Automation & Orchestration with Check Point IaaS / Azure

Hi everyone!

For those that don't know me, I'm Phil and I'm a Security Engineer working for Check Point (the coolest Cyber Security company on the planet ;)) in Australia. Like most people I found myself with a lot of extra time during COVID and for one of my hobbies, I decided to start a project a while ago with the intention to demonstrate some automation capabilities of Check Point & Azure using various 3rd party tools.

The idea here was to:

- Document my own learnings on this topic and use this as a reference point for future improvements to the project

- Educate people on how useful DevOps and IaC (Infrastructure as Code) can be when working with Infrastructure deployment and configuration from saving you time to eliminating human error and optimizing processes

- Apply this concept with a security focus to the Public Cloud – an area that is slowly becoming the new norm for many enterprises today.

- Provide a basic explanation how some of these tools work so that it is easy to consume for people who are not developers / programmers by trade

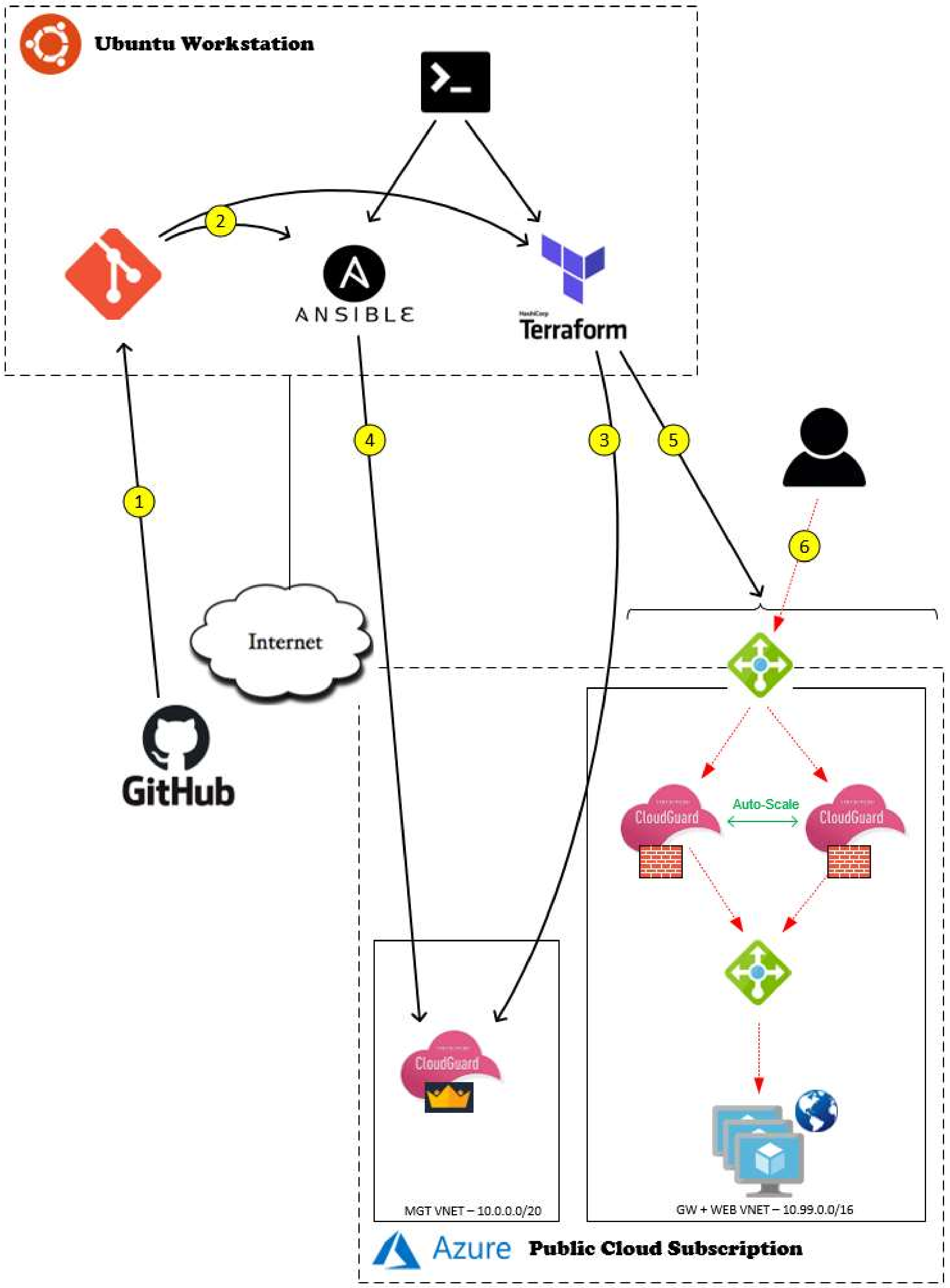
- Provide a template and a lab for interested people to try this out for themselves – after all the best way to learn is by doing!

Note - whilst this article goes into great detail about the inner workings of the automation tools and process for the benefit of learning, the point is, all this only needs to be built and understood once. Going forward, the environment can be stood up very quickly with minimal manual effort plus any slight changes in the setup will only require a small modification to the code. This allows the engineer to focus on more advanced tasks that require more attention, reducing the menial repetitive work required.

As always with any DevOps community project, it is important to collaborate and share information so I would welcome and ideas, comments or feedback however good or bad!

Overview of environment

A picture paints a thousand words – here is an overview of the environment we will be building:



Now let me explain a few of the key components:

- **Ubuntu Workstation** – A Linux Mgmt workstation hosting many of the tools required for Automation & Orchestration

- **“SECUREME” shell script** – A wrapper script written to initiate different tasks at appropriate times, wait when needed and report back to the user any information / required actions

- **Git repository** – A local code store on the workstation synced from the public github site – this contains the master secureme.sh shell script as well as the directory structure and config files for terraform / ansible

- **Terraform** – an automation tool in this instance used to define Infrastructure as Code and deploy an environment in MS Azure at the “click of a button”

- **Ansible** – an automation tool in this instance used to communicate with the Check Point Mgmt API and configure various access policies / CLI tasks

- **Azure Subscription** – a tenancy within the public cloud consisting of resource groups containing the various Infrastructure as a Service (IaaS) components being deployed and configured

- **Check Point Management Server** – a Mgmt server for the Check Point security infrastructure used to host the policies and handle the orchestration of the cloud gateway instances in an automated fashion

- **Check Point Auto-Scaling Gateways** – Azure VMs running the Check Point Gaia OS and operating in a “Scale Set” to protect the back-end web infrastructure adapting in changes to load

- **Azure Web Servers**– Azure VMs running web servers and operating in a “Scale Set” for resiliency / scalability

- **Azure Load Balancers** – Azure internal and external services designed to evenly distribute traffic from incoming listeners to the various scale set back-end nodes

And finally I will summarize the step-by-step process labelled on the picture:

1. The main pool of code used is currently in my github repository here:

   https://github.com/philipatkinson86/azure-checkpoint-automation

   The 1st step is to copy this onto the Ubuntu workstation ideally via a ‘git pull’ to the local repository

2. Once completed, there will be 2 different Terraform directories and 1 Ansible directory plus the secureme.sh script at the top level

3. When the shell script is run, this will activate the sequence of events to build / configure the environment. The first main action will be to run a '`terraform apply`' in the TfSms directory which will build the Check Point Mgmt server in Azure

4. Once the Mgmt server is built, an ansible playbook will be run to configure the objects / access policy / NAT policy on the Mgmt server plus activate a tool called CME (Cloud Management Extension) which will orchestrate the configuration of any gateways deployed in the Azure subscription

5. Once complete, the final action will be to deploy the remainder of the infrastructure in Azure consisting of auto-scaling Check Point Gateways, auto-scaling web servers and some load balancers to support this infrastructure

6. Once the script is complete, you will be able to test live web traffic to the new web environment

Pre-Requisites / Assumptions / Disclaimers

The main part of this article discusses in detail the automation process which is a repeatable action, however there will be some initial activities required to setup the environment for 1st time use – please see Appendix A at the end of the article for a full list here.

