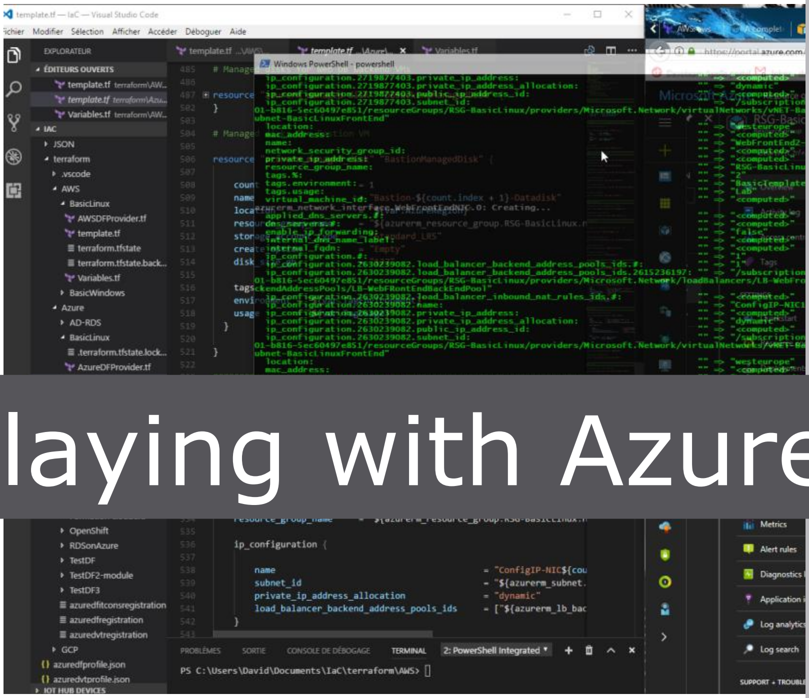


Teknews.cloud

Playing with Azure Firewall

David Frappart

09/08/2018

Contributor

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

Version Control

Version	Date	State
1.0	2018/08/09	Final

Table of Contents

1 Introduction

While Cloud concepts rely heavily on distributed architecture, IT people still live (for now) in a world where some legacy concepts apply. For example, it is frequently required that outbound flow should go through a proxy, or at least that filtering capabilities on URL are available for front-end servers.
Usually, it means that we have to depends on IaaS proxy since Azure does not provide a managed proxy service.
Except it does, since the Azure Firewall is out. Let's have a look

2 Azure Firewall concepts

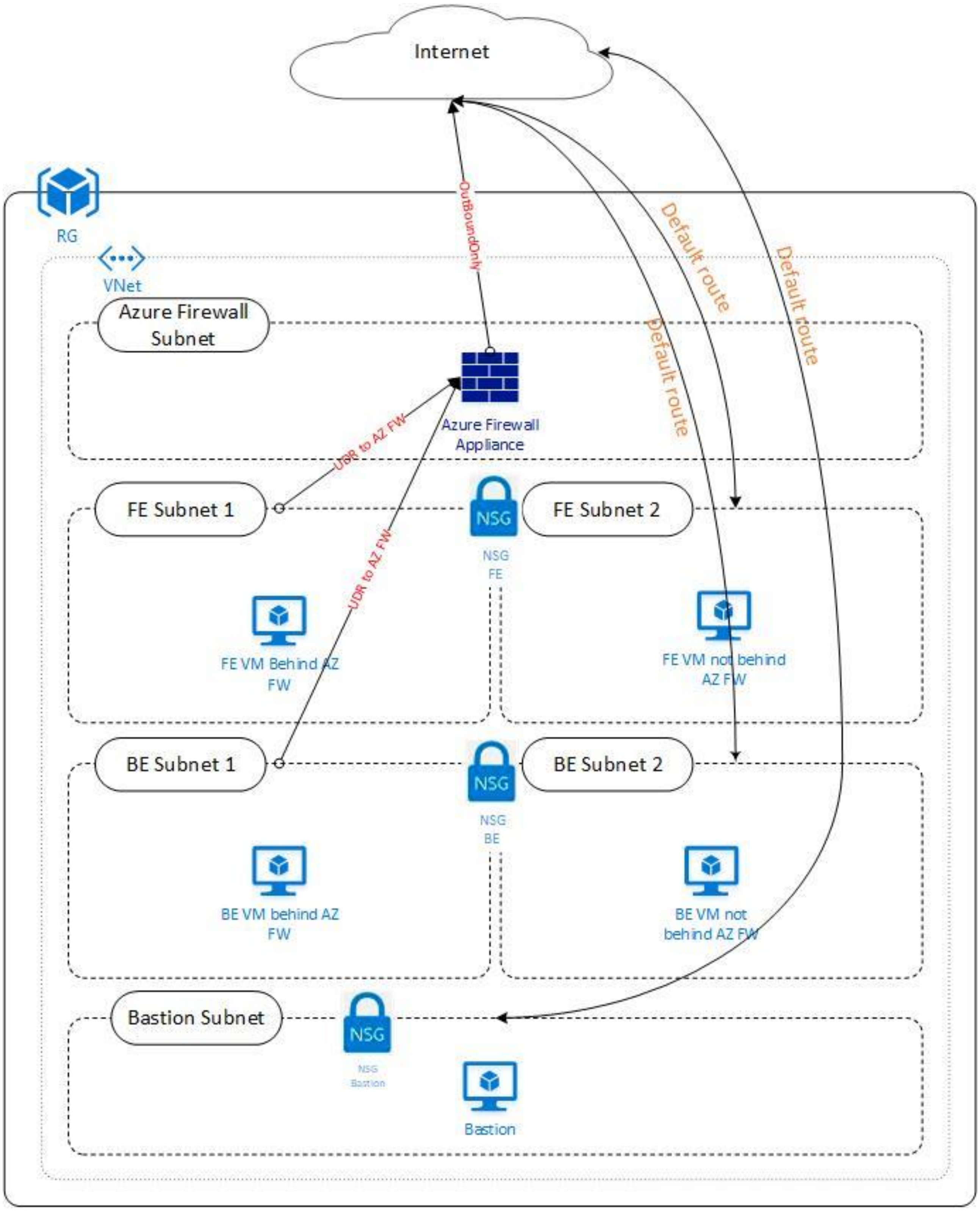
The Azure Firewall is an Azure managed service providing firewalling capabilities through a virtual appliance. While the NSG are distributed firewall applying rules to the NIC Level, the Azure Firewall act as an appliance firewall and it is required to add a User Defined Route to force traffic from subnet through it.
The nice point as opposed to IaaS Firewall appliance, is that it is managed on Azure side. Which means that (once it is in GA), Microsoft will take responsibility for the availability of this NFV. The HA is thus built-in.
On a technical point of view, it is still a stateful Firewall. But It allows to add rules comprised in between L3 and L7. We are thus able to filter traffic to certain URI if required, which we could not do with only NSG.
Also, with the support for SNAT, it becomes possible to identify and allow traffic originating from Vnet to remote Internet destinations.
Strange thing, while it is named Firewall, it does not allow inbound traffic. It is aimed solely to provide outbound traffic filtering. In case of L7 Inbound protection, we should use the Application gateway, which is GA since quite some time.
Last but not least, the Azure Firewall has to live in its own subnet, which should be at minimum a CIDR range /25, and on which no NSG should be applied.

3 Architecture playground

In this article, we will play with a dedicated Azure environment.
This environment will be composed of the classic building blocks:

- A VNet
- Front-End Subnets with VMs accessible from internet
- Back-End subnets
- 1 dedicated subnet for the Azure Firewall
- 1 bastion subnet to access the VMs on their private IP address
- NSG and ASG
- A bunch of VMs in each subnet to conduct some testing.

Below is a schema of the playground:



Now that the playground is defined, let's have a look at some IaC.

4 JSON syntax for provisioning Azure Firewall

The Azure Firewall being still in preview, it is one of those case where using the terraform template deployment is necessary. Below is the JSON part I use, which comes from a quick start template and that I modified to match my usage:

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "location": {
          "type": "string",
          "metadata": {
            "description": "Location for all resources."
          }
        },
        "aZFWName": {
            "type": "string",
            "defaultValue": "AZFWPoc",
            "metadata": {
              "description": "Azure FW Appliance"
            }
          },
        "aZFWSubnetId": {
            "type": "string",
            "metadata": {
              "description": "Subnet in which resides the AZ FW"
            }
          },
        "fESubnetRange": {
          "type": "string",
          "metadata": {
            "description": "Subnet Front-End protected by the AZ FW"
          }
        },
        "bESubnetRange": {
          "type": "string",
          "metadata": {
            "description": "Subnet Back-End protected by the AZ FW"
          }
        },
        "aZFWPublicIpId": {
            "type": "string",
            "metadata": {
              "description": "Id of the AZ FW PIP"
            }
          }
      },
    "resources": [
        {
            "apiVersion": "2018-06-01",
            "type": "Microsoft.Network/azureFirewalls",
            "name": "[parameters('aZFWName')]",
            "location": "[parameters('location')]",
            "properties": {
              "ipConfigurations": [
                {
                  "name": "IpConf",
                  "properties" : {
                    "subnet": {
                      "id": "[parameters('aZFWSubnetId')]"
                    },
                    "InternalPublicIPAddress": {
                      "id": "[parameters('aZFWPublicIpId')]"
                    }
                  }
                }
              ],
              "applicationRuleCollections": [
                {
                  "name": "appRc1",
                  "properties": {
                    "priority": 101,
                    "action": {"type": "Allow"},
                    "rules": [
                      {
                        "name": "appRule1",
                        "protocols": [{"port": "80", "protocolType": "http"},
{"port": "443", "protocolType": "https"}],
                        "targetUrls": ["*microsoft.com","*github.com"]
                      }
                    ]
                  }
                }
              ],
              "networkRuleCollections": [
                {
                  "name": "NetworkRulesCollection1",
                  "properties": {
                    "priority": 200,
                    "action":  { "type":  "Allow"},
                    "rules": [
                      {
                        "name": "FERule1",
                        "protocols": ["TCP"],
                        "sourceAddresses": ["[parameters('fESubnetRange')]"],
                        "destinationAddresses": ["*"],
                        "destinationPorts": ["80","443","53"]
                      },
                      {
                        "name": "BERule1",
                        "protocols": ["TCP"],
                        "sourceAddresses": ["[parameters('bESubnetRange')]"],
                        "destinationAddresses": ["*"],
                        "destinationPorts": ["80","443","53"]
                      },

                      {
                        "name": "InboundHTTP",
                        "protocols": ["TCP"],
                        "sourceAddresses": ["0.0.0.0/0"],
                        "destinationAddresses": ["[parameters('fESubnetRange')]"],
                        "destinationPorts": ["80","443","8080"]
                      }
                    ]
                  }
                }
              ]
          }
        }
    ]
}
```

Since I am only using a small JSON template, I do not (and in truth, do not know how to) use variables in the template. I do however use parameters, which are inserted to the template by terraform interpolation through the azurerm_template_deployment.
This ARM resource is described by its type Microsoft.Network/azureFirewalls, its name, its location and a set of properties.

In the properties, we find the ipConfigurations, the applicationRuleCollections and the networkRuleCollections.
Easily enough, we understand that the applicationRuleCollections is what gives us the capabilities outgoing traffic at the level 7, while the networkRuleCollections is what gives us the capabilities to create level 4 rules.
In the network rule collection 1, I added a rule which supposedly allows inbound from internet to the Front-End. We will test it but since Azure Firewall is only for outgoing traffic, it should not work.
Below is the code to invoke this template with the azurerm_template resource:

```
data "template_file" "templateAZFW" {
  template = "${file("./Templates/templateazfw.json")}"
}

resource "azurerm_template_deployment" "Template-AZFW" {
  name                = "azurefwtemplate"
  resource_group_name = "${module.ResourceGroupInfra.Name}"

  template_body = "${data.template_file.templateAZFW.rendered}"

  parameters {
    "location"       = "${var.AzureRegion}"
    "aZFWSubnetId"   = "${module.FW_Subnet.Id}"
    "aZFWPublicIpId" = "${element(module.FW_PIP.Ids,0)}"
    "fESubnetRange"  = "${lookup(var.SubnetAddressRange, 0)}"
    "bESubnetRange"  = "${lookup(var.SubnetAddressRange, 2)}"
  }

  deployment_mode = "Incremental"
}
```

The template resource is used to hide from my terraform .tf file the JSON which is read as a long string.
IT means that in this case, the firewall JSON configuration is located in the file templateazfw.json.

5 Deploying the infrastructure with Terraform

The Terraform configuration to deploy the infrastructure is available on my github here. We will not go in the detail of each module here, only those that are new in my repo. Which means we will look at the route table and routes, but also at the impact on NSG and subnet config.

5.1 Route and route table modules

Until now, even if I used UDR a few times, I did not push my terraform code to any Github repo. So let’s have a look at how it works.
AS a reminder, by default, a default route applies on Network inside a VNet. TO change the default behavior or routing, we use the User Defined Route, which we then attach to a subnet with the desired route to define a traffic which differ from the default behavior.
With Terraform, we have 2 resources to define and create:

- The route table with azurerm_route_table
- The route with azurerm_route

5.1.1 Terraform for route table

The route table is defined as follow:

```
resource "azurerm_route_table" "TerraRouteTable" {
  name                          = "${var.RouteTableName}"
  location                      = "${var.RTLocation}"
  resource_group_name           = "${var.RGName}"
  disable_bgp_route_propagation = "${var.BGPDisabled}"

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

We have a few parameters available which are name, location, resource_group_name and disable_bgp_route_propagation. We can also add tags, which we should do.
Missing from this extract of code is the route code block. Similarly, to VNet and Subnet or NSG and NSG rules, it is possible to insert the route inside the route table resource. However, here we keep those in different module. In my point of view, it gives us more flexibility in the provisioning and in the use of our module.
The output available are the Route Table Id and the associated subnet. In my code, I also add outputs for the name and the resource group:

```
#Output

output "Name" {
  value = "${azurerm_route_table.TerraRouteTable.name}"
}

output "Id" {
  value = "${azurerm_route_table.TerraRouteTable.id}"
}

output "RGName" {
  value = "${azurerm_route_table.TerraRouteTable.resource_group_name}"
}

output "Subnet" {
  value = "${azurerm_route_table.TerraRouteTable.subnets}"
}
```

The module is called as follow:

```
module "RouteTable" {
  #Module location
  source = "./Modules/17 RouteTable"

  #Module variable
  RouteTableName       = "RouteTabletoAzFW"
  RGName               = "${module.ResourceGroupInfra.Name}"
  RTLocation           = "${var.AzureRegion}"
  BGPDisabled          = "false"
  EnvironmentTag       = "${var.EnvironmentTag}"
  EnvironmentUsageTag = "${var.EnvironmentUsageTag}"
}
```

With, in this example, a local copy of the module available. After that we need to define the route to add in this route table.

5.1.2 Terraform for route

To create the route, we will use the azurerm_route resource. The aim here is to force traffic through our Azure Firewall. The resource is created as follow;

```
# Route table creation

resource "azurerm_route" "TerraRoute" {
  name                   = "${var.RouteName}"
  route_table_name       = "${var.RTName}"
  resource_group_name    = "${var.RGName}"
  address_prefix         = "${var.DestinationCIDR}"
  next_hop_type          = "${var.NextHop}"
  next_hop_in_ip_address = "${var.NextHopinIPAddress}"
}
```

The address_prefix parameter define the destination network for which we want to define the route behavior. In this case, we are talking about Internet traffic, so the value should be 0.0.0.0/0.
The next_hop_type define the type available in Azure. We can put the values as follow:

- VirtualNetworkGateway
- VnetLocal
- Internet
- VirtualAppliance
- None

In the case here, as we defined the Azure Firewall as an appliance, we will use the VirtualAppliance hop type.
Lest, the next_hop_in_ip_address is the corresponding ip of the virtual appliance here. AS of now, I do not have a satisfying way of referring the private ip of the Azure Firewall. However, we use a dedicated subnet for this Firewall, and following Azure provisioning logic, we know that it should use the first available IP address of the range. Thus in our case, we will use the built-in function cidrhost().
This gives us the following to define the route with the module:

```
module "Route" {
  #Module location
  source = "./Modules/16 Route"

  #Module variable
  RouteName            = "RoutetoAzFW"
  RGName               = "${module.ResourceGroupInfra.Name}"
  RTName               = "${module.RouteTable.Name}"
  DestinationCIDR      = "0.0.0.0/0"
  NextHop              = "VirtualAppliance"
  NextHopinIPAddress = "${cidrhost(var.SubnetAddressRange[3],4)}"
}
```

Ok, now the route and route table are defined, but attached to nothing. We need to add the rout table id to a subnet object to apply the route.

5.2 Subnet with UDR

We have been playing with module for Subnet for quite some time now. But until now, we did not use UDRs so it was missing in the subnet terraform config. Below is the code for a subnet with this parameter added.

```
resource "azurerm_subnet" "TerraSubnet" {
  name                       = "${var.SubnetName}"
  resource_group_name        = "${var.RGName}"
  virtual_network_name       = "${var.vNetName}"
  address_prefix             = "${var.Subnetaddressprefix}"
  network_security_group_id = "${var.NSGid}"
  service_endpoints         = "${var.SVCEP}"
  route_table_id            = "${var.RouteTableId}"
}
```

In this module, we only added the route_table_id. Adding e default value of null to the variable RouteTableId to keep only one module did not work for me, so I had to create a new module specifically for this case.
Calling the module is done as follow:

```
module "BE_Subnet1" {
  #Module location
  source = "./Modules/06-3 Subnet with routetable"

  #Module variable
  SubnetName            = "${lookup(var.SubnetName, 2)}"
  RGName                = "${module.ResourceGroupInfra.Name}"
  vNetName              = "${module.SampleArchi_vNet.Name}"
  Subnetaddressprefix = "${lookup(var.SubnetAddressRange, 2)}"
  NSGid                 = "${module.NSG_BE_Subnet.Id}"
  RouteTableId          = "${module.RouteTable.Id}"
  EnvironmentTag        = "${var.EnvironmentTag}"
  EnvironmentUsageTag = "${var.EnvironmentUsageTag}"
}
```

5.3 A check on the provision duration

Here just to have an idea, I got the output from the terraform apply for the Azure firewall template deployment:

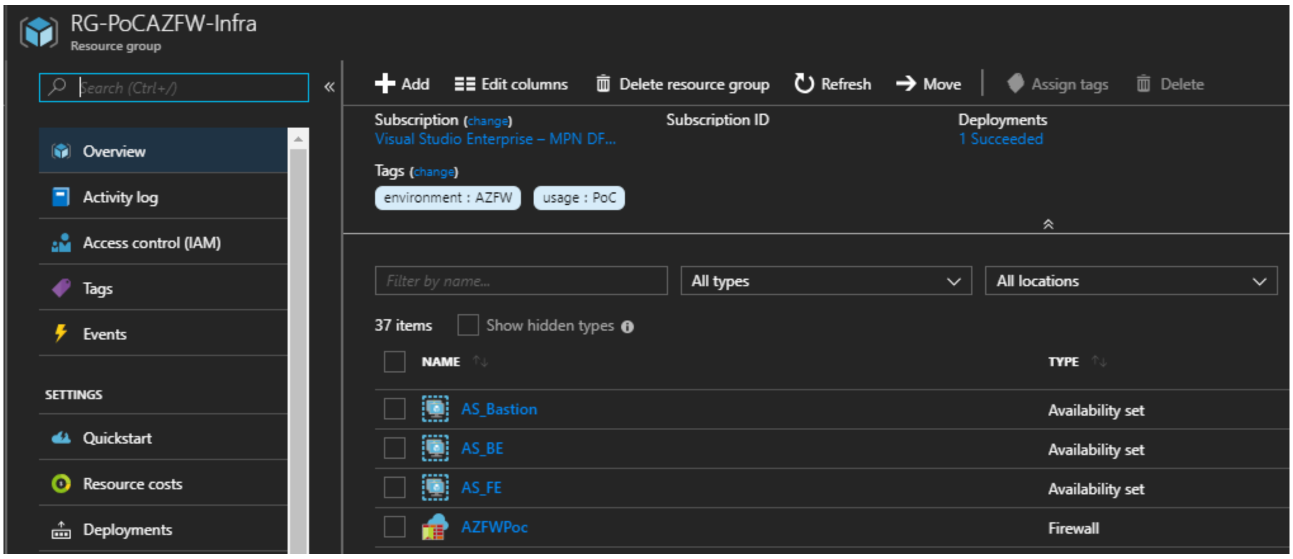
```
azurerm_template_deployment.Template-AZFW: Creation complete after 5m2s (ID:
/subscriptions/0cfadfdd-0ccd-468a-bdf4-....Resources/deployments/azurefwtemplate)
```

So if things go well, we should have an available firewall after a bunch of minutes, ready to use. That's not too long and that's pretty cool.

6 Let's play

6.1 The Azure Firewall in the portal

Once the provisioning is completed, we should have a big bunch of object and 1 registered deployment, as displayed below:

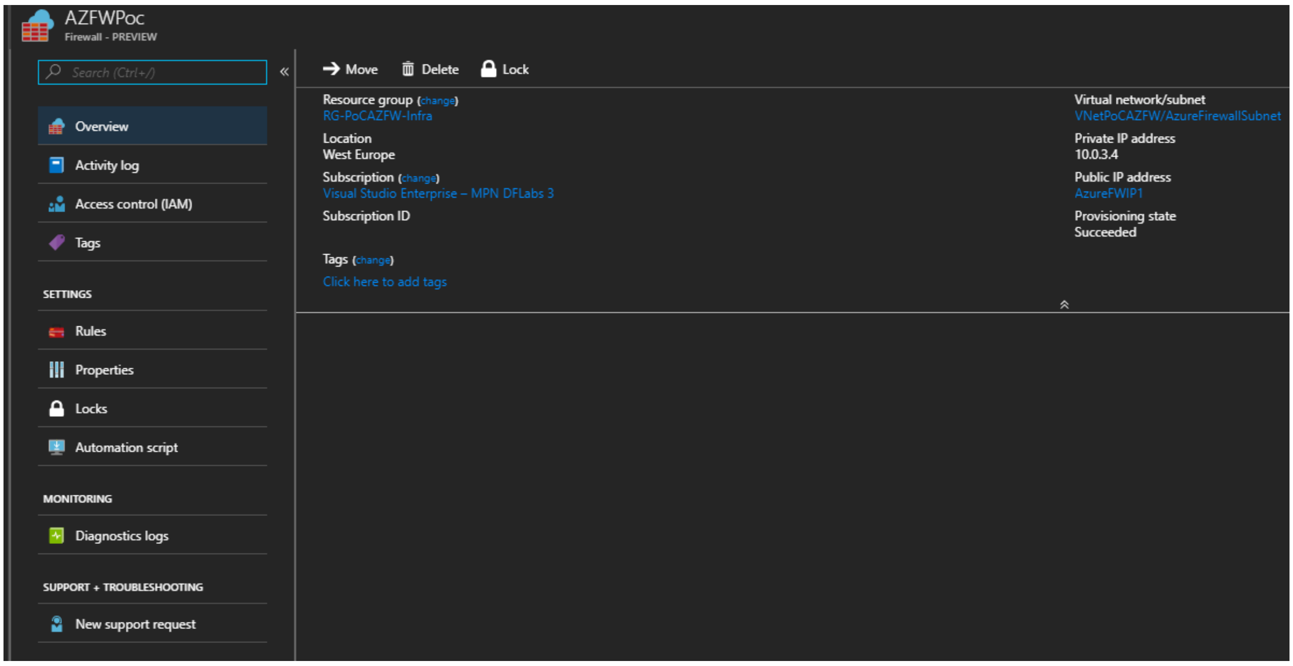

RG-PoCAZFW-Infra
Resource group

Subscription (change)
Visual Studio Enterprise – MPN DF...
Subscription ID
Deployments
1 Succeeded
Tags (change)
environment : AZFW
usage : PoC

