**🚀 Déploiement d'un projet Angular + ASP.NET Web API + SQLite sur Azure**

**🌐 Objectif**

Mettre en ligne un projet fullstack avec :

* Frontend : Angular
* Backend : ASP.NET Web API (.NET 6+)
* Base de données : SQLite (locale dans l'API)
* Nom de domaine personnalisé : JFDidgeridoo.ca

**✅ Prérequis**

* Node.js géré avec NVM (Windows) et Angular CLI installé localement
* Projet avec structure :
* /mon-projet
* /frontend # Angular
* /backend # ASP.NET Web API
* Compte Azure actif (avec crédit gratuit ou payant)
* Domaine acheté chez PlanetHoster, WHC ou autre

**1. 📄 Préparer le projet**

**Backend : ASP.NET Web API avec SQLite**

* Utilise Microsoft.Data.Sqlite dans le projet
* Placer la base de données dans un dossier : Data/ma-base.sqlite
* Utiliser un chemin relatif pour pointer vers la DB

var dbPath = Path.Combine(Directory.GetCurrentDirectory(), "Data/ma-base.sqlite");

**Frontend : Angular**

* Code Angular dans /frontend
* Angular CLI installé localement via :

npm install @angular/cli@16 --save-dev

**2. 🌐 Créer les ressources Azure**

**Backend : Azure App Service**

1. Aller sur [portal.azure.com](https://portal.azure.com/)
2. Créer une ressource > App Service
   * Nom : jf-api-backend
   * OS : Windows ou Linux
   * Runtime : .NET 6 (ou 7)
   * Plan : Basic B1 (ou Free F1 pour test)

**Frontend : Azure Static Web App**

1. Créer une ressource > Static Web App
   * Nom : jf-frontend
   * Lier à ton repo GitHub si possible
   * Build : npm run build
   * Output : dist/nom-projet

**3. 🚚 Déploiement**

**Backend**

1. Dans /backend, publier en Release

dotnet publish -c Release -o ./publish

1. Aller dans App Service > Déploiement FTP ou Zip
2. Envoyer le contenu du dossier /publish
3. Configurer le chemin d’accès à la DB (SQLite)

**Frontend**

1. Dans /frontend, build Angular

npx ng build --configuration production

1. Pour App Service : uploader le contenu de dist/ dans un App Service
2. Pour Static Web App : push sur GitHub (déploiement auto)

**4. 🌐 Ajouter le domaine JFDidgeridoo.ca**

1. Aller dans App Service ou Static Web App > Custom Domains
2. Ajouter www.jfdidgeridoo.ca
3. Azure fournit un enregistrement **CNAME** à ajouter chez ton registraire DNS
4. Une fois propagé, Azure gère le certificat SSL automatiquement

**5. 🔐 Configuration API (CORS, HTTPS)**

Dans Program.cs ou Startup.cs de l’API :

builder.Services.AddCors(options => {

options.AddPolicy("AllowFrontend", builder =>

builder.WithOrigins("https://www.jfdidgeridoo.ca")

.AllowAnyHeader()

.AllowAnyMethod());

});

app.UseCors("AllowFrontend");

**6. ⚖️ Conseils & bonnes pratiques**

* Teste tout localement avant de déployer
* Utilise environment.ts pour pointer vers l’API
* Active les alertes de coûts dans Azure pour ne pas dépasser ton budget
* SQLite est parfait pour un projet perso; prévois une migration vers SQL Server si besoin futur

**🌟 Résultat final**

* Frontend Angular déployé
* API ASP.NET en ligne avec SQLite locale
* Site accessible en HTTPS via JFDidgeridoo.ca

Tu peux ensuite ajouter d’autres services (monitoring, sauvegardes, CI/CD, etc.) N’hésite pas à demander si tu veux un diagramme visuel ou un fichier de configuration type (web.config, GitHub Actions, etc.)