Additionally, I am assuming the reader has some basic level of knowledge around Check Point, Azure and linux, if you wish to learn the basics of any of these, here are some good starting points:

- https://community.checkpoint.com/t5/Check-Point-for-Beginners-2-0/bg-p/check-point- for-beginners-2-0

- https://azure.microsoft.com/en-au/get-started/

- https://www.digitalocean.com/community/tutorials/an-introduction-to-linux-basics

Finally, this lab is purely a testing and learning environment in that it is not meant to be replicated directly into production, for ideas on improving this setup to be ready for a more critical environment, please refer to the later section 'Future Improvements & Suggestions'

Analysis of the end-to-end process

Now I will try to explain in detail what is happening in the background when the process is initiated

- **Run the shell script**

Once the code is downloaded to the linux workstation, you can kick the process off by running `./secureme.sh` (assuming it has executable permissions). Here is an output of the contents of the script, redacting a few of my private API keys:

```
#! /bin/sh
echo "`date -u`\e[1;33m [SECUREME SCRIPT] Building Check Point Mgmt server in Azure...\e[0m"
cd TfSms
terraform init -input=false
terraform apply -auto-approve
echo "<tf apply>"
my_ip=`cat terraform.tfstate | grep value | grep -o ': ".*"' | tr -d ' :"'`
echo "`date -u`\e[1;33m [SECUREME SCRIPT] Waiting for build to finish, this should take around
25mins...\e[0m"
echo "`date -u`\e[1;33m [SECUREME SCRIPT] If you want to see progress, SSH to $my_ip
(admin/VPN123vpn123!) and tail -f /var/log/ftw_install.log\e[0m"
echo "`date -u`\e[1;33m [SECUREME SCRIPT] after FTW run \$MDS_FWDIR/scripts/cpm_status.sh to see if
the SMS is running \e[0m"
sleep 1400
echo "`date -u`\e[1;33m [SECUREME SCRIPT] Mgmt build should be finished, sync-ing latest public IP to
Ansiblie config files...\e[0m"
file1=../ansible/vars.yml
file2=../ansible/play.yml
file3=../ansible/hosts
sed -i 's/mgmt_server: .*/mgmt_server: '$my_ip'/' $file1
sed -i 's/hosts: .*/hosts: '$my_ip'/' $file2
sed -i '$d' $file3
echo $my_ip >> $file3
# sed -i '45s/.*/'$my_ip'/' $file3
echo "`date -u`\e[1;33m [SECUREME SCRIPT] New Public IP synched into ansible config files\e[0m"
echo "`date -u`\e[1;33m [SECUREME SCRIPT] Running Ansible Playbook to configure Mgmt Server
policy\e[0m"
cd ../ansible
ansible-playbook -i ./hosts play.yml
#echo "<ansible play>"
echo "`date -u`\e[1;33m [SECUREME SCRIPT] Ansible configuration complete.\e[0m"
echo ""
echo "`date -u`\e[1;33m [SECUREME SCRIPT] *** Manual Task required ***\e[0m"
echo "`date -u`\e[1;33m [SECUREME SCRIPT] Login to $my_ip via R80.40 SmartConsole
(api_user/VPN123vpn123!) and update the SMS IP to $my_ip - don't forget to Publish the session.\e[0m"
printf 'Once Manual task is complete, press [ENTER] to continue with SECUREME SCRIPT...'
read _
echo ""
echo "`date -u`\e[1;33m [SECUREME SCRIPT] *** Manual Task required ***\e[0m"
echo "`date -u`\e[1;33m [SECUREME SCRIPT] Login to $my_ip via SSH (admin/VPN123vpn123!) and run the
following commands to configure CME\e[0m"
echo ""
echo "autoprov_cfg -f init Azure -mn r80dot40mgmt -tn Azure_VisualStudio_R80.40 -otp vpn12345 -ver
R80.40 -po Standard -cn Azure -sb <Azure-Subscription> -at <Tenant-ID> -aci <Client-ID> -acs <Client-
Secret>"
echo ""
echo "autoprov_cfg -f set template -tn Azure_VisualStudio_R80.40 -av -ab -ips"
echo ""
printf 'Once Manual task is complete, press [ENTER] to continue with SECUREME SCRIPT...'
read _
echo "`date -u`\e[1;33m [SECUREME SCRIPT] Building Check Point Gateway Scale-Set and web server
infrastructure in Azure...\e[0m"
cd ../TfGwWeb
terraform init -input=false
terraform apply -auto-approve
#echo "<tf apply>"
echo "`date -u`\e[1;33m [SECUREME SCRIPT] Waiting for build to finish, this should take around
5mins...\e[0m"
```

```
echo "`date -u`\e[1;33m [SECUREME SCRIPT] If you want to see progress, SSH to $my_ip
(admin/VPN123vpn123!) and tail -f /var/log/CPcme/cme.log\e[0m"
sleep 300
my_ip2=`cat terraform.tfstate | grep value | grep -o ': ".*"' | tr -d ' :"\n'`
echo "`date -u`\e[1;33m [SECUREME SCRIPT] Gateway / Web build should be finished, please test by
browsing to http://$my_ip2...\e[0m"
```

The script essentially starts by echoing a message back to the user and changes directory into the TfSms folder

- **Terraform apply (Mgmt)**

The first action is to run a terraform init followed by a terraform apply within the TfSms folder and you will see the output as follows:

```
Thu 03 Sep 2020 03:58:28 AM UTC [SECUREME SCRIPT] Building Check Point Mgmt server in Azure...

Initializing the backend...

Initializing provider plugins...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.azurerm: version = "~> 2.25"
* provider.random: version = "~> 2.3"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
azurerm_resource_group.rg: Creating...
azurerm_marketplace_agreement.checkpoint: Creating...
azurerm_resource_group.rg: Creation complete after 9s
[id=/subscriptions/<REDACTED>/resourceGroups/tf-rg-sms]
azurerm_public_ip.mgmtpublicip: Creating...
random_id.randomId: Creating...
random_id.randomId: Creation complete after 0s [id=0JUZ0iQO9dU]
azurerm_virtual_network.vnet: Creating...
```

Usually this command would require you to confirm the changes by typing ‘yes’ at the prompt but I added the `–auto-approve` switch to avoid this behavior.

