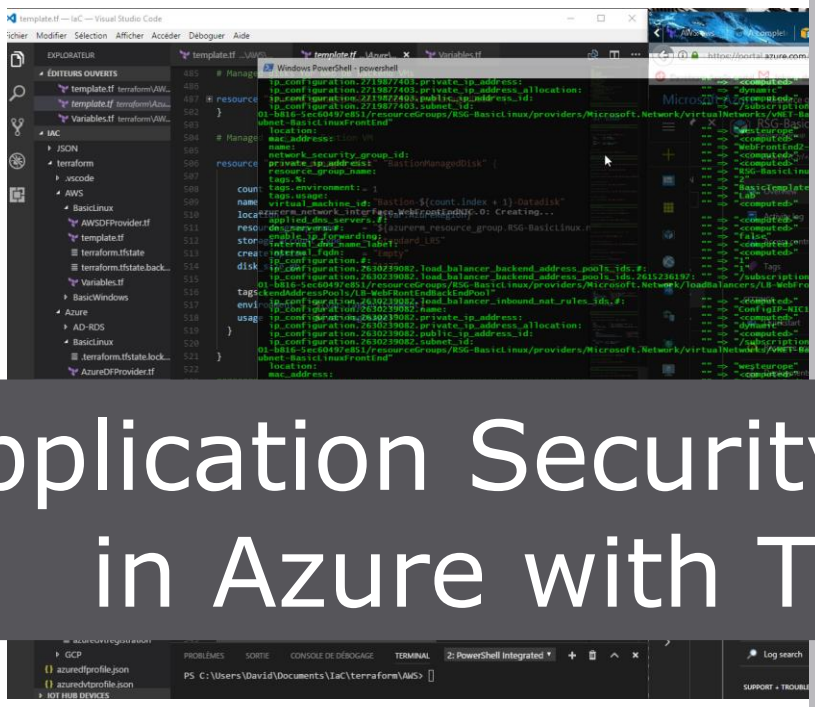


Teknews.cloud

Application Security Groups in Azure with Terraform

David Frappart

21/01/2018

Contributor

Name	Title	Email
David Frappart	Cloud Architect	david@teknews.cloud

Version Control

Version	Date	State
1.0	2018/08/07	Final

Table of Contents

1 Introduction

When dealing with Network filtering with Azure, we have the Network Security Group (NSG) which provides a basic way of defining Network rules, in a distributed way, as a stateless firewall. I wrote a previous article defining tips in the usage of those, available here.
Up until now, we could attach a NSG to the subnet level, and it was necessary to play with a combination of IP or IP range or to add additional NSG applied at the NIC level of VMs requiring different sets of rules.
While it does the job, let's say that from time to time, it could be a little headache to manage all those NSG and rules. To simplify all this, enters the Application Security groups

