Cloud Automation with FCA Lab Guide

Lab Objectives

The main objective of the Azure-lab guide is to demonstrate what was learned in the previous cloud courses. Using manual intervention or ARM templates, we discovered in previous cloud classes that there are many different ways to launch the VM series Palo Alto Firewalls in multiple cloud provider solutions.

This course will help teach other methods to launch the VM Series firewalls but instead using automation. The goal is to teach the benefits of automation and demonstrate the (FCA) tool's capabilities.

The Flexible Cloud Automation (FCA) tool was developed as a data driven model framework to address the orchestration needs of all major cloud providers that use Palo Alto Networks Next Generation VM Series Firewalls.

It is done by using a data driven model to agnostically drive global variable inputs to generate a custom cloud orchestration template. This is built not only around the VM Series firewalls but also around the cloud provider infrastructure as well.

- Lab 1 involves preparation work to build a virtual Docker container and use Github to prepare FCA to be deployed.

- Lab 2 involves using the (FCA) Cloud Automation tool to build a basic Azure environment and launch the Palo Alto Networks Firewall.

Lab Design

All Labs can be accomplished by using Docker and FCA Framework. FCA framework will build different topologies per cloud provider based on specified inputs created in the FCA virtual_networks and group_vars file.

Table of Contents

1. Lab: Using Docker and Github to prepare FCA tool.

1.1. Lab Objectives

- Learn how to prepare FCA off of Github.
- Understand Github and Git-core functionality
- Learn how to download and build docker containers related to FCA.
- Learn all the files in the FCA Framework to build your first Azure Topology.

1.2. Prepare and verify all lab files.

Important Note Please Read!

You will need permissions to the: https://github.com/PaloAltoNetworks/pan-fca upstream/master to download Dockerfile-FCA.

(You should already have this by Forking and Cloning upstream/master FCA Github repository. If not please contact administrator of Repo if not public!) If you have trouble, please ask the course instructor for assistance.

- Use Github to install FCA structure
- Use Docker to build, run and install infrastructure via container.
- Verify Prerequisites are loaded on local machine or Virtual machine which are Docker and Git hub.
- Verify access by visiting https://github.com/PaloAltoNetworks/pan-fca/ docs/Docker_instructions_Readme.md
- Verify files for Labs exist and are mounted properly in the Docker container.

1.3. Download Prepared Docker

Download Prepared Docker containers or build one off of the Github Cloned Fork.

```
AMSMACF1WWG8WL:pan-fca tostern$ ls -la
total 592
drwxr-xr-x  42 tostern  192360288    1344 Mar 11 12:58 .
drwxr-xr-x  12 tostern  192360288     384 Mar  7 10:25 ..
-rw-r--r--   1 tostern  192360288      77 Feb 27 00:25 .dockerignore
drwxr-xr-x  15 tostern  192360288     480 Mar 11 12:50 .git
drwxr-xr-x   5 tostern  192360288     160 Feb 27 00:25 .github
-rw-r--r--   1 tostern  192360288     645 Feb 27 00:25 .gitignore
drwxr-xr-x   4 tostern  192360288     128 Mar 11 12:01 .terraform
-rw-r--r--   1 tostern  192360288    1281 Mar  8 00:30 Dockerfile-Full
-rw-r--r--   1 tostern  192360288    1206 Mar 11 12:50 Dockerfile-Slim
-rw-r--r--   1 tostern  192360288     546 Feb  7 07:02 Jenkinsfile
-rw-r--r--   1 tostern  192360288   11357 Feb  7 07:02 LICENSE
-rw-r--r--   1 tostern  192360288     493 Feb  7 07:02 Makefile
-rw-r--r--   1 tostern  192360288    1529 Mar  8 00:30 README.md
-rw-r--r--   1 tostern  192360288     945 Feb  7 07:02 SUPPORT_POLICY
-rw-r--r--   1 tostern  192360288      65 Feb  7 07:02 ansible.cfg
-rw-r--r--   1 tostern  192360288     559 Mar  8 00:30 aws_vars.tf
drwxr-xr-x   3 tostern  192360288      96 Feb  7 07:02 bootstrap
-rw-r--r--   1 tostern  192360288    7563 Mar  8 00:30 configuration_push.yml
-rw-r--r--   1 tostern  192360288     543 Feb  7 07:02 destroy.yml
drwxr-xr-x   6 tostern  192360288     192 Feb 27 00:25 docs
drwxr-xr-x   4 tostern  192360288     128 Mar  1 04:53 filter_plugins
drwxr-xr-x   3 tostern  192360288      96 Mar  1 04:54 group_vars
drwxr-xr-x   6 tostern  192360288     192 Feb 27 00:25 images
drwxr-xr-x   5 tostern  192360288     160 Feb  7 07:02 iron-skillet
drwxr-xr-x  38 tostern  192360288    1216 Mar  8 00:30 library
-rw-r--r--   1 tostern  192360288    8997 Mar 11 12:51 main.tf
drwxr-xr-x   5 tostern  192360288     160 Feb  7 07:02 modules
-rw-r--r--   1 tostern  192360288     423 Mar 11 12:51 provider.yml
-rw-r--r--   1 tostern  192360288     453 Mar 11 12:50 provider.yml.example
-rw-r--r--   1 tostern  192360288     146 Mar 11 11:53 provider.yml.ttt
-rw-r--r--   1 tostern  192360288      57 Feb  7 07:02 requirements.txt
drwxr-xr-x  18 tostern  192360288     576 Mar  8 00:30 roles
drwxr-xr-x   5 tostern  192360288     160 Feb  7 07:02 sandbox_modules
```

1.4. Obtaining Container Image

1.4.1. Option #1: Pull the Dockerfile from the Docker Repository

Type the following command in your cli to download the Docker build:

```
docker pull panfca/tool:fca
```

1.4.2. Option #2: Obtaining Container Image from GitHub

Browse to the folder where your forked FCA is located see example in Chapter 1.3

1.5. Build from Dockerfile

When used Option #2 we have to build the Docker instance out of the Dockerfile. For this type the following command. The "." At the end of command below is really important!

```
docker build –t <StudentName> –f Dockerfile–Full .
```

1.6. Launch Docker Container and Custom Tag

Note: If you want to create your own image name you can tag it with a custom tag.

Example: docker tag panfca/tool:fca myfca

Then *"docker images"* command will list all of your docker images.

Verify that your image with your tag is present

Make sure you are in the root fca folder of the cloned repository or specify full path before mounting.

1.7. Start docker container and mount the root folder

```
docker run –v ${PWD}:/fca –it panfca/tool:fca
```

This mounts the local. /fca directory in the fca directory of the container, and launches container

Verify once container launches with ls then **cd** to ./fca mounted directory once inside the container to run playbook commands. See the example below:

```
AMSMACF1WWG8WL:panos-fca tostern$ docker run –v ${PWD}:/fca –it fca
root@56ad2dc1cbb4:/# cd fca
root@56ad2dc1cbb4:/fca#
```

Verify with the command “ls” that you are in the right folder. You have to see the following files at minimum:

- configuration_push.yml
- destroy.yml
- modules
- group_vars

```
root@56ad2dc1cbb4:/fca# ls
Dockerfile-FCA  SUPPORT POLICY  configuration_push.yml  images        modules              sandbox_modules                  terraform.tfstate.backup
Jenkinsfile     TO-DO           destroy.yml             inventory     provider.yml         schema_files                     test.yml
LICENSE         ansible.cfg     docs                    iron-skillet  provider.yml.example state_backup_2019-01-17-11-06-04 virtual_networks
Makefile        aws_vars.tf     filter_plugins          library       requirements.txt     templates                        yamllint.yml
README.md       bootstrap       group_vars              main.tf       roles                terraform.tfstate
root@56ad2dc1cbb4:/fca#
```

1.8. Removal of Docker images installed.

Once all the labs are completed you can run the following command that will destroy all docker containers and volumes. This is also useful if there is limited space or are running on a temporary bastion box.

```
docker container stop $(docker container ls -a -q) && docker system prune -a -f --volumes
```

*Docker images are important and used to maintain code support version levels without the need to install on local machines. This also can help speed deployment times fast if FCA is building the infrastructure on a customer environment.

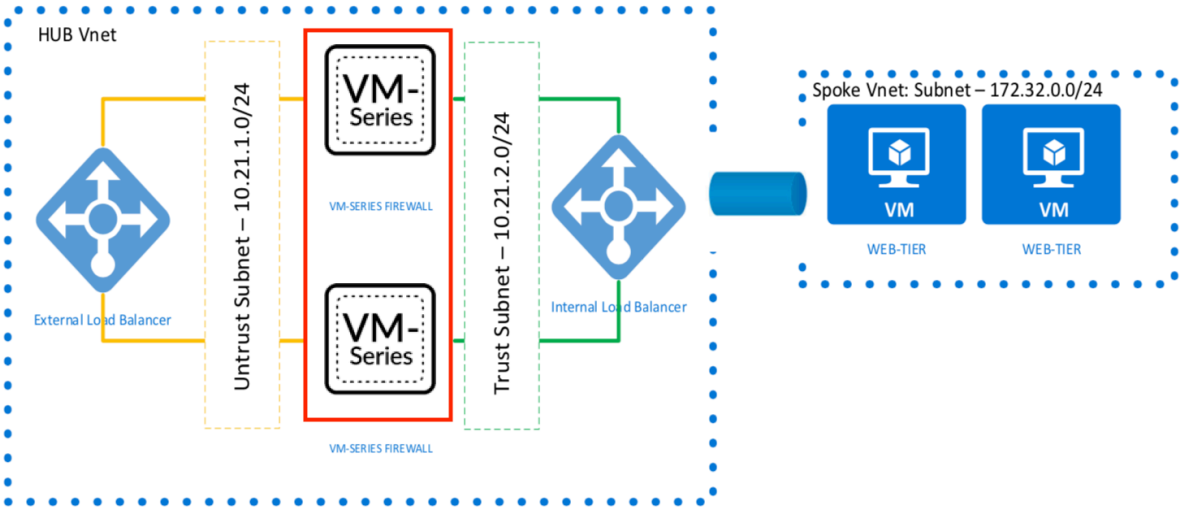
End of Lab #1

Following Labs #2, #3 #4 Lab will consist of preparing the files from Lab#1 objectives 1.7 and 1.8 for using specific parameters related to Instructor based Topology.

Every Code Snippets are only examples and don't provide you an 100% Solution to the Requirements of the Lab's.

2. Lab: Using FCA for Azure Cloud

2.1. Azure Lab Topology

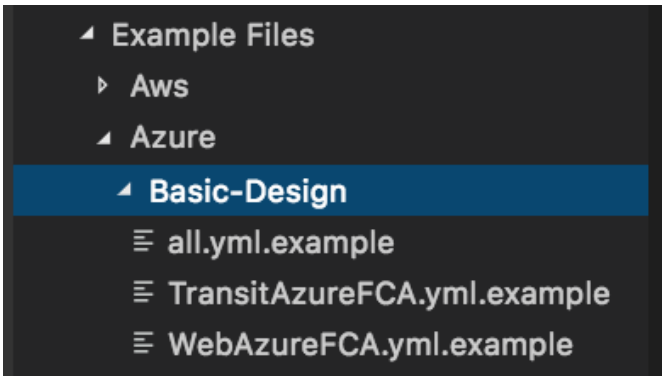


- 1 Transit VNet – 10.217. XXX..0/24
- 4 Transit Subnets – Management (10.217.XXX.64/27), Trust (10.217.XXX.32/27), Untrust (10.217.XXX.0/27), <StudentName>Net (10.217. XXX..96/27)
- 1 Spoke VNet – 172.17. XXX..0/24
- 2 Spoke Subnets – Webserver (172.17. XXX..0/26), DB (172.17. XXX..64/26).
- 1 UDR in Spoke
- 2 Firewalls (Transit VNet)
- 3 Interfaces per FW (Trust, Untrust and Management)
- Public IPs on the firewall
- 1 Public Load Balancer
    - Probe: TCP 22
    - Load Balancing: TCP 80
- 1 Private Load Balancer
    - Probe: TCP 22
    - Load Balancing: HA Ports
- Peering between Transit and Spoke
- 1 Test VM in the Webserver Subnet in Spoke RG

USE EVERYTIME THIS AS REFERENCE FOR YOUR CONFIGURATION

2.2. Virtual Networks files

Inside the "pan-fca" folder find the "Example Files" folder and copy and paste the two Azure example files under the "Basic-Design" folder inside the "Virtual Networks" folder.
Change the name of the two files to:
"Transit<StudentName>.yml"
"Web<StudentName>.yml"



The Filenames are important. The Filename will be later the name of your Resource Group.