What happens here is terraform reviews a series of .tf files within the directory that are defining the Infrastructure as Code (IaC) for the Check Point Mgmt server. The various files are as follows:

- **chkp_mgmt.tf** – defines the main settings for the server
- **customdata.sh** – this is a script that will run on the OS post-deployment to add a few settings like creating an API user / kick off the first time wizard
- **main-azure.tf** – defines the version of ‘azurerm’ provider in use (the terraform azure plugin)

- **nics.tf** – defines interfaces and IPs attached to the server

- **outputs.tf** – defines output variables to print when the apply is finished (this is important for the next section)

- **public_ip.tf** – defines a random static public IP to associate to the server (this will be different each time the environment is stood up)

- **random.tf** – random text generator required for a unique storage account name

- **resource_group.tf** – defines the name and region (Australia East) of the resource group in Azure all the components will be contained under

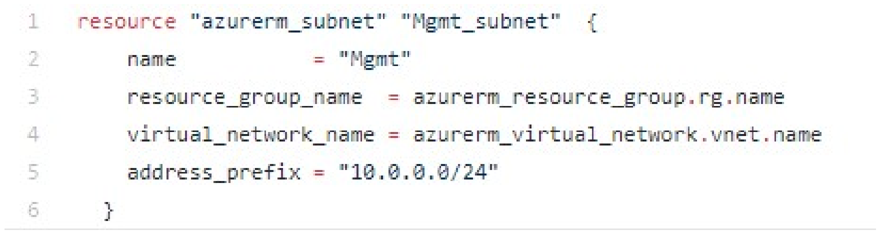
- **storage_account.tf** – defines a storage account for the server with LRS replication

- **subnets.tf** – defines the network subnet address space the new machine will reside on

- **vnet.tf** – defines the virtual network the server will reside on along with the region

The process will also create a series of .tfstate files which define the environment state – these can be deleted if starting from scratch but are important if you are going to apply / destroy your environment on demand.

Here is an example of a .tf file – this format is easy to read by humans but also easily parsed by a machine:



```
resource "azurerm_subnet" "Mgmt_subnet"  {
    name           = "Mgmt"
    resource_group_name  = azurerm_resource_group.rg.name
    virtual_network_name = azurerm_virtual_network.vnet.name
    address_prefix = "10.0.0.0/24"
  }
```

Once completed, the script needs to wait for around 25 mins for the build to finish which allows the first time wizard to run and the CPUs to cool down on the new server – if you want to track this progress there is a log file that can be tailed, otherwise go take a break for 25mins – you have earned it!

```
Thu 03 Sep 2020 04:03:39 AM UTC [SECUREME SCRIPT] Waiting for build to finish, this should take
around 25mins...
Thu 03 Sep 2020 04:03:39 AM UTC [SECUREME SCRIPT] If you want to see progress, SSH to 20.188.205.25
(admin/VPN123vpn123!) and tail -f /var/log/ftw_install.log
Thu 03 Sep 2020 04:03:39 AM UTC [SECUREME SCRIPT] after FTW run $MDS_FWDIR/scripts/cpm_status.sh to
see if the SMS is running
```

- **Sync ansible variables**

Once the 1st terraform apply is complete, the script will gather some information from the tfstate files to figure out what is the new IP of the server and it subsequently updates a few ansible config files (using the `sed` command to do a find & replace) so that the next phase of the script will reach out to the correct server:

```
Thu 03 Sep 2020 04:26:59 AM UTC [SECUREME SCRIPT] Mgmt build should be finished, sync-ing latest
public IP to Ansiblie config files...
Thu 03 Sep 2020 04:26:59 AM UTC [SECUREME SCRIPT] New Public IP synched into ansible config files
```

```
my_ip=`cat terraform.tfstate | grep value | grep -o ': ".*"' | tr -d ' :"'`
…
file1=../ansible/vars.yml
file2=../ansible/play.yml
file3=../ansible/hosts
sed -i 's/mgmt_server: .*/mgmt_server: '$my_ip'/' $file1
sed -i 's/hosts: .*/hosts: '$my_ip'/' $file2
sed -i '$d' $file3
echo $my_ip >> $file3
```

Here is a section of one of the config files following an update:

```
ansible_network_os=checkpoint
[checkpointr8040]
52.156.185.18
```

- **Ansible playbook**

The next phase of the script is to run an ansible playbook:

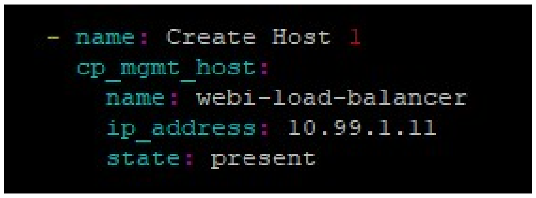
```
echo "`date -u`\e[1;33m [SECUREME SCRIPT] Running Ansible Playbook to configure Mgmt Server
policy\e[0m"
cd ../ansible
ansible-playbook -i ./hosts play.yml
```

Again, you will receive some output as the various plays are executed via the Check Point Mgmt API:

```
Thu 03 Sep 2020 04:26:59 AM UTC [SECUREME SCRIPT] Running Ansible Playbook to configure Mgmt Server
policy

PLAY [20.188.205.25]
****************************************************************************************************
***************************************************************************

TASK [Gathering Facts]
****************************************************************************************************
*************************************************************************
ok: [20.188.205.25]

TASK [Create Dynamic object 1]
****************************************************************************************************
*****************************************************************
changed: [20.188.205.25]

TASK [Create Dynamic object 2]
****************************************************************************************************
*****************************************************************
changed: [20.188.205.25]

TASK [Create Host 1]
****************************************************************************************************
***************************************************************************
changed: [20.188.205.25]
```

```
TASK [Create Host 2]
******************************************************************************************************
****************************************************************************
changed: [20.188.205.25]

TASK [Create Host 3]
******************************************************************************************************
****************************************************************************
changed: [20.188.205.25]

TASK [Create Service 1]
******************************************************************************************************
*************************************************************************
changed: [20.188.205.25]
```

Here is an example of 1 of the plays:



```
- name: Create Host 1
  cp_mgmt_host:
    name: webi-load-balancer
    ip_address: 10.99.1.11
    state: present
```

This correlates to an ansible module that is detailed here:

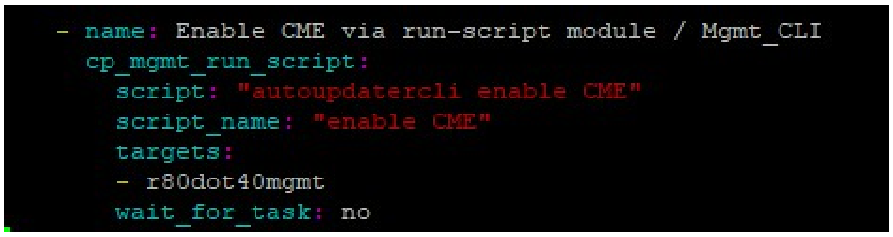
https://docs.ansible.com/ansible/latest/modules/cp_mgmt_host_module.html

And in the background, this will be talking to the Mgmt API on the new server via REST web services:

https://sc1.checkpoint.com/documents/latest/APIs/index.html?#web/set-host~v1.6%20

A series of tasks are completed and the full set of policy created can be found in Appendix B at the end of this article.

