

14/11/2022

TP2

COLLATY Srikanth
KOBESSI Bessel



Fabieng <fabien@flexcorp.fr>

Objet : RE: Bienvenue chez FlexCorp !

Bonjour *Bessel et Srikanth*,

Bravo pour votre job sur la démo client hier. Comme vous avez pu le voir nos serveurs en interne sont clairement en PLS comme disent les djeuns. Je pense qu'il est temps que FlexCorp devienne une vraie entreprise du Web 5.0 et passe son infrastructure dans le cloud. Je vous laisse me faire des propositions sur ce qu'il est possible de faire.

D'ailleurs sinon Dave le Dev a développé une superbe intelligence artificielle qui permet de gérer la distribution de croquettes pour la future gamelle connectée de notre client Pédigree. J'aurai besoin que vous installiez asap pour une démo client. Je compte on you.

A très bientôt au babyfoot !

Fabieng
Chief Chef Manager

Travail à effectuer

Lien gitlab : <https://gitlab.esiea.fr/bkobeissi/admin>

Découverte de Terraform

1. Installez Terraform ainsi que la console Azure sur un environnement Unix/Linux sur votre ordinateur. Il est déconseillé de prendre la version disponible dans votre gestionnaire de paquets favoris. Expliquez pourquoi il vaut mieux utiliser la version présente sur le site des éditeurs directement (dans certains cas).

```
[bessel@localhost ~]$ sudo yum -y install terraform
hashicorp Stable - x86_64                2.0 MB/s | 915 KB    00:00
Dernière vérification de l'expiration des métadonnées effectuée il y a 0:00:01 le mar. 15 nov. 2022 09:22:53 CET.
Dépendances résolues.
=====
Paquet      Architecture  Version  Dépôt      Taille
-----
Installation:
terraform   x86_64       1.3.4-1  hashicorp  13 M
Résumé de la transaction
=====
Installer 1 Paquet
Taille totale des téléchargements : 13 M
Taille des paquets installés : 58 M
Téléchargement des paquets :
terraform-1.3.4-1.x86_64.rpm          4.0 MB/s | 13 MB    00:03
-----
Total                                     3.9 MB/s | 13 MB    00:03
hashicorp Stable - x86_64              49 KB/s | 3.1 KB    00:00
Import de la clé GPG 0xA3219F7B :
Utilisateur : « HashiCorp Security (HashiCorp Package Signing) <security+packaging@hashicorp.com> »
Empreinte : 88A0 32E0 94D8 1B4E A2B9 0270 04A1 8C8B A321 9F7B
Provenance : https://rpm.releases.hashicorp.com/gpg
La clé a bien été importée
Test de la transaction
La vérification de la transaction a réussi.
Lancement de la transaction de test
Transaction de test réussie.
Exécution de la transaction
Préparation      :
Installation     : terraform-1.3.4-1.x86_64          1/1
Vérification de : terraform-1.3.4-1.x86_64          1/1
Installé:
terraform-1.3.4-1.x86_64
Terminé !
```

Installation de Terraform

```
[bessel@localhost ~]$ sudo dnf install azure-cli
packages-microsoft-com-prod
packages-microsoft-com-prod
Dernière vérification de l'expiration des métadonnées effectuée il y a 0:00:04 le mar. 15 nov. 2022 09:25:25 CET.
Dépendances résolues.
=====
Paquet      Architecture  Version
-----
Installation:
azure-cli    x86_64       2.42.0-1.el8
Résumé de la transaction
=====
Installer 1 Paquet
Taille totale des téléchargements : 54 M
Taille des paquets installés : 808 M
Voulez-vous continuer ? [o/N] : o
Téléchargement des paquets :
azure-cli-2.42.0-1.el8.x86_64.rpm
-----
Total
packages-microsoft-com-prod
Import de la clé GPG 0x0E1229CF :
Utilisateur : « Microsoft (Release signing) <gpgsecurity@microsoft.com> »
Empreinte : B552 0606 B560 79E3 30B3 721C FB3F 94AD 0E12 29CF
Provenance : https://packages.microsoft.com/keys/microsoft.asc
Voulez-vous continuer ? [o/N] : o
La clé a bien été importée
Test de la transaction
La vérification de la transaction a réussi.
Lancement de la transaction de test
Transaction de test réussie.
Exécution de la transaction
Préparation      :
Installation     : azure-cli-2.42.0-1.el8.x86_64      1/1
Exécution du scriptlet : azure-cli-2.42.0-1.el8.x86_64
Vérification de : azure-cli-2.42.0-1.el8.x86_64      1/1
Installé:
azure-cli-2.42.0-1.el8.x86_64
Terminé !
```

Installation d'Azure-cli

➔ Il est préférable d'utiliser les versions présentes sur le site de l'éditeur car les cli ont un risque de ne pas être à jour, nous utiliserons ceux présents via l'éditeur.