2.3. Define Hub Resources

2.3.1. Create the Firewalls

Open the up the Tranist<StudentName>.yml file that you have created in the previous step.
Starting from the top, modify the necessary fields:

```
firewalls:
  - name: Firewall-Set-1
    vmcount: 1
    fwname: TSVMPAN-
    fwvmsize: Standard_D3_v2
    #https://knowledgebase.paloaltonetworks.com/KCSArticleDetail?id=kA10g000000ClD7CAK
    avsetname: "AzureAV1"
    fw_version: 8.1.0
    # 7.1.0 / 8.0.0 / 8.1.0 = latest
    fw_sku: byol
    # byol / bundl1 / bundle2
    Username: tstern
    Password: PaloAlto1234!
    attachtrustpool: "yes"
    attachuntrustpool: "yes"
    lbnameuntrust: External-LB
    lbnametrust: Internal-LB
```

name:	This is Generic name not visible in Azure.	Add your initials to make the name unique.
vmcount:	Select how many Firewalls you want to deploy in the AVSet.	2 are required to complete the lab.
fwvmsize:	Here you can define which VM Size you want to.	No changes needed.
avsetname:	Here you define the name of the Avaibility-Set.	Add your initials to make the name unique.
fw version:	Here you can define the PANOS Version (latest = 9.0.0)	Change it to 8.1.0.
Username:	Define a username (min. 5 characters)	Change as indicated
Password:	Define a Password (min 12 characters, min 1 capital and min 1 number)	Change as indicated
Attach Untrust/trustpool	Here you can define if you want to have your FW assigned to an Loadbalancer Backendpool.	Default = „NO“
Lbname[trust/untrust]	Here you define to which Load Balancer the FW hast o be assigned	Change it to the name oft he LB when you changed it.

*****Don’t use the Default Values for Username and Password*****

2.3.2. Create a Availability Set

In this part you create the Availability Set for your pair of firewalls.

```
availabilty_set:
  - avsetname: AzureAV1
```

name:	This is Generic name visible in Azure.	Add your initials to make the name unique.
avsetname:	Define here your name oft he AV-Set	Change it to your value

2.3.3. Creating Load Balancers

In this part you can configure Load Balancer. The script will by default create a Standard Load Balancer. It is not needed to define an “Private” and “Public Load Balancer”.

```
load_balancers:
  - name: External-LB
    fename: Untrust
    bename: Untrust
    floating_ip: false
    type: public
    lbrulename: HTTP-80
    lbrulefrontport: 80
    lbruleprotocol: tcp
    lbrulebackport: 80
    lbprobename: TCP-22
    lbruleprobeport: 22

  - name: Internal-LB
    fename: Trust
    bename: Trust
    floating_ip: true
    type: private
    lbrulename: HA
    lbrulefrontport: 0
    lbruleprotocol: All
    lbrulebackport: 0
    lbprobename: ssh
    lbruleprobeport: 22
```

Name:	Defines the name of the Loadbalancer	Add your initials to make the name unique.
FeName:	Defines the name of the Frontendpool	No changes needed.
BeName:	Defines the name of the Backendpool	No changes needed.
Type:	Defines which type you want to have (private or public)	No changes needed.
LBRuleName:	Defines a name for the Loadbalancer rule	No changes needed.
LBRuleFrontEndPort:	Defines the port of the Frontendpool	No changes needed.
LBRuleFrontBackndPort:	Defines the port of the Backendpool	No changes needed.
LBRuleProtocol:	Defines the Loadbalancer Probe Protocol	No changes needed.
LBProbeName:	Defines the LoadBalancer Probe name	No changes needed.
LBRuleProbePort:	Defines the Loadbalancer Probe port	No changes needed.
Floating_ip	This defines if you want use floating ip for your LB rules	No changes needed

In this Section is not really needed to change the default values. But you can change it to make it for better.

2.3.4. Creating Virtual Networks

In this Section you define the Virtual Network (VNet) and your Subnets for the Hub Resource Group (RG).

Please Replace the third CIDR Octet with your Student number provided by the Instructor.

```
vnet_network:
  name: Transit-Azure
  network: "10.217.XXX.0/24"
  peers:
    -Web<StudentName>
  subnet:
  - name: Management
    network: "10.217.XXX.64/27"
  - name: Trust
    network: "10.217.XXX.32/27"
  - name: Untrust
    network: "10.217.XXX.0/27"
```

name:	Defines the name of the Virtual Network	Add your initials to make the name unique.
network:	Defines the VNet CIDR	Enter the number provided by the instructor
peers:	Defines the Peer config	Type the filename of your spoke but without **".yml"**
subnet:	Defines the subnets of the VNet	Enter the number provided by the instructor

IMPORTANT the first Network has to be the Firewall Network
1 Subnet = Management; 2 Subnet = Trust; 3 Subnet = Untrust

2.3.5. Creating Security Groups

In this section you can define your Network Security Groups (NSG) what you can then later attach to the needed interfaces. By default, is to every interface an “allow all” NSG applied to not having any issues in the beginning.

```yaml
security_groups:
  - name: Outbound allow
    priority: 1000
    direction: inbound
    action: allow
    src_ip: 0.0.0.0
    src_port: any
    dst_port: any
    dst_network: 0.0.0.0
    protocol: any
```

name:	Defines the name of the NSG	Add your initials to make the name unique.
priority:	Defines the Priority of the Rule	No changes needed.
direction:	Defines the direction (inbound or outbound)	No changes needed.
action:	Defines the action (allow or deny)	No changes needed.
src_ip:	Defines the Source IP	No changes needed.
src_port:	Defines the Source port	No changes needed.
dst_port:	Defines the destination port	No changes needed.
dst_network:	Defines the destination network	No changes needed.
protocol:	Defines the protocol (tcp, udp or any)	No changes needed.

2.3.6. Create a Resource Group

In this section you can define your Ressource Group (RG) where your objects are deployed

```
resource_group:
  - resource_group_name: Transit_RG
```

name:	This is Generic name visible in Azure.	Add your initials to make the name unique.
Resource group name:	Defines the name of the RG	Change it to your value

2.4. Define Spoke Resources

2.4.1. Creating Test Host VM

Open the up the „Web<StudentName>.yml“ file that you have created in the previous step.
In the top of the file you will find the Example for the Testhost VM. Starting from the top, modify the necessary fields.

```
testhost:
- name: Testhost
  hostname: Cloud-Test
  username: creator
  password: PaloAto123456789
  dnsname: ubuntutestvm
```

name:	This is Generic name visible in Azure.	Add your initials to make the name unique.
hostname:	Defines the name of the Testhost	No changes needed.
username:	Defines your username	Change as indicated
password:	Defines your password (min. 12 characters)	Change as indicated
dnsname:	Defines the DNS name of the host ***(only use lower case)***	Add your initials to make the name unique.

*****Don’t use the Default Values for Username and Password*****

2.4.2. Creating Security Groups

In this section you can define a Security Group (NSG) when you don't define one is no NSG applied.

```
security_groups:
  - name: Outbound allow
    priority: 100
    direction: inbound
    action: allow
    src_ip: 1.1.1.1
    src_port: any
    dst_port: any
    dst_network: 0.0.0.0/0
    protocol: any
```

name:	here you define the name of the NSG	Add your initials to make the name unique.
priority:	here you define the Priority of the Rule	No changes needed.
direction:	here you define the direction (inbound or outbound)	No changes needed.
action:	here you define the action (allow or deny)	No changes needed.
src_ip:	here you define the Source IP	Change it to 0.0.0.0/0 or your PIP
src_port:	here you define the Source port	No changes needed.
dst_port:	here you define the destination port	No changes needed.
dst_network:	here you define the destination network	No changes needed.
protocol:	here you define the protocol (tcp, udp or any)	No changes needed.

2.4.3. Creating Virtual Network

In this Section you define the Virtual Network (VNet) and your Subnets for the Spoke Resource Group (RG). There is now default Value available.

Please Replace the third CIDR Octet with your Student number provided by the Instructor.

You have to type filename of the Hub in the peer section (case sensitive).

```
vnet_network:
  name: Web-Azure
  peers:
    - Transit<StudentName>
  network: "172.16.XXX.0/24"
  subnet:
  - name: Webserver
    network: "172.16.XXX.0/26"
  - name: DB
    network: "172.16.XXX.0/26"
  - name: Test
    network: "172.16.XXX.0/26"
```

name:	Defines name of the Virtual Network	Add your initials to make the name unique.
network:	Defines the VNet CIDR	Enter the number provided by the instructor
peers:	Defines the name of peer hub config.	You have to type the filename of your HUB without ".yml"
subnet:	Defines the subnets of the VNet	Enter the number provided by the instructor

2.4.4. Create Route Table

The Route table in this Section is needed to forward the traffic from the Hub to the Transit.

```
route_tables:
  - name: web-test
    routes:
      - cidr: 0.0.0.0/0
        name: VirtualAppliance
        gateway: 1.1.1.1
```

name:	Defines the name of the Route Table (UDR)	**No changes needed.**
cidr:	Defines the CIDR of your route	**No changes needed.**
name:	Defines the type of nexthop type.	**No changes needed.**
gateway:	Defines the gateway ip of the "VirtualApplinace"	**No changes needed. (Placeholder)**

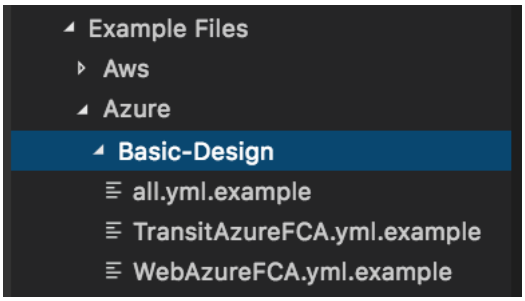
The value "1.1.1.1" under gateway is only a placeholder and we will change this value later in the Azure Porta

When you done all your changes to the files above, check if you had saved them.

2.4.5. Group Vars file

You can find the all.yml.example under the **„Examples -> Azure -> Basic-Desing“** folder.

Copy and Paste the all.yml.example file into the **„group_vars -> all“** folder and remove the **“.example”**.



This file is where a lot of your Firewall configuration will take place. This is Ansible driven, but you will be able to set your security policy rules, routes and admin username and passwords here. See the next Steps.

Important. By default, is the Iron-Skillet loaded in the beginning to the Firewalls
Here is the Link where you can see what is loaded.
https://github.com/PaloAltoNetworks/iron-skillet/blob/panos_v8.0/loadable_configs/sample-cloud-Azure/panos/iron_skillet_panos_full.xml

2.5. Set Parameters in all.yml file

Under the section **"credentials"** type the username and password from the firewall what you have defined before in the **"Transit<StudentName>.yml".** If you haven't defined this, let the values in the **"all.yml"**. These are the default values.

```
---

ansible_python_interpreter: "python"
location: "{{ cloud_provider_location }}"
credentials:
  azure:
    username: 'fwadmin'
    password: 'Paloalto123'

username: "{{ credentials[cloud_provider]['username'] }}"
password: "{{ credentials[cloud_provider]['password'] }}"
```

2.5.1. Include Networks

You have to type under virtual_networks the filenames, which you have created under the Chapter **"Virtual Networks files".** You must remove the **'.yml'** as you can see in the example below.

```
virtual_networks:
  - Transit<StudentName>
  - Web<StudentName>
```

2.5.2. Firewall Configuration

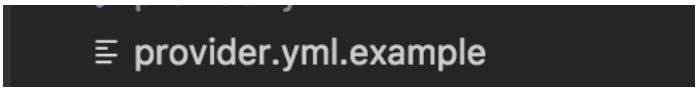
Everything below the "virtual_networks" section is a part of the Firewall Configuration. You can use almost everything from this example. Review all CIDR's that it matches to your update Student CIDR.