37 items

NAME	TYPE
AS_Bastion	Availability set
AS_BE	Availability set
AS_FE	Availability set
AZFWPoc	Firewall

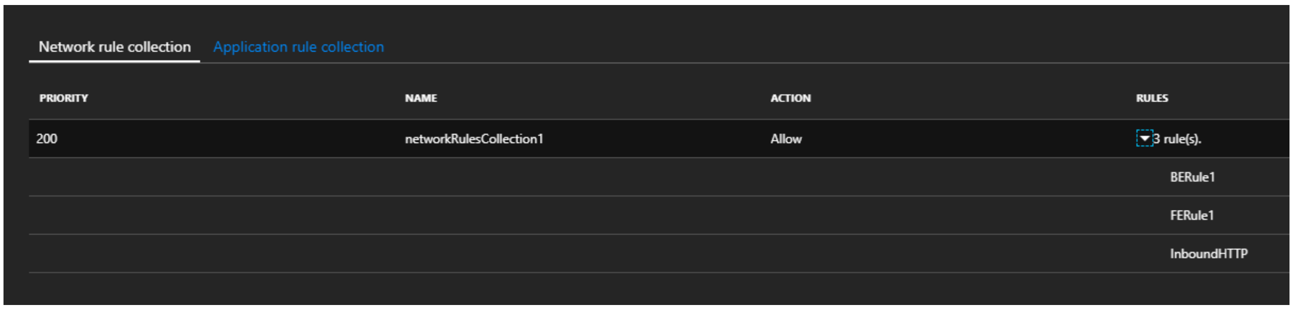
We can see below deployment 1 succeeded and ahte AZFWPoC object which is categorized as a firewall type.
Clocking on the object brings us to the following screen:


AZFWPoc
Firewall - PREVIEW

Resource group (change)
RG-PoCAZFW-Infra
Location
West Europe
Subscription (change)
Visual Studio Enterprise – MPN DFLabs 3
Subscription ID
Tags (change)
Click here to add tags
Virtual network/subnet
VNetPoCAZFW/AzureFirewallSubnet
Private IP address
10.0.3.4
Public IP address
AzureFWIP1
Provisioning state
Succeeded

The interesting part is in the Rules section. There are 2 available kind of rules:

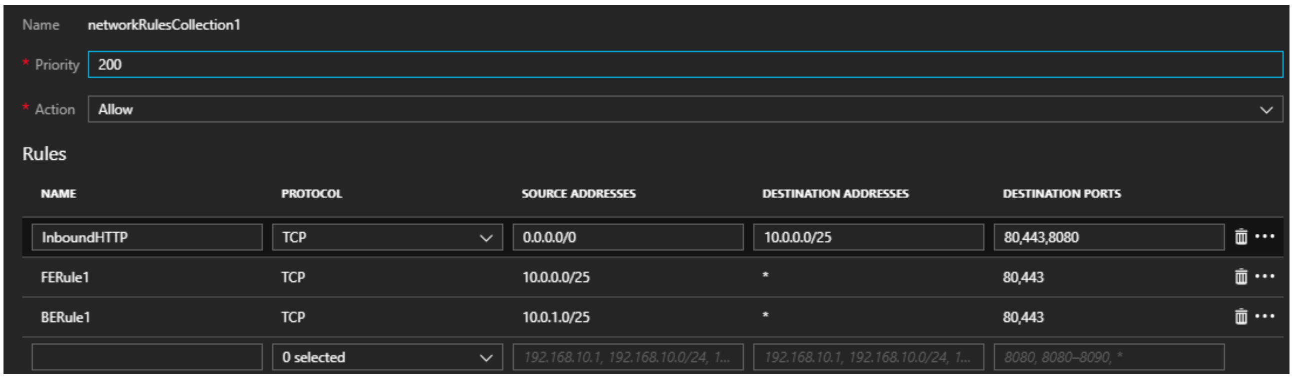
- The network rule collection
- The Application rule collection


Network rule collection | Application rule collection

PRIORITY	NAME	ACTION	RULES
200	networkRulesCollection1	Allow	3 rule(s).
			BERule1
			FERule1
			InboundHTTP

In the Network rule collection we find what was defined in ARM:

```
              "networkRuleCollections": [
                {
                  "name": "networkRulesCollection1",
                  "properties": {
                    "priority": 200,
                    "action":  { "type":  "Allow"},
                    "rules": [
                      {
                        "name": "FERule1",
                        "protocols": ["TCP"],
                        "sourceAddresses": ["[parameters('fESubnetRange')]"],
                        "destinationAddresses": ["*"],
                        "destinationPorts": ["80","443","53"]
                      },
                      {
                        "name": "BERule1",
                        "protocols": ["TCP"],
                        "sourceAddresses": ["[parameters('bESubnetRange')]"],
                        "destinationAddresses": ["*"],
                        "destinationPorts": ["80","443","53"]
                      },
                      {
                        "name": "InboundHTTP",
                        "protocols": ["TCP"],
                        "sourceAddresses": ["0.0.0.0/0"],
                        "destinationAddresses": ["[parameters('fESubnetRange')]"],
                        "destinationPorts": ["80","443","8080"]
                      }
                    ]
```



Name networkRulesCollection1

* Priority 200

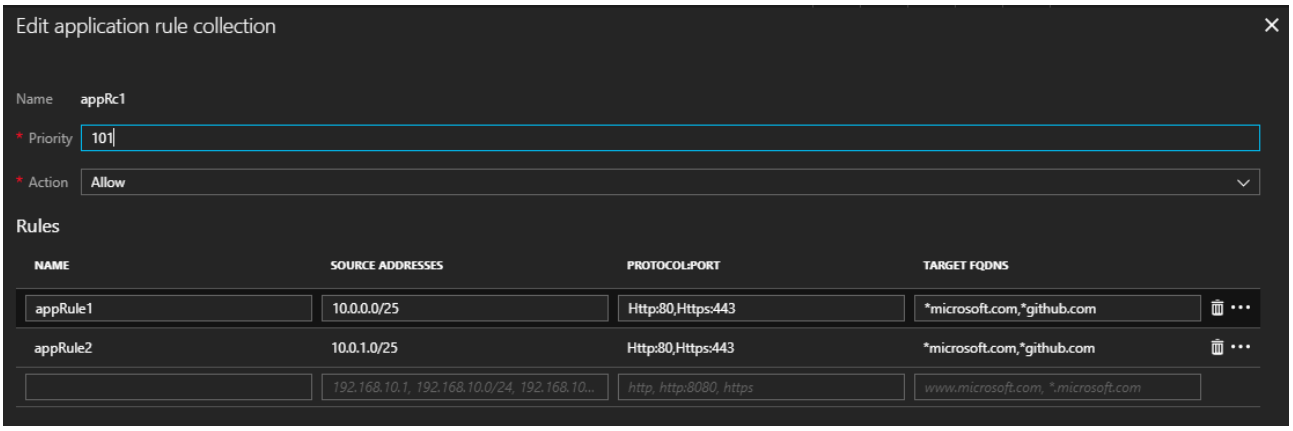
* Action Allow

Rules

NAME	PROTOCOL	SOURCE ADDRESSES	DESTINATION ADDRESSES	DESTINATION PORTS
InboundHTTP	TCP	0.0.0.0/0	10.0.0.0/25	80,443,8080
FERule1	TCP	10.0.0.0/25	*	80,443
BERule1	TCP	10.0.1.0/25	*	80,443
	0 selected	192.168.10.1, 192.168.10.0/24, 1...	192.168.10.1, 192.168.10.0/24, 1...	8080, 8080–8090, *

And something similar in the application rule collection:

```
                "applicationRuleCollections": [
                  {
                    "name": "appRc1",
                    "properties": {
                      "priority": 101,
                      "action": {"type": "Allow"},
                      "rules": [
                        {
                          "name": "appRule1",
                          "protocols": [{"port": "80", "protocolType": "http"},
{"port": "443", "protocolType": "https"}],
                          "targetUrls": ["*microsoft.com","*github.com"]
                        }
                      ]
                    }
                  }
                ],
```



Edit application rule collection

Name appRc1

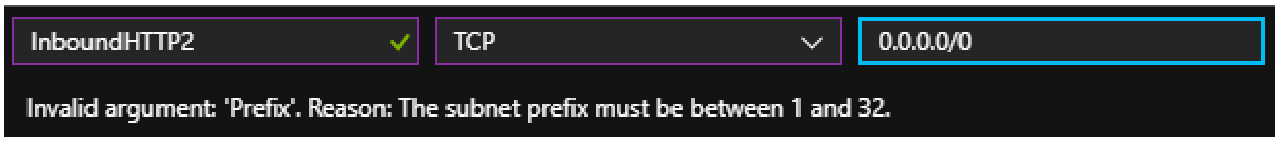
* Priority 101

* Action Allow

Rules

NAME	SOURCE ADDRESSES	PROTOCOL:PORT	TARGET FQDNS
appRule1	10.0.0.0/25	Http:80,Https:443	*microsoft.com,*github.com
appRule2	10.0.1.0/25	Http:80,Https:443	*microsoft.com,*github.com
	192.168.10.1, 192.168.10.0/24, 192.168.10...	http, http:8080, https	www.microsoft.com, *.microsoft.com

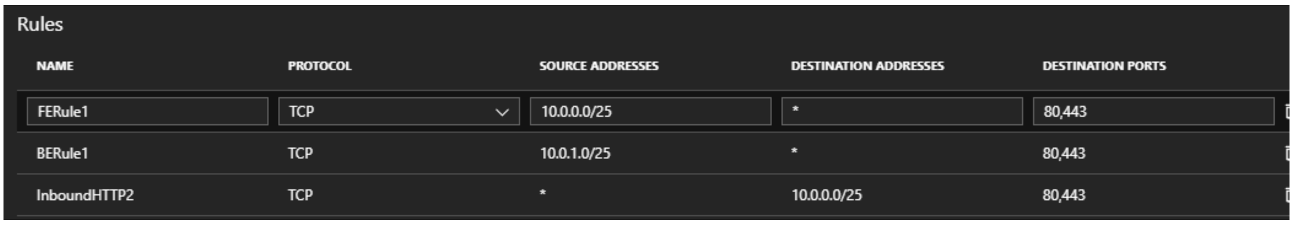
Now before going further in the testing, if we try to create a rule in the network rule collection, we will see that Azure does not accept the source as 0.0.0.0/0, indicating that the mask should have a value between 1 and 32.:



InboundHTTP2 | TCP | 0.0.0.0/0

Invalid argument: 'Prefix'. Reason: The subnet prefix must be between 1 and 32.

We should correct that in the template JSON. For now, let's modify it in the portal directly.



