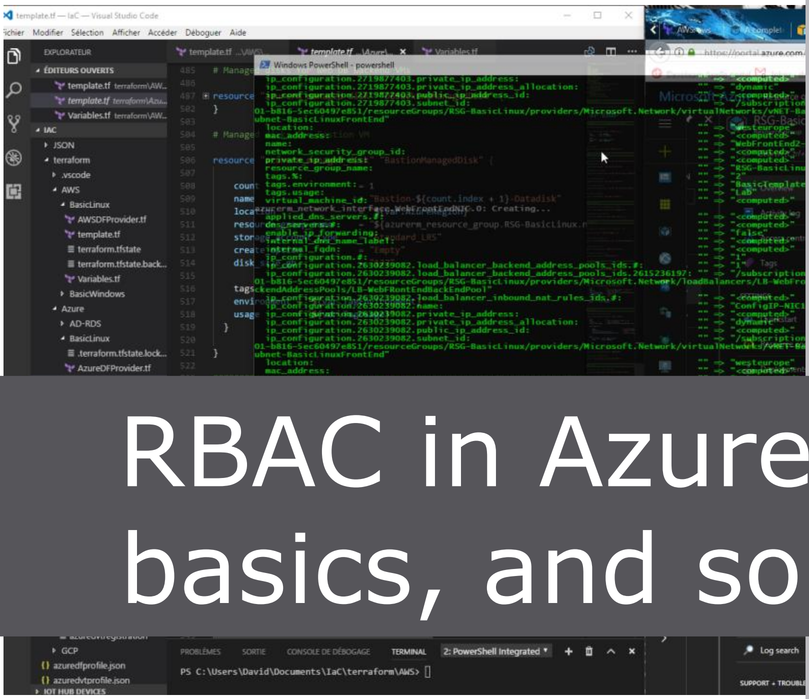


Teknews.cloud

RBAC in Azure, back to basics, and some more

David Frappart

03/01/2018

Contributor

Name	Title	Email
David Frappart	Cloud Architect	david@dfitcons.info

Version Control

Version	Date	State
1.0	2018/02/06	Final

Table of Contents

1 Introduction

In a previous article, we came accross a requirement to create a custom RBAC Role for an Azure Network Watcher instance which required many different permissions. While at this time I was planning to use Azure RM PowerShell, I came across Terraform resources for managing RBAC configuration in Azure. I did not go too much in details on Azure RBAC in this article since the topic was more on Network Watcher. However, i thought it would be nice to go in details on RBAC.
I learned by writing or manipulating. So this article is also an excuse for myself to re clarify the RBAC concepts in Azure. You may find some of the thing written here already written in Azure documentation. This is my way of remembering it.
You’ve been warned ☺

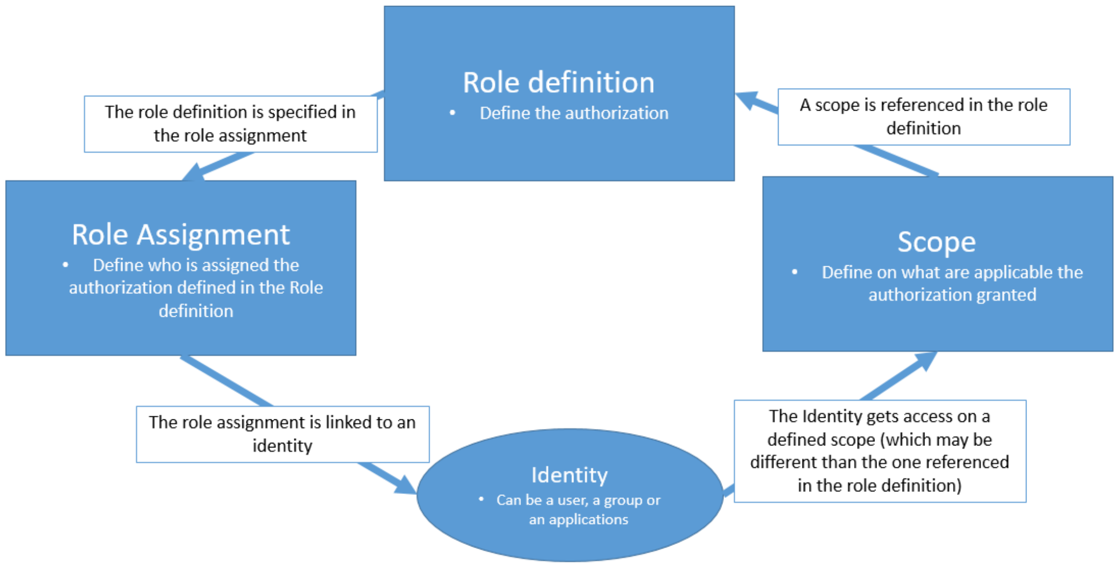
2 Short reminder on RBAC in Azure

2.1 RBAC Concepts

For a reminder, let’s take Wikipedia definition of the concept of Role Base Access Control:

