

Automatisation d'Infrastructure Multi-Cloud

Déployez votre infrastructure cloud en 15 minutes au lieu de plusieurs semaines



GitHub Actions

Réalisé par :
M. Fode Mangane

Table des matières

1. Introduction à l'automatisation d'infrastructure multi-cloud	1
1.1 Pourquoi automatiser votre infrastructure multi-cloud ?	1
1.2 Les outils de la révolution DevOps	1
1.3 Architecture de la solution multi-cloud	2
2. Préparation de l'environnement multi-cloud.....	2
2.1 Configuration des comptes cloud.....	2
2.2 Création de l'utilisateur IAM terraform-user (AWS).....	2
2.3 Installation et configuration des CLI	4
3. Architecture du projet DevOps multi-cloud	7
3.1 Structure modulaire du projet	7
3.2 Organisation des composants Terraform.....	10
3.3 Configuration Ansible.....	10
3.4 Pipeline CI/CD avec GitHub Actions	10
4. Mise en place de l'infrastructure avec Terraform.....	11
4.1 Génération des clés SSH	11
4.2 Initialisation du projet Terraform	12
4.3 Planification des ressources	12
4.4 Déploiement de l'infrastructure.....	14
4.5 Vérification des ressources créées	17
5. Automatisation avec Ansible	19
5.1 Installation d'Ansible sur WSL.....	19
5.2 Configuration de l'inventaire dynamique	20
5.3 Test de connectivité	21
5.4 Exécution des playbooks.....	21
6. Intégration GitHub Actions.....	23
6.1 Configuration du repository.....	23
6.2 Mise en place du workflow	24
6.3 Déploiement automatisé.....	24
6.4 Surveillance en temps réel	26
6.5 Extension multi-cloud : Azure et Google Cloud Platform	28
7. Résultats et optimisation.....	37
7.1 Analyse des performances	37
7.2 Bonnes pratiques	38

7.3 Évolution vers l'entreprise.....	38
Conclusion.....	39

1. Introduction à l'automatisation d'infrastructure multi-cloud

1.1 Pourquoi automatiser votre infrastructure multi-cloud ?

Dans le monde professionnel actuel, les entreprises perdent des semaines entières à déployer manuellement leurs infrastructures sur différents fournisseurs cloud. Imaginez pouvoir accomplir en 10 minutes ce qui prend habituellement des semaines à une équipe entière, et ce simultanément sur AWS, Azure et Google Cloud Platform. C'est exactement ce que nous allons découvrir ensemble.

L'automatisation d'infrastructure multi-cloud représente l'avenir du développement et du déploiement. Elle élimine les erreurs humaines, garantit la reproductibilité sur tous les providers, et permet une mise à l'échelle rapide. Plus important encore, elle libère les équipes techniques pour se concentrer sur la valeur ajoutée plutôt que sur les tâches répétitives.

Important : Pendant ce parcours d'apprentissage, vous remarquerez des changements d'adresses IP fréquents. Cela est normal car nous procédons à plusieurs git add . et nous détruisons souvent l'infrastructure pour la reconstruire afin de corriger les erreurs rencontrées. Cette approche itérative fait partie intégrante du processus d'apprentissage.

Attention aux coûts : Les trois principaux fournisseurs cloud (AWS, Azure, GCP) proposent des niveaux gratuits, mais il est crucial de suivre attentivement cette procédure pour éviter de générer d'énormes factures. Nous terminerons par une implémentation multi-provider complète - soyez particulièrement vigilant avec vos requêtes et n'hésitez pas à détruire vos ressources après les tests.

1.2 Les outils de la révolution DevOps

Notre approche s'appuie sur trois piliers technologiques complémentaires :

Terraform fonctionne comme un architecte numérique multi-cloud. Il décrit votre infrastructure sous forme de code, puis la construit automatiquement selon vos spécifications sur AWS, Azure et GCP simultanément. C'est comparable à avoir un plan architectural qui se transforme instantanément en bâtiments réels sur trois continents différents.

Ansible joue le rôle du chef d'orchestre universel. Une fois l'infrastructure créée sur tous les providers, il configure automatiquement tous les services et applications de manière uniforme. Pensez à un assistant personnel qui prépare vos nouveaux appartements selon vos goûts, peu importe leur localisation.

GitHub Actions devient votre chef de projet automatisé multi-cloud. Il surveille vos modifications de code et déclenche automatiquement tous les processus de déploiement sur les trois plateformes simultanément. C'est comme avoir un assistant qui exécute parfaitement vos instructions sur tous vos environnements dès que vous les donnez.

1.3 Architecture de la solution multi-cloud

Notre solution déploie une infrastructure complète sur les trois principaux fournisseurs cloud : AWS, Azure et Google Cloud Platform. Chaque déploiement inclut un réseau virtuel privé, une instance de calcul, et un espace de stockage. Cette architecture respecte les bonnes pratiques de sécurité de chaque provider tout en restant dans les limites des niveaux gratuits.

L'ensemble du processus sera orchestré depuis GitHub, permettant un déploiement multi-cloud en un simple clic. Cette approche professionnelle vous donnera une longueur d'avance considérable sur le marché du travail, vous positionnant comme un expert en infrastructure multi-cloud.

2. Préparation de l'environnement multi-cloud

2.1 Configuration des comptes cloud

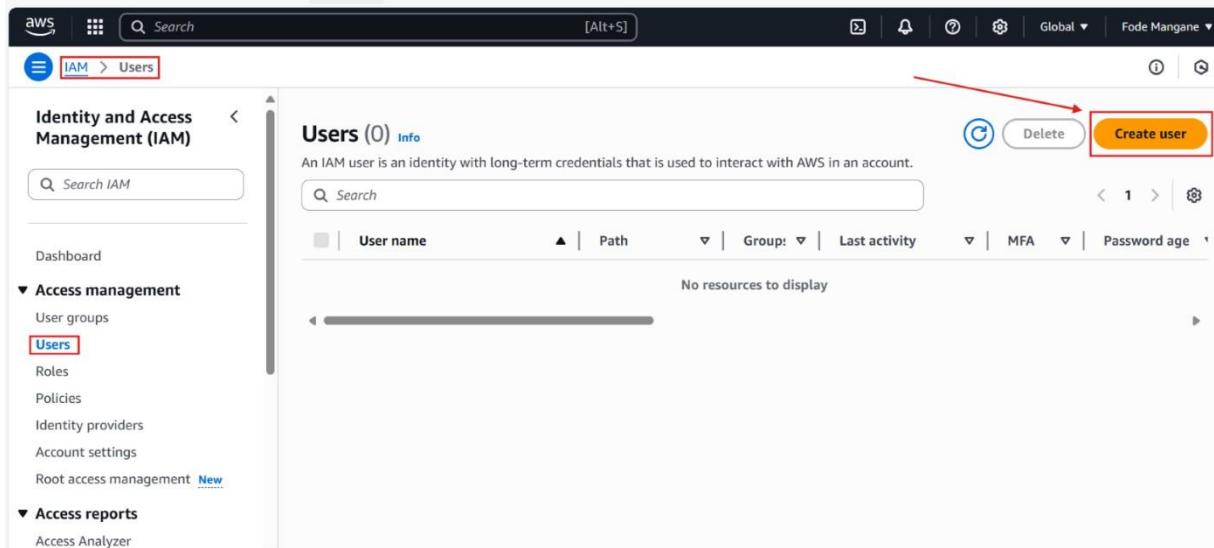
La première étape consiste à créer des comptes sur les trois principales plateformes cloud si vous n'en possédez pas déjà. Choisissez soigneusement vos régions, car elles détermineront la localisation physique de vos ressources et peuvent influencer les performances.

2.2 Création de l'utilisateur IAM terraform-user (AWS)

La sécurité AWS repose sur le principe de moindre privilège. Plutôt que d'utiliser votre compte root, nous créons un utilisateur dédié avec des permissions spécifiques.

Connectez-vous à la console AWS et naviguez vers le service IAM (Identity and Access Management). Cette interface vous permet de gérer finement les accès à vos ressources.

Créez un nouvel utilisateur nommé "terraform-user". Ce nom explicite facilitera la gestion ultérieure de vos accès.



The screenshot shows the AWS IAM Users page. A green success message at the top states "User created successfully" and "You can view and download the user's password and email instructions for signing in to the AWS Management Console." Below this, a table lists one user: "terraform-user". The table includes columns for User name, Path, Group, Last activity, MFA, and Password age. The "terraform-user" row is highlighted with a red arrow pointing to it.

Sélectionnez "Programmatic access" pour permettre à Terraform d'interagir avec AWS via des clés d'API. Cette méthode d'authentification est sécurisée et recommandée pour les outils d'automatisation.

The screenshot shows the AWS IAM User details page for "terraform-user". The "Summary" section displays the ARN (arn:aws:iam::597088023048:user/terraform-user), Console access status (Disabled), and Creation date (June 18, 2025, 20:08 (UTC)). A red arrow points to the "Create access key" button in the "Access key 1" section. The "Permissions" tab is selected, showing the "Permissions policies" section which includes a search bar and a "Filter by Type" dropdown set to "All types".

Puis attachez les politiques nécessaires à votre utilisateur. Pour ce projet, nous utiliserons des politiques AWS gérées qui couvrent EC2, VPC, et S3. Cette approche modulaire respecte les bonnes pratiques de sécurité.

Step 1

Access key best practices & alternatives

Step 2 - optional

Set description tag

Step 3

Retrieve access keys

Access key best practices & alternatives Info

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.

Local code
You plan to use this access key to enable application code in a local development environment to access your AWS account.

Application running on an AWS compute service
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

Après cela nous téléchargeons impérativement le fichier CSV contenant nos clés d'accès. Ces identifiants sont générés une seule fois et ne pourront pas être récupérés ultérieurement. Conservez-les en lieu sûr.

Access key created

This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

Step 3

Retrieve access keys

Access key | Secret access key

AKIAYWBJYGIKFZYOG6AZ | ***** Show

Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the [best practices for managing AWS access keys](#).

[Download .csv file](#) [Done](#)

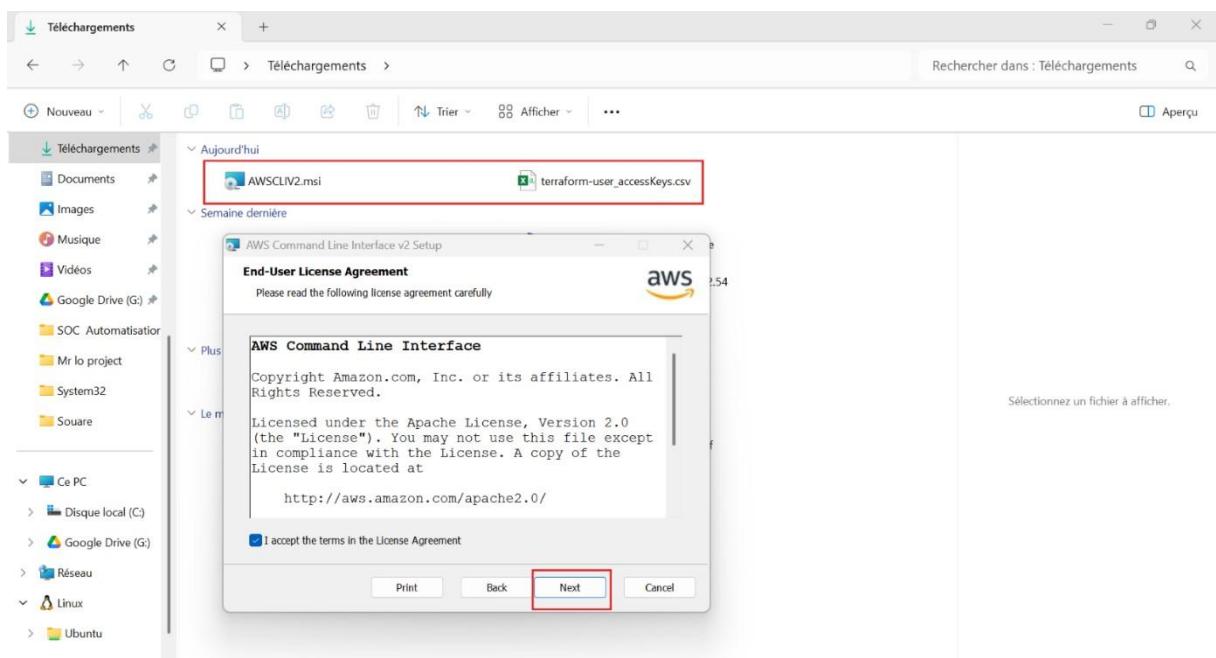
2.3 Installation et configuration des CLI