2. Connectez-vous à Azure avec la console.

```
[bessel@localhost ~]$ az login
A web browser has been opened at https://login.microsoftonline.com/organizations/oauth2/v2.0/authorize. Please continue the login
open, use device code flow with 'az login --use-device-code'.
The following tenants don't contain accessible subscriptions. Use 'az login --allow-no-subscriptions' to have tenant level access
07bd03e6-f6a2-4c1f-83be-b47d1a6453f9 'SKEMA Business School'
{
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "2397b885-afe4-4bc7-9e1e-e4899ef44d1a",
    "id": "ec0774c4-9816-4709-a591-61e482e014ea",
    "isDefault": true,
    "managedByTenants": [],
    "name": "Azure pour les étudiants",
    "state": "Enabled",
    "tenantId": "2397b885-afe4-4bc7-9e1e-e4899ef44d1a",
    "user": {
      "name": "bkobeissi@et.esiea.fr",
      "type": "user"
    }
  }
}
```

Connexion à Azure avec la console

3. Lancez votre premier script Terraform « main.tf » comme suit :

```
# Configure the Microsoft Azure Provider
terraform {
  required_providers {
    azurerm = {
      source = "hashicorp/azurerm"
      version = "~>2.0"
    }
  }
}
provider "azurerm" {
  features {}
}

# Create a resource group if it doesn't exist
resource "azurerm_resource_group" "myterraformgroup" {
  name     = "myResourceGroup"
  location = "eastus"

  tags = {
    environment = "Terraform Demo"
  }
}

# Create virtual network
resource "azurerm_virtual_network" "myterraformnetwork" {
  name                = "myVnet"
  address_space       = ["10.0.0.0/16"]
  location             = "eastus"
  resource_group_name = azurerm_resource_group.myterraformgroup.name

  tags = {
    environment = "Terraform Demo"
  }
}

# Create subnet
resource "azurerm_subnet" "myterraformsubnet" {
  name                 = "mySubnet"
  resource_group_name = azurerm_resource_group.myterraformgroup.name
  virtual_network_name = azurerm_virtual_network.myterraformnetwork.name
  address_prefixes     = ["10.0.1.0/24"]
}

# Create public IPs
resource "azurerm_public_ip" "myterraformpublicip" {
  name     = "myPublicIP"
  location = "eastus"
}
```

```

    resource_group_name      =
azurerm_resource_group.myterraformgroup.name
    allocation_method        = "Dynamic"

    tags = {
        environment = "Terraform Demo"
    }
}

# Create Network Security Group and rule
resource "azurerm_network_security_group" "myterraformnsg" {
    name                = "myNetworkSecurityGroup"
    location             = "eastus"
    resource_group_name = azurerm_resource_group.myterraformgroup.name

    security_rule {
        name                = "SSH"
        priority            = 1001
        direction           = "Inbound"
        access              = "Allow"
        protocol            = "Tcp"
        source_port_range   = "*"
        destination_port_range = "22"
        source_address_prefix = "*"
        destination_address_prefix = "*"
    }

    tags = {
        environment = "Terraform Demo"
    }
}

# Create network interface
resource "azurerm_network_interface" "myterraformnic" {
    name                = "myNIC"
    location             = "eastus"
    resource_group_name = azurerm_resource_group.myterraformgroup.name

    ip_configuration {
        name                = "myNicConfiguration"
        subnet_id           = azurerm_subnet.myterraformsubnet.id
        private_ip_address_allocation = "Dynamic"
        public_ip_address_id =
azurerm_public_ip.myterraformpublicip.id
    }

    tags = {
        environment = "Terraform Demo"
    }
}

# Connect the security group to the network interface
resource "azurerm_network_interface_security_group_association" "example" {
    network_interface_id = azurerm_network_interface.myterraformnic.id
    network_security_group_id =
azurerm_network_security_group.myterraformnsg.id
}

# Generate random text for a unique storage account name
resource "random_id" "randomId" {
    keepers = {
        # Generate a new ID only when a new resource group is defined
        resource_group = azurerm_resource_group.myterraformgroup.name
    }
}

```

```

    }

    byte_length = 8
}

# Create storage account for boot diagnostics
resource "azurerm_storage_account" "mystorageaccount" {
    name                = "diag${random_id.randomId.hex}"
    resource_group_name =
azurerm_resource_group.myterraformgroup.name
    location            = "eastus"
    account_tier        = "Standard"
    account_replication_type = "LRS"

    tags = {
        environment = "Terraform Demo"
    }
}

# Create the SSH key
resource "tls_private_key" "example_ssh" {
    algorithm = "RSA"
    rsa_bits = 4096
}

# Create virtual machine
resource "azurerm_linux_virtual_machine" "myterraformvm" {
    name                = "myVM"
    location            = "eastus"
    resource_group_name = azurerm_resource_group.myterraformgroup.name
    network_interface_ids = [azurerm_network_interface.myterraformnic.id]
    size                = "Standard_DS1_v2"

    os_disk {
        name                = "myOsDisk"
        caching             = "ReadWrite"
        storage_account_type = "Premium_LRS"
    }

    source_image_reference {
        publisher = "Canonical"
        offer     = "0001-com-ubuntu-server-focal"
        sku       = "20_04-lts"
        version   = "latest"
    }

    computer_name     = "myvm"
    admin_username    = "azureuser"
    disable_password_authentication = true

    admin_ssh_key {
        username     = "azureuser"
        public_key   = tls_private_key.example_ssh.public_key_openssh
    }

    boot_diagnostics {
        storage_account_uri =
azurerm_storage_account.mystorageaccount.primary_blob_endpoint
    }

    tags = {
        environment = "Terraform Demo"
    }
}