Role-based-access-control (RBAC) is a policy neutral access control mechanism defined around roles and privileges. The components of RBAC such as role-permissions, user-role and role-role relationships make it simple to perform user assignments.

The schema below is my representation of the RBAC process:


Role definition
• Define the authorization
A scope is referenced in the role definition
The role definition is specified in the role assignment
Role Assignment
• Define who is assigned the authorization defined in the Role definition
Scope
• Define on what are applicable the authorization granted
The role assignment is linked to an identity
The Identity gets access on a defined scope (which may be different than the one referenced in the role definition)
Identity
• Can be a user, a group or an applications

2.2 RBAC in Azure

In Azure, a subscription is associated with an Azure Active Directory Tenant. This tenant stores the identity which may access to the subscription resource (with the exception of guest accounts).
For the differentiation of the access on Azure resources, Microsoft relies on RBAC and the 2 following concepts:

- Role definition
- Role Assignment

Simply, for access delegation, first a role definition is created, then the role is assigned to users who can get the required access on Azure resource with the role definition assigned to them.
Let's get into the details wth some PowerShell (and JSON files).

2.3 RBAC in Azure with PowerShell and JSON definition

2.3.1 Manipulating Role Definition

Microsoft comes with builtin roles. Those roles are accessible through PowerShell with the following command and its associated output:

```
PS C:\Users\User1> Get-AzureRmRoleDefinition | ? {$_.iscustom -eq $false} |ft
name,id

Name                                                  Id
----                                                  --
API Management Service Contributor                    312a565d-c81f-4fd8-895a-
4e21e48d571c
API Management Service Operator Role                  e022efe7-f5ba-4159-bbe4-
b44f577e9b61
API Management Service Reader Role                    71522526-b88f-4d52-b57f-
d31fc3546d0d
Application Insights Component Contributor            ae349356-3a1b-4a5e-921d-
050484c6347e
Application Insights Snapshot Debugger                08954f03-6346-4c2e-81c0-
ec3a5cfae23b
Automation Job Operator                               4fe576fe-1146-4730-92eb-
48519fa6bf9f
Automation Operator                                   d3881f73-407a-4167-8283-
e981cbba0404
```

```
Automation Runbook Operator                       5fb5aef8-1081-4b8e-bb16-
9d5d0385bab5
Azure Stack Registration Owner                    6f12a6df-dd06-4f3e-bcb1-
ce8be600526a
Backup Contributor                                5e467623-bb1f-42f4-a55d-
6e525e11384b
Backup Operator                                   00c29273-979b-4161-815c-
10b084fb9324
Backup Reader                                     a795c7a0-d4a2-40c1-ae25-
d81f01202912
Billing Reader                                    fa23ad8b-c56e-40d8-ac0c-
ce449e1d2c64
BizTalk Contributor                               5e3c6656-6cfa-4708-81fe-
0de47ac73342
CDN Endpoint Contributor                          426e0c7f-0c7e-4658-b36f-
ff54d6c29b45
CDN Endpoint Reader                               871e35f6-b5c1-49cc-a043-
bde969a0f2cd
CDN Profile Contributor                           ec156ff8-a8d1-4d15-830c-
5b80698ca432
CDN Profile Reader                                8f96442b-4075-438f-813d-
ad51ab4019af
Classic Network Contributor                       b34d265f-36f7-4a0d-a4d4-
e158ca92e90f
Classic Storage Account Contributor               86e8f5dc-a6e9-4c67-9d15-
de283e8eac25
Classic Storage Account Key Operator Service Role 985d6b00-f706-48f5-a6fe-
d0ca12fb668d
Classic Virtual Machine Contributor               d73bb868-a0df-4d4d-bd69-
98a00b01fccb
ClearDB MySQL DB Contributor                      9106cda0-8a86-4e81-b686-
29a22c54effe
Contributor                                       b24988ac-6180-42a0-ab88-
20f7382dd24c