AWS CLI

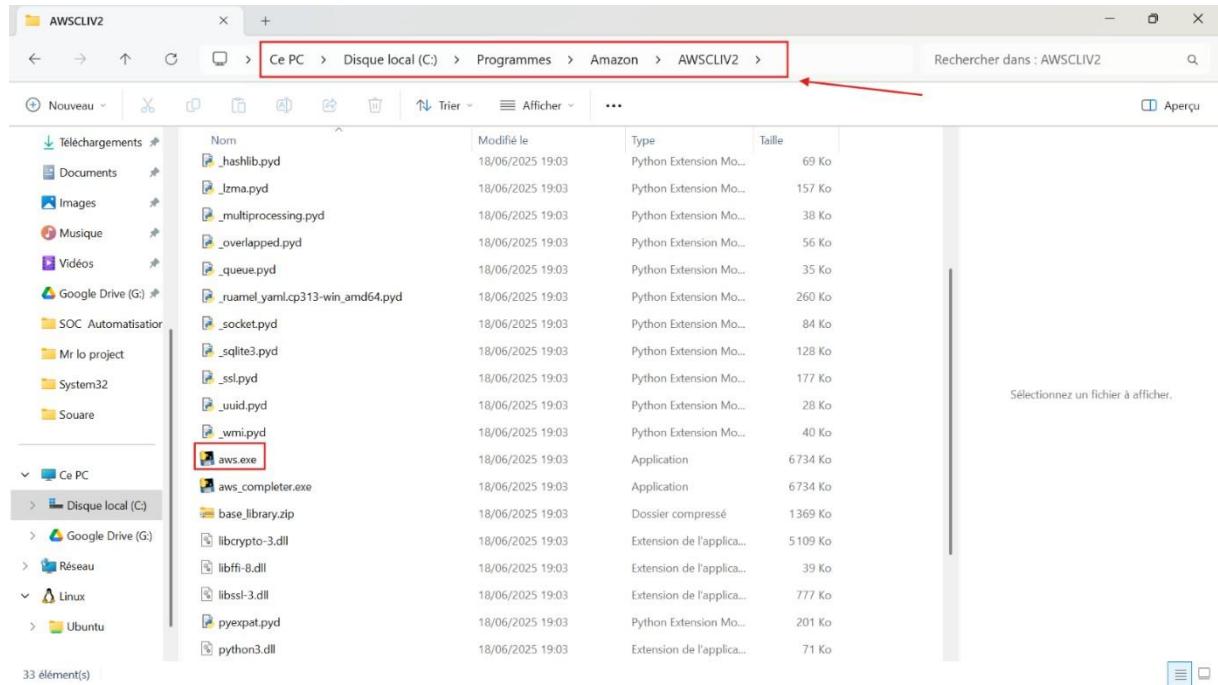
AWS CLI (Command Line Interface) constitue votre interface de communication avec les services AWS. Cet outil puissant vous permettra d'automatiser de nombreuses tâches.

The screenshot shows the AWS Command Line Interface User Guide for Version 2. The left sidebar contains links for About the AWS CLI, Get started, Prerequisites, Install/Update (which is highlighted), Past releases, Build and install from source, Amazon ECR Public/Docker, Setup, Configure the AWS CLI, Authentication and access credentials, Using the AWS CLI, and Code examples. The main content area has a heading 'Install or update the AWS CLI'. It lists requirements: 'We support the AWS CLI on Microsoft-supported versions of 64-bit Windows.' and 'Admin rights to install software'. Below this is a section titled 'Install or update the AWS CLI' with the sub-section 'Install or update the AWS CLI'. It provides instructions: '1. Download and run the AWS CLI MSI installer for Windows (64-bit): <https://awscli.amazonaws.com/AWSCLIV2.msi>'. An arrow points from this link to a screenshot of a command prompt window showing the command 'C:\> msieexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi'. The command prompt window also shows a progress bar. Below the command prompt is the text 'For various parameters that can be used with msieexec, see [msieexec](#)'.

Téléchargez la version appropriée depuis le site officiel AWS. L'installation est simple et guidée sur tous les systèmes d'exploitation.



Exécutez l'installateur en suivant les instructions. L'installation ajoute automatiquement AWS CLI à votre PATH système.



Vérifiez que AWS CLI est correctement installé en ouvrant une nouvelle fenêtre de terminal et en exécutant : aws --version

```
C:\Users\Fode>aws --version
aws-cli/2.27.38 Python/3.13.4 Windows/11 exe/AMD64 ←
C:\Users\Fode>
```

Maintenant que AWS CLI est installé, configurez-le avec vos identifiants. La commande aws configure vous guide dans cette configuration :

aws configure

```
C:\Users\Fode> aws configure
AWS Access Key ID [None]: [REDACTED]
AWS Secret Access Key [None]: [REDACTED]
Default region name [None]: eu-west-1
Default output format [None]: json
```

Renseignez les informations suivantes :

- Access Key ID** : Trouvée dans votre fichier CSV
- Secret Access Key** : Également dans le fichier CSV
- Default region** : Choisissez eu-west-1 pour la compatibilité
- Default output format** : json pour une lisibilité optimale

2.4 Vérification de la connectivité

Testons immédiatement notre configuration pour éviter les surprises ultérieures.

La commande suivante vérifie votre identité auprès d'AWS :

fodemangane@gmail.com

```
aws sts get-caller-identity
```

```
C:\Users\Fode>aws sts get-caller-identity
{
    "UserId": "AIDA[REDACTED]SIQ",
    "Account": "55[REDACTED]48",
    "Arn": "arn:aws:iam::55[REDACTED]48:user/terraform-user"
}

C:\Users\Fode>
```

Note : La même démarche est appliquée pour Azure et GCP.

3. Architecture du projet DevOps multi-cloud

3.1 Structure modulaire du projet

L'organisation de votre projet détermine sa maintenabilité et sa scalabilité. Notre architecture suit les meilleures pratiques DevOps avec une séparation claire des responsabilités et un support multi-cloud.

```
fode-devops-infrastructure/
|
|   .github/
|   |   workflows/
|   |   |   deploy.yml      # Pipeline CI/CD GitHub Actions
|
|   ansible/          # Configuration Management
|   |   ansible.cfg      # Fichier de configuration Ansible
|   |   inventory/
|   |   |   hosts.yml      # Inventaire des serveurs
|   |   playbooks/
|   |   |   site.yml        # Playbook principal multi-OS
|   |   |   site.yml.backup # Sauvegarde du playbook
|   |   |   maintenance.yml # Playbook de maintenance
|   |   |   templates/
|   |   |   |   ansible-test.html.j2 # Template page web
|   |   |   |   monitor.sh.j2  # Script de monitoring
|   |   |   |   nginx-site.conf.j2 # Config Nginx
```

```

|   └── scripts/
|       └── generate_inventory.py      # Script pour inventaire dynamique
| &²
└── terraform/                      # Infrastructure as Code
    ├── main.tf                      # Configuration principale multi-cloud
    ├── variables.tf                 # Variables globales
    ├── outputs.tf                   # Outputs multi-cloud
    ├── providers.tf                 # Providers AWS / Azure / GCP
    ├── terraform.tfvars            # Valeurs des variables
    ├── iam.tf                       # Rôles et IAM
    └── .checkov.yml                # Configuration sécurité (Checkov)
    |
    |
    └── keys/                        # Clés SSH
        └── id_rsa.pub              # Clé publique
    |
    |
    └── modules/                     # Modules Terraform réutilisables
        ├── aws/                      # AWS
        |   ├── vpc/
        |   |   ├── main.tf          # VPC, Subnets, Routage
        |   |   ├── outputs.tf
        |   |   ├── variables.tf
        |   |   └── versions.tf
        |   ├── ec2/
        |   |   ├── main.tf          # Instances EC2, ALB, SG
        |   |   ├── outputs.tf
        |   |   ├── variables.tf
        |   |   └── user_data.sh
        |   └── s3/
        |       ├── main.tf          # Bucket S3
        |       ├── outputs.tf
        |       ├── variables.tf

```

```

|   |   └── provider.tf
|   |   └── templates/
|   |       └── readme.md      # Documentation S3
|   |
|   └── azure/          # Azure
|       ├── resource_group/
|       |   ├── main.tf      # Groupe de ressources
|       |   ├── outputs.tf
|       |   └── variables.tf
|       └── vm/
|           ├── main.tf      # VM + réseau
|           ├── outputs.tf
|           ├── variables.tf
|           └── user_data.sh
|       └── storage/
|           ├── main.tf      # Stockage Azure
|           ├── outputs.tf
|           └── variables.tf
|
|   └── gcp/          # Google Cloud
|       ├── compute/
|       |   ├── main.tf      # Compute Engine + réseau
|       |   ├── outputs.tf
|       |   ├── variables.tf
|       |   ├── versions.tf
|       |   └── startup_script.sh
|       └── storage/
|           ├── main.tf      # Bucket Cloud Storage
|           ├── outputs.tf
|           ├── variables.tf
|           └── versions.tf

```

```
|      └── readme.md      # Documentation GCP  
|  
|  
└── .gitignore          # Fichiers à ignorer par Git  
└── terraform.tfstate   # Fichier d'état Terraform (local)
```

Cette structure reflète une approche professionnelle où chaque composant a sa place définie. Les modules Terraform permettent la réutilisation du code sur tous les providers, tandis que les playbooks Ansible organisent les tâches de configuration de manière uniforme.

3.2 Organisation des composants Terraform

Terraform organise votre infrastructure multi-cloud en modules réutilisables. Cette approche modulaire facilite la maintenance et permet de réutiliser des composants dans différents projets et sur différents providers.

Le **module AWS** gère vos ressources Amazon : VPC, EC2, et S3.

Le **module Azure** déploie vos ressources Microsoft : Virtual Network, Virtual Machine, et Storage Account.

Le **module GCP** provisionne vos ressources Google : VPC Network, Compute Engine, et Cloud Storage.

Cette organisation modulaire garantit une cohérence dans vos déploiements tout en respectant les spécificités de chaque fournisseur cloud.

3.3 Configuration Ansible

Ansible transforme vos serveurs bruts en environnements configurés et opérationnels, peu importe le provider cloud. Ses playbooks décrivent l'état souhaité de vos systèmes de manière uniforme.

Le fichier ansible.cfg définit les paramètres globaux d'Ansible. Il configure notamment la gestion des clés SSH et les options de parallélisation pour tous les environnements.

L'inventaire hosts.yml répertorie tous vos serveurs cibles sur les trois providers. Il peut être statique ou généré dynamiquement à partir des outputs Terraform.

Les playbooks orchestrent les tâches de configuration. Ils installent les logiciels, configurent les services et déplacent les applications de manière identique sur AWS, Azure et GCP.

3.4 Pipeline CI/CD avec GitHub Actions

GitHub Actions automatise l'ensemble du processus de déploiement multi-cloud. Un simple push sur la branche principale déclenche automatiquement le pipeline complet sur les trois plateformes.

Le workflow deploy.yml définit les étapes d'automatisation :

1. **Validation** : Vérification du code Terraform pour tous les providers
2. **Sécurité** : Scan de sécurité avec Checkov
3. **Déploiement parallèle** : Application de l'infrastructure sur AWS, Azure et GCP
4. **Configuration** : Exécution des playbooks Ansible sur tous les serveurs
5. **Tests** : Validation du déploiement multi-cloud
6. **Notification** : Rapport des résultats

Cette approche garantit des déploiements cohérents et réduit drastiquement les erreurs humaines.

4. Mise en place de l'infrastructure avec Terraform

4.1 Génération des clés SSH

La sécurité de votre infrastructure multi-cloud commence par une authentification robuste. Les clés SSH offrent une méthode d'authentification bien plus sécurisée que les mots de passe traditionnels et fonctionnent de manière uniforme sur tous les providers.

Générez une paire de clés SSH avec la commande suivante :

```
ssh-keygen -t rsa -b 4096 -C "votre-email@example.com" -f ~/.ssh/id_rsa
```

```
C:\Users\Fode>ssh-keygen -t rsa -b 4096 -C "fodemangane@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (C:/Users/Fode/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:/Users/Fode/.ssh/id_rsa
Your public key has been saved in C:/Users/Fode/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:drCD6op[REDACTED] fodemangane@gmail.com
The key's randomart image is:
+---[RSA 4096]---+
|...=.o=|
|ooo.ooo .|
|. o+.*..|
| . B E o|
| ..+ * S .|
| ..o= . o|
| ..+o |
|=B.+o |
|B==o.|
+---[SHA256]---+
C:\Users\Fode>
```

Cette commande crée une clé privée (que vous gardez secrète) et une clé publique (que vous partagez avec vos serveurs). La clé publique sera intégrée dans vos instances sur tous les providers lors du déploiement.

