本章介紹如何建立獨立的 service principal (service account / iam role) 專門給 terraform 執行。

前面的例子，我們執行 Terraform 都是用 azure ad User 的身份執行。

Review: 我們使用 az-cli az login，讓 terraform 使用本地的 credential 檔案。也就是說 terraform 是使用 login user 的權限來運行。我們可以檢查一下目前使用者的 Azure 權限

az role assignment list --include-inherited --assignee 12345678-1234-1234-1234-123456789012

[

{

"principalName": "chechia\_chechia.net#EXT#@chechianet.onmicrosoft.com",

"principalType": "User",

"roleDefinitionName": "User Access Administrator",

"scope": "/",

"type": "Microsoft.Authorization/roleAssignments"

}

]

Azure AD 的權限則需要通過 Azure portal 查看

* Azure portal -> Azure Active Directory -> User -> Assigned Roles
* 看到 AAD role 是 Global administrator

Global administrator 基本上是超級管理員，權限其實蠻大的，Terraform 的操作並不需要這麼大的權限。

**Issues with Global Admin / POLP(Principle of least privilege)**

Linux 管理常說，沒必要不要使用 root 的權限。

Terraform 也是同樣道理。沒事不要開著權限很大的帳號亂逛。

administrator / owner 權限太大，什麼事都能做很方便。但實務上常常使用 Admin role 其實是有相當風險的，使用 administrator role 操作 Terraform ，違反最小授權原則(POLP: Principle of least privilege)

所有 credential 只要使用，就可能有洩漏或被害的風險，只是管理方式不同，風險高低差異

* 考慮 user credential 不小心洩漏 (exposed)，對全公司的風險
  + administrator 有關閉整個 project 的權限
  + User Access Administrator 有權更改其他 User 的 RBAC，可能竄改 iam 例如：會把其他 admin 移除，綁架整個公有雲

我們上堂課展示過 CI / CD

* 如果要在 Github Action 上有完整的 terraform 執行權限，自然需要把 azure credential 放到 Github Action 上，才能透過 Github Action 操作 terraform plan 與 apply
* 比起個人電腦上的 user credential，放到 Github 上增加暴露的風險
  + 這邊是討論風險，當然是相信 Github Action secret 管理安全才會使用
  + 信任解決方案，與事先考量風險管理是不衝突的（超前部署）

通常 iam / RBAC 管理的權限，與其他 resource provision 的權限會切開，特別管理，連工程師的 User 都不應該有 iam 管理權限

許多整合性的角色權限 administrator, owner, editor, collaborater, ... 之類的權限其實都太大，最佳實踐是用到什麼權限，就開什麼權限

NOTE:

* Azure AD + Azure RBAC 管理在其他公有雲近似於 GCP IAM / AWS IAm role
* Azure service principal 在其他公有雲近似於 GCP service account / AWS iam role

**Prepate Service Principal for Terraform**

在實務上，我們為 terraform 設定專屬的 service principal。

* [Terraform 官方文件](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/guides/service_principal_client_certificate)
* [Azure 官方文件：Service Principal](https://docs.microsoft.com/zh-tw/cli/azure/create-an-azure-service-principal-azure-cli)

我們會需要產生 terraform 專屬的 service principal

登入認證使用非對稱加密的 rsa key/crt，遠比起傳統密碼更加安全

* 首先使用 openssl 產生 key, csr, crt
  + 上傳 public crt 檔案到 azure
  + 本地保留 .key，這隻 private key 就代表 service principal 的完整權限，妥善保管不要外流
* 產生 terraform config 需要的 pfx 格式
  + export pfx key 時會需要輸入密碼，讓每隻私鑰 export 時都有密碼加密是好習慣
  + 這個密碼要記下來，等等 terraform config 時會用到
* 範例中產生的路徑都放在 ~/.ssh，請找一個更適合的地方收藏

KEY\_NAME=~/.ssh/terraform-30-days

openssl req -newkey rsa:4096 -nodes -subj '/CN=terraform-30-days' -keyout ${KEY\_NAME}.key -out ${KEY\_NAME}.csr

openssl x509 -signkey ${KEY\_NAME}.key -in ${KEY\_NAME}.csr -req -days 365 -out ${KEY\_NAME}.crt

openssl pkcs12 -export -out ${KEY\_NAME}.pfx -inkey ${KEY\_NAME}.key -in ${KEY\_NAME}.crt

Getting Private key

Enter Export Password:

Verifying - Enter Export Password:

ls ~/.ssh/terraform-30-days\*

/Users/che-chia/.ssh/terraform-30-days.crt

/Users/che-chia/.ssh/terraform-30-days.csr

/Users/che-chia/.ssh/terraform-30-days.key

/Users/che-chia/.ssh/terraform-30-days.pfx

**Provision service principal**

產生完後我們使用 terraform 來創建 azure service principal。這裡還是使用有 admin 權限的 User 帳號來操作 terraform，範例在 azure/foundation/servcie\_principal，看一下內容

* 為 service principal 命名為 terraform-30-days
* enable\_service\_principal\_certificate=true 使用 certificate 來認證
  + 如果為 false，則會產生傳統密碼，使用密碼登入
  + 指定本地 certificate 的路徑，terraform provision service principal 時會上傳
* role 指派 Contributor，是 [azure 內建的 role](https://docs.microsoft.com/zh-tw/azure/role-based-access-control/built-in-roles#contributor)，沒有 Azure RBAC 的權限，比 owner role 好一點，但其實還是權限過大，先當作範例

*# azure/foundation/service\_principal*

service\_principal\_name = "terraform-30-days"

enable\_service\_principal\_certificate = true

*# https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/guides/service\_principal\_client\_certificate)*

certificate\_path = "/Users/che-chia/.ssh/terraform-30-days.crt"

password\_rotation\_in\_years = 1

*# Adding roles to service principal*

*# The principle of least privilege*

role\_definition\_names = [

"Contributor"

]

這裡參考 [Github kumarvna/terraform-azuread-service-principal](https://github.com/kumarvna/terraform-azuread-service-principal" \t "_blank) 來修改的是 repo 內的 module。module 本身有放上 Terraform Registry 所以可以直接使用，我自己調整內容所以放到本 repo。細節可以看 azure/modules/azuread\_service\_principal

確定沒問題後 provision resource

terragrunt init

terragrunt plan

terragrunt apply

Azure 上就會產生 service principal，使用 certificate 做登入認證

**Test az login**

測試一下 az-cli 是否能夠登入 service principal，[Azure 官方文件：Service Principal](https://docs.microsoft.com/zh-tw/cli/azure/create-an-azure-service-principal-azure-cli" \t "_blank)

* 首先把 .key 與 .crt 合併成另一隻 keycrt 檔案
* az ad sp list 中尋找 service principal 的細節
  + 需要 AD 的 tenant id
  + 需要 service principal 的 name id
* 然後執行 az login --service-principal，使用 keycrt 檔案做認證
* 登入後拿到 service principal 的登入訊息，此時本地 az-cli credential 就不是 User 的身份了

cat ~/.ssh/terraform-30-days.key > ~/.ssh/terraform-30-days.keycrt

cat ~/.ssh/terraform-30-days.crt >> ~/.ssh/terraform-30-days.keycrt

APP\_NAME=terraform-30-days

az ad sp list --display-name ${APP\_NAME}

TENANT\_ID=$(az ad sp list --display-name ${APP\_NAME} | jq -r '.[0].appOwnerTenantId')

SERVICE\_NAME=$(az ad sp list --display-name ${APP\_NAME} | jq -r '.[0].servicePrincipalNames[0]')

az login --service-principal \

--username ${SERVICE\_NAME} \

--tenant ${TENANT\_ID} \

--password ~/.ssh/terraform-30-days.keycrt > /tmp/azure-login-profile

[

{

"cloudName": "AzureCloud",

"homeTenantId": "12345678-1234-1234-1234-123456789012",

"id": "12345678-1234-1234-1234-123456789012",

"isDefault": true,

"managedByTenants": [],

"name": "Microsoft Azure Sponsorship",

"state": "Enabled",

"tenantId": "12345678-1234-1234-1234-123456789012",

"user": {

"name": "12345678-1234-1234-1234-123456789012",

"type": "servicePrincipal"

}

}

]

目前身份是 service principal，檢查一下自身權限 role assignment，看見正確設定的 role: Contributor

az role assignment list

[

{

"principalType": "ServicePrincipal",

...

"roleDefinitionName": "Contributor",

...

}

]

**Config Terraform to use service principal**

接下來要設定 terraform

有兩個方法

* 直接更改 provider.tf 裡面的 azurerm {} 把 client id, ceritificate 等等參數填入
* 使用環境變數傳入參數，terraform 會使用 environment variable，overwrite azurerm 內的設定

