Microsoft Azure Cloud Services

SummitWorks™
GLOBAL SOLUTION ARCHITECTS

Authorized & published by Summitworks Technologies Inc

Agenda

- **Azure services**
  - Compute services
    - App Services
      - Web Apps
      - Mobile Apps
      - API Apps

App Services – Web Apps

Web Apps

Azure App Service Web Apps (or just Web Apps) is a service for hosting web applications, REST APIs, and mobile back ends. You can develop in your favorite language, be it *.NET, .NET Core, Java, Ruby, Node.js, PHP, or Python.* Applications run and scale with ease on Windows-based environments.

Web Apps not only adds the power of Microsoft Azure to your application, such as security, load balancing, auto scaling, and automated management. You can also take advantage of its DevOps capabilities, such as continuous deployment from Azure DevOps, GitHub, Docker Hub, and other sources, package management, staging environments, custom domain, and SSL certificates.

Web Apps

Why use Web Apps?

- **Multiple languages and frameworks -** Web Apps has first-class support for ASP.NET, ASP.NET Core, Java, Ruby, Node.js, PHP, or Python. You can also run PowerShell and other scripts or executables as background services.

- **DevOps optimization -** Set up continuous integration and deployment with Azure DevOps, GitHub, BitBucket, Docker Hub, or Azure Container Registry. Promote updates through test and staging environments. Manage your apps in Web Apps by using Azure PowerShell or the cross-platform command-line interface (CLI).

- **Global scale with high availability -** Scale up or out manually or automatically. Host your apps anywhere in Microsoft's global datacenter infrastructure, and the App Service SLA promises high availability.

Web Apps

Why use Web Apps?

- **Connections to SaaS platforms and on-premises data -** Choose from more than 50 connectors for enterprise systems (such as SAP), SaaS services (such as Salesforce), and internet services (such as Facebook). Access on-premises data using Hybrid Connections and Azure Virtual Networks.

- **Security and compliance -** App Service is ISO, SOC, and PCI compliant. Authenticate users with Azure Active Directory or with social login (Google, Facebook, Twitter, and Microsoft). Create IP address restrictions and manage service identities.

- **Application templates -** Choose from an extensive list of application templates in the Azure Marketplace, such as WordPress, Joomla, and Drupal.

Web Apps

Why use Web Apps?

- **Visual Studio integration -** Dedicated tools in Visual Studio streamline the work of creating, deploying, and debugging.

- **API and mobile features -** Web Apps provides turn-key CORS support for RESTful API scenarios, and simplifies mobile app scenarios by enabling authentication, offline data sync, push notifications, and more.

- **Serverless code -** Run a code snippet or script on-demand without having to explicitly provision or manage infrastructure, and pay only for the compute time your code actually uses (see Azure Functions).

Create an ASP.NET Web App in Azure

In Visual Studio, create a project by selecting

File > New > Project

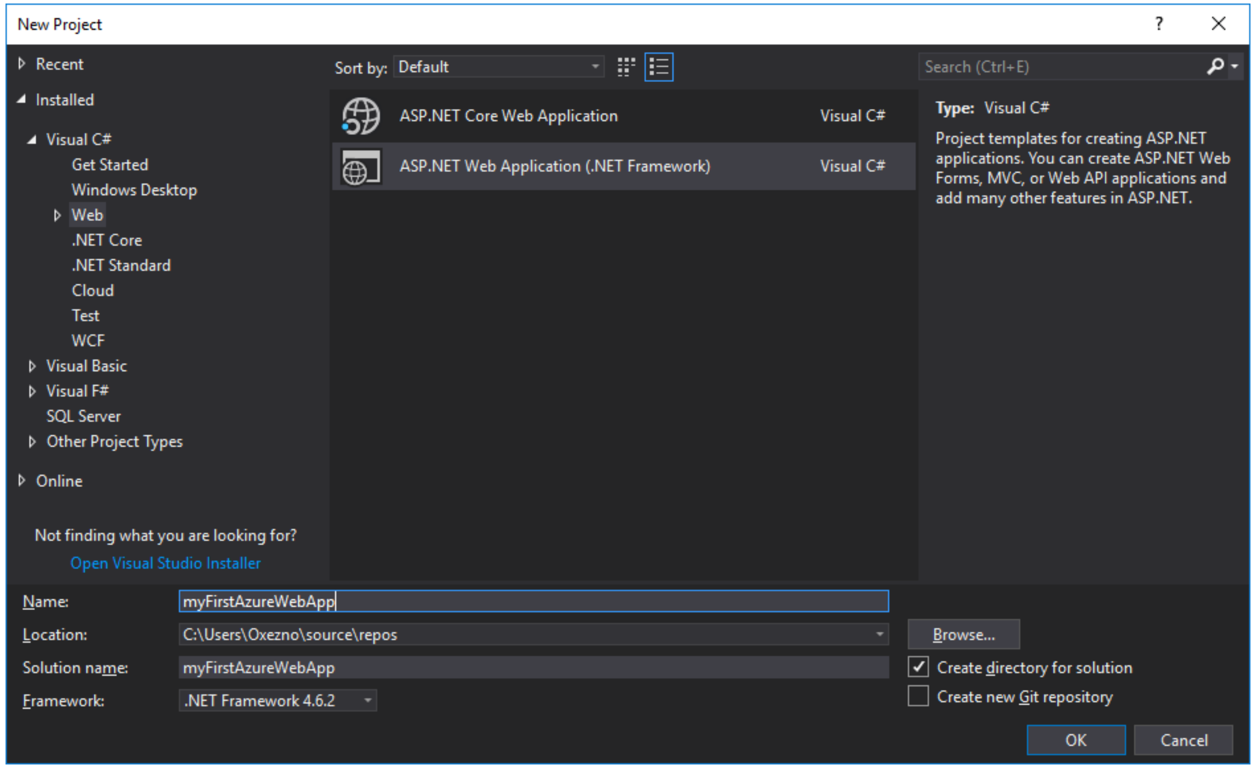
In the **New Project** dialog, select

Visual C# > Web > ASP.NET Web Application (.NET Framework).

Name the application

myFirstAzureWebApp,

and then select **OK**.



New Project

- Recent
- Installed
  - Visual C#
    - Get Started
    - Windows Desktop
    - Web
    - .NET Core
    - .NET Standard
    - Cloud
    - Test
    - WCF
  - Visual Basic
  - Visual F#
  - SQL Server
  - Other Project Types
- Online

Not finding what you are looking for?
Open Visual Studio Installer

Sort by: Default

ASP.NET Core Web Application	Visual C#
ASP.NET Web Application (.NET Framework)	Visual C#

Search (Ctrl+E)

Type: Visual C#

Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.

Name:	myFirstAzureWebApp
Location:	C:\Users\Oxezno\source\repos
Solution name:	myFirstAzureWebApp
Framework:	.NET Framework 4.6.2

Browse...

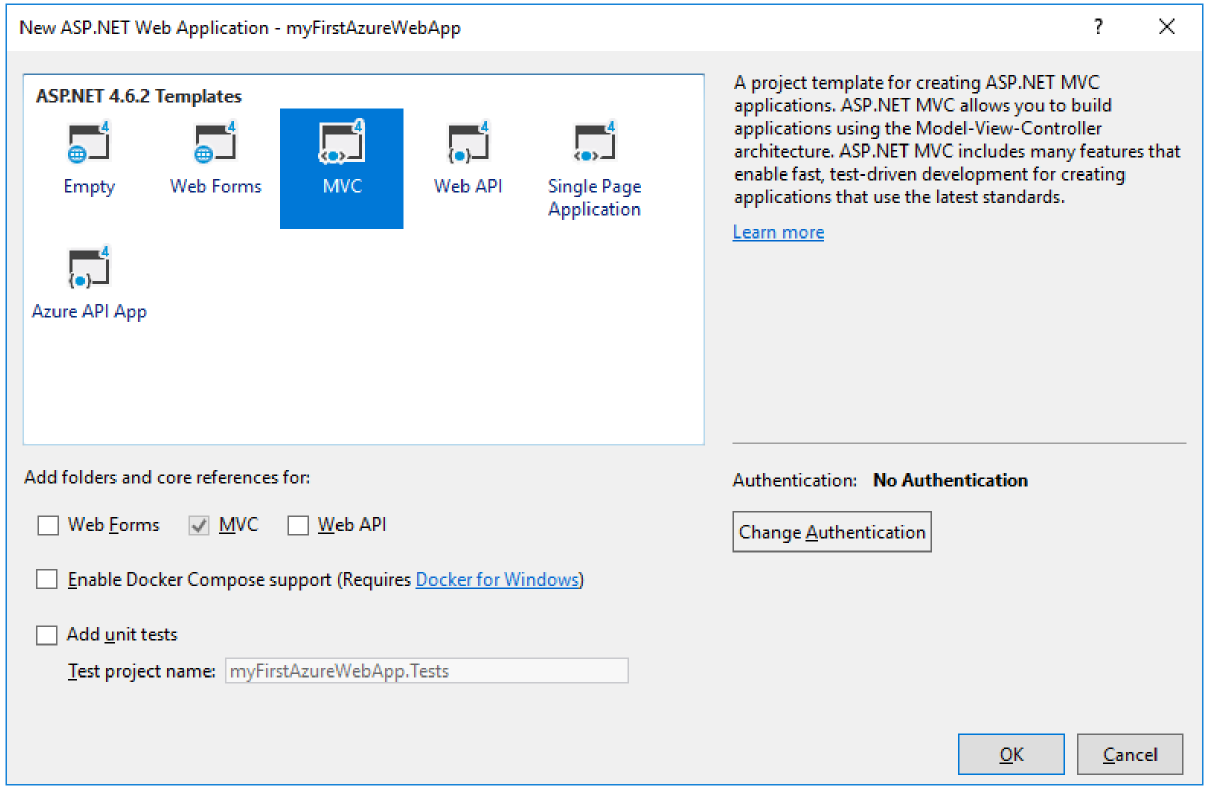
[✓] Create directory for solution

[ ] Create new Git repository

OK Cancel

Create an ASP.NET Web App in Azure

You can deploy any type of ASP.NET web app to Azure. Select the **MVC** template, and make sure authentication is set to **No Authentication**.


New ASP.NET Web Application - myFirstAzureWebApp

ASP.NET 4.6.2 Templates

A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

Learn more

Add folders and core references for:

- [ ] Web Forms
- [✓] MVC
- [ ] Web API

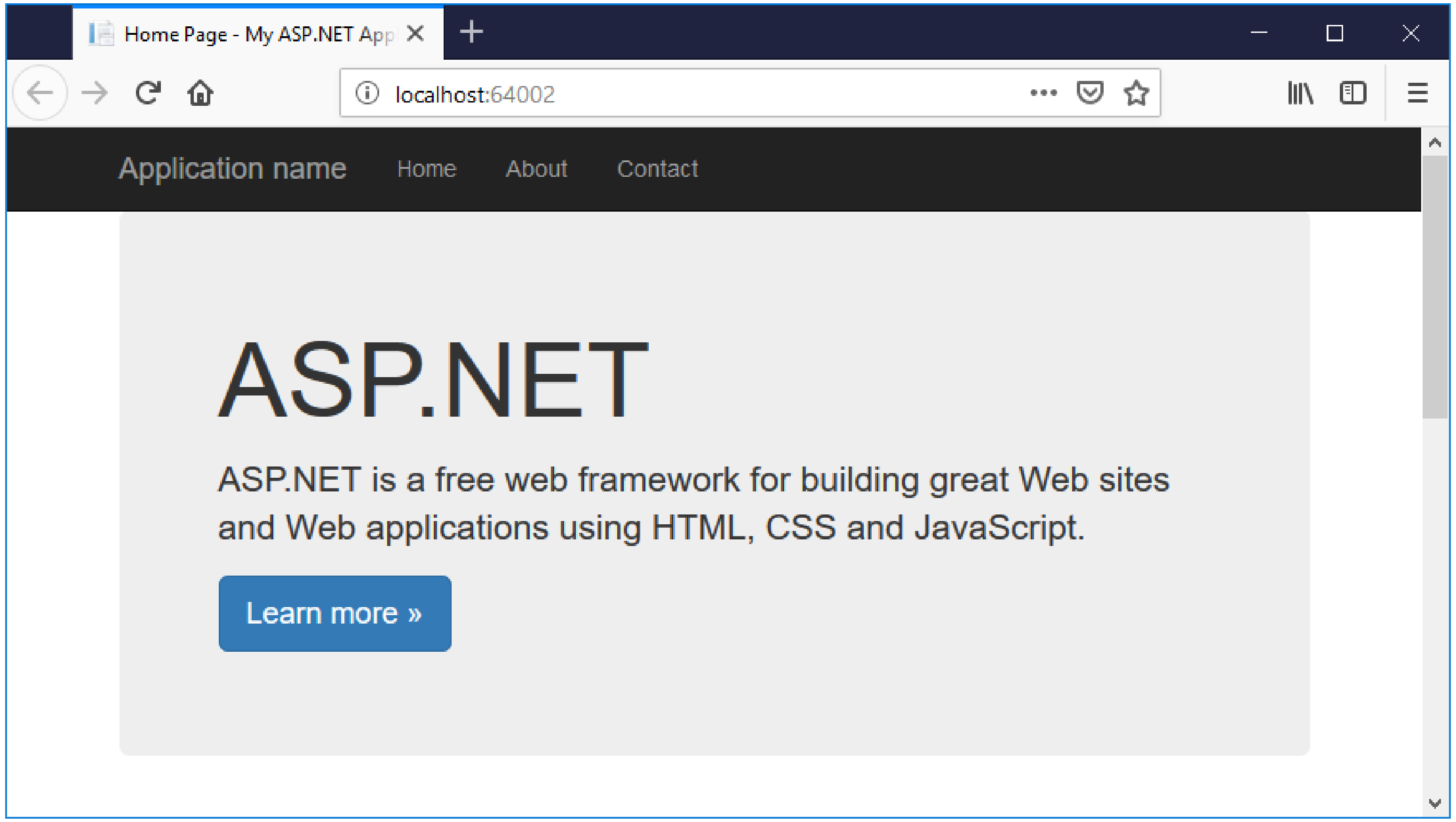
- [ ] Enable Docker Compose support (Requires Docker for Windows)
- [ ] Add unit tests