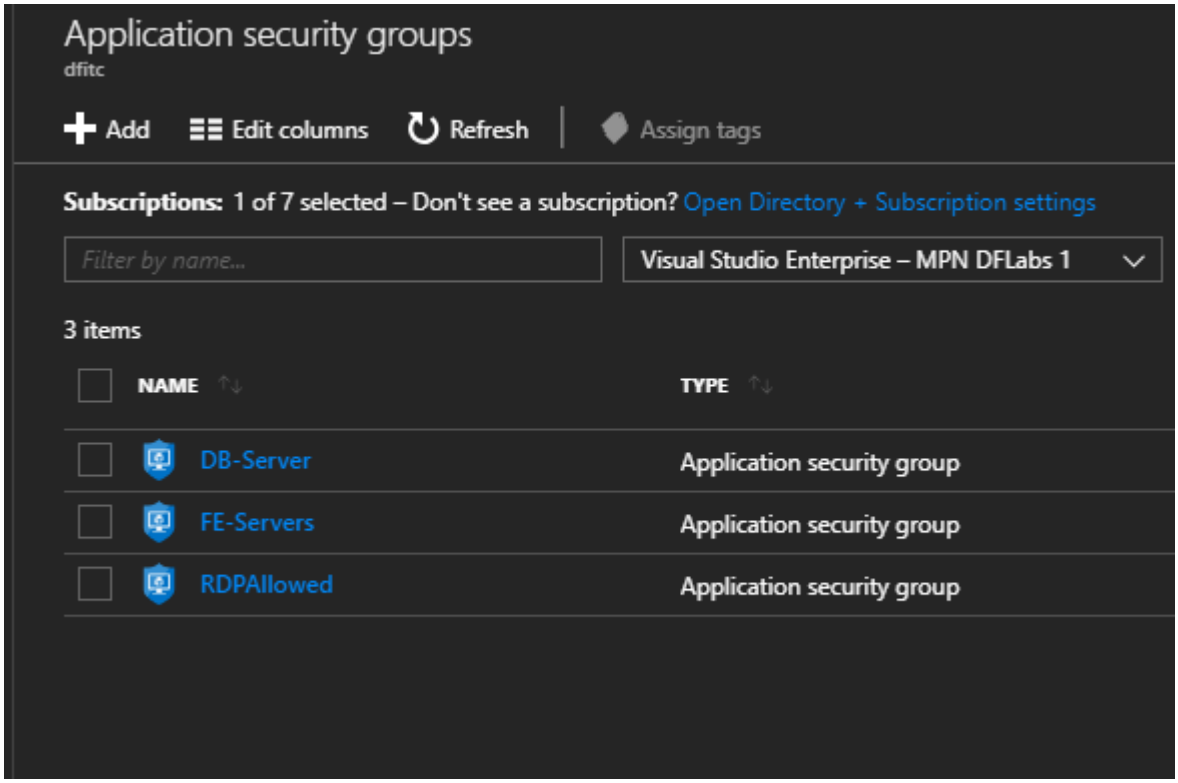
2 Application Security Groups concepts

The application Security Groups (ASG) have been in preview for some times and are available in GA since… sometimes. However, it was not immediately available in Terraform and also, I had not the opportunity to use those immediately.
The concept is really quite simple. With ASG, we are able to create a kind of label which will be available to define Network Security rules. For a comparison this is an equivalent of a custom service tag.
For example, let's imagine that we have a 2 tiers application with front-end web servers and back-end database servers. Often, we use different subnets and attach a different NSG.
The ASG offers an elegant way of defining rules on the same NSG without the headache of the source and target IP.
First, we define a Front-End Web ASG for the front-end web servers and a DB ASG for the database servers. Next, we define the rule to allow the required traffic on the web servers (for example TCP 80/443 from Internet) and on the Database servers (for example TCP 1433-1434 only from web servers).
WE have granular rules, not dependant of a full IP range, defining the traffic that we require.
Let's have a look in the portal how it's look like.

3 Peek in ASG on Azure Portal

Ok, let's try the steps described earlier. First, in the portal, we will create some ASG:

- 1 ASG to target Front-End servers
- 1 ASG to target Database servers
- 1 ASG to target servers reachable through RDP from the Internet



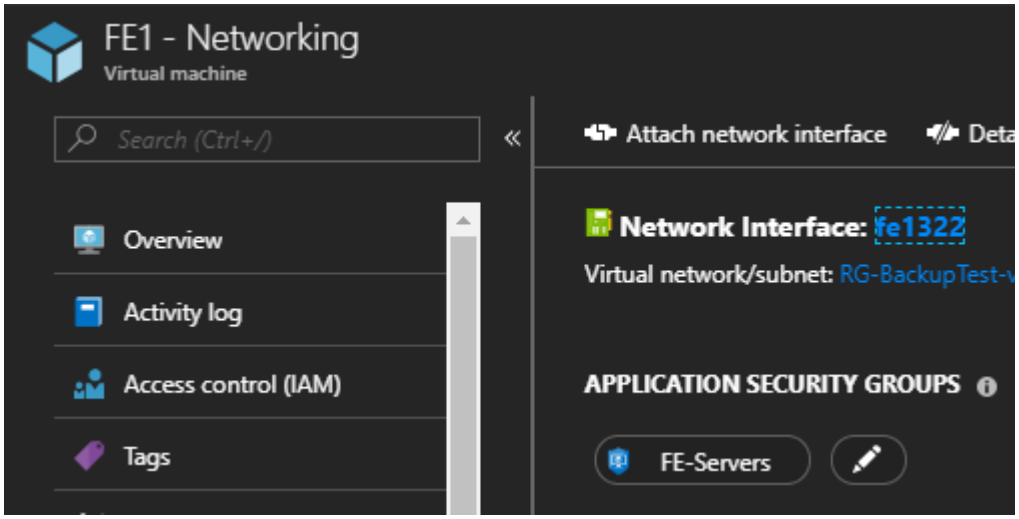
Then we will create rules using the ASG for FE servers and the ASG for Database servers:

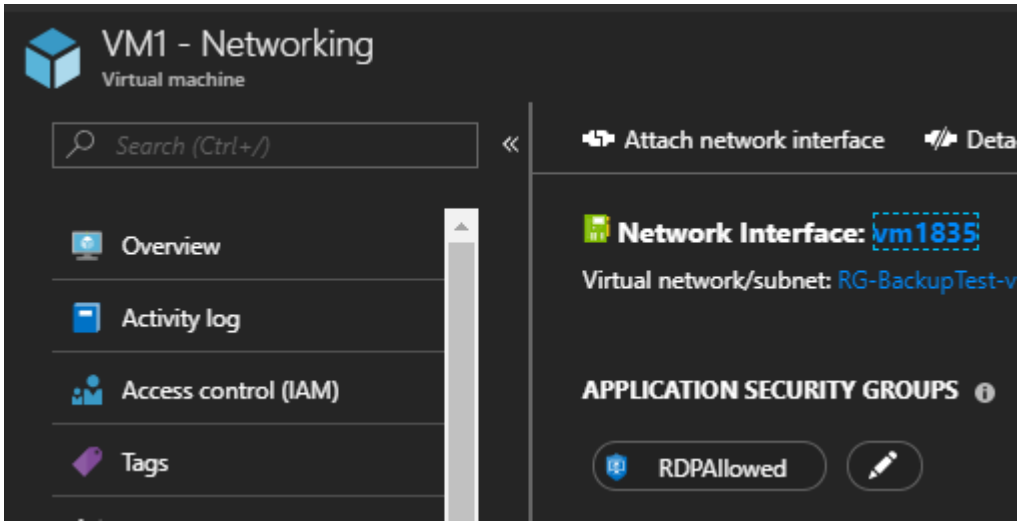
- 1 rule to allow RDP from Internet to ASG RDPAllowed
- 1 rule to allow TCP 80 and 443 to ASG FE-Servers
- 1 rule to allow TCP 1433 from ASG FE-Servers to ASG DB-Servers
- 1 rule to allow RDP from ASG RDPAllowed to ASG DB-Server

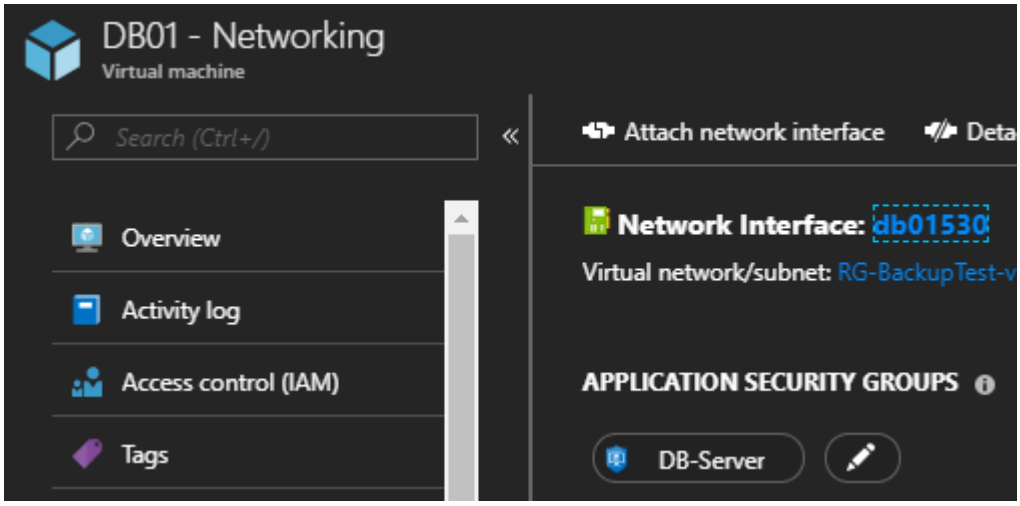
In addition, we add a deny rule for all traffic to DB-Server.