這邊示範使用環境變數，之後夾到 Github Action 上，或是其他 CI 上會比較方便，不用再更改 provider.tf 的原始碼

* 這邊可以自己從 az-cli 取得登入資訊
* export 各個環境變數
* env | grep ARM 檢查剛剛 export 的參數

export ARM\_CLIENT\_ID="00000000-0000-0000-0000-000000000000"

export ARM\_CLIENT\_CERTIFICATE\_PATH="/Users/che-chia/.ssh/terraform-30-days.pfx"

export ARM\_CLIENT\_CERTIFICATE\_PASSWORD=<password> *# change this*

export ARM\_SUBSCRIPTION\_ID="00000000-0000-0000-0000-000000000000"

export ARM\_TENANT\_ID="00000000-0000-0000-0000-000000000000"

export ARM\_CLIENT\_ID=$(cat /tmp/azure-login-profile | jq -r '.[0].user.name')

export ARM\_CLIENT\_CERTIFICATE\_PATH="/Users/che-chia/.ssh/terraform-30-days.pfx"

export ARM\_CLIENT\_CERTIFICATE\_PASSWORD=<password> *# change this*

export ARM\_SUBSCRIPTION\_ID=$(az account subscription list | jq -r '.[0].subscriptionId')

export ARM\_TENANT\_ID=$(cat /tmp/azure-login-profile | jq -r '.[0].tenantId')

env | grep ARM

ARM\_CLIENT\_CERTIFICATE\_PATH=/Users/che-chia/.ssh/terraform-30-days.pfx

ARM\_CLIENT\_ID=

ARM\_SUBSCRIPTION\_ID=

ARM\_TENANT\_ID=

ARM\_CLIENT\_CERTIFICATE\_PASSWORD=

如果參數都正常，

terragrunt init && terragrunt plan

嘗試更改自己 service principal 的 role，看能不能 Privilege escalation，把自己從 Contributor 變成 Owner。如果可以的話，terraform 可以自己提升權限變成 Owner，然後近來 azure 亂改

*# azure/foundation/service\_principal*

role\_definition\_names = [

"Contributor",

"Owner" *# try Privilege escalation*

]

然後再次 apply 看看計謀會不會得逞

terragrunt plan

terragrunt apply

│ Error: Could not set Owners

│

│ with azuread\_application.main,

│ on application.tf line 4, in resource "azuread\_application" "main":

│ 4: owners = [data.azuread\_client\_config.current.object\_id]

│ Error: authorization.RoleAssignmentsClient#Create: Failure responding to request: StatusCode=403 *-- Original Error: autorest/azure: Service returned an error. Status=403 Code="AuthorizationFailed" Message="The client '12345678-1234-1234-1234-123456789012' with object id '12345678-1234-1234-1234-123456789012' does not have authorization to perform action 'Microsoft.Authorization/roleAssignments/write' over scope '/subscriptions/ba8eb346-d19e-4f65-96d9-8c783e6eea61' or the scope is invalid. If access was recently granted, please refresh your credentials."*

Plan 都正常，表示 read / refresh azure 上的資源時，service principal 的 contributor 全縣市足夠的

apply 則回傳 403 permission denied 以及相關錯誤訊息，azure 表示 terraform 的 service principal 沒有權限可以調整 RBAC。表示 service principal 權限是受到限制的，符合我們的預期。

之後 Terraform 的操作，我們就不再使用 az login 來產生本地的 credential，而是使用 service principal

* 各位可以熟悉一下 terraform 環境變數的設定
* 除了調整 RBAC 時，還是需要開有 rbac 管理權限的 Owner 帳號，其他時間使用 service principal 就足夠
* 覺得每次 export 很麻煩的話，可以自行修改 azure/terragrunt.hcl，這段，具體修改可以參考 [Terraform 官方說明文件](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/guides/service_principal_client_certificate#configuring-the-service-principal-in-terraform)

# azure/provider

generate "provider" {

path = "provider.tf"

if\_exists = "overwrite\_terragrunt"

contents = <<EOF

provider "azurerm" {

features {}

}

EOF

}

但是拜託

密碼跟 certificate .key .pfx 拜託拜託不要 commit 到 git 裏面，務必每次手動輸入  
密碼跟 certificate .key .pfx 拜託拜託不要 commit 到 git 裏面，務必每次手動輸入  
密碼跟 certificate .key .pfx 拜託拜託不要 commit 到 git 裏面，務必每次手動輸入

很重要所以講三次，但還是會看到有人 git add 就把 password commit 進去

**Homework**

* LOPL，移除 contributor role，[參考 Azure 內建的 role 清單](https://docs.microsoft.com/en-us/azure/role-based-access-control/built-in-roles#contributor)，來做更精細的 RBAC，可能的 role
  + Virtual Machine Contributor
  + Network Contributor
  + Storage Account Contributor
  + Storage Blob Data Owner
* 承上，apply service principal 之後，回去修改以前的範例，是否仍有足夠的權限？還是需要再調整？
* provision 另外一個 service account，只有 RBAC 權限，但沒有更改其他 resource 的權限，例如：不能存取 storage, compute, networking, ...

**References**

* <https://github.com/kumarvna/terraform-azuread-service-principal>