Lab 6 — Multi-Environments (Teams) and Multi-Services in Kubernetes

Sommaire:

Introduction

L’objectif de ce Lab 6 est de mettre en place 3 environnements séparés (dev / stage / prod) avec des comptes AWS différents, puis d’utiliser OpenTofu (workspaces) et Kubernetes pour déployer des services de façon propre et reproductible.

Part 1: Setting Up Multiple AWS Accounts

1.1. Why Multiple Accounts?

On utilise plusieurs comptes AWS pour avoir un environnement par compte. Comme ça, ce qu’on teste en dev ne peut pas casser la prod, chaque environnement a ses propres ressources et c’est plus clair à gérer (droits, budgets, nettoyage).

1.2. Creating Child Accounts

Objectif : créer 3 “child accounts” (dev/stage/prod) via AWS Organizations, depuis le compte principal, en utilisant OpenTofu.

Actions réalisées :
Modification du fichier main.tf dans : td6/scripts/tofu/live/child-accounts/main.tf

Configuration AWS :
Région : us-east-2

Module OpenTofu
Utilisation du module en local : source = "../../modules/aws-organizations"

Création des emails des comptes :

- devops.development@outlook.fr
- devops.staging@outlook.fr
- devops.production@outlook.fr

Déploiement :
Commandes exécutées :
tofu init
tofu apply

Résultat obtenue

```
Apply complete! Resources: 4 added, 0 changed, 0 destroyed.

Outputs:

dev_account_id = "693001789530"
dev_role_arn = "arn:aws:iam::693001789530:role/OrganizationAccountAccessRole"
prod_account_id = "810134835322"
prod_role_arn = "arn:aws:iam::810134835322:role/OrganizationAccountAccessRole"
stage_account_id = "547520639874"
stage_role_arn = "arn:aws:iam::547520639874:role/OrganizationAccountAccessRole"
```

Le déploiement s'est terminé correctement ("Apply complete") et a généré les IDs et les rôles pour accéder aux comptes.
Ces role_arn servent pour la suite : créer des profils AWS et assumer le rôle pour gérer chaque compte.

J'ai choisi de créer de nouveaux emails auxquels j'ai accès, au cas où AWS envoie des validations ou alertes.

1.3 Accessing Child Accounts

Objectif :
Configurer des profils AWS pour accéder aux comptes dev, stage et prod avec le rôle OrganizationAccountAccessRole.

Problème rencontré :
Au début, la commande aws sts get-caller-identity ne fonctionnait pas avec les profils dev-admin, stage-admin et prod-admin.
La cause venait de credential_source = Environment dans le fichier ~/.aws/config : ce mode attend des credentials AWS dans les variables d'environnement, mais elles n'étaient pas définies sur mon PC.

Correction :
J'ai remplacé credential_source = Environment par source_profile = default dans ~\.aws\config, puis j'ai vérifié que le profil default contenait bien les credentials AWS dans ~\.aws\credentials.

```
PS C:\Users\antoi> $env:AWS_PROFILE="dev-admin"
PS C:\Users\antoi> aws sts get-caller-identity
{
    "UserId": "AROA2CWRAEBNOHEHUJ53W:botocore-session-1771579645",
    "Account": "693001789530",
    "Arn": "arn:aws:sts::693001789530:assumed-role/OrganizationAccountAccessRole/botocore-session-1771579645"
}

PS C:\Users\antoi>
PS C:\Users\antoi> $env:AWS_PROFILE="stage-admin"
PS C:\Users\antoi> aws sts get-caller-identity
{
    "UserId": "AROAX66WF6OBOV7CGXQMZ:botocore-session-1771579655",
    "Account": "547520639874",
    "Arn": "arn:aws:sts::547520639874:assumed-role/OrganizationAccountAccessRole/botocore-session-1771579655"
}

PS C:\Users\antoi> $env:AWS_PROFILE="prod-admin"
PS C:\Users\antoi> aws sts get-caller-identity
{
    "UserId": "AROA3ZH6MSB5D6XAXZAV7:botocore-session-1771579665",
    "Account": "810134835322",
    "Arn": "arn:aws:sts::810134835322:assumed-role/OrganizationAccountAccessRole/botocore-session-1771579665"
```

Les 3 profils AWS sont prêts. Je peux maintenant déployer la suite du lab sur chaque compte (dev / stage / prod)..

Part 2: Managing Deployments with OpenTofu Workspaces

2.1 Understanding Workspaces

Le principe des workspaces OpenTofu est d'utiliser le même code d'infrastructure pour plusieurs environnements, sans dupliquer les fichiers.
Chaque workspace a son propre state, donc on peut gérer séparément :

- development
- staging
- production

Dans ce lab, ça permet de déployer la même Lambda sur 3 environnements différents, tout en gardant une seule base de code.

