Azure Kubernetes Services

Documentation Terraform

Objectif

Le but de ce fichier est de créer les ressources suivantes sur Azure :

- Un groupe de ressources
- Un ACR (Azure Container Registry)
- Un AKS (Azure Kubernetes Service)

Structure du Projet

Le projet sera constitué d'un fichier principal appelé main.tf, qui fera appel à d'autres scripts organisés sous forme de modules. Ces modules seront situés dans un dossier nommé module.

Pré-requis

Les ressources seront déployées sur un tenant Azure. Vous devez donc fournir votre subscription_id et tenant_id. Dans notre exemple, ces informations se trouvent au début de notre fichier main.tf.

```
provider "azurerm" {
  features {}
  subscription_id =      # Remplacez par votre ID d'abonnement Azure
  tenant_id       =     # Remplacez par votre ID de locataire Azure
}
```

Connexion à Azure

Avant de commencer à taper des commandes pour envoyer votre configuration sur Azure, vous devez connecter votre terminal à votre tenant Azure. Pour cela, utilisez la commande az login.

Confirmation de connexion :

Après avoir exécuté la commande, une fenêtre de navigateur s'ouvrira pour vous permettre de vous authentifier sur le portail Azure. Une fois l'authentification réussie, le terminal affichera une confirmation avec votre tenantId. Cette confirmation vous assure que vous êtes bien connecté à votre environnement Azure.

Vous pourrez commencer à taper 'terraform init'

Description

terraform init initialise un répertoire contenant des fichiers de configuration Terraform. Cette commande prépare le répertoire en téléchargeant les plugins nécessaires, configurant le backend, et effectuant diverses autres tâches pour s'assurer que Terraform peut fonctionner correctement dans ce répertoire.

Étapes de l'Initialisation

Téléchargement des Plugins :

Terraform télécharge les fournisseurs (providers) spécifiés dans vos fichiers de configuration. Dans votre cas, il s'agirait du fournisseur azurerm.
Ces plugins sont stockés dans un répertoire nommé. terraform au sein de votre répertoire de travail.

Configuration du Backend

Si vous avez spécifié un backend dans votre configuration (par exemple, un backend pour stocker l'état de votre infrastructure de manière sécurisée et partagée), Terraform le configure.

Par défaut, Terraform utilise un backend local, ce qui signifie que l'état est stocké dans un fichier local.

Vérification des Fichiers de Configuration :

Terraform vérifie que tous les fichiers de configuration sont bien formés et qu'il n'y a pas d'erreurs syntaxiques.

Vous poursuivrez avec la commande 'terraform plan'.

Description

La commande terraform plan est utilisée pour créer un plan d'exécution de Terraform. Ce plan montre ce que Terraform fera lorsque vous exécuterez terraform apply. En d'autres termes, terraform plan simule les modifications qui seront apportées à votre infrastructure et affiche un aperçu détaillé de ces changements sans réellement les appliquer.

Objectifs

Vérification de la Configuration :

Terraform analyse vos fichiers de configuration (.tf) et vérifie s'ils sont corrects et bien formés.

Préparation du Plan :

Terraform compare l'état actuel de votre infrastructure (stocké dans le fichier d'état ou dans le backend) avec la configuration souhaitée décrite dans vos fichiers .tf.

Il génère une liste des actions nécessaires pour aligner l'état actuel avec l'état désiré.

Aperçu des Changements :

Le plan montre une liste des ressources qui seront créées, modifiées ou détruites.

Évitement des Erreurs :

En examinant le plan avant de l'appliquer, vous pouvez repérer et corriger d'éventuelles erreurs dans votre configuration avant qu'elles n'affectent votre infrastructure réelle.

Pour finir, vous taperez dans votre terminal 'terraform apply'

Description

La commande terraform apply est utilisée pour appliquer les modifications décrites dans le plan généré par terraform plan. Une fois que vous êtes satisfait du plan et que vous souhaitez réellement apporter les modifications à votre infrastructure, vous exécutez terraform apply.

Objectifs

Application des Modifications :

Terraform prend les actions décrites dans le plan généré par terraform plan et les applique à votre infrastructure.