- Create Management Profile = cm_mgmt_profile
- Configure Interfaces = cm_interface:
- Create Address Objects = cm_object_address:
- Create Service Objects = cm_object_service:
- Create Service Groups = cm_object_service_group:
- Create Address Groups = cm_object_address_group:
- Create Application Groups = cm_object_app_group:
- Configure NTP = cm_ntp:
- Configure DNS = cm_dns:
- Configure Panorama = cm_panorama1:
- Create Firewall accounts = cm_fw_user:
- Create Security Rules = cm_security_rule:
- Create NAT Rules = cm_nat_rule: - Edit NAT to the appropriate CIDR
- Create Static Routes = cm_panos_static_route: - Edit routes to the Appropriate CIDR
- Activate Firewall = cm_panos_lic:
    - Example file is commented out. remove hash, continue steps below, after the setup is complete and you receive the licensing error, add the hash and run again config again.
- Update Firewall = cm_panos_software:
    - Commented out and not required to complete the lab.

Important. Don't use the "cm_panos_lic" module. This module occurs an error. We will provide you during the class all needed information to fix this!

2.6. Cloud Provider Information

In the root folder structure, you can find the “provider.yml.example” file, clone this file and remove the .example extensions



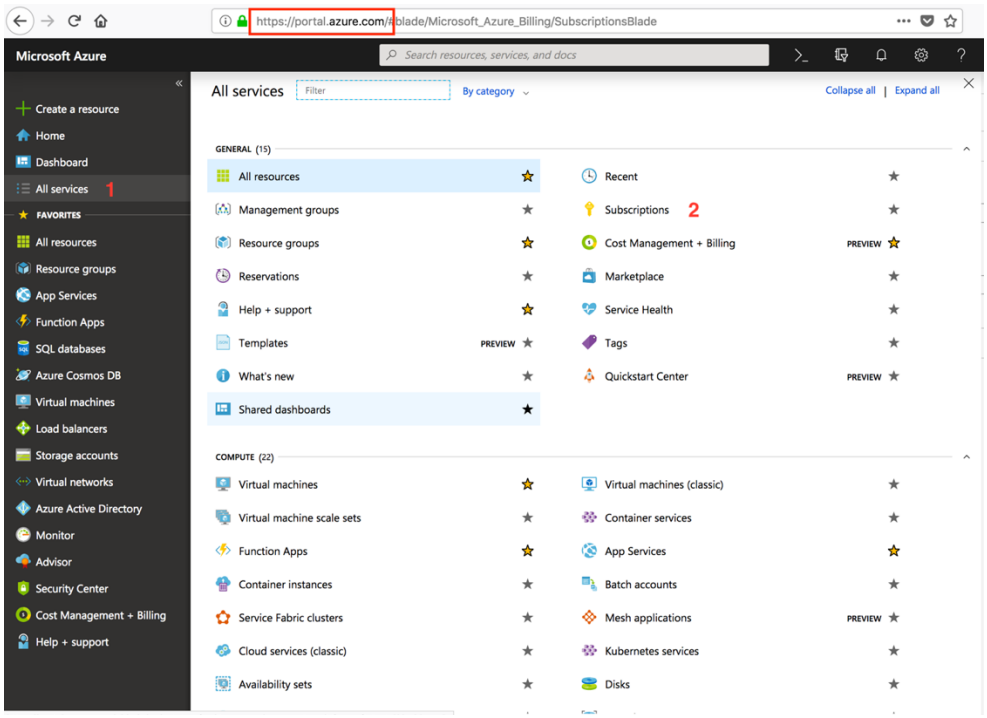
Inside this file you should delete all AWS related things.
As next update the fields as you can see below:

```
## AZURE EXAMPLE
cloud_provider: azure
cloud_provider_location: "north europe"
azure_subscription_id: "d47XXXXX-XXXX-XXXX-XXXX-da72XXXXXXX"

#Operating System used to select go binary type
go_os: "ubuntu"
```

If you want to change your “cloud_provider_location” look at this this link https://azure.microsoft.com/en-us/global-infrastructure/locations/ and replace “north Europe” with you location.

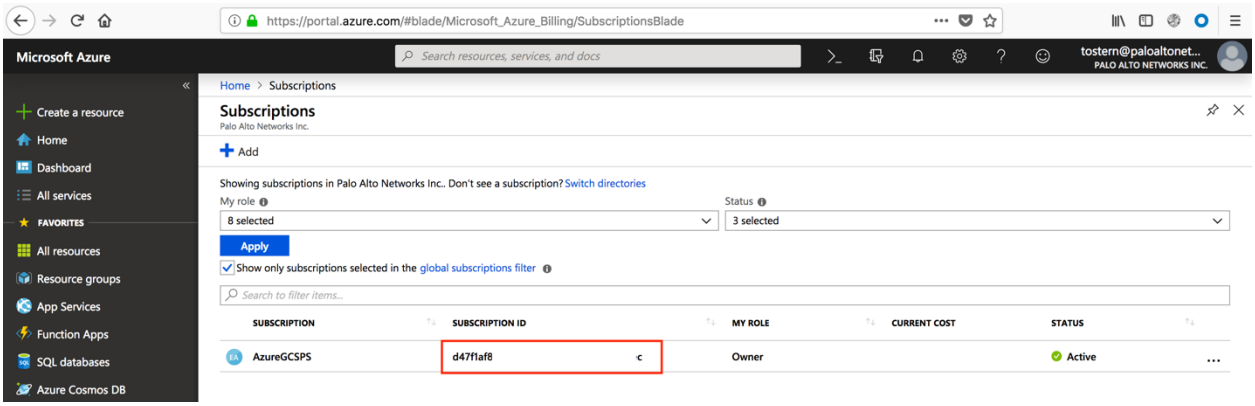
You can find your **“azure_subscription_id”** in your Azure Portal. See below:



Go to https://portal.azure.com and sign in with your Palo Alto Credentials. When you have problems with that contact please any of the Instructors.

In the click on **"All services"** and then on "**Subscriptions**".

Copy as next the Subscription ID and paste it in the **"azure_subscription_id"** filed in the **"provider.yml"** file



2.7. Connect to Docker Container and to the Azure Portal

When you have defined everything in related file above we can then start to connect to the Azure Portal and the Docker container

2.7.1. Connect to Docker Instance

We are now going to run the master configuration push from your Docker container. For that connect at first to your Docker instance what you have created in the Lab 1.

1. Go to the location where you have stored the FCA GitHub content (see example below)

```
AMSMACF1WWG8WL:panos-fca tostern$ pwd
/Users/tostern/Documents/GitHub/panos-fca
AMSMACF1WWG8WL:panos-fca tostern$
```

2. Type the following command to run the Docker Container

```
docker run -v $(PWD):/fca -it panfca/tool:fca
```

3. If, you have defined another tag instead fca, replace please

```
Docker run –v $(PWD):/fca –it  panfca/tool:*your tag*
```

4. After running this command, you should be connected to your Docker instance and in the root folder of the container. Type “ls -la” to verify that you can see the “ fca” folder.

```
root@2e98edd7bbbd:/# ls -la
total 72
drwxr-xr-x   1 root root 4096 Jan 21 10:29 .
drwxr-xr-x   1 root root 4096 Jan 21 10:29 ..
-rwxr-xr-x   1 root root    0 Jan 21 10:29 .dockerenv
drwxr-xr-x   1 root root 4096 Dec  3 18:08 bin
drwxr-xr-x   2 root root 4096 Jun 14  2018 boot
drwxr-xr-x   5 root root  360 Jan 21 10:29 dev
drwxr-xr-x   1 root root 4096 Jan 21 10:29 etc
drwxr-xr-x  40 root root 1280 Jan 21 09:46 fca
drwxr-xr-x   2 root root 4096 Jun 14  2018 home
drwxr-xr-x   1 root root 4096 Dec  3 18:04 lib
drwxr-xr-x   2 root root 4096 Nov 12 00:00 lib64
drwxr-xr-x   2 root root 4096 Nov 12 00:00 media
drwxr-xr-x   2 root root 4096 Nov 12 00:00 mnt
drwxr-xr-x   2 root root 4096 Nov 12 00:00 opt
dr-xr-xr-x 201 root root    0 Jan 21 10:29 proc
drwx------   1 root root 4096 Dec  3 18:08 root
drwxr-xr-x   1 root root 4096 Dec  3 18:04 run
drwxr-xr-x   1 root root 4096 Dec  3 18:04 sbin
drwxr-xr-x   2 root root 4096 Nov 12 00:00 srv
dr-xr-xr-x  13 root root    0 Jan 21 10:29 sys
drwxrwxrwt   1 root root 4096 Dec  3 18:08 tmp
drwxr-xr-x   1 root root 4096 Nov 12 00:00 usr
drwxr-xr-x   1 root root 4096 Nov 12 00:00 var
root@2e98edd7bbbd:/#
```

5. As next type “cd fca” to access the FCA folder and you should be able to see all folders and files of the fca repo.

```
root@2e98edd7bbbd:/# cd fca
root@2e98edd7bbbd:/fca# ls -la
total 312
drwxr-xr-x 40 root root   1280 Jan 21 09:46 .
drwxr-xr-x  1 root root   4096 Jan 21 10:29 ..
-rw-r--r--  1 root root     77 Nov 26 15:33 .dockerignore
drwxr-xr-x 16 root root    512 Jan 21 10:18 .git
drwxr-xr-x  5 root root    160 Jan 15 14:30 .github
-rw-r--r--  1 root root    641 Jan 17 18:38 .gitignore
drwxr-xr-x  4 root root    128 Jan 17 11:06 .terraform
-rw-r--r--  1 root root   1284 Jan 17 18:43 Dockerfile-FCA
-rw-r--r--  1 root root   1162 Jan 18 06:41 Dockerfile-FCA-slim
-rw-r--r--  1 root root    546 Nov 26 15:33 Jenkinsfile
-rw-r--r--  1 root root   1075 Nov 26 15:33 LICENSE
-rw-r--r--  1 root root    493 Nov 26 15:33 Makefile
-rw-r--r--  1 root root   1081 Dec  4 21:32 README.md
-rw-r--r--  1 root root    984 Nov 26 15:33 SUPPORT POLICY
-rw-r--r--  1 root root     39 Jan 15 15:00 TO-DO
-rw-r--r--  1 root root     65 Nov 26 15:33 ansible.cfg
-rw-r--r--  1 root root    149 Nov 26 15:33 aws_vars.tf
-rw-r--r--  1 root root   6123 Jan 18 13:25 configuration_push.yml
-rw-r--r--  1 root root    543 Nov 26 15:33 destroy.yml
drwxr-xr-x  6 root root    192 Dec 16 06:45 docs
drwxr-xr-x  5 root root    160 Dec  3 23:07 filter_plugins
drwxr-xr-x  3 root root     96 Jan 17 18:36 group_vars
drwxr-xr-x  8 root root    256 Dec 12 07:59 images
-rw-r--r--  1 root root     53 Dec  3 23:08 inventory
drwxr-xr-x 33 root root   1056 Jan 17 12:03 library
-rw-r--r--  1 root root   4873 Jan 21 09:46 main.tf
drwxr-xr-x  4 root root    128 Nov 26 15:33 modules
-rw-r--r--  1 root root    146 Jan 21 09:06 provider.yml
```

We are now ready to go to the next step, if you are can see the folders above. In case you cannot see it review the steps above or consult any Instructor.

2.7.2. Connect to the Azure Portal

Now we can connect to the Azure Portal, when you are successful connect to your Docker Container.