Cosmos DB Account Reader Role                     fbdf93bf-df7d-467e-a4d2-
9458aa1360c8
Data Factory Contributor                          673868aa-7521-48a0-acc6-
0f60742d39f5
Data Lake Analytics Developer                     47b7735b-770e-4598-a7da-
8b91488b4c88
DevTest Labs User                                 76283e04-6283-4c54-8f91-
bcf1374a3c64
DNS Zone Contributor                              befefa01-2a29-4197-83a8-
272ff33ce314
```

```
DocumentDB Account Contributor                    5bd9cd88-fe45-4216-938b-
f97437e15450
Intelligent Systems Account Contributor           03a6d094-3444-4b3d-88af-
7477090a9e5e
Key Vault Contributor                             f25e0fa2-a7c8-4377-a976-
54943a77a395
Lab Creator                                       b97fb8bc-a8b2-4522-a38b-
dd33c7e65ead
Log Analytics Contributor                         92aaf0da-9dab-42b6-94a3-
d43ce8d16293
Log Analytics Reader                              73c42c96-874c-492b-b04d-
ab87d138a893
Logic App Contributor                             87a39d53-fc1b-424a-814c-
f7e04687dc9e
Logic App Operator                                515c2055-d9d4-4321-b1b9-
bd0c9a0f79fe
Managed Identity Contributor                      e40ec5ca-96e0-45a2-b4ff-
59039f2c2b59
Managed Identity Operator                         f1a07417-d97a-45cb-824c-
7a7467783830
Monitoring Contributor                            749f88d5-cbae-40b8-bcfc-
e573ddc772fa
Monitoring Reader                                 43d0d8ad-25c7-4714-9337-
8ba259a9fe05
Network Contributor                               4d97b98b-1d4f-4787-a291-
c67834d212e7
New Relic APM Account Contributor                 5d28c62d-5b37-4476-8438-
e587778df237
Owner                                             8e3af657-a8ff-443c-a75c-
2fe8c4bcb635
Reader                                            acdd72a7-3385-48ef-bd42-
f606fba81ae7
Redis Cache Contributor                           e0f68234-74aa-48ed-b826-
c38b57376e17
Resource Policy Contributor (Preview)             36243c78-bf99-498c-9df9-
86d9f8d28608
Scheduler Job Collections Contributor             188a0f2f-5c9e-469b-ae67-
2aa5ce574b94
Search Service Contributor                        7ca78c08-252a-4471-8644-
bb5ff32d4ba0
Security Admin                                    fb1c8493-542b-48eb-b624-
b4c8fea62acd
Security Manager                                  e3d13bf0-dd5a-482e-ba6b-
9b8433878d10
```

```
Security Reader                                   39bc4728-0917-49c7-9d2c-
d95423bc2eb4
Site Recovery Contributor                         6670b86e-a3f7-4917-ac9b-
5d6ab1be4567
Site Recovery Operator                            494ae006-db33-4328-bf46-
533a6560a3ca
Site Recovery Reader                              dbaa88c4-0c30-4179-9fb3-
46319faa6149
SQL DB Contributor                                9b7fa17d-e63e-47b0-bb0a-
15c516ac86ec
SQL Security Manager                              056cd41c-7e88-42e1-933e-
88ba6a50c9c3
SQL Server Contributor                            6d8ee4ec-f05a-4a1d-8b00-
a9b17e38b437
Storage Account Contributor                       17d1049b-9a84-46fb-8f53-
869881c3d3ab
Storage Account Key Operator Service Role         81a9662b-bebf-436f-a333-
f67b29880f12
Storage Blob Data Contributor (Preview)           ba92f5b4-2d11-453d-a403-
e96b0029c9fe
Storage Blob Data Reader (Preview)                2a2b9908-6ea1-4ae2-8e65-
a410df84e7d1
Storage Queue Data Contributor (Preview)          974c5e8b-45b9-4653-ba55-
5f855dd0fb88
Storage Queue Data Reader (Preview)               19e7f393-937e-4f77-808e-
94535e297925
Support Request Contributor                       cfd33db0-3dd1-45e3-aa9d-
cdbdf3b6f24e
Traffic Manager Contributor                       a4b10055-b0c7-44c2-b00f-
c7b5b3550cf7
User Access Administrator                         18d7d88d-d35e-4fb5-a5c3-
7773c20a72d9
Virtual Machine Contributor                       9980e02c-c2be-4d73-94e8-
173b1dc7cf3c
Web Plan Contributor                              2cc479cb-7b4d-49a8-b449-
8c00fd0f0a4b
Website Contributor                               de139f84-1756-47ae-9be6-
808fbbe84772
```