The final action here is to enable the Cloud Management Extension (CME) to prepare for the next phase:



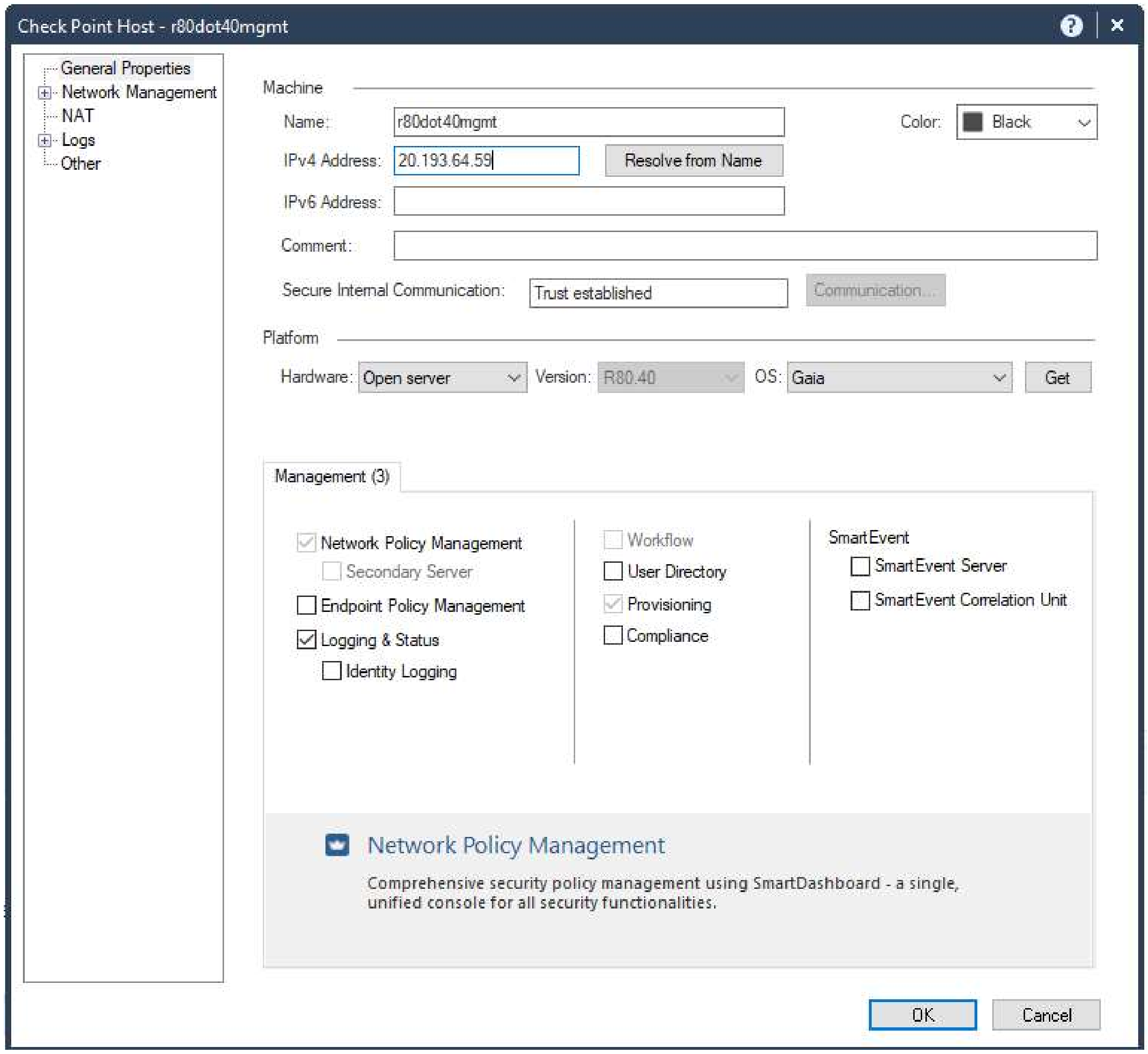
```
- name: Enable CME via run-script module / Mgmt_CLI
  cp_mgmt_run_script:
    script: "autoupdatercli enable CME"
    script_name: "enable CME"
    targets:
    - r80dot40mgmt
    wait_for_task: no
```

- **A few manual tasks**

There were a few small tasks I was not able to automate at the time of writing this and the next phase in the script gives instructions / requires the user to hit [ENTER] to continue once complete:

```
Thu 03 Sep 2020 04:27:40 AM UTC [SECUREME SCRIPT] Ansible configuration complete.

Thu 03 Sep 2020 04:27:40 AM UTC [SECUREME SCRIPT] *** Manual Task required ***
Thu 03 Sep 2020 04:27:40 AM UTC [SECUREME SCRIPT] Login to 20.188.205.25 via R80.40 SmartConsole
(api_user/VPN123vpn123!) and update the SMS IP to 20.188.205.25 - don't forget to Publish the
session.
Once Manual task is complete, press [ENTER] to continue with SECUREME SCRIPT...

Thu 03 Sep 2020 04:29:33 AM UTC [SECUREME SCRIPT] *** Manual Task required ***
Thu 03 Sep 2020 04:29:33 AM UTC [SECUREME SCRIPT] Login to 20.188.205.25 via SSH
(admin/VPN123vpn123!) and run the following commands to configure CME
```

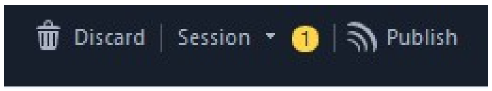
The first action will be to login to the SmartConsole GUI and set the new public IP on the Mgmt server (the private IP is configured by default):


Check Point Host - r80dot40mgmt
General Properties
Network Management
NAT
Logs
Other
Machine
Platform
Resolve from Name
Communication...
Get
Management (3)
Network Policy Management
Comprehensive security policy management using SmartDashboard - a single, unified console for all security functionalities.
OK
Cancel

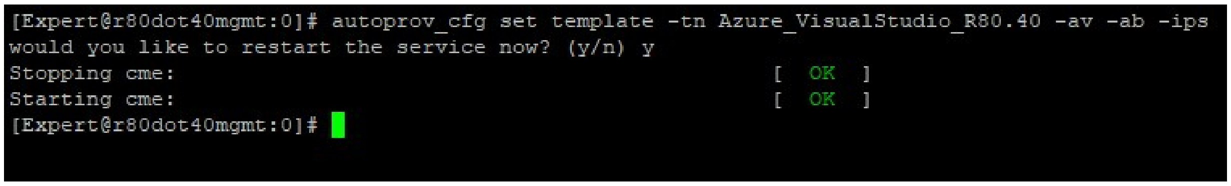
Name: r80dot40mgmt
Color: Black
IPv4 Address: 20.193.64.59
IPv6 Address:
Comment:
Secure Internal Communication: Trust established
Hardware: Open server
Version: R80.40
OS: Gaia

[✓] Network Policy Management
[ ] Secondary Server
[ ] Endpoint Policy Management
[✓] Logging & Status
[ ] Identity Logging
[ ] Workflow
[ ] User Directory
[✓] Provisioning
[ ] Compliance
SmartEvent
[ ] SmartEvent Server
[ ] SmartEvent Correlation Unit

You are then required to publish the session at the top of the console:



Secondly, you need to run a few commands on the CLI from an SSH session in order to configure the CME process – you will be prompted to restart the service following these:



```
[Expert@r80dot40mgmt:0]# autoprov_cfg set template -tn Azure_VisualStudio_R80.40 -av -ab -ips
would you like to restart the service now? (y/n) y
Stopping cme:                                                  [  OK  ]
Starting cme:                                                  [  OK  ]
[Expert@r80dot40mgmt:0]#
```

Note – the code in github has some API keys redacted for this session – please see Appendix A on how to setup a new API key on your private Azure subscription.