Test project name: myFirstAzureWebApp.Tests

Authentication: **No Authentication**

Create an ASP.NET Web App in Azure

From the menu, select ***Debug > Start without Debugging*** to run the web app locally.



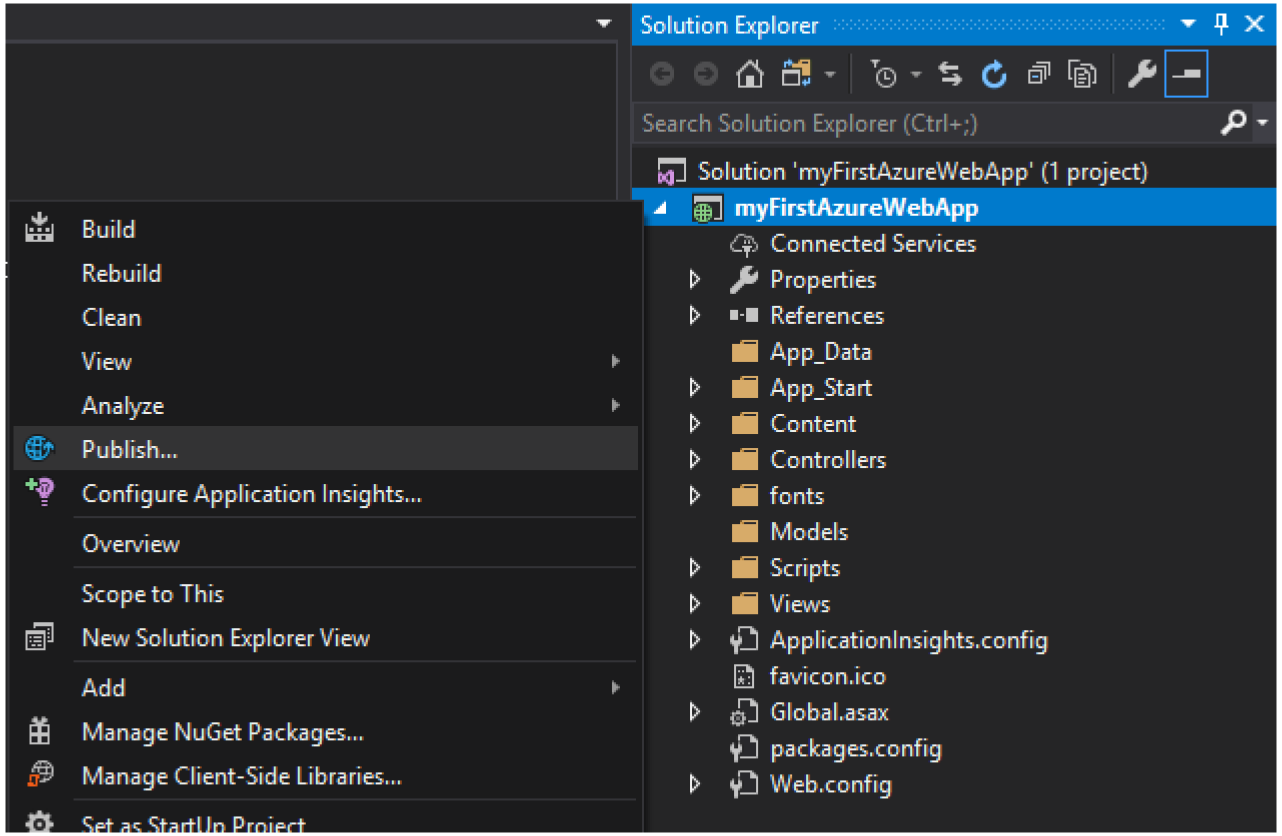
ASP.NET

ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.

Create an ASP.NET Web App in Azure

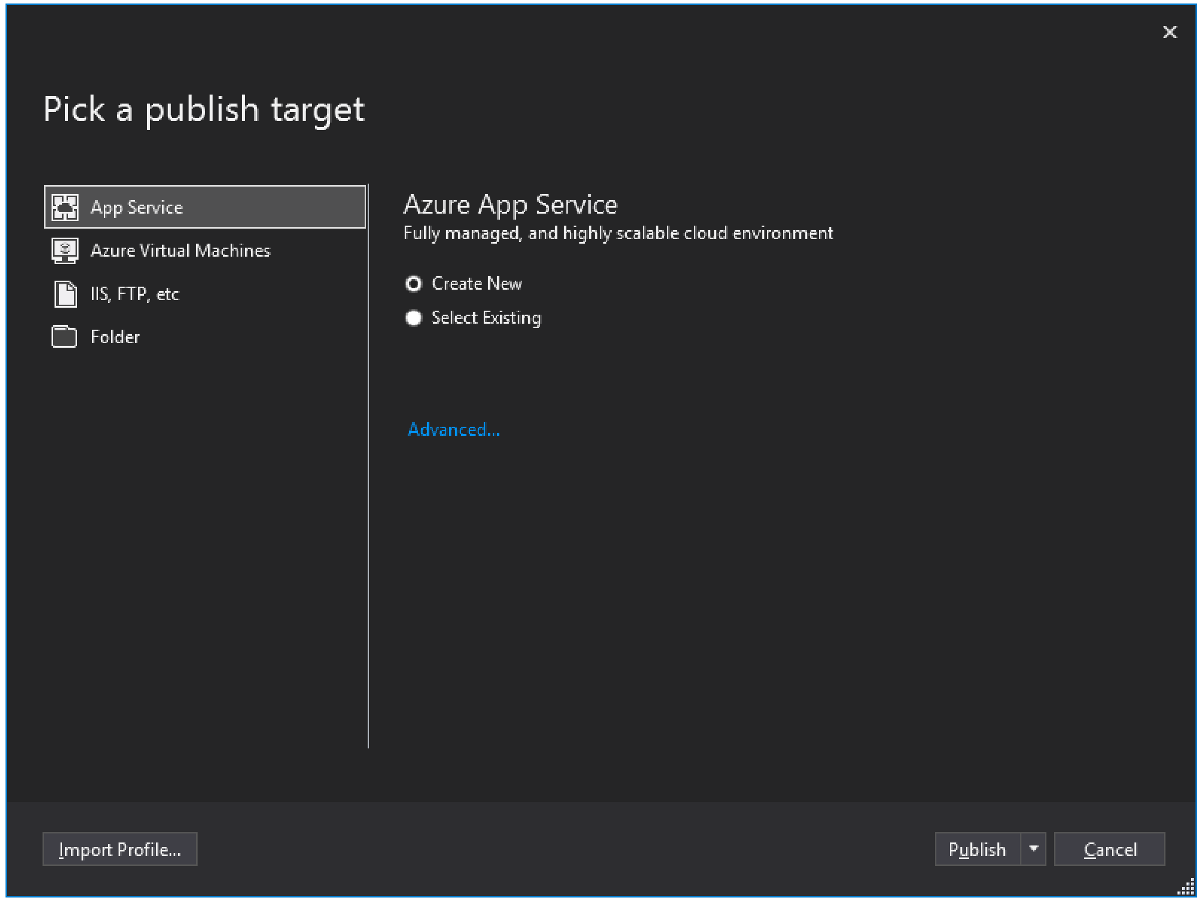
Launch the publish wizard

In the **Solution Explorer**, right-click the **myFirstAzureWebApp** project and select **Publish**.



Create an ASP.NET Web App in Azure

The publish wizard is automatically launched. Select **App Service** > **Publish** to open the **Create App Service** dialog.



Pick a publish target

App Service

Azure Virtual Machines

IIS, FTP, etc

Folder

Azure App Service

Fully managed, and highly scalable cloud environment

[✓ Create New]

Select Existing

Advanced...

Import Profile...

Publish

Cancel

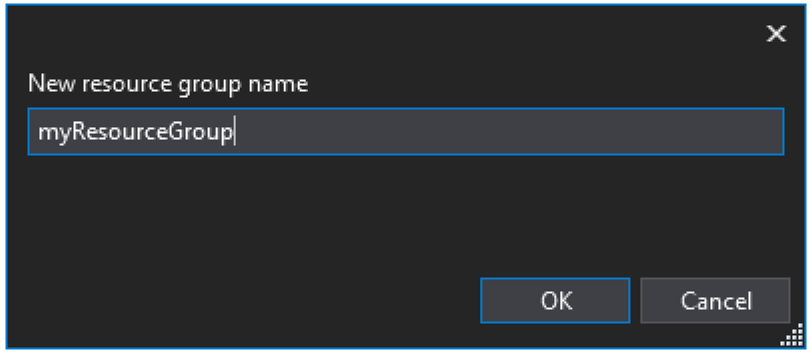
Create an ASP.NET Web App in Azure

Create a resource group

A resource group is a logical container into which Azure resources like web apps, databases, and storage accounts are deployed and managed. For example, you can choose to delete the entire resource group in one simple step later.

Next to **Resource Group**, select **New**.

Name the resource group **myResourceGroup** and select **OK**.


