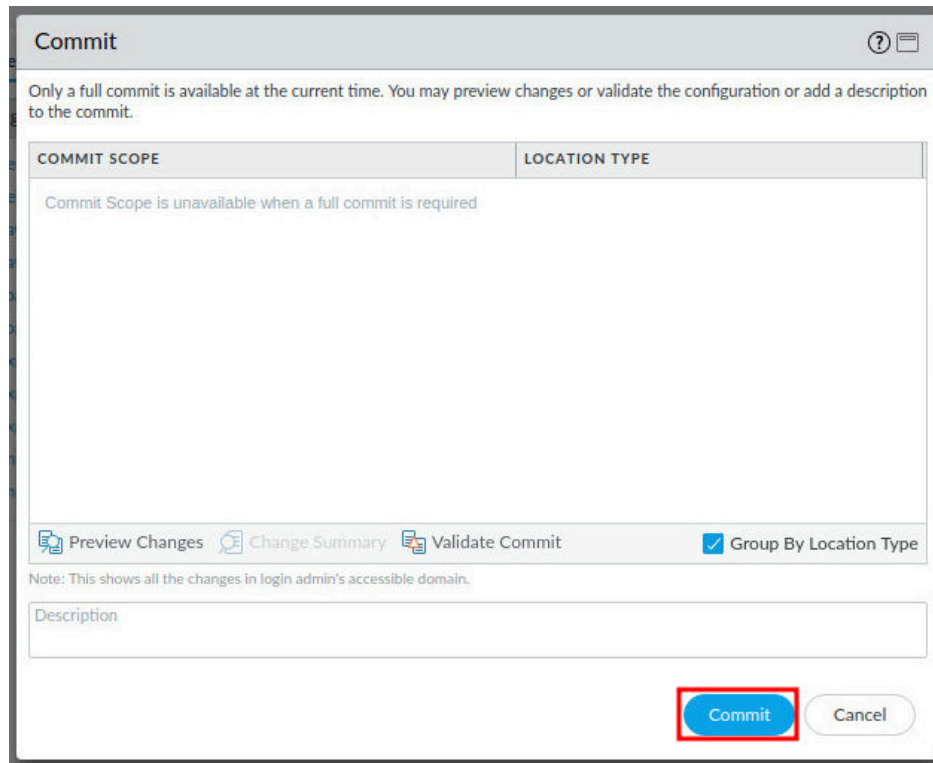
12. In the *Task Manager – All Tasks* window, verify the *Load* type has successfully completed. Click **Close**.



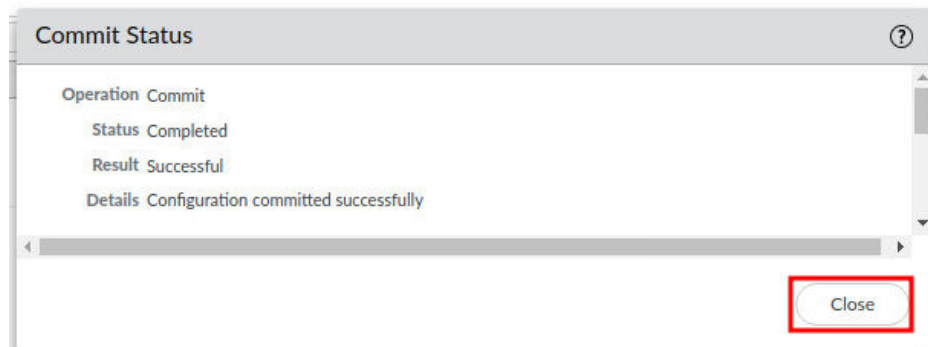
13. Click the **Commit** link located at the top-right of the web interface.



14. In the *Commit* window, click **Commit** to proceed with committing the changes.



15. When the commit operation successfully completes, click **Close** to continue.



The commit process takes changes made to the Firewall and copies them to the running configuration, which will activate all configuration changes since the last commit.

## 1.1 Create a Docker Container Network

In this section, you will create a docker container network. Docker provides two network drivers, bridge and overlay. Once the Docker Container network has been installed, it will include three networks.

1. Launch **Xfce** Terminal in the lower-left of the student *Desktop*.



2. SSH to the *DMZ* by typing the command below. Use **Pa10Alt0!** for the password. If prompted, enter yes to continue connecting. Press **Enter**.

```
C:\home\lab-user> ssh root@192.168.50.10
```



3. Display the default docker daemon networks by typing the command below.

```
[root@pod-dmz ~]# docker network ls
```

```
[root@pod-dmz ~]# docker network ls  
NETWORK_ID          NAME             DRIVER          SCOPE  
7b784bc3206f        bridge          bridge          local  
4ffda97343ab        host            host            local  
709d1e0f4031        none            null            local  
[root@pod-dmz ~]#
```



The “bridge” is the default docker network that running containers will attach to if a network is not specified when running the container. The host network is the docker network that is attached to the host running the docker daemon.