```

- Vérifiez, planifiez et appliquez ce script Terraform. Vérifiez sur votre console Azure que la VM de test est bien créée.

```
[bessel@localhost ~]$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/random...
- Finding latest version of hashicorp/tls...
- Finding hashicorp/azurerm versions matching "~> 2.0"...
- Installing hashicorp/random v3.4.3...
- Installing hashicorp/random v3.4.3 (signed by HashiCorp)
- Installing hashicorp/tls v4.0.4...
- Installing hashicorp/tls v4.0.4 (signed by HashiCorp)
- Installing hashicorp/azurerm v2.99.0...
- Installing hashicorp/azurerm v2.99.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!
```

Terraform init qui va initialiser le contexte recherché ainsi que les variables

```
[bessel@localhost ~]$ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# azure_rm_linux_virtual_machine.myterraformvm will be created
+ resource "azure_rm_linux_virtual_machine" "myterraformvm" {
  + admin_username      = "azureuser"
  + allow_extension_operations = true
  + computer_name        = "myvm"
  + disable_password_authentication = true
  + extensions_time_budget = "PT1H30M"
  + id                  = (known after apply)
  + location             = "eastus"
  + max_bid_price        = 1
  + name                 = "myVM"
  + network_interface_ids = (known after apply)
  + patch_mode           = "ImageDefault"
  + platform_fault_domain = -1
  + priority              = "Regular"
  + private_ip_address    = (known after apply)
  + private_ip_addresses  = (known after apply)
  + provision_vm_agent    = true
  + public_ip_address     = (known after apply)
  + public_ip_addresses   = (known after apply)
  + resource_group_name   = "MYRESOURCEGROUP"
  + size                 = "Standard_D1s_v2"
  + tags                 = {
    + "environment" = "Terraform Demo"
  }
  + virtual_machine_id    = (known after apply)
  + zone                  = (known after apply)
}

# tls_private_key.example_ssh will be created
+ resource "tls_private_key" "example_ssh" {
  + algorithm      = "RSA"
  + ecdsa_curve    = "P224"
  + id             = (known after apply)
  + private_key_openssh = (sensitive value)
  + private_key_pem    = (sensitive value)
  + private_key_pem_pkcs8 = (sensitive value)
  + public_key_fingerprint_md5 = (known after apply)
  + public_key_fingerprint_sha256 = (known after apply)
  + public_key_openssh = (known after apply)
  + public_key_pem     = (known after apply)
  + rsa_bits          = 4096
}

Plan: 11 to add, 0 to change, 0 to destroy.
```

Terraform apply qui va faire un test à blanc afin de s'assurer que tout fonctionne bien

```
[bessel@localhost ~]$ terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# azure_rm_linux_virtual_machine.myterraformvm will be created
+ resource "azure_rm_linux_virtual_machine" "myterraformvm" {
  + admin_username      = "azureuser"
  + allow_extension_operations = true
  + computer_name        = "myvm"
  + disable_password_authentication = true
  + extensions_time_budget = "PT1H30M"
  + id                  = (known after apply)
  + location             = "eastus"
  + max_bid_price        = 1
  + name                 = "myVM"
  + network_interface_ids = (known after apply)
  + patch_mode           = "ImageDefault"
  + platform_fault_domain = -1
  + priority              = "Regular"
}

Apply complete! Resources: 11 added, 0 changed, 0 destroyed.

[bessel@localhost ~]$ yum install graphviz
Erreur : Cette commande doit être exécutée avec les privilèges super-utilisateur (sous l'utilisateur root sur la plupart des systèmes).
[bessel@localhost ~]$ sudo yum install graphviz
[sudo] Mot de passe de bessel :
Dernière vérification de l'expiration des métadonnées effectuée il y a 0:24:01 le mar. 15 nov. 2022 09:25:25 CET.
Dépendances résolues.

=====
Paquet                               Architecture      Version           Dépôt             Taille
-----
Installation:
graphviz                             x86_64            2.40.1-43.el8     appstream          1.7 M
Installation des dépendances:
libXaw                               x86_64            1.0.13-10.el8     appstream          194 k
xorg-x11-fonts-ISO8859-1-100dpi      noarch            7.5-19.el8        appstream          1.1 M
=====
Résumé de la transaction
Installer 3 Paquets

Taille totale des téléchargements : 3.0 M
Taille des paquets installés : 9.0 M
Voulez-vous continuer ? [y/N] : 0
Téléchargement des paquets :
(1/3): libXaw-1.0.13-10.el8.x86_64.rpm                               554 kB/s | 194 kB  00:00
(2/3): xorg-x11-fonts-ISO8859-1-100dpi-7.5-19.el8.noarch.rpm        1.6 MB/s | 1.1 MB  00:00
(3/3): graphviz-2.40.1-43.el8.x86_64.rpm                             2.3 MB/s | 1.7 MB  00:00
```

