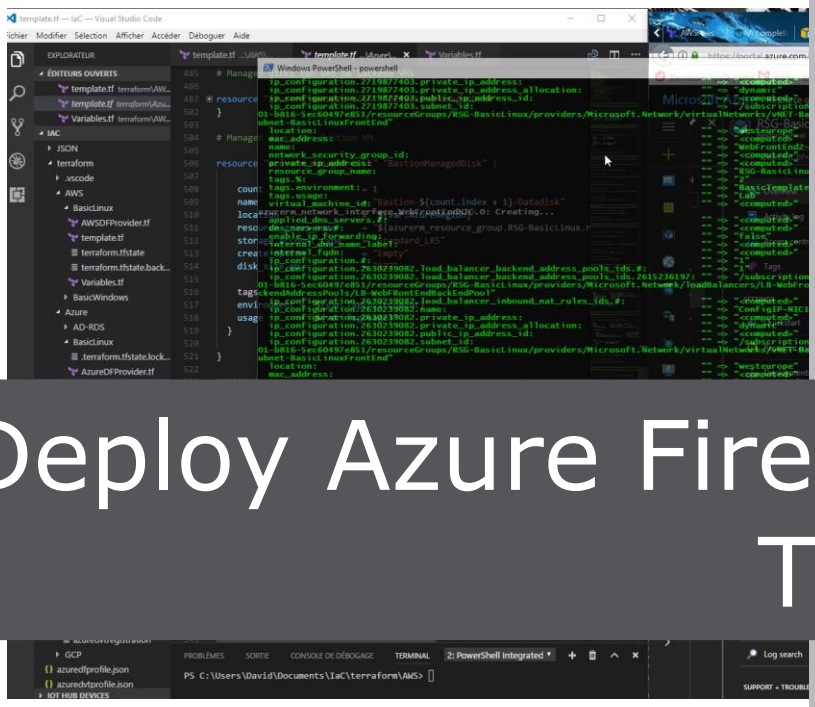


Teknews.cloud

Deploy Azure Firewall with Terraform

David Frappart

09/08/2018

Contributor

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

Version Control

Version	Date	State
1.0	2018/10/08	Final

Table of Contents

1 Introduction

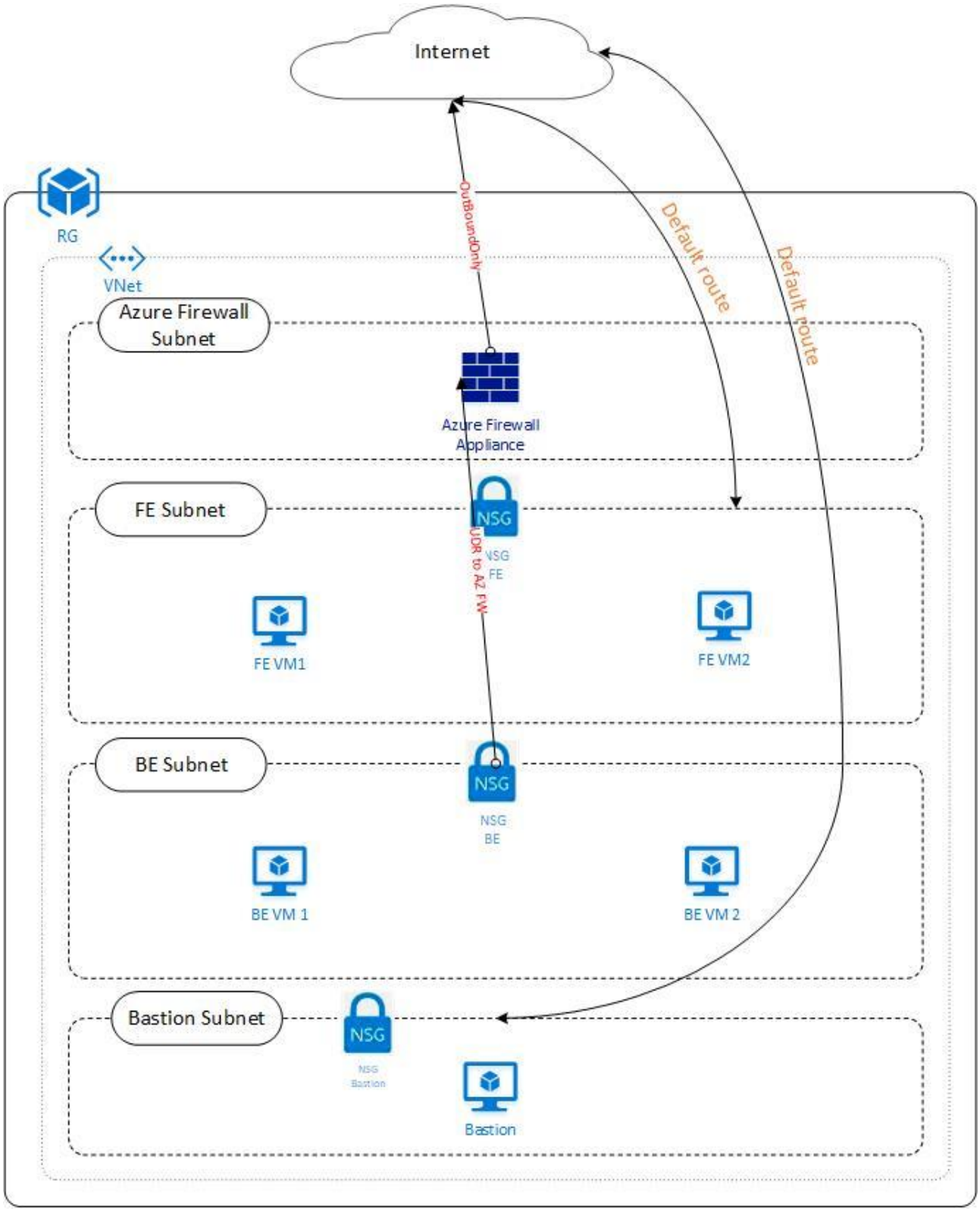
In a previous article, we played with the Azure firewall which offers as a managed service in Azure capabilities around the proxy type service. While still in preview, Hashicorp made the things well and already provides 2 new resources in the AzureRM provider to deploy this nice service.
In this article we will have a look at the terraform resource for Azure firewall and look into the code to use the corresponding resources.