4. Create a bridge network by typing the command below.

```
[root@pod-dmz ~]# docker network create --driver=bridge \
--subnet=172.16.3.0/24 csf-br0
```

```
[root@pod-dmz ~]# docker network create --driver=bridge --subnet=172.16.3.0/24 csf-br0
b3bec780149f9c371e2c4e7b902cd850336874f9d9f070cabe88c256a5fbaa
[root@pod-dmz ~]#
```



The command above is a single line of code that is too long to display, so it is “split” into multiple lines by using the “\” followed by pressing the Enter key. This will proceed to the next line allowing you to type the remaining command.

5. Confirm the newly created bridge network has been created by typing the command below.

```
[root@pod-dmz ~]# docker network ls
```

```
[root@pod-dmz ~]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
7b784bc3206f        bridge              bridge               local
4ffda97343ab        host                host                 local
709d1e0f4031        none                null                 local
[root@pod-dmz ~]# docker network create --driver=bridge --subnet=172.16.3.0/24 csf-br0
b3bec780149f9c371e2c4e7b902cd850336874f9d9f070cabe88c256a5fbaa
[root@pod-dmz ~]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
7b784bc3206f        bridge              bridge               local
b3bec780149f        csf-br0              bridge               local
4ffda97343ab        host                host                 local
709d1e0f4031        none                null                 local
[root@pod-dmz ~]#
```

Default docker networks

Newly created bridge network

**Please  
Note**

As you can see from the CLI output, you have created a docker bridge network named csf-br0. You will use this network to attach your running containers.

6. View the details about the **csf-br0** network by typing the command below. The CLI output will be in json file format, and you will be able to view all the details about the bridge network to include its assigned subnet.

```
[root@pod-dmz ~]# docker network inspect csf-br0
```

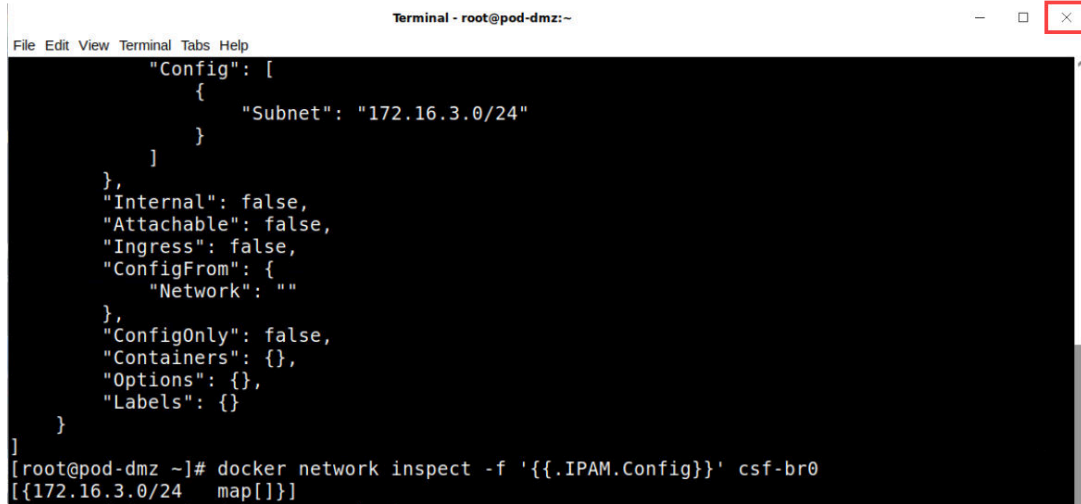
```
[root@pod-dmz ~]# docker network inspect csf-br0
[
  {
    "Name": "csf-br0",
    "Id": "b3bec780149f9c371e2cff4e7b902cd850336874f9d9f070cabe88c256a5fbaa",
    "Created": "2021-01-03T18:26:58.662237584Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.16.3.0/24"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
```

7. Parse the json file to describe your bridge network and display its subnet address by typing the command below.

```
[root@pod-dmz ~]# docker network inspect -f '{{.IPAM.Config}}' csf-br0
```

```
[root@pod-dmz ~]# docker network inspect -f '{{.IPAM.Config}}' csf-br0
[[{"Subnet": "172.16.3.0/24"}]]
[root@pod-dmz ~]#
```

8. Close the *Terminal* window.



```
Terminal - root@pod-dmz:~  
File Edit View Terminal Tabs Help  
    "Config": [  
      {  
        "Subnet": "172.16.3.0/24"  
      }  
    ],  
    "Internal": false,  
    "Attachable": false,  
    "Ingress": false,  
    "ConfigFrom": {  
      "Network": ""  
    },  
    "ConfigOnly": false,  
    "Containers": {},  
    "Options": {},  
    "Labels": {}  
  }  
]  
[root@pod-dmz ~]# docker network inspect -f '{{.IPAM.Config}}' csf-br0  
[{"172.16.3.0/24": "map[]"}]
```

9. If a *Warning* window appears, click **Close Window**. Continue to the next task.



## 1.2 Pull a Container Image and Run a Docker Container

In this section, you will pull a container image from the docker public repository and run a docker container.