Localisez votre clé publique qui sera nécessaire pour la configuration Terraform. Elle se trouve généralement dans `~/.ssh/id_rsa.pub`.

```

PS C:\Users\Fode> Get-ChildItem $HOME\.ssh
Répertoire : C:\Users\Fode\.ssh ←

Mode                LastWriteTime     Length Name
----                -----          ---- 
-a----       19/06/2025    00:42        3389 id_rsa
-a----       19/06/2025    00:42        748 id_rsa.pub
-a----       10/04/2025   23:35      3211 known_hosts
-a----       10/04/2025   23:35      2463 known_hosts.old

PS C:\Users\Fode>

```

4.2 Initialisation du projet Terraform

Maintenant que notre environnement est prêt, commençons par comprendre le processus avant de l'automatiser. Cette approche pédagogique vous permettra de mieux appréhender l'importance de l'automatisation.

Naviguez vers le dossier terraform de votre projet et initialisez Terraform :

```
cd terraform
terraform init
```

```

PS C:\Users\Fode\Desktop\fode-devops-infrastructure\terraform> terraform init
Initializing the backend...
Initializing modules...
- ec2 in modules\ec2
- s3 in modules\s3
- vpc in modules\vpc
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of hashicorp/random from the dependency lock file
- Installing hashicorp/random v3.7.2...
- Installed hashicorp/random v3.7.2 (signed by HashiCorp)
- Installing hashicorp/aws v5.100.0...
- Installed hashicorp/aws v5.100.0 (signed by HashiCorp)

Terraform has been successfully initialized!

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

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS C:\Users\Fode\Desktop\fode-devops-infrastructure\terraform>

```

Cette commande télécharge les providers AWS, nécessaires et prépare votre environnement de travail. Terraform crée un répertoire .terraform contenant tous les composants nécessaires pour les trois plateformes.

4.3 Planification des ressources

Avant tout déploiement, Terraform vous permet de visualiser exactement ce qui sera créé sur chaque provider. Cette étape de planification évite les mauvaises surprises :

```
terraform plan
```

La première exécution peut échouer en raison de permissions insuffisantes. C'est normal et attendu dans un environnement de production multi-cloud.

The screenshot shows the VS Code interface with the Terraform extension installed. The left sidebar displays the project structure under the 'TERRAFORM' folder:

- .terraform
- modules
 - ec2
 - s3
 - vpc
- templates
- variables
- user_data.sh
- main.tf
- outputs.tf
- variables.tf
- versions.tf
- .terraform.lock.hcl
- readme.md
- terraform.tfstate
- terraform.tfstate.backup
- terraform.tfvars

The 'main.tf' file is selected in the Explorer view. The main code editor shows the Terraform configuration for EC2, S3, and VPC modules. The terminal at the bottom shows the command 'terraform plan' being run, and the output lists various resources being refreshed.

Retournez dans les consoles AWS IAM, Azure Active Directory et Google Cloud IAM pour ajouter les permissions nécessaires à vos utilisateurs. Les permissions requises dépendent des ressources que vous souhaitez créer sur chaque plateforme.

Attachez les politiques suivantes à vos utilisateurs :

AWS :

- AmazonEC2FullAccess pour les instances
- AmazonVPCFullAccess pour le réseau
- AmazonS3FullAccess pour le stockage

Azure : Contributor sur votre subscription

GCP : Editor sur votre projet

Relancez la planification une fois les permissions ajoutées. Terraform affiche maintenant un plan détaillé des ressources à créer sur les trois providers.

```

1 terraform {
2   required_version = ">= 1.0"
3   required_providers {
4     aws = {
5       source  = "hashicorp/aws"
6       version = "~> 5.0"
7     }
8     random = {
9       source  = "hashicorp/random"
10      version = "~> 3.4"
11    }
12  }
13 }
14 provider "aws" {
15   region = var.aws_region
16 }
17 default_tags {
18   tags = {
19 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Plan: 16 to add, 0 to change, 0 to destroy.

Changes to Outputs:

- + ec2_instance_id = (known after apply)
- + ec2_public_ip = (known after apply)
- + public_subnet_id = (known after apply)
- + s3_bucket_name = (known after apply)
- + ssh_command = (known after apply)
- + vpc_id = (known after apply)
- + website_url = (known after apply)

Remarque : ici, vous voyez principalement les outputs d'AWS, car j'ai suivi les étapes progressivement en automatisant depuis mon environnement local. Par la suite, j'ai intégré Azure et GCP au fur et à mesure, en améliorant continuellement le code.

Ce plan liste précisément chaque ressource qui sera créée, modifiée ou supprimée sur AWS, Azure et GCP. Examinez-le attentivement avant de procéder au déploiement.

4.4 Déploiement de l'infrastructure

Maintenant que le plan est validé, déployons notre infrastructure : terraform

apply

```

PS C:\Users\ode\Desktop\Terraform> terraform apply

```

The terminal output shows the creation of various AWS resources across three regions (us-east-1, eu-west-1, us-west-2), including EC2 instances, S3 buckets, VPCs, and security groups. Terraform is refreshing state for each module and resource.

Terraform vous demande une confirmation avant de créer les ressources. Tapez yes pour confirmer le déploiement sur les trois plateformes.

```

42 # Module EC2 Fode-DevOps
43 module "ec2" {
44   source      = "./modules/ec2"
45   project_name = var.project_name
46   environment  = var.environment
47   vpc_id       = module.vpc.vpc_id
48   public_subnet_id = module.vpc.public_subnet_ids[0]
49   instance_type = var.instance_type
50   key_name     = var.key_pair_name
51 }
52
53 # Module S3 Fode-DevOps
54 module "s3" {
55   source      = "./modules/s3"
56   project_name = var.project_name
57   environment  = var.environment
58
59   > -> (known after apply)
60   id          = "config/fode-devops.json"
61   tags        = {}
62   ~ version_id = "null" -> (known after apply)
63   # (24 unchanged attributes hidden)
64 }

```

Plan: 0 to add, 1 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

Le processus de création prend quelques minutes. Terraform affiche le progrès en temps réel et vous informe de chaque ressource créée sur AWS, Azure et GCP.

Parfois, des permissions supplémentaires sont nécessaires pendant le déploiement. Terraform vous indique précisément quelles permissions ajouter sur quel provider.

Step 1
 Specify permissions
 Step 2
 Review and create

Specify permissions Info

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

Policy editor

```

1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "ec2:*",
8         "s3:)"
9       ],
10      "Resource": "*"
11    }
12  ]
13 }
14

```

Edit statement

Select a statement

Select an existing statement in the policy or add a new statement.

+ Add new statement

The screenshot shows the AWS IAM 'Add permissions' interface. Under 'Permissions options', the 'Attach policies directly' radio button is selected. Below it, a table lists policies: 'terraform-project-policy' (Customer managed) is highlighted with a red arrow pointing to it. Other policies like 'Copy permissions' and 'Add user to group' are also listed.

Ajoutez les permissions manquantes et relancez le déploiement. Cette approche itérative est normale lors de la première configuration .

Une fois le déploiement terminé, Terraform affiche un résumé des ressources créées sur les trois plateformes et leurs caractéristiques principales.

The screenshot shows the VS Code interface with the Terraform extension. The left sidebar shows the project structure with files like main.tf, variables.tf, outputs.tf, and provider files. The right panel shows the terminal output of a Terraform apply command. A red box highlights the output for an S3 module:

```

module.s3.aws_s3_object.config: Modifications complete after 1s [id=config/fode-devops.json]
Apply complete! Resources: 0 added, 1 changed, 0 destroyed.

Outputs:
ec2_instance_id = "i-0b9f9c5b0ed3cab9"
ec2_public_ip = "54.76.235.211"
public_subnet_id = "subnet-0245c3ce9efdc755a"
s3_bucket_name = "Fode-devops-prod-enhdkatg"
ssh_command = "ssh -i ~/.ssh/id_rsa ec2-user@54.76.235.211"
vpc_id = "vpc-092bd445c63c77d6"
website_url = "http://54.76.235.211"

```

4.5 Vérification des ressources créées

Votre infrastructure est maintenant opérationnelle. Vérifions sa création dans la console AWS.

Connectez-vous à la console AWS pour visualiser vos ressources nouvellement créées.

Votre instance EC2 est maintenant visible dans la console. Elle possède une adresse IP publique et est configurée avec votre clé SSH.

The screenshot shows the AWS EC2 Instances page. On the left sidebar, under the 'Instances' section, the 'Instances' link is highlighted. In the main content area, the 'Instances (1/1)' table shows one item: 'i-0b9f9c5b04ed3cab9 (fode-devops-prod-web-server)'. The 'Public IPv4 address' field contains '54.76.235.211 | open address'.

Le VPC créé par Terraform inclut tous les composants réseau nécessaires : sous-réseaux, tables de routage, et passerelle Internet.

The screenshot shows the AWS VPC console. On the left sidebar, under the 'Cloud privé virtuel' section, the 'Vos VPC' link is highlighted. In the main content area, the 'Tableau de bord du VPC' section shows two VPC entries: 'vpc-00383cd004ec31d72' and 'fode-devops-prod-vpc'. The 'fode-devops-prod-vpc' entry is selected and highlighted with a red border. The details for 'vpc-092b1d445c03c77d6 / fode-devops-prod-vpc' are shown in the bottom panel.

Votre bucket S3 est prêt à recevoir des fichiers. Il est configuré avec les bonnes politiques de sécurité et de chiffrement.

Aperçu du compte : mis à jour toutes les 24 heures [Toutes les régions AWS]

Storage Lens offre une visibilité sur l'utilisation du stockage et les tendances d'activité. Les métriques ne comprennent pas les compartiments de répertoire. [En savoir plus](#)

Nom	Région AWS	Analyseur d'accès IAM	Date de création
fode-devops-prod-enhdkatg	Europe (Irlande) eu-west-1	Afficher l'analyseur pour eu-west-1	19 Jun 2025 11:46:25 PM GMT

Testons maintenant la connectivité SSH à notre instances : ssh

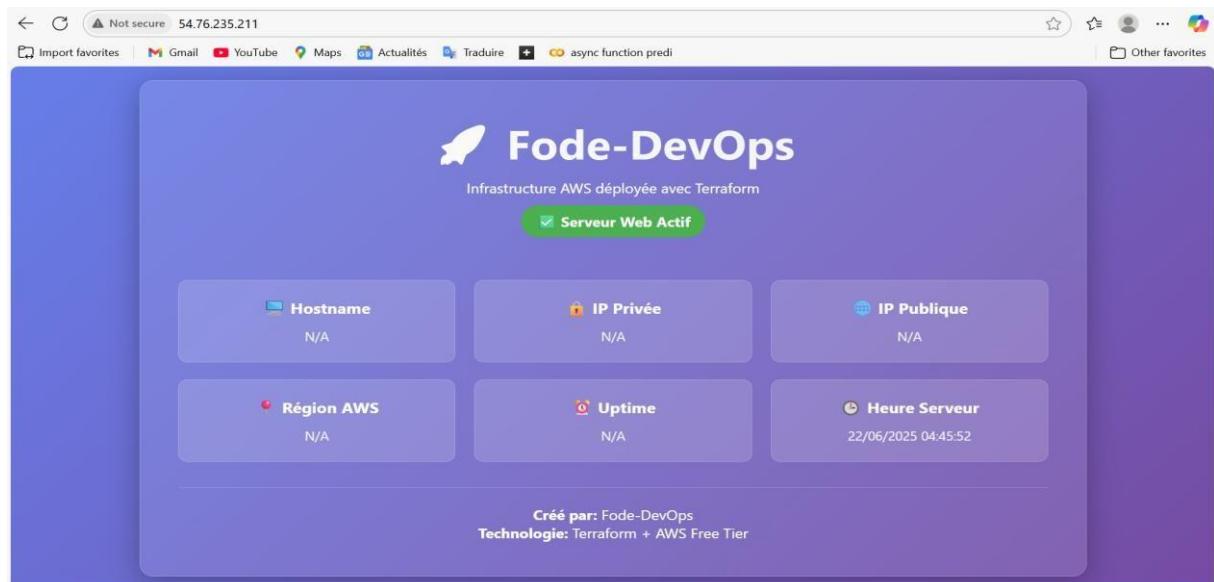
-i ~/.ssh/id_rsa ec2-user@<votre-ip-publique-aws>

```

PS C:\Users\Fode> ssh -i ~/.ssh/id_rsa ec2-user@54.76.235.211
The authenticity of host '54.76.235.211 (54.76.235.211)' can't be established.
ED25519 key fingerprint is SHA256:+bswSw596rHpjM56NZCo8DSbGMq1aBS0tiUDv7ZLtrM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.76.235.211' (ED25519) to the list of known hosts.