New resource group name

myResourceGroup

Create an ASP.NET Web App in Azure

Create an App Service plan

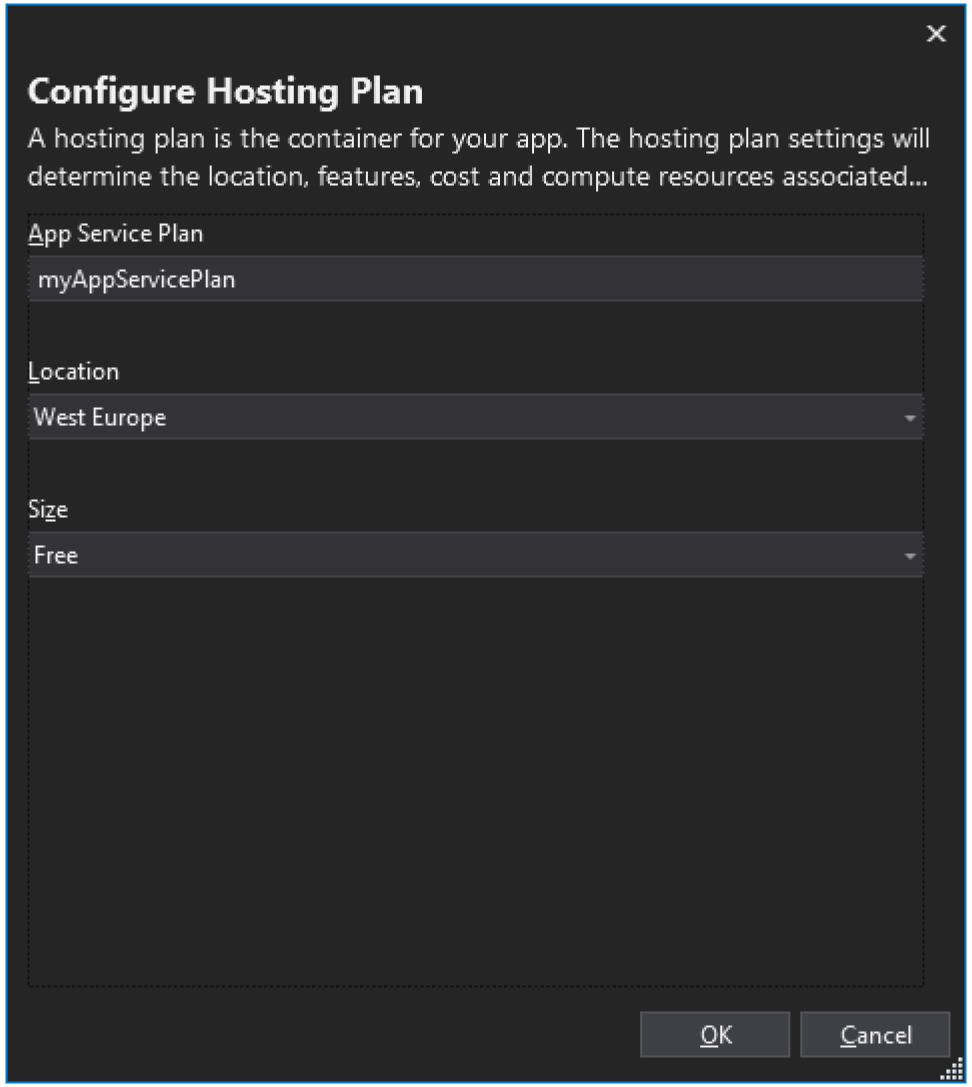
An App Service plan specifies the location, size, and features of the web server farm that hosts your app. You can save money when hosting multiple apps by configuring the web apps to share a single App Service plan.

App Service plans define:

- Region (North Europe, East US or Asia)
- Instance size (small, medium, or large)
- Scale count (1 to 20 instances)
- SKU (Free, Shared, Basic, Standard, or Premium)

Next to **Hosting Plan**, select **New**.

In the **Configure Hosting Plan** dialog, use the following settings.



Configure Hosting Plan

A hosting plan is the container for your app. The hosting plan settings will determine the location, features, cost and compute resources associated...

App Service Plan: myAppServicePlan

Location: West Europe

Size: Free

Create an ASP.NET Web App in Azure

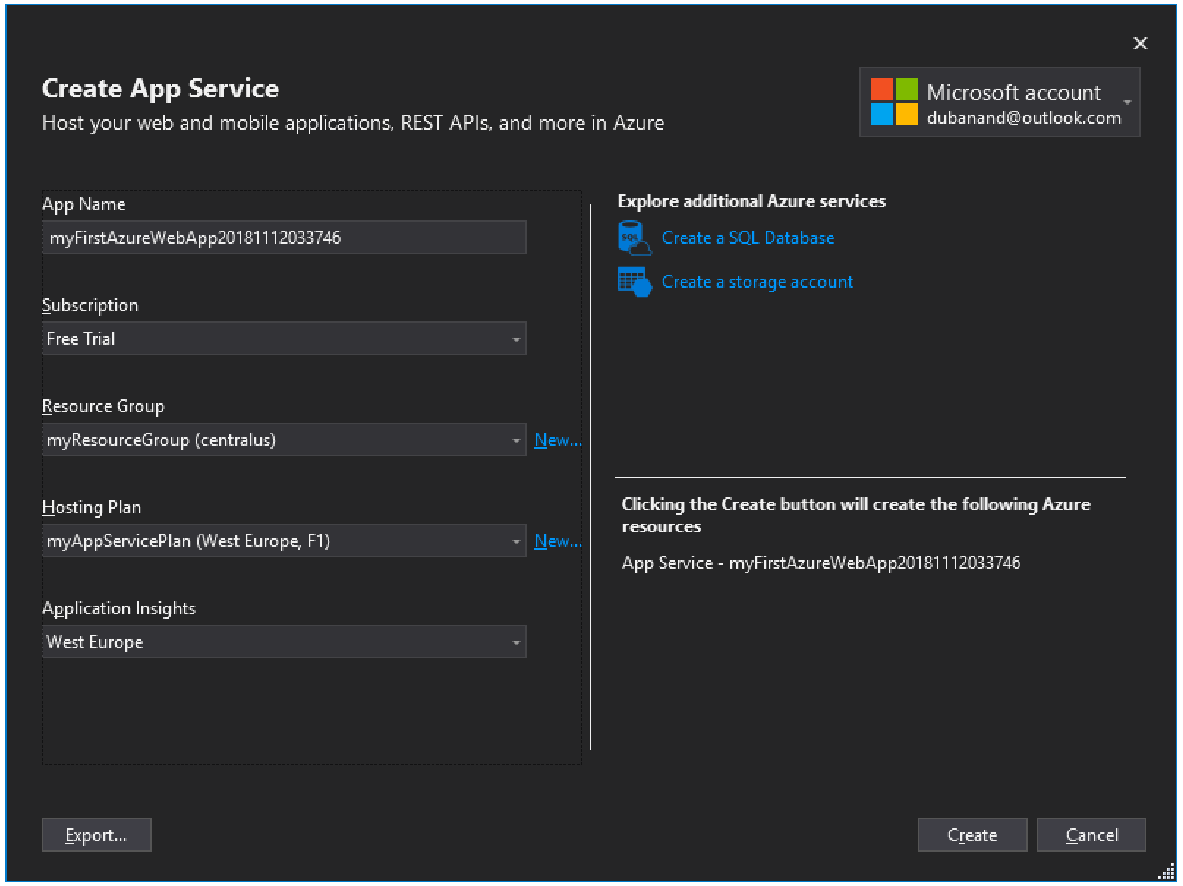
Create and publish the web app

In App Name, type a unique app name, or accept the automatically generated unique name. The URL of the web app is

http://<app_name>.azurewebsites.net

where <app_name> is your app name.

Select Create to start creating the Azure resources.



Create App Service

Host your web and mobile applications, REST APIs, and more in Azure

App Name: myFirstAzureWebApp20181112033746

Subscription: Free Trial

Resource Group: myResourceGroup (centralus)

Hosting Plan: myAppServicePlan (West Europe, F1)

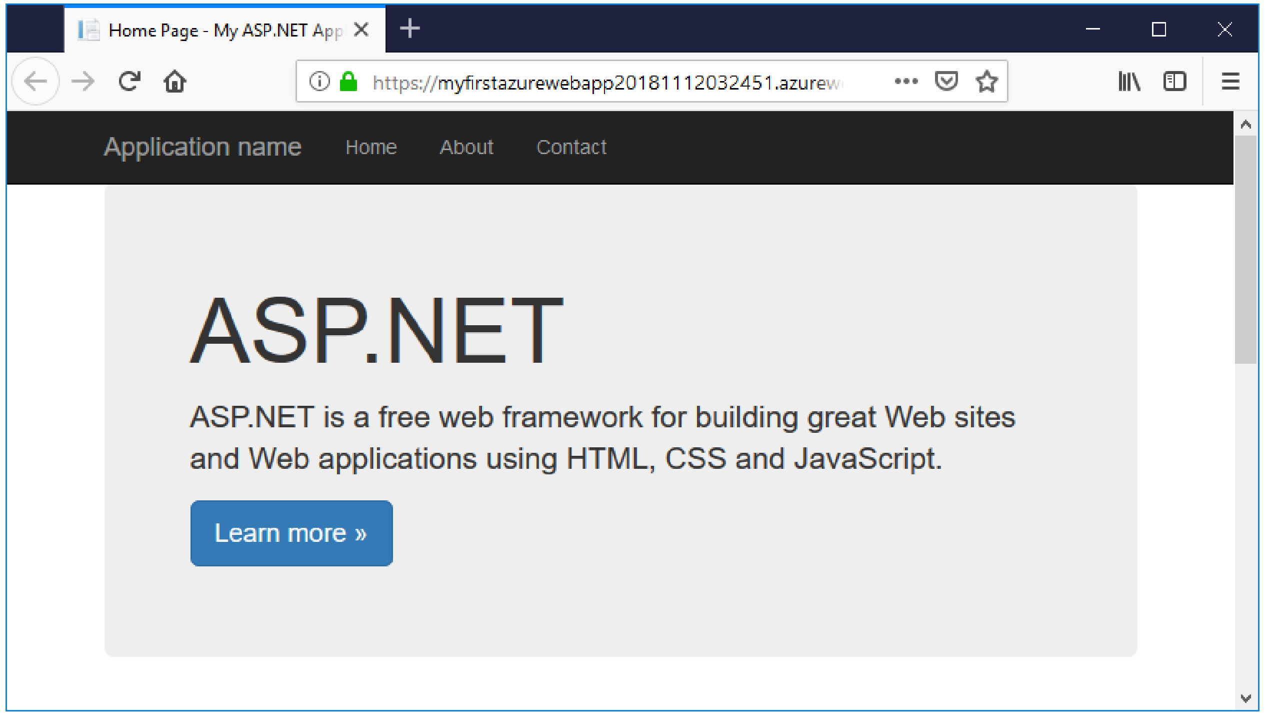
Application Insights: West Europe

Clicking the Create button will create the following Azure resources

App Service - myFirstAzureWebApp20181112033746

Create an ASP.NET Web App in Azure

Once the wizard completes, it publishes the ASP.NET web app to Azure, and then launches the app in the default browser.


Home Page - My ASP.NET App

https://myfirstazurewebapp20181112032451.azurew

ASP.NET

ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.

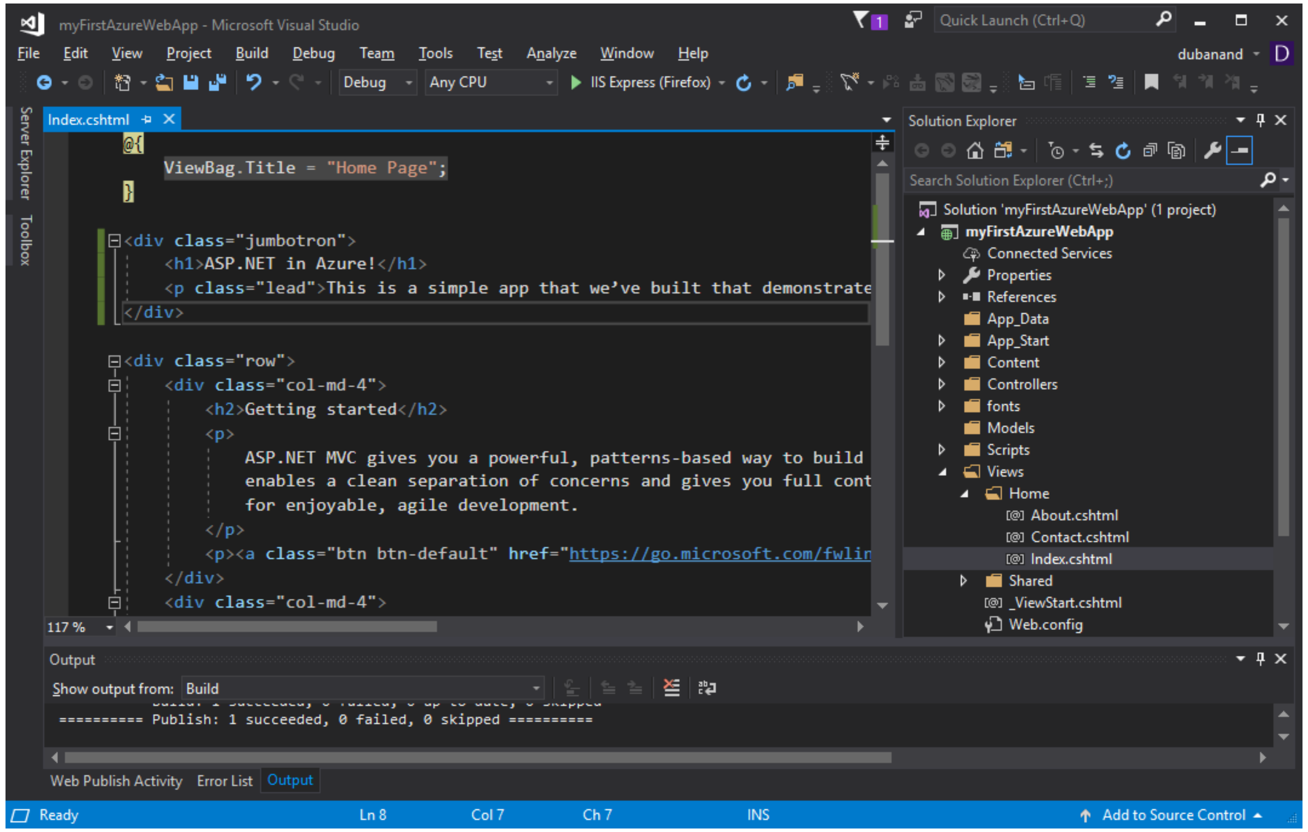
The app name specified in the create and publish step is used as the URL prefix in the format *http://<app_name>.azurewebsites.net.*

Update and Redeploy ASP.NET Web App

From the Solution Explorer, open Views\Home\Index.cshtml.