In your Docker instance type the following command to connect to the Azure Portal

```
az login –u username@paloaltonetworks.com
```

See the two different behaviors below:

1. A browser window is opening automatically, and you have to authenticate yourself with username and password. **OR**
2. In the command line you get an information back to go to the page https://microsoft.com/devicelogin and authenticate with a device code and your username and password. See the example below:

```
root@2e98edd7bbbd:/fca# az login
To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code D     JT to authenticate.
[
  {
    "cloudName": "AzureCloud",
    "id": "d47f1                              3ec",
    "isDefault": true,
    "name": "AzureGCSPS",
    "state": "Enabled",
    "tenantId": "66                                    ",
    "user": {
      "name": "tostern@paloaltonetworks.com",
      "type": "user"
    }
  }
]
root@2e98edd7bbbd:/fca#
```

You can use the following command in the cli, when you know your username and password. **"az login -u <username>"**. See the example below

```
root@2e98edd7bbbd:/fca# az login -u tostern@paloaltonetworks.com
Password:
[
  {
    "cloudName": "AzureCloud",
    "id": "d4                                        c",
    "isDefault": true,
    "name": "AzureGCSPS",
    "state": "Enabled",
    "tenantId": "6                                        e",
    "user": {
      "name": "tostern@paloaltonetworks.com",
      "type": "user"
    }
  }
]
root@2e98edd7bbbd:/fca#
```

2.7.3. Push Configuration to the Azure Cloud

By now you should be happy with your configurations in the files above and you should be successful connected to Azure Portal and your Docker Instance.

Now we can start with the deployment of the Azure environment and the Firewall configuration, when the above criteria are right.

In your Docker instance the following command

```
Ansible-playbook configuration_push.yml
```

The deployment and the configuration could take **5 between 20 minutes** when you have not deployed more than 4 Firewalls.

```
TASK [CREATING azure TEMPLATE] ************************************************************************************************************************************************
changed: [localhost]

TASK [RUN FMT FOR azure] ******************************************************************************************************************************************************
changed: [localhost]

TASK [include_role : terraform] ***********************************************************************************************************************************************

TASK [terraform : EXECUTE PLAN FOR AZURE] *************************************************************************************************************************************
included: /fca/roles/terraform/tasks/azure.yml for localhost

TASK [terraform : EXECUTE TERRAFORM PLAN] *************************************************************************************************************************************
ok: [localhost]

TASK [GET CREATED FIREWALLS FROM TERRAFORM OUTPUT] *****************************************************************************************************************************
changed: [localhost]

TASK [USE TERRAFORM OUTPUT STORE TEMP VALUES] **********************************************************************************************************************************
ok: [localhost]

TASK [ADD HOSTS FROM TF OUTPUT TO FIREWALL GROUP] ******************************************************************************************************************************
```

When the Script is finish it will commit the configuration and you should see the following output.

```
PLAY RECAP ********************************************************************************************************************
VMPAN-1                    : ok=36   changed=18   unreachable=0    failed=0
localhost                  : ok=29   changed=4    unreachable=0    failed=0
```

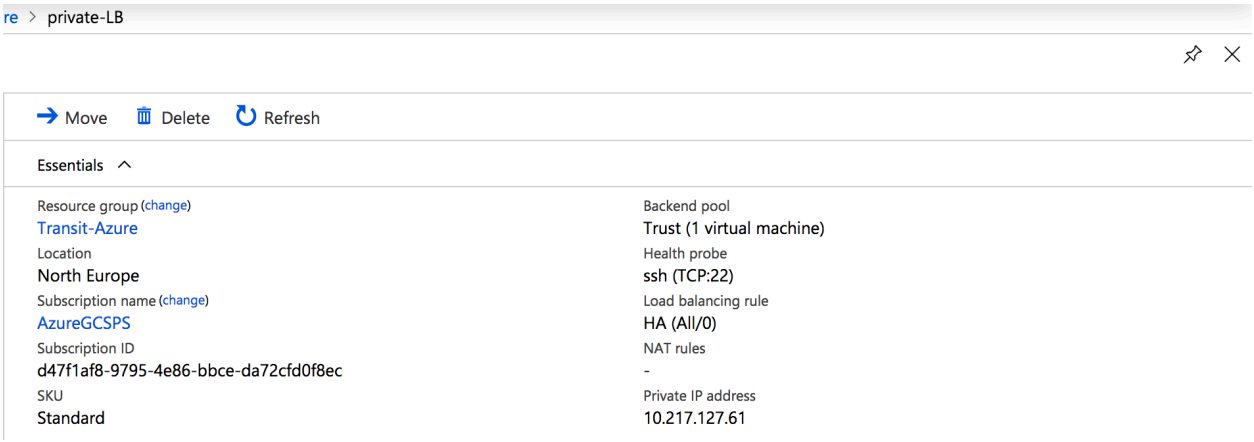
If you don't get the following output and the script is failing, please contact the Session instructor.

2.8. Review configuration and adopt some manual Changes

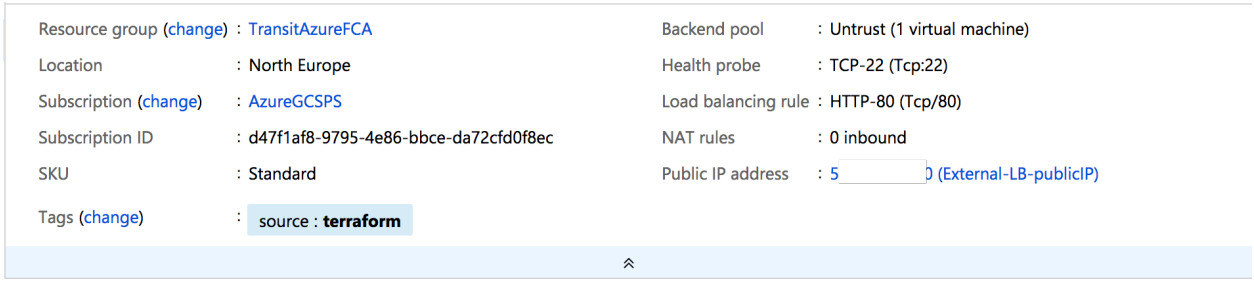
2.8.1. Review Azure Private/Public Load Balancer

The Overview of the Load Balancers should look like the following examples:

Private Load Balancer



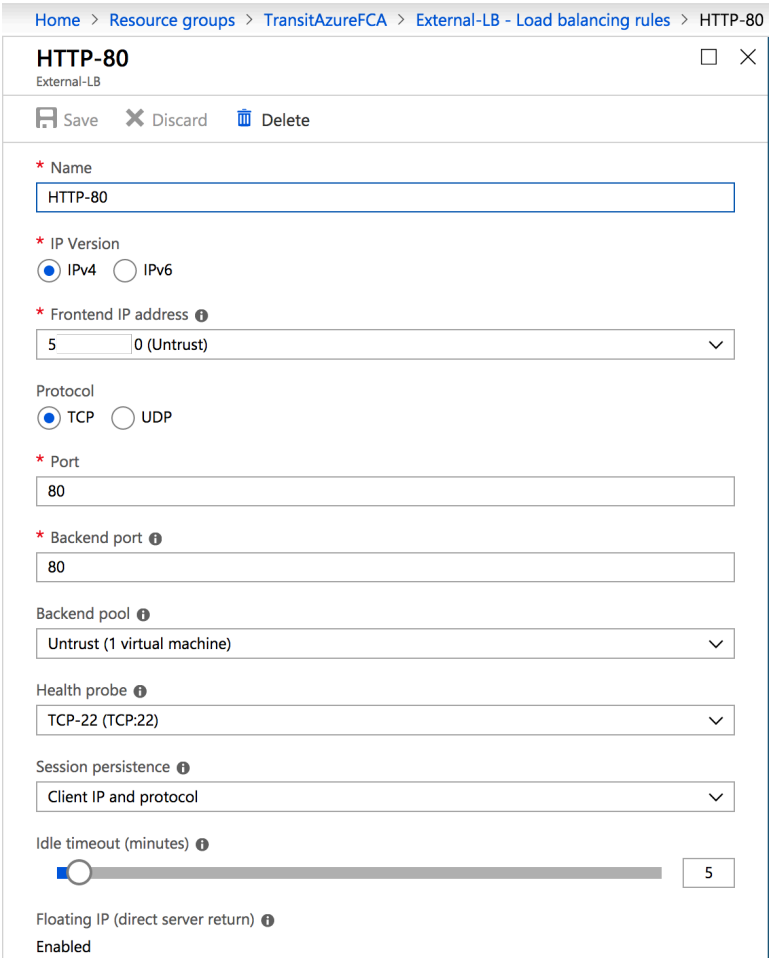
Public Load Balancer



Please take a note of the Public IP of the Load Balancer. This IP is needed in later step.

2.8.2. Review Azure Load Balancing Rule

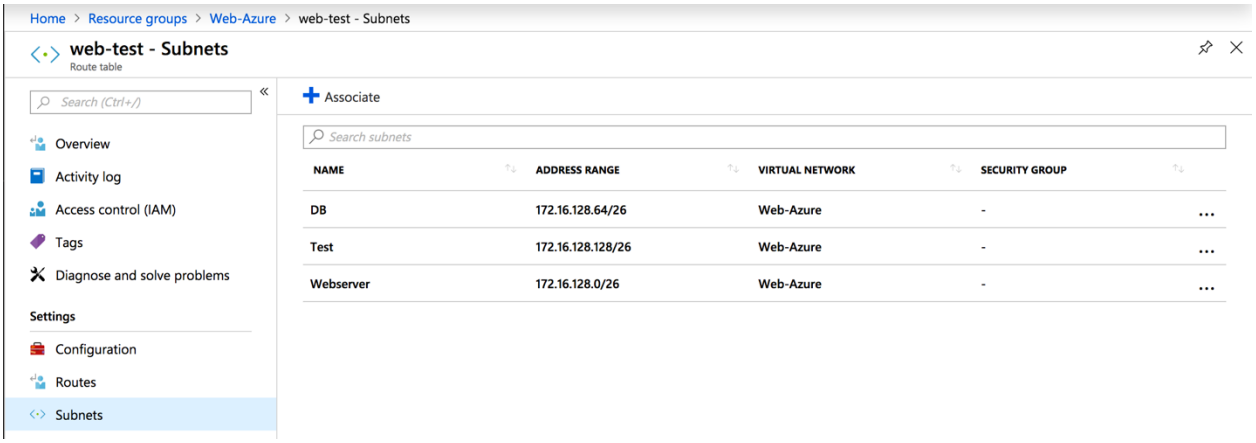
The FCA script will create for you Load Balancing Rule. Review that the rule is in place. This rule will forward all traffic on the port TCP 80.