      _#
     /_#####
    \###\
   \##|
  \#/
  V~'-->
  /`_
 /_`/
 /m/`_

For documentation, visit http://aws.amazon.com/documentation/ecs
[Bienvenue sur l'infrastructure Fode-DevOps!]
Commandes utiles: monitor, web-status, web-restart, web-logs
[ec2-user@ip-10-0-1-163 ~]$
```



Ces connexions confirment que votre infrastructure est parfaitement opérationnelle et sécurisée.

5. Automatisation avec Ansible

5.1 Installation d'Ansible sur WSL

Maintenant que vous comprenez le processus de déploiement Terraform , explorons Ansible pour automatiser la configuration de nos serveurs sur tous les providers.

Ansible nécessite un environnement Linux. Si vous utilisez Windows, activez WSL (Windows Subsystem for Linux) pour obtenir un environnement Linux natif.

The screenshot shows the VS Code interface with a dark theme. The left sidebar has an 'EXPLORER' view showing a 'TERRAFORM' folder containing 'modules' (ec2, s3), 'templates' (readme.md), and 'vpc' subfolders, each with its own main.tf, outputs.tf, variables.tf, and versions.tf files. The main editor area shows a portion of the 'main.tf' file with code for VPC and EC2 modules. Below the editor is a 'TERMINAL' tab showing a WSL session with the command 'ls' run in the directory 'C:\Users\fode\Desktop\Terraform'. The terminal output lists files like main.tf, modules, outputs.tf, terraform.tfstate, terraform.tfstate.backup, terraform.tfvars, and variables.tf.

Une fois WSL configuré, installez Ansible : sudo

apt update sudo apt install ansible -y

The terminal window shows the command 'sudo apt install ansible -y' being run in a WSL session. The output indicates that Ansible is already at the newest version (2.10.7+merged+base+2.10.8+dfsg-1) and no upgrades or installations were needed. The command 'ansible --version' is then run, showing the installed version as 2.10.8.

Vérifiez l'installation : ansible

--version

5.2 Configuration de l'inventaire dynamique

Ansible a besoin de connaître vos serveurs cibles sur tous les providers. Créez d'abord la structure de votre projet Ansible :

mkdir -p ~/fode-devops-ansible/{inventory,playbooks,roles}

```
fode@DESKTOP-80470AP:/mnt/c/Users/Fode/Desktop/Terraform$ mkdir -p ansible/{inventory,playbooks/templates}
fode@DESKTOP-80470AP:/mnt/c/Users/Fode/Desktop/Terraform$ cd ansible
fode@DESKTOP-80470AP:/mnt/c/Users/Fode/Desktop/Terraform/ansible$ vim ansible.cfg ←
fode@DESKTOP-80470AP:/mnt/c/Users/Fode/Desktop/Terraform/ansible$
```



```
fode@DESKTOP-80470AP:~$ mkdir -p ~/.fode-devops-ansible
fode@DESKTOP-80470AP:~$ cp -r /mnt/c/Users/Fode/Desktop/Terraform/ansible/* ~/.fode-devops-ansible/
fode@DESKTOP-80470AP:~$ cd ~/.fode-devops-ansible
fode@DESKTOP-80470AP:~/fode-devops-ansible$ ls
ansible.cfg inventory playbooks
fode@DESKTOP-80470AP:~/fode-devops-ansible$ chmod 755 .
fode@DESKTOP-80470AP:~/fode-devops-ansible$ chmod 644 ansible.cfg inventory/hosts.yml playbooks/*.yml playbooks/templates/*
.j2
fode@DESKTOP-80470AP:~/fode-devops-ansible$ chmod 644 playbooks/templates/*.html.j2
fode@DESKTOP-80470AP:~/fode-devops-ansible$
```

Déplacez-vous dans le projet où se trouve notre terraform :

```
fode@DESKTOP-80470AP:~$ mkdir -p ~/fode-devops-ansible
fode@DESKTOP-80470AP:~$ cp -r /mnt/c/Users/Fode/Desktop/Terraform/ansible/* ~/fode-devops-ansible/
fode@DESKTOP-80470AP:~$ cd ~/fode-devops-ansible
fode@DESKTOP-80470AP:~/fode-devops-ansible$ ls
ansible.cfg inventory playbooks
fode@DESKTOP-80470AP:~/fode-devops-ansible$ chmod 755 .
fode@DESKTOP-80470AP:~/fode-devops-ansible$ chmod 644 ansible.cfg inventory/hosts.yml playbooks/*.yml playbooks/templates/*
.j2
fode@DESKTOP-80470AP:~/fode-devops-ansible$ chmod 644 playbooks/templates/*.html.j2
fode@DESKTOP-80470AP:~/fode-devops-ansible$
```

Créez votre fichier d'inventaire hosts.yml avec les adresses IP de vos instances sur AWS, Azure et GCP (obtenues depuis les outputs Terraform).

5.3 Test de connectivité

Avant d'exécuter vos playbooks, testez la connectivité vers tous vos serveurs :

ansible all --list-hosts

```
fode@DESKTOP-80470AP:~/fode-devops-ansible$ ansible all --list-hosts
hosts (1):
  fode-web-server ←
fode@DESKTOP-80470AP:~/fode-devops-ansible$
```

Testez la connexion SSH sur tous les providers :

ansible all -m ping

```
fode@DESKTOP-80470AP:~/fode-devops-ansible$ ansible all -m ping
fode-web-server | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
fode@DESKTOP-80470AP:~/fode-devops-ansible$
```

Cette commande confirme qu'Ansible peut communiquer avec vos serveurs sur AWS. Si elle échoue, vérifiez vos clés SSH et votre configuration réseau.

5.4 Exécution des playbooks

Maintenant qu'Ansible est configuré, exécutons notre playbook principal : ansible-playbook playbooks/site.yml

```

1 all:
2   children:
3     webservers:
4       hosts:
5         fode-web-server:
6           ansible_host: 54.76.235.211
7           ansible_user: ec2-user

```

The screenshot shows the VS Code interface with the 'hosts.yml' file selected in the Explorer sidebar. The code editor displays the YAML configuration for an Ansible host. A red box highlights the host entry for 'fode-web-server'.

Ansible configure automatiquement vos serveurs selon les spécifications du playbook sur tous les providers. Il installe les logiciels, configure les services et déploie les applications de manière uniforme.

```

PLAY [ Configuration complète infrastructure Fode-DevOps] ****
*
TASK [ Gathering Facts ] ****
ok: [fode-web-server]

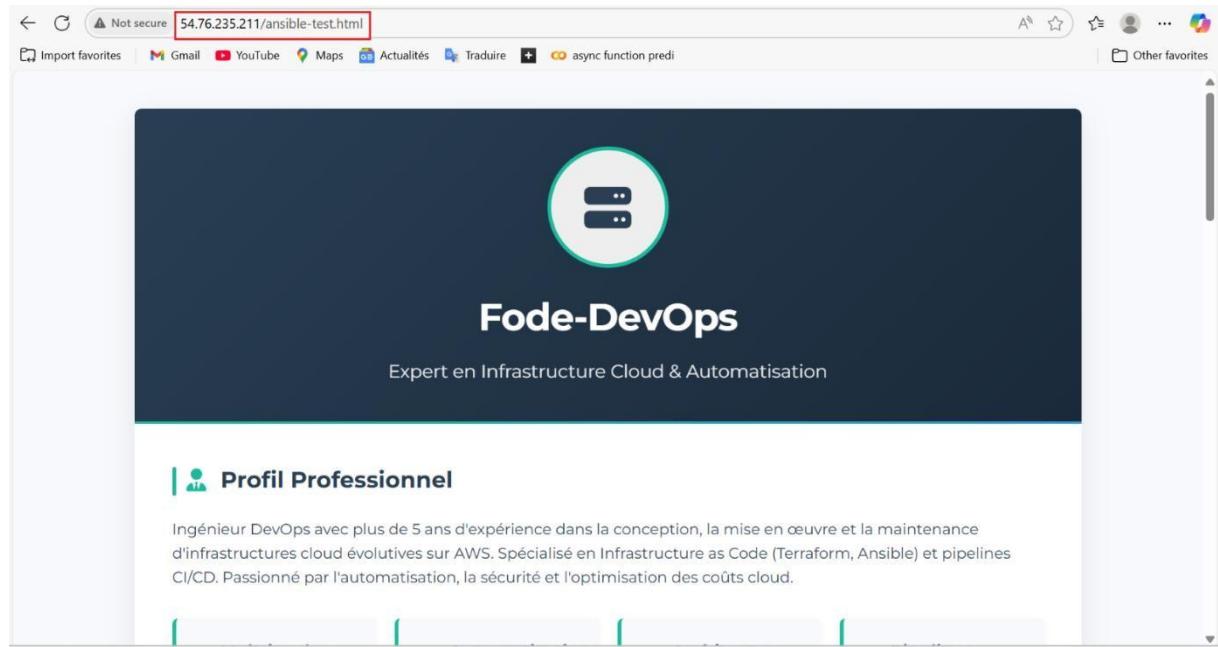
TASK [ Affichage bannière Fode-DevOps ] ****
*
ok: [fode-web-server] =>
  msg: |2-
    =====
    # FODE-DEVS ANSIBLE AUTOMATION
    =====
  Infrastructure: fode-devops-[]
  Date: 2025-06-22

```

The screenshot shows the terminal output of the Ansible playbook run. It includes validation results and success messages for the deployment. A red box highlights the success message: 'Configuration Ansible terminée avec succès !'.

Le playbook a terminé avec succès. Vos serveurs sont maintenant entièrement configurés et opérationnels.

Vérifiez que les services sont correctement déployés en accédant à votre application via l'adresse IP publique.



6. Intégration GitHub Actions

6.1 Configuration du repository

Maintenant que vous maîtrisez les processus manuels, automatisons tout avec GitHub Actions. Cette automatisation transforme des heures de travail manuel en quelques minutes d'exécution automatique sur les trois plateformes.

Configurez d'abord votre repository GitHub avec un token d'accès personnel :

```
git remote set-url origin https://votre-username:votre-token@github.com/votreusername/votre-repo.git
```

```
PS C:\Users\Fode\Desktop\fode-devops-infrastructure> git remote set-url origin https://FodeMangane:github_pat_11BFKEAQQ0h98x1Q0tIvay_KmuhUUy
m6fvZ8IjBmRJ@github.com/FodeMangane/fode-devops-infrastructure.git
PS C:\Users\Fode\Desktop\fode-devops-infrastructure>
```

Ce token permet à GitHub Actions d'accéder à votre repository de manière sécurisée.

Pour gérer l'état Terraform de manière collaborative, créons une table DynamoDB pour le verrouillage :

```

● PS C:\Users\Fode\Desktop\fode-devops-infrastructure\terraform> aws dynamodb create-table --table-name fode-devops-terraform-locks --attribute-definitions AttributeName=LockID,AttributeType=S --key-schema AttributeName=LockID,KeyType=HASH --provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5 --region us-east-1 --tags Key=Name,Value="Terraform State Lock" Key=Project,Value="Fode-DevOps"
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "LockID",
                "AttributeType": "S"
            }
        ],
        "TableName": "fode-devops-terraform-locks",
        "KeySchema": [
            {
                "AttributeName": "LockID",
                "KeyType": "HASH"
            }
        ],
        "TableStatus": "CREATING",
        "CreationDateTime": "2025-06-26T06:08:04.761000+00:00",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 5,
            "WriteCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:us-east-1:1597088023048:table/fode-devops-terraform-locks",
        "TableId": "3ecd39e4-bf95-4bd2-83c4-3242e655fac5",
        "DeletionProtectionEnabled": false
    }
}
PS C:\Users\Fode\Desktop\fode-devops-infrastructure\terraform>

```

Cette table évite les conflits lors de déploiements simultanés multi-cloud, une pratique essentielle en environnement professionnel.

6.2 Mise en place du workflow

Créez le fichier `.github/workflows/deploy.yml` qui définit votre pipeline d'automatisation multicloud. Ce workflow orchestre l'ensemble du processus de déploiement sur AWS, Azure et GCP.

Le workflow comprend plusieurs étapes :

1. **Validation** : Vérification du code Terraform
2. **Sécurité** : Scan de sécurité avec Checkov
3. **Déploiement parallèle** : Application de l'infrastructure
4. **Configuration** : Exécution des playbooks Ansible
5. **Tests** : Validation du déploiement
6. **Notification** : Rapport des résultats consolidés

6.3 Déploiement automatisé

Commitez vos modifications et poussez-les vers GitHub :

`git add .`