Find the <div class="jumbotron"> HTML tag near the top, and replace the entire element with the following code:

<div class="jumbotron"> <h1>ASP.NET in Azure!</h1> <p class="lead">This is a simple app that we’ve built that demonstrates how to deploy a .NET app to Azure App Service.</p> </div>



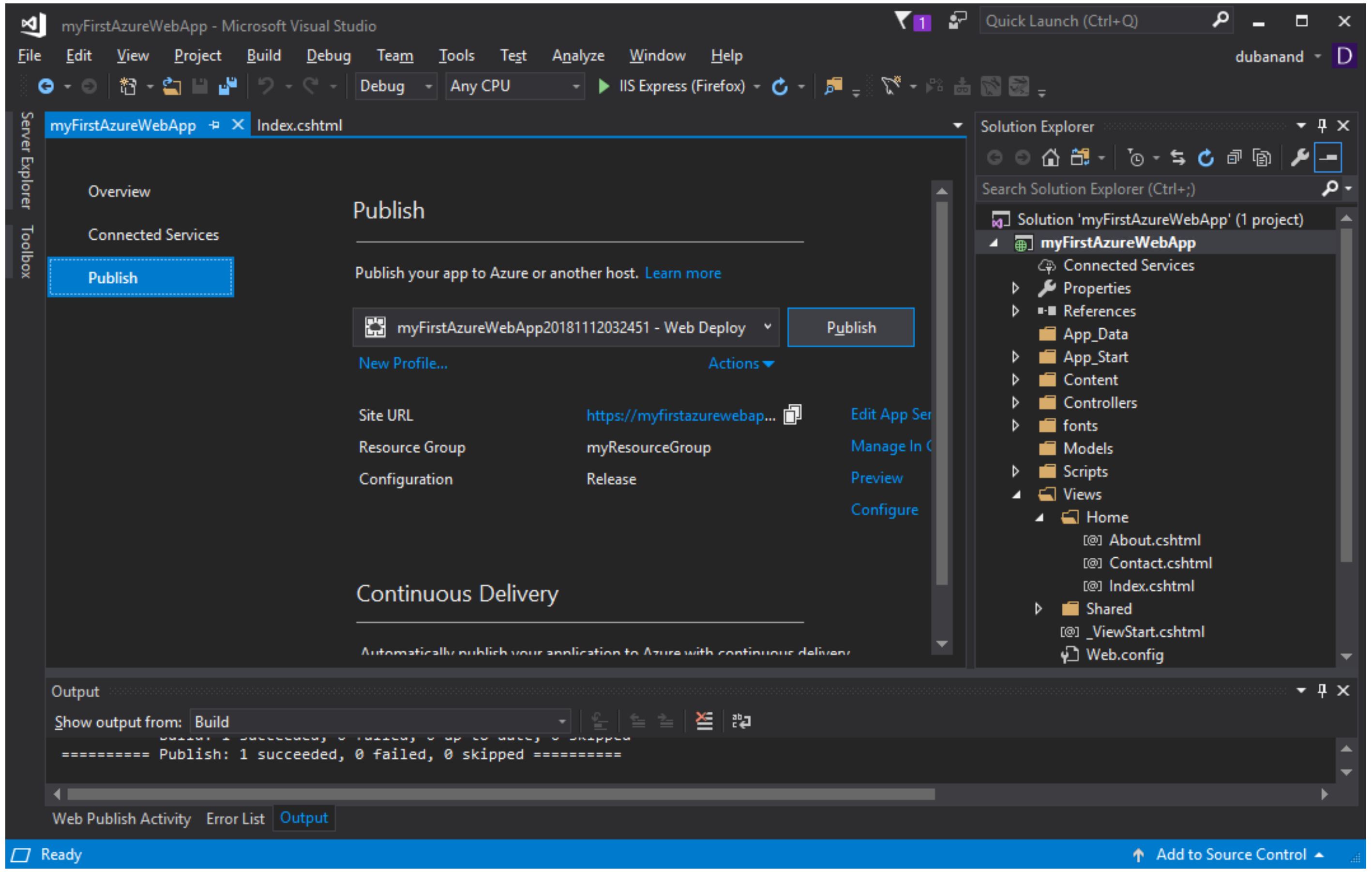
```
@{
    ViewBag.Title = "Home Page";
}

<div class="jumbotron">
    <h1>ASP.NET in Azure!</h1>
    <p class="lead">This is a simple app that we’ve built that demonstrate
</div>

<div class="row">
    <div class="col-md-4">
        <h2>Getting started</h2>
        <p>
            ASP.NET MVC gives you a powerful, patterns-based way to build
            enables a clean separation of concerns and gives you full cont
            for enjoyable, agile development.
        </p>
        <p><a class="btn btn-default" href="https://go.microsoft.com/fwlin
    </div>
    <div class="col-md-4">
```

```
========== Publish: 1 succeeded, 0 failed, 0 skipped ==========
```

Update and Redeploy ASP.NET Web App

To redeploy to Azure, right-click the **myFirstAzureWebApp** project in **Solution Explorer** and select **Publish**.

On the publish page, select **Publish**.



Publish

Publish your app to Azure or another host. Learn more

myFirstAzureWebApp20181112032451 - Web Deploy | Publish

New Profile... Actions

Site URL	https://myfirstazurewebap...	Edit App Ser
Resource Group	myResourceGroup	Manage In C
Configuration	Release	Preview
		Configure

Continuous Delivery

Automatically publish your application to Azure with continuous delivery

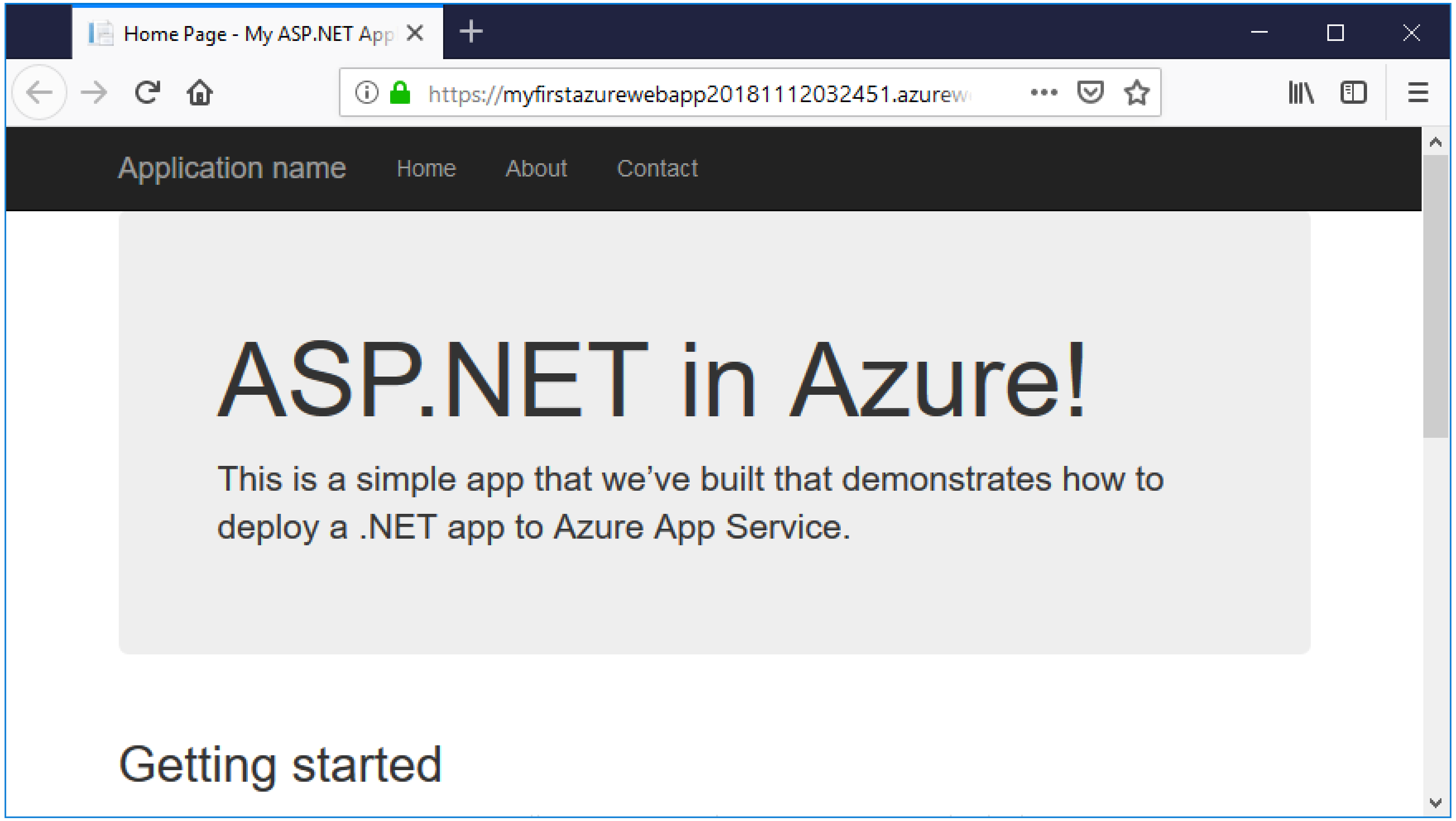
Output

Show output from: Build

```
========== Publish: 1 succeeded, 0 failed, 0 skipped ==========
```

Update and Redeploy ASP.NET Web App

When publishing completes, Visual Studio launches a browser to the URL of the web app.


Home Page - My ASP.NET App
https://myfirstazurewebapp20181112032451.azurew
Application name
Home
About
Contact

ASP.NET in Azure!

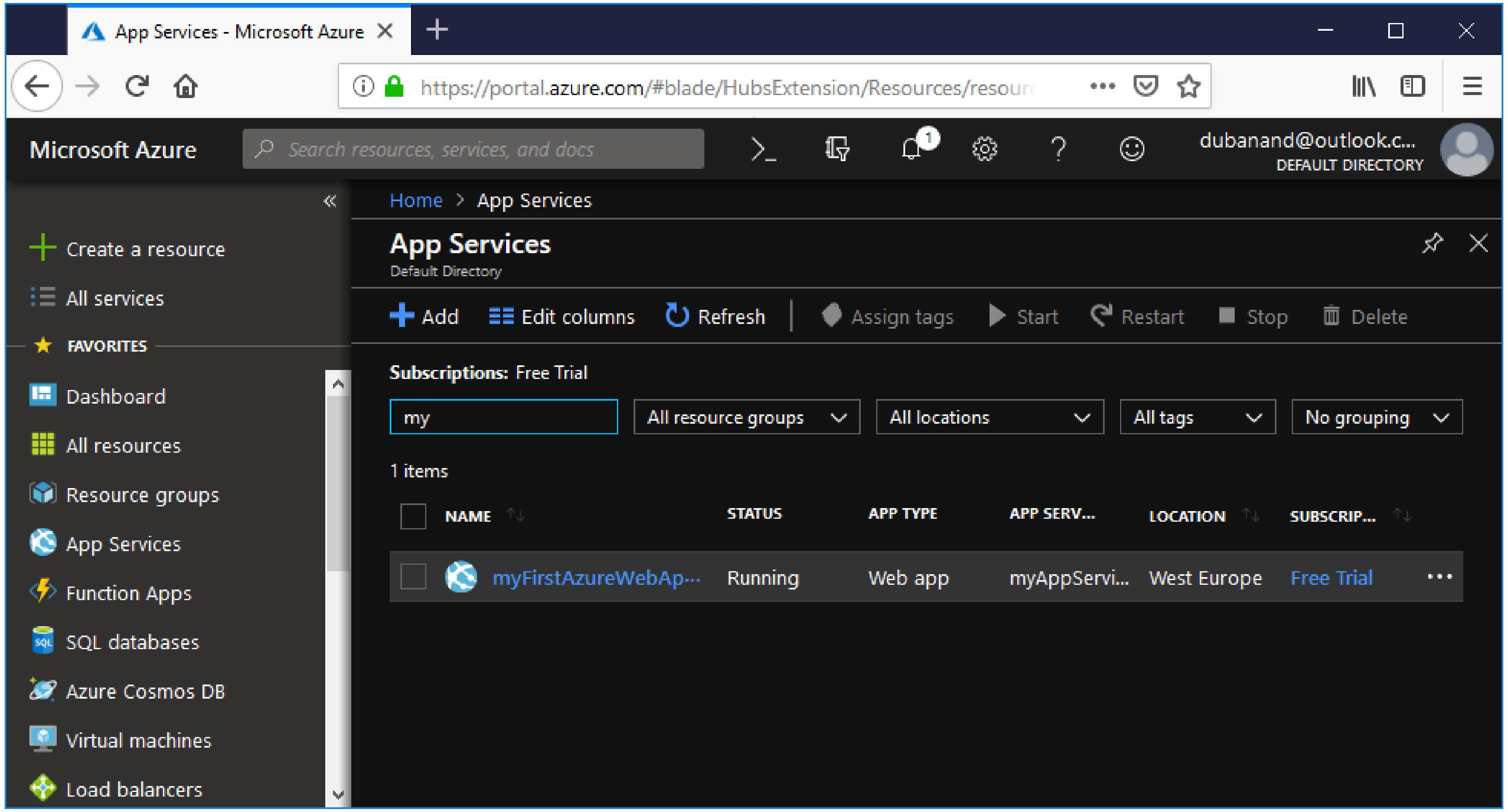
This is a simple app that we've built that demonstrates how to deploy a .NET app to Azure App Service.

