

TP : Déploiement sécurisé de l'application Todo sur Azure avec Terraform, PRA et DNS

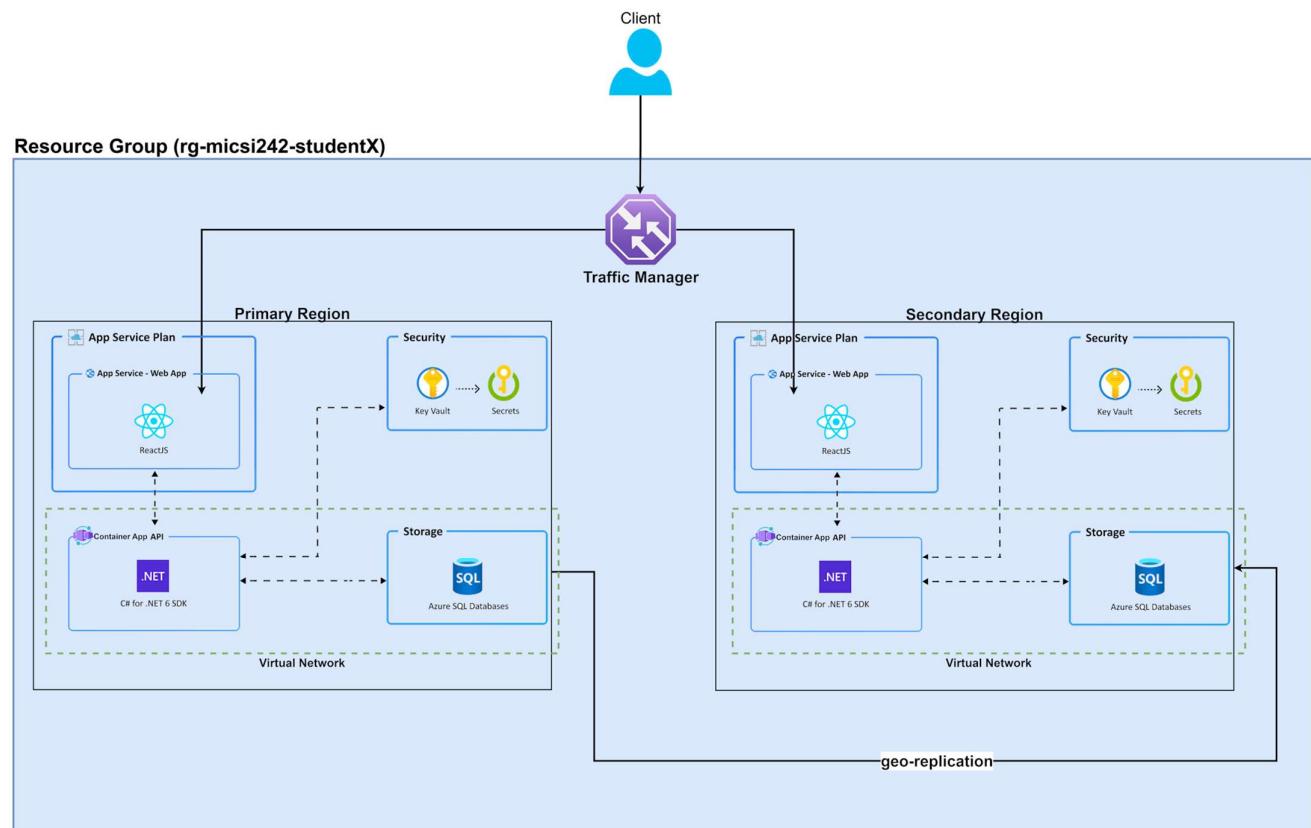
Objectif du TP

Dans ce TP, vous allez déployer une application Todo composée d'un frontend en React, d'une API en C# et d'une base de données MSSQL sur Azure, en mettant en place les bonnes pratiques de sécurité, haute disponibilité et reprise d'activité (PRA).

Vous utiliserez **Terraform** pour automatiser l'infrastructure et vous vous appuierez sur **deux régions Azure distinctes** afin d'assurer la résilience du système en cas de panne.

À la fin de ce TP, vous aurez mis en place une **architecture cloud sécurisée et résiliente**, avec une gestion fine des accès réseau et du routage du trafic.

Architecture cible



L'application repose sur les composants suivants :

Sécurisation et réseau

- Un réseau privé (VNet) pour isoler l'API et la base de données.
- Des Private Endpoints pour interdire l'accès public à l'API et à la base MSSQL.
- Azure Key Vault pour la gestion des secrets.