```
git commit -m "Fode DevOps" git
```

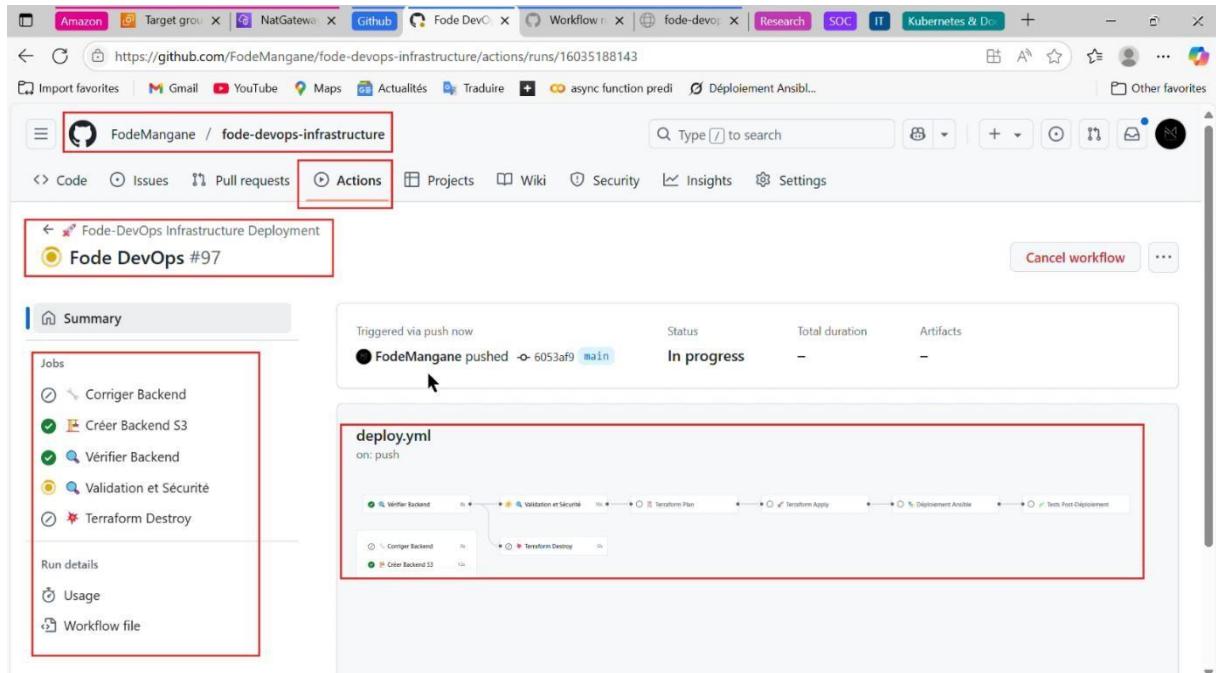
```
push origin main
```

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows the project structure for "FODE-DEVP-INFRASTRUCTURE".
- TERMINAL:** Displays the command history:

```
PS C:\Users\Fode\Desktop\Fode-devops-infrastructure> git commit -m "Fode DevOps"
[main 6053af9] Fode DevOps
4 files changed, 270 insertions(+), 54 deletions(-)
PS C:\Users\Fode\Desktop\Fode-devops-infrastructure> git push origin main
```
- STATUS BAR:** Shows the current file is "main" and has 01:11 remaining.

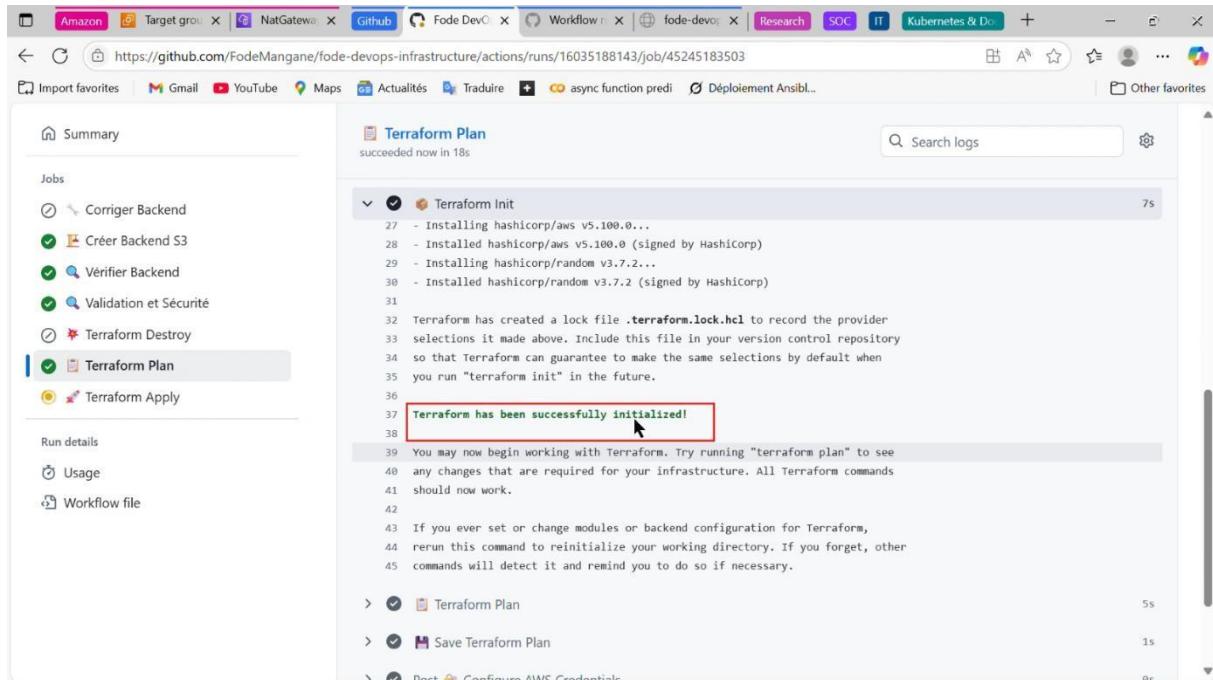
Le simple push déclenche automatiquement votre pipeline GitHub Actions. Plus besoin d'interventions manuelles.



Surveillez l'exécution depuis l'interface GitHub Actions. Vous pouvez suivre chaque étape en temps réel sur les trois plateformes.

6.4 Surveillance en temps réel

Le workflow exécute automatiquement toutes les étapes que vous avez réalisées manuellement précédemment.



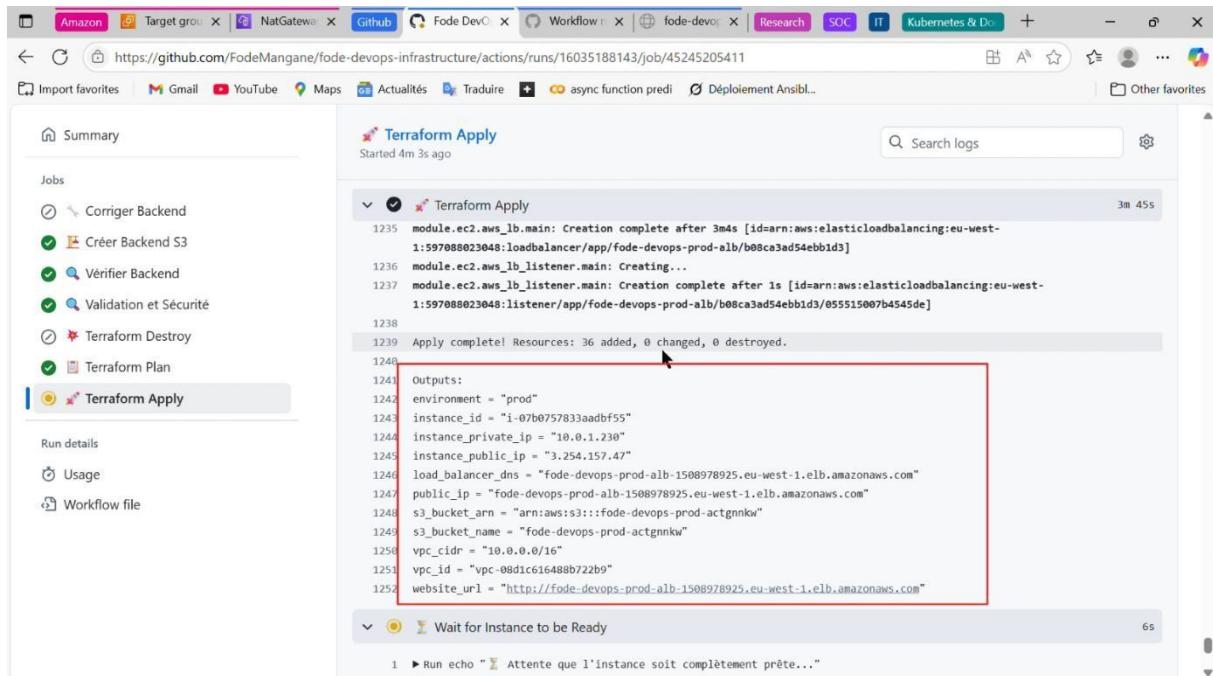
```
https://github.com/FodeMangane/fode-devops-infrastructure/actions/runs/16035188143/job/45245183503

Summary
Jobs
Corriger Backend
Créer Backend S3
Vérifier Backend
Validation et Sécurité
Terraform Destroy
Terraform Plan
Terraform Apply
Run details
Usage
Workflow file

Terraform Plan
succeeded now in 18s
Terraform Init
27 - Installing hashicorp/aws v5.100.0...
28 - Installed hashicorp/aws v5.100.0 (signed by HashiCorp)
29 - Installing hashicorp/random v3.7.2...
30 - Installed hashicorp/random v3.7.2 (signed by HashiCorp)
31
32 Terraform has created a lock file .terraform.lock.hcl to record the provider
33 selections it made above. Include this file in your version control repository
34 so that Terraform can guarantee to make the same selections by default when
35 you run "terraform init" in the future.
36
37 Terraform has been successfully initialized!
38
39 You may now begin working with Terraform. Try running "terraform plan" to see
40 any changes that are required for your infrastructure. All Terraform commands
41 should now work.
42
43 If you ever set or change modules or backend configuration for Terraform,
44 rerun this command to reinitialize your working directory. If you forget, other
45 commands will detect it and remind you to do so if necessary.

Terraform Plan
Save Terraform Plan
Dont Configure AWS Credentials
```

Terraform valide et applique votre infrastructure automatiquement, en respectant les mêmes bonnes pratiques.



```
https://github.com/FodeMangane/fode-devops-infrastructure/actions/runs/16035188143/job/45245205411

Summary
Jobs
Corriger Backend
Créer Backend S3
Vérifier Backend
Validation et Sécurité
Terraform Destroy
Terraform Plan
Terraform Apply
Run details
Usage
Workflow file

Terraform Apply
Started 4m 3s ago
Terraform Apply
1235 module.ec2.aws_lb.main: Creation complete after 3m4s [id=arn:aws:elasticloadbalancing:eu-west-1:597088023048:loadbalancer/app/Fode-devops-prod-alb/b08ca3ad54ebbd3]
1236 module.ec2.aws_lb_listener.main: Creating...
1237 module.ec2.aws_lb_listener.main: Creation complete after 1s [id=arn:aws:elasticloadbalancing:eu-west-1:597088023048:listener/app/Fode-devops-prod-alb/b08ca3ad54ebbd3/055515007b4545de]
1238
1239 Apply complete! Resources: 36 added, 0 changed, 0 destroyed.
1240
1241 Outputs:
1242 environment = "prod"
1243 instance_id = "i-07b075783aaadbf55"
1244 instance_private_ip = "10.0.1.230"
1245 instance_public_ip = "3.254.157.47"
1246 load_balancer_dns = "Fode-devops-prod-alb-1508978925.eu-west-1.elb.amazonaws.com"
1247 public_ip = "fode-devops-prod-alb-1508978925.eu-west-1.elb.amazonaws.com"
1248 s3_bucket_arn = "arn:aws:s3:::fode-devops-prod-actgnnk"
1249 s3_bucket_name = "fode-devops-prod-actgnnk"
1250 vpc_cidr = "10.0.0.0/16"
1251 vpc_id = "vpc-08dic6488b722b9"
1252 website_url = "http://fode-devops-prod-alb-1508978925.eu-west-1.elb.amazonaws.com"

Wait for Instance to be Ready
1 ► Run echo "Attente que l'instance soit complètement prête..."
2 Attente que l'instance soit complètement prête...
```

Ansible configure vos serveurs selon vos spécifications, sans intervention humaine, de manière uniforme.

The screenshot shows the GitHub Actions interface for a deployment job. The left sidebar lists various jobs: Corriger Backend, Créer Backend S3, Vérifier Backend, Validation et Sécurité, Terraform Destroy, Terraform Plan, Terraform Apply, Déploiement Ansible (which is selected), and Tests Post-Déploiement. The main area displays the logs for the 'Déploiement Ansible' step, which succeeded in 58 seconds. The logs show the execution of Ansible tasks to configure infrastructure, including output from FODE-DEVOPS ANSIBLE AUTOMATION and details about the Amazon Linux 2023 server.