Getting started

Manage the Azure Web App

Go to the Azure portal to manage the web app.

From the left menu, select **App Services**, and then select the name of your Azure web app.



Subscriptions: Free Trial

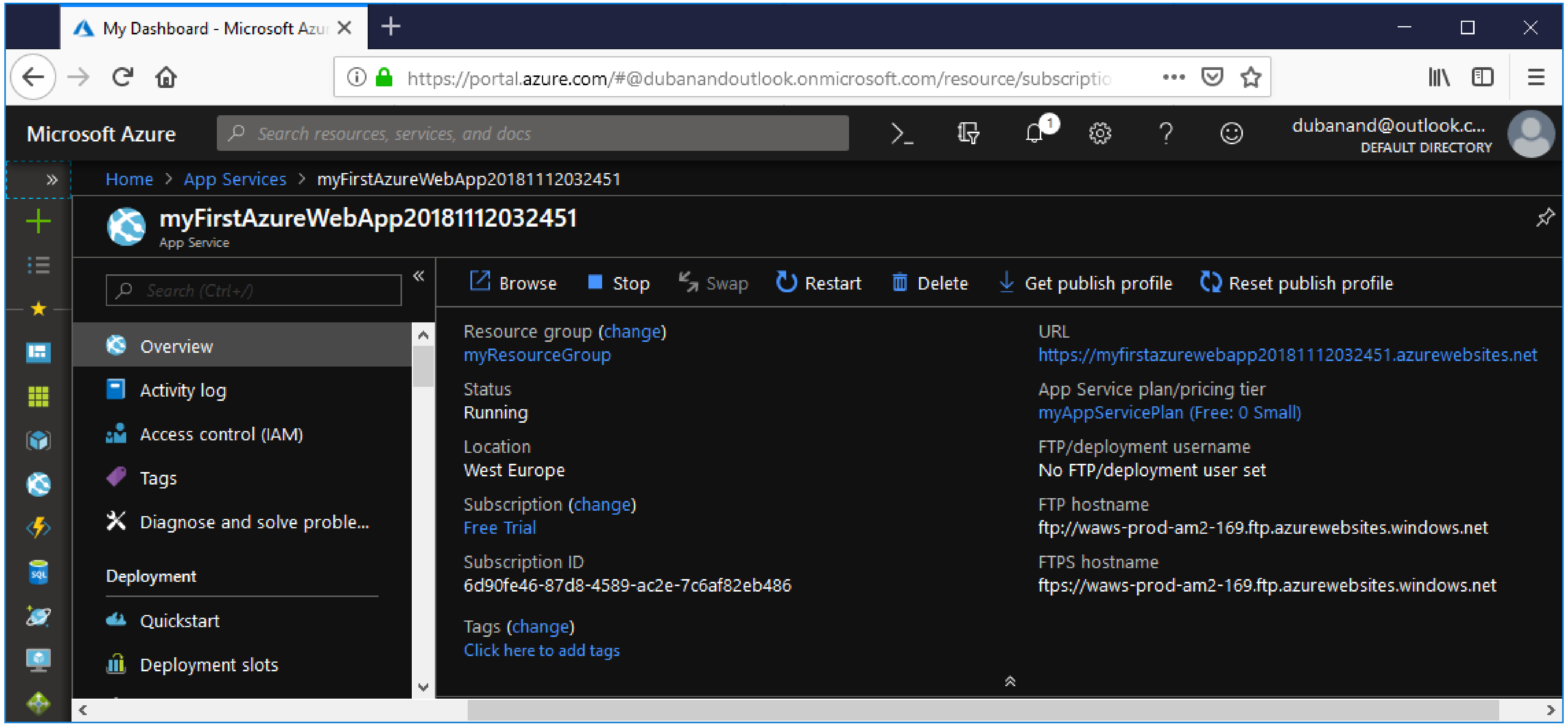
my

1 items

NAME	STATUS	APP TYPE	APP SERV...	LOCATION	SUBSCRIP...
myFirstAzureWebAp...	Running	Web app	myAppServi...	West Europe	Free Trial

Manage the Azure Web App

You see your web app's Overview page. Here, you can perform basic management tasks like browse, stop, start, restart, and delete.


My Dashboard - Microsoft Azu
https://portal.azure.com/#@dubanandoutlook.onmicrosoft.com/resource/subscriptio
Microsoft Azure
Search resources, services, and docs
dubanand@outlook.c...
DEFAULT DIRECTORY
Home > App Services > myFirstAzureWebApp20181112032451
myFirstAzureWebApp20181112032451
App Service
Search (Ctrl+/)
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve proble...
Deployment
Quickstart
Deployment slots
Browse
Stop
Swap
Restart
Delete
Get publish profile
Reset publish profile

Resource group (change)
myResourceGroup

Status
Running

Location
West Europe

Subscription (change)
Free Trial

Subscription ID
6d90fe46-87d8-4589-ac2e-7c6af82eb486

Tags (change)
Click here to add tags

URL
https://myfirstazurewebapp20181112032451.azurewebsites.net

App Service plan/pricing tier
myAppServicePlan (Free: 0 Small)

FTP/deployment username
No FTP/deployment user set

FTP hostname
ftp://waws-prod-am2-169.ftp.azurewebsites.windows.net

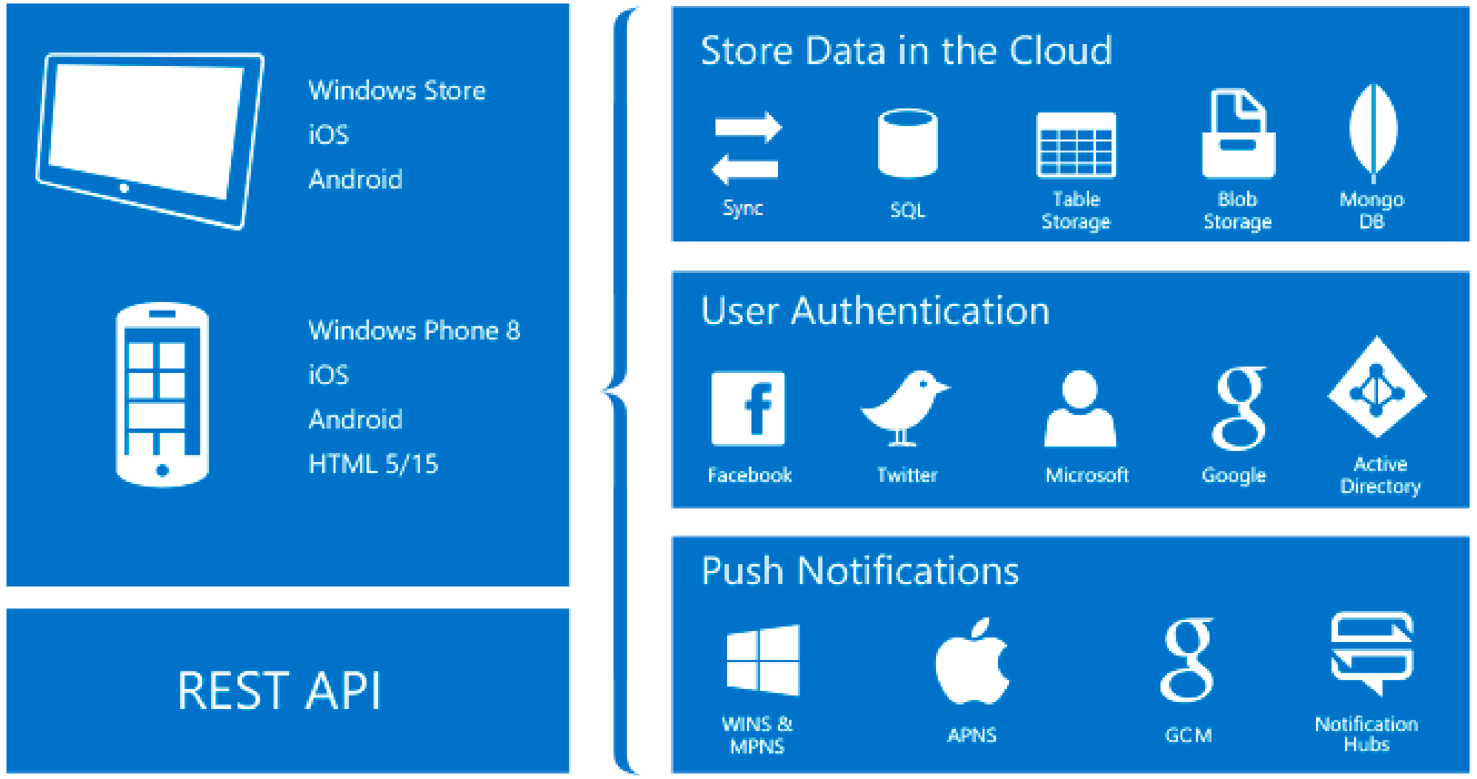
FTPS hostname
ftps://waws-prod-am2-169.ftp.azurewebsites.windows.net

App Services – Mobile Apps

Mobile Apps

Azure App Service is a fully managed platform as a service (PaaS) offering for professional developers. The service brings a rich set of capabilities to web, mobile, and integration scenarios.

The Mobile Apps feature of Azure App Service gives enterprise developers and system integrators a mobile-application development platform that's highly scalable and globally available.



Mobile Apps

Why Mobile Apps?

- **Build native and cross-platform apps**: Whether you're building native iOS, Android, and Windows apps or cross-platform Xamarin or Cordova (PhoneGap) apps, you can take advantage of App Service by using native SDKs.

- **Connect to your enterprise systems**: With the Mobile Apps feature, you can add corporate sign-in in minutes, and connect to your enterprise on-premises or cloud resources.

- **Build offline-ready apps with data sync**: Make your mobile workforce more productive by building apps that work offline, and use Mobile Apps to sync data in the background when connectivity is present with any of your enterprise data sources or software as a service APIs.

- **Push notifications to millions in seconds**: Engage your customers with instant push notifications on any device, personalized to their needs, and sent when the time is right.

App Services - API Apps

API Apps

API apps in Azure App Service offer features that make it easier to develop, host, and consume APIs in the cloud and on-premises. With API apps you get enterprise grade security, simple access control, hybrid connectivity, automatic SDK generation, and seamless integration with Logic Apps.

In simple words, it is a platform to host the Web apps with the most common API features for which you don't have to code.

