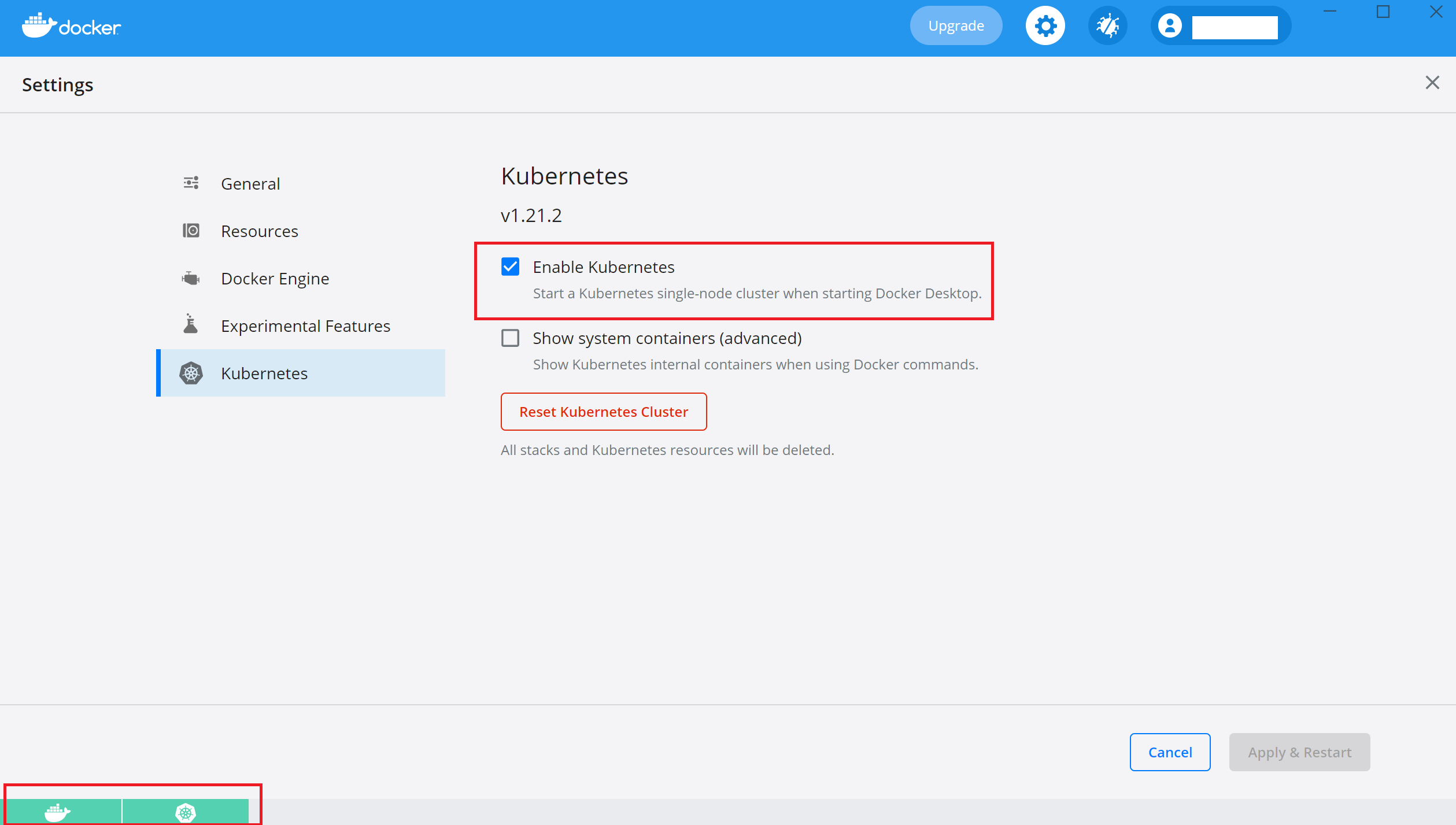
# Blazor+Dapr+K8s微服务之服务调用

## Dapr环境配置

### 在开发机安装Docker Desktop并启用Kubernetes

安装过程略，安装好后效果如下：（左下角两个绿色指示Docker和K8s正在运行）