Terraform apply qui permet de créer un dossier et générer la machine virtuelle

```
[bessel@localhost ~]$ az vm list -otable
Name      ResourceGroup  Location  Zones
-----
myVM      MYRESOURCEGROUP  eastus
```

Vérification de la VM créée via le script

5. Ajoutez votre script Terraform à un repository git stockée sur le gitlab de l'ESIEA (gitlab.esiea.fr). Cela vous servira pour votre rendu final.

```
[bessel@localhost ~]$ ssh-keygen -t rsa -C bkobeissi@et.esiea.fr
Generating public/private rsa key pair.
Enter file in which to save the key (/home/bessel/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/bessel/.ssh/id_rsa.
Your public key has been saved in /home/bessel/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:xcRGH7r5oxywEtLbNzpfSL6JKKiMsxt2YddwhcBjE bkobeissi@et.esiea.fr
The key's randomart image is:
+---[RSA 3072]---+
|  Eo oo . |
| .o 000 . |
| .o .+. . |
| + o o |
| . * S o |
| + % +. |
| o..o * X o o |
| +0++..o * + o . |
| =0+ . . . . o |
+---[SHA256]-----+
```

Génération de la clé SSH

```
[bessel@localhost ~]$ cat < ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGDwQDwLwX+Yw4op2F8u0bJDe+njHS1tK0uzwwe+AiH+87dNg40gmvlqYFotKuGp8b8x5Zkn15HafxfyT40+65Ipi45Gv41hcTCTF4q/q/VRssbErLos8v4tr9M0v3my/1S++0617/z0wB1kaF9FFv4v56H5mKQ8se0zCTT3R0
R/R2XK7067mdy42K1u8REzX7b5B5A3bR1gjkBy4FgTdgSh1fLx0LmgckN0BAFVlnt+Xjn1SwqavL5K9quXLx0DcP//0MP+Dnpz0j312PdbvTh7X5qb0F0RcnKQK0Cj9/xx26rv6VvFpBLQEXq1bwYDWR0CuRaRN1dpCz2ULm0LjYjACmMshJruCwh+JPsbvT12XCNz0eWt
VF13B373CfK3Y7npHaw9SA/uVbcyPUXEFzK1MC4cdr/TNArj+Khc08CRgFEYw4BPy7Uv04e3yb0dNZ0e4mm0BoLYC5b+AEYCF9wZD044Q9GvH1Jkq0KfYDms- bkobeissi@et.esiea.fr
```

Clé SSH

```
[bessel@localhost ~]$ git clone git@gitlab.esiea.fr:bkobeissi/admin.git
Clonage dans 'admin'...
The authenticity of host 'gitlab.esiea.fr (185.235.207.37)' can't be established.
ECDSA key fingerprint is SHA256:a3ZcEzodnZyiv3tZzuVb9+UMn2VnxRRw0B1qyVaahJw.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'gitlab.esiea.fr,185.235.207.37' (ECDSA) to the list of known hosts.
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Réception d'objets: 100% (3/3), fait.
```

Le répertoire a bien été récupéré

```
[bessel@localhost ~]$ ls
admin  Bureau  Documents  Images  main.tf  Modèles  Musique  nginx_signing.key  Public  setup  Téléchargements  terraform-graph.png  terraform.tfstate  Vidéos
[bessel@localhost ~]$ mv main.tf admin
[bessel@localhost ~]$ cd admin
[bessel@localhost admin]$ ls
main.tf  README.md
```

main.tf a bien été ajouté dans le dossier Admin

```
[bessel@localhost admin]$ git init
Dépôt Git existant réinitialisé dans /home/bessel/admin/.git/
[bessel@localhost admin]$ git add main.tf
[bessel@localhost admin]$ git commit -m "first push"
[main 0a21b01] first push
1 file changed, 169 insertions(+)
create mode 100644 main.tf
[bessel@localhost admin]$ git push
Énumération des objets: 4, fait.
Décompte des objets: 100% (4/4), fait.
Compression des objets: 100% (3/3), fait.
Écriture des objets: 100% (3/3), 1.55 Mio | 1.55 Mio/s, fait.
Total 3 (delta 0), réutilisés 0 (delta 0), réutilisés du pack 0
To gitlab.esiea.fr:bkobeissi/admin.git
368825a..0a21b01 main -> main
[bessel@localhost admin]$
```