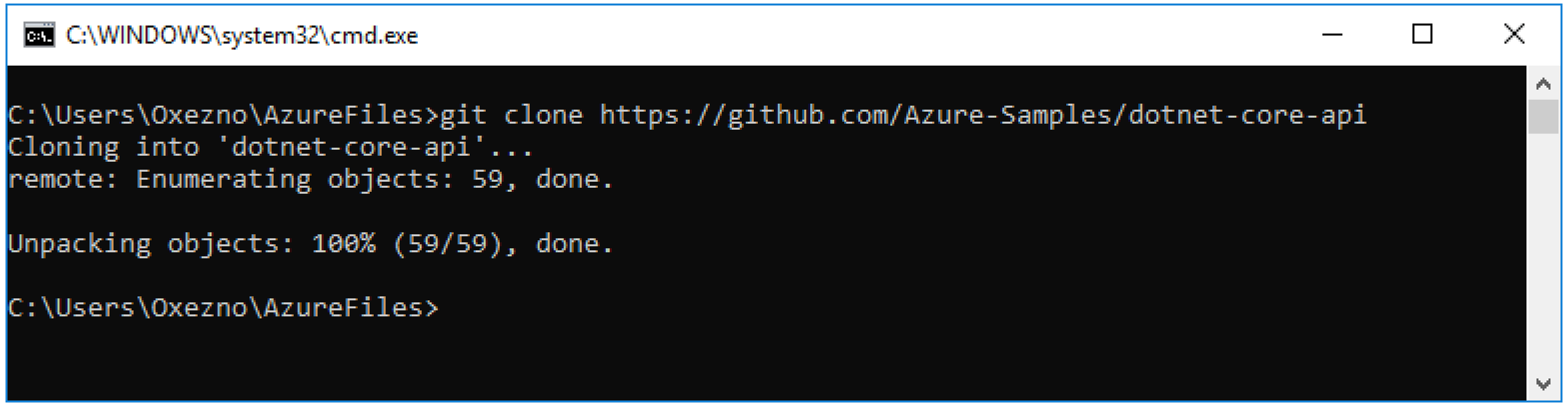
Host a RESTful API with CORS in Azure

Clone the sample application

In the terminal window, cd to a working directory.

Run the following command to clone the sample repository.

git clone https://github.com/Azure-Samples/dotnet-core-api



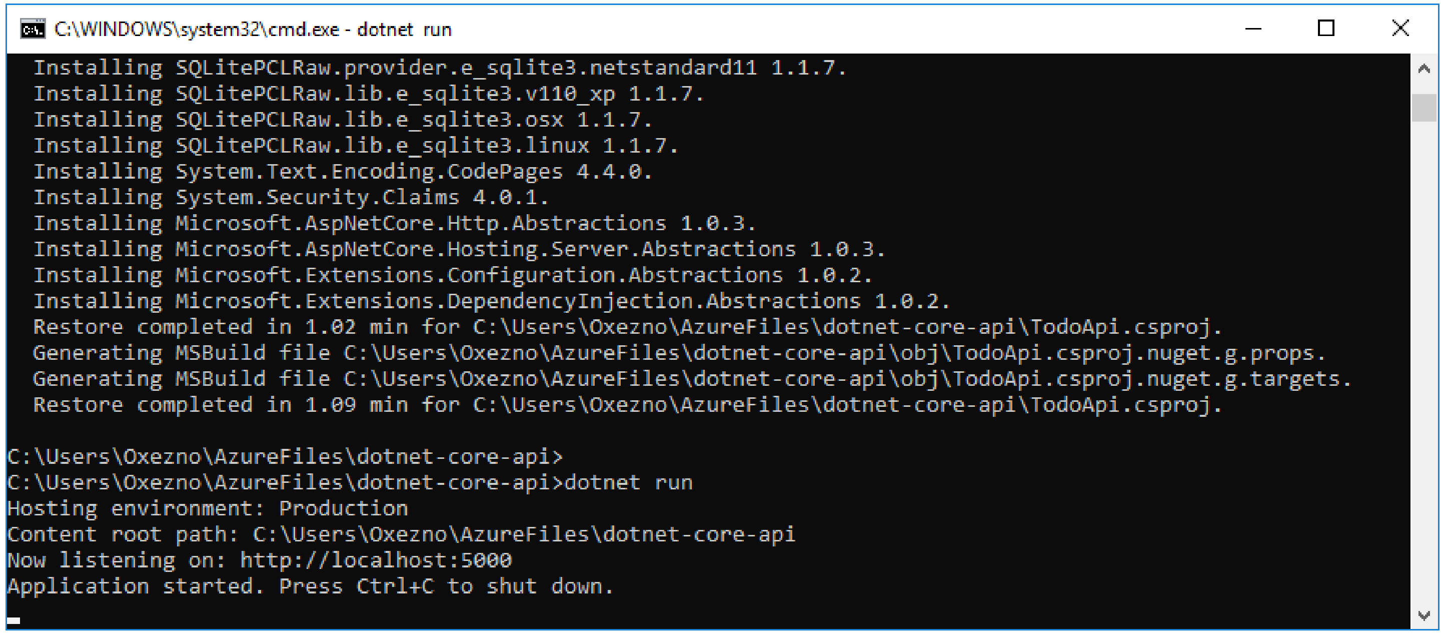
```
C:\Users\Oxezno\AzureFiles>git clone https://github.com/Azure-Samples/dotnet-core-api
Cloning into 'dotnet-core-api'...
remote: Enumerating objects: 59, done.

Unpacking objects: 100% (59/59), done.

C:\Users\Oxezno\AzureFiles>
```

Host a RESTful API with CORS in Azure

Run the application

Run the following commands to install the required packages, run database migrations, and start the application.

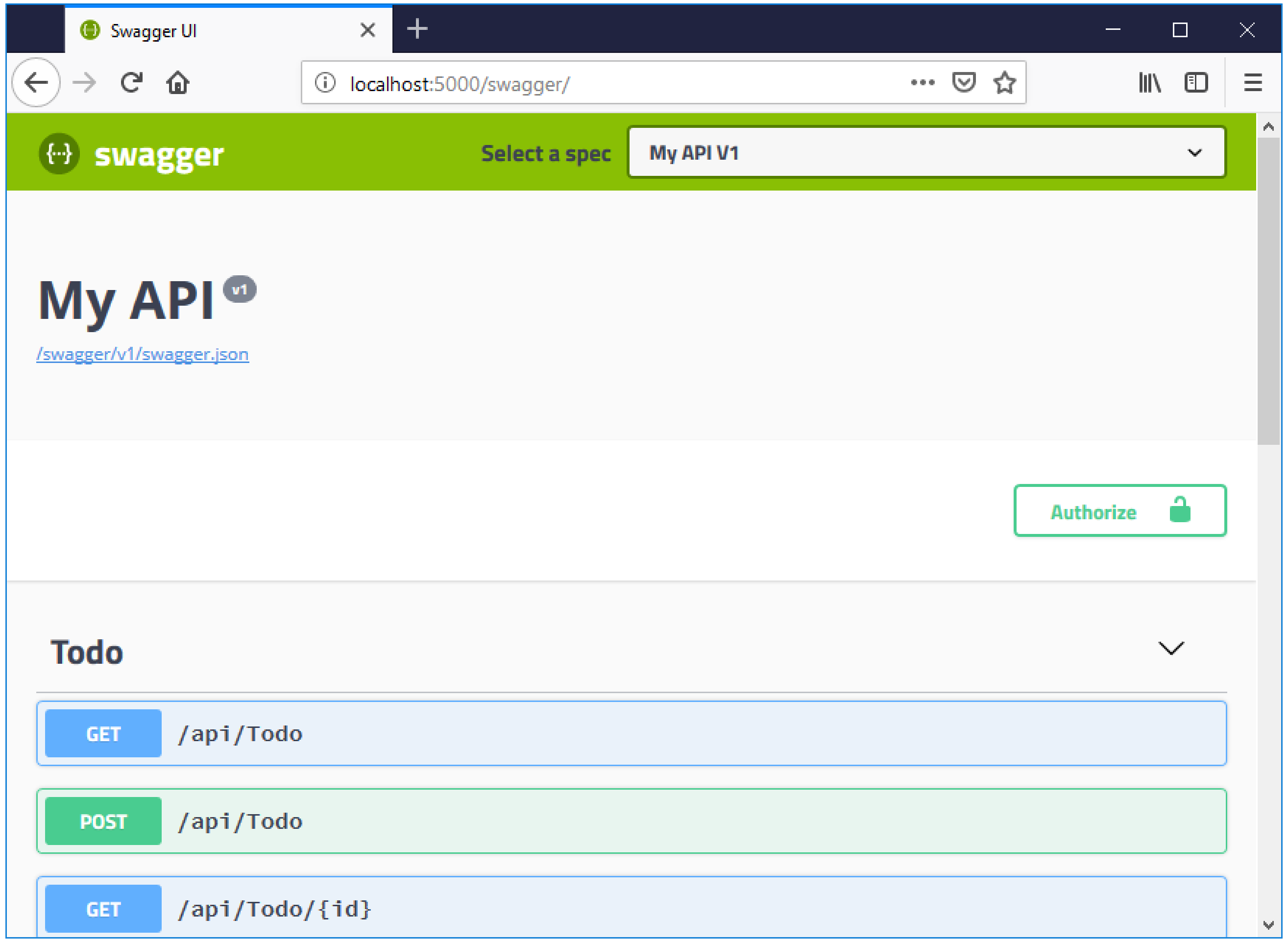
- *cd dotnet-core-api*
- *dotnet restore*
- *dotnet run*



```
  Installing SQLitePCLRaw.provider.e_sqlite3.netstandard11 1.1.7.
  Installing SQLitePCLRaw.lib.e_sqlite3.v110_xp 1.1.7.
  Installing SQLitePCLRaw.lib.e_sqlite3.osx 1.1.7.
  Installing SQLitePCLRaw.lib.e_sqlite3.linux 1.1.7.
  Installing System.Text.Encoding.CodePages 4.4.0.
  Installing System.Security.Claims 4.0.1.
  Installing Microsoft.AspNetCore.Http.Abstractions 1.0.3.
  Installing Microsoft.AspNetCore.Hosting.Server.Abstractions 1.0.3.
  Installing Microsoft.Extensions.Configuration.Abstractions 1.0.2.
  Installing Microsoft.Extensions.DependencyInjection.Abstractions 1.0.2.
  Restore completed in 1.02 min for C:\Users\Oxezno\AzureFiles\dotnet-core-api\TodoApi.csproj.
  Generating MSBuild file C:\Users\Oxezno\AzureFiles\dotnet-core-api\obj\TodoApi.csproj.nuget.g.props.
  Generating MSBuild file C:\Users\Oxezno\AzureFiles\dotnet-core-api\obj\TodoApi.csproj.nuget.g.targets.
  Restore completed in 1.09 min for C:\Users\Oxezno\AzureFiles\dotnet-core-api\TodoApi.csproj.

C:\Users\Oxezno\AzureFiles\dotnet-core-api>
C:\Users\Oxezno\AzureFiles\dotnet-core-api>dotnet run
Hosting environment: Production
Content root path: C:\Users\Oxezno\AzureFiles\dotnet-core-api
Now listening on: http://localhost:5000
Application started. Press Ctrl+C to shut down.
_
```

Host a RESTful API with CORS in Azure

Navigate to *http://localhost:5000/swagger* in a browser to play with the Swagger UI.