1. Launch **Xfce** Terminal in the lower-left of the student *Desktop*.



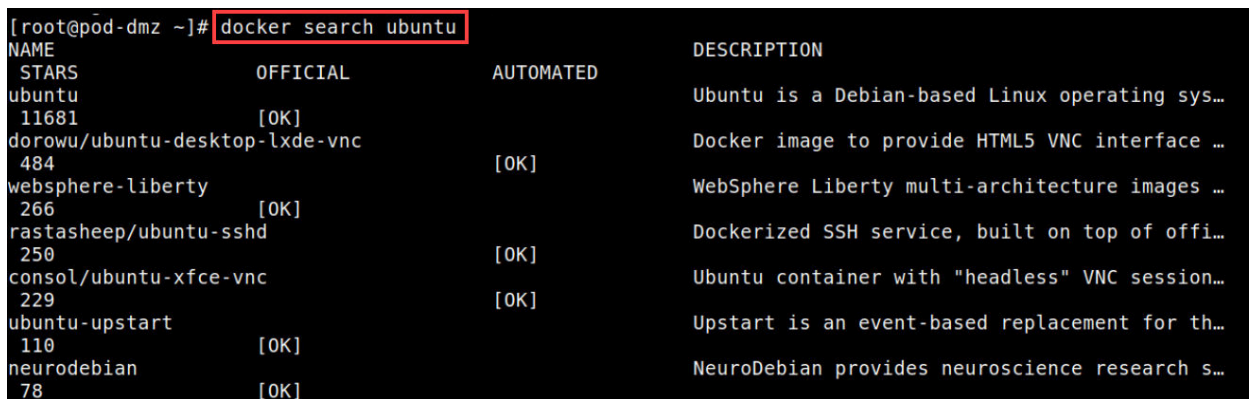
2. SSH to the *DMZ* by typing the command below. Use **Pa10Alt0!** for the password. If prompted, enter yes to continue connecting. Press **Enter**.

```
C:\home\lab-user> ssh root@192.168.50.10
```



3. In the Terminal window, display the ubuntu container images by typing the command below.

```
[root@pod-dmz ~]# docker search ubuntu
```





4. Pull the 20.04 "Focal Fossa" container image of ubuntu from the docker public repository by typing the command below.

```
[root@pod-dmz ~]# docker image pull ubuntu:focal
```

```
[root@pod-dmz ~]# docker image pull ubuntu:focal
focal: Pulling from library/ubuntu
3b65ec22a9e9: Pull complete
Digest: sha256:af5efa9c28de78b754777af9b4d850112cad01899a5d37d2617bb94dc63a49aa
Status: Downloaded newer image for ubuntu:focal
docker.io/library/ubuntu:focal
[root@pod-dmz ~]#
```

5. Display all the docker images stored on the DMZ by typing the command below.

```
[root@pod-dmz ~]# docker image ls
```

```
[root@pod-dmz ~]# docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ubuntu	focal	3bc6e9f30f51	3 weeks ago	72.8MB
icarossio/metasploitable2	latest	7a129e1a0be3	2 years ago	1.51GB
vulnerables/web-dvwa	latest	ab0d83586b6e	3 years ago	712MB

```
[root@pod-dmz ~]#
```

6. Run the docker container named **csf-ubuntu1** by typing the command below.  
This command will also attach the container to the **csf-br0** network.

```
[root@pod-dmz ~]# docker run -i -d --name csf-ubuntu1 --network=csf-br0 \
ubuntu:focal
```

```
[root@pod-dmz ~]# docker run -i -d --name csf-ubuntu1 --network=csf-br0 \
> ubuntu:focal
5aff54739090b4aa45254e72ea751ee4bb920fe8778264c4ded5402a4866226c
[root@pod-dmz ~]#
```

7. View the running containers by typing the command below.