We do have quite a few built-in roles. However, the main ones are owner, which define a role havig all access, contributor which is the same as owner with a few rstrcition and reader which as the name implies allow to view all the resources in the assigned scope.
To get a view of the actual assign right, we can simply use the commands:

```
(Get-AzureRMRoleDefinition -Name <RoleDefinitionName>).Actions
(Get-AzureRMRoleDefinition -Name <RoleDefinitionName>).NotActions
```

For example with the contributor role:

```
PS C:\Users\Users1> (Get-AzureRmRoleDefinition -Name contributor).Actions
*
PS C:\Users\Users1> (Get-AzureRmRoleDefinition -Name contributor).NotActions
Microsoft.Authorization/*/Delete
Microsoft.Authorization/*/Write
Microsoft.Authorization/elevateAccess/Action
```

And the website contributor:

```
PS C:\Users\Users1> (Get-AzureRmRoleDefinition -Name "Website
Contributor").NotActions
PS C:\Users\Users1> (Get-AzureRmRoleDefinition -Name "Website Contributor").Actions
Microsoft.Authorization/*/read
Microsoft.Insights/alertRules/*
Microsoft.Insights/components/*
Microsoft.ResourceHealth/availabilityStatuses/read
Microsoft.Resources/deployments/*
Microsoft.Resources/subscriptions/resourceGroups/read
Microsoft.Support/*
Microsoft.Web/certificates/*
Microsoft.Web/listSitesAssignedToHostName/read
Microsoft.Web/serverFarms/join/action
Microsoft.Web/serverFarms/read
Microsoft.Web/sites/*
```

Microsoft provides a quite exhaustive documentation on roles and actions here. For PowerShell addict, the cmdlet Get-AzureRMProviderOperation can provides information on the operations associated with a provider, the provider being described as <Microsoft.ProviderName>. Below is an example of getting provider operations:

```
PS C:\Users\User1> Get-AzureRmProviderOperation
Microsoft.Compute/virtualmachines/start/* | ft operation,operationname -autosize

Operation                                    OperationName
---------                                    -------------
Microsoft.Compute/virtualMachines/start/action Start Virtual Machine
```

Now all those cmdlets allow us to get information roles definition and the associated operation.
To create a new role definition, we use the cmdlet New-AzureRMRoleDefinition.

Now this cmdlet ca be used with a full PowerShell approach to add the required provider operations, or it can be used in conjunction with a JSON file in which the providers operations are listed either in the Actions list or the NotActions List. The first one defining the available actions in the role definition, the second one the unavailable actions.
An important thing to consider is that the NotActions list is not a deny list. If an action listed in the NotActions list is also present in the Actions list in another role definition, the operation will be granted to the service principal assigned to those 2 role.
The JSON file defining the providers actions (and potentially notactions) is something like that:

```
{
    "Name": "NetworkWatcherCustomRole",
    "Id": null,
    "IsCustom": true,
    "Description": "Custom Role for access to Network Watcher",
    "Actions": [
      "Microsoft.Storage/*/read",
      "Microsoft.Authorization/*/read",
      "Microsoft.Resources/subscriptions/resourceGroups/*/read",
      "Microsoft.Storage/storageAccounts/listServiceSas/*/Action",
      "Microsoft.Storage/storageAccounts/listAccountSas/*/Action",
      "Microsoft.Storage/storageAccounts/listKeys/*/Action",
      "Microsoft.Compute/virtualMachines/*/read",
      "Microsoft.Compute/virtualMachines/*/write",
      "Microsoft.Compute/virtualMachineScaleSets/*/read",
      "Microsoft.Compute/virtualMachineScaleSets/*/write",
      "Microsoft.Network/networkWatchers/packetCaptures/*/read",
      "Microsoft.Network/networkWatchers/packetCaptures/*/write",
      "Microsoft.Network/networkWatchers/packetCaptures/*/delete",
      "Microsoft.Network/networkWatchers/*/write",
      "Microsoft.Network/networkWatchers/*/read",
      "Microsoft.Insights/alertRules/*",
      "Microsoft.Support/*",
      "Microsoft.Network/networkWatchers/*",
      "Microsoft.Network/networkWatchers/*/Action"
    ],
    "NotActions": [
    ],
    "AssignableScopes": [
      "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxxxxxxxxxxxxx"
    ]
  }
```

We can see also a line defining the scope. If, as in the example, the AssignableScope is a subscritpions id, it means that the role can be assigned to a subscription level. If we have a "/*", it means that it can be assigned to multiple subscription.

