

<b>COMPUTING SUBJECT:</b>	Restful ASP.Net Core-services For .Net
<b>TYPE:</b>	Assignment
<b>IDENTIFICATION:</b>	RestService#4
<b>COPYRIGHT:</b>	<i>Peter Levinsky &amp; Michael Claudius</i>
<b>LEVEL:</b>	Medium
<b>TIME CONSUMPTION:</b>	2-2½ hours
<b>EXTENT:</b>	50-60 lines
<b>OBJECTIVE:</b> to the Restful services	Publishing in Azure and adding supporting CORS  version .Net
<b>PRECONDITIONS:</b>	Rest service theory. Http-concepts Computer Networks Ch. 2.2
<b>COMMANDS:</b>	

**IDENTIFICATION:** RestService#4 / PELE with kindly respect and inspiration from MICL

### Overall Purpose

The overall purpose for the group of 'RestService' assignments is to be able to provide and consume restful ASP.Net Core web services, to prepare the 'RestService' to be published in Azure, including testing the service and finally to setup the 'RestService' to be consumed from a browser (e.g. using Typescript) i.e. support CORS.

The whole group of assignments consist of 7 steps:

1. [A simple REST Service with CRUD.](#)
2. [More advanced and complex URI's.](#)
3. [Testing a REST Service.](#)
- 4. Adding Support for CORS to the REST Service  
(this assignment)**
5. Consuming a REST service from a C# Console application.
6. A REST Service using a database

### Background Material:

The HTTP protocol: See Computer Network chap 2 pp. 111-136

Note of REST (Peter Levinsky): See [NetHttpNote.pdf](#)

Oswago Universitet: RESTful Service Best Practices: Recommendations for Creating Web Services: See <http://cs.oswego.edu/~alex/teaching/csc435/RESTful.pdf>

Usefull tools (Postman & Fiddler): See [Tools.htm](#) (tool #3 & tool #4)

**Note:** <https://www.moesif.com/blog/technical/cors/Authoritative-Guide-to-CORS-Cross-Origin-Resource-Sharing-for-REST-APIs/#>

## This Assignment: RestService#4

### Purpose

The purpose of this assignment is to publish your REST service in Azure and to refactor your REST Service so it can manage call from javascript-pages in a browser in other words to support CORS.

### Mission

You are to upload your REST service up to an AZURE App-service.

You are to design and implement CORS (Cross Origin Resource Sharing). There are three different way to design and implement CORS, they varying in the granularity of access control.

1. Publishing the REST service.
2. GlobalWare, Quick, but not so configurable and UNSECURE in Azure  
(*do NOT work when using localhost!*)
3. MVC, Specific setup CORS for each URI.

Now you have tested your REST Service functional as well as an integration, so it is ready to be published to the cloud – at Zealand meaning Microsoft Cloud ‘Azure’.

