Lab 9 – Terraform Commands

1. Quelle commande peut être utilisée pour créer une représentation visuelle de nos ressources Terraform ?
   1. terraform graph
   2. terraform flow
   3. terraform map
   4. terraform view
2. Dans un répertoire lab9, créer un fichier main.tf avec le contenu suivant :

resource "local\_file" "key\_data" {

        filename       = "/tmp/.pki/private\_key.pem"

        content = tls\_private\_key.private\_key.private\_key\_pem

        file\_permission =  "0400"

}

resource "tls\_private\_key" "private\_key" {

  algorithm   = "RSA"

  bits  = 2048

  ecdsa\_curve = "P384"

}

resource "tls\_cert\_request" "csr" {

  private\_key\_pem = file("/tmp/.pki/private\_key.pem")

  depends\_on = [ local\_file.key\_data ]

  subject {

    common\_name  = "flexit.com"

    organization = "FlexIT Consulting Services"

  }

}

1. Utilisez la commande terraform validate pour valider et fixer l’erreur de configuration.

Remplacer bits par rsa\_bits

1. Exécutez maintenant terraform plan pour générer un plan de configuration. Si tout va bien créez les ressources avec la commande adéquate. Est-ce que la commande a réussi ?

*NOTE : La commande terraform apply a échoué malgré le bon fonctionnement de notre validation ! En effet, la commande validate effectue uniquement une vérification générale de la configuration. Elle valide le bloc de ressources et la syntaxe des arguments mais pas les valeurs attendues par les arguments pour une ressource spécifique !*

1. L'erreur dans la configuration se trouve à l'intérieur du bloc de ressources pour la ressource de type tls\_private\_key.

Il contient la configuration dont nous avions besoin pour générer la clé de type rsa.

Inspectez le bloc de ressources et fixez le problème.

Une fois terminé, exécutez le terraform plan, puis terraform applypour créer les ressources.

*Indice : L'algorithme utilisé par tls\_private\_key est RSA mais la configuration mentionne également l'argument ecdsa\_curve.*

1. Formatez maintenant le fichier main.tf dans un format canonique.
2. Dans ces labs Terraform, nous avons utilisé jusqu'à présent plusieurs providers. Mais que sont les providers ?
   1. Registry
   2. None
   3. Plugins
   4. Protocol
3. Laquelle est une sous-commande de Terraform Providers ?
   1. status
   2. list
   3. update
   4. mirror
4. Dans un dossier provider, créer les deux fichiers project.tf et provider.tf avec les contenus suivants :
   1. project.tf

resource "local\_file" "cloud" {

     filename = "/tmp/plugins"

     content = "multiple cloud providers."

}

resource "aws\_ebs\_volume" "myvolume" {

  availability\_zone = "us-east-2"

  size              = 20

}

* 1. provider.tf

terraform {

  required\_providers {

    aws = {

      source = "hashicorp/aws"

      version = "4.15.0"

    }

  }

}

provider "aws" {

  region                      = lookup(var.region, terraform.workspace)

  skip\_credentials\_validation = true

  skip\_requesting\_account\_id  = true

  s3\_use\_path\_style = true

  endpoints {

    ec2 = "http://aws:4566"

    dynamodb = "http://aws:4566"

    s3 = "http://aws:4566"

  }

}

1. Exécuter terraform init et vérifiez les plugins du provider qui ont été téléchargés à partir de la ligne de commande.
   1. local && aws
   2. kubernetes && random
   3. google && helm
   4. azurerm && dns