- **Terraform apply (Gateways / web infra)**

The final stage of the script is to change directory to the 2nd Terraform folder TfGwWeb and run a terraform apply to deploy the remainder of the infrastructure:

```
Thu 03 Sep 2020 04:29:34 AM UTC [SECUREME SCRIPT] Building Check Point Gateway Scale-Set and web
server infrastructure in Azure...

Initializing the backend...

Initializing provider plugins...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.random: version = "~> 2.3"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
azurerm_marketplace_agreement.checkpoint: Creating...
azurerm_resource_group.rg: Creating...
azurerm_resource_group.rg: Creation complete after 5s
[id=/subscriptions/<REDACTED>/resourceGroups/tf-rg-gw-web]
random_id.randomId: Creating...
random_id.randomId: Creation complete after 0s [id=zEf7IBXeYbw]
azurerm_public_ip.albvip1: Creating...
azurerm_virtual_network.vnet: Creating...
azurerm_availability_set.avset: Creating...
azurerm_route_table.DMZ1RT: Creating...
```

There are a few more .tf files present in this directory as this infrastructure is more complicated than the 1st:

- **avset.tf** – defines some auto-scaling settings for the web server scale-set
- **check_point_scaleset.tf** – main config file for the Check Point Gateways scale set
- **loadbalancers.tf** – defines all required load balancers along with frontend, backend and probe configuration

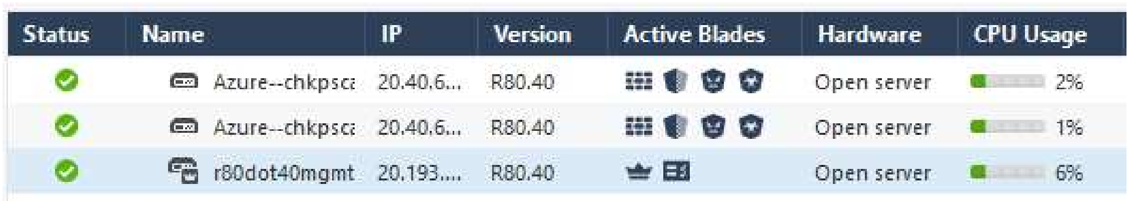
- **route_table.tf** – defines routing tables and individual routes for the various subnets to ensure all the web traffic is channeled through the Check Point gateways

- **security_group.tf** – define a permissive rule for the mandatory security group (all access policy will be defined on the Check Point gateways instead of NSGs)

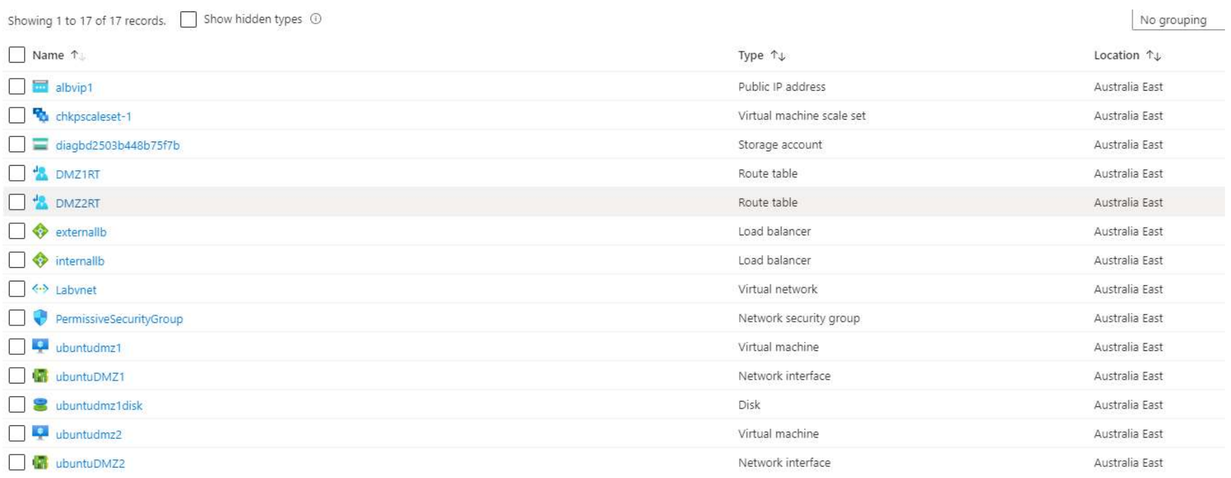
- **ubuntu_servers.tf** – main config file for the Ubuntu web server scale set

Once complete, the script will provide a final message and you will be able to see the configured assets defined in the Azure / Check Point consoles respectively:

```
Thu 03 Sep 2020 04:34:53 AM UTC [SECUREME SCRIPT] Waiting for build to finish, this should take
around 5mins...
Thu 03 Sep 2020 04:34:53 AM UTC [SECUREME SCRIPT] If you want to see progress, SSH to 20.188.205.25
(admin/VPN123vpn123!) and tail -f /var/log/CPcme/cme.log
Thu 03 Sep 2020 04:39:53 AM UTC [SECUREME SCRIPT] Gateway / Web build should be finished, please test
by browsing to http://20.193.44.150...
```



Status	Name	IP	Version	Active Blades	Hardware	CPU Usage
✓	Azure--chkpsca	20.40.6...	R80.40		Open server	2%
✓	Azure--chkpsca	20.40.6...	R80.40		Open server	1%
✓	r80dot40mgmt	20.193....	R80.40		Open server	6%



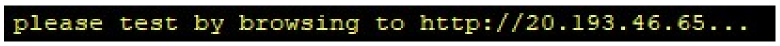
Showing 1 to 17 of 17 records. Show hidden types

No grouping

Name	Type	Location
albvip1	Public IP address	Australia East
chkpscaleset-1	Virtual machine scale set	Australia East
diagbd2503b448b75f7b	Storage account	Australia East
DMZ1RT	Route table	Australia East
DMZ2RT	Route table	Australia East
externallb	Load balancer	Australia East
internallb	Load balancer	Australia East
Labvnet	Virtual network	Australia East
PermissiveSecurityGroup	Network security group	Australia East
ubuntudmz1	Virtual machine	Australia East
ubuntuDMZ1	Network interface	Australia East
ubuntudmz1disk	Disk	Australia East
ubuntudmz2	Virtual machine	Australia East
ubuntuDMZ2	Network interface	Australia East

- **Testing**

Now all that is left to do is test by browsing to the URL in the output in your chosen web browser and you should be presented with the below HTML page with an indicator of which web server the load balancer has redirected you to (dmz1 or dmz2):



```
please test by browsing to http://20.193.46.65...
```

dmz1

Hello World - Check Point CloudGuard IAAS Demo !