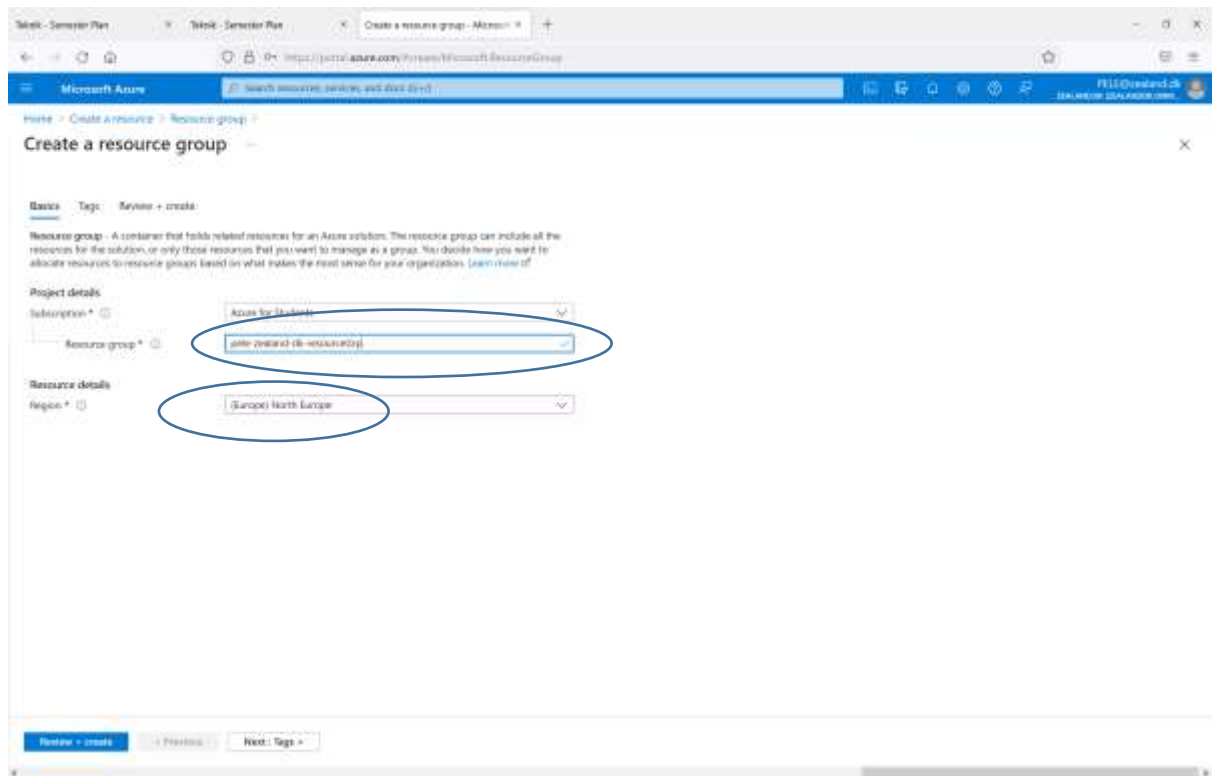
## Assignment 1: Publish in Azure

To publish in Azure you have to create an app-service in Azure i.e. make a virtual machine with a web-server (Microsoft IIS-server). Then to upload (publish) your REST-Service. You need to make a quick fix to support typescript (to be precise javascript) accessing your REST service

All this require you have an Azure account – see <https://helpdesk.zealand.dk/hc/en-us/articles/360023571932-Microsoft-Imagine> for more information.

- a. Go to your portal of Azure ( <https://portal.azure.com/> ) and log in.  
You must create a new APP-service ‘**ItemService**’

Step 1 – Add Resource group – if you do not already have one – then go to step 2



Naming resource group like ‘pele-zealand-dk-resourceGrp’  
Resource detail choose ‘North Europe’

## Step 2 – Add web service

Microsoft Azure portal - Create Web App page. The page shows the configuration for a new web application. The following fields are highlighted with blue circles:

- Subscription: Azure for Students
- Resource Group: pele-zealand-dk-resourceGrp
- Name: pele-zealand-dk-REST
- Publish: Code (selected), Docker Container
- Runtime stack: .NET 5
- Operating System: Windows (selected), Linux
- Region: North Europe
- App Service Plan: [New] ASP-pelezealanddkresourceGrp-bbd4
- Sku and size: Free F1

Buttons at the bottom: Review + create, < Previous, Next : Deployment >

Choose your resource group from step 1

Name your web application e.g. like 'pele-zealand-dk-REST'

Choose the runtime stack here .Net 5

Choose region i.e. Noth Europe

The default size is just fine – keep it that way

### Step 3 - Overview of services

Microsoft Azure

Search resources, services, and docs (G+Y)

PELE@zealand.dk  
ZEALAND.DK (ZEALAND.DK.ONM...)

## Microsoft.Web-WebApp-Portal-3182a3e6-bcad | Overview

Deployment

Search (Ctrl+J)

Delete Cancel Redeploy Refresh

We'd love your feedback! →

✓ Your deployment is complete

Deployment name: Microsoft.Web-WebApp-Portal-3182a3e6-...  
Subscription: Azure for Students  
Resource group: pele-zealand-dk-resourceGrp

Start time: 9/20/2021, 9:43:06 AM  
Correlation ID: 191e3fb9-e522-405c-89d8-8646d...