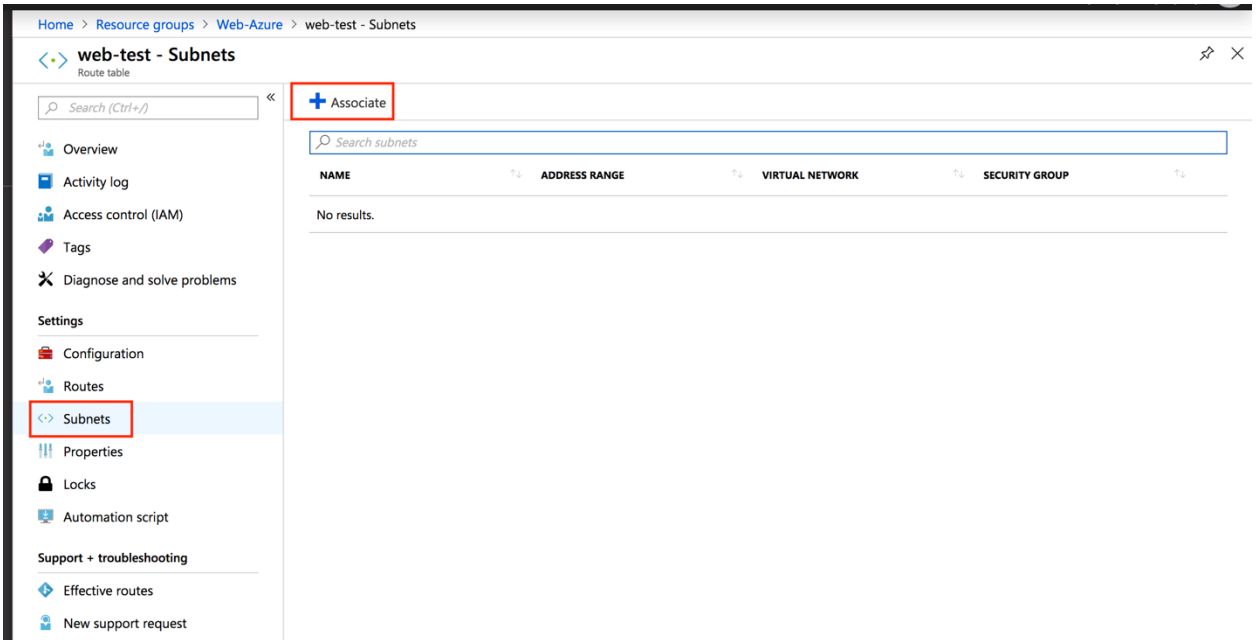
This Rule is needed that the traffic is forwarded to the Web-Server.

2.8.3. Review Azure Route Tables

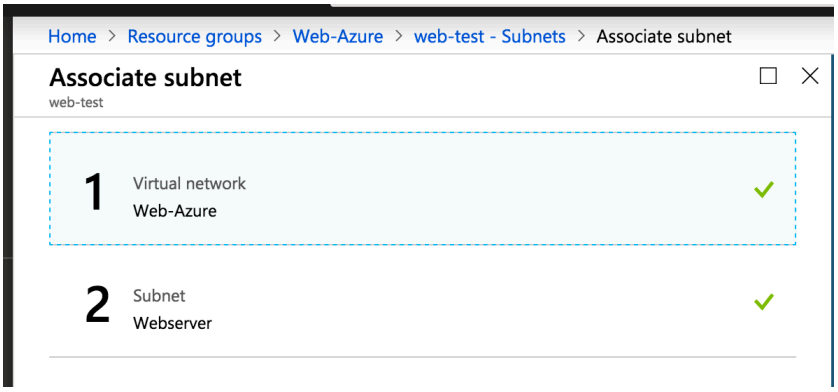
Go to your Spoke Resource Group and verify there if all Subnets are Associated to the route table. See example below.



If the Subnets are not associated to the route table, do this by yourself. Click on "Associate".



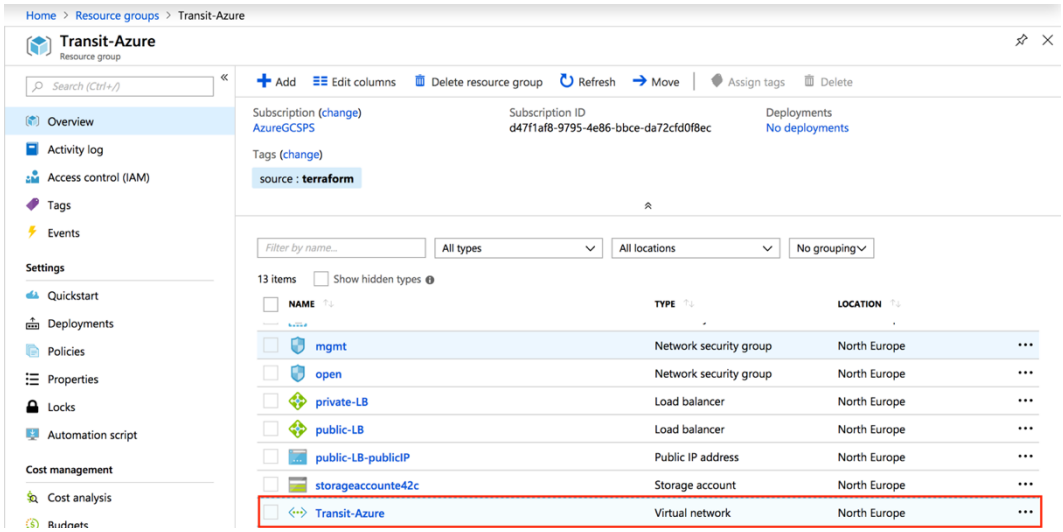
Associate all related Subnets to this route table. Repeat this step so long if you have added all subnets.



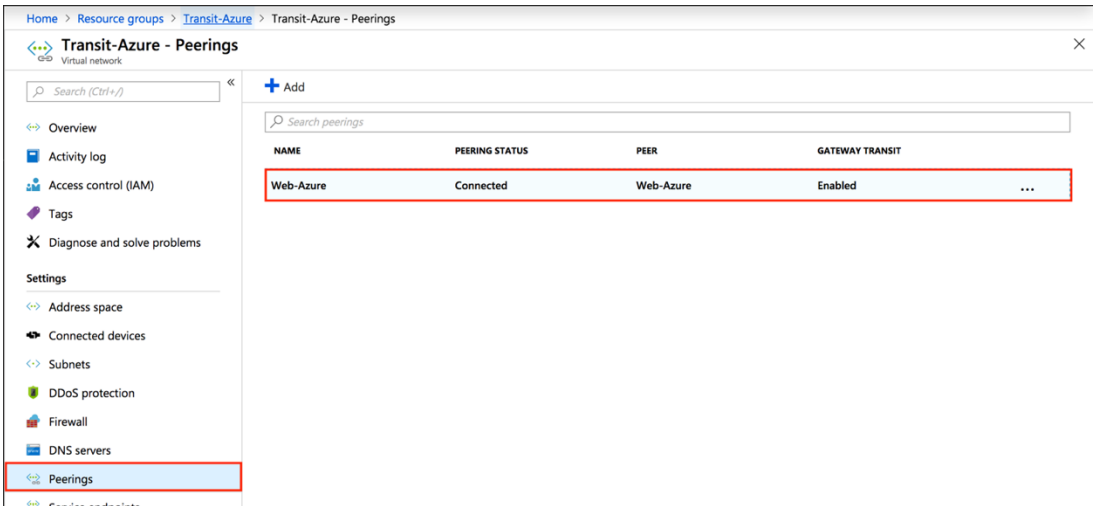
This is not really needed that the traffic is passing, but it is needed to use “effective routes”

2.8.4. Review Azure Virtual Networks

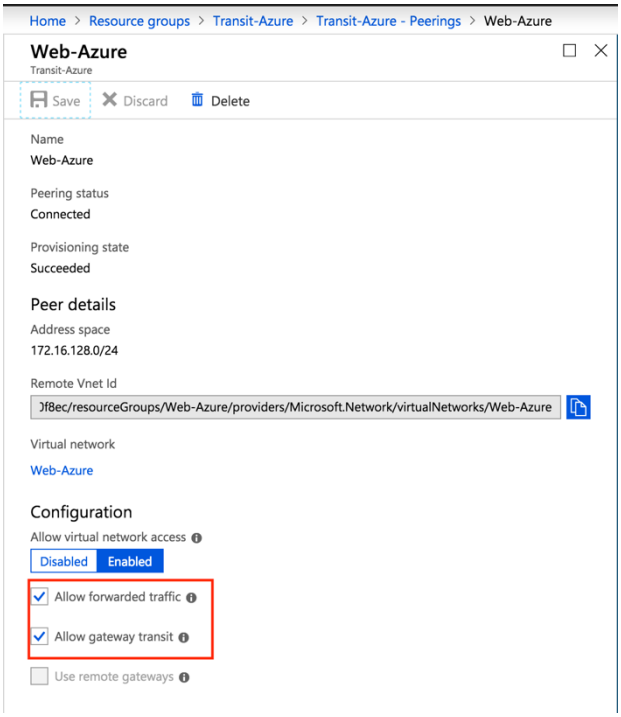
In this Part we have to review does the VNet Peering in both Resource Group (RG) are right. At first check the Virtual Network (VNet) in your Transit RG.



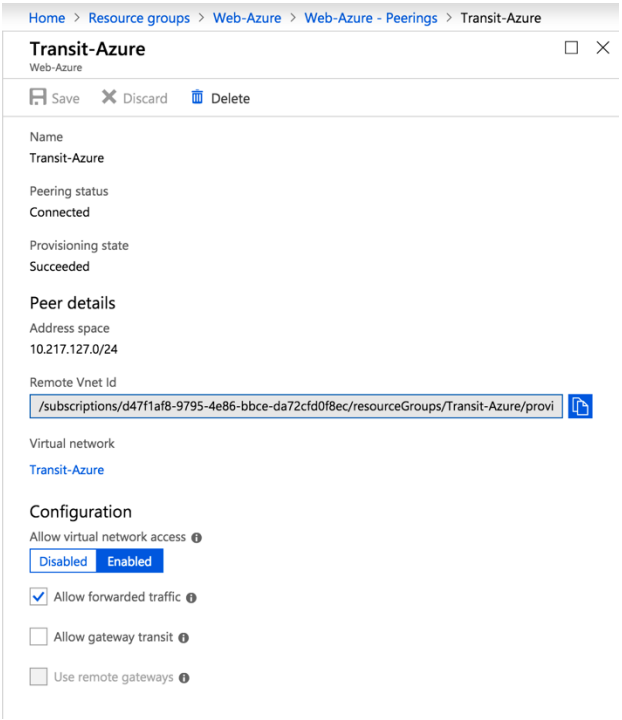
In the Transit VNet click on “Peerings” in the left column. When there is a peering existing click on it to review if the settings are right. When there is no peering visible, go directly to next step and create it by yourself.



The settings in the Transit VNet peering should looks like this. Is this not the case change it and click on “Save”.



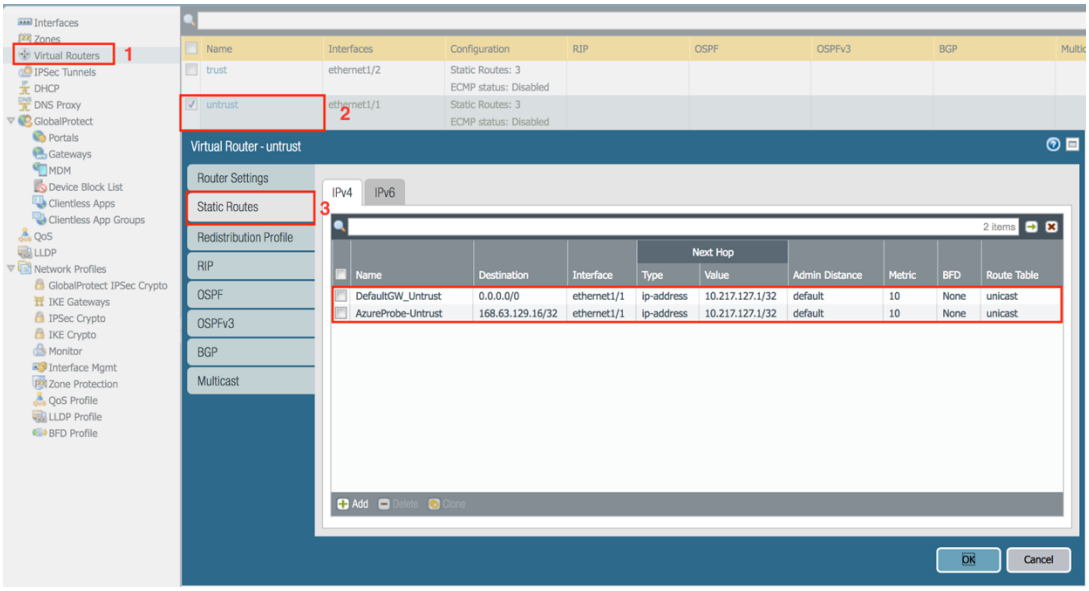
Repeat the above described steps for the Hub VNet. Below you can find the settings of route table. Please adopt the settings when it is not the same.