My API v1

/swagger/v1/swagger.json

Todo

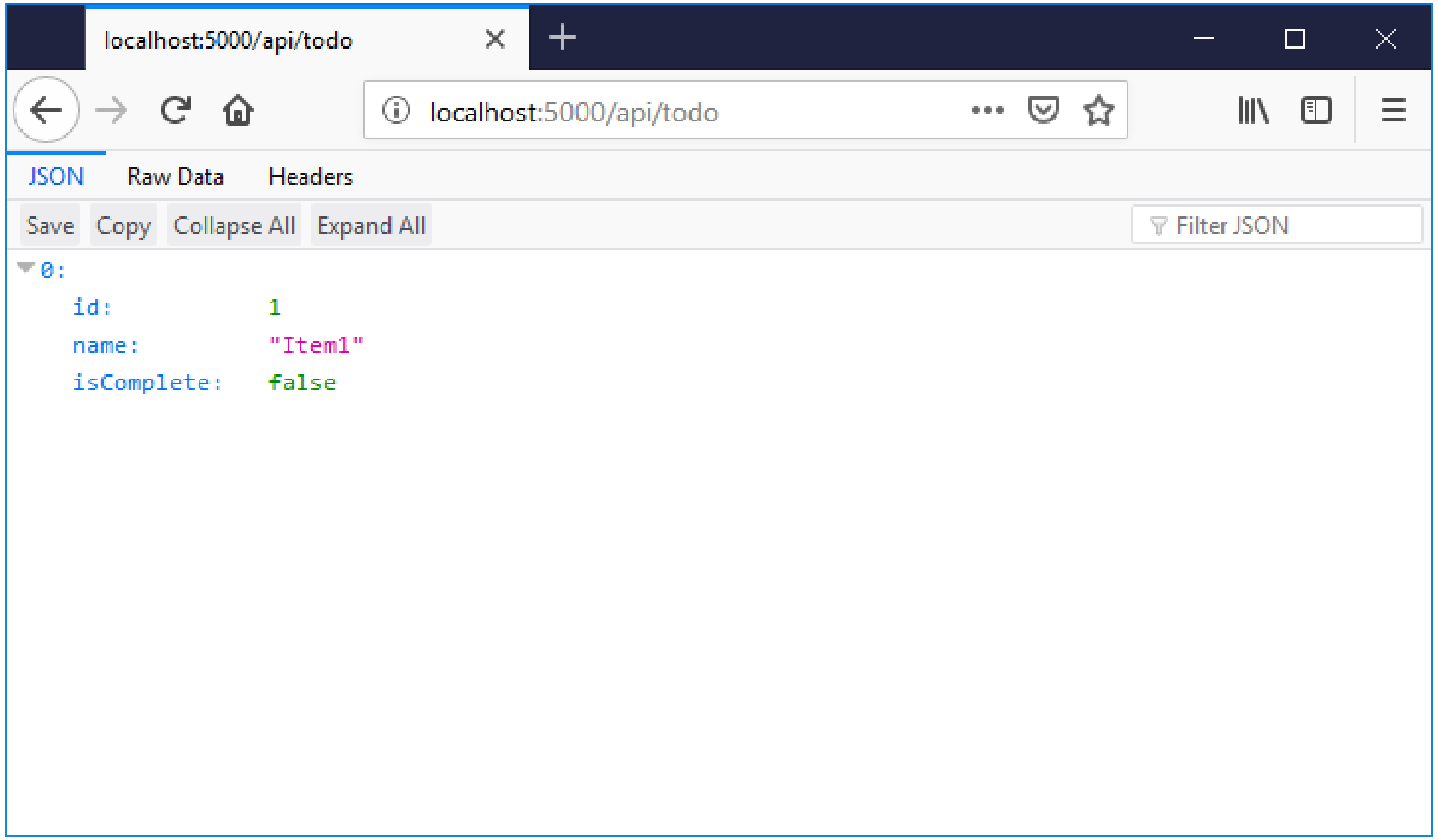
GET /api/Todo

POST /api/Todo

GET /api/Todo/{id}

Host a RESTful API with CORS in Azure

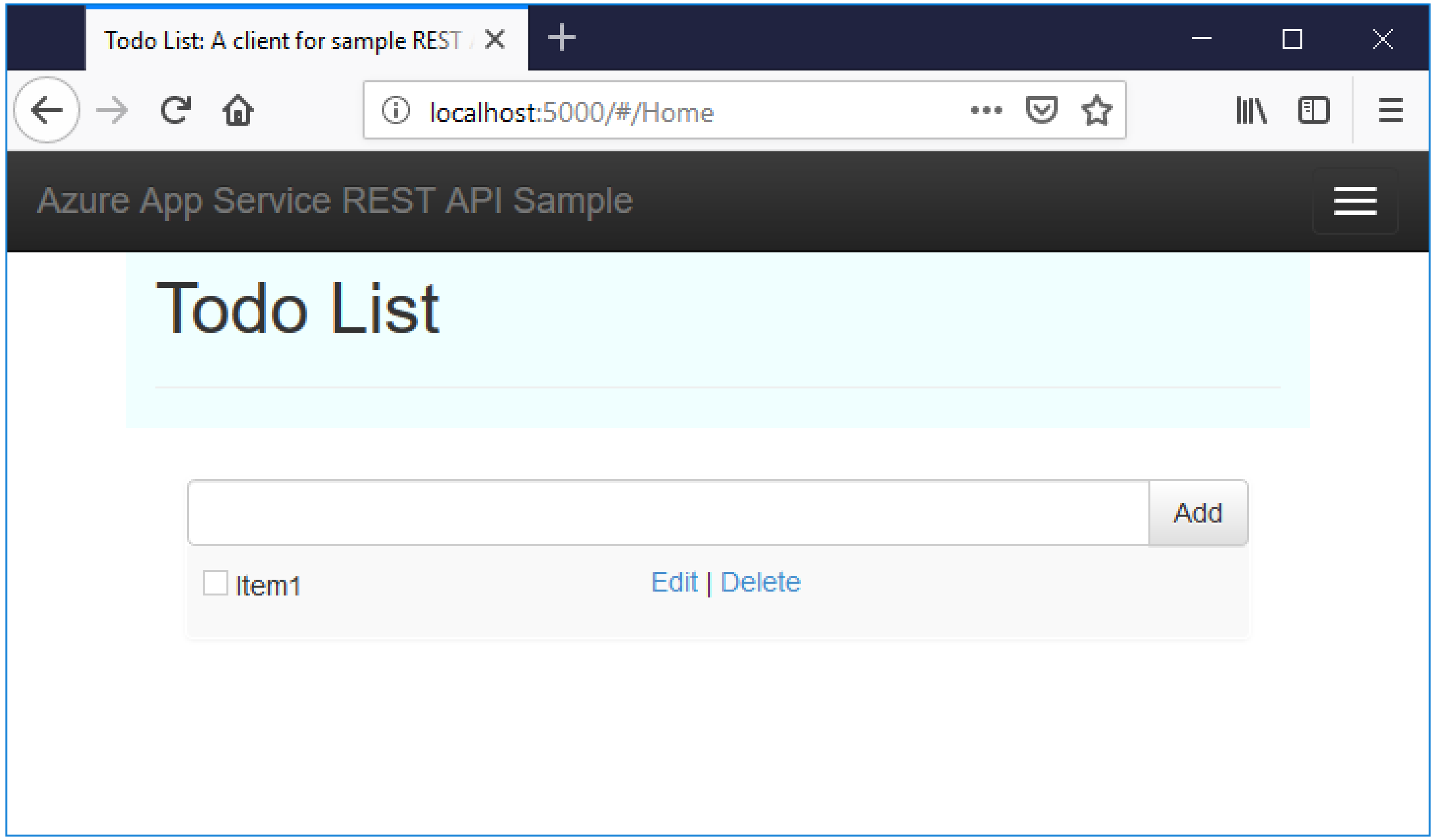
Navigate to *http://localhost:5000/api/todo* and see a list of ToDo JSON items.



```
0:
    id:          1
    name:        "Item1"
    isComplete:  false
```

Host a RESTful API with CORS in Azure

Navigate to *http://localhost:5000* and play with the browser app. Later, you will point the browser app to a remote API in App Service to test CORS functionality. Code for the browser app is found in the repository's www root directory.


Todo List: A client for sample REST
localhost:5000/#/Home
Azure App Service REST API Sample
Add
Edit | Delete

Todo List

Item1

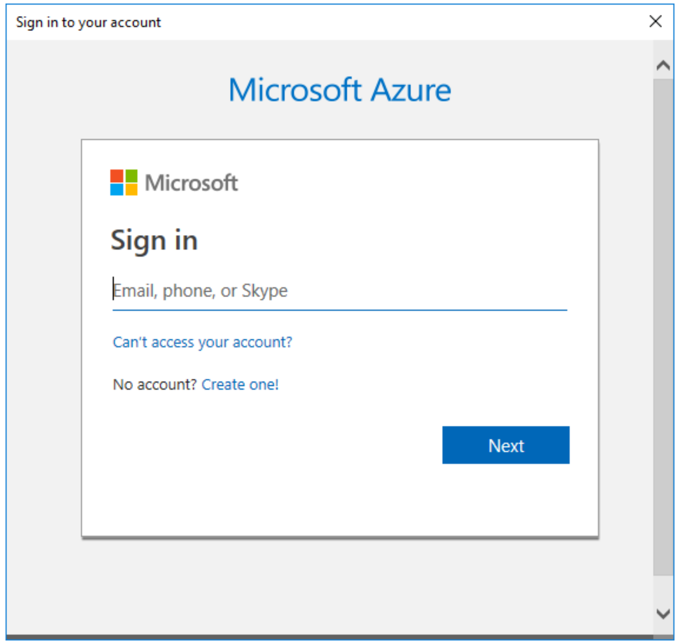
Host a RESTful API with CORS in Azure

Connect to Your Azure Account

Enter the following cmdlet in PowerShell.

```
Connect-AzureRmAccount
```

The screen will pop up and ask for credentials of your account. Enter the credentials and sign in.



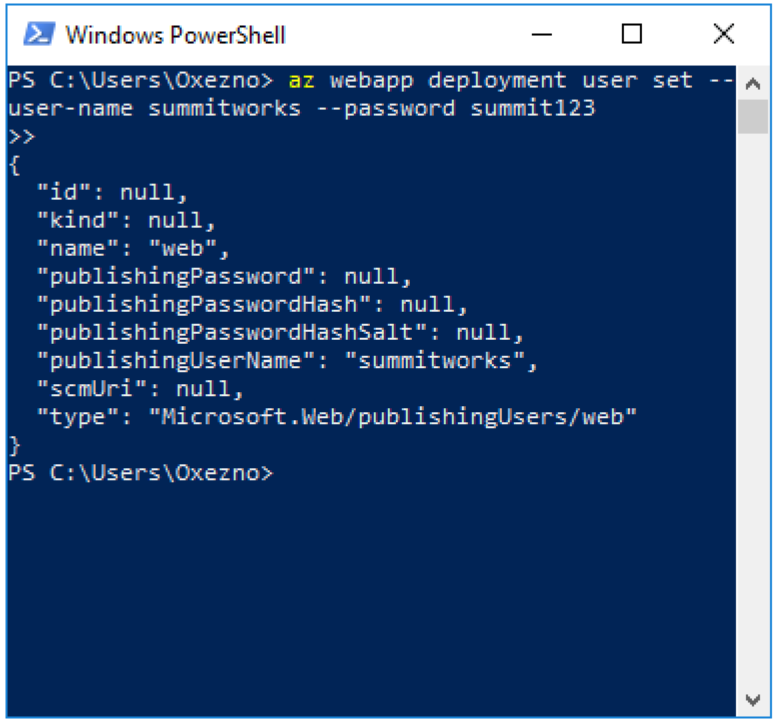
Host a RESTful API with CORS in Azure

Configure local git deployment

In the Cloud Shell, create deployment credentials with the *az webapp deployment user set* command. This deployment user is required for FTP and local Git deployment to a web app. The user name and password are account level. They are different from your Azure subscription credentials.

> *az webapp deployment user set --user-name <username> --password <password>*

You should get a JSON output, with the password shown as null. If you get a 'Conflict'. Details: 409 error, change the username. If you get a 'Bad Request'. Details: 400 error, use a stronger password.



Windows PowerShell

```
PS C:\Users\Oxezno> az webapp deployment user set --
user-name summitworks --password summit123
>>
{
  "id": null,
  "kind": null,
  "name": "web",
  "publishingPassword": null,
  "publishingPasswordHash": null,
  "publishingPasswordHashSalt": null,
  "publishingUserName": "summitworks",
  "scmUri": null,
  "type": "Microsoft.Web/publishingUsers/web"
}
PS C:\Users\Oxezno>
```

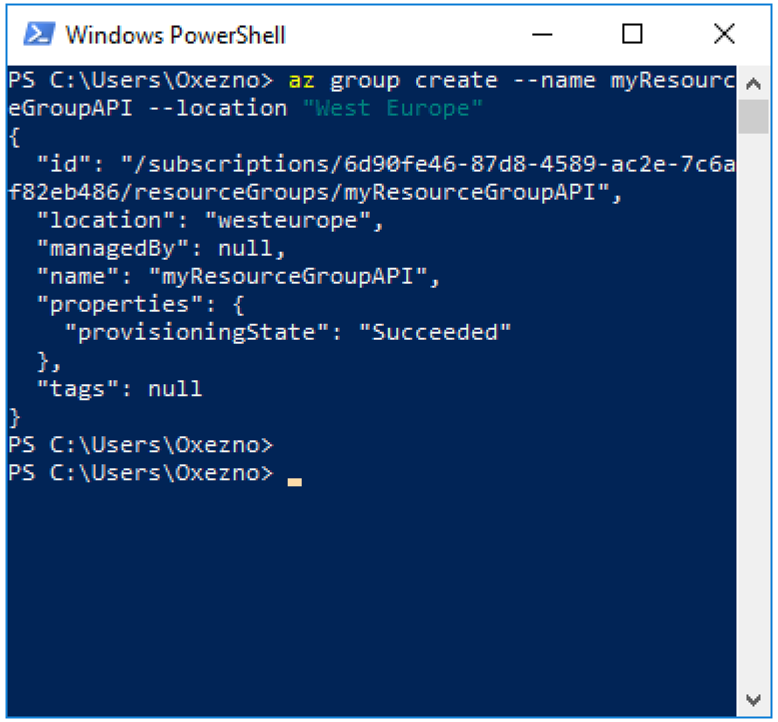
Host a RESTful API with CORS in Azure

Create a resource group

A resource group is a logical container into which Azure resources like web apps, databases, and storage accounts are deployed and managed. For example, you can choose to delete the entire resource group in one simple step later.