```

23
24 PLAY [ Configuration infrastructure Fode-DevOps avec Nginx] ****
25
26 TASK [Gathering Facts] ****
27 ok: [web-server-prod]
28
29 TASK [ Affichage bannière Fode-DevOps] ****
30 ok: [web-server-prod] => {
31     "msg": "=====\\n # FODE-DEVOPS ANSIBLE AUTOMATION #\\n=====\\nInfrastructure: Fode-DevOps-[ ]\\nDate: 2025-07-02 20:38:52\\nServeur: web-server-prod\\nos: Amazon Linux 2023\\n=====\\n"
32 }
33
34 TASK [ Mise à jour du cache des packages] ****
35 ok: [web-server-prod] => {"attempts": 1, "changed": false, "msg": "Cache updated", "rc": 0, "results": []}
36
37 TASK [ Mise à jour du système Amazon Linux] ****
38 ok: [web-server-prod] => {"attempts": 1, "changed": false, "msg": "Nothing to do", "rc": 0, "results": []}
39
40 TASK [ Installation des packages essentiels] ****
41 changed: [web-server-prod] => {"attempts": 1, "changed": true, "msg": "", "rc": 0, "results": ["Installed: nginx-filesystem-1:1.28.0-1.amzn2023.0.1.noarch", "Installed: nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch", "Installed: libunwind-1.4.0-5.amzn2023.0.2.x86_64", "Installed: generic-logos-httplib-18.0.0-12.amzn2023.0.3.noarch", "Installed: http-3.2.1-87.amzn2023.0.3.x86_64", "Installed: nginx-1:1.28.0-1.amzn2023.0.1.x86_64", "Installed: libmetalink-0.1.3-1.amzn2023.0.1.noarch"]}
```

Des tests automatisés vérifient que votre déploiement fonctionne correctement.

The screenshot shows the GitHub Actions interface for a deployment job. The left sidebar lists the same set of jobs as the previous screenshot. The main area displays the logs for the 'Tests Post-Déploiement' step, which succeeded in 3m 4s. The logs show the execution of a 'Test Load Balancer' task, which includes running a command to check the service and performing an HTTP test. The task completed successfully in 3m 0s.

```

1 Run LB_DNS="fode-devops-prod-alb-1508978925.eu-west-1.elb.amazonaws.com"
2 Attente le démarrage du service web...
3 Test de connectivité HTTP vers http://fode-devops-prod-alb-1508978925.eu-west-1.elb.amazonaws.com...
4 Service web accessible et contenu correct détecté!
```

Le processus complet s'exécute en moins de 15 minutes, transformant des semaines de travail en quelques minutes d'automatisation.

6.5 Extension multi-cloud : Azure et Google Cloud Platform

Maintenant que vous maîtrisez le déploiement AWS, étendons notre solution aux autres principaux fournisseurs cloud. Cette extension multi-provider représente le niveau le plus avancé de l'automatisation d'infrastructure moderne.

Configuration Azure

Pour Azure, commençons par nous connecter avec nos identifiants via Azure CLI : az

login

```
PS C:\Users\Fode\Desktop\fode-devops-infrastructure> az login
Select the account you want to log in with. For more information on login with Azure CLI, see https://go.microsoft.com/fwlink/?linkid=2271136
Retrieving tenants and subscriptions for the selection...
[Tenant and subscription selection]
No      Subscription name      Subscription ID          Tenant
-----[1] *  Azure subscription 1  9f0258a3-dfb0-47cc-b1ed-c47d76fc9e6  Default Directory
The default is marked with an *; the default tenant is 'Default Directory' and subscription is 'Azure subscription 1' (9f0258a3-dfb0-47cc-b1ed-c47d76fc9e6).
Select a subscription and tenant (Type a number or Enter for no changes): 1
Tenant: Default Directory
Subscription: Azure subscription 1 (9f0258a3-dfb0-47cc-b1ed-c47d76fc9e6)
[Announcements]
With the new Azure CLI login experience, you can select the subscription you want to use more easily. Learn more about it and its configuration at https://go.microsoft.com/fwlink/?linkid=2271236
If you encounter any problem, please open an issue at https://aka.ms/azclibug
```

Une fois connecté, vérifions notre configuration : az

account show

```
PS C:\Users\Fode\Desktop\fode-devops-infrastructure> az account show
{
  "environmentName": "AzureCloud",
  "homeTenantId": "5d0826be-c51f-4f77-9bb7-5e489c051998",
  "id": "9f0258a3-dfb0-47cc-b1ed-c47d76fc9e6",
  "isDefault": true,
  "managedByTenants": [],
  "name": "Azure subscription 1",
  "state": "Enabled",
  "tenantDefaultDomain": "fodemicrosoft@gmail.onmicrosoft.com",
  "tenantDisplayName": "Default Directory",
  "tenantId": "5d0826be-c51f-4f77-9bb7-5e489c051998",
  "user": {
    "name": "fodemicrosoft@gmail.com",
    "type": "user"
  }
}
PS C:\Users\Fode\Desktop\fode-devops-infrastructure>
```

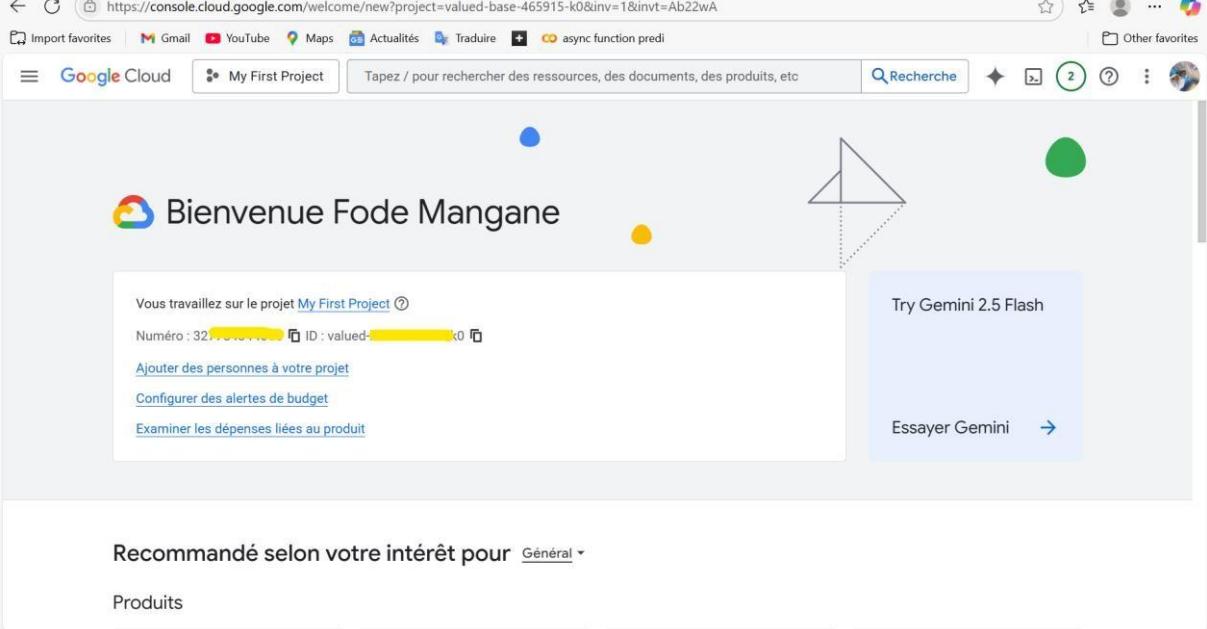
Ajoutons les secrets nécessaires à notre GitHub repository pour l'authentification Azure :

```
PS C:\Users\Fode\Desktop\fode-devops-infrastructure\terraform> az account show --query id -o tsv
9f0258a3-dfb0
PS C:\Users\Fode\Desktop\fode-devops-infrastructure\terraform> az account show --query tenantId -o tsv
5d0826be-c51f-4f77-9bb7-5e489c051998
PS C:\Users\Fode\Desktop\fode-devops-infrastructure\terraform>
PS C:\Users\Fode\Desktop\fode-devops-infrastructure\terraform> az ad sp create-for-rbac --name "github-terraform-sp" --role="Contributor" --scopes="/subscriptions/9f0258a3-dfb0-47cc-b1ed-c47d76fc9e6"
Creating 'Contributor' role assignment under scope '/subscriptions/9f0258a3-dfb0-47cc-b1ed-c47d76fc9e6'
The output includes credentials that you must protect. Be sure that you do not include these credentials in your code or check the credentials into your source control. For more information, see https://aka.ms/azadsp-cli
{
  "appId": "51a9fb9c-[REDACTED]-ca5f36a801a7",
  "displayName": "github-terraform-sp",
  "password": "F.28Q~cOMxI-[REDACTED]-trONBqb3r",
  "tenant": "5d0826be-c51f-4f77-9bb7-5e489c051998"
}
PS C:\Users\Fode\Desktop\fode-devops-infrastructure\terraform>
```

Repository secrets		New repository secret
Name	Last updated	
AWS_ACCESS_KEY_ID	3 weeks ago	 
AWS_ROLE_ARN	2 weeks ago	 
AWS_SECRET_ACCESS_KEY	3 weeks ago	 
AZURE_CLIENT_ID	2 minutes ago	 
AZURE_CLIENT_SECRET	2 minutes ago	 
AZURE_SUBSCRIPTION_ID	now	 
AZURE_TENANT_ID	1 minute ago	 
SSH_PRIVATE_KEY	3 weeks ago	 
SSH_PUBLIC_KEY	3 weeks ago	 

Configuration Google Cloud Platform

Pour GCP, créons d'abord un compte Google Cloud si ce n'est pas déjà fait :



The screenshot shows the Google Cloud Console homepage. At the top, it says "Bienvenue Fode Mangane". Below that, it displays the project information: "Vous travaillez sur le projet My First Project" and "Numéro : 321... ID : valued-base-465915-k0". There are also links to "Ajouter des personnes à votre projet", "Configurer des alertes de budget", and "Examiner les dépenses liées au produit". To the right, there's a "Try Gemini 2.5 Flash" button and a "Essayer Gemini" link. The bottom of the page has a "Recommandé selon votre intérêt pour" section with a "Général" dropdown and a "Produits" link.

Connectons nos identifiants avec Google Cloud CLI (prérequis : installer Google CLI) : gcloud auth login

```
● PS C:\Users\Fode\Desktop\fode-devops-infrastructure> gcloud auth login
Your browser has been opened to visit:
https://accounts.google.com/o/oauth2/auth?response_type=code&client_id=[REDACTED]&redirect_uri=http%3A%2F%2Flocalhost%3A8085%2F&scope=openid+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloud-platform+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fadmin+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fiam+https%3A%2F%2Fwww.googleapis.com%2Faccounts.reauth&state=18fxvUWb4...&access_type=offline&code_challenge=TkGBTFXA96oNoB8ECL6mAT05rtbyvInaw...&code_verifier=[REDACTED]
You are now logged in as [fodemangane@gmail.com].
Your current project is [fode-mangane]. You can change this setting by running:
$ gcloud config set project PROJECT_ID
○ PS C:\Users\Fode\Desktop\fode-devops-infrastructure>
```

Définissons et vérifions notre projet GCP : gcloud

config set project VOTRE_PROJECT_ID

```
● PS C:\Users\Fode\Desktop\fode-devops-infrastructure> gcloud config set project "valued-base-465915-k0"
Updated property [core/project].
● PS C:\Users\Fode\Desktop\fode-devops-infrastructure> gcloud config get-value project
val[REDACTED]
○ PS C:\Users\Fode\Desktop\fode-devops-infrastructure>
```

Activons les APIs nécessaires : gcloud services

enable compute.googleapis.com

```
PS C:\Users\Fode\Desktop\fode-devops-infrastructure> gcloud services enable compute.googleapis.com
Operation "operations/acf" finished successfully.
PS C:\Users\Fode\Desktop\fode-devops-infrastructure> gcloud services enable storage.googleapis.com
PS C:\Users\Fode\Desktop\fode-devops-infrastructure>
```

Créons le service account :

```
gcloud iam service-accounts create terraform-sa --display-name="Terraform Service Account"
```

