**Deploying .NET Core gRPC Service on AKS**

* ***how to deploy dotnet core grpc service on aks and consume it***

**Deploying a .NET Core gRPC service on Azure Kubernetes Service (AKS) and consuming it involves several steps. Here is a high-level overview of the process:**

* **Create a .NET Core gRPC Service:**
* **Develop your gRPC service using .NET Core.**
* **Ensure your service is running correctly locally.**
* **Containerize the gRPC Service:**
* **Create a Dockerfile to containerize your gRPC service.**
* **Build and push the Docker image to a container registry (e.g., Azure Container Registry).**
* **Deploy the gRPC Service to AKS:**
* **Create an AKS cluster if you don't have one.**
* **Create Kubernetes deployment and service manifests.**
* **Apply the manifests to deploy your gRPC service to AKS.**
* **Consume the gRPC Service:**
* **Create a .NET Core client application to consume the gRPC service.**
* **Configure the client to call the gRPC service deployed on AKS.**
* **Below are detailed steps and code snippets for each part.**

**step1. Create a .NET Core gRPC Service**

**First, create a new gRPC service project using the .NET CLI:**

**bash**

**dotnet new grpc -o MyGrpcService**

**cd MyGrpcService**

**step:2. Containerize the gRPC Service**

**Create a Dockerfile in the root of your project:**

**Dockerfile**

**# Use the official .NET Core SDK image to build the app**

**FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build**

**WORKDIR /app**

**# Copy and restore dependencies**

**COPY \*.csproj .**

**RUN dotnet restore**

**# Copy the remaining files and build the app**

**COPY . .**

**RUN dotnet publish -c Release -o out**

**# Use the official .NET Core runtime image**

**FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS runtime**

**WORKDIR /app**

**COPY --from=build /app/out .**

**# Expose the port on which the gRPC server will listen**

**EXPOSE 5000**

**ENV ASPNETCORE\_URLS=http://+:5000**

**# Run the app**

**ENTRYPOINT ["dotnet", "MyGrpcService.dll"]**

**Build and push the Docker image:**

**bash**

**# Build the Docker image**

**docker build -t mygrpcservice .**

**# Tag the image for Azure Container Registry (ACR)**

**docker tag mygrpcservice <your\_acr\_name>.azurecr.io/mygrpcservice:latest**

**# Push the image to ACR**

**docker push <your\_acr\_name>.azurecr.io/mygrpcservice:latest**

**3. Deploy the gRPC Service to AKS**

**Create AKS Cluster**

**If you don't have an AKS cluster, create one using the Azure CLI:**

**az aks create --resource-group <your\_resource\_group> --name <your\_aks\_name> --node-count 1 --enable-addons monitoring --generate-ssh-keys**

**Create Kubernetes Deployment and Service Manifests**

**Create a deployment.yaml file:**

**YAML**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: mygrpcservice-deployment**

**spec:**

**replicas: 1**

**selector:**

**matchLabels:**

**app: mygrpcservice**

**template:**

**metadata:**

**labels:**

**app: mygrpcservice**

**spec:**

**containers:**

**- name: mygrpcservice**

**image: <your\_acr\_name>.azurecr.io/mygrpcservice:latest**

**ports:**

**- containerPort: 5000**

**Create a service.yaml file:**

**YAML**

**apiVersion: v1**

**kind: Service**

**metadata:**

**name: mygrpcservice**

**spec:**

**type: LoadBalancer**

**ports:**

**- port: 80**

**targetPort: 5000**

**selector:**

**app: mygrpcservice**

**Apply the manifests to deploy your service:**

**bash**

**kubectl apply -f deployment.yaml**

**kubectl apply -f service.yaml**

**4. Consume the gRPC Service**

**Create a .NET Core client application to consume the gRPC service:**

**bash**

**dotnet new console -o GrpcClient**

**cd GrpcClient**

**dotnet add package Grpc.Net.Client**

**dotnet add package Grpc.Tools**

**Add the following code in Program.cs:**

**C#**

**using System;**

**using System.Threading.Tasks;**

**using Grpc.Net.Client;**

**using MyGrpcService;**

**class Program**

**{**

**static async Task Main(string[] args)**

**{**

**// The address should be the external IP address of the LoadBalancer service in AKS**

**using var channel = GrpcChannel.ForAddress("http://<external\_ip>");**

**var client = new Greeter.GreeterClient(channel);**

**var reply = await client.SayHelloAsync(new HelloRequest { Name = "World" });**

**Console.WriteLine("Greeting: " + reply.Message);**

**}**

**}**

**Replace <external\_ip> with the external IP address of your LoadBalancer service.**

**Finally, run your client application to consume the gRPC service:**

**dotnet run**

**This process should help you deploy and consume a .NET Core gRPC service on AKS.**