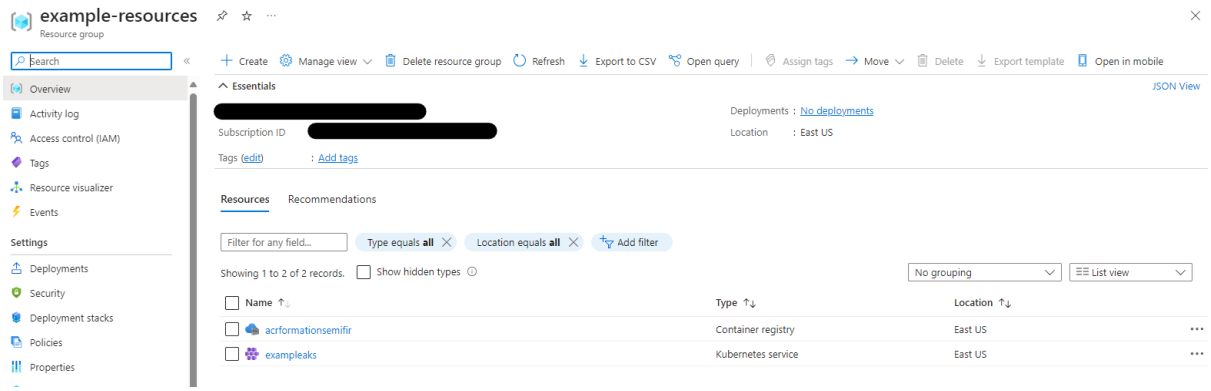
Création, Modification ou Suppression de Ressources :

En fonction du plan, Terraform crée de nouvelles ressources, modifie les ressources existantes ou supprime les ressources obsolètes.

Mise à Jour de l'État :

Terraform met à jour son état interne pour refléter les modifications apportées à votre infrastructure.

Une fois que vous avez tapé ces trois commandes, vous trouverez sur votre compte Azure un nouveau groupe de ressources avec à l'intérieur un ACR et un AKS déployés.



Ensuite, pour lancer un pod grâce à vos ressources nouvellement déployées, il va falloir exécuter quelques commandes et s'assurer que vous avez bien Kubernetes installé sur votre machine en local.

Vous allez vous connecter à votre Azure Container Registry pour lui envoyer une image nginx. Cela se fait avec la commande :

az acr login --name acrformationsemifir

Ensuite, pour obtenir l'image en local, vous allez devoir taper la commande :

docker pull nginx

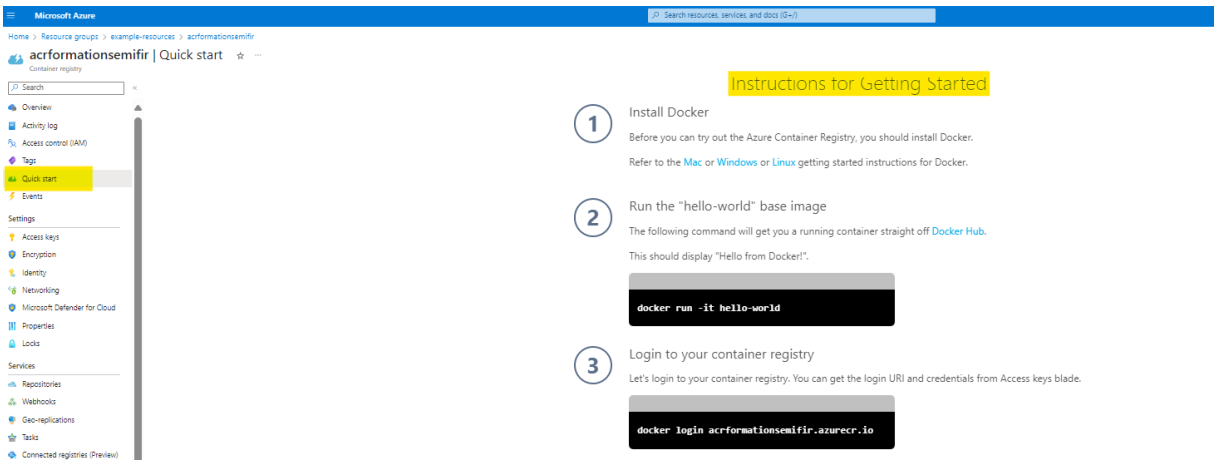
Pour pouvoir envoyer une image sur un ACR, il vous faut faire un tag avec la commande :

docker tag nginx:latest acrformationsemifir/nginx:latest

Pour avoir un tutoriel tout fait, vous pouvez vous rendre dans votre ACR sur Azure et accéder à la catégorie Quick Start où vous trouverez un tutoriel élaboré par Azure.

Pour finir vous allez devoir taper la commande :

docker push nom_acr.io/nginx:latest



Pour déployer un node, il faut mettre en place un script YAML.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
spec:
  containers:
  - name: nginx-container
    image:  #remplacer par votre image
    ports:
    - containerPort: 80
```

Enfin, pour déployer un nœud sur AKS avec notre ACR, vous taperez la commande :

kubectl apply -f nom-du-fichier.yaml

Dans votre compte Azure, vous devriez voir apparaître un nouveau groupe de ressources avec votre nœud construit à l'intérieur.