PRIORITY	NAME	PORT	PROTOCOL	SOURCE	DESTINATION	ACTION
100	RDPIn	3389	Any	Internet	RDPAllowed	Allow
110	http-s	80,443	Any	Internet	FE-Servers	Allow
120	MSSQL	1433-1434	Any	FE-Servers	DB-Server	Allow
130	RDPtoDB	3389	Any	RDPAllowed	DB-Server	Allow
140	DenyAllDB	Any	Any	Any	DB-Server	Deny
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

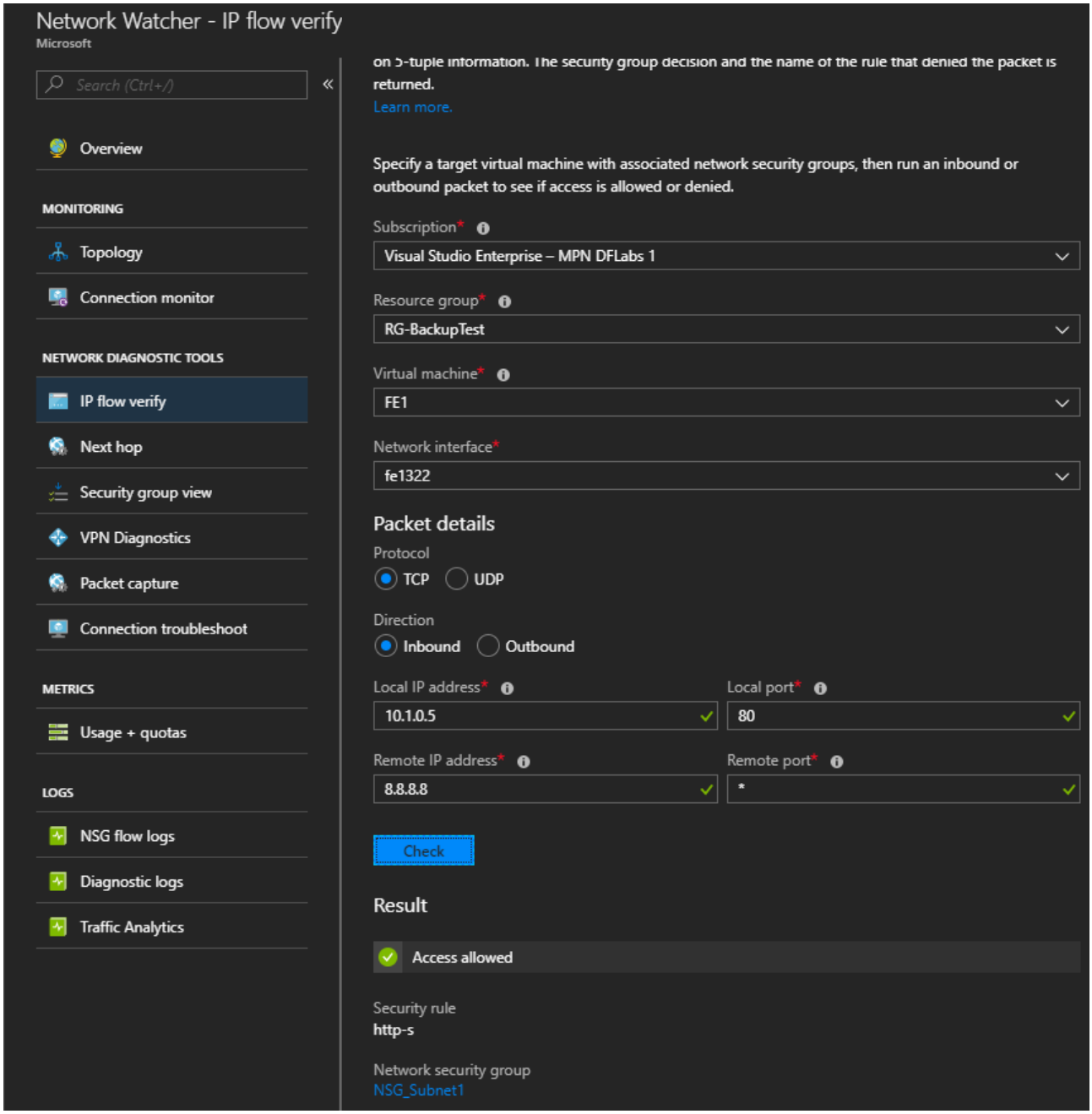
Next, we attach the ASG to VMs:


FE1 - Networking
Virtual machine
Search (Ctrl+/)
Attach network interface
Overview
Activity log
Access control (IAM)
Tags
Network Interface: fe1322
Virtual network/subnet: RG-BackupTest-v
APPLICATION SECURITY GROUPS
FE-Servers

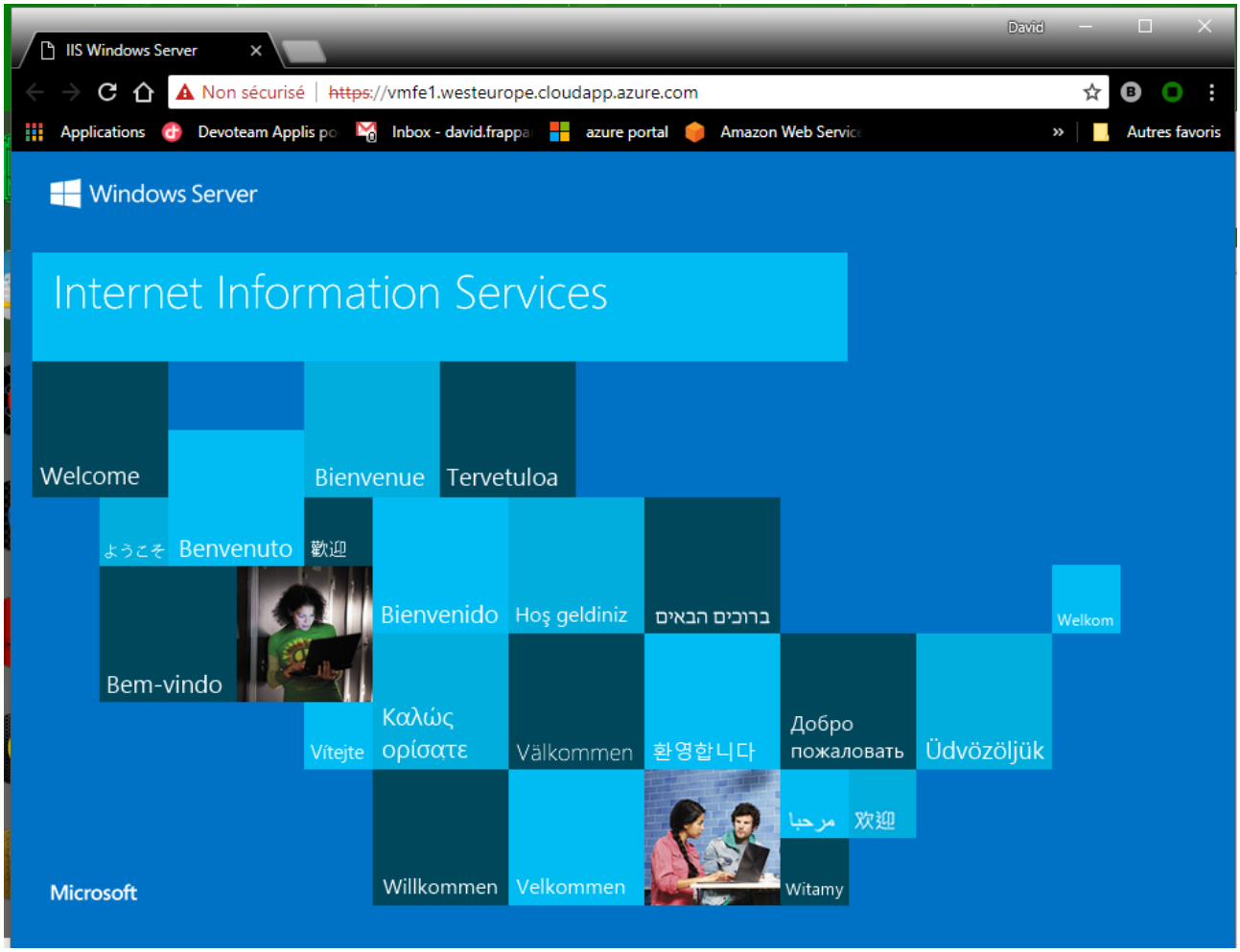

VM1 - Networking
Virtual machine
Search (Ctrl+/)
Attach network interface
Overview
Activity log
Access control (IAM)
Tags
Network Interface: vm1835
Virtual network/subnet: RG-BackupTest-v
APPLICATION SECURITY GROUPS
RDPAllowed


DB01 - Networking
Virtual machine
Search (Ctrl+/)
Attach network interface
Overview
Activity log
Access control (IAM)
Tags
Network Interface: db01530
Virtual network/subnet: RG-BackupTest-v
APPLICATION SECURITY GROUPS
DB-Server

And to finish we evaluate the access with Network Watcher. Below an example for the flow from Internet to TCP 80 on FE server:



And directly for the front end:



4 ASG and Terraform

Ok, now that the ASG concepts are refreshed, let's see how to do it through Terraform. Available to us is the dedicated resource for the application security group azurerm_application_security_group. For the remaining aspect of coding ASG, we rely on NSG rules parameters and Network interface parameters. Let's have a look.

4.1 A module for ASG usage

Below is a module to make use of ASG:

```
#####################################################################
#This module allows the creation of an Application Security
#Group
#####################################################################

#Variable declaration for Module

variable "ASGName" {
  type    = "string"
  default = "DefaultNSG"
}

variable "RGName" {
  type    = "string"
  default = "DefaultRSG"
}

variable "ASGLocation" {
  type    = "string"
  default = "Westeurope"
}

variable "EnvironmentTag" {
  type    = "string"
  default = "Poc"
}

variable "EnvironmentUsageTag" {
  type    = "string"
  default = "Poc usage only"
}

#Creation of the ASG
resource "azurerm_application_security_group" "Terra-ASG" {