Deployment details (Download)

Resource	Type	Status	Operation details
pele-zealand-dk-REST	Microsoft.Web/sites	OK	Operation details
pele-zealand-dk-REST	microsoft.insights/compon...	OK	Operation details
pele-zealand-dk-REST	microsoft.insights/compon...	OK	Operation details
ASP-pelezeslanddkresourceC	Microsoft.Web/serverfarms	OK	Operation details
newWorkspaceTemplate	Microsoft.Resources/deplo...	OK	Operation details

Next steps

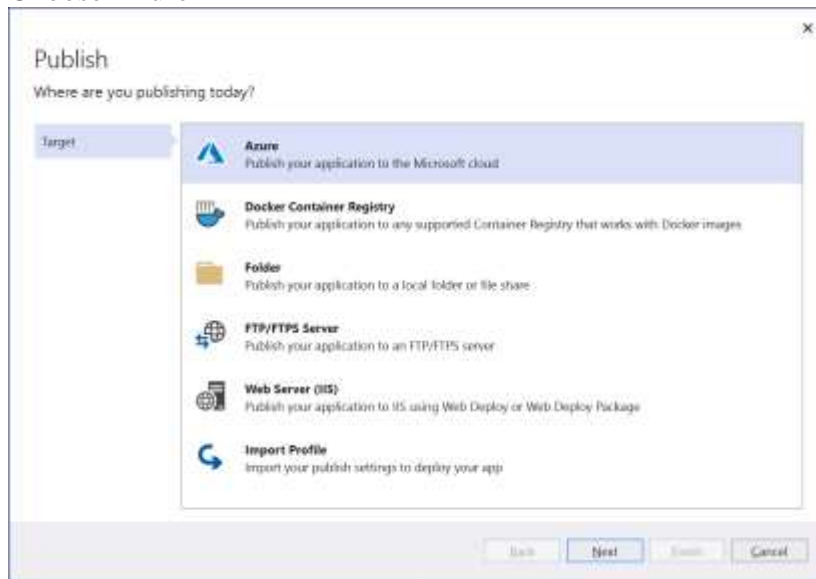
Manage deployments for your app. Recommended