Initialisation et configuration de git

Le fichier a bien été push

6. Modifiez le script Terraform afin d'afficher la ou les adresses IP publiques de la VM à la fin de l'exécution.

```
output "azurerm_public_ip" {  
    # description = "Public IP address"  
    value = "${azurerm_public_ip.myterraformpublicip.ip_address}"  
}
```

Modification du script terraform dans main.tf

Outputs:

azurerm_public_ip = "40.117.198.210"

Sortie attendue avec l'adresse IP publique de la VM

7. Modifiez le script Terraform afin de stocker la clé SSH de connexion (format PEM) avec les bonnes autorisations. Ajoutez ensuite cette clé à votre trousseau SSH et tentez de vous connecter à votre VM.

```
# local_file.ssh will be created  
+ resource "local_file" "ssh" {  
+   content           = (sensitive)  
+   directory_permission = "0777"  
+   file_permission   = "0777"  
+   filename          = "fichier.pem"  
+   id                = (known after apply)  
+ }  
  
Plan: 1 to add, 0 to change, 0 to destroy.  
  
Do you want to perform these actions?  
Terraform will perform the actions described above.  
Only 'yes' will be accepted to approve.  
  
Enter a value: yes  
  
local_file.ssh: Creating...  
local_file.ssh: Creation complete after 0s [id=73ce71489e00b103045a26ebbe383a0edabbb696]
```

Création de la clé SSH au format PEM

```
[bessel@localhost admin]$ ssh azureuser@40.117.198.210 -i fichier.pem  
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.15.0-1022-azure x86_64)  
  
 * Documentation:  https://help.ubuntu.com  
 * Management:    https://landscape.canonical.com  
 * Support:       https://ubuntu.com/advantage  
  
System information as of Tue Nov 15 14:47:14 UTC 2022  
  
System load:  0.0               Processes:    113  
Usage of /:   5.0% of 28.89GB   Users logged in:  0  
Memory usage: 7%               IPv4 address for eth0: 10.0.1.4  
Swap usage:   0%  
  
 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s  
   just raised the bar for easy, resilient and secure K8s cluster deployment.  
  
   https://ubuntu.com/engage/secure-kubernetes-at-the-edge  
  
0 updates can be applied immediately.  
  
The list of available updates is more than a week old.  
To check for new updates run: sudo apt update  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
azureuser@myvm:~$
```

Connexion à la VM grâce à notre clé fichier.pem

Félicitations, vous devriez maintenant être en mesure de déployer et de vous connecter à votre machine virtuelle sur Microsoft Azure 😊

Ansible + Terraform = <3

1. En vous connectant « à la main » à votre machine virtuelle qui vient d'être déployée, installez Ansible.

```
[bessel@localhost admin]$ ansible --version
ansible [core 2.13.5]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/bessel/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.9/site-packages/ansible
  ansible collection location = /home/bessel/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.9.13 (main, Jun 24 2022, 15:32:51) [GCC 8.5.0 20210514 (Red Hat 8.5.0-13)]
  jinja version = 3.1.2
  libyaml = True
[bessel@localhost admin]$
```

Ansible est bien installé sur notre machine

2. En créant un nouveau script Terraform, déployez 3 serveurs Linux de la distribution de votre choix.

```
+ tags = {
+   + "environment" = "Terraform Demo"
+ }
+ virtual_machine_id = (known after apply)

+ ip_configuration {
+   + gateway_load_balancer_frontend_ip_configuration_id = (known after apply)
+   + name = "myNicConfiguration"
+   + primary = (known after apply)
+   + private_ip_address = (known after apply)
+   + private_ip_address_allocation = "Dynamic"
+   + private_ip_address_version = "IPv4"
+   + subnet_id = (known after apply)
+ }
}

# azurerm_network_interface_security_group_association.example[0] will be created
+ resource "azurerm_network_interface_security_group_association" "example" {
+   id = (known after apply)
+   network_interface_id = (known after apply)
+   network_security_group_id = (known after apply)
+ }






























# azurerm_network_interface_security_group_association.example[1] will be created
+ resource "azurerm_network_interface_security_group_association" "example" {
+   id = (known after apply)
+   network_interface_id = (known after apply)
+   network_security_group_id = (known after apply)
+ }

# azurerm_network_interface_security_group_association.example[2] will be created
+ resource "azurerm_network_interface_security_group_association" "example" {
+   id = (known after apply)
+   network_interface_id = (known after apply)
+   network_security_group_id = (known after apply)
+ }
```

Les 3 VM ont été créés

Machines virtuelles ...