  name                = "${var.ASGName}"
  location            = "${var.ASGLocation}"
  resource_group_name = "${var.RGName}"

  tags {
    environment = "${var.EnvironmentTag}"
    usage       = "${var.EnvironmentUsageTag}"
  }
}
```

```
#Output for the ASG module

output "Name" {
  value = "${azurerm_application_security_group.Terra-ASG.name}"
}

output "Id" {
  value = "${azurerm_application_security_group.Terra-ASG.id}"
}

output "RGName" {
  value = "${azurerm_application_security_group.Terra-ASG.resource_group_name}"
}
```

This module is quite simple and comes with few parameters. Indeed, as for the portal actions, an ASG is just a logical object that is used to tag resource for network rules. We do need however to output the id of the terraform created resource. And we need to change the config for the NIC.

4.2 Adapting the config for the NIC

The fun fact here, is that while we associate ASG to VM on the portal, we associate ASG to NIC on Terraform. It is as simple as adding a single line to the NIC's IP config, with a parameter called application_security_group_ids. This parameter is expecting a list type.

```
resource "azurerm_network_interface" "TerraNICwpip" {
  count               = "${var.NICCount}"
  name                = "${var.NICName}${count.index+1}"
  location            = "${var.NICLocation}"
  resource_group_name = "${var.RGName}"

  ip_configuration {
    name                          = "ConfigIP-NIC-${var.NICName}"
    subnet_id                     = "${var.SubnetId}"
    private_ip_address_allocation  = "dynamic"
    public_ip_address_id          = "${element(var.PublicIPId,count.index)}"
    primary                       = "${var.Primary}"
    application_security_group_ids = ["${var.ASGIds}"]
  }

  tags {
    environment = "${var.EnvironmentTag}"
```

```
    usage       = "${var.EnvironmentUsageTag}"
  }
}
```

After that, we have to define the NSG rules to finally make usage of the ASG.