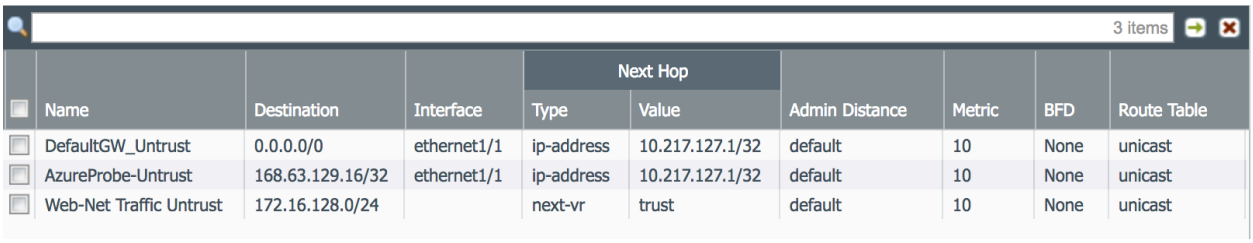
2.8.5. Review Firewall Virtual Router

In this Part we have to review does the Virtual Routers on the Firewalls are right configured. At first check the “untrust-vr”. You should have at least two static routes.

- DefaultGW_Untrust
- AzureProbe-Untrust



You have to adopt now a new Static Route for your Hub Network. See my example below.

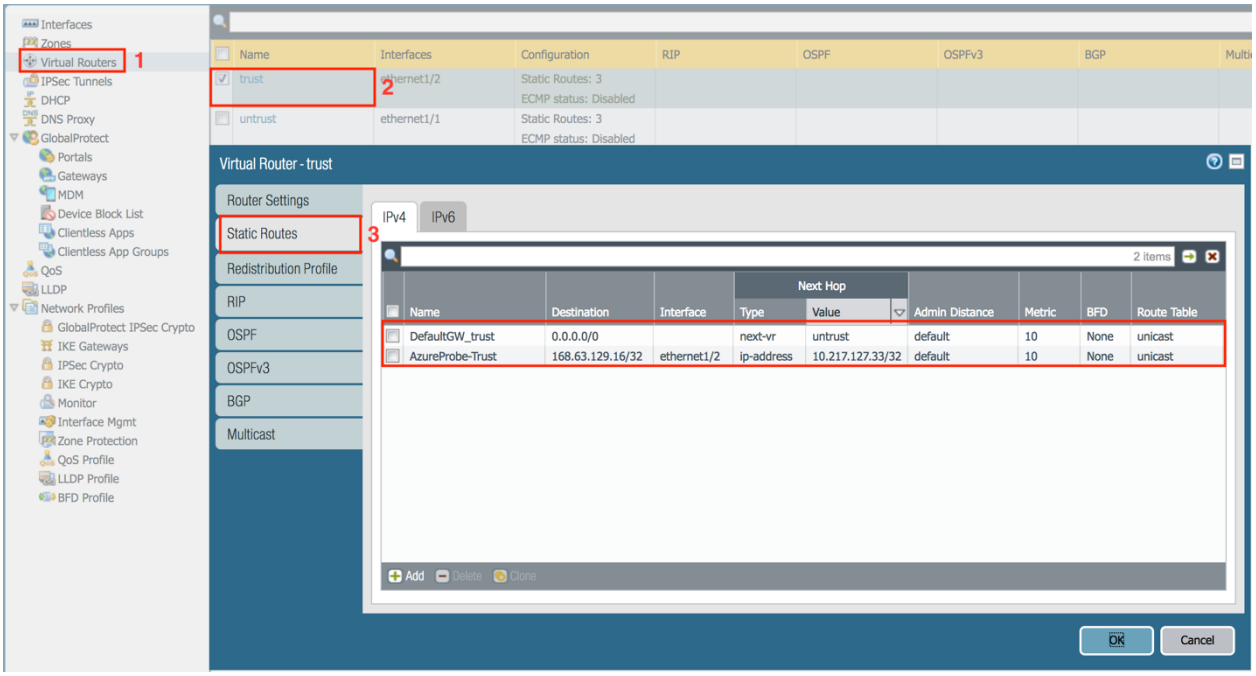


3 items

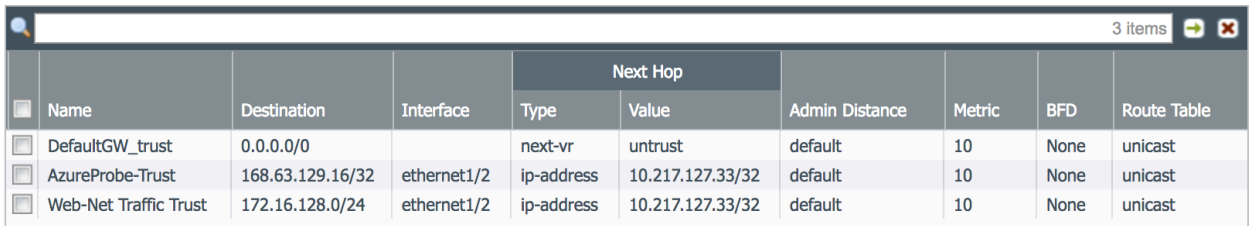
Name	Destination	Interface	Next Hop		Admin Distance	Metric	BFD	Route Table
			Type	Value				
DefaultGW_Untrust	0.0.0.0/0	ethernet1/1	ip-address	10.217.127.1/32	default	10	None	unicast
AzureProbe-Untrust	168.63.129.16/32	ethernet1/1	ip-address	10.217.127.1/32	default	10	None	unicast
Web-Net Traffic Untrust	172.16.128.0/24		next-vr	trust	default	10	None	unicast

the “trust-vr”. You should have at least two static routes.

- DefaultGW_trust
- AzureProbe-Trust



You have to adopt now a new Static Route for your Hub Network. See my example below.

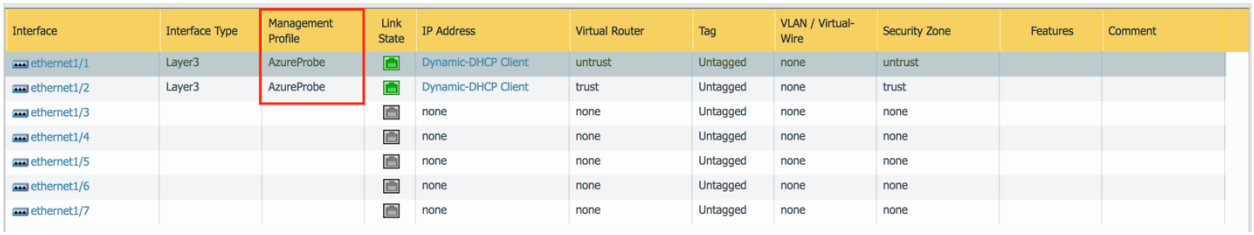


3 items

Name	Destination	Interface	Next Hop		Admin Distance	Metric	BFD	Route Table
			Type	Value				
DefaultGW_trust	0.0.0.0/0		next-vr	untrust	default	10	None	unicast
AzureProbe-Trust	168.63.129.16/32	ethernet1/2	ip-address	10.217.127.33/32	default	10	None	unicast
Web-Net Traffic Trust	172.16.128.0/24	ethernet1/2	ip-address	10.217.127.33/32	default	10	None	unicast

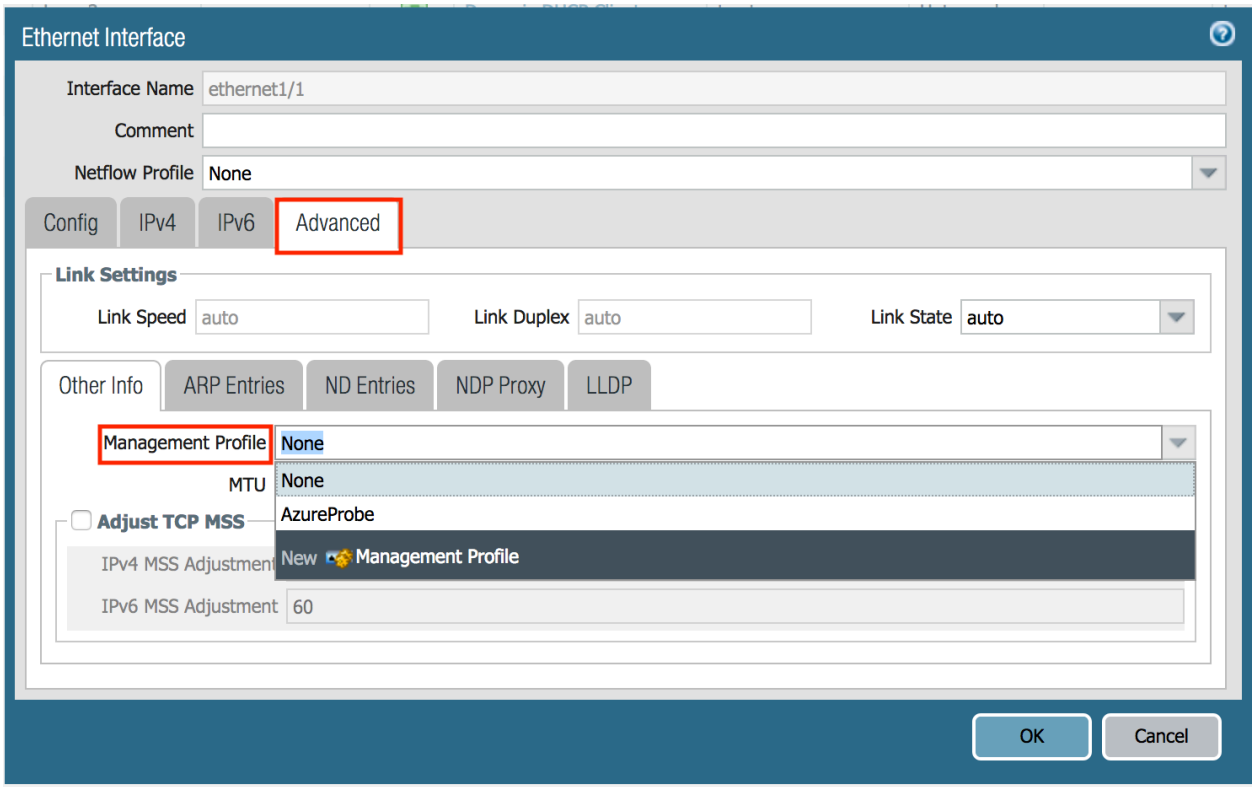
2.8.6. Review Firewall Interfaces

As next step we have to check if the Firewall interfaces are right configured. When a Load Balancer is attached is it important that on that interface is a Management Profile attached for the Azure health probe. When there is no profile attached do this please by yourself.

