TD1 – Conception d’une stratégie IAM sécurisée

Objectifs

- Concevoir une stratégie IAM cohérente
- Appliquer le principe du moindre privilège
- Comprendre les différences entre users, groupes et rôles
- Industrialiser IAM avec Terraform

Contexte

Vous intervenez comme ingénieur Cloud dans une startup.

L’infrastructure AWS existe déjà, mais aucun cadre de sécurité IAM n’a été défini.

Chaque développeur dispose actuellement de droits administrateur.

Un audit de sécurité impose :

- une refonte complète des accès
- une gestion par rôles
- une limitation stricte des permissions

Objectif technique

Mettre en place une architecture IAM sécurisée pour :

- les administrateurs
- les développeurs
- les analystes

Le tout doit être :

- reproductible
- versionné
- automatisé avec Terraform

Contraintes

Vous devez respecter les règles suivantes :

- Aucun utilisateur ne doit avoir de droits administrateur globaux
- Le principe du moindre privilège est obligatoire
- Les permissions doivent être mutualisées (pas de duplication inutile)
- Les accès doivent être compréhensibles et maintenables
- Toute la configuration doit être réalisée avec Terraform

Infrastructure existante

Une première tentative de mise en place IAM a été réalisée.

Fichier : `iam/main.tf`

```
provider "aws" {
  region = "eu-west-1"
}

resource "aws_iam_user" "dev1" {
  name = "dev1"
}

resource "aws_iam_user" "dev2" {
  name = "dev2"
}

resource "aws_iam_user_policy" "dev_policy" {
  name = "dev-policy"
  user = aws_iam_user.dev1.name

  policy = jsonencode({
    Version = "2012-10-17",
    Statement = [
      {
        Effect = "Allow",
        Action = "*",
        Resource = "*"
      }
    ]
  })
}

resource "aws_iam_user_policy_attachment" "dev2_admin" {
  user       = aws_iam_user.dev2.name
  policy_arn = "arn:aws:iam::aws:policy/AdministratorAccess"
}
```

Ressources

Documentation officielle :

- AWS IAM : concepts (users, groups, roles)
- AWS IAM Policy JSON
- Bonnes pratiques AWS IAM (least privilege)

Commandes utiles :

Fichier : `commandes/terraform.txt`

```
terraform init
```

```
terraform plan
terraform apply
terraform destroy
```

Travail à réaliser

Vous devez produire une configuration Terraform permettant de :

1. Partir de l'existant et le corriger
2. Conserver les utilisateurs actuels
3. Reconcevoir entièrement la gestion des permissions

1. Créer les entités suivantes :
   - 1 groupe Admin
   - 1 groupe Dev
   - 1 groupe Analyst

1. Définir des policies adaptées :
   - Admin : gestion complète de l'infrastructure
   - Dev : gestion EC2 uniquement
   - Analyst : lecture S3 uniquement

1. Associer correctement :
   - users → groupes
   - groupes → policies

1. Mettre en place au moins un rôle IAM pour un service AWS

Livrables attendus :

- code Terraform fonctionnel
- structure claire des fichiers
- justification des choix

Point d'attention (volontaire)

Une mauvaise pratique courante consiste à utiliser :

- Action: "*"
- Resource: "*"

Cette pratique est interdite dans ce TD.

Pourquoi cette pratique apparaît-elle souvent dans des projets réels ?

Dans quels cas peut-elle sembler "pratique" ?

Phase d'analyse

Quels sont les problèmes de sécurité présents dans l'infrastructure fournie ?

Lesquels sont critiques ? Lesquels sont acceptables temporairement ?

Phase de correction

Vous devez revoir votre architecture si :

- un utilisateur a trop de droits
- une policy est trop large
- des permissions sont dupliquées

Extension

Ajouter :

- une séparation environnement DEV / PROD
- une restriction par ressource (ex : un seul bucket S3)

Challenge final

Un développeur doit pouvoir :

- lancer une instance EC2
- mais ne pas pouvoir supprimer une instance existante

Implémenter cette contrainte.

Pourquoi cette règle est-elle difficile à implémenter proprement en IAM ?

Évolution du besoin

L’entreprise prévoit :

- l’arrivée de 10 nouveaux développeurs
- la création d’une équipe data
- un environnement supplémentaire (staging)

Votre implémentation actuelle est-elle :

- facilement extensible ?
- duplicable ?
- maintenable ?

Quels changements sont nécessaires pour passer à l’échelle ?

Proposer au moins une amélioration concrète dans le code Terraform.

Cette amélioration doit illustrer votre réflexion.

Restitution

Vous devez être capable d’expliquer :

- votre architecture IAM
- vos choix de policies
- les compromis réalisés