Groupe ESIEA (et.esiea.fr)

| + Créer  Passer au mode classique  Réservations  Gérer la vue  Actualiser  Exporter au format CSV  Ouvrir une requête  Attribuer des étiquettes  Démarrer  Redémarrer  Arrêter  Supprimer  | | | | | | | |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------|
| Filtrer un champ...  Abonnement égal à tout  Type égal à tout  Groupe de ressources égal à tout  Emplacement égal à tout   Ajouter un filtre | | | | | | | |
| <input type="checkbox"/> Nom  | Type  | Abonnement  | Groupe de ressources  | Emplacement  | État  | Système d'exploitation  | Taille  |
| <input type="checkbox"/>  myterraformvm0 | Machine virtuelle | Azure pour les étudiants | myResourceGroup | East US | En cours d'exécution | Linux | Standard_DS1_v2 |
| <input type="checkbox"/>  myterraformvm1 | Machine virtuelle | Azure pour les étudiants | myResourceGroup | East US | En cours d'exécution | Linux | Standard_DS1_v2 |
| <input type="checkbox"/>  myterraformvm2 | Machine virtuelle | Azure pour les étudiants | myResourceGroup | East US | En cours d'exécution | Linux | Standard_DS1_v2 |

Les 3 VM sont bien visibles sur le portail Azure

```
# Create Network Security Group and rule
resource "azurerm_network_security_group" "myterraformmsg" {
  count = 3
  # name = "myNetworkSecurityGroup"
  #pour créer 3 différents groupe de sécurité
  name = "myterraformmsg${count.index}"
  location = "eastus"
  resource_group_name = azurerm_resource_group.myterraformgroup.name

  security_rule {
    name = "SSH"
    priority = 1001
    direction = "Inbound"
    access = "Allow"
    protocol = "Tcp"
    source_port_range = "*"
    destination_port_range = "22"
    source_address_prefix = "*"
    destination_address_prefix = "*"
  }

  tags = {
    environment = "Terraform Demo"
  }
}
```

Modification du main.tf pour créer les 3 différents groupes de sécurité

```
# Create network interface
resource "azurerm_network_interface" "myterraformnic" {
  count = 3
  # name = "myNIC"
  name = "myterraformnic${count.index}"
  location = "eastus"
  resource_group_name = azurerm_resource_group.myterraformgroup.name

  ip_configuration {
    name = "myNicConfiguration"
    subnet_id = azurerm_subnet.myterraformsubnet.id
    private_ip_address_allocation = "Dynamic"
    # nous avons désactivé l'IP publique pour ne pas que les 3 vm ont une ip identiques
    # public_ip_address_id = azurerm_public_ip.myterraformpublicip.id
  }

  tags = {
    environment = "Terraform Demo"
  }
}
```

Désactivation de l'IP publique pour ne pas que les 3 vms ont une ip identiques

```
# Connect the security group to the network interface
resource "azurerm_network_interface_security_group_association" "example" {
  count = 3
  # network_interface_id = azurerm_network_interface.myterraformnic.id
  # pour allouer les IP aux 3 VMs
  network_interface_id = azurerm_network_interface.myterraformnic[count.index].id
  # network_security_group_id = azurerm_network_security_group.myterraformmsg.id
  # pour allouer les IP à un groupe de réseau
  network_security_group_id = azurerm_network_security_group.myterraformmsg[count.index].id
}
```

Modification des groupes de sécurité afin d'allouer des IP aux 3 VMs et ainsi les allouer à un groupe de réseau

- Grâce au playbook Ansible fait à la fin de votre précédent TP, déployez nginx et les 3 serveurs que vous venez de créer. Pour économiser des ressources (financières), ne mettez pas d'IP publique à vos 3 serveurs. Configurez votre serveur « principal » (celui fait précédemment) comme machine de rebond pour vos connexions SSH. Cette machine doit aussi pouvoir rediriger les connexions réseaux du réseau interne à internet (ip forwarding, route à configurer, etc). Configurez votre serveur « principal » en reverse proxy et load balancer comme pour le précédent TP. Vérifiez que vous pouvez avoir accès à votre serveur depuis l'extérieur.

| | | | | | | | | | | | |
|--------------------------|----------------|-------------------|--------------------------|-----------------|---------|----------------------|-------|-----------------|----------------|---|-----|
| <input type="checkbox"/> | myterraformvm0 | Machine virtuelle | Azure pour les étudiants | myResourceGroup | East US | En cours d'exécution | Linux | Standard_DS1_v2 | - | 1 | ... |
| <input type="checkbox"/> | myterraformvm1 | Machine virtuelle | Azure pour les étudiants | myResourceGroup | East US | En cours d'exécution | Linux | Standard_DS1_v2 | - | 1 | ... |
| <input type="checkbox"/> | myterraformvm2 | Machine virtuelle | Azure pour les étudiants | myResourceGroup | East US | En cours d'exécution | Linux | Standard_DS1_v2 | - | 1 | ... |
| <input type="checkbox"/> | myVM | Machine virtuelle | Azure pour les étudiants | myResourceGroup | East US | En cours d'exécution | Linux | Standard_DS1_v2 | 172.174.25.149 | 1 | ... |