### 在开发机安装Dapr Cli

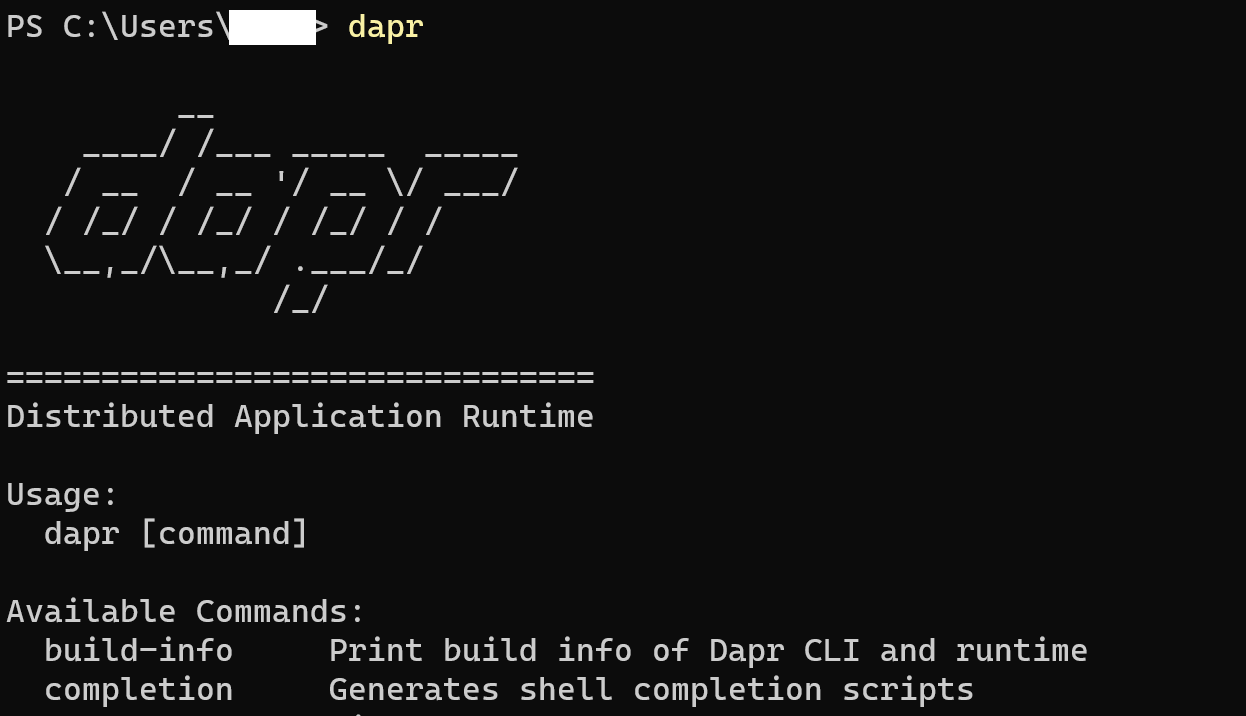
安装命令：powershell -Command "iwr -useb https://raw.githubusercontent.com/dapr/cli/master/install/install.ps1 | iex"