Rules

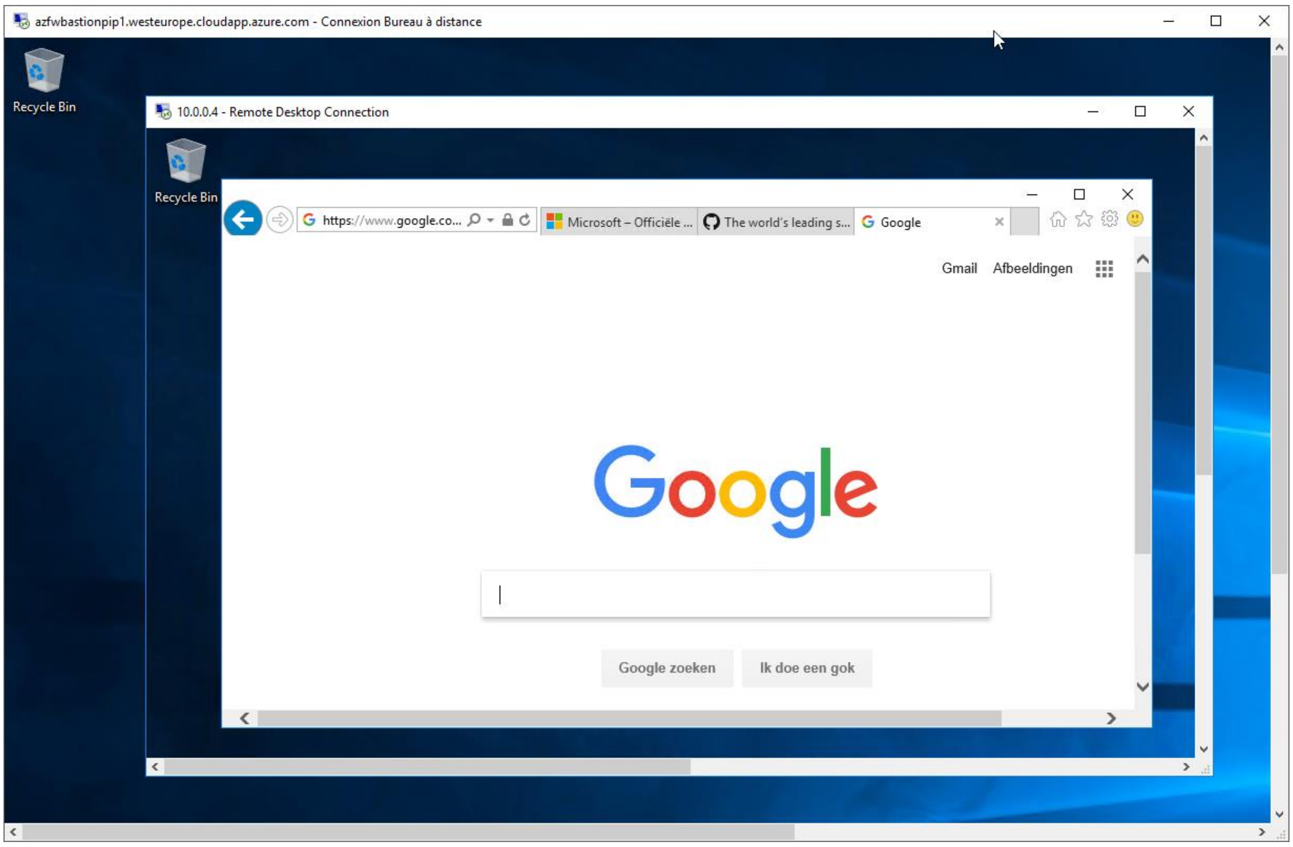
NAME	PROTOCOL	SOURCE ADDRESSES	DESTINATION ADDRESSES	DESTINATION PORTS
FERule1	TCP	10.0.0.0/25	*	80,443
BERule1	TCP	10.0.1.0/25	*	80,443
InboundHTTP2	TCP	*	10.0.0.0/25	80,443

Ok, let's perform some test now.

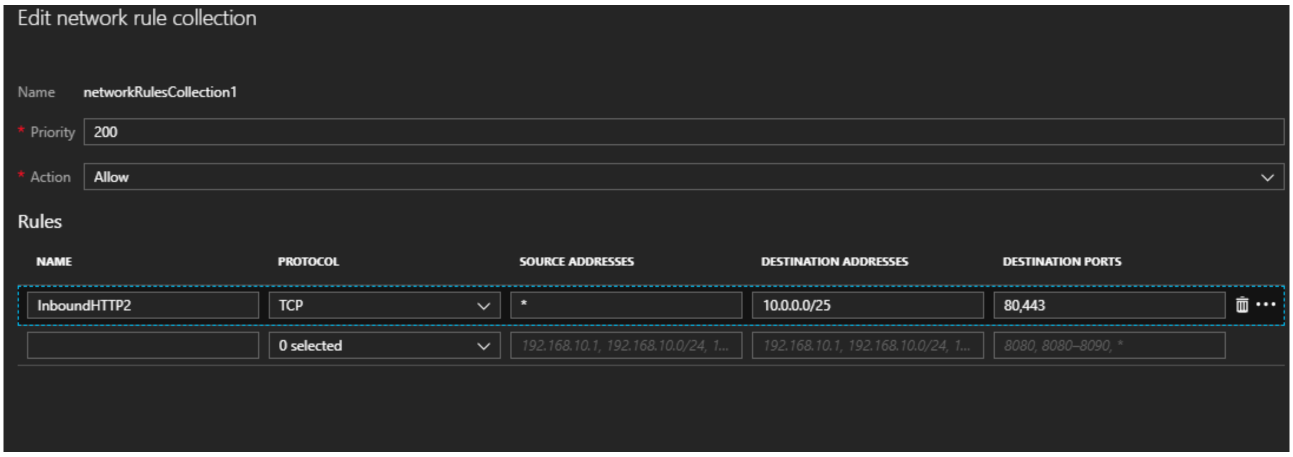
6.2 Testing the app rules from a test server

6.2.1 Testing outgoing traffic

Let's connect to each FE and BE servers, through the Bastion and test some of our rules. We expect to be able to go to Microsoft.com and github.com. However, we should not be able to access to google.com for example. On the other hand, we do not know which rule takes precedence between the app rule and the network rule. We do have a 200 for the network rule and a 101 for the app rules. But we also have a display showing first the network rule and second the app rule.
Let's try to reach google from the front end which is behind the Firewall due to the UDR:


azfwbastionpip1.westeurope.cloudapp.azure.com - Connexion Bureau à distance
Recycle Bin
10.0.0.4 - Remote Desktop Connection
Recycle Bin
https://www.google.co...
Microsoft – Officiële ...
The world's leading s...
Google
Gmail
Afbeeldingen
Google
Google zoeken
Ik doe een gok

Since it's working, let's try to change some of our rules. Let's keep only the app rules for outgoing traffic:


Edit network rule collection

Name networkRulesCollection1

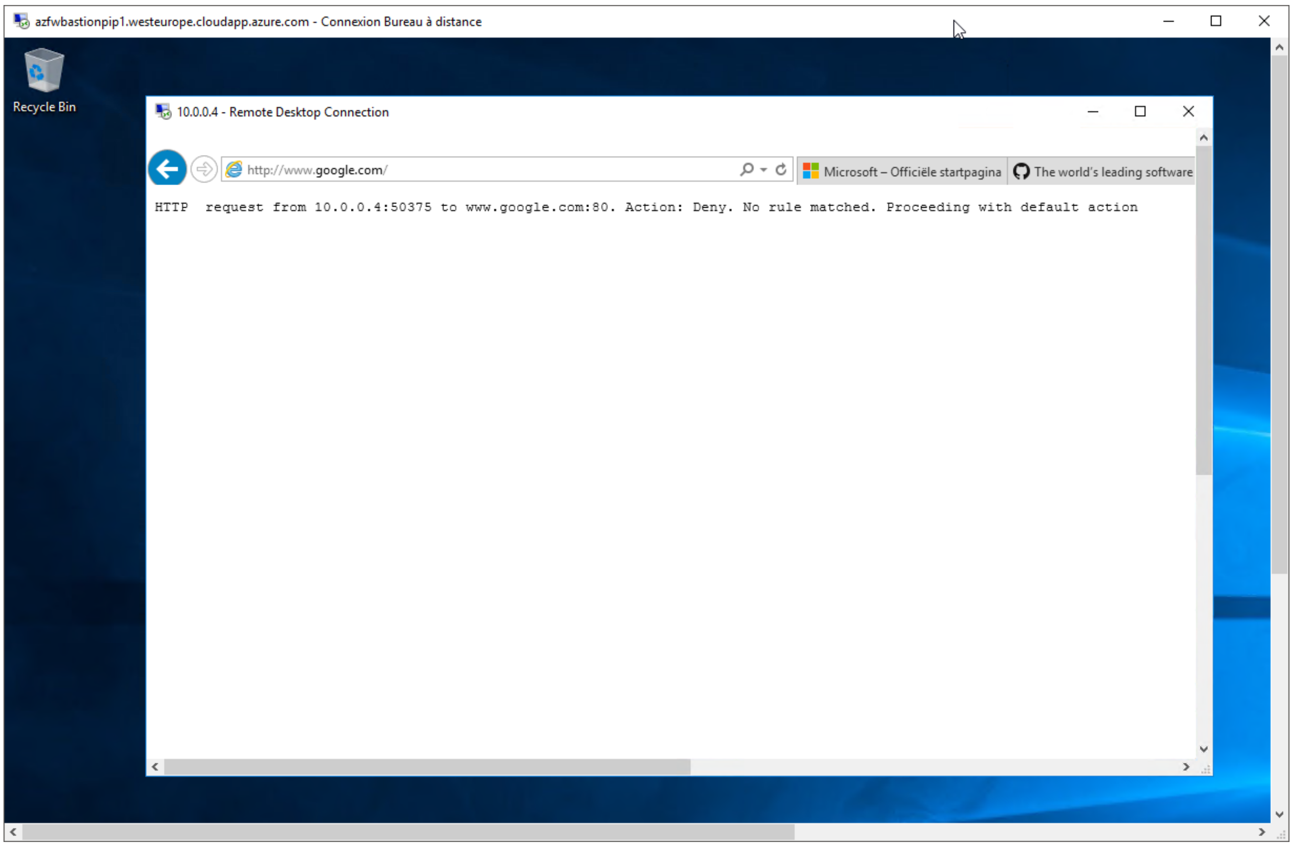
* Priority 200

* Action Allow

Rules

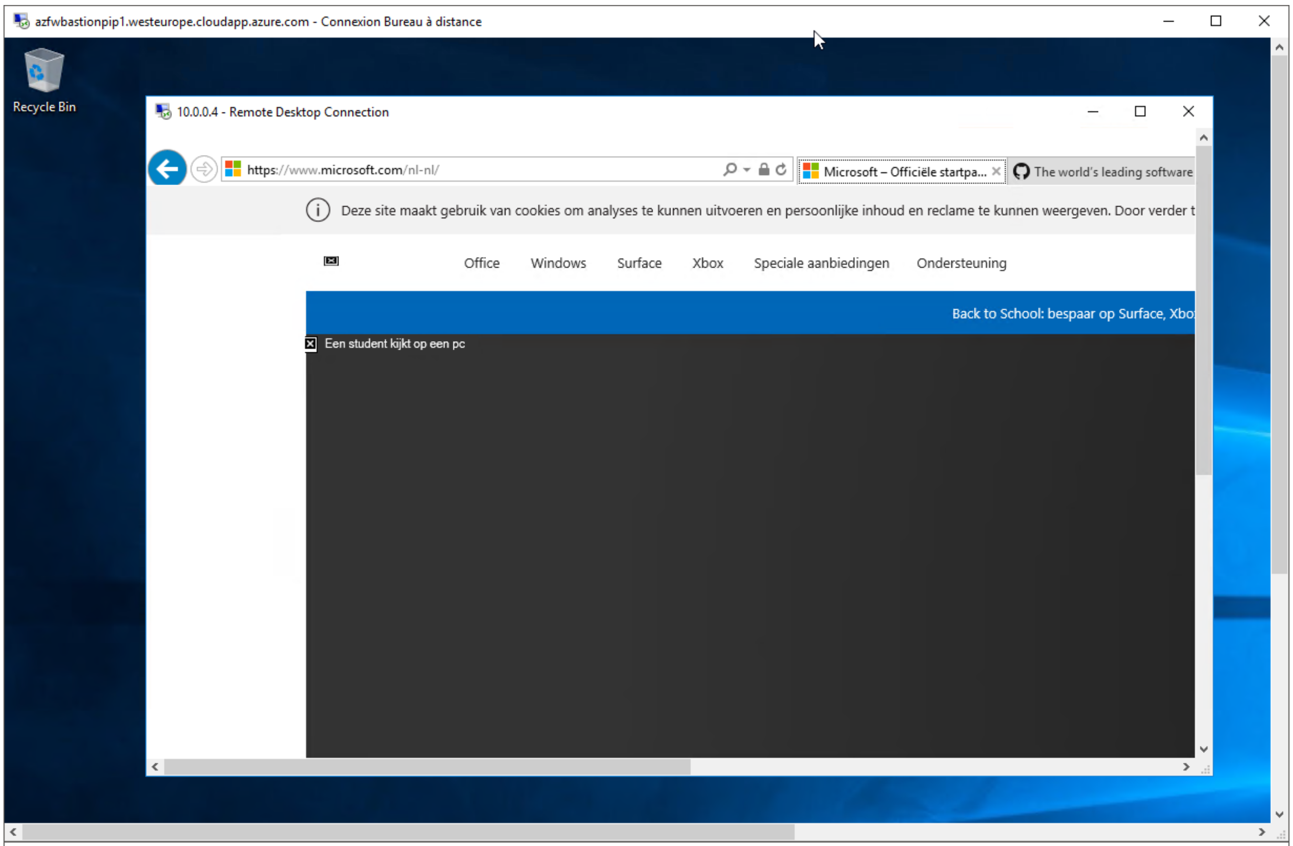
NAME	PROTOCOL	SOURCE ADDRESSES	DESTINATION ADDRESSES	DESTINATION PORTS
InboundHTTP2	TCP	*	10.0.0.0/25	80,443
	0 selected	192.168.10.1, 192.168.10.0/24, 1...	192.168.10.1, 192.168.10.0/24, 1...	8080, 8080-8090, *

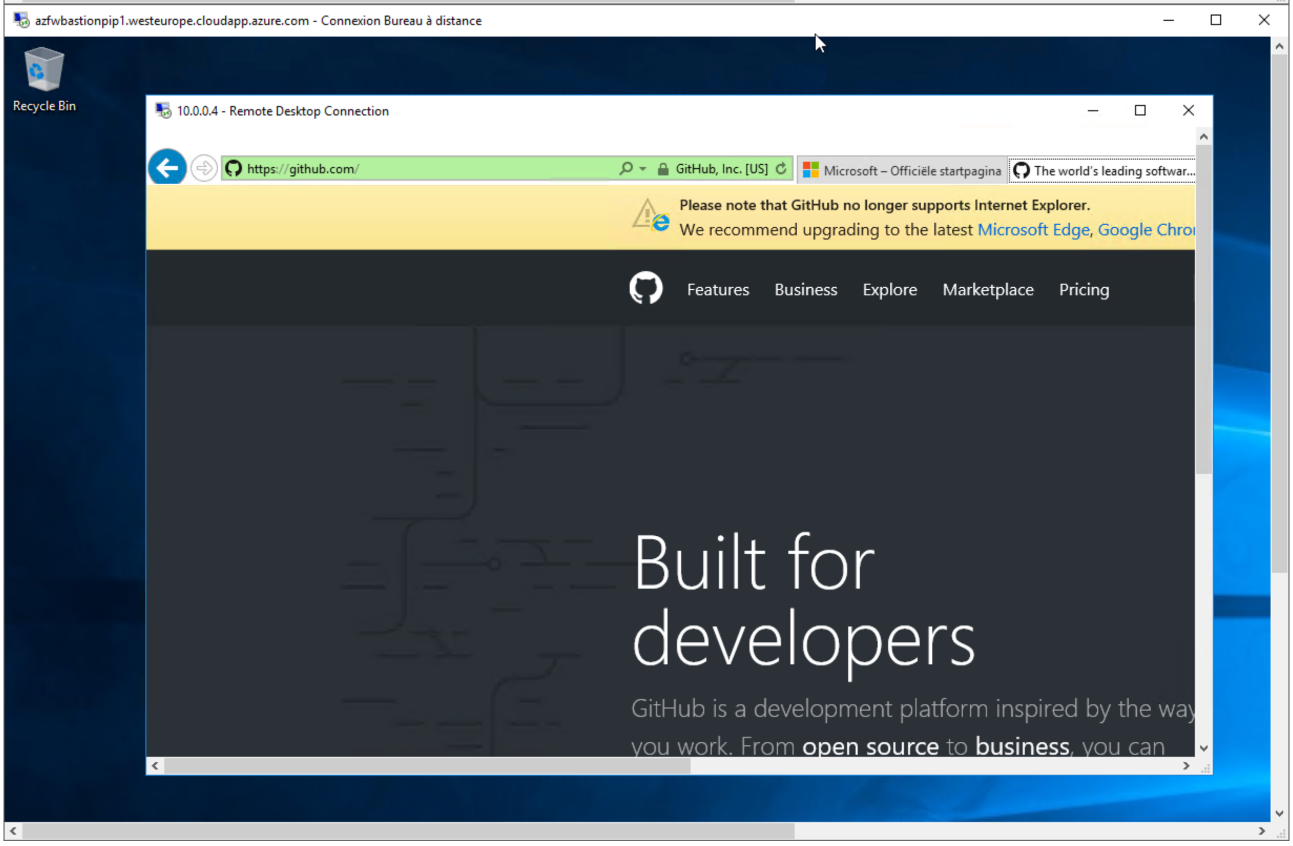
And we get the following:


azfwbastionpip1.westeurope.cloudapp.azure.com - Connexion Bureau à distance
Recycle Bin
10.0.0.4 - Remote Desktop Connection
http://www.google.com/
Microsoft – Officiële startpagina
The world's leading software

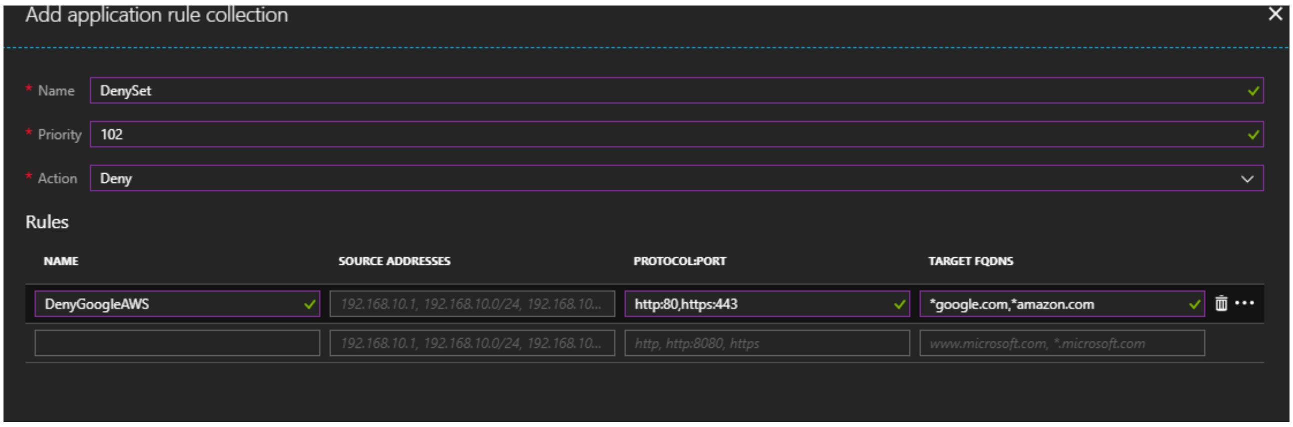
```
HTTP  request from 10.0.0.4:50375 to www.google.com:80. Action: Deny. No rule matched. Proceeding with default action
```

A short message is displayed, indicating that no rule matching the access to google.com is found. On the other hand, access to github.com and Microsoft.com is working:


azfwbastionpip1.westeurope.cloudapp.azure.com - Connexion Bureau à distance
Recycle Bin
10.0.0.4 - Remote Desktop Connection
https://www.microsoft.com/nl-nl/
Microsoft – Officiële startpa...
The world's leading software
Deze site maakt gebruik van cookies om analyses te kunnen uitvoeren en persoonlijke inhoud en reclame te kunnen weergeven. Door verder t
Office
Windows
Surface
Xbox
Speciale aanbiedingen
Ondersteuning
Back to School: bespaar op Surface, Xbo
Een student kijkt op een pc


azfwbastionpip1.westeurope.cloudapp.azure.com - Connexion Bureau à distance
Recycle Bin
10.0.0.4 - Remote Desktop Connection
https://github.com/
GitHub, Inc. [US]
Microsoft – Officiële startpagina
The world's leading softwar...
Please note that GitHub no longer supports Internet Explorer.
We recommend upgrading to the latest Microsoft Edge, Google Chro
Features
Business
Explore
Marketplace
Pricing
Built for developers
GitHub is a development platform inspired by the way you work. From open source to business, you can

Now let's test a little more and add an explicit rule denying access to google and aws. This is an Azure article after all ☺