With this file, we can create a new role definition with the cmdlet New-AzureRMRoleDefinition and by specifying the InputFile parameter as follow:

```
PS C:\Users\User1> New-AzureRMRoleDefinition –InputFile <Path_to_JSON_file>
```

After that we have a custom RBAC role added to the built-in role and we can start the assignment part. The new custom role(s) can be displayed as follow:

```
PS C:\Users\User1> Get-AzureRmRoleDefinition | ? {$_.iscustom -eq $true} | ft
name,description,id

Name                                 Description
Id
----                                 -----------
--
NetworkWatcherCustomRole             Custom Role for access to Network Watcher
xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
Terra-Dummy_CustomRBACRoleDefinition Custom Role for Network Watcher created
through Terraform xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

2.3.2 Manipulating Role Assignment

For getting Role assignment, we simply use the cmdlet Get-AzureRMRoleAssignment, as displayed below:

```
PS C:\Users\User1> Get-AzureRmRoleAssignment |ft
roledefinitionname,displayname,objecttype -AutoSize

RoleDefinitionName       DisplayName              ObjectType
------------------       -----------              ----------
NetworkWatcherCustomRole NetworkWatcherOperators Group
Owner                    TerraformDF              ServicePrincipal
Owner                    DFLab-TenantAdmins       Group
Contributor              TerraformDF              ServicePrincipal
```

This command display the assigment made in the environment. For Assignment creation, we would use the New-AzureRMRoleAssignment, which requires a target for the assignment, a role definition for the access to grant and a scope. Now, we explored this subject in a previous article without going into the details, but for a role assignment we need an Azure Active Directory Object ID to
 associate it with the desired role.
We need the Azure Active Directory PowerShell Module and to connect on the target Azure Active Directory tenant which provides identities for the subscription on which we cnofigure roles.
A simple command is used to connect to the Azure AD tenant, with an account having enough permissions:

```
PS C:\Users\User1> Connect-AzureAD -Credential (get-credential)

applet de commande Get-Credential à la position 1 du pipeline de la commande
Fournissez des valeurs pour les paramètres suivants :
Credential

Account                        Environment  TenantId
User1@dflab.onmicrosoft.com    AzureCloud   xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

After the connection to the Azure AD tenant, we either get or create new Azure AD object. The creation requires as usual some secret for the password. However, in this case, we are asked for a passwordprofile as display the example below:

```
PS C:\Users\David> help New-AzureADUser -Examples

NOM
    New-AzureADUser

RÉSUMÉ
    Crée un utilisateur AD.


    Exemple 1 : créer un utilisateur

    -AzureADUser -DisplayName "New User" -PasswordProfile $PasswordProfile -
UserPrincipalName "NewUser@contoso.com" -AccountEnabled $true
    -MailNickName "Newuser"

    ObjectId                             DisplayName UserPrincipalName
UserType
    --------                             ----------- -----------------
--------
    5e8b0f4d-2cd4-4e17-9467-b0f6a5c0c4d0 New user    NewUser@contoso.com
Member

    Cette commande crée un utilisateur.
```

The password profile object is thankfully referenced in the Azure AD Module documentation. We can populate the required value in a PS Object:

```
PS C:\Users\David> $PasswordProfile = New-Object -TypeName
Microsoft.Open.AzureAD.Model.PasswordProfile
```

this object properties can be displayed:

```
PS C:\Users\David> $PasswordProfile | fl

Password                      :
ForceChangePasswordNextLogin :
EnforceChangePasswordPolicy   :
```

And we populate the properties as we need it:

```
PS C:\Users\David> $PasswordProfile.Password = "hybr5DDw!"
PS C:\Users\David> $PasswordProfile.ForceChangePasswordNextLogin = $false
```

Before creating the azure AD user

```
PS C:\Users\David> New-AzureADUser -AccountEnabled $true -DisplayName TestUser -
PasswordProfile $PasswordProfile -MailNickName TestUser -UserPrincipalName
testuser@devoteamdflab.onmicrosoft.com

ObjectId                             DisplayName UserPrincipalNameUserType
--------                             ----------- -----------------
2a65dc30-d562-40f1-b0ab-46889b3c3c37 TestUser    testuser@lab.onmicrosoft.com
```

Fun fact, or is it, the $passwordProfile.Password is not encrypted. I do not as of now know why.