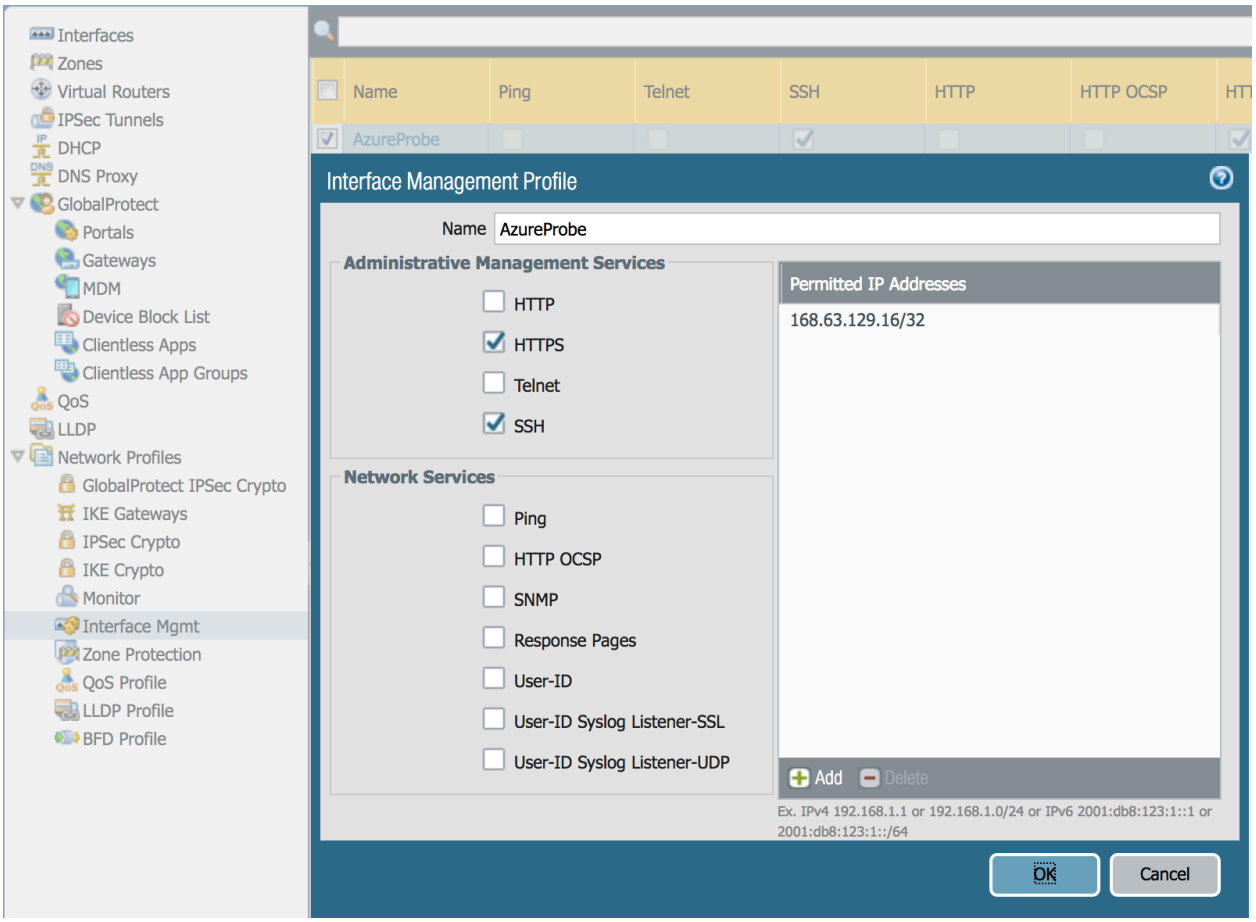


Interface	Interface Type	Management Profile	Link State	IP Address	Virtual Router	Tag	VLAN / Virtual-Wire	Security Zone	Features	Comment
ethernet1/1	Layer3	AzureProbe		Dynamic-DHCP Client	untrust	Untagged	none	untrust		
ethernet1/2	Layer3	AzureProbe		Dynamic-DHCP Client	trust	Untagged	none	trust		
ethernet1/3				none	none	Untagged	none	none		
ethernet1/4				none	none	Untagged	none	none		
ethernet1/5				none	none	Untagged	none	none		
ethernet1/6				none	none	Untagged	none	none		
ethernet1/7				none	none	Untagged	none	none		

By default, is a Management Profile created (AzureProbe). Attach this profile to all needed interface where a Load Balancer is assigned.

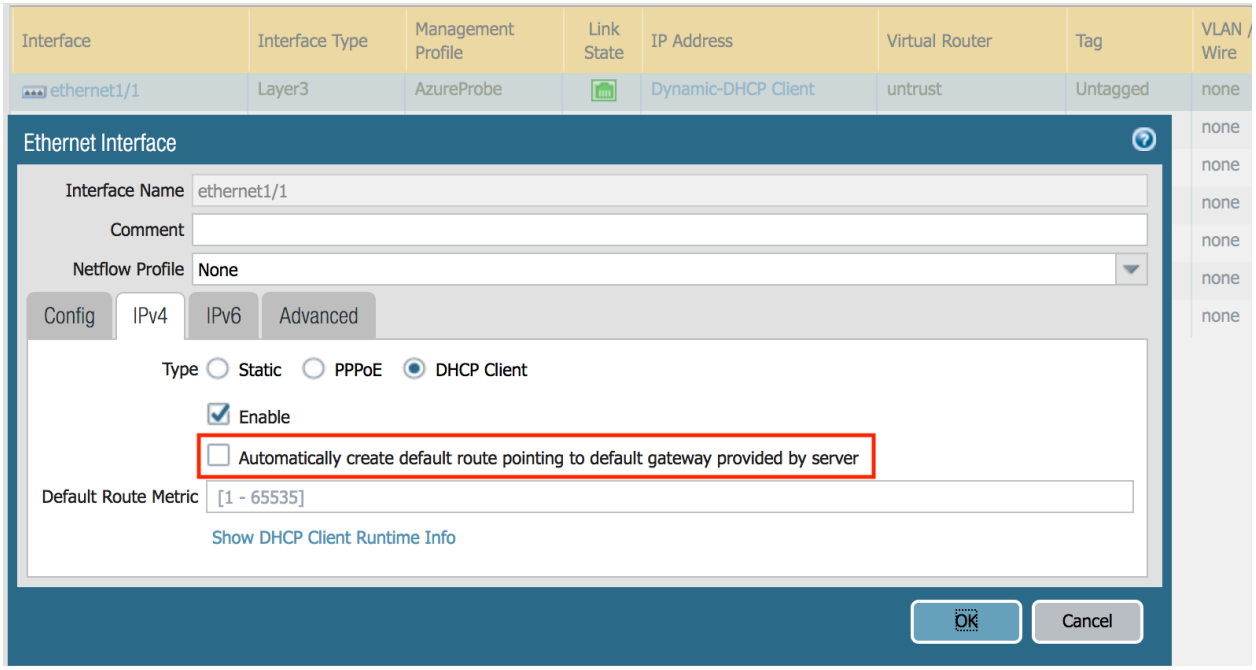


In case there is nothing created, create one Management Profile with the following settings.



When you have created an Health Probe rule on Port 80 you have then to allow on the Interface Management Profile “HTTP”

As next verify that on all configured interfaces the option “Automatically create default route pointing to default gateway provided by server” is unchecked. If this is not the case, uncheck this and commit the configuration.



2.8.7. Review Firewall NAT / Security Rules

At least you should see the following Security Rules. The “Allow Any” Security Rule is in the beginning only there to be safe that the Firewall is not blocking any traffic. This Rule should be at the end disabled or deleted.

				Source				Destination	
	Name	Tags	Type	Zone	Address	User	HIP Profile	Zone	Address
1	Azure Health Probe Traffic	Azure	universal	trust untrust	Azure-Health-Probe	any	any	trust untrust	any
2	DNS Sinkhole Block	Outbound	universal	any	any	any	any	any	Sinkhole-IPv4 Sinkhole-IPv6
3	*Outbound Bogon Block Rule*	*Outbound*	*universal*	*any*	*any*	*any*	*any*	*any*	*Team Cymru Bogons IPv4* *Team Cymru Bogons IPv6*
4	*Inbound Bogon Block Rule*	*Inbound*	*universal*	*any*	*Team Cymru Bogons IPv4* *Team Cymru Bogons IPv6*	*any*	*any*	*any*	*any*
5	Allow Any	Delete	universal	any	any	any	any	any	any
6	intrazone-default	none	intrazone	any	any	any	any	(intrazone)	any
7	interzone-default	none	interzone	any	any	any	any	any	any

This is the minimum what you need for your initial Setup in Azure. When you want to have special traffic to a Webserver you have to create a rule for them. This is not done Automatically.

			Original Packet						Translated Packet	
	Name	Tags	Source Zone	Destination Zone	Destination Interface	Source Address	Destination Address	Service	Source Translation	Destination Translation
1	No-NAT-Azure-Probe-Trust	Azure	trust	trust	any	Azure-Health-Probe	any	any	none	none
2	No-NAT-Azure-Probe-Untrust	Azure	untrust	untrust	any	Azure-Health-Probe	any	any	none	none
3	NAT default Outbound	Outbound	trust	untrust	any	any	any	any	dynamic-ip-and-port ethernet1/1	none
4	NAT default Inbound	Inbound	untrust	untrust	any	any	10.217.127.4/32	any	dynamic-ip-and-port ethernet1/2	none

In Case that there are not at least all Rules of them above deployed, create them by yourself and inform the Instructor about it.

2.8.8. Review Firewall Health Probe Traffic

When everything is right configured you should be able to see the same traffic flow as blow. In this case is it normal that the traffic on Port 80 in incomplete. Don't worry about this.

If your traffic doesn't look like this below review the configuration from the previous steps or consult an Instructor.

		Receive Time	Type	From Zone	To Zone	Source	Source User	Destination	To Port	Application	Action	Rule	Session End Reason	Bytes
		03/01 03:23:07	end	trust	trust	168.63.129.16		10.217.130.36	22	ssh	allow	Azure Health Probe Traffic	tcp-rst-from-client	370
		03/01 03:23:04	end	untrust	untrust	168.63.129.16		10.217.130.4	22	ssh	allow	Azure Health Probe Traffic	tcp-rst-from-client	370
		03/01 03:23:01	end	trust	trust	168.63.129.16		10.217.130.36	22	ssh	allow	Azure Health Probe Traffic	tcp-rst-from-client	370
		03/01 03:22:58	end	untrust	untrust	168.63.129.16		10.217.130.4	22	ssh	allow	Azure Health Probe Traffic	tcp-rst-from-client	370
		03/01 03:22:55	end	trust	trust	168.63.129.16		10.217.130.36	22	ssh	allow	Azure Health Probe Traffic	tcp-rst-from-client	370
		03/01 03:22:52	end	untrust	untrust	168.63.129.16		10.217.130.4	22	ssh	allow	Azure Health Probe Traffic	tcp-rst-from-client	370
		03/01 03:22:49	end	trust	trust	168.63.129.16		10.217.130.36	22	ssh	allow	Azure Health Probe Traffic	tcp-rst-from-client	370
		03/01 03:22:46	end	untrust	untrust	168.63.129.16		10.217.130.4	22	ssh	allow	Azure Health Probe Traffic	tcp-rst-from-client	370
		03/01 03:22:43	end	trust	trust	168.63.129.16		10.217.130.36	22	ssh	allow	Azure Health Probe Traffic	tcp-rst-from-client	370
		03/01 03:22:40	end	untrust	untrust	168.63.129.16		10.217.130.4	22	ssh	allow	Azure Health Probe Traffic	tcp-rst-from-client	370
		03/01 03:22:37	end	trust	trust	168.63.129.16		10.217.130.36	22	ssh	allow	Azure Health Probe Traffic	tcp-rst-from-client	370
		03/01 03:22:34	end	untrust	untrust	168.63.129.16		10.217.130.4	22	ssh	allow	Azure Health Probe Traffic	tcp-rst-from-client	370
		03/01 03:22:31	end	trust	trust	168.63.129.16		10.217.130.36	22	ssh	allow	Azure Health Probe Traffic	tcp-rst-from-client	370
		03/01 03:22:28	end	untrust	untrust	168.63.129.16		10.217.130.4	22	ssh	allow	Azure Health Probe Traffic	tcp-rst-from-client	370
		03/01 03:22:25	end	trust	trust	168.63.129.16		10.217.130.36	22	ssh	allow	Azure Health Probe Traffic	tcp-rst-from-client	370
		03/01 03:22:22	end	untrust	untrust	168.63.129.16		10.217.130.4	22	ssh	allow	Azure Health Probe Traffic	tcp-rst-from-client	370
		03/01 03:22:19	end	trust	trust	168.63.129.16		10.217.130.36	22	ssh	allow	Azure Health Probe Traffic	tcp-rst-from-client	370
		03/01 03:22:16	end	untrust	untrust	168.63.129.16		10.217.130.4	22	ssh	allow	Azure Health Probe Traffic	tcp-rst-from-client	370
		03/01 03:22:13	end	trust	trust	168.63.129.16		10.217.130.36	22	ssh	allow	Azure Health Probe Traffic	tcp-rst-from-client	370
		03/01 03:22:10	end	untrust	untrust	168.63.129.16		10.217.130.4	22	ssh	allow	Azure Health Probe Traffic	tcp-rst-from-client	370