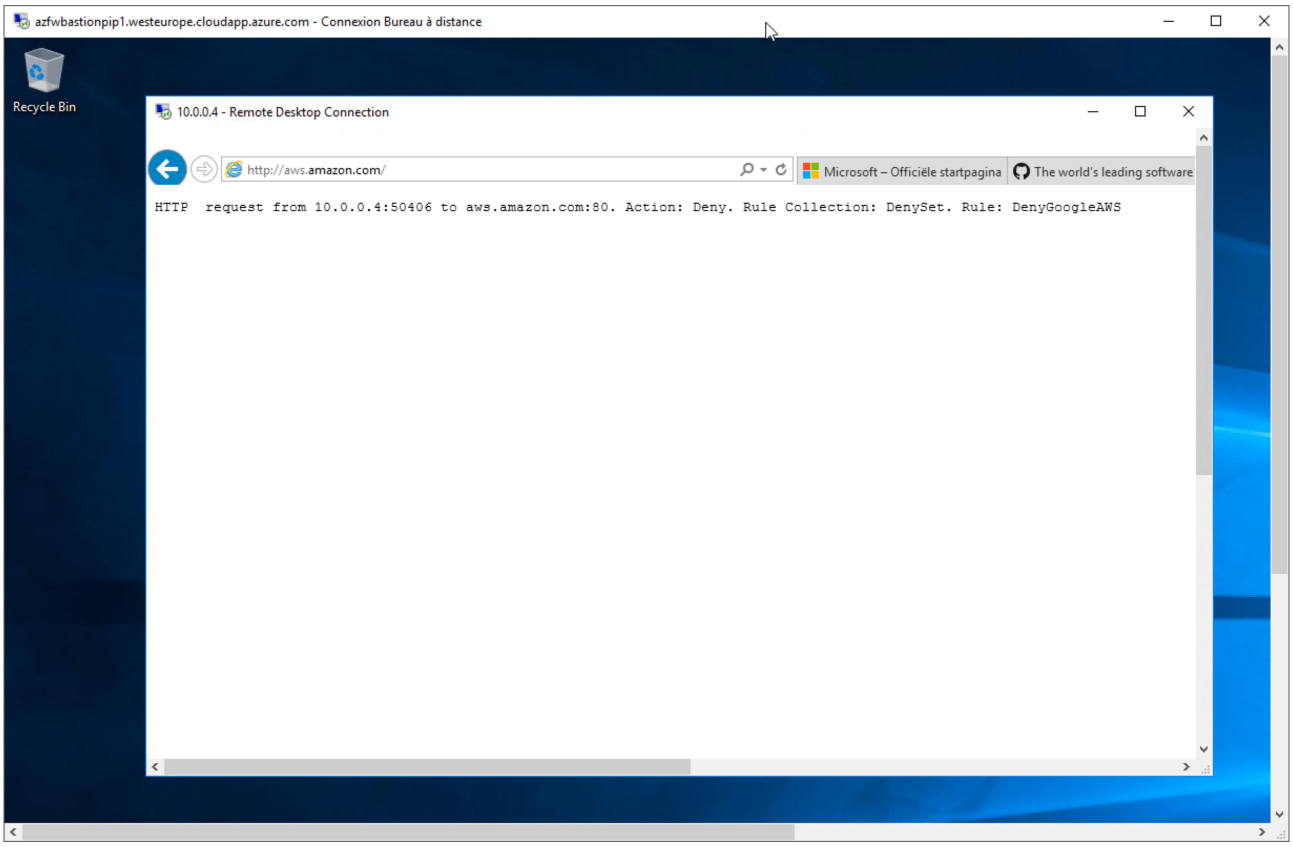

Add application rule collection

* Name	DenySet
* Priority	102
* Action	Deny

Rules

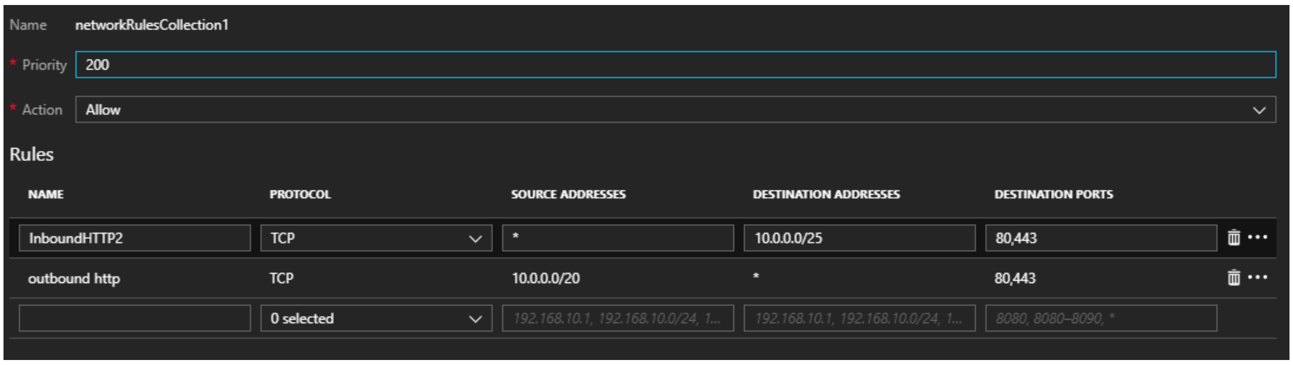
NAME	SOURCE ADDRESSES	PROTOCOL:PORT	TARGET FQDNS
DenyGoogleAWS	192.168.10.1, 192.168.10.0/24, 192.168.10...	http:80,https:443	*google.com,*amazon.com
	192.168.10.1, 192.168.10.0/24, 192.168.10...	http, http:8080, https	www.microsoft.com, *.microsoft.com

And let's try the connection to google and to amazon:


azfwbastionpip1.westeurope.cloudapp.azure.com - Connexion Bureau à distance
Recycle Bin
10.0.0.4 - Remote Desktop Connection
http://aws.amazon.com/
Microsoft – Officiële startpagina
The world's leading software

```
HTTP  request from 10.0.0.4:50406 to aws.amazon.com:80. Action: Deny. Rule Collection: DenySet. Rule: DenyGoogleAWS
```

And now we have a new message indicating that a specific rule deny access to aws.amazon.com. Nice. So we do know that no apps rule equals to traffic deny, similarly to explicit deny obviously. What about a deny app rule explicitly defined with a generic network rule allowing traffic to TCP 80? Let's try this out. After recreating the network rules, we get the following result:



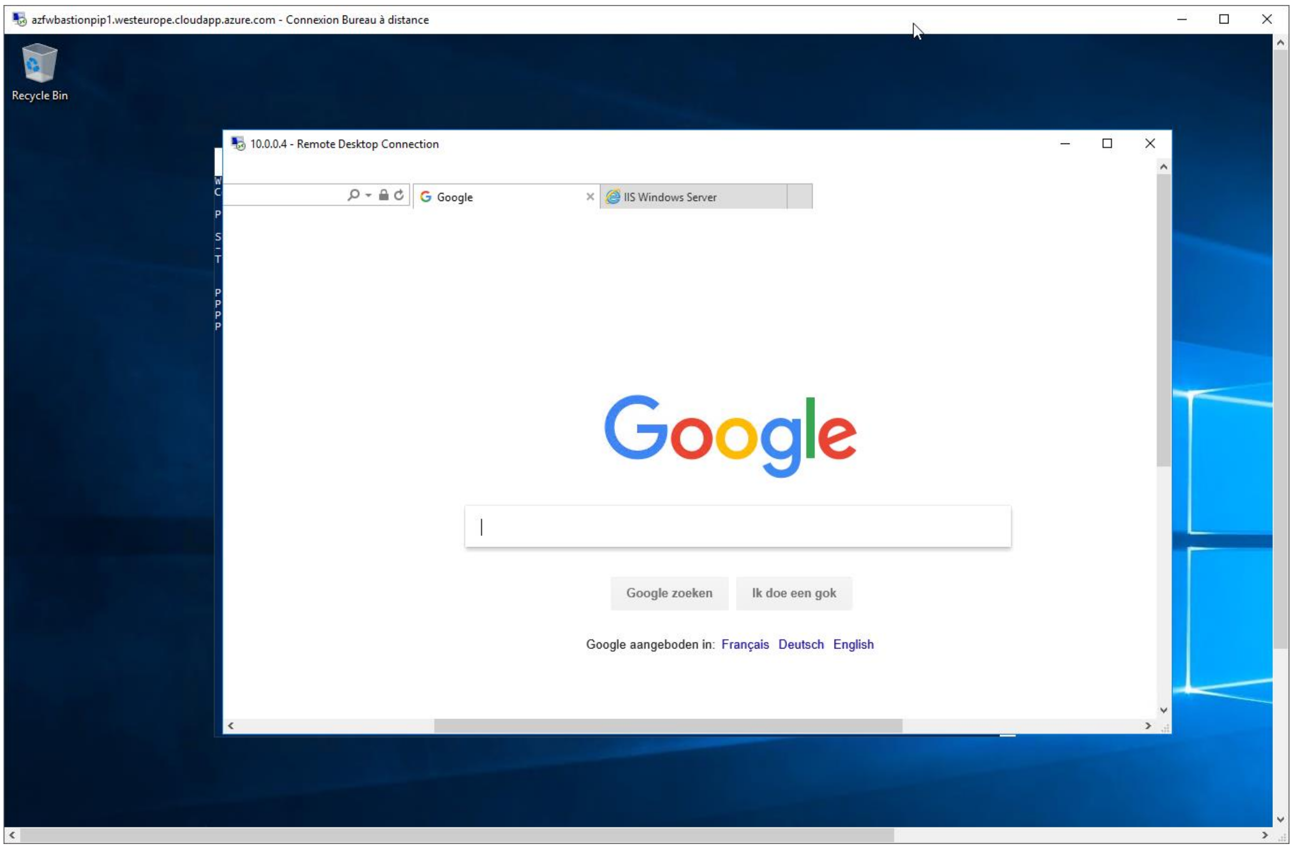
Name networkRulesCollection1

* Priority 200

* Action Allow

Rules

NAME	PROTOCOL	SOURCE ADDRESSES	DESTINATION ADDRESSES	DESTINATION PORTS
InboundHTTP2	TCP	*	10.0.0.0/25	80,443
outbound http	TCP	10.0.0.0/20	*	80,443
	0 selected	192.168.10.1, 192.168.10.0/24, 1...	192.168.10.1, 192.168.10.0/24, 1...	8080, 8080-8090, *



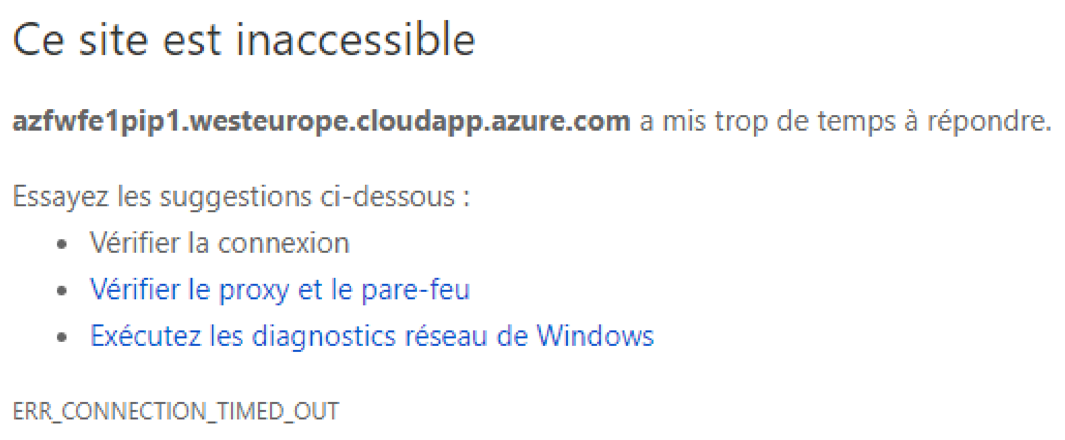
azfwbastionpip1.westeurope.cloudapp.azure.com - Connexion Bureau à distance

10.0.0.4 - Remote Desktop Connection

So deny rules does not win!

6.2.2 Testing inbound traffic

While the name does not say it, the Azure Firewall is not aimed to protect front end server from the Internet. For this kind of use, case, we can use Application gateway for exemple. One thing that hould convince people trying that out is the fact that there is no parameter available to configure NAT rules. Anf for those who believe only to what they can see, below is the print screen of a navigator when trying to reach a Front-End server routed through the firewall:



Ce site est inaccessible

azfwfe1pip1.westeurope.cloudapp.azure.com a mis trop de temps à répondre.