```
[root@pod-dmz ~]# docker ps -a
```

```
[root@pod-dmz ~]# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND NAMES	CREATED
a6beb7299947	ubuntu:focal	"bash"	11 seconds ago
Up 9 seconds		csf-ubuntu1	

```
[root@pod-dmz ~]#
```



8. View the details of the running container in json format by typing the command below. Record the IP Address of **172.16.3.2** as you will need this in a later step.

```
[root@pod-dmz ~]# docker inspect csf-ubuntu1
```

```
[root@pod-dmz ~]# docker inspect csf-ubuntu1
[
  {
    "Id": "fde507414e2d1959702ede099d4a69924ee7ca96c695b3a52d4030adab8a1c99",
    "Created": "2021-01-03T20:25:31.735149734Z",
    "Path": "/bin/bash",
    "Args": [],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 2127,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2021-01-03T20:25:32.426930876Z",
      "FinishedAt": "0001-01-01T00:00:00Z"
    },
    "Image": "sha256:f643c72bc25212974c16f3348b3a898b1ec1eb13ec1539e10a103e6e217eb2f1",
    "ResolvConfPath": "/var/lib/docker/containers/fde507414e2d1959702ede099d4a69924ee7ca96c695b3a52d4030adab8a1c99/resolv.conf",
```

```
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
    "IPAddress": "",
    "IPPrefixLen": 0,
    "IPv6Gateway": "",
    "MacAddress": "",
    "Networks": {
      "csf-br0": {
        "IPAMConfig": null,
        "Links": null,
        "Aliases": [
          "fde507414e2d"
        ],
        "NetworkID": "b58feee35613dce8e7788b2b4ad44d66d160c5f6c1827118113200bf57b45b99",
        "EndpointID": "aa1a541be76a4b65d7b6a8999ac7701c33d8c18d2475535b93f82a17bde0ac7c",
        "Gateway": "172.16.3.1",
        "IPAddress": "172.16.3.2",
        "IPPrefixLen": 24,
        "IPv6Gateway": "",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "MacAddress": "02:42:ac:10:03:02",
        "DriverOpts": null
      }
    }
  }
]
```

9. Parse the json file with just the containers IP Address by typing the command below.

```
[root@pod-dmz ~]# docker inspect \
--format='{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' \
csf-ubuntu1
```

```
[root@pod-dmz ~]# docker inspect \
> --format='{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' \
> csf-ubuntu1
172.16.3.2
[root@pod-dmz ~]#
```

10. Run another container called **csf-ubuntu2** and attach the container to the **csf-br0** network bridge by typing the command below.

```
[root@pod-dmz ~]# docker run -i -d --name csf-ubuntu2 --network=csf-br0 \
ubuntu:focal
```

```
[root@pod-dmz ~]# docker run -i -d --name csf-ubuntu2 --network=csf-br0 \
> ubuntu:focal
44ffcd5293a160b33f449368017017c01a71003e1246881f723693d2ae6b464d
[root@pod-dmz ~]#
```

11. Interact with the docker container bash shell to run commands by typing the command below.

```
[root@pod-dmz ~]# docker exec -it csf-ubuntu2 /bin/bash
```

```
[root@pod-dmz ~]# docker exec -it csf-ubuntu2 /bin/bash
root@27b737a1a215:/#
```

12. After the container's bash shell appears, type the command below to update the container repository to the newest version of packages and their dependencies.

```
root@27b737a1a215:/# apt-get update
```

```
root@27b737a1a215:/# apt-get update
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [109 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:4 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [103 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal-backports InRelease [101 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 Packages [1167 B]
Get:7 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [495 kB]
Get:8 http://archive.ubuntu.com/ubuntu focal/multiverse amd64 Packages [177 kB]
Get:9 http://archive.ubuntu.com/ubuntu focal/restricted amd64 Packages [33.4 kB]
Get:10 http://archive.ubuntu.com/ubuntu focal/main amd64 Packages [1275 kB]
Get:11 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [645 kB]
Get:12 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages [11.3 MB]
Get:13 http://archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 Packages [30.4 kB]
Get:14 http://archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [136 kB]
Get:15 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [885 kB]
Get:16 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [885 kB]
Get:17 http://archive.ubuntu.com/ubuntu focal-backports/universe amd64 Packages [4250 B]
Fetched 16.6 MB in 3s (6587 kB/s)
Reading package lists... Done
root@27b737a1a215:/#
```

13. Install the ping utility by typing the command below.

```
root@27b737a1a215:/# apt-get install iputils-ping -y
```

```
root@27b737a1a215:/# apt-get install iputils-ping -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libcap2 libcap2-bin libpam-cap
The following NEW packages will be installed:
  iputils-ping libcap2 libcap2-bin libpam-cap
0 upgraded, 4 newly installed, 0 to remove and 2 not upgraded.
Need to get 90.5 kB of archives.
After this operation, 333 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/main amd64 libcap2 amd64 1:2.32-1 [15.9 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal/main amd64 libcap2-bin amd64 1:2.32-1 [26.2 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal/main amd64 iputils-ping amd64 3:20190709-3 [40.1 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal/main amd64 libpam-cap amd64 1:2.32-1 [8352 B]
Fetched 90.5 kB in 0s (192 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package libcap2:amd64.
(Reading database ... 4121 files and directories currently installed.)
Preparing to unpack .../libcap2_1%3a2.32-1_amd64.deb ...
Unpacking libcap2:amd64 (1:2.32-1) ...
Selecting previously unselected package libcap2-bin.
Preparing to unpack .../libcap2-bin_1%3a2.32-1_amd64.deb ...
Unpacking libcap2-bin (1:2.32-1) ...
```