Now that we have added a user we can at last it assign a role definition. First, we need the object id from the user that we just created, then we also need the role definition name and the subscription id fro the scope:

```
PS C:\Users\User1> Get-AzureRmRoleDefinition | ? {$_.iscustom -eq $true}


Name             : NetworkWatcherCustomRole
Id               : 15983214-2748-4202-b9eb-c372770c64f4
IsCustom         : True
Description      : Custom Role for access to Network Watcher
Actions          : {Microsoft.Storage/*/read, Microsoft.Authorization/*/read,
Microsoft.Resources/subscriptions/resourceGroups/*/read,
                   Microsoft.Storage/storageAccounts/listServiceSas/*/Action...}
NotActions       : {}
AssignableScopes : {/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}
```

```
PS C:\Users\User1> New-AzureRmRoleAssignment -ObjectId 2a65dc30-d562-40f1-b0ab-
46889b3c3c37 -RoleDefinitionName NetworkWatcherCustomRole -Scope /subscriptions/
xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx


RoleAssignmentId   : /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
/providers/Microsoft.Authorization/roleAssignments/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx
Scope              : /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
DisplayName        : TestUser
SignInName         : testuser@devoteamdflab.onmicrosoft.com
RoleDefinitionName : NetworkWatcherCustomRole
RoleDefinitionId   : 15983214-2748-4202-b9eb-c372770c64f4
ObjectId           : 2a65dc30-d562-40f1-b0ab-46889b3c3c37
ObjectType         : User
CanDelegate        : False
```

Fun fact 2: i did not specify the Role definition id which is in the form of a GUID. That will takes more meaning in the Terraform part.

3 Terraform resources for RBAC

After this not so short introduction to Azure RBAC features, let’s have a look at how we can manage role definition and role assignment in Terraform.

3.1 Role definition and role assignment

Terraform comes with two resources associated with RBAC features in Azure. The first one is the Role definition which define the set of permissions in Azure, with a scope and the second one is the role assignment which allows us to associate the Custom Role to Azure AD users. The documentation is giving everything that we may need, as usual. However, all the resources in Azure AD that we used / needed in the previous section are still needed here, except that we do not have Terraform resources associated.

3.2 Sample code

In the following section, we are going in details in samples for both Role definition and Role assignment with Terraform.
For once, i did not make module out of those resource. It’s not that it is impossible to do so but more that i am not convinced yet of the use of a module for a Custom RBAC Role. Indeed, we tend to define role when the builtin ones are not granualar or specific enough. Thus we have to define custom role to fill in the gap for a specific organization. However, once defined, there is not really a need to repeat this process because the role is defined.

I hear some people saying "yes but we could just create the role in the module and add in input the required permissions".
Probably. However, as of now, i think i would rather start with a few workshop reviewing the permissions for each providers and then define custom roles, if needed, by listing those providers operations.
After that i would probably dedicate a template or a section of a template to create in one shot all the custom role and assignment that i would require. But this is only me ☺

3.2.1 Code for Role definition

A simple resource that do not requires so many parameters. Fun fact 3, the role definition id is a GUID and it is required to choose one. We define it through a variable block:

```
# Variable to define the RBAC Role Definition ID

variable "AzureRoleDefinitionId" {

    type    = "string"
    default = "3f0f0bc9-5bf9-463a-95a6-e5920bcc66a8"
}
```