2.2 Copying the Lambda Sample App

Dans mon cas, cette étape était déjà prête car le dossier lambda-sample était fourni dans Labs/LAB6/scripts/tofu/live/.
Je n'ai donc pas eu besoin de recopier le projet depuis un autre lab.

2.3 Preparing the Lambda Sample App

J’ai vérifié le contenu du projet lambda-sample avant de lancer le déploiement.

Vérifications faites :
Le code Lambda (src/index.js) retourne bien un message avec le nom de l’environnement (process.env.ENV_NAME).
Le fichier main.tf utilise terraform.workspace pour injecter le nom du workspace dans la variable d’environnement ENV_NAME.

Problème rencontré :
Au premier tofu init, OpenTofu a échoué avec une erreur sur le module api-gateway :

- subdir "ch3/tofu/modules/api-gateway" not found
- puis ..\..\modules\api-gateway: The system cannot find the file specified

Correction :

- J’ai modifié main.tf pour utiliser le module local :
    - source = "../../modules/api-gateway"
- J’ai copié le dossier api-gateway dans Labs/LAB6/scripts/tofu/modules/
- J’ai relancé tofu init

Résultat

tofu init a ensuite fonctionné correctement.

```
OpenTofu has been successfully initialized!

You may now begin working with OpenTofu. Try running "tofu plan" to see
any changes that are required for your infrastructure. All OpenTofu commands
should now work.

If you ever set or change modules or backend configuration for OpenTofu,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

2.4 Deploying to Multiple Environments

Après l’initialisation, j’ai créé les workspaces OpenTofu demandés par la consigne :

- development
- staging
- production

Commande utilisée :

- tofu workspace new development
- tofu workspace new staging
- tofu workspace new production

J’ai ensuite vérifié la liste des workspaces avec :

tofu workspace list

Résultat observé :

- les 3 workspaces sont bien présents

```
PS C:\Users\antoi\Desktop\DevOps-ESIEE\Labs\LAB6\scripts\tofu\live\lambda-sample> tofu workspace new development
Created and switched to workspace "development"!

You're now on a new, empty workspace. Workspaces isolate their state,
so if you run "tofu plan" OpenTofu will not see any existing state
for this configuration.
PS C:\Users\antoi\Desktop\DevOps-ESIEE\Labs\LAB6\scripts\tofu\live\lambda-sample> tofu workspace new staging
Created and switched to workspace "staging"!

You're now on a new, empty workspace. Workspaces isolate their state,
so if you run "tofu plan" OpenTofu will not see any existing state
for this configuration.
PS C:\Users\antoi\Desktop\DevOps-ESIEE\Labs\LAB6\scripts\tofu\live\lambda-sample> tofu workspace new production
Created and switched to workspace "production"!

You're now on a new, empty workspace. Workspaces isolate their state,
so if you run "tofu plan" OpenTofu will not see any existing state
for this configuration.
```

- le workspace actif est production (car c’est le dernier créé)

```
PS C:\Users\antoi\Desktop\DevOps-ESIEE\Labs\LAB6\scripts\tofu\live\lambda-sample> tofu workspace list
  default
  development
* production
  staging
```

Déploiement en development

J’ai sélectionné le workspace development, puis j’ai utilisé le profil AWS dev-admin pour déployer l’infrastructure avec OpenTofu.

Commandes exécutées :

- tofu workspace select development
- $env:AWS_PROFILE="dev-admin"
- tofu apply

```
Apply complete! Resources: 8 added, 0 changed, 0 destroyed.

Outputs:

api_endpoint = "https://k1mgiaol2l.execute-api.us-east-2.amazonaws.com"
```

Résultat :

```
StatusCode        : 200
StatusDescription : OK
Content           : Hello from development!
RawContent        : HTTP/1.1 200 OK
                    Connection: keep-alive
                    Apigw-Requestid: ZE5C_haOCYcEMJA=
                    Content-Length: 23
                    Content-Type: text/plain; charset=utf-8
                    Date: Fri, 20 Feb 2026 10:47:08 GMT

                    Hello from development!
Forms             : {}
Headers           : {[Connection, keep-alive], [Apigw-Requestid,
                    ZE5C_haOCYcEMJA=], [Content-Length, 23],
                    [Content-Type, text/plain; charset=utf-8]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 23
```

Le déploiement s'est terminé correctement (Apply complete) et OpenTofu a généré l'endpoint suivant :

https://k1mgiaol2l.execute-api.us-east-2.amazonaws.com

J'ai ensuite testé l'URL depuis PowerShell. La réponse retournée est :

Hello from development!

Déploiements en staging et production

Après validation du déploiement en development, j'ai appliqué la même procédure pour les workspaces staging et production, en changeant uniquement le profil AWS utilisé.

Profils utilisés :

- stage-admin pour staging
- prod-admin pour production

Les deux déploiements se sont terminés correctement (Apply complete) et OpenTofu a généré un endpoint API pour chaque environnement :

```
Apply complete! Resources: 8 added, 0 changed, 0 destroyed.