Protect your app with authentication. Recommended

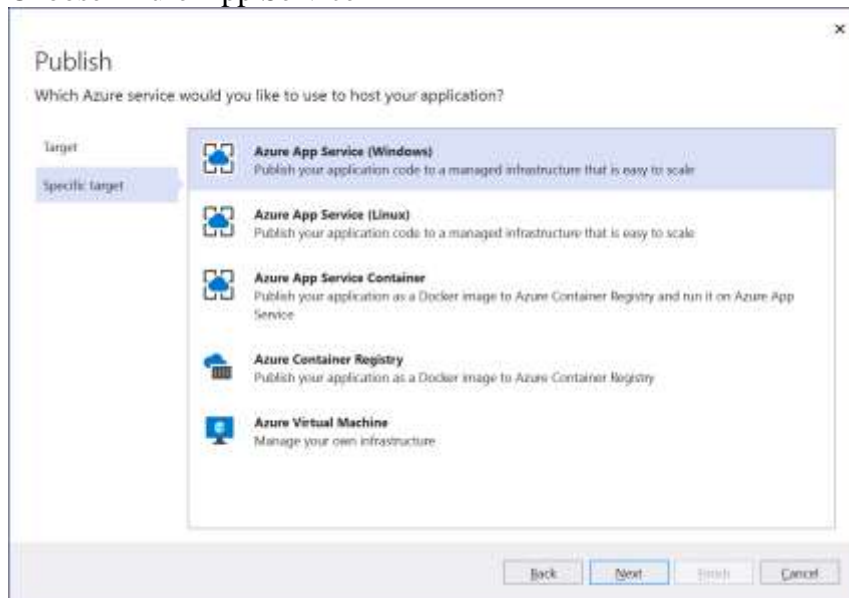
Go to resource

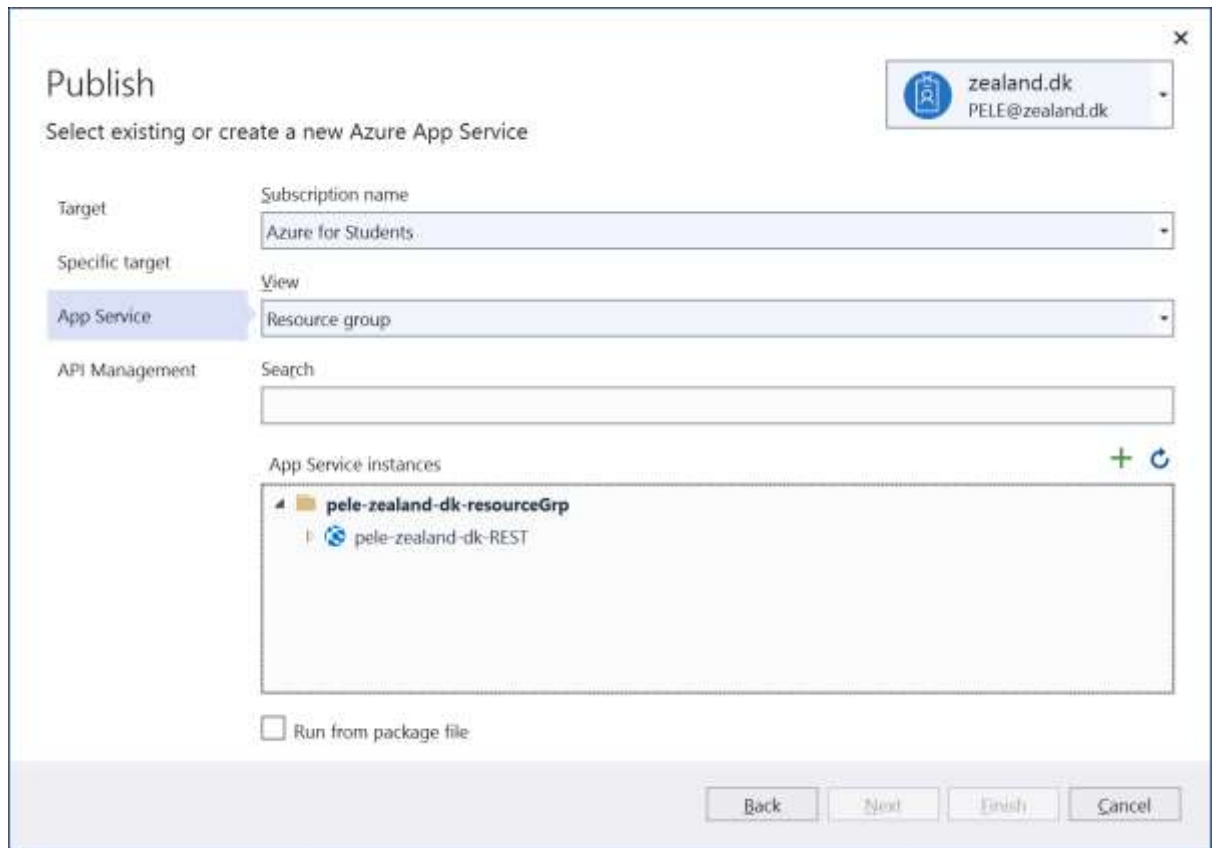
Now you are ready for the next step.

- b. In Visual Studio open the Solution Explore.  
Right-click at the project -> choose publish  
Choose Azure