将会安装到如下目录：

Graphical user interface, text, application

Description automatically generated

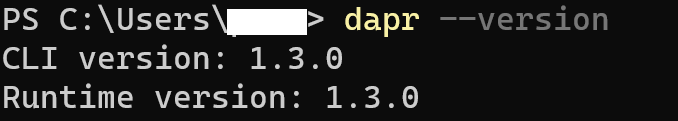
验证安装命令：dapr



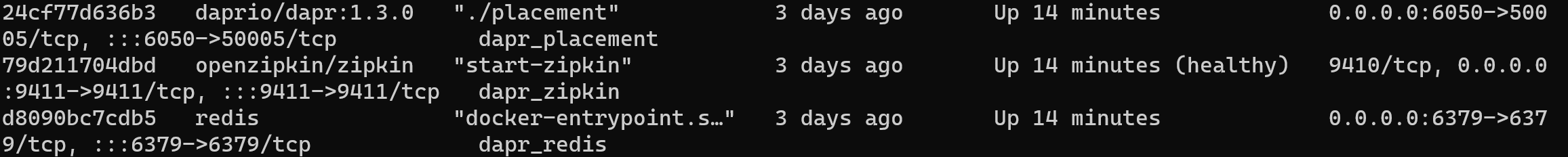
### 为开发机配置Dapr

配置命令：dapr init

验证命令：dapr --version



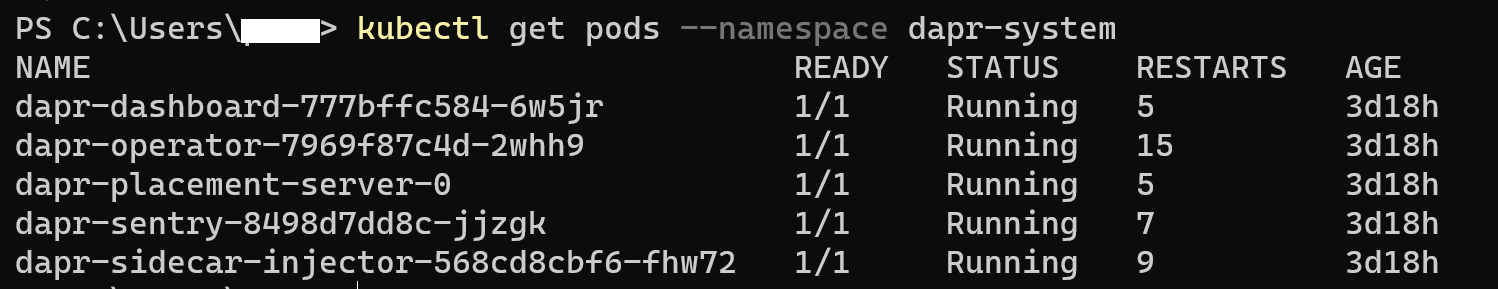
命令：Docker ps 可以看到多了Dapr自动加了三个容器：



### 为K8s配置Dapr

配置命令：dapr init -k

验证命令：kubectl get pods --namespace dapr-system



可通过命令: kubectl port-forward deploy/dapr-dashboard --namespace dapr-system 8080:8080

访问Dapr Dashbord

Graphical user interface, application

Description automatically generated

## 创建项目

### 创建Blazor WebAssembly项目

注意选择Asp.net Core hosted

Graphical user interface, text, application, email

Description automatically generated

### 向解决方案添加Asp.net Core WebApi项目

添加完如图,我增加了两个解决方案文件夹BlazorWeb和ServiceAPI以方便区分前端和后端。

Graphical user interface, application

Description automatically generated

启动DaprTest1.Server Project运行：

Table

Description automatically generated

## 直接调用微服务

当前解决方案：DaprTest1.Client Blazor 项目的Fetch data 功能会调用DaprTest1.Server项目的WeatherForecastController.cs 中的Get接口以获取天气预报信息。

我们需要修改为：在DaprTest1.Server项目的WeatherForecastController.cs 中的Get接口不再直接返回天气预报信息，而是调用DaprTest1.ServiceApi1项目的WeatherForecastController.cs Get 接口获取天气预报。

### 引用Refit包

在项目DaprTest1.Server中安装Refit 包。

Graphical user interface, text, application

Description automatically generated

### 修改相关代码

在项目DaprTest1.ServiceApi1中引用DaprTest1.Shared项目，并删除原来的Wheatherforcast.cs实体类。

Graphical user interface, application

Description automatically generated

在项目DaprTest1.Server 中新增文件ICallServiceApi1.cs，并添加如下代码：

public interface ICallServiceApi1

{

[Get("/WeatherForecast")]

Task<IEnumerable<WeatherForecast>> GetWeatherForecast();

}

在项目DaprTest1.Server 的Startup.cs 中新增如下代码，其中<http://localhost:8001> 是ServiceApi1接口访问地址

// 注入httpClient

services.AddHttpClient("HttpClient")

.AddTypedClient(client =>

{

client.BaseAddress = new Uri("http://localhost:8001");

return RestService.For<ICallServiceApi1>(client);

});

修改项目DaprTest1.Server WeatherForecastController.cs 中的Get接口如下:

public class WeatherForecastController : ControllerBase

{

private readonly ILogger<WeatherForecastController> \_logger;

private readonly ICallServiceApi1 \_callServiceApi1;

public WeatherForecastController(ILogger<WeatherForecastController> logger, ICallServiceApi1 callServiceApi1)

{

\_logger = logger;

\_callServiceApi1 = callServiceApi1;

}

[HttpGet]

public async Task<IEnumerable<WeatherForecast>> Get()

{

return await \_callServiceApi1.GetWeatherForecast();

//var client = DaprClient.CreateInvokeHttpClient(appId: "serviceapi1");

//var response = await client.GetFromJsonAsync<IEnumerable<WeatherForecast>>("WeatherForecast");

//return response;

}

}

分别启动DaprTest1.Server 和 DaprTest1.ServiceApi1项目,访问http://localhost:8000/可看到效果.