14. Ping the newly created container by typing the command below. Notice the pings are successful.

```
root@27b737a1a215:/# ping 172.16.3.2
```

```
root@27b737a1a215:/# ping 172.16.3.2
PING 172.16.3.2 (172.16.3.2) 56(84) bytes of data:
64 bytes from 172.16.3.2: icmp_seq=1 ttl=64 time=0.129 ms
64 bytes from 172.16.3.2: icmp_seq=2 ttl=64 time=0.070 ms
64 bytes from 172.16.3.2: icmp_seq=3 ttl=64 time=0.114 ms
64 bytes from 172.16.3.2: icmp_seq=4 ttl=64 time=0.098 ms
64 bytes from 172.16.3.2: icmp_seq=5 ttl=64 time=0.146 ms
64 bytes from 172.16.3.2: icmp_seq=6 ttl=64 time=0.111 ms
64 bytes from 172.16.3.2: icmp_seq=7 ttl=64 time=0.072 ms
64 bytes from 172.16.3.2: icmp_seq=8 ttl=64 time=0.108 ms
64 bytes from 172.16.3.2: icmp_seq=9 ttl=64 time=0.138 ms
64 bytes from 172.16.3.2: icmp_seq=10 ttl=64 time=0.108 ms
^C
--- 172.16.3.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9000ms
rtt min/avg/max/mdev = 0.070/0.109/0.146/0.023 ms
root@27b737a1a215:/#
```

15. Enter **Ctrl+C** to stop the pings and exit the container's bash shell. Type **Exit** to return to the **DMZ** server's bash shell.

```
root@27b737a1a215:/# ping 172.16.3.2
PING 172.16.3.2 (172.16.3.2) 56(84) bytes of data.
64 bytes from 172.16.3.2: icmp_seq=1 ttl=64 time=0.129 ms
64 bytes from 172.16.3.2: icmp_seq=2 ttl=64 time=0.070 ms
64 bytes from 172.16.3.2: icmp_seq=3 ttl=64 time=0.114 ms
64 bytes from 172.16.3.2: icmp_seq=4 ttl=64 time=0.098 ms
64 bytes from 172.16.3.2: icmp_seq=5 ttl=64 time=0.146 ms
64 bytes from 172.16.3.2: icmp_seq=6 ttl=64 time=0.111 ms
64 bytes from 172.16.3.2: icmp_seq=7 ttl=64 time=0.072 ms
64 bytes from 172.16.3.2: icmp_seq=8 ttl=64 time=0.108 ms
64 bytes from 172.16.3.2: icmp_seq=9 ttl=64 time=0.138 ms
64 bytes from 172.16.3.2: icmp_seq=10 ttl=64 time=0.108 ms
^C
--- 172.16.3.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9000ms
rtt min/avg/max/mdev = 0.070/0.109/0.146/0.023 ms
root@27b737a1a215:/# exit
exit
[root@pod-dmz ~]#
```

16. Close the Terminal window and continue to the next task. If a *Warning* window appears, click **Close Window**.



### 1.3 Map the Host Port to the Running Web Container and Access the Container using the Web Browser.

In this section, you will map the host port and run the container in detached mode. Mapping the host port will allow access to the container's nginx default web page using your client's browser.

1. Launch **Xfce** Terminal in the lower-left of the student Desktop.



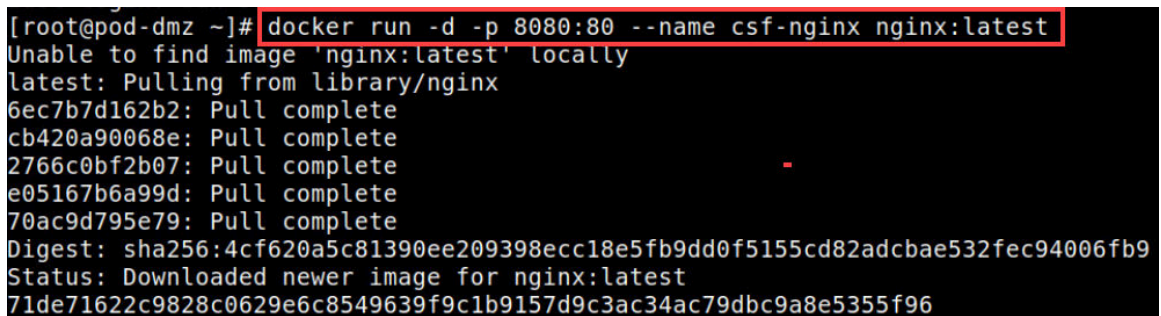
2. SSH to the *DMZ* by typing the command below. Use `Pa10Alt0!` for the password. If prompted, enter yes to continue connecting. Press **Enter**.

```
C:\home\lab-user> ssh root@192.168.50.10
```



3. Pull the `nginx:latest` container image and run the container in detached mode and then use this image to run a container in detached mode. The command will map the host port 8080 to your container's port 80. Type the command below.

```
[root@pod-dmz ~]# docker run -d -p 8080:80 --name csf-nginx nginx:latest
```



- View your newly created nginx container by typing the command below.