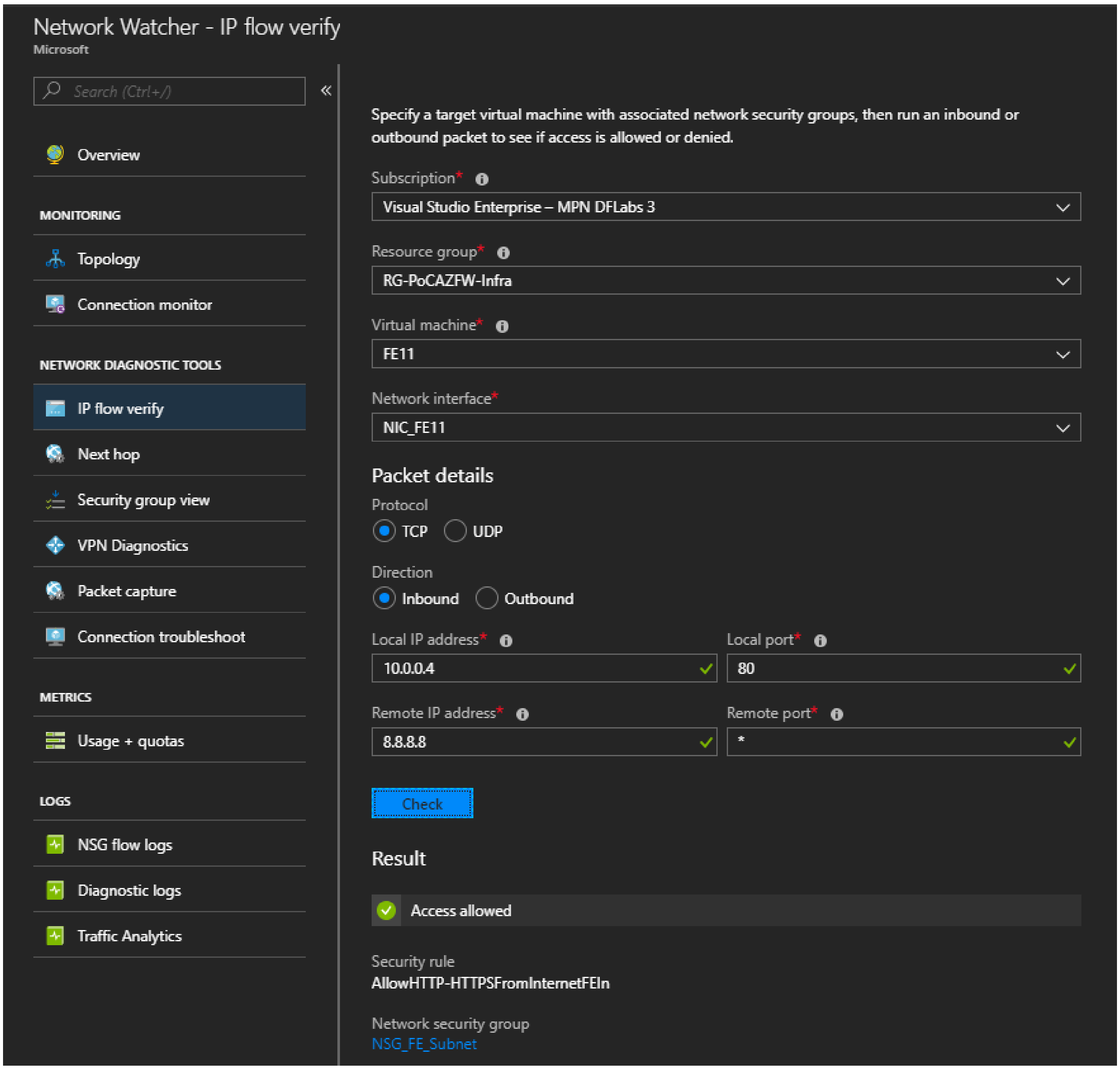
Essayez les suggestions ci-dessous :

- Vérifier la connexion
- Vérifier le proxy et le pare-feu
- Exécutez les diagnostics réseau de Windows

ERR_CONNECTION_TIMED_OUT

6.3 Testing the flow with Azure Network Watcher

Next, we will have a look at some Network watcher feature. First, let’s check the IP flow verify, to see how it works with the Azure Firewall. If we enter the details for the front-end VM behind the Firewall and try the flow from Internet to the VM, we get an allowed flow. That’s not surprising, since Network Watcher seems to see only the NSG for this feature:



Specify a target virtual machine with associated network security groups, then run an inbound or outbound packet to see if access is allowed or denied.

Subscription*: Visual Studio Enterprise – MPN DFLabs 3
Resource group*: RG-PoCAZFW-Infra
Virtual machine*: FE11
Network interface*: NIC_FE11

Packet details

Protocol: [✓ TCP] UDP
Direction: [✓ Inbound] Outbound
Local IP address*: 10.0.0.4
Local port*: 80
Remote IP address*: 8.8.8.8
Remote port*: *

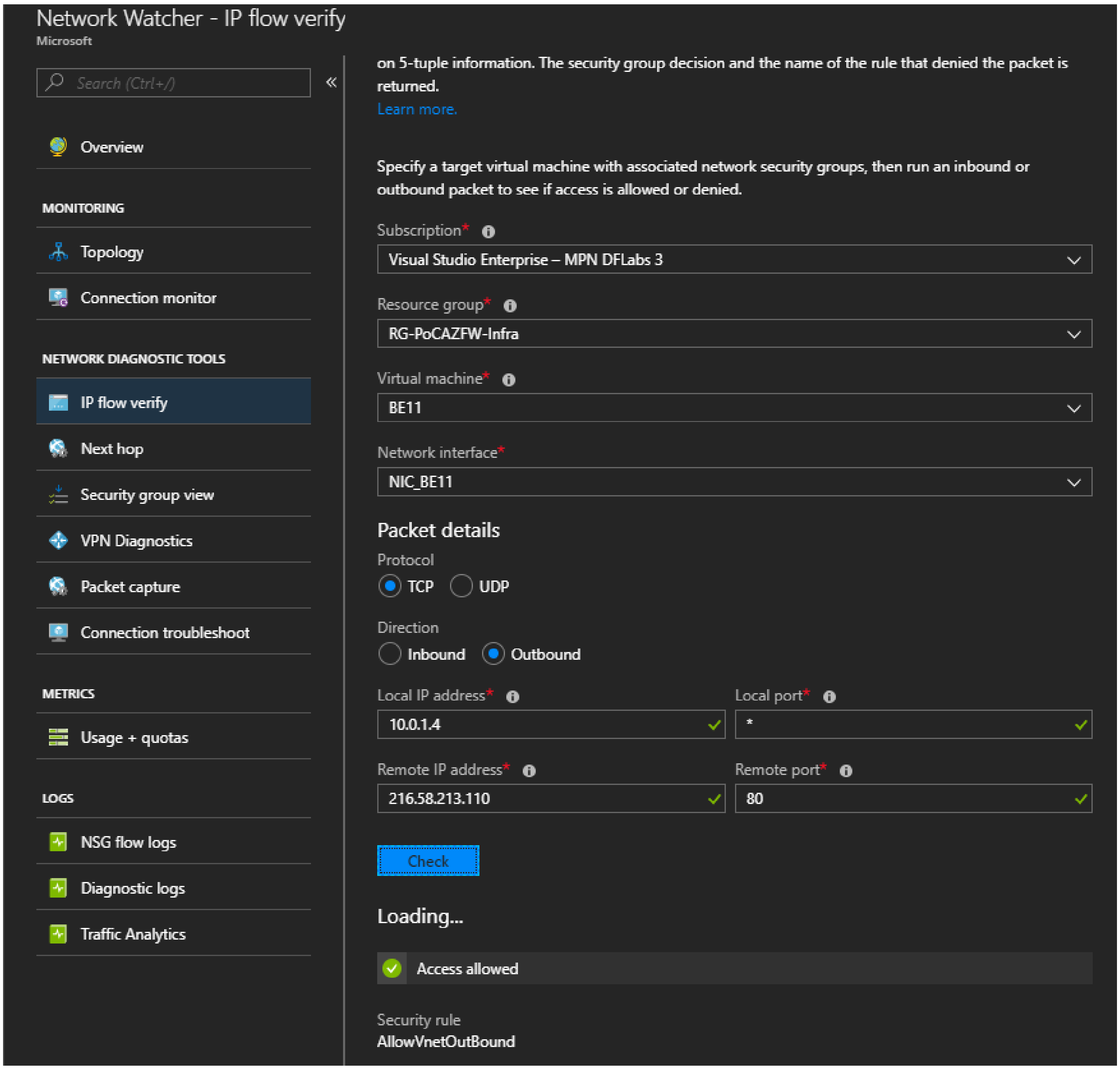
Result

Access allowed

Security rule
AllowHTTP-HTTPSFromInternetFEIn

Network security group
NSG_FE_Subnet

Now if we check the Outbound traffic to Google, after deleting the network rule collection and keeping only the deny rule:



on 5-tuple information. The security group decision and the name of the rule that denied the packet is returned.
Learn more.

Specify a target virtual machine with associated network security groups, then run an inbound or outbound packet to see if access is allowed or denied.

Subscription*: Visual Studio Enterprise – MPN DFLabs 3
Resource group*: RG-PoCAZFW-Infra
Virtual machine*: BE11
Network interface*: NIC_BE11

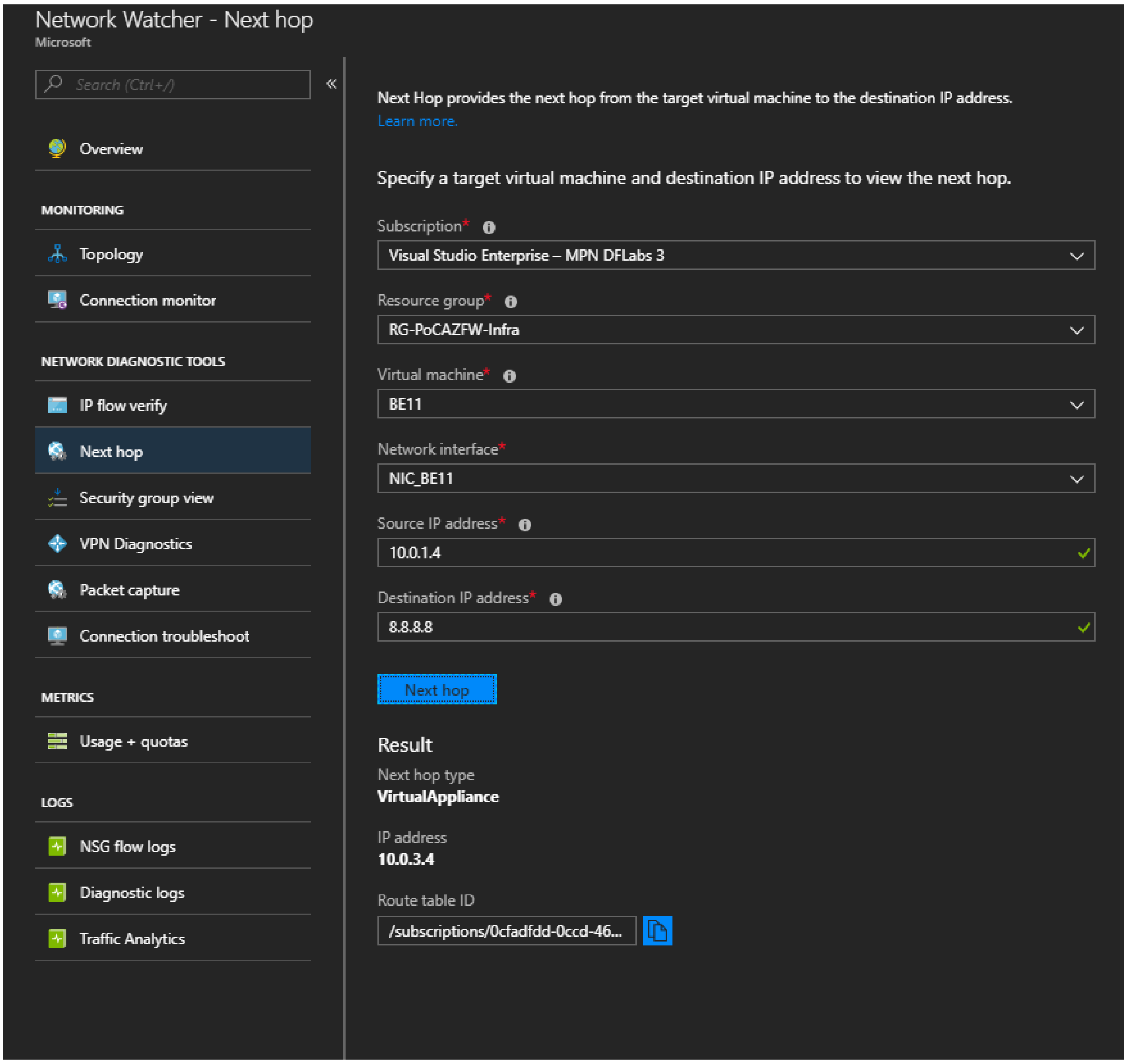
Packet details
Protocol: [✓ TCP] UDP
Direction: Inbound [✓ Outbound]
Local IP address*: 10.0.1.4
Local port*: *
Remote IP address*: 216.58.213.110
Remote port*: 80

Loading...