## 通过Dapr调用微服务

我们需要修改为：在DaprTest1.Server中不直接调用DaprTest1.ServiceApi1的接口地址获取天气预报信息,而是通过Dapr调用DaprTest1.ServiceApi1 的服务ID来获取天气预报信息.

### 引用Dapr.Client包

在项目DaprTest1.Server中安装Dapr.Client包

Graphical user interface, application

Description automatically generated

### 修改相关代码

在项目DaprTest1.Server 的Startup.cs 中修改如下代码: 本代码参考了 张善友大神的

[Dapr 客户端 搭配 WebApiClientCore 玩耍服务调用](https://www.cnblogs.com/shanyou/p/15106293.html)

services.AddScoped<InvocationHandler>();

// 注入httpClient

services.AddHttpClient("HttpClient").AddHttpMessageHandler<InvocationHandler>()

.AddTypedClient(client =>

{

client.BaseAddress = new Uri("http://serviceapi1");

return RestService.For<ICallServiceApi1>(client);

});

### 在Dapr中运行项目

分别在DaprTest1.Server和DaprTest1.ServiceApi1项目中添加文件dapr-selfhosted.ps1

文件内容分别为:

dapr run `

--app-id blazorweb `

--app-port 8000 `

--dapr-http-port 3600 `

--dapr-grpc-port 60000 `

dotnet run

dapr run `

--app-id serviceapi1 `

--app-port 8001 `

--dapr-http-port 3601 `

--dapr-grpc-port 60001 `

dotnet run

PowerShell分别启动DaprTest1.Server 和 DaprTest1.ServiceApi1项目下面的dapr-selfhosted.ps1文件, 访问 <http://localhost:8000/> 即可看到效果

Text

Description automatically generated

Text

Description automatically generated

## 在K8s中运行项目

### 添加Docker文件生成镜像

分别在DaprTest1.Server和DaprTest1.ServiceApi1项目中添加Docker文件dockerfile, 并修改文件内容如下:

ARG NET\_IMAGE=5.0-buster-slim

FROM mcr.microsoft.com/dotnet/aspnet:${NET\_IMAGE} AS base

WORKDIR /app

EXPOSE 5000

FROM mcr.microsoft.com/dotnet/sdk:${NET\_IMAGE} AS build

WORKDIR /src

COPY ["Server/DaprTest1.Server.csproj", "Server/"]

COPY ["Shared/DaprTest1.Shared.csproj", "Shared/"]

COPY ["Client/DaprTest1.Client.csproj", "Client/"]

RUN dotnet restore "Server/DaprTest1.Server.csproj"

COPY . .

WORKDIR "/src/Server"

RUN dotnet build "DaprTest1.Server.csproj" -c Release -o /app/build

FROM build AS publish

RUN dotnet publish "DaprTest1.Server.csproj" -c Release -o /app/publish

FROM base AS final

WORKDIR /app

COPY --from=publish /app/publish .

ENTRYPOINT ["dotnet", "DaprTest1.Server.dll"]

在解决方案根目录增加文件build-docker-images.ps1 以生成Docker image,文件内容如下:

docker build -t dapr-test1/blazorweb:1.0 -f Server/Dockerfile .

docker build -t dapr-test1/serviceapi1:1.0 -f ServiceApi/DaprTest1.ServiceApi1/Dockerfile .

在解决方案根目录执行build-docker-images.ps1后, 可看到生成的两个镜像:



### 部署Docker镜像到K8s

在解决方案根目录新建文件夹 Deploy, 放置k8s部署文件

Table

Description automatically generated with medium confidence

其中deploy.ps1 为部署命令,内容如下:

kubectl apply `

-f namespace.yaml `

-f dapr-config.yaml `

-f blazorweb.yaml `

-f serviceapi1.yaml

部署完成后,可通过端口转发给node看到效果.

Text

Description automatically generated

Table

Description automatically generated