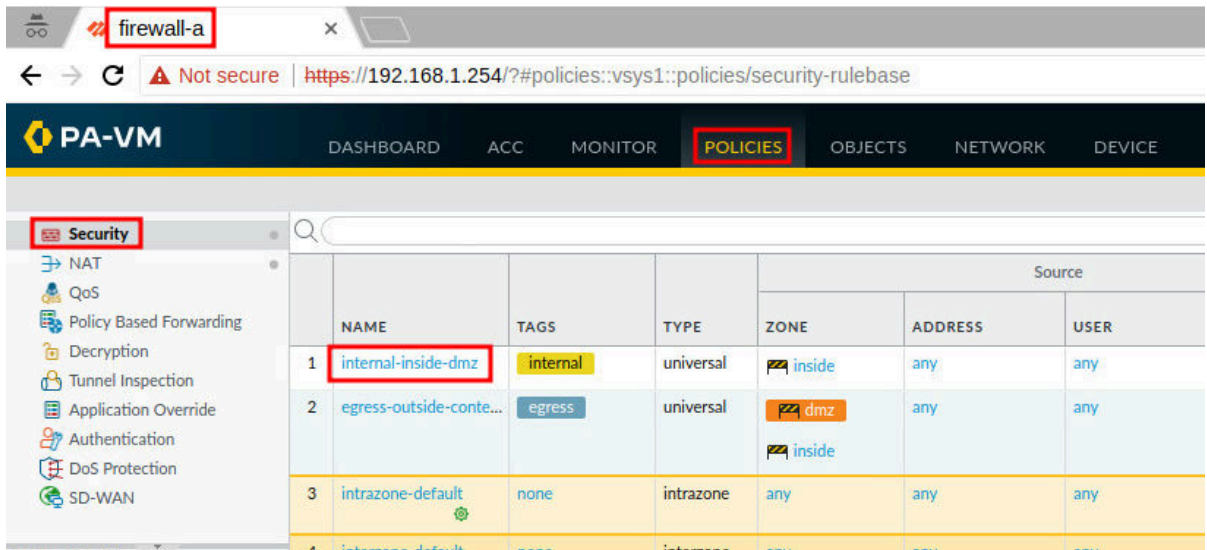
```
[root@pod-dmz ~]# docker ps
```

```
[root@pod-dmz ~]# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
71de71622c98	nginx:latest	"/docker-entrypoint. ...."	About a minute ago	Up About a minute
0.0.0.0:8080->80/tcp	csf-nginx			
27b737a1a215	ubuntu:latest	"/bin/bash"	17 minutes ago	Up 17 minutes
	csf-ubuntu2			
fde507414e2d	ubuntu:latest	"/bin/bash"	25 minutes ago	Up 25 minutes
	csf-ubuntu1			