- **Final Note**

  Probably the most important point to note is don't forget to shut down your environment after you have finished testing as you will be charged a small amount via Azure for the time it is in use. This can be done by deleting the resource groups manually in the Azure console or even better – run a '`terraform destroy`' from the TfGwWeb and TfSms directories to automate this process and keep the tfstate files up to date (note – there is a `destroy.sh` script in the github repo to automate all of this):

```
root@ubuntu1:/home/phil/git_environment/azure-checkpoint-automation# ./destroy.sh
Thu 03 Sep 2020 04:52:30 AM UTC [Destroy SECUREME SCRIPT] Destroy the SMS in Azure...
azurerm_resource_group.rg: Refreshing state... [id=/subscriptions/<REDACTED>/resourceGroups/tf-rg-
sms]
azurerm_marketplace_agreement.checkpoint: Refreshing state...
[id=/subscriptions/<REDACTED>/providers/Microsoft.MarketplaceOrdering/agreements/checkpoint/offers/ch
eck-point-cg-r8040/plans/mgmt-byol]
random_id.randomId: Refreshing state... [id=0JUZ0iQO9dU]
azurerm_virtual_network.vnet: Refreshing state... [id=/subscriptions/<REDACTED>/resourceGroups/tf-rg-
sms/providers/Microsoft.Network/virtualNetworks/myVNET]
azurerm_public_ip.mgmtpublicip: Refreshing state... [id=/subscriptions/<REDACTED>/resourceGroups/tf-
rg-sms/providers/Microsoft.Network/publicIPAddresses/r80dot40mgmt]
azurerm_storage_account.mystorageaccount: Refreshing state...
[id=/subscriptions/<REDACTED>/resourceGroups/tf-rg-
sms/providers/Microsoft.Storage/storageAccounts/diagd09519d2240ef5d5]
azurerm_subnet.Mgmt_subnet: Refreshing state... [id=/subscriptions/<REDACTED>/resourceGroups/tf-rg-
sms/providers/Microsoft.Network/virtualNetworks/myVNET/subnets/Mgmt]
azurerm_network_interface.mgmtexternal: Refreshing state...
[id=/subscriptions/<REDACTED>/resourceGroups/tf-rg-
sms/providers/Microsoft.Network/networkInterfaces/r80dot40mgmt-eth0]
azurerm_virtual_machine.chkpmgmt: Refreshing state...
[id=/subscriptions/<REDACTED>/resourceGroups/tf-rg-
sms/providers/Microsoft.Compute/virtualMachines/r80dot40mgmt]

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  - destroy
```

```
Terraform will perform the following actions:

  # azurerm_marketplace_agreement.checkpoint will be destroyed
  - resource "azurerm_marketplace_agreement" "checkpoint" {
      - id                  =
"/subscriptions/<REDACTED>/providers/Microsoft.MarketplaceOrdering/agreements/checkpoint/offers/check
-point-cg-r8040/plans/mgmt-byol" -> null
      - license_text_link   =
"https://storelegalterms.blob.core.windows.net/legalterms/3E5ED_legalterms_CHECKPOINT%253a24CHECK%253
a2DPOINT%253a2DCG%253a2DR8040%253a24MGMT%253a2DBYOL%253a24U2R6YKHF2KWHXN7Y4Q4Q4OEKEYL6JZJCCZGIIGQBSB7
FNDUBYTDIRQY6QPT5XMT7NGAH5XWH3LHSQY22URTFS3X7HZHQXZ3CIVJKC2Y.txt" -> null
      - offer               = "check-point-cg-r8040" -> null
      - plan                = "mgmt-byol" -> null
      - privacy_policy_link = "http://www.checkpoint.com/privacy" -> null
      - publisher           = "checkpoint" -> null
    }
```

Future Improvements & Suggestions

Given this example is far from perfect, here are a few suggestions on how improvements could be made to make it even better – I'm open to other ideas here:

- Automation of the 2 manual tasks – whilst I ran out of time to do this, my suggestion would be to explore using plink and / or DBedit to achieve this:
  - https://www.thegeekstuff.com/2017/05/putty-plink-examples/
  - https://supportcenter.checkpoint.com/supportcenter/portal?eventSubmit_doGoviewsolutiondetails=&solutionid=skI3301

- Applying a production license to the deployment – in its current form the services will only work for a 15 day grace period – if you are looking to run the environment for longer than that – go have a conversation with your local friendly Check Point sales rep!

- It's always a good idea to put the latest GA (general availability) hotfix package on Check Point gateways – this could be also automated with CME:
  - https://sc1.checkpoint.com/documents/IaaS/WebAdminGuides/EN/CP_CME/Content/Topics-Cloud_Management_Extension_CME/Automatic_Hotfix_Deployment.htm?tocpath=Automatic%20Hotfix%20Deployment%7C_____0#Configuring_the_Automatic_Hotfix_Deployment

- More of a testing topic than improvement, but I would recommend ramping the load up on your scale sets to test their ability to scale out horizontally (use the simulate_cpu_load.sh script below), plus check the threat prevention is working by trying to connect to the below benign link from one of the web servers:
  - https://github.com/CheckPointSW/CloudGuardIaaS/blob/master/common/simulate_cpu_load.sh
  - http://sc1.checkpoint.com/za/images/threatwiki/pages/TestAntiBotBlade.html

I Hope you found this article useful and you either learned something new or are inspired to go test out the power of Terraform / Ansible – if there are any questions, comments or feedback, please feel free to contact me directly at philipa@checkpoint.com

I will hopefully be posting more content like this in the coming months!

Appendix A – Setting up the Environment

I would like to list all the pre-requisite setup that is required – depending on your knowledge, some or all of this may be trivial or obvious, but I thought it best to list it all out anyway to cater for beginners:

- **Ubuntu setup** – I have used Ubuntu for my automation workstation, technically any linux environment would work but you can download Ubuntu from here:

https://ubuntu.com/download/server

  I would recommend installing this on a virtual machine on top of windows and then you can use tools from both windows and linux where required.

  Don't forget to run '`apt get update`' and '`apt get upgrade`' once the installation has finished to ensure all the latest updates are installed.

- **Terraform install** – Once Ubuntu is installed, providing it has access to the internet, you can install terraform by following these instructions from HashiCorp:

https://learn.hashicorp.com/tutorials/terraform/install-cli

- **Ansible install** – to install ansible on Ubuntu you can use the main package manager by simply running '`apt install ansible`'

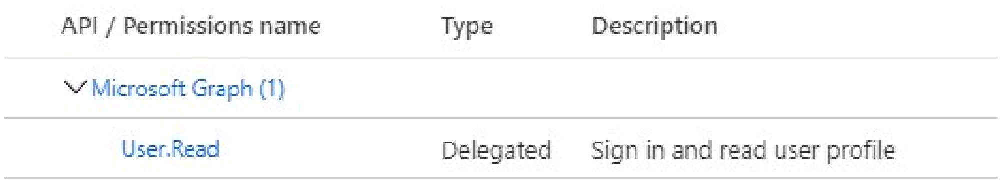
- **Create an azure account** – If you do not already have a Microsoft Azure account, one can be created at:

https://azure.microsoft.com/en-au/free

- **Configure Azure API user** – Part of the script will require you to specify some details about your Azure subscription:
  - Azure Subscription ID
  - Tenant ID
  - Client ID
  - Client Secret

  Whilst the 1st 2 of these can be gathered from the main Azure console by default, for the client ID / secret, an API user will need to be configured.