2.9. Configure / Test Connection to the Web Server

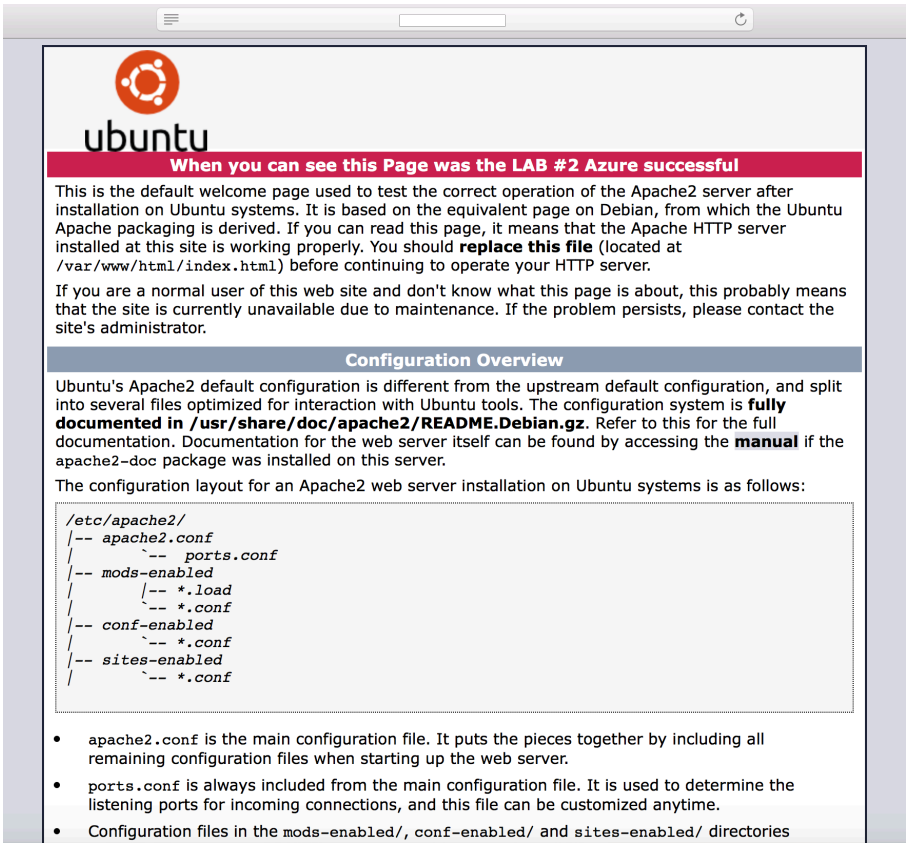
In this step we will configure at first the NAT Rules on the Firewall to access the Webserver in the Spoke Resource Group. In case you had already done the configuration of the NAT rule in the "all.yml" please review the rule.

Now we have to create/update the NAT Rules that translate the traffic on the "Untrust Interface" to the Web-Server IP address.

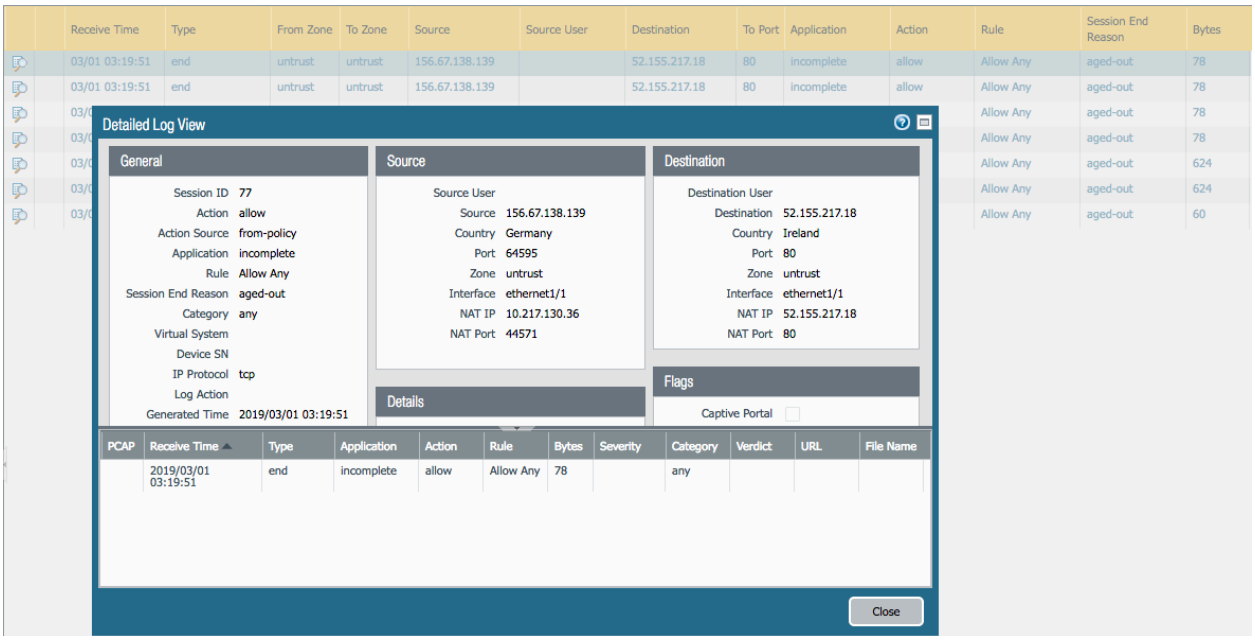
Replace the Destination address ip with your frontend IP of your Public Load balancer. In the Chapter "Review Private/Public Load Balancer" is described where you can find this ip.

			Original Packet						Translated Packet	
	Name	Tags	Source Zone	Destination Zone	Destination Interface	Source Address	Destination Address	Service	Source Translation	Destination Translation
1	NAT-Traffic to WebServer	none	untrust	untrust	any	any	52.155.219.10/32	any	dynamic-ip-and-port ethernet1/2	destination-translation address: 172.16.130.4/32 port: 80

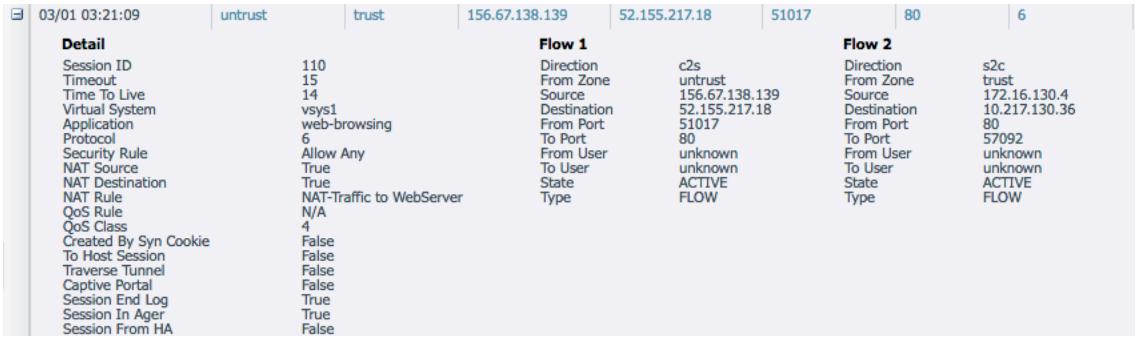
You should be now able to browse to your internal Web-Server. Type in your browser the following URL http:// <FIP-Public-LB>:80 and you should see the following page.



In the Firewall Monitor you should be able to see the traffic. See my example below



In the Session browser you can verify that the traffic is hitting the NAT Policy.



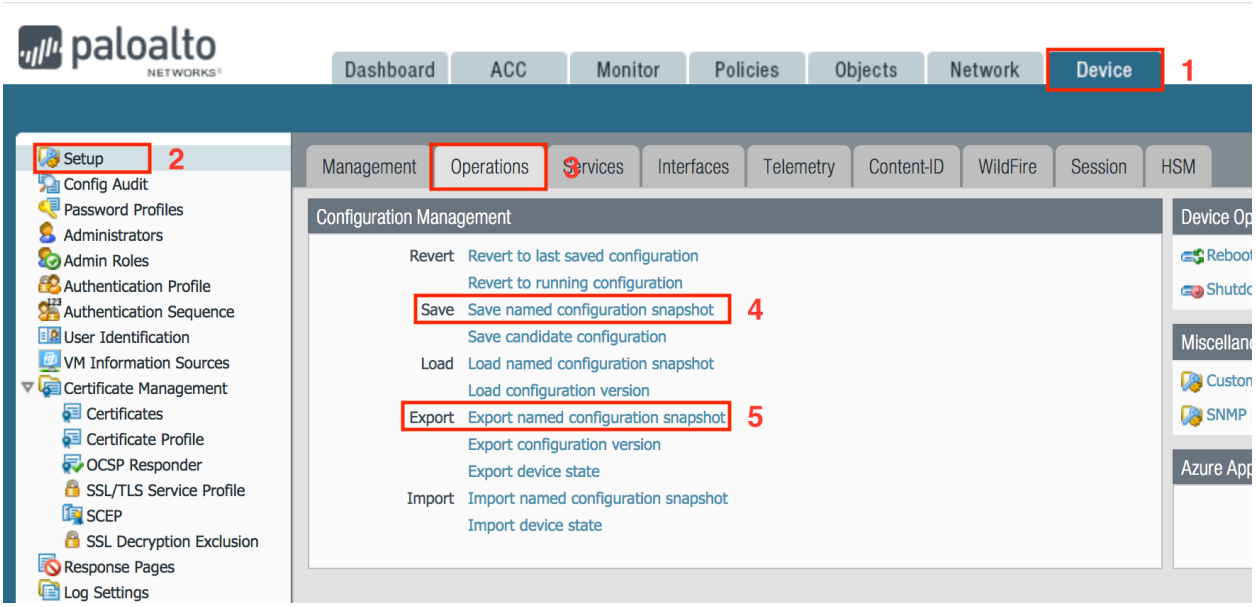
2.10. Save and Destroy Lab Environment

We will now Save and Export the Firewall configuration file for your documents and after that we will destroy the LAB environment.

2.10.1. Save and Export Firewall Configuration

Go in the Firewall to “Device (1) → Setup (2) → Operations (3)”.
As next click on “Save named configuration Snapshot (4)”. Type a proper name and save it.

When the firewall configuration is saved click on “Export named configuration snapshot (5)” and select the file what you have in the previous step created.



2.10.2. Delete Azure Environment

Now we can destroy the previous build environment, when you had saved and export the Firewall Configuration snapshot. To destroy the Azure environment, type the following command: ***"ansible-playbook destroy.yml".***

You will get prompt if you want to back up the **"terraform.tfstate"** files. You have to type **"NO"**

See below the example:

```
root@2e98edd7bbbd:/fca# ansible-playbook destroy.yml
 [WARNING]: Unable to parse /etc/ansible/hosts as an inventory source

 [WARNING]: No inventory was parsed, only implicit localhost is available

 [WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'


PLAY [DESTROY ALL RESOURCES] ***************************************************************************************************************************************************
 [WARNING]: While constructing a mapping from /fca/group_vars/all/all.yml, line 268, column 5, found a duplicate dict key (service). Using last defined value
only.

 [WARNING]: While constructing a mapping from /fca/group_vars/all/all.yml, line 277, column 5, found a duplicate dict key (service). Using last defined value
only.
```

You will see the following output when the deletion of the Azure Environment was successful.

```
TASK [DESTROY RESOURCES] *******************************************************************************************************************************************************
ok: [localhost]

PLAY RECAP *********************************************************************************************************************************************************************
localhost                  : ok=5    changed=0    unreachable=0    failed=0
```

End of LAB #2 AZURE