```
[root@pod-dmz ~]#
```

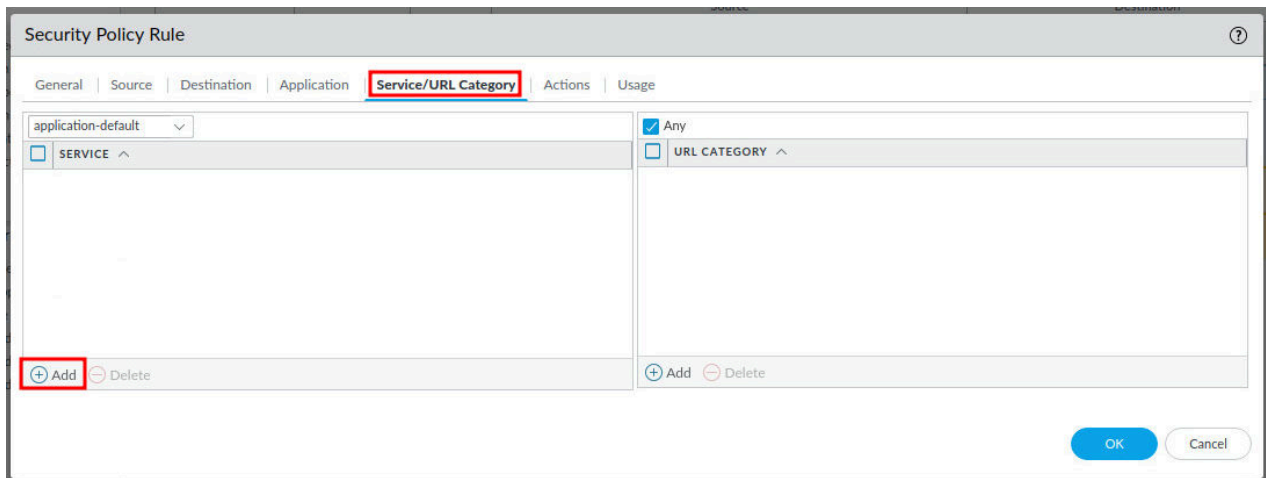
- Ensure that you are in the *firewall-a* interface. Navigate to **Policies > Security**. Click the **internal-inside-dmz** security policy.



The screenshot shows the PA-VM web interface for the 'firewall-a' instance. The 'POLICIES' tab is selected in the top navigation bar. On the left sidebar, 'Security' is highlighted. The main table lists security policies. The first policy, 'internal-inside-dmz', is highlighted with a red box. It has a tag of 'internal', type 'universal', and is associated with the 'inside' zone. The source address is 'any' and the user is 'any'.

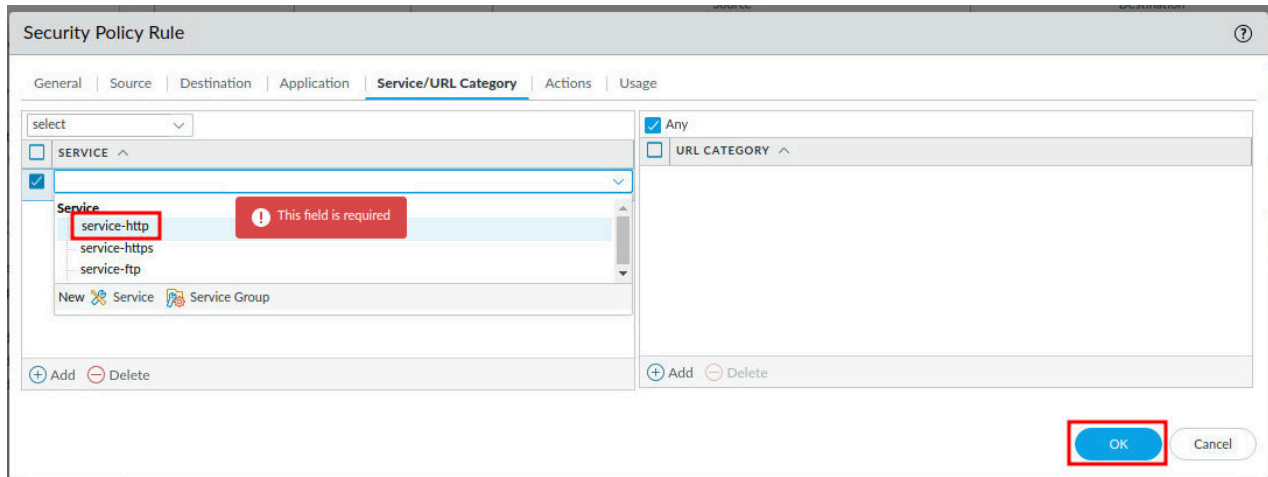
	NAME	TAGS	TYPE	ZONE	ADDRESS	USER
1	internal-inside-dmz	internal	universal	inside	any	any
2	egress-outside-conte...	egress	universal	dmz	any	any
3	intrazone-default	none	intrazone	any	any	any
4	interzone-default	none	interzone	any	any	any

- In the *Security Policy Rule* window, select **Service/URL Category**. Click **Add** in the *Service* pane.



