TP — Déployer une application web 100 % serverless

Étape 1 — Déploiement de l'API REST en mode serverless

1. **Créer une API Flask minimale**

   Développer une application Flask simple contenant une seule route REST (exemple : `/hello`) qui renvoie un message JSON.

   L'application doit pouvoir être exécutée de manière autonome, écouter sur toutes les interfaces et utiliser un port standard.

   Elle servira de base au conteneur déployé sur le cloud.

2. **Préparer un conteneur de l'application**

   Créer une image conteneur à partir du code de l'API.

   Cette image doit contenir :

   - le code source Python ;
   - les dépendances ;
   - un point d'entrée permettant de lancer l'application.

     L'image sera utilisée pour le déploiement sur le cloud.

3. **Publier l'image sur un registre de conteneurs**

   Chaque plateforme cloud dispose de son propre registre :

   - **GCP :** Artifact Registry
   - **Azure :** Azure Container Apps

4. **Vérifier le fonctionnement**

   Une fois le déploiement terminé, récupérer l'URL publique générée par la plateforme.

   Tester le point d'accès principal de l'API (exemple : `/hello`) pour vérifier la réponse attendue.

Étape 2 — Exposer l'API via une passerelle publique

1. **Définir l'interface de l'API**
    - Décrire les routes et méthodes (ex. `/hello` en `GET`) dans une spécification d'API (OpenAPI).
2. **Créer la passerelle et lier le backend**
    - **GCP :** créer une API, importer la spec, créer un déploiement de API gateway pointant vers l'URL du service **Cloud Run**.
    - **Azure :** créer une instance API managment , créer une API et configurer le **backend** vers l'URL publique de **Container Apps**.
3. **Configurer l'accès public**
    - **GCP :** laisser l'API Gateway accessible publiquement
    - **Azure :** publier l'API et autoriser l'accès anonyme.
4. **Tester l'URL de la passerelle**

Étape 3 — Héberger un site web statique simple sur un service de stockage

Création du site statique

1. Créer un petit site composé d'un fichier `index.html` et d'un fichier JavaScript séparé.
2. Le site doit :
    - afficher un contenu basique (titre, texte, bouton, etc.) ;
    - appeler l'API exposée précédemment (exemple : `/hello`) via une requête HTTP;
    - afficher la réponse dans la page.

Publication du site

1. **Créer un service de stockage configuré pour l'hébergement statique :**
   - **GCP :** utiliser Cloud Storage ;
   - **Azure :** utiliser Azure Storage – Static Website ;
2. **Téléverser les fichiers du site :**
   - Ajouter `index.html`, les fichiers JS et CSS au service de stockage.
3. **Tester l'affichage et l'appel à l'API :**
   - Vérifier que le site s'affiche correctement.
   - Tester le bouton ou la requête JavaScript déclenchant l'appel API.

Étape 4 — Distribuer le site via un CDN global en HTTPS

Mise en place

1. Créer la distribution CDN
   - GCP : créer un Load Balancer HTTP(S) global avec un **backend bucket** et activer **Cloud CDN**.
     - Load Balancing HTTP(S) : https://cloud.google.com/load-balancing/docs/https
     - Activer Cloud CDN : https://cloud.google.com/cdn/docs/setting-up-cdn-with-bucket
   - Azure : créer un profil **Azure Front Door**, ajouter l'origine de votre compte de stockage et publier l'endpoint.
     - Vue d'ensemble : https://learn.microsoft.com/en-us/azure/frontdoor/standard-premium/overview
     - Créer une instance : https://learn.microsoft.com/en-us/azure/frontdoor/standard-premium/create-front-door-portal
2. HTTPS et nom de domaine
   - GCP : utiliser un **certificat géré** sur le Load Balancer et lier un domaine (CNAME/A) DNS.

    - Certificats gérés : https://cloud.google.com/load-balancing/docs/ssl-certificates/google-managed-certs
- Azure : ajouter un **custom domain** sur Front Door.
    - Domaine personnalisé : https://learn.microsoft.com/en-us/azure/frontdoor/standard-premium/how-to-add-custom-domain

3. Routage du trafic API via le même point d'accès
    - Azure : ajouter une seconde origin vers **API Management** et créer une route basée sur le chemin (`/api/*`).
        - Configurer des routes : https://learn.microsoft.com/en-us/azure/frontdoor/standard-premium/how-to-configure-route
    - GCP : publier également l'API via le même Load Balancer avec un **Serverless NEG** pointant sur **Cloud Run**, et router `/api/*` vers ce backend.
        - Concepts Serverless NEG : https://cloud.google.com/load-balancing/docs/negs/serverless-neg-concepts
4. Tester et valider