UTOPIOS
TRANSFORMATION DIGITALE INTELLIGENTE

Terraform avec OpenStack

Programme

1. Introduction
   1.1 Objectifs de la formation
   1.2 Présentation de Terraform
   1.3 Présentation d'OpenStack
2. Mise en place de l'environnement
   2.1 Installation de Terraform
   2.2 Installation du client OpenStack
   2.3 Configuration des accès API pour OpenStack
3. Concepts de base de Terraform
   3.1 Les providers
   3.2 Les ressources
   3.3 Les variables
   3.4 Les outputs
   3.5 Les modules

Programme

4. Provider OpenStack pour Terraform
   4.1 Configuration du provider OpenStack
   4.2 Exemples de ressources OpenStack

5. Gestion de l'état de Terraform
   5.1 Comprendre l'état de Terraform
   5.2 Utilisation de backends pour stocker l'état

6. Création d'une infrastructure avec Terraform et OpenStack
   6.1 Création d'un réseau et d'un sous-réseau
   6.2 Création d'une instance
   6.3 Création d'un groupe de sécurité
   6.4 Création d'un volume
   6.5 Création d'un routeur et connexion au réseau externe
   6.6 Utilisation des modules pour organiser le code

Programme

7. Les meilleures pratiques pour Terraform et OpenStack
   7.1 Planification des changements
   7.2 Utilisation de la documentation
   7.3 Collaboration avec d'autres membres de l'équipe

Objectifs de la formation

- Comprendre les concepts clés de Terraform et OpenStack
- Apprendre à créer et gérer une infrastructure cloud avec Terraform et OpenStack
- Maîtriser les meilleures pratiques pour utiliser Terraform avec OpenStack

Présentation de Terraform

- Définition de Terraform :
  - Terraform est un outil open-source développé par HashiCorp qui permet de définir, provisionner et gérer des ressources d'infrastructure en tant que code (IaC) dans divers fournisseurs de cloud, tels qu'OpenStack.
  - Terraform utilise son propre langage de configuration déclaratif appelé HCL (HashiCorp Configuration Language) pour décrire les ressources souhaitées, puis génère un plan d'exécution pour atteindre l'état souhaité.
- Avantages de l'Infrastructure as Code (IaC) :
  - Versionnage : Le code de l'infrastructure peut être versionné et contrôlé dans des systèmes de gestion de version (comme Git), permettant un suivi clair des modifications.
  - Reproductibilité : Les infrastructures peuvent être déployées de manière cohérente et reproductible, réduisant les erreurs humaines et les différences entre les environnements.
  - Automatisation : Le processus de déploiement de l'infrastructure peut être automatisé, réduisant les coûts et les efforts manuels.

Présentation d'OpenStack

- OpenStack est un logiciel open-source qui permet de créer et gérer des infrastructures de cloud computing.
- Composants principaux d'OpenStack :
  - Nova (compute),
  - Swift (stockage objet),
  - Cinder (stockage bloc),
  - Neutron (réseau)
  - Keystone (identité).

Installation de Terraform

- Télécharger Terraform à partir du site officiel : https://www.terraform.io/downloads.html
- Installer Terraform en suivant les instructions pour votre système d'exploitation

Installation du client OpenStack

- Installer le client OpenStack en utilisant la commande :

```
pip install python-openstackclient
```

- Vérifier l'installation en exécutant la commande :

```
openstack --version
```

Configuration des accès API pour OpenStack

- Récupérer le fichier de configuration OpenStack (souvent nommé openrc.sh) auprès de votre administrateur OpenStack ou via le tableau de bord
- Charger le fichier de configuration en exécutant :

```
source openrc.sh
```

Exercice 1 - Installation et configuration

- Installer Terraform et le client OpenStack sur votre machine
- Configurer l'accès API pour OpenStack en utilisant un fichier openrc.sh

Les providers

- Les providers sont des plugins qui permettent à Terraform de communiquer avec des services externes
- Le provider OpenStack permet à Terraform de gérer les ressources OpenStack

Les ressources

- Les ressources représentent des éléments d'infrastructure, tels que des instances, des réseaux ou des volumes
- Exemple de ressource OpenStack :

```
resource "openstack_compute_instance_v2" "example"
```

Les variables

- Les variables permettent de paramétrer les configurations Terraform et de les rendre réutilisables
- Exemple :

```
variable "image_id" { default = "abc123" }
```