And then we create the resource. The scope, in our case, is the subscription id. After that, we have to specify the actions allowed and the notactions:

```
######################################################
# This file allows the creation of a Custom RBAC role
# for Azure
######################################################

resource "azurerm_role_definition" "Terra-Test_CustomRBACRoleDefinition" {

    role_definition_id      = "${var.AzureRoleDefinitionId}"
    name                    = "Terra-Test_CustomRBACRoleDefinition2"
    scope                   = "/subscriptions/${var.AzureSubscriptionID}"
    description             = "Custom Role for Network Watcher created through
Terraform"

    permissions {

        actions = [
                    "Microsoft.Storage/*/read",
                    "Microsoft.Authorization/*/read",
                    "Microsoft.Resources/subscriptions/resourceGroups/*/read",
                    "Microsoft.Storage/storageAccounts/listServiceSas/*/Action",
                    "Microsoft.Storage/storageAccounts/listAccountSas/*/Action",
                    "Microsoft.Storage/storageAccounts/listKeys/*/Action",
                    "Microsoft.Compute/virtualMachines/*/read",
                    "Microsoft.Compute/virtualMachines/*/write",
                    "Microsoft.Compute/virtualMachineScaleSets/*/read",
                    "Microsoft.Compute/virtualMachineScaleSets/*/write",
                    "Microsoft.Network/networkWatchers/packetCaptures/*/read",
                    "Microsoft.Network/networkWatchers/packetCaptures/*/write",
                    "Microsoft.Network/networkWatchers/packetCaptures/*/delete",
                    "Microsoft.Network/networkWatchers/*/write",
                    "Microsoft.Network/networkWatchers/*/read",
                    "Microsoft.Insights/alertRules/*",
                    "Microsoft.Support/*",
                    "Microsoft.Network/networkWatchers/*",
                    "Microsoft.Network/networkWatchers/*/Action"
            ]
        not_actions = []

    }

    assignable_scopes       = [
                            "/subscriptions/${var.AzureSubscriptionID}",

                            ]
}


output "RoleDefinitionID" {
    value = "${azurerm_role_definition.Terra-Test_CustomRBACRoleDefinition.id}"

}

output "RoleDefinitionName" {
    value = "${azurerm_role_definition.Terra-Test_CustomRBACRoleDefinition.name}"

}
```

I did specify 2 output, just in case.

3.2.2 Code for Role Assignment

In the Role assignment, we make use of the Azure RM provider data sources with a use of the role definition data source. Then we create the resource. But there are some tips to know. First, we need to have a principal_id parameter, which is the object id of the identity which we want to associate to the assignment (a user, a group, or a registered app). Second, while in the data source, the role definition id parameters is the GUID of the role definition, the role definition id in the resource is actually the id of the Azure resource

```
data "azurerm_role_definition" "CustomRBACRole" {

    role_definition_id = "${var.AzureRoleDefinitionId}"
    scope              = "/subscriptions/${var.AzureSubscriptionID}"
}

resource "azurerm_role_assignment" "Terra-TestCustomRBACRoleAssignment" {

    scope                    = "/subscriptions/${var.AzureSubscriptionID}"
    role_definition_id       = "${data.azurerm_role_definition.CustomRBACRole.id}"
    principal_id             = "${var.AZurePrincipalId}"


}

output "CustomRBACRole_DefinitionID" {

    value = "${data.azurerm_role_definition.CustomRBACRole.id}"
}

output "CustomRBACRole_AssignmentID" {

    value = "${azurerm_role_assignment.Terra-TestCustomRBACRoleAssignment.id}"
}
```

After that the same plan and apply will get us the role definition and assignment that we may want.

3.3 Required access in the subscription

In some case, the following error may happen during the apply phase:

```
Error: Error applying plan:

1 error(s) occurred:

* azurerm_role_definition.Terra-Test_CustomRBACRoleDefinition: 1 error(s) occurred:

* azurerm_role_definition.Terra-Test_CustomRBACRoleDefinition:
authorization.RoleDefinitionsClient#CreateOrUpdate: Failure responding to request:
StatusCode=403 -- Original Error: autorest/azure: Service returned an error.
Status=403 Code="AuthorizationFailed" Message="The client '15bcda94-ef96-4708-a900-
```

```
7787c25597d3' with object id '15bcda94-ef96-4708-a900-7787c25597d3' does not have
authorization to perform action 'Microsoft.Authorization/roleDefinitions/write'
over scope '/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'."
```

To solve the issue, we need to identify the impacted resource listed in the error. It seems that the resource with the GUID Is in fact the GUID for the application registered in Azure AD for Terraform.
To make thing simple, the contributor role assigned to Terraform is not enough anymore. We do need more resource to be able to perform the Terraform deployment.
We can (re)check that in the associated actions for the role definition contributor:

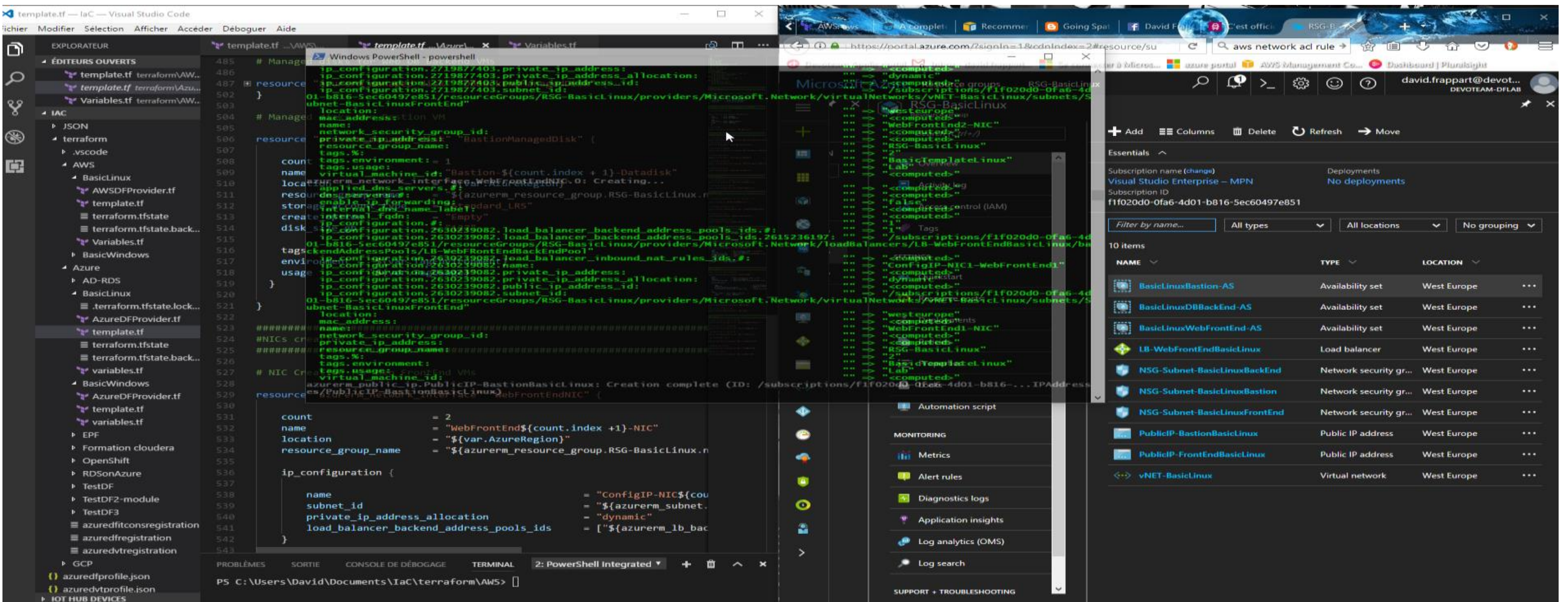
```
PS C:\Users\Users1> (Get-AzureRmRoleDefinition -Name contributor).Actions
*
PS C:\Users\Users1> (Get-AzureRmRoleDefinition -Name contributor).NotActions
Microsoft.Authorization/*/Delete
Microsoft.Authorization/*/Write
Microsoft.Authorization/elevateAccess/Action
```

4 Conclusion

In this article, we reviewed the RBAC concept in Azure, mainly through PowerShell but also a little with Terraform. I hope it was fun ☺
Another fun thing now that we explored the RBAC custom role would be to try diffrent role definition and assignment on a Terraform registered apps. After all, we do register an app at the contributor level at minimum but if only provisioning of VMs are required, it should be possible to limit the available actions on a Terraform app. I'll come back to you when i am finished with this.

Another Tech blog

My thoughts and experiences on Cloud related tek



template.tf — IaC — Visual Studio Code
Fichier Modifier Sélection Afficher Accéder Déboguer Aide
EXPLORATEUR
ÉDITEURS OUVERTS
IAC
Windows PowerShell - powershell
PROBLÈMES SORTIE CONSOLE DE DÉBOGAGE TERMINAL 2: PowerShell Integrated

```
    resource "azurerm_network_interface" "WebFrontEndNIC" {

        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
        }
```

```
  ip_configuration.2719877403.private_ip_address:
  ip_configuration.2719877403.private_ip_address_allocation:
  ip_configuration.2719877403.public_ip_address_id:
  ip_configuration.2719877403.subnet_id:
01-b816-5ec60497e851/resourceGroups/RSG-BasicLinux/providers/Microsoft.Network
ubnet-BasicLinuxFrontEnd"
azurerm_network_interface.WebFrontEndNIC.0: Creating...
azurerm_public_ip.PublicIP-BastionBasicLinux: Creation complete (ID: /subscri
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