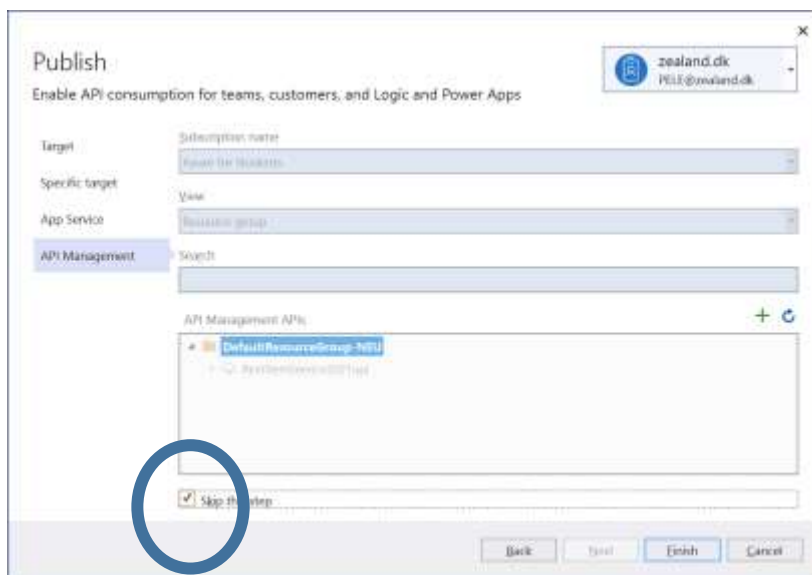


Choose Azure App Service





Choose your APP-Service e.g. ItemService (in example 'pele-zealand-dk-REST'):



**Skip this step.**

So that was it. A browser window will open with your RestService running in Azure with a URL-name like: <http://pele-zealand-dk-REST.azurewebsites.net/>.



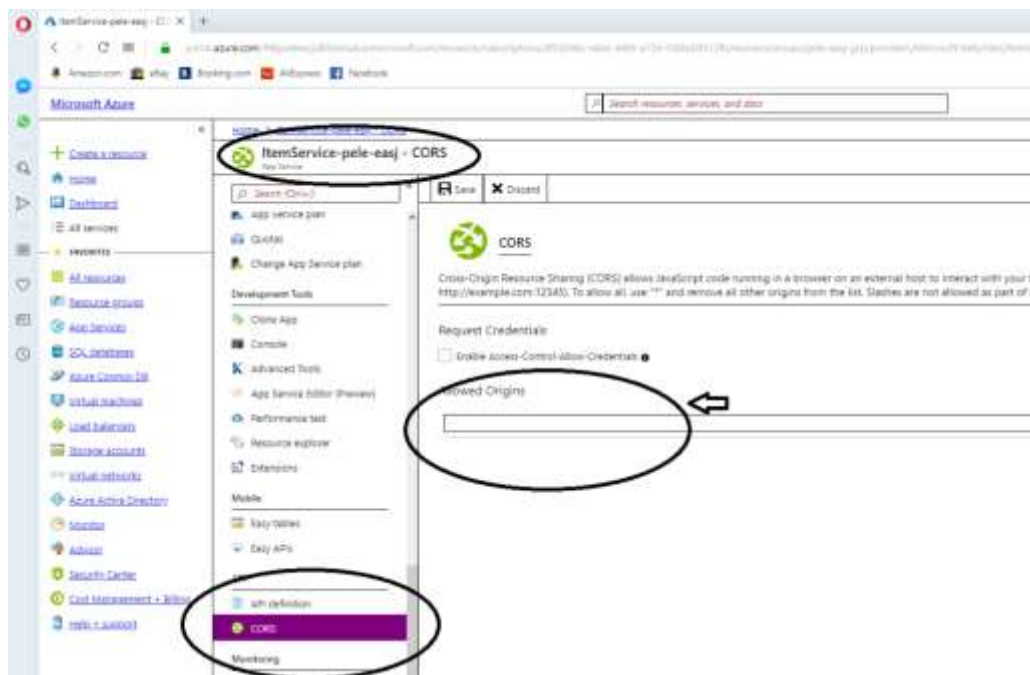
## Assignment 2: Support CORS (cross origin resource sharring) – Quick but dirty

To be able to access your REST-service from a javascript application in a browser your REST-service need to support CORS.

### a. Quick solution in Azure

Open the Azure portal <https://portal.azure.com/>

Open your APP Service that hold your REST Service, it will similar to this:



For Allowed origins insert '\*', meaning everything from anywhere.