```
● PS C:\Users\Fode\Desktop\fode-devops-infrastructure> gcloud iam service-accounts create fode-devops-sa --display-name="Fode DevOps Service Account" --description="Service account pour le deploiement automatise Fode-DevOps"
● Created service account [fode-devops-sa].
○ PS C:\Users\Fode\Desktop\fode-devops-infrastructure>
```

Attribuons les rôles nécessaires :

```

● PS C:\Users\Fode\Desktop\fode-devops-infrastructure> $PROJECT_ID = "valued-base-465915-k0"
● PS C:\Users\Fode\Desktop\fode-devops-infrastructure> gcloud projects add-iam-policy-binding $PROJECT_ID --member="serviceAccount:fode-devops-sa@$PROJECT_ID.iam.gserviceaccount.com" --role="roles/compute.instanceAdmin.v1"
Updated IAM policy for project [valued-base-465915-k0].
● bindings:
  - members:
    - serviceAccount:fode-devops-sa@valued-base-465915-k0.iam.gserviceaccount.com
      role: roles/compute.instanceAdmin.v1
  - members:
    - serviceAccount:service-327754344065@compute-system.iam.gserviceaccount.com
      role: roles/compute.serviceAgent
  - members:
    - serviceAccount:327754344065-compute@developer.gserviceaccount.com
    - serviceAccount:327754344065@cloudservices.gserviceaccount.com
      role: roles/editor
  - members:
    - user:fodemangane@gmail.com
      role: roles/owner
  etag: BwY6AGhUsAQ=
  version: 1
PS C:\Users\Fode\Desktop\fode-devops-infrastructure>
○ PS C:\Users\Fode\Desktop\fode-devops-infrastructure> gcloud projects add-iam-policy-binding $PROJECT_ID --member="serviceAccount:fode-devops-sa@$PROJECT_ID.iam.gserviceaccount.com" --role="roles/storage.admin"
Updated IAM policy for project [valued-base-465915-k0].
bindings:
  - members:
    - serviceAccount:fode-devops-sa@valued-base-465915-k0.iam.gserviceaccount.com
      role: roles/compute.instanceAdmin.v1
  - members:
    - serviceAccount:service-327754344065@compute-system.iam.gserviceaccount.com
      role: roles/compute.serviceAgent
  - members:
    - serviceAccount:327754344065-compute@developer.gserviceaccount.com
    - serviceAccount:327754344065@cloudservices.gserviceaccount.com
      role: roles/editor
  - members:
    - user:fodemangane@gmail.com
      role: roles/owner
  - members:
    - serviceAccount:fode-devops-sa@valued-base-465915-k0.iam.gserviceaccount.com
      role: roles/storage.admin
  etag: BwY6AGigd4E=
  version: 1
○ PS C:\Users\Fode\Desktop\fode-devops-infrastructure> gcloud projects add-iam-policy-binding $PROJECT_ID --member="serviceAccount:fode-devops-sa@$PROJECT_ID.iam.gserviceaccount.com" --role="roles/compute.networkAdmin"
● Updated IAM policy for project [valued-base-465915-k0].
bindings:
  - members:
    - serviceAccount:fode-devops-sa@valued-base-465915-k0.iam.gserviceaccount.com
      role: roles/compute.instanceAdmin.v1
  - members:
    - serviceAccount:fode-devops-sa@valued-base-465915-k0.iam.gserviceaccount.com
      role: roles/compute.networkAdmin
  - members:
    - serviceAccount:service-327754344065@compute-system.iam.gserviceaccount.com
      role: roles/compute.serviceAgent
  - members:
    - serviceAccount:327754344065-compute@developer.gserviceaccount.com
    - serviceAccount:327754344065@cloudservices.gserviceaccount.com
      role: roles/editor
  - members:
    - user:fodemangane@gmail.com
      role: roles/owner
  - members:
    - serviceAccount:fode-devops-sa@valued-base-465915-k0.iam.gserviceaccount.com
      role: roles/storage.admin
  etag: BwY6AGk79bQ=
  version: 1

```

Créons la clé JSON pour l'authentification : gcloud iam service-accounts keys create terraform-key.json \ --iam-account=terraformsa@VOTRE_PROJECT_ID.iam.gserviceaccount.com

```

● PS C:\Users\Fode\Desktop\fode-devops-infrastructure> gcloud iam service-accounts keys create "$env:USERPROFILE\fode-devops-gcp-key.json" --iam-account="fode-devops-sa@$PROJECT_ID.iam.gserviceaccount.com"
● created key [c913b94e-1000-4094-b094] of type [json] as [C:\Users\Fode\fode-devops-gcp-key.json] for [fode-devops-sa@valued-base-465915-k0.iam.gserviceaccount.com]
○ PS C:\Users\Fode\Desktop\fode-devops-infrastructure>

```

Visualisons le contenu de la clé :

fodemangane@gmail.com

32

```
PS C:\Users\Fode\Desktop\fode-devops-infrastructure> Get-Content "$env:USERPROFILE\fode-devops-gcp-key.json"
{
  "type": "service_account",
  "project_id": "val[REDACTED]0",
  "private_key_id": "c913b6[REDACTED]094",
  "private_key": "-----BEGIN PRIVATE KEY-----\nMI[REDACTED]IwYJKoZIhvcNAQcBQ[REDACTED]0MgQwGmh0dHA6LyQ[REDACTED]0Cj[REDACTED]0Iw[REDACTED]0OSS1S6ns3NGg2JS6fjTgJVktDFYWgRb+QV5oego60LG9tn2AsRD1Nr3uo\nnvw0NSHu3p7t8wprvJUTDQ1uwkfos6Rc2T9exS1xgno0+QbUM2vsJsP0nvvR/EI[REDACTED]nu/m1P/iS3YqbyCIU05CXFPioNgIkvQUPxMWeh/Y4rH7BP2DKeodBYGTmThGtp2\nns53basNpuTFDQva@6tCzonLc6horhl4T2vkf21MNhmkRa0DezvAemQi5nFosFn1T\nnXU57/4n+lVm2sZP4Mc1fq6\nxdvWZd++570t/gPe9FwKBgQDRfr6vc2uQAH+/7wIw5HPJY22QFcbX8yV\\niTMjAw61WrvGqWy8PFTgvcJjG/goBSC95FU9/i6197NkPRU9gNuBfFvt2Qmhie/nbm9DvH[REDACTED]b[REDACTED]\n7Gh5ITQ+Tjb[REDACTED]\nH+9lg5p8VraQvvTAoGRAMyI\nn+Tkr8+kNY660e1AsrxUCXIa13V8pAYwRA[REDACTED]77rD/Mkk1FPYiu[REDACTED]s6bHMChvDOKPDF\\n4Y7/AKMSIHkVja7qCcRDOf3E6ydmNIDeiSTVAm[REDACTED]\nAe6/FeeLoeaMpLI[REDACTED]veXad017Mj43K+9KOTBdzrvNSFU\\n18dwjvNNuQ7aGtQf+WeXN9PvxJsgFMBJfgE4tC0cPLwDy09m01hINSRRsqxt[REDACTED]4LG\\nFxdrJox+vx1T8LHTAxwgEE\\n----END PRIVATE KEY-----\\n",
  "client_email": "fode-devops-sa@[REDACTED].iam.gserviceaccount.com",
  "client_id": "103[REDACTED]148962",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://[REDACTED]apis.com/token",
  "auth_provider_x509_cert_url": "https://[REDACTED]2/v1/certs",
  "client_x509_cert_url": "https://www[REDACTED].iam.gserviceaccount.com",
  "universe_domain": "googleapis.com"
}
PS C:\Users\Fode\Desktop\fode-devops-infrastructure>
```

Important : Sélectionnez TOUT le contenu depuis la première { jusqu'à la dernière } et copiez-le dans GitHub Secrets en tant que GOOGLE_CREDENTIALS.

Supprimons le fichier local pour des raisons de sécurité :

```
PS C:\Users\Fode\Desktop\fode-devops-infrastructure> Remove-Item "$env:USERPROFILE\fode-devops-gcp-key.json"
PS C:\Users\Fode\Desktop\fode-devops-infrastructure>
```

Déploiement Multi-Cloud Automatisé

Maintenant, ajoutons les secrets Azure et GCP à notre repository GitHub :

Pour **Azure**, ajoutez ces secrets :

- AZURE_CLIENT_ID
- AZURE_CLIENT_SECRET
- AZURE_SUBSCRIPTION_ID
- AZURE_TENANT_ID

Pour **GCP**, ajoutez :

- GOOGLE_CREDENTIALS (le contenu JSON complet)
- GCP_PROJECT_ID

The screenshot shows the GitHub Secrets and variables page. On the left, there's a sidebar with 'Security' (Advanced Security, Deploy keys, Secrets and variables), 'Actions', 'Codespaces', 'Dependabot', and 'Integrations' (GitHub Apps, Email notifications). The main area lists secrets with their names, last updated time, and edit/delete icons. Some secrets are highlighted with colored boxes: AWS_ACCESS_KEY_ID (yellow), AWS_ROLE_ARN (yellow), AWS_SECRET_ACCESS_KEY (yellow), AZURE_CLIENT_ID (pink), AZURE_CLIENT_SECRET (pink), AZURE_SUBSCRIPTION_ID (pink), AZURE_TENANT_ID (pink), GCP_PROJECT_ID (purple), GOOGLE_CREDENTIALS (purple), SSH_PRIVATE_KEY (light blue), and SSH_PUBLIC_KEY (light blue).

Name	Last updated
AWS_ACCESS_KEY_ID	3 weeks ago
AWS_ROLE_ARN	3 weeks ago
AWS_SECRET_ACCESS_KEY	3 weeks ago
AZURE_CLIENT_ID	3 days ago
AZURE_CLIENT_SECRET	3 days ago
AZURE_SUBSCRIPTION_ID	3 days ago
AZURE_TENANT_ID	3 days ago
GCP_PROJECT_ID	2 days ago
GOOGLE_CREDENTIALS	2 days ago
SSH_PRIVATE_KEY	3 weeks ago
SSH_PUBLIC_KEY	3 weeks ago

Une fois tous les secrets configurés, relançons le workflow. Cette fois, GitHub Actions déployera automatiquement sur les trois plateformes simultanément : git add .

```
git commit -m "Add multi-cloud support for Azure and GCP"
git push origin main
```

Observons le déploiement multi-cloud en action :

The screenshot shows the GitHub Actions run details for a 'Terraform Plan' job. The left sidebar lists other jobs like 'Corriger Backend', 'Créer Backend S3', 'Vérifier Backend', 'Nettoyage Azure Basic IPs ...', 'Terraform Destroy', 'Validation et Sécurité', 'Terraform Plan' (which is selected and highlighted with a red arrow), 'Terraform Apply', 'Déploiement Ansible Multi...', and 'Tests Post-Déploiement Mu...'. The right pane shows the logs for the 'Terraform Plan' job, which succeeded 2 days ago in 35s. The logs list several steps: Checkout code, Setup Terraform, Setup SSH Key, Configure AWS Credentials, Configure Azure Credentials, Configure GCP Credentials, Terraform Init, Terraform Plan, Save Terraform Plan, Post Configure AWS Credentials, Post Checkout code, and Complete job. The 'Configure AWS Credentials', 'Configure Azure Credentials', and 'Configure GCP Credentials' steps are highlighted with a red box.

<https://github.com/FodeMangane/fode-devops-infrastructure/actions/runs/16325053090/job/46112958575#logs>

```

    ➜ Terraform Apply
    succeeded 1 minute ago in 6m 31s

    └── Export Terraform Outputs
        └── Pre-Ansible Connectivity Test (AWS)
            1 ► Run PUBLIC_IP="34.245.157.25"
            34 ✓ Test de connectivité pré-Ansible vers 34.245.157.25 (AWS)
            35 ⚠ Ping réseau échoué (peut être normal si ICMP bloqué)
            36 ✅ Port SSH 22 accessible

        └── Pre-Ansible Connectivity Test (Azure)
            1 ► Run AZURE_PUBLIC_IP="20.169.162.46"
            34 ✓ Test de connectivité pré-Ansible vers 20.169.162.46 (Azure)
            35 ⚠ Ping réseau échoué (peut être normal si ICMP bloqué)
            36 ✅ Port SSH 22 accessible

        └── Pre-Ansible Connectivity Test (GCP)
            1 ► Run GCP_PUBLIC_IP="104.197.154.25"
            34 ✓ Test de connectivité pré-Ansible vers 104.197.154.25 (GCP)
            35 ⚠ Ping réseau échoué (peut être normal si ICMP bloqué)
            36 ✅ Port SSH 22 accessible

    └── Create Deployment Summary
        1 ► Run ENVIRONMENT="prod"

```

<https://github.com/FodeMangane/fode-devops-infrastructure/actions/runs/16325053090/job/46112958575#logs>