Les outputs

- Les outputs permettent d'afficher des informations sur les ressources créées par Terraform
- Exemple :

```
output "instance_ip" { value = openstack_compute_instance_v2.example.access_ip_v4 }
```

Les modules

- Les modules permettent de regrouper et réutiliser des configurations Terraform
- Ils facilitent la gestion et l'organisation du code

Configuration du provider OpenStack

- Ajouter le provider OpenStack à votre configuration Terraform

Exemple :

```
provider "openstack" {
  user_name   = "myuser"
  tenant_name = "mytenant"
  password    = "mypassword"
  auth_url    = "https://myopenstack.example.com:5000/v3"
}
```

Planification avec Terraform

- Exécuter `terraform init` pour initialiser le projet
- Exécuter `terraform plan` pour générer un plan d'exécution
- Analyser le plan d'exécution pour vérifier les modifications apportées à l'infrastructure

Déploiement avec Terraform

- Exécuter `terraform apply` pour appliquer le plan d'exécution et déployer l'infrastructure
- Vérifier que les ressources ont été créées dans OpenStack

Mise à jour de l'infrastructure

- Modifier la configuration Terraform pour mettre à jour les ressources
- Exécuter `terraform plan` et `terraform apply` pour appliquer les modifications

Destruction de l'infrastructure

- Exécuter `terraform destroy` pour supprimer toutes les ressources créées par Terraform
- Vérifier que les ressources ont été supprimées dans OpenStack

Création d'un réseau et d'un sous-réseau

- Créer un réseau avec openstack_networking_network_v2
- Créer un sous-réseau avec openstack_networking_subnet_v2
- Exemple de configuration :

```
resource "openstack_networking_network_v2" "example_network" {
  name = "example-network"
}

resource "openstack_networking_subnet_v2" "example_subnet" {
  name = "example-subnet"
  network_id = openstack_networking_network_v2.example_network.id
  cidr = "192.168.0.0/24"
}
```

Création d'une instance

- Créer une instance avec openstack_compute_instance_v2
- Spécifier l'image, le flavor, le réseau et le groupe de sécurité
- Exemple de configuration :

```
resource "openstack_compute_instance_v2" "example" {
  name = "example-instance"
  image_id = "image_id"
  flavor_id = "flavor_id"
  key_pair = "key_pair_name"
  security_groups = ["default", "example_security_group"]

  network {
    uuid = openstack_networking_network_v2.example_network.id
  }
}
```

Création d'un groupe de sécurité

- Créer un groupe de sécurité avec openstack_compute_secgroup_v2
- Ajouter des règles avec openstack_compute_secgroup_rule_v2
  Exemple de configuration :

```
resource "openstack_compute_secgroup_v2" "example" {
  name = "example_security_group"
  description = "Example security group"
}

resource "openstack_compute_secgroup_rule_v2" "example_rule" {
  direction = "ingress"
  ethertype = "IPv4"
  protocol = "tcp"
  port_range_min = 22
  port_range_max = 22
  remote_ip_prefix = "0.0.0.0/0"
  security_group_id = openstack_compute_secgroup_v2.example.id
}
```

Exercice 2 - Création d'une instance simple

- Créer un fichier de configuration Terraform pour déployer une instance OpenStack
- Utiliser le provider OpenStack et la ressource openstack_compute_instance_v2
- Appliquer la configuration

Exercice 3 - Configuration d'un réseau et d'un sous-réseau

- Modifier la configuration de l'exercice 2 Terraform pour inclure un réseau et un sous-réseau personnalisés
- Utiliser les ressources openstack_networking_network_v2 et openstack_networking_subnet_v2
- Appliquer les modifications

Exercice 4 - Ajout d'un groupe de sécurité

- Ajouter un groupe de sécurité à la configuration Terraform
- Utiliser les ressources openstack_compute_secgroup_v2 et openstack_compute_secgroup_rule_v2
- Appliquer les modifications

Création d'un volume

- Créer un volume avec openstack_blockstorage_volume_v2
- Attacher le volume à l'instance avec openstack_compute_volume_attach_v2
- Exemple de configuration :

```
resource "openstack_blockstorage_volume_v2" "example" {
  name = "example-volume"
  size = 10
}

resource "openstack_compute_volume_attach_v2" "example_attach" {
  instance_id = openstack_compute_instance_v2.example.id
  volume_id = openstack_blockstorage_volume_v2.example.id
}
```

Exercice 6 - Création de volumes

- Créez un fichier pour définir les variables nécessaires.
- Créez un fichier pour définir les ressources suivantes :
  - openstack_blockstorage_volume_v3 (volume)
  - openstack_compute_instance_v2 (instance)
  - openstack_compute_volume_attach_v2 (attachement du volume)
- Utilisez les variables pour paramétrer les ressources.
- Configurez la taille du volume, le type de volume et les autres paramètres nécessaires.
- Déployez l'infrastructure.
- Vérifiez que le volume a été créé et attaché à l'instance dans OpenStack.
- Supprimez l'infrastructure.