The screenshot shows the 'Security Policy Rule' configuration window. The 'Service/URL Category' tab is selected. The 'SERVICE' pane on the left has an 'Add' button highlighted with a red box. The 'URL CATEGORY' pane on the right is empty. The 'Any' checkbox is checked in the 'URL CATEGORY' pane.

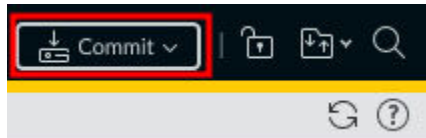
7. In the dropdown menu, select **service-http** and click **OK**.



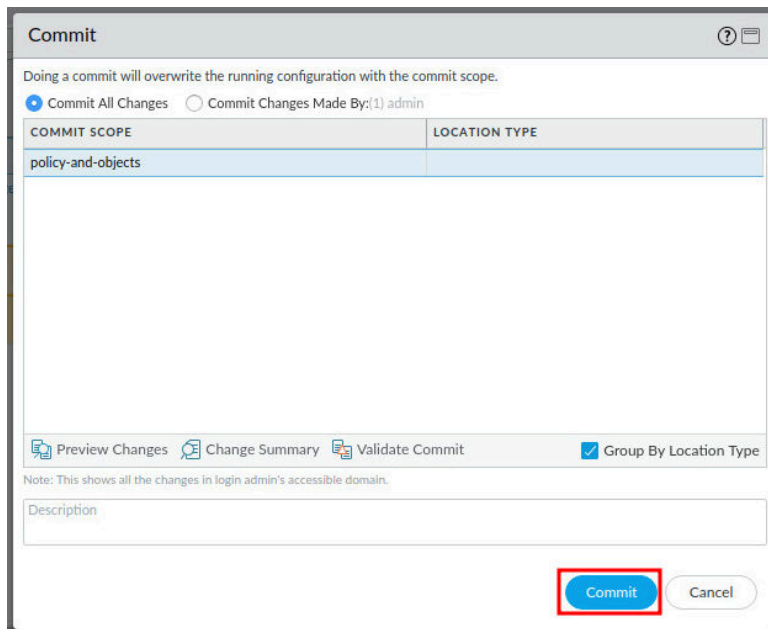
**Please  
Note**

Because you are doing this over port 80, you will change it to port 8080 / service-http so it will not be blocked.

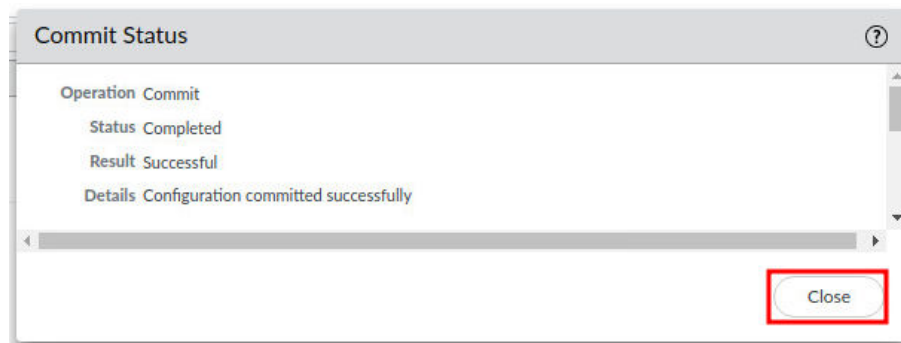
8. Click the **Commit** link located at the top-right of the web interface.



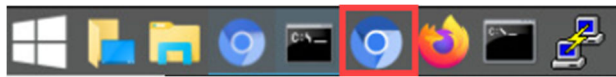
9. In the *Commit* window, click **Commit** to proceed with committing the changes.



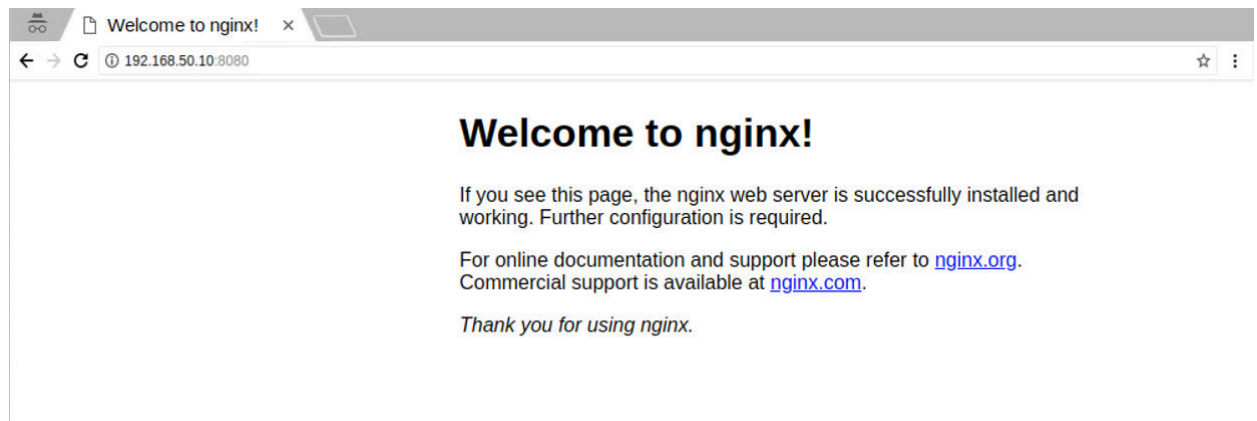
10. When the commit operation successfully completes, click **Close** to continue.



11. Launch a new *Chromium Web Browser* by clicking the **Chromium** icon on the lower-left of the student *Desktop*.



12. Access the *nginx* container's default web page by typing the `http://192.168.50.10:8080` in the *Chromium* browser address bar.



13. The lab is now complete; you may end your reservation.