Déploiement des services applicatifs

- Un frontend React**, exposé publiquement, déployé sur **Azure App Service**.
- Une API C# déployée sur Azure Container Apps inaccessible depuis Internet**.
- Une base de données MSSQL inaccessible depuis Internet, répliquée** sur une **seconde région** pour garantir la continuité de service.

Plan de Reprise d'Activité (PRA)

- Active Geo-Replication** sur la base MSSQL pour assurer la disponibilité des données en cas de panne.
- Azure Traffic Manager** pour rediriger automatiquement le trafic vers un second environnement en cas de défaillance du frontend principal.

Gestion des accès et DNS

- Une zone DNS publique Azure** vous sera fournie pour exposer l'application sous une **URL spécifique** (students.mfolabs.me).
- Un CNAME configuré** pour associer le nom de domaine au Traffic Manager.
- Un certificat SSL** géré automatiquement par Azure pour sécuriser les connexions.

Monitoring et supervision

- Azure Monitor et Log Analytics** pour surveiller l'état de l'infrastructure.
- Application Insights** pour suivre les performances et les erreurs de l'application.

Ce que vous devrez faire

- Automatiser le déploiement de l'**infrastructure** avec Terraform.
 - Configurer la **réplication des données** entre deux régions Azure.
 - Restreindre l'accès réseau pour ne pas exposer l'API ni la base MSSQL.
 - Configurer Azure Traffic Manager pour gérer le routage du frontend.
 - Associer l'application à une URL personnalisée via la zone DNS publique fournie.
 - Tester la bascule automatique en simulant une panne.
-

Livrables attendus

- Code Terraform fonctionnel**
 - Captures des tests de PRA et de sécurité**
 - Rapport expliquant vos choix techniques**
-

Durée du TP

- Matin (3h) : Déploiement de l'infrastructure et configuration.**
 - Après-midi (3h) : Déploiement de l'application, test du PRA, DNS et validation.**
-

Ce TP vous permettra de comprendre comment **déployer une application sécurisée et résiliente sur Azure**, en combinant **Infrastructure as Code, sécurité, PRA et DNS** dans un environnement cloud. 

Documentations

Terraform App Service Linux :

https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/linux_web_app

Terraform App Service VNet Integration :

https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/app_service_virtual_network_swift_connection

Terraform Container App :

https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/container_app

Terraform Key Vault : https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/key_vault

Terraform Key Vault Secrets :

https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/key_vault_secret

Terraform MSSQL Database :

https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/mssql_database

Terraform MSSQL Failover Group :

https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/mssql_failover_group

MS sur la géo-réPLICATION asynchrone pour MSSQL : <https://learn.microsoft.com/en-us/azure/azure-sql/database/active-geo-replication-overview?view=azuresql&tabs=tsql>

Terraform Traffic Manager :

https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/traffic_manager_profile

Terraform Traffic Manager Endpoint :

https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/traffic_manager_azure_endpoint

Terraform Application Insights :

https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/application_insights#example-usage---workspace-mode

Variables d'environnements

React Web App (Linux App Service)

VITE_API_BASE_URL = défini lors du provisioning de l'infra

VITE_APPLICATIONINSIGHTS_CONNECTION_STRING (optionnel) = défini lors du provisioning de l'infra si Application Insights a été utilisé

C# API (Container App)

AZURE_KEY_VAULT_ENDPOINT = défini lors du provisioning de l'infra

AZURE_SQL_CONNECTION_STRING_KEY = doit pointer sur le secret **connectionStringKey** dans le Key Vault

Key Vault Secrets

Les secrets suivants devront être créés dans le Key Vault pour que l'application puisse fonctionner.

sqlAdminPassword = défini lors du provisioning de l'infra

appUserPassword = défini lors du provisioning de l'infra

```
connectionStringKey = 'Server=${mssql_fqdn}; Database=${database_name}; User=${appUserPassword}; Password=${sqlAdminPassword}'
```

Authentification

Activer l'**Identity** (SystemAssigned) de la Container App pour lui permettre d'accéder au Key Vault en lui affectant le rôle **Key Vault Secrets User**.

Pensez à vous affecter le rôle **Key Vault Secrets Officer** pour pouvoir créer les secrets automatiquement via Terraform avec votre identifiant studentX@mfolabs.me.

Images Docker

React Web App : acrmfolabsmicsi242.azurecr.io/csharp_api:latest

C# API : acrmfolabsmicsi242.azurecr.io/react_app:latest