```

    ➜ Tests Post-Déploiement Multi-Cloud
    Started 4m 28s ago

    └── Test Load Balancer (AWS)
        1 ▼ Run AZURE_PUBLIC_IP="20.172.133.178"
        2 AZURE_PUBLIC_IP="20.172.133.178"
        3 TEST_URL="http://$AZURE_PUBLIC_IP"
        4 echo "⏳ Attente du démarrage du service web (Azure)..."
        5 sleep 180
        6 echo "🌐 Test de connectivité HTTP vers $TEST_URL..."
        7 for i in {1..15}; do
        8     if curl -f -s "$TEST_URL/ansible-test.html" | grep -q "Fode-DevOps"; then
        9         echo "✅ Service web Azure accessible et contenu correct détecté!"
        10        break
        11    else
        12        echo "⚠ Tentative $i/15..."
        13        sleep 20
        14    fi
        15    if [ $i -eq 15 ]; then
        16        echo "✗ Service web Azure non accessible après 5 minutes"
        17        echo "⚠ Contenu reçu:"
        18        curl -s "$TEST_URL/ansible-test.html" || echo "Aucun contenu"
        19        exit 1
        20    fi
        21 done
        22 shell: /usr/bin/bash -e {0}

```

État final attendu :

The screenshot shows a GitHub Actions workflow summary for a push event. The status is Success, and the total duration is 15m 57s. There is 1 artifact. The workflow file is `deploy.yml`, triggered on push. The workflow consists of several steps: Corriger Backend, Créer Backend S3, Vérifier Backend, Nettoyage Azure Basic IPs..., Terraform Destroy, Validation et Sécurité, Terraform Plan, Terraform Apply, Déploiement Ansible Multi..., and Tests Post-Déploiement Multi... . All steps are marked as completed with green checkmarks.

Le workflow s'exécute maintenant sur les trois providers en parallèle, créant une infrastructure complète et cohérente :

Test Pour Azure :

The screenshot shows a web browser displaying a landing page for the Fode-DevOps project. The page is titled "Fode-DevOps" and features a rocket ship icon. The text on the page includes: "Bienvenue sur l'infrastructure Fode-DevOps !", "Ce serveur Nginx a été configuré automatiquement par Ansible.", "Serveur : web-server-prod-azure", "OS : Ubuntu 22.04", "Cloud : Azure", and "Déployé le : 2025-07-16". Below the main content, it says "Projet Fode-DevOps - Multi-Cloud Infrastructure". The URL in the browser bar is `20.172.133.178`.

Dans la console Azure Cloud

The screenshot shows the Microsoft Azure portal's 'Toutes les ressources' (All resources) page. At the top, there are navigation links for Import favorites, Gmail, YouTube, Maps, Actualités, Traduire, and async function predi. The main title is 'Microsoft Azure' with a 'Mettre à niveau' button. A search bar says 'Rechercher dans les ressources, services et documents (G)'. On the right, there are icons for Copilot, settings, and user information (fodemicrosoft@gmail.com). Below the header, it says 'Default Directory (fodemicrosoft@gmail.onmicrosoft.com)'.

The main content area has a toolbar with 'Créer', 'Gérer l'affichage', 'Actualiser', 'Exporter au format CSV', 'Ouvrez une requête', 'Attribuer des balises', 'Supprimer', and a 'Aucun regroupement' dropdown. A message at the top says 'Vous affichez une nouvelle version de l'expérience de navigation. Cliquez ici pour accéder à l'ancienne expérience.' Below this are several filter buttons: 'Filtrer un champ...', 'Abonnement est égal à tout', 'Groupe de ressources est égal à tout', 'Type est égal à tout', 'Emplacement est égal à tout', and '+ Ajouter un filtre'.

The main table lists resources with columns: Nom (Name), Type, Groupe de ressources (Resource group), Emplacement (Location), and Abonnement (Subscription). The table contains the following data:

Nom	Type	Groupe de ressources	Emplacement	Abonnement
fode-devops-vm-nsg	Groupe de sécurité réseau	fode-devops-rg	West US 2	Azure subscription 1
fode-devops-vm-public-ip	Adresse IP publique	fode-devops-rg	West US 2	Azure subscription 1
fode-devops-vm-vnet	Réseau virtuel	fode-devops-rg	West US 2	Azure subscription 1
fodevdevopsprodstorage	Compte de stockage	fode-devops-rg	West US 2	Azure subscription 1
NetworkWatcher_eastus	Network Watcher	NetworkWatcherRG	East US	Azure subscription 1
NetworkWatcher_westeurope	Network Watcher	NetworkWatcherRG	West Europe	Azure subscription 1
NetworkWatcher_westus2	Network Watcher	NetworkWatcherRG	West US 2	Azure subscription 1

At the bottom left, it says 'Affichage de 1 – 7 sur 7. Afficher le nombre : auto'. At the bottom right, there is a link 'Envoyer des commentaires'.

Et nous venons de mettre en place une infrastructure multi-provider incluant AWS, Azure et GCP.

Remarque : comme je l'ai précédemment mentionné, le changement d'adresse IP est dû au fait que, lorsque je rencontre une erreur, je détruis l'infrastructure existante. À chaque destruction suivie d'un git add, une nouvelle infrastructure est automatiquement recréée, ce qui génère de nouvelles adresses IP.

7. Résultats et optimisation

7.1 Analyse des performances

Votre infrastructure multi-cloud est maintenant opérationnelle et automatisée sur AWS, Azure et Google Cloud Platform simultanément. Le déploiement complet, qui prendrait des semaines à une équipe traditionnelle pour un seul provider, s'exécute désormais en moins de 15 minutes sur les trois plateformes principales.

Cette transformation représente un gain de productivité révolutionnaire à l'échelle de l'industrie. Là où des équipes entières passaient des semaines à configurer manuellement des environnements sur un seul cloud, avec des risques d'erreurs humaines constants et des incohérences entre providers, vous obtenez maintenant des déploiements parfaitement reproductibles, uniformes et fiables sur l'ensemble de l'écosystème cloud mondial.

Le fait de maîtriser le déploiement simultané sur AWS (leader du marché), Azure (écosystème Microsoft) et GCP (innovation Google) vous positionne parmi l'élite des professionnels cloud. Cette expertise tri-cloud vous donne une flexibilité stratégique unique, permettant d'optimiser les coûts, d'éviter le vendor lock-in et de tirer parti des forces spécifiques de chaque plateforme.

7.2 Bonnes pratiques

La mise en place d'une infrastructure multi-cloud automatisée répond aujourd'hui à des exigences opérationnelles concrètes et stratégiques. Dans un contexte où la complexité des environnements cloud augmente, les entreprises recherchent activement des solutions capables d'assurer cohérence, fiabilité et agilité sur plusieurs plateformes à la fois.

L'approche Infrastructure as Code multi-provider, exploitée tout au long de ce projet, incarne pleinement cette modernisation. Elle permet de garantir :

- **Une reproductibilité inter-cloud** : des déploiements standardisés sur tous les providers ;
- **Scalabilité universelle** : des ressources adaptables à tous les contextes d'évolution ;
- **Fiabilité renforcée** : réduction des erreurs humaines grâce à l'automatisation complète ;
- **Efficacité maximale** : optimisation du temps et des coûts de gestion ;
- **Flexibilité stratégique** : adaptation dynamique aux besoins métier sur n'importe quelle plateforme.

Cette expertise vous distingue non seulement des professionnels mono-cloud, mais aussi de ceux qui travaillent encore avec des approches manuelles. Vous représentez maintenant l'évolution naturelle de l'expertise IT vers l'automatisation intelligente et la gestion multiprovider.

7.3 Évolution vers l'entreprise

Cette compétence multi-cloud vous distingue immédiatement comme un professionnel d'élite dans l'écosystème IT mondial. Pendant que la majorité des équipes IT perdent encore du temps sur des tâches manuelles répétitives sur un seul provider, vous maîtrisez les outils qui révolutionnent l'industrie sur l'ensemble du marché cloud.

Les entreprises modernes investissent massivement dans les stratégies multi-cloud pour éviter la dépendance à un seul fournisseur et optimiser leurs coûts. Votre expertise en Terraform multiprovider, Ansible cross-platform et GitHub Actions vous positionne comme un acteur clé de cette transformation. Ces compétences sont particulièrement valorisées car elles permettent aux entreprises de :

- Négocier avec plusieurs fournisseurs cloud pour optimiser les coûts
- Implémenter des stratégies de disaster recovery cross-cloud
- Respecter les exigences de souveraineté des données selon les régions
- Tirer parti des services spécialisés de chaque provider

Ces compétences s'inscrivent dans une dynamique de **haute spécialisation**, correspondant aux profils actuellement les plus sollicités dans l'industrie IT : Multi-Cloud Architect, DevOps Engineer, Cloud Infrastructure Specialist ou Platform Engineering Lead.

Au-delà de la technicité, ce type de démarche incarne une **vision stratégique globale**, orientée vers l'efficacité, la fiabilité et la résilience. La capacité à automatiser, orchestrer et piloter ces environnements dans leur ensemble est aujourd'hui un **levier de compétitivité majeur** pour les organisations.

Plus qu'une simple compétence technique, cette maîtrise démontre votre capacité à penser de manière stratégique à l'échelle globale et à apporter une valeur ajoutée immédiate à toute organisation cherchant à moderniser son infrastructure. Les entreprises valorisent particulièrement les profils capables de transformer des processus manuels chronophages en solutions automatisées élégantes, évolutives et économiquement optimisées.

Cette expertise vous positionne comme un élément clé de la transformation digitale des entreprises, capable de concevoir et déployer des infrastructures modernes qui s'adaptent aux besoins changeants du marché tout en maximisant l'efficacité opérationnelle et en minimisant les risques de dépendance technologique.

Conclusion

Au terme de ce projet révolutionnaire, vous venez de franchir une étape déterminante qui vous place parmi l'élite des professionnels de l'infrastructure cloud mondiale. Vous avez non seulement découvert les concepts de l'automatisation d'infrastructure, mais vous avez également mis en pratique les outils les plus puissants et actuels de l'écosystème DevOps sur l'ensemble des plateformes cloud majeures : **Terraform**, **Ansible** et **GitHub Actions** déployés simultanément sur **AWS**, **Azure** et **Google Cloud Platform**.

Grâce à **Terraform multi-provider**, vous avez appris à déployer automatiquement une infrastructure Cloud complète sur les trois principales plateformes, incluant la création de réseaux virtuels, d'instances de calcul et d'espaces de stockage, en respectant les bonnes pratiques de sécurité spécifiques à chaque fournisseur.

Ansible vous a permis de transformer ces infrastructures brutes en environnements fonctionnels et configurés de manière uniforme, en automatisant l'installation des logiciels et la configuration des services sur tous les providers.

Enfin, avec **GitHub Actions**, vous avez intégré un pipeline CI/CD qui orchestre l'ensemble des opérations de déploiement et de configuration multi-cloud de manière automatique et contrôlée depuis un simple commit.

Ce projet vous aura ainsi permis de transformer des processus manuels et fastidieux en solutions automatisées, fiables et reproductibles sur l'ensemble de l'écosystème cloud, réduisant le temps de déploiement multi-provider de plusieurs semaines à moins de 15 minutes. Il illustre parfaitement la puissance et l'importance des approches Infrastructure as Code (IaC) et DevOps dans les environnements multi-cloud modernes.

Ce projet marque ainsi le début d'une nouvelle étape dans votre carrière : celle d'un professionnel d'élite capable d'apporter une réelle valeur stratégique aux organisations en concevant, déployant et automatisant des infrastructures Cloud performantes, évolutives, sécurisées et économiquement optimisées sur l'ensemble de l'écosystème cloud mondial.

Important : Comme mentionné en introduction, n'oubliez pas de détruire entièrement toutes vos ressources de test à l'aide de terraform destroy, puis de vérifier manuellement dans les consoles des trois plateformes (AWS, Azure et GCP) que plus aucune ressource active ne subsiste. Cela vous permettra d'éviter toute facturation inattendue.

Remerciements

Je tiens à exprimer ma profonde gratitude à Monsieur **Massamba Lo** pour son accompagnement et son engagement tout au long de ce projet révolutionnaire. C'est grâce à lui que j'ai eu l'opportunité de découvrir et de maîtriser ces technologies d'automatisation d'infrastructure multi-cloud, qui constituent aujourd'hui un atout majeur et différenciant dans mon parcours professionnel. Son sens du partage, sa pédagogie exceptionnelle et sa passion pour l'innovation cloud ont été des sources précieuses d'inspiration et de motivation pour atteindre ce niveau d'expertise. Qu'il trouve ici l'expression de ma reconnaissance la plus sincère pour m'avoir guidé vers la maîtrise de l'écosystème cloud global.