Nos 4 VMs sont bien présentes (myVM est la seule avec une IP publique comme demandé)

```
[bessel@localhost admin]$ chmod 700 fichier.pem
[bessel@localhost admin]$ ssh azureuser@172.174.25.149 -i fichier.pem
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.15.0-1023-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Thu Nov 17 18:04:46 UTC 2022

System load:  0.22               Processes:    116
Usage of /:   5.0% of 28.89GB    Users logged in: 0
Memory usage: 8%                IPv4 address for eth0: 10.0.1.7
Swap usage:   0%

0 updates can be applied immediately.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

azureuser@myvm:~$
```

Connexion réussie en SSH vers myVM

```
azureuser@myvm:~$ ping 10.0.1.5
PING 10.0.1.5 (10.0.1.5) 56(84) bytes of data.
64 bytes from 10.0.1.5: icmp_seq=1 ttl=64 time=2.50 ms
64 bytes from 10.0.1.5: icmp_seq=2 ttl=64 time=2.58 ms
^C
--- 10.0.1.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 2.503/2.540/2.578/0.037 ms
azureuser@myvm:~$ ping 10.0.1.4
PING 10.0.1.4 (10.0.1.4) 56(84) bytes of data.
64 bytes from 10.0.1.4: icmp_seq=1 ttl=64 time=2.21 ms
64 bytes from 10.0.1.4: icmp_seq=2 ttl=64 time=2.98 ms
^C
--- 10.0.1.4 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 2.208/2.593/2.978/0.385 ms
azureuser@myvm:~$ ping 10.0.1.6
PING 10.0.1.6 (10.0.1.6) 56(84) bytes of data.
64 bytes from 10.0.1.6: icmp_seq=1 ttl=64 time=2.36 ms
64 bytes from 10.0.1.6: icmp_seq=2 ttl=64 time=1.44 ms
64 bytes from 10.0.1.6: icmp_seq=3 ttl=64 time=1.23 ms
64 bytes from 10.0.1.6: icmp_seq=4 ttl=64 time=0.812 ms
64 bytes from 10.0.1.6: icmp_seq=5 ttl=64 time=1.31 ms
^C
--- 10.0.1.6 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4019ms
rtt min/avg/max/mdev = 0.812/1.431/2.361/0.510 ms
azureuser@myvm:~$
```

Ping réussie entre myVM et les 3 serveurs déployés

```
- hosts: webservices

user: azureuser
become: yes

tasks:

  - name: install nginx
    apt: pkg=nginx state=present

  - name: start nginx ever bootstrap
    service: name=nginx state=started enabled=yes
```

Modification du playbook.yml pour lui donner les droits sudo

```
[webservices]
10.0.1.5 ansible_ssh_private_key_file=/home/azureuser/fichier-0.pem
10.0.1.6 ansible_ssh_private_key_file=/home/azureuser/fichier-1.pem
10.0.1.4 ansible_ssh_private_key_file=/home/azureuser/fichier-2.pem
```

Modification de webservices.yml pour donner l'emplacement des clés

```

root@myvm:/home/azureuser/admin# ansible-playbook -i hosts playbook.yml

PLAY [webservices] *****

TASK [Gathering Facts] *****
ok: [10.0.1.5]
ok: [10.0.1.6]
ok: [10.0.1.4]

TASK [install nginx] *****
ok: [10.0.1.6]
ok: [10.0.1.5]
ok: [10.0.1.4]

TASK [start nginx ever bootup] *****
ok: [10.0.1.5]
ok: [10.0.1.6]
ok: [10.0.1.4]

PLAY RECAP *****
10.0.1.4      : ok=3  changed=0  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0
10.0.1.5      : ok=3  changed=0  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0
10.0.1.6      : ok=3  changed=0  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

```

NGINX est bien déployé sur nos 3 serveurs

4. Demandez à votre enseignant de créer un sous domaine flexcorp.fr qui accède à votre serveur « principal ». Cela permettra de lancer des tests depuis l'extérieur.
5. Via Ansible, installer sur vos 3 serveurs la super API d'intelligence artificielle de Dave le Dev accessible ici : <https://gitlab.esiea.fr/daveledev/super-ai-croquette>. Elle s'exécute via le script « run.sh » et est accessible sur le port 80 par défaut, vous pouvez le changer si Nginx utilise le même port.

```

- hosts: webservices

user: azureuser
become: yes

tasks:
  - name: install nginx
    apt: pkg=nginx state=present

  - name: start nginx ever bootup
    service: name=nginx state=started enabled=yes

  - name: Clone repo on 3VM's
    git:
      repo: "https://gitlab.esiea.fr/daveledev/super-ai-croquette.git"
      dest: "/src/project"
      update: no