  This can be done as follows:
  - Home > App Registrations > New Registration
  - Ensure to grant the following permissions to the application:



API / Permissions name	Type	Description
Microsoft Graph (1)		
User.Read	Delegated	Sign in and read user profile

Once this is done and all IDs are collected, you can update the secureme.sh script on the following line 33:

```
echo "autoprov_cfg -f init Azure -mn r80dot40mgmt -tn Azure_VisualStudio_R80.40 -otp vpn12345 -ver R80.40 -po Standard -cn Azure -sb <Azure-Subscription> -at <Tenant-ID> -aci <Client-ID> -acs <Client-Secret>"
```

- **Accept marketplace agreements** – If you have not created similar Azure VMs before, you will need to manually do that before the machine deployments can be created and the Marketplace Agreement accepted. The machines in use are:
  - check-point-cg-r8040 (sg) - BYOL
  - check-point-cg-r8040 (mgmt) - BYOL

  Note – there is a set of powershell commands to automate this in the TfSms directory under the file 'PowerShell_info.txt'

- **Install azure-cli** – In order to interact with Azure automatically from Ubuntu, you will need to install a tool called azure-cli. You can use the main package manager by simply running '`apt install azure-cli`'

- **Az login** – Once azure-cli is installed, you can run the command '`az login`' and this will give you a link to manually open in a browser where credentials can be entered. Once this process is complete you will not need to enter credentials again, enabling the automation to happen.

- **Download / install SmartConsole R80/40 / Putty (windows)** – For the manual tasks you will need to use the putty and SmartConsole applications – these can be downloaded from here:

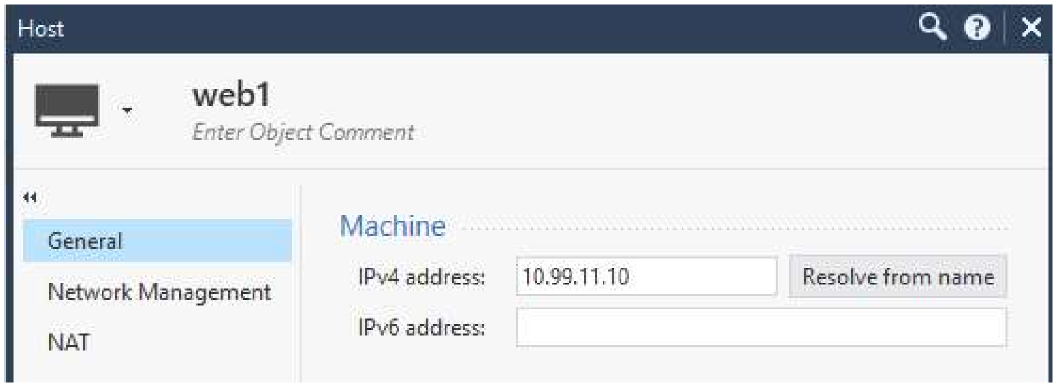
  - https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html
  - https://supportcenter.checkpoint.com/supportcenter/portal?action=portlets.DCFileAction&eventSubmit_doGetdcdetails=&fileid=101086

  These tools are also generally useful for troubleshooting and learning the setup of the infrastructure.

Appendix B – Check Point Policy

In this section, I would like to detail the Check Point policy created with the ansible playbook

- Standard Network Objects:


Host

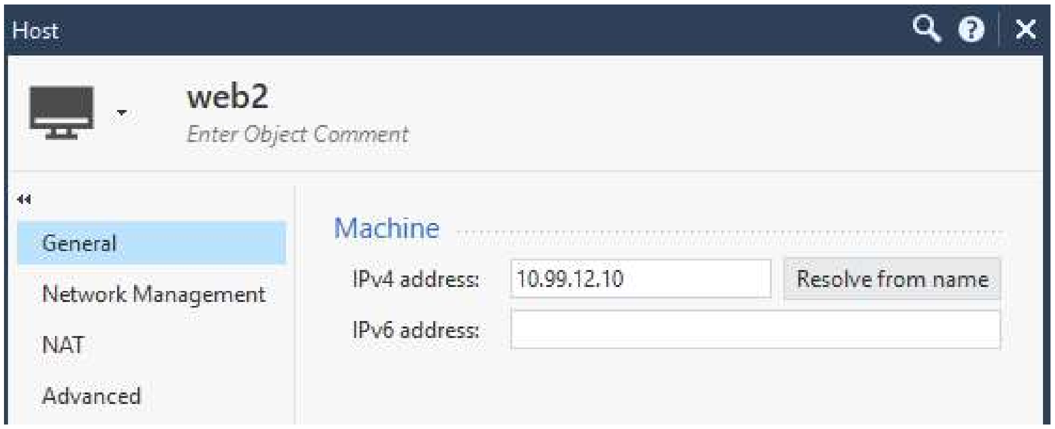
web1

Enter Object Comment

Machine

IPv4 address: 10.99.11.10

IPv6 address:


Host

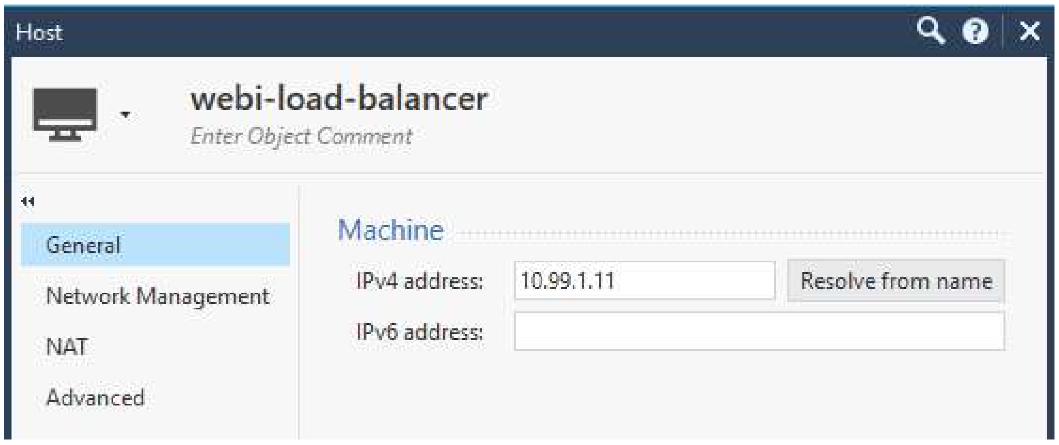
web2

Enter Object Comment

Machine

IPv4 address: 10.99.12.10

IPv6 address:


Host

webi-load-balancer

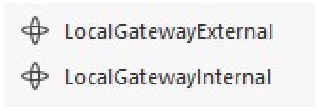
Enter Object Comment

Machine

IPv4 address: 10.99.1.11

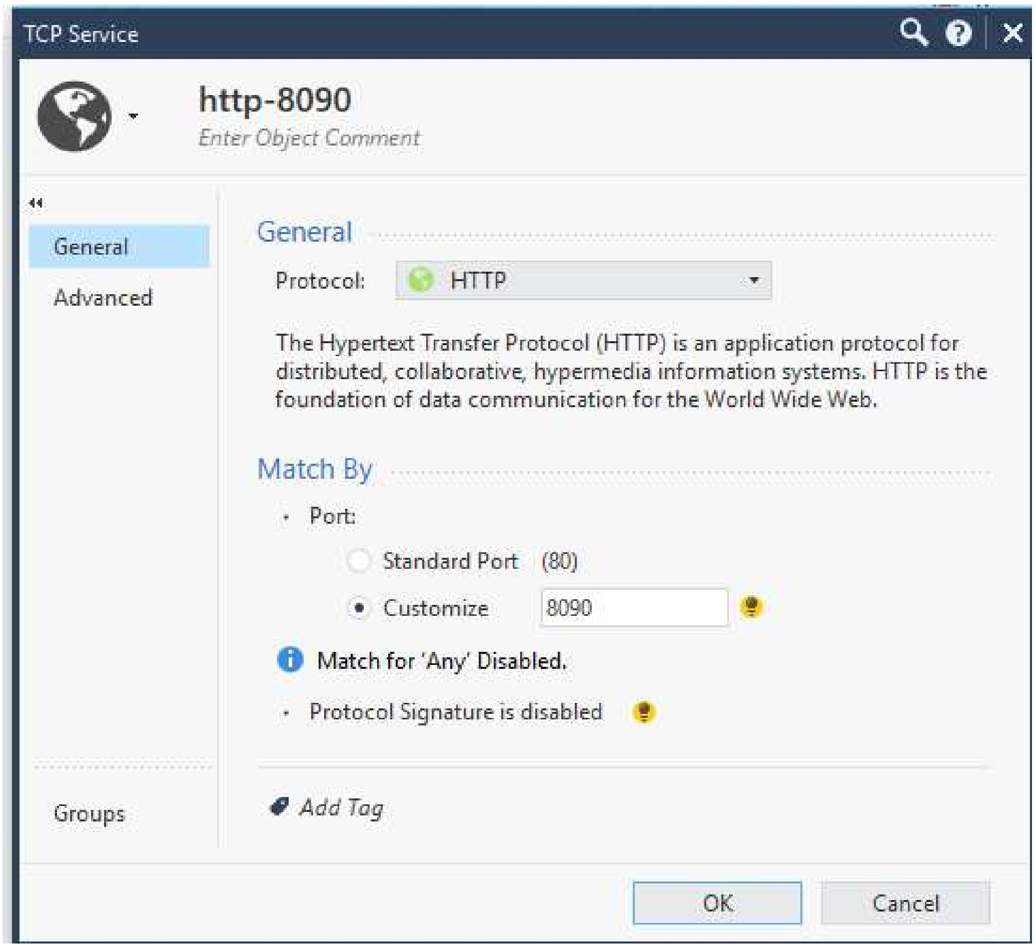
IPv6 address:

- Dynamic Network Objects:



LocalGatewayExternal
LocalGatewayInternal

- Service Objects:



TCP Service

http-8090
Enter Object Comment

General

Protocol: HTTP

The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web.

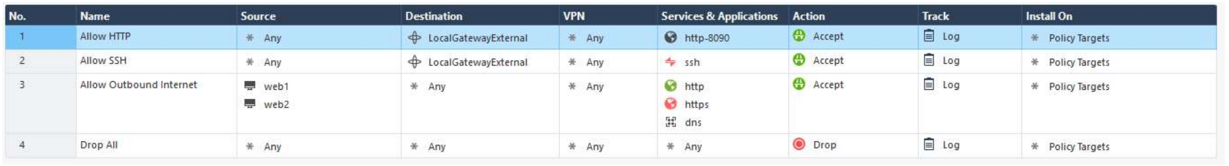
Match By

- Port:
  - Standard Port (80)
  - [✓ Customize] 8090

Match for 'Any' Disabled.

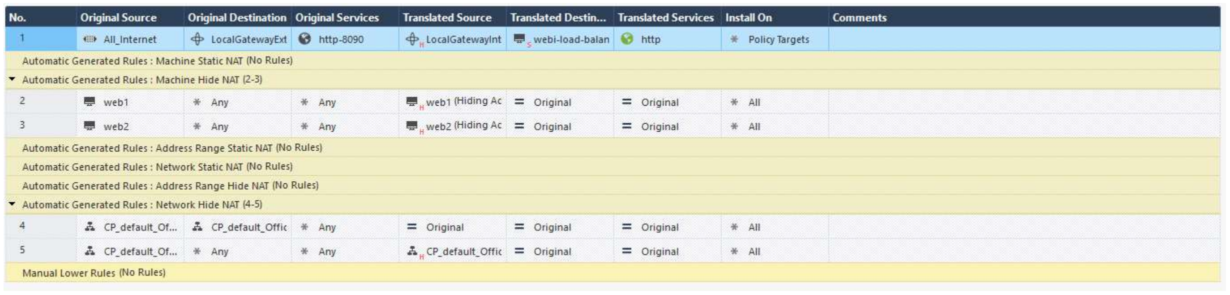
- Protocol Signature is disabled

- Access Rules:



No.	Name	Source	Destination	VPN	Services & Applications	Action	Track	Install On
1	Allow HTTP	Any	LocalGatewayExternal	Any	http-8090	Accept	Log	Policy Targets
2	Allow SSH	Any	LocalGatewayExternal	Any	ssh	Accept	Log	Policy Targets
3	Allow Outbound Internet	web1 web2	Any	Any	http https dns	Accept	Log	Policy Targets
4	Drop All	Any	Any	Any	Any	Drop	Log	Policy Targets

- NAT Rules:



No.	Original Source	Original Destination	Original Services	Translated Source	Translated Destin...	Translated Services	Install On	Comments
1	All_Internet	LocalGatewayExt	http-8090	LocalGatewayInt	webi-load-balan	http	Policy Targets	
Automatic Generated Rules : Machine Static NAT (No Rules)								
Automatic Generated Rules : Machine Hide NAT (2-3)								
2	web1	Any	Any	web1 (Hiding Ac	Original	Original	All	
3	web2	Any	Any	web2 (Hiding Ac	Original	Original	All	
Automatic Generated Rules : Address Range Static NAT (No Rules)								
Automatic Generated Rules : Network Static NAT (No Rules)								
Automatic Generated Rules : Address Range Hide NAT (No Rules)								
Automatic Generated Rules : Network Hide NAT (4-5)								
4	CP_default_Of...	CP_default_Offic	Any	Original	Original	Original	All	
5	CP_default_Of...	Any	Any	CP_default_Offic	Original	Original	All	
Manual Lower Rules (No Rules)								

Appendix C – Useful Git Commands

In this section, I would like to detail a few useful actions and commands RE git and github.

- **Forking** – as mentioned earlier, the main code repo is stored here:

https://github.com/philipatkinson86/azure-checkpoint-automation

If you have a separate github account, you can easily 'fork' or make a copy of this repo to your own account by hitting the following button:

- **Pull from your repo** – Once copied, the code can be downloaded locally as a zip file or if you have git installed on Ubuntu (can be done by running '`apt install git`'), you can do the following:
    - Create a new folder on ubuntu
    - cd to that folder
    - Add your forked URL as origin:

```
git remote add origin https://github.com/USERNAME/REPOSITORY_NAME.git
```

    - Run the following to download the files:

```
git pull origin master
```

- **Making changes** – If you make some changes to your local repository an want to sync them back to your github online repository, you can run the following commands to do this:

```
git add -A

git commit -m "comment"

git push -u origin master
```