Création d'un routeur et connexion au réseau externe

- Créer un routeur avec openstack_networking_router_v2
- Connecter le routeur au réseau externe avec openstack_networking_router_interface_v2
- Exemple de configuration :

```
resource "openstack_networking_router_v2" "example" {
  name = "example-router"
  external_gateway = "external_network_id"
}

resource "openstack_networking_router_interface_v2" "example_interface" {
  router_id = openstack_networking_router_v2

}
```

Création d'un équilibreur de charge

- Créer un équilibreur de charge avec openstack_lb_loadbalancer_v2
- Ajouter des auditeurs avec openstack_lb_listener_v2
- Ajouter des pools avec openstack_lb_pool_v2
- Ajouter des membres au pool avec openstack_lb_member_v2
- Exemple de configuration :

Création d'un équilibreur de charge

```
resource "openstack_lb_loadbalancer_v2" "example" {
name = "example-loadbalancer"
vip_subnet_id = openstack_networking_subnet_v2.example_subnet.id
}
    resource "openstack_lb_listener_v2" "example" {
      name = "example-listener"
      protocol = "HTTP"
      protocol_port = 80
      loadbalancer_id = openstack_lb_loadbalancer_v2.example.id
    }

    resource "openstack_lb_pool_v2" "example" {
      name = "example-pool"
      protocol = "HTTP"
      lb_method = "ROUND_ROBIN"
      listener_id = openstack_lb_listener_v2.example.id
    }

    resource "openstack_lb_member_v2" "example" {
      address = "192.168.0.10"
      protocol_port = 80
      subnet_id = openstack_networking_subnet_v2.example_subnet.id
      pool_id = openstack_lb_pool_v2.example.id
    }
```

Exercice 7 - Création d'un équilibreur de charge

- Créez un fichier pour définir les variables nécessaires.
- Créez un fichier pour définir les ressources suivantes :
  - openstack_networking_network_v2 (réseau)
  - openstack_networking_subnet_v2 (sous-réseau)
  - openstack_compute_instance_v2 (2 instances)
  - openstack_lb_loadbalancer_v2 (équilibreur de charge)
  - openstack_lb_listener_v2 (auditeur)
  - openstack_lb_pool_v2 (pool)
  - openstack_lb_member_v2 (membres du pool)
- Utilisez les variables pour paramétrer les ressources.
- Déployez l'infrastructure.
- Supprimez l'infrastructure.

Bonnes pratiques - Utilisation de modules

- Utiliser des modules pour organiser et réutiliser des configurations Terraform
- Créer des modules personnalisés ou utiliser des modules existants dans le registre Terraform

Exercice 8 - Création d'un module

- Créer un module Terraform pour encapsuler la configuration du réseau, du sous-réseau et du groupe de sécurité
- Utiliser le module dans la configuration principale

Bonnes pratiques - Gestion des états

- Comprendre l'importance de la gestion des états Terraform
- Utiliser des backends distants pour stocker et partager les états entre les membres de l'équipe

Bonnes pratiques - Sécurité et confidentialité

- Ne pas inclure les informations sensibles, telles que les mots de passe, directement dans les fichiers de configuration
- Utiliser des variables d'environnement ou des fichiers d'entrée pour gérer les informations sensible

Collaboration et versionnage

- Utiliser des systèmes de gestion de version, tels que Git, pour suivre les modifications apportées aux fichiers de configuration Terraform
- Collaborer avec les membres de l'équipe en utilisant des outils comme GitHub ou GitLab pour les demandes de fusion (merge requests) et les revues de code

Test et validation

- Utiliser des outils de validation et de vérification de la syntaxe pour les fichiers de configuration Terraform, tels que `terraform validate` et `terraform fmt`
- Mettre en place des tests d'intégration pour valider que les ressources créées fonctionnent comme prévu

Test et validation

- Comprendre les erreurs courantes et comment les résoudre
- Utiliser des commandes comme `terraform refresh` et `terraform import` pour résoudre les problèmes liés à l'état de l'infrastructure