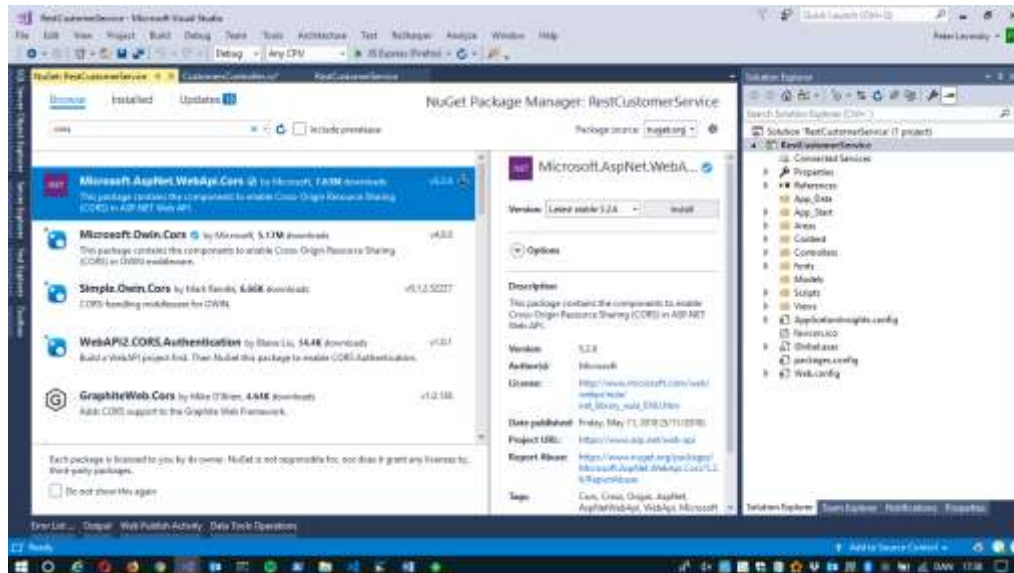
Remember to save.

Now it's working in your Azure REST Service.

*That was all for now creating the REST service – next step is to consume the rest service (very similar to what you did at 2 semester)*

### Assignment 3: MVC, Specific setup CORS for each URI

- a. First you need a NuGet-package installed; at your project open the NuGet manager and choose ‘**Microsoft.AspNetCore.Cors**’ version 2.2.0 to be installed:



and yes ... It do take some time ☹

- b. In the solution (Solution Explore); Open the file ‘**Startup.cs**’.  
In the ‘**ConfigureServices**’ method, add the line

```
services.AddCors(options =>
{
    options.AddPolicy("AllowSpecificOrigin",
        builder => builder.WithOrigins("http://zealand.dk").
            AllowAnyMethod().
            AllowAnyHeader()
        );

    options.AddPolicy("AllowAny",
        builder => builder.AllowAnyOrigin().
            AllowAnyMethod().
            AllowAnyHeader()
        );

    options.AddPolicy("AllowOnlyGetPut",
        builder => builder.AllowAnyOrigin().
            WithMethods("GET", "PUT").
            AllowAnyHeader()
        );
});
```

- c. Still in the Startup.cs – class in the ‘**Configure**’-method:  
Add the lines after ‘**app.UseRouting()**’

```
app.UseCors("AllowOnlyGetPut"); // one of the other policy names
```

And before ‘**app.UseAuthorization()**’

- d. Now extend this for those services i.e. methods you will have to support CORS.
- e. Publish your Rest Service in Azure and try with some of your Typescript applications.  
*(if you have solved assignment 1, then go back to Azure and remove the ‘\*’)*

### Check your setup of CORS

- f. Check that your REST service is correctly configured for CORS using Postman or Fiddler. Compose a **simple request** (i.e. GET) with a header-field :

