Création d'un utilisateur IAM pour CI/CD

Vous pouvez créer un utilisateur AWS IAM qui permettra à vos pipelines GitHub Actions (ou autre outil CI/CD) de déployer automatiquement votre infrastructure.

1. Dans la barre de recherche, tapez **IAM** pour accéder au service IAM
2. Dans le menu latéral gauche, cliquez sur **Personnes**
3. Ensuite **Créer un utilisateur** en haut à droite
4. Détails de l'utilisateur :
    - Nom d'utilisateur : `git-user-gXmgXX` pour personnaliser avec votre id groupe (ex. `git-user-g0mg00` pour G0-MG00)
    - Laissez la case **"Fournir aux utilisateurs l'accès à la console de gestion AWS" DÉCOCHÉE** (cet utilisateur n'a pas besoin de se connecter à la console)
    - Cliquez sur **Suivant**
5. Attacher les permissions (politiques)
    - Très importante, vous allez donner les permissions à votre utilisateur.
    - Sélectionnez **Attacher directement des politiques**
    - Dans la barre de recherche, cherchez et **cochez** les politiques suivantes (une par une) :
        - `AmazonEC2ContainerRegistryFullAccess` - Pour gérer les images Docker dans ECR
        - `AmazonECS_FullAccess` - Pour gérer les clusters ECS
        - `AmazonRDSFullAccess` - Pour gérer les bases de données RDS
        - `AmazonS3FullAccess` - Pour gérer les buckets S3
        - `AmazonVPCFullAccess` - Pour gérer les réseaux VPC
        - `AWSAppRunnerFullAccess` - Pour gérer les services App Runner
        - `CloudWatchLogsFullAccess` - Pour gérer les logs CloudWatch
        - `ElasticLoadBalancingFullAccess` - Pour gérer les load balancers
        - `IAMFullAccess` - Pour créer des rôles IAM (nécessaire pour Terraform)

    - `SecretsManagerReadWrite` - Pour gérer les secrets de vos BD par exemple

6. Cliquez sur **Suivant**, puis **Créer un utilisateur**

Création des clés d'accès programmatique

1. Toujours dans le service IAM, ensuite dans Personnes, **cliquez sur le nom de votre utilisateur**
2. Cliquez sur l'onglet **Informations d'identification de sécurité**
3. Descendez jusqu'à la section **Clés d'accès** et cliquez sur **Créer une clé d'accès**
4. **Sélectionnez le cas d'utilisation** : choisissez **"Application exécutée en dehors d'AWS"**
5. **Balise de description** (facultatif)
6. Cliquez sur **Créer une clé d'accès**
7. **ATTENTION :** cliquez sur "Télécharger le fichier .csv", puis cliquez sur **Terminé**

Utiliser les clés dans GitHub

1. Allez sur votre repository project GitHub
2. Cliquez sur **Settings** (Paramètres)
3. Dans le menu latéral, cliquez sur **Secrets and variables** > **Actions**
4. Cliquez sur **New repository secret**
5. Ajoutez ces 3 secrets :
    - Secret 1 : Name : `AWS_ACCESS_KEY_ID`, Secret : Collez votre Access Key ID (celle du fichier CSV)
    - Secret 2 : Name : `AWS_SECRET_ACCESS_KEY`, Secret : Collez votre Secret Access Key (celle du fichier CSV)
    - Secret 3 : Name : `AWS_REGION`, Secret : `eu-west-3`

Configuration d'un Backend S3 pour Terraform

Quand Terraform crée votre infrastructure, il garde une "mémoire" (état actuel) de ce qui a été créé dans un fichier appelé `terraform.tfstate`. Comme souligne en cours,

par défaut, ce fichier est stocké localement, mais ça pose problème quand vous utilisez GitHub Actions car le fichier est perdu après chaque exécution du workflow.

1. Créer le bucket depuis la console AWS
2. Laisser GitHub Actions créer le bucket automatiquement

```
- name: Créer le bucket S3 pour le state Terraform
  run: |
    aws s3api create-bucket \
      --bucket infrastats-gXmgXX \
      --region eu-west-3 \
      --create-bucket-configuration LocationConstraint=eu-west-3 \
      2>/dev/null || echo "Bucket déjà existant"

    aws s3api put-bucket-versioning \
      --bucket infrastats-gXmgXX \
      --versioning-configuration Status=Enabled
```

**N'oubliez pas de remplacer `gXmgXX` par votre référence **

3. Configurer le backend dans main.tf
    - Dans votre fichier `main.tf`
    - Dans le bloc `terraform {}`, ajoutez la configuration du backend S3.

```
terraform {
required_version = ">= 1.5.0"

required_providers {
   aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
   }
}

# Configuration du backend S3 pour stocker le state
backend "s3" {
   bucket  = "infrastats-gXmgXX"
   key     = "gXmgXX.tfstate"
   region  = "eu-west-3"
   encrypt = true
}
}
```

- `bucket` : Le nom du bucket S3 où stocker le fichier state
- `key` : Le nom du fichier state dans le bucket
- `encrypt` : Active le chiffrement du fichier state pour plus de sécurité

Vous devrez supprimer manullement depuis la console AWS les ressources initialement crée par votre Terraform avant l'ajout du backend S3, pour éviter des conflits d'état.