✓ Access allowed

Security rule
AllowVnetOutBound

Again, Network watcher does not know the Azure Firewall and thus does not check anything on it. And the thing is also that we defined app rules while Network watcher seems to check only on a TCP level. Now then, one last test, let’s see if Network watcher can see the route applied to the subnet:



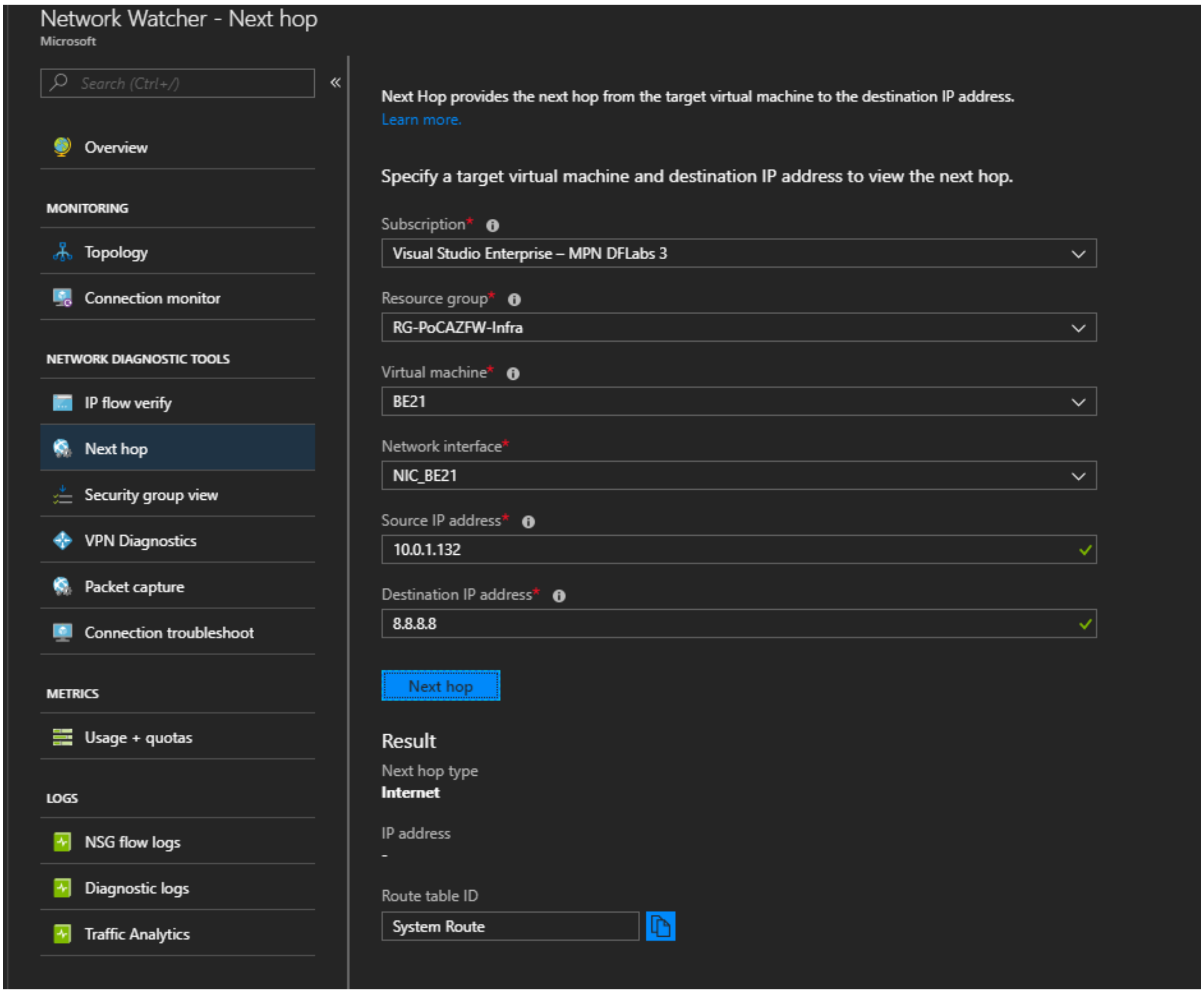
Next Hop provides the next hop from the target virtual machine to the destination IP address.
Learn more.

Specify a target virtual machine and destination IP address to view the next hop.

Subscription*: Visual Studio Enterprise – MPN DFLabs 3
Resource group*: RG-PoCAZFW-Infra
Virtual machine*: BE11
Network interface*: NIC_BE11
Source IP address*: 10.0.1.4
Destination IP address*: 8.8.8.8

Result
Next hop type
VirtualAppliance
IP address
10.0.3.4
Route table ID
/subscriptions/0cfadfdd-0ccd-46...

On this subnet routed through the Azure Firewall, we do see the next hop type as a virtual appliance, with its private IP address. The test on the other subnet shows a next hop to Internet, with the system route:


Network Watcher - Next hop
Microsoft

Next Hop provides the next hop from the target virtual machine to the destination IP address.
Learn more.

Specify a target virtual machine and destination IP address to view the next hop.

Subscription*: Visual Studio Enterprise – MPN DFLabs 3
Resource group*: RG-PoCAZFW-Infra
Virtual machine*: BE21
Network interface*: NIC_BE21
Source IP address*: 10.0.1.132
Destination IP address*: 8.8.8.8

Result
Next hop type
Internet
IP address
-
Route table ID
System Route

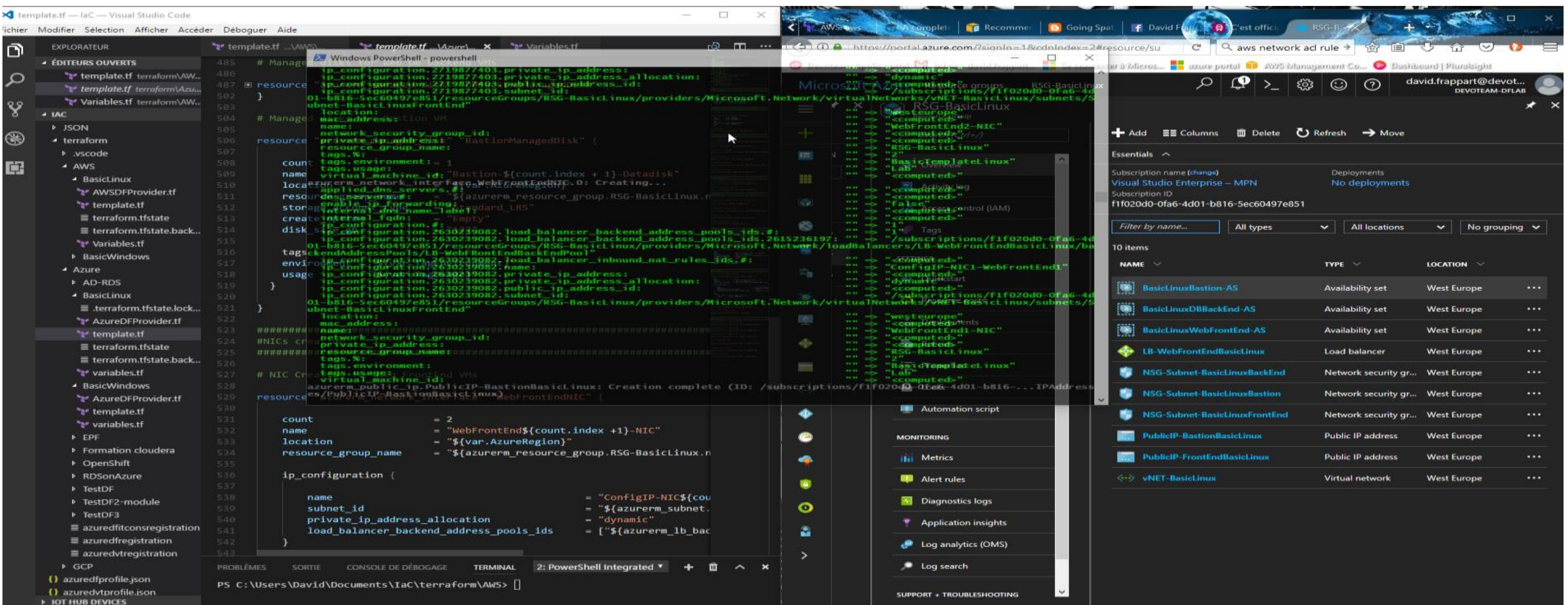
And that’s all for the Network Watcher testing.

7 Conclusion

In this article, we started playing with Azure Firewall. Note that the functionality is still in preview. IT means that it has to be activated on the subscription first. Also, some features may evolve before the GA. Next, it should be interesting to have a look on the logging capabilities of the appliance. We will have a look at it in a future article, hopefully. Until then…

Another Tech blog

My thoughts and experiences on Cloud related tek



```
    count                          = 2
    name                           = "WebFrontEnd${count.index +1}-NIC"
    location                       = "${var.AzureRegion}"
    resource_group_name            = "${azurerm_resource_group.RSG-BasicLinux.n

    ip_configuration {

        name                                    = "ConfigIP-NIC${cou
        subnet_id                               = "${azurerm_subnet.
        private_ip_address_allocation           = "dynamic"
        load_balancer_backend_address_pools_ids = ["${azurerm_lb_bac
    }
```

```
azurerm_public_ip.PublicIP-BastionBasicLinux: Creation complete (ID: /subscrip
es/PublicIP-BastionBasicLinux)
```

```
PS C:\Users\David\Documents\IaC\terraform\AWS>
```

Subscription name (change)
Visual Studio Enterprise – MPN
Subscription ID
f1f020d0-0fa6-4d01-b816-5ec60497e851

Deployments
No deployments

10 items

NAME	TYPE	LOCATION
BasicLinuxBastion-AS	Availability set	West Europe
BasicLinuxDBBackEnd-AS	Availability set	West Europe
BasicLinuxWebFrontEnd-AS	Availability set	West Europe
LB-WebFrontEndBasicLinux	Load balancer	West Europe
NSG-Subnet-BasicLinuxBackEnd	Network security gr...	West Europe
NSG-Subnet-BasicLinuxBastion	Network security gr...	West Europe
NSG-Subnet-BasicLinuxFrontEnd	Network security gr...	West Europe
PublicIP-BastionBasicLinux	Public IP address	West Europe
PublicIP-FrontEndBasicLinux	Public IP address	West Europe
vNET-BasicLinux	Virtual network	West Europe