*Origin: { your location e.g. http://easj.dk }*

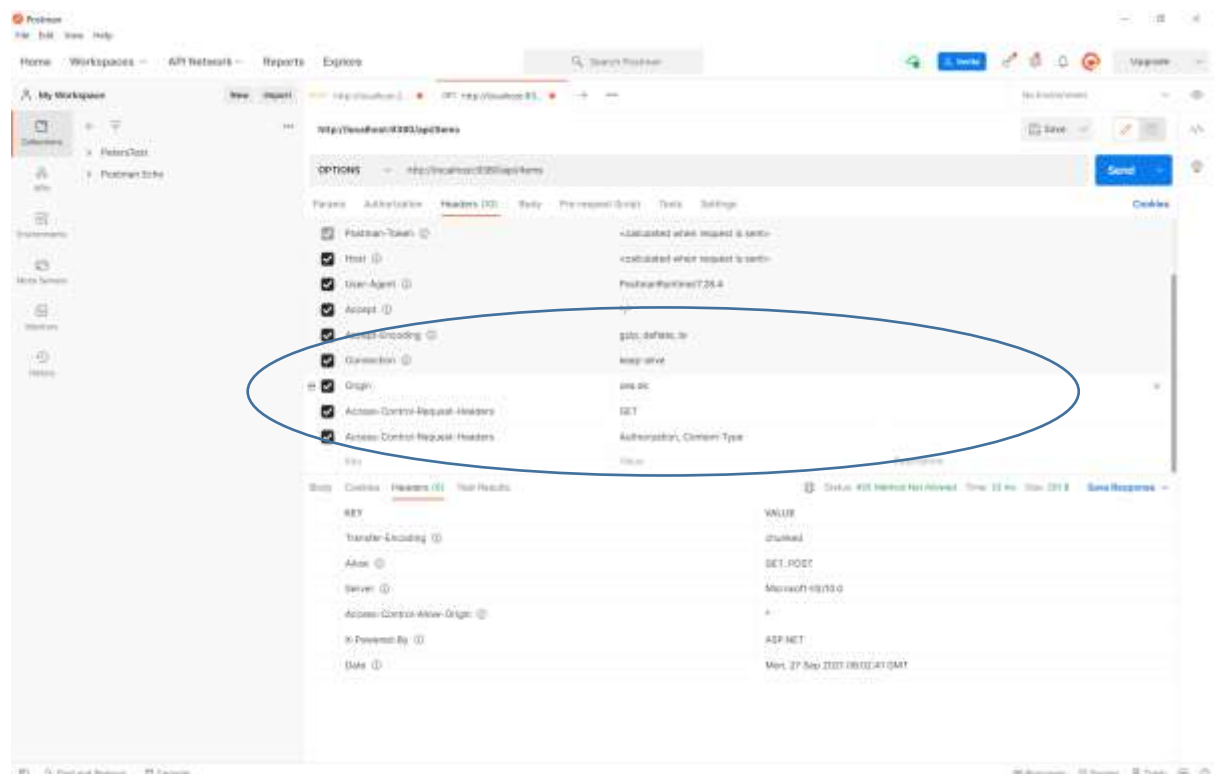
It should return a header field:

*Access-Control-Allow-Origin: {your location e.g. easj.dk}*

Or check for more complex request (i.e. PUT, POST, and DELETE) by a **preflighted request** using an ‘**OPTION**’ request.

*Origin: {your location e.g. http://easj.dk}*  
*Access-Control-Request-Method: GET*  
*Access-Control-Request-Headers: Authorization, Content-Type*

