**CMAP Azure Cloud Deployment Options**

The Best Deployment Strategy for CMAP as per current scenario is choosing Option 1 for all frontend, backend and database.

**Frontend:**

Option 1: Using Azure Static Web App

1. Make a repository on Azure Repos
2. Push the frontend code to the repository
3. Create an Azure Static Web App
4. Choose the environment as Custom
5. Select the frontend Azure Repo and branch
6. Set output as ‘dist’ for Vite
7. Choose a plan and hit save

This creates an automatic CI/CD pipeline for the frontend

Option 2: Using Docker Container

1. Create a frontend docker image using dockerfile
2. Push the image to Azure Container Registry
3. Create Azure Container App
4. Link it to the container image
5. Set environment variables and secret

Option 3: Using Virtual Machine

1. Create a Virtual Machine
2. Remote into the VM, and setup the react environment
3. Install git to pull the latest code from prod branch
4. Deploy the code on IIS to serve it.

**Backend:**

Option 1: Azure App Service

1. Push Node.js app to Azure Repos
2. Create Azure App Service
3. Choose Runtime: Node.js LTS and choose the repository if Azure Repos option is available.
4. If Azure Repos is not available go to the Repo and build automatic Node Js with Linux build and deploy pipeline while choosing the App service you just created in the yaml
5. Add environment variables in App Settings (like MONGO\_URI)
6. Enable Logging

This creates an automatic CI/CD pipeline for the backend

Option 2: Using Docker Container

1. Create a backend docker image using dockerfile
2. Push the image to Azure Container Registry
3. Create Azure Container App
4. Link it to the container image
5. Set environment variables and secret

Option 3: Using Virtual Machine

1. Create a Virtual Machine
2. Remote into the VM, and setup the Node.js environment
3. Install git to pull the latest code from prod branch
4. Deploy the code on IIS to serve it.

**Database:**

Option 1: Using Azure Cosmos DB

1. Create Cosmos DB with MongoDB API
2. Choose a region close to your App Service
3. Get Connection String
4. Use Primary Connection String
5. Add to your API MONGO\_URI environment variable

Option 2: Using Docker

1. Create a dockerfile with mongodb environment
2. Add your db import location
3. Set username and password
4. Push it to Azure Container Registry

Option 3: Using Virtual Machine

1. Create a Virtual Machine
2. Remote into the VM, and setup the MongoDB environment
3. Install git to pull the latest database dump from prod branch
4. Deploy the code on IIS to serve it or use MongoDB Cloud.

**Summary:**

Frontend:

| **Option** | **Description** | **Pros** | **Cons** | **Rank** |
| --- | --- | --- | --- | --- |
| 1 | Azure Static Web App | * CI/CD auto-setup * Global CDN * Free SSL * Simple config | * Limited server-side logic * Needs repo integration | 1 |
| 2 | Docker Container | * Full control * Good for microservices * Can run anywhere | * Requires custom CI/CD * No CDN by default * Slightly more costly | 2 |
| 3 | Virtual Machine | * Full manual control * Good for legacy or stateful apps | * No scaling * No CI/CD * High maintenance effort | 3 |

Backend:

| **Option** | **Description** | **Pros** | **Cons** | **Rank** |
| --- | --- | --- | --- | --- |
| 1 | Azure App Service | * Easy CI/CD * App Settings for env vars * Scales automatically | * Less control over runtime * Some limitations in advanced configs | 1 |
| 2 | Docker Container | * Full runtime control * Good for microservices | * Slightly more complex setup * Logging and monitoring must be set up | 2 |
| 3 | Virtual Machine | * Complete flexibility | * No auto-deploy * Manual scaling * High admin burden | 3 |

Database:

| **Option** | **Description** | **Pros** | **Cons** | **Rank** |
| --- | --- | --- | --- | --- |
| 1 | Azure Cosmos DB | * Fully managed * Global replication * Auto-scaling | * More expensive than local MongoDB * API compatibility quirks | 1 |
| 2 | Docker Container | * Portable DB image * Works with microservices | * Not a managed service * Needs persistent volume setup | 2 |
| 3 | Virtual Machine | * Full control * Supports any Mongo version | * No managed backups * High risk of data loss without setup | 3 |

Best Case:

| **Layer** | **Best Option** |
| --- | --- |
| Frontend | **Azure Static Web App** |
| Backend | **Azure App Service** |
| Database | **Azure Cosmos DB MongoDB API** |