```

Playbook mis à jour pour installer la super API sur nos 3 serveurs

```

root@myvm:/home/azureuser/admin# ansible-playbook -i hosts playbook.yml

PLAY [webservices] *****

TASK [Gathering Facts] *****
ok: [10.0.1.5]
ok: [10.0.1.6]
ok: [10.0.1.4]

TASK [install nginx] *****
ok: [10.0.1.6]
ok: [10.0.1.5]
ok: [10.0.1.4]

TASK [start nginx ever bootup] *****
ok: [10.0.1.5]
ok: [10.0.1.6]
ok: [10.0.1.4]

TASK [Clone repo on 3VM's] *****
changed: [10.0.1.6]
changed: [10.0.1.4]
changed: [10.0.1.5]

PLAY RECAP *****
10.0.1.4      : ok=4  changed=1  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0
10.0.1.5      : ok=4  changed=1  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0
10.0.1.6      : ok=4  changed=1  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

```

La super AI est bien déployée sur nos 3 serveurs

```

- hosts: webservices

  user: azureuser
  become: yes

  tasks:
    - name: install nginx
      apt: pkg=nginx state=present

    - name: start nginx ever bootup
      service: name=nginx state=started enabled=yes

    - name: Clone repo on 3VM's
      git:
        repo: "https://gitlab.esiea.fr/daveledev/super-ai-croquette.git"
        dest: "/src/project/"
        update: no

    - name: change port
      replace:
        path: "/src/project/run.sh"
        replace: "port=2534"
        regexp: "port=80"

```

```

azureuser@myvm:~/admin$ sudo ansible-playbook -i hosts playbook.yml

PLAY [webservices] *****
TASK [Gathering Facts] *****
ok: [10.0.1.5]
ok: [10.0.1.6]
ok: [10.0.1.4]

TASK [install nginx] *****
ok: [10.0.1.6]
ok: [10.0.1.5]
ok: [10.0.1.4]

TASK [start nginx ever bootup] *****
ok: [10.0.1.5]
ok: [10.0.1.6]
ok: [10.0.1.4]

TASK [Clone repo on 3VM's] *****
ok: [10.0.1.5]
ok: [10.0.1.6]
ok: [10.0.1.4]

TASK [change port] *****
ok: [10.0.1.5]
ok: [10.0.1.6]
ok: [10.0.1.4]

PLAY RECAP *****
10.0.1.4 : ok=5 changed=0 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
10.0.1.5 : ok=5 changed=0 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
10.0.1.6 : ok=5 changed=0 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

```

Changement de ports pour exécuter la super AI

```

PLAY [webservices] *****
TASK [Gathering Facts] *****
ok: [10.0.1.5]
ok: [10.0.1.6]
ok: [10.0.1.4]

TASK [install nginx] *****
ok: [10.0.1.5]
ok: [10.0.1.6]
ok: [10.0.1.4]

TASK [start nginx ever bootup] *****
ok: [10.0.1.5]
ok: [10.0.1.6]
ok: [10.0.1.4]

TASK [Install flask for AI] *****
ok: [10.0.1.5]
ok: [10.0.1.6]
ok: [10.0.1.4]

TASK [Install gunicorn for AI] *****
ok: [10.0.1.5]
ok: [10.0.1.6]
ok: [10.0.1.4]

TASK [Clone repo on 3VM's] *****
ok: [10.0.1.5]
ok: [10.0.1.6]
ok: [10.0.1.4]

TASK [change port] *****
ok: [10.0.1.6]
ok: [10.0.1.5]
ok: [10.0.1.4]

PLAY RECAP *****
10.0.1.4 : ok=7 changed=0 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
10.0.1.5 : ok=7 changed=0 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
10.0.1.6 : ok=7 changed=0 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

```

Installation de flask et de gunicorn déployé sur nos 3 serveurs

6. Créez un service systemd permettant de la lancer automatiquement cette IA. Vous pouvez le déployer avec Ansible.

Automatisation

1. Créez un script ayant pour objectif d'automatiser complètement la création d'un n^{ième} serveur web de A à Z grâce à Terraform et Ansible. Vous pouvez choisir le langage que vous préférez.

```
bessel@LAPTOP-72IDHQAM: ~  
from http.server import HTTPServer, BaseHTTPRequestHandler  
  
class helloHandler(BaseHTTPRequestHandler):  
    def do_GET(self):  
        self.send_response(200)  
        self.send_header('content-type', 'text/html')  
        self.end_headers()  
        self.wfile.write('Projet Automatisation !'.encode())  
  
def main():  
    PORT = 8000  
    server = HTTPServer(('',PORT), helloHandler)  
    print('Server running on port %s' % PORT)  
    server.serve_forever()  
  
if __name__ == '__main__':  
    main()
```

Script en python pour automatiser webserver
Source : vidéo YouTube (Conor Bailey)

```
# - name: run webserver.py  
#command: python2 /home/azureuser/admin/Projet/webserver.py  
  
# - name: read file content  
#command: "cat /home/azureuser/admin/Projet/webserver.py"  
#register: out
```

On a tenté d'exécuter un script python sur ansible de différentes façons mais sans réussite

2. Intégrez à votre script la possibilité de regarder s'il faut rajouter dynamiquement une nouvelle machine virtuelle ou en retirer une en fonction de la charge (nombre de connexions externes) et de modifier la configuration Terraform si nécessaire.