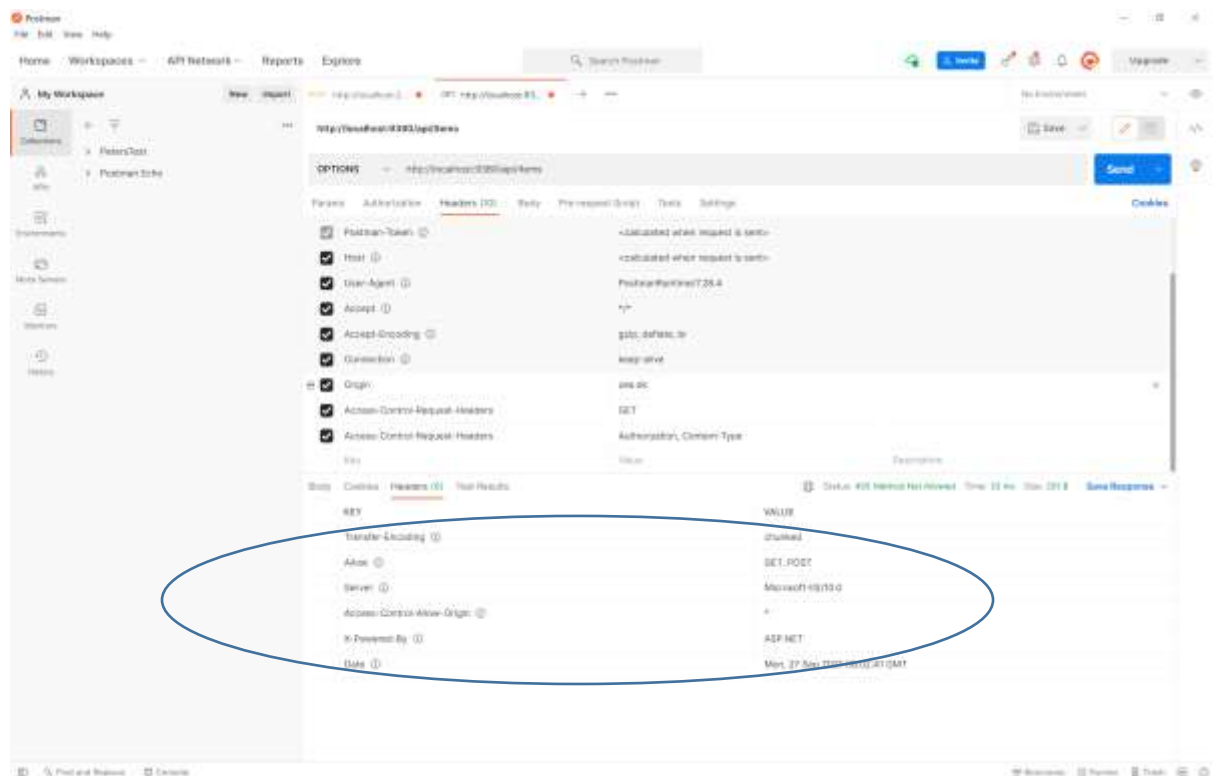
E.g. (Postman) :



The server (with your CORS REST service) should return:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: {your location e.g. easj.dk}
Access-Control-Allow-Methods: GET, PUT
Access-Control-Allow-Headers: Authorization, Content-Type
```

E.g. (Postman):



What happen if you request access to POST or DELETE ??

### More detailed setup

- f. For even more detailed CORS setup: In the controller, **ItemsController**, specify the policy you want on the controller itself, like:

```
[Route("api/[controller]")]
[ApiController]
```

Still the controller, specify the policy for the methods, suppressing the controller-policy.

```
[HttpDelete("{id}")]
// no policy i.e. inherits the controller policy
```

```
[HttpPost]
[EnableCors("AllowSpecificOrigin")]

[HttpGet]
[DisableCors] //disable the controller policy
```

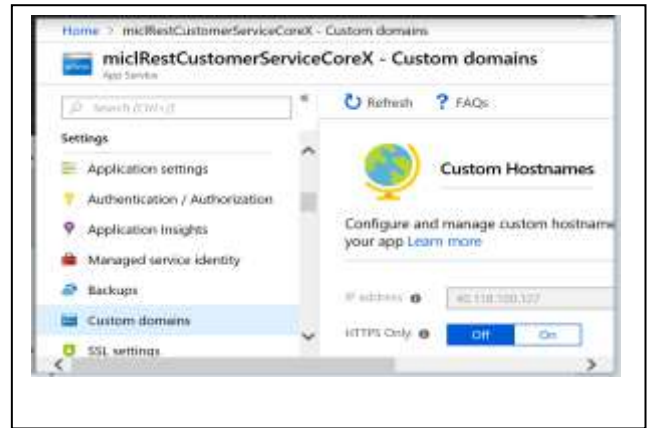
- g. After published in Azure, check your new configuration using Fiddler or Postman like in previous assignment.

- h. **If you miss to uncheck the https when creating the REST service in assignment 1 do this and the following bullet.** Unfortunately you probably get a 301/502 error security error.

*Why?*

*The issue is that if your project was created it was configured for Https and Fiddler uses Http-scheme for Azure. Read on...*

- i. Go to your Azure Portal
1. Open your Web-App project
  2. Find *Custom domains* in the left scroll-bar
  3. Set *Https-Only* to OFF
  4. Click *Refresh*



*Congratulations your REST service can now be used from e.g. a typescript application, the last step is to provide persistence in your REST service through a Database instead of a static-list.*