4.3 NSG rules

For the NSG rule object, optional parameters to use ASG either as a source or as a target are available. Those parameters are:

- source_application_security_group_ids
- destination_application_security_group_ids

Those parameters expect also a list. Below is a module which expect as a destination an ASG:

```
######################################################################
#This module allows the creation of a Network Security Group Rule
######################################################################

#Variable declaration for Module

# The NSG rule requires a RG location in which the NSG for which the rule is
created is located
variable "RGName" {
  type    = "string"
  default = "DefaultRSG"
}

#The NSG rule requires a reference to a NSG
variable "NSGReference" {
  type = "string"
}

#The NSG Rule Name, a string value allowing to identify the rule after deployment
variable "NSGRuleName" {
  type    = "string"
  default = "DefaultNSGRule"
}

#The NSG rule priority is an integer value defining the priority in which the rule
is applyed in the NSG
variable "NSGRulePriority" {
  type = "string"
```

```
}

#The NSG rule direction define if the rule is for ingress or egress trafic. Allowed
value are inbound and outbound
variable "NSGRuleDirection" {
  type = "string"
}

#The NSG Rule Access value,  a string value defining if the rule allow or block the
specified traffic. Accepted value are Allow or Block
variable "NSGRuleAccess" {
  type    = "string"
  default = "Allow"
}

#The NSG rule protocol define which type of trafic to allow/block. It accept the
string tcp, udp, icmp or *
variable "NSGRuleProtocol" {
  type = "string"
}

#The NSG rule source port range define the port(s) from which the trafic origing is
allowed/blocked
variable "NSGRuleSourcePortRange" {
  type    = "string"
  default = "*"
}

#The NSG rule destination port range define the port(s) on which the trafic
destination is allowed/blocked
variable "NSGRuleDestinationPortRange" {
  type = "string"
}

#The NSG rule address prefix defines the source address(es) from which the trafic
origin is allowed/blocked
variable "NSGRuleSourceAddressPrefix" {
  type = "string"
}

#The NSG rule address preifx defines the source address(es) from which the trafic
origin is allowed/blocked
variable "NSGRuleDestinationASG" {
  type = "list"
```

```
}

# creation of the rule

resource "azurerm_network_security_rule" "Terra-NSGRulewDestASG" {
  name                                        = "${var.NSGRuleName}"
  priority                                    = "${var.NSGRulePriority}"
  direction                                   = "${var.NSGRuleDirection}"
  access                                      = "${var.NSGRuleAccess}"
  protocol                                    = "${var.NSGRuleProtocol}"
  source_port_range                           = "${var.NSGRuleSourcePortRange}"
  destination_port_range                      = "${var.NSGRuleDestinationPortRange}"
  source_address_prefix                       = "${var.NSGRuleSourceAddressPrefix}"
  destination_application_security_group_ids = ["${var.NSGRuleDestinationASG}"]
  resource_group_name                         = "${var.RGName}"
  network_security_group_name                 = "${var.NSGReference}"
}

# Module output

output "Name" {
  value = "${azurerm_network_security_rule.Terra-NSGRulewSTags.name}"
}

output "Id" {
  value = "${azurerm_network_security_rule.Terra-NSGRulewSTags.id}"
}
```

The coding is quite simple. However, while those parameters are optional, it would be logical that those cannot coexist with either the source_address_prefix(es) and destination_address_prefix(es). So, the module is not so much reusable except if we write one with conditional depending of what we want to address as a rule. Presently, I have not tested a code with such conditional.

5 Conclusion

In this article, we covered the ASG concepts and how to deploy ASG and NSG rule taking this ASG into account. The seemingly next step should be to try out a module with conditional for the NSG rule if there is or not ASG or Ip range for source and target. I may work on that if I happen to have some time. Until then…

Another Tech blog

My thoughts and experiences on Cloud related tek