2 Azure Firewall Terraform related resources

2.1 A reminder of the Azure Firewall capabilities

Just as a reminder, the Azure Firewall is a managed service which act as an appliance on which Outbound traffic is routed to. It gives the capabilities to filter traffic on the layer 4 but also on the layer 7, meaning it gives url filtering capabilities.
To do so, Azure Firewall relies on two kind of rules. The Network collection rules allows to group layer 4 rules while the Application collection rules allows to group layer 7 filtering rules.
To do so it requires a user defined route which will override the default route to force traffic to go through the Azure Firewall.
At its first release, it did not provide inbound filtering capabilities which could have make people wonder about the choice for the service name. Now however, DNAT capabilities are available but that's the subject for another day.

The schema below represents a deployment with Back-End subnet routed through the AZ Firewall while the other subnets rely on the default route:


Internet
RG
VNet
OutBoundOnly
Default route
Default route
Azure Firewall Subnet
Azure Firewall Appliance
FE Subnet
NSG
NSG FE
UDR to AZ FW
FE VM1
FE VM2
BE Subnet
NSG
NSG BE
BE VM 1
BE VM 2
Bastion Subnet
NSG
NSG Bastion
Bastion

2.2 Terraform resources for Azure Firewall

Now that things are clear again, let's have a look at the new terraform resources.
The azurerm_firewall allows to create the firewall object. It requires a dedicated subnet, without any NSG applied to and a public IP which is used for outbound traffic. Also, since it acts as a proxy, a user defined route applied to subnet hosting the workload which should be proxied through the Azure Firewall.
The collections rules can be created with the azurerm_firewall_network_rule_collection. As the name implies, it allows only the creation of network collections rules. So the layer 7 filtering rules are still not available.

3 Coding modules for Azure Firewall and Rules Collection

3.1 AzureRM Firewall

The following syntaxes is required for the resource creation:

```
resource "azurerm_firewall" "TerraFirewall" {
  name                          = "${var.FWName}"
  location                      = "${var.FWLocation}"
  resource_group_name           = "${var.RGName}"

  ip_configuration {
    name                          = "${var.FWName}${var.FWIPConfigName}"
    subnet_id                     = "${var.FWSubnetId}"
    internal_public_ip_address_id = "${var.FWPIPId}"
  }

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

The important part for the configuration is in the ip_configuration block. In this block is defined the target subnet id and the public IP.

The module should provide output to be called in the rule collection resource. The associated resource for the collection rule requires in input the firewall name. Available outputs are the firewall name, the firewall resource id and the firewall ip config:

```
output "Name" {
  value = "${azurerm_firewall.TerraFirewall.name}"
}

output "Id" {
  value = "${azurerm_firewall.TerraFirewall.id}"
}

output "IPConfig" {
  value = "${azurerm_firewall.TerraFirewall.ip_configuration}"
}
```

Once the config is deployed, with the proper output define as follow:

```
#########################################################
#AZ FW output

output "AZFW_VNet2IPConfig" {
  value = "${module.FW_VNet2.IPConfig}"
}

output "AZFW_VNet2Id" {
  value = "${module.FW_VNet2.Id}"
}

output "AZFW_VNet2Name" {
  value = "${module.FW_VNet2.Name}"
}
```

We can call the module as follow, with the appropriate modules outputs available:

```
#FW Creation

module "FW_VNet2" {

  #Module location
  source = "./Modules/18 AzureRMFW"

  #Module variables
  FWName              = "AZFWPocVNet2"
```

```
  RGName               = "${module.ResourceGroupInfra.Name}"
  FWLocation           = "${var.AzureRegion}"
  FWSubnetId           = "${module.FW_Subnet_VNet2.Id}"
  FWPIPId              = "${element(module.FW_Vnet2_PIP.Ids,0)}"
  EnvironmentTag       = "${var.EnvironmentTag}"
  EnvironmentUsageTag = "${var.EnvironmentUsageTag}"
}
```

We get the following output in terraform, after the deployment:

```
AZFW_VNet2IPConfig = [
    {
        internal_public_ip_address_id = /subscriptions/bb4e88a4-879f-4b1d-94f4-
1f48bfc1c5a8/resourceGroups/RG-PoCAZFW-
Infra/providers/Microsoft.Network/publicIPAddresses/azurefwvnet2pip1,
        name = AZFWPocVNet2Config,
        private_ip_address = 10.0.19.4,
        subnet_id = /subscriptions/bb4e88a4-879f-4b1d-94f4-1f48bfc1c5a8/resourceGroups/RG-
PoCAZFW-Infra/providers/Microsoft.Network/virtualNetworks/VNetPoCAZFW-
02/subnets/AzureFirewallSubnet
    }
]
AZFW_VNet2Id = /subscriptions/bb4e88a4-879f-4b1d-94f4-1f48bfc1c5a8/resourceGroups/RG-PoCAZFW-
Infra/providers/Microsoft.Network/azureFirewalls/AZFWPocVNet2
AZFW_VNet2Name = AZFWPocVNet2
```

3.2 AzureRM Collection rules

Creating the collection rules will require the firewall name as an input. The module call should look like that, with reference to other modules:

```
module "FW_VNet2_CollectionRule1" {

  #Module location
  source = "./Modules/19 AzureRM FW Collection Rules"

  #Module variables
  FWRuleCollecName              = "AZFWPocVNet2RulesCollec1"
  RGName                        = "${module.ResourceGroupInfra.Name}"
  FWName                        = "${module.FW_VNet2.Name}"
  FWRuleCollecAction            = "Allow"
  FWRuleName                    = "TestRule"
```

```
  FWRuleDesc                    = "Terraform created rule"
  FWRuleCollecSourceAddresses   = ["${module.BE_Subnet_VNet2.AddressPrefix}"]
  FWRuleCollecDestPorts         = ["80","443","53"]
  FWRuleCollecDestAddresses     = ["0.0.0.0/0"]


}
```

The resource creation is as follow:

```
resource "azurerm_firewall_network_rule_collection" "TerraFirewallRuleCollec" {
  name                          = "${var.FWRuleCollecName}"
  azure_firewall_name           = "${var.FWName}"
  resource_group_name           = "${var.RGName}"
  priority                      = "${var.FWRuleCollecPriority}"
  action                        = "${var.FWRuleCollecAction}"

  rule {
    name                    = "${var.FWRuleName}"
    description             = "${var.FWRuleDesc}"
    source_addresses        = ["${var.FWRuleCollecSourceAddresses}"]
    destination_ports       = ["${var.FWRuleCollecDestPorts}"]
    destination_addresses   = ["${var.FWRuleCollecDestAddresses}"]
    protocols               = ["${var.FWRuleProtos}"]
  }


}
```

The nice part is that we do have the possibility to create collections rules. But the resource in terraform lacks the granularity to define one by one the rule to be put in the collection rule. So in a module based configuration, it implies that we should either stick with a fixed number of rules in the collection rule module (or we could think about a crazy combination of conditional, count, and module variable default value, yeah that might work…)

4 What we still can’t do with Terraform

Ok, now we have tested some stuff with the new terraform resources. It is a good start. However, we still lack an object for the apps collection rules and also for the new NAT rules (which means that yes, now we can have inbound traffic through the Azure Firewall and thus it becomes worthy of its name).
The thing that would be nice is a resource for a single rule to integrate to a collection. That would be help avoid the crazy conditional counted module option.

5 Conclusion

In this article we took a rapid overview of the Azure Firewall terraform resource. Now we do have the capability to deploy through Terraform part of it. The other collection rules resources will probably be coming soon. Also the service being in preview we should get other additional capabilities. Look forward to it

Another Tech blog

My thoughts and experiences on Cloud related tek