Outputs:

api_endpoint = "https://0auseht8le.execute-api.us-east-2.amazonaws.com"
```

staging : https://0auseht8le.execute-api.us-east-2.amazonaws.com

```
Apply complete! Resources: 8 added, 0 changed, 0 destroyed.

Outputs:

api_endpoint = "https://w19t4kwpaa.execute-api.us-east-2.amazonaws.com"
```

production : https://w19t4kwpaa.execute-api.us-east-2.amazonaws.com

J'ai ensuite testé les deux endpoints avec Invoke-WebRequest (PowerShell).

```
PS C:\Users\antoi\Desktop\DevOps-ESIEE\Labs\LAB6\scripts\tofu\live\lambda-sample> (Invoke-WebRequest -UseBasicParsing "https://0auseht8le.execute-api.us-east-2.amazonaws.com").Content
Hello from staging!
PS C:\Users\antoi\Desktop\DevOps-ESIEE\Labs\LAB6\scripts\tofu\live\lambda-sample> (Invoke-WebRequest -UseBasicParsing "https://w19t4kwpaa.execute-api.us-east-2.amazonaws.com").Content
Hello from production!
```

Bilan :

La partie 2 est validée :

- les workspaces development, staging et production ont bien été créés
- la même infrastructure a été déployée sur les 3 environnements
- chaque environnement possède son propre endpoint API
- le test de l'API en development est validé

2.5 Using Different Configurations for Different Environments

Pour cette étape, j'ai utilisé le dossier lambda-sample-with-config fourni dans les scripts du lab. Ce projet permet de charger une configuration différente selon l'environnement (development, staging, production).

Problème rencontré

Lors du premier déploiement, AWS a retourné une erreur IAM EntityAlreadyExists (rôle déjà existant).

Cause :

Conflit de nom avec des ressources déjà créées dans les mêmes comptes AWS.

Correction :
J'ai modifié le nom de base dans variables.tf pour éviter la collision entre lambda-sample et lambda-sample-with-config, puis j'ai relancé les déploiements.

Déploiement et tests

J'ai déployé l'application sur les 3 environnements avec les profils AWS correspondants :

- development (dev-admin)
  Endpoint : https://z43vdowo6e.execute-api.us-east-2.amazonaws.com
  Réponse : Hello from dev config!

```
Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

api_endpoint = "https://z43vdowo6e.execute-api.us-east-2.amazonaws.com"
PS C:\Users\antoi\Desktop\DevOps-ESIEE\Labs\LAB6\scripts\tofu\live\lambda-sample-with-config> tofu output api_endpoint
"https://z43vdowo6e.execute-api.us-east-2.amazonaws.com"
PS C:\Users\antoi\Desktop\DevOps-ESIEE\Labs\LAB6\scripts\tofu\live\lambda-sample-with-config> (Invoke-WebRequest -UseBasicParsing "https://z43vdowo6e.execut
e-api.us-east-2.amazonaws.com").Content
Hello from dev config!
```

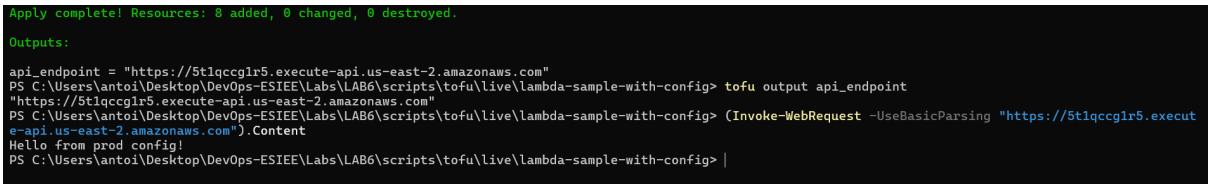
- staging (stage-admin)
  Endpoint : https://nkxrsf1e9j.execute-api.us-east-2.amazonaws.com
  Réponse : Hello from staging config!

```
Apply complete! Resources: 8 added, 0 changed, 0 destroyed.

Outputs:

api_endpoint = "https://nkxrsf1e9j.execute-api.us-east-2.amazonaws.com"
PS C:\Users\antoi\Desktop\DevOps-ESIEE\Labs\LAB6\scripts\tofu\live\lambda-sample-with-config> (Invoke-WebRequest -UseBasicParsing "https://nkxrsf1e9j.execut
e-api.us-east-2.amazonaws.com").Content
Hello from staging config!
```

- production (prod-admin)
  Endpoint : https://5t1qccg1r5.execute-api.us-east-2.amazonaws.com
  Réponse : Hello from prod config!



Cette étape valide que la même infrastructure OpenTofu peut déployer la même application sur plusieurs environnements, avec un comportement différent selon la configuration (dev / staging / prod).