In the Cloud Shell, create a resource group with the az group create command. The following example creates a resource group named myResourceGroup in the West Europe location. To see all supported locations for App Service in Free tier, run the az appservice list-locations --sku FREE command.

az group create --name myResourceGroup --location "West Europe"



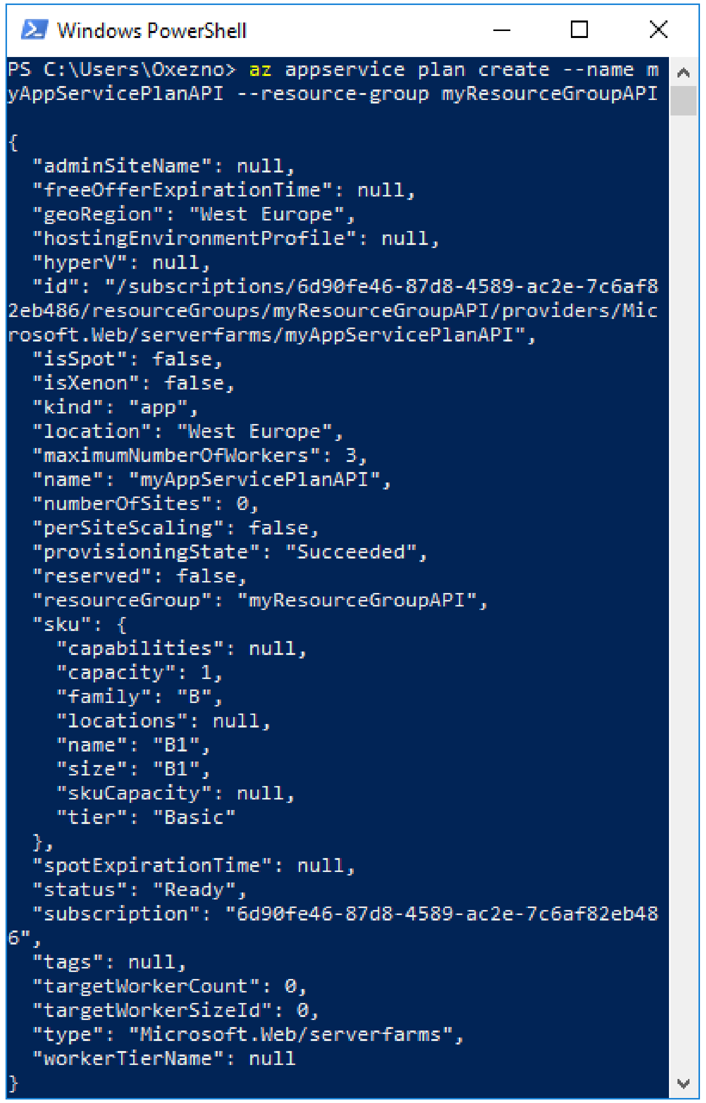
```
PS C:\Users\Oxezno> az group create --name myResourc
eGroupAPI --location "West Europe"
{
  "id": "/subscriptions/6d90fe46-87d8-4589-ac2e-7c6a
f82eb486/resourceGroups/myResourceGroupAPI",
  "location": "westeurope",
  "managedBy": null,
  "name": "myResourceGroupAPI",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null
}
PS C:\Users\Oxezno>
PS C:\Users\Oxezno> _
```

Host a RESTful API with CORS in Azure

Create an App Service plan

The following example creates an App Service plan named myAppServicePlan in the Free pricing tier:

az appservice plan create --name myAppServicePlan --resource-group myResourceGroup --sku FREE



Windows PowerShell

```
PS C:\Users\Oxezno> az appservice plan create --name m
yAppServicePlanAPI --resource-group myResourceGroupAPI

{
  "adminSiteName": null,
  "freeOfferExpirationTime": null,
  "geoRegion": "West Europe",
  "hostingEnvironmentProfile": null,
  "hyperV": null,
  "id": "/subscriptions/6d90fe46-87d8-4589-ac2e-7c6af8
2eb486/resourceGroups/myResourceGroupAPI/providers/Mic
rosoft.Web/serverfarms/myAppServicePlanAPI",
  "isSpot": false,
  "isXenon": false,
  "kind": "app",
  "location": "West Europe",
  "maximumNumberOfWorkers": 3,
  "name": "myAppServicePlanAPI",
  "numberOfSites": 0,
  "perSiteScaling": false,
  "provisioningState": "Succeeded",
  "reserved": false,
  "resourceGroup": "myResourceGroupAPI",
  "sku": {
    "capabilities": null,
    "capacity": 1,
    "family": "B",
    "locations": null,
    "name": "B1",
    "size": "B1",
    "skuCapacity": null,
    "tier": "Basic"
  },
  "spotExpirationTime": null,
  "status": "Ready",
  "subscription": "6d90fe46-87d8-4589-ac2e-7c6af82eb48
6",
  "tags": null,
  "targetWorkerCount": 0,
  "targetWorkerSizeId": 0,
  "type": "Microsoft.Web/serverfarms",
  "workerTierName": null
}
```

Host a RESTful API with CORS in Azure

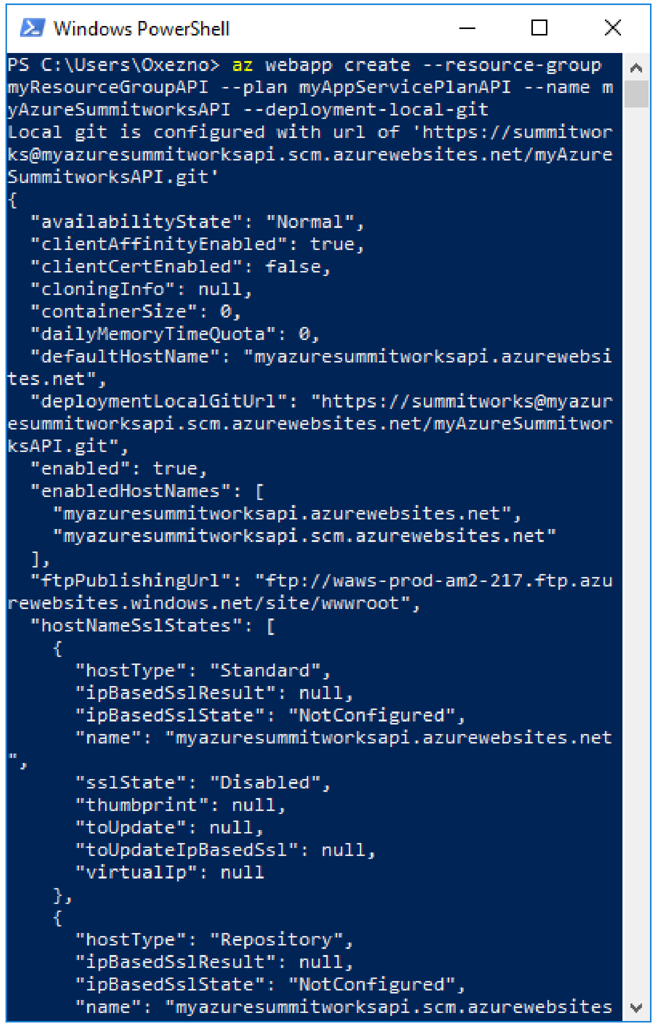
Create a web app

In the Cloud Shell, you can use the az webapp create command.

- *az webapp create --resource-group myResourceGroup --plan myAppServicePlan --name <app_name> --deployment-local-git*

Note

The URL of the Git remote is shown in the deploymentLocalGitUrl property, with the format https://<username>@<app_name>.scm.azurewebsites.net/<app_name>.git. Save this URL as you need it later.



Windows PowerShell

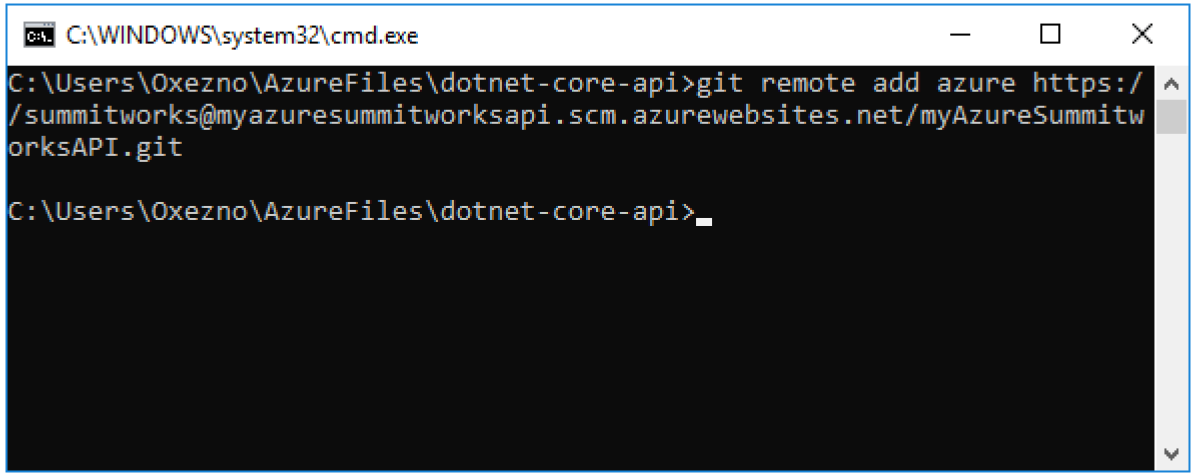
```
PS C:\Users\Oxezno> az webapp create --resource-group
myResourceGroupAPI --plan myAppServicePlanAPI --name m
yAzureSummitworksAPI --deployment-local-git
Local git is configured with url of 'https://summitwor
ks@myazuresummitworksapi.scm.azurewebsites.net/myAzure
SummitworksAPI.git'
{
  "availabilityState": "Normal",
  "clientAffinityEnabled": true,
  "clientCertEnabled": false,
  "cloningInfo": null,
  "containerSize": 0,
  "dailyMemoryTimeQuota": 0,
  "defaultHostName": "myazuresummitworksapi.azurewebsi
tes.net",
  "deploymentLocalGitUrl": "https://summitworks@myazur
esummitworksapi.scm.azurewebsites.net/myAzureSummitwor
ksAPI.git",
  "enabled": true,
  "enabledHostNames": [
    "myazuresummitworksapi.azurewebsites.net",
    "myazuresummitworksapi.scm.azurewebsites.net"
  ],
  "ftpPublishingUrl": "ftp://waws-prod-am2-217.ftp.azu
rewebsites.windows.net/site/wwwroot",
  "hostNameSslStates": [
    {
      "hostType": "Standard",
      "ipBasedSslResult": null,
      "ipBasedSslState": "NotConfigured",
      "name": "myazuresummitworksapi.azurewebsites.net
",
      "sslState": "Disabled",
      "thumbprint": null,
      "toUpdate": null,
      "toUpdateIpBasedSsl": null,
      "virtualIp": null
    },
    {
      "hostType": "Repository",
      "ipBasedSslResult": null,
      "ipBasedSslState": "NotConfigured",
      "name": "myazuresummitworksapi.scm.azurewebsites
```

Host a RESTful API with CORS in Azure

Push to Azure from Git

Back in the *local terminal window*, add an Azure remote to your local Git repository. Replace *<deploymentLocalGitUrl-from-create-step>* with the URL of the Git remote that you saved from Create a web app.

git remote add azure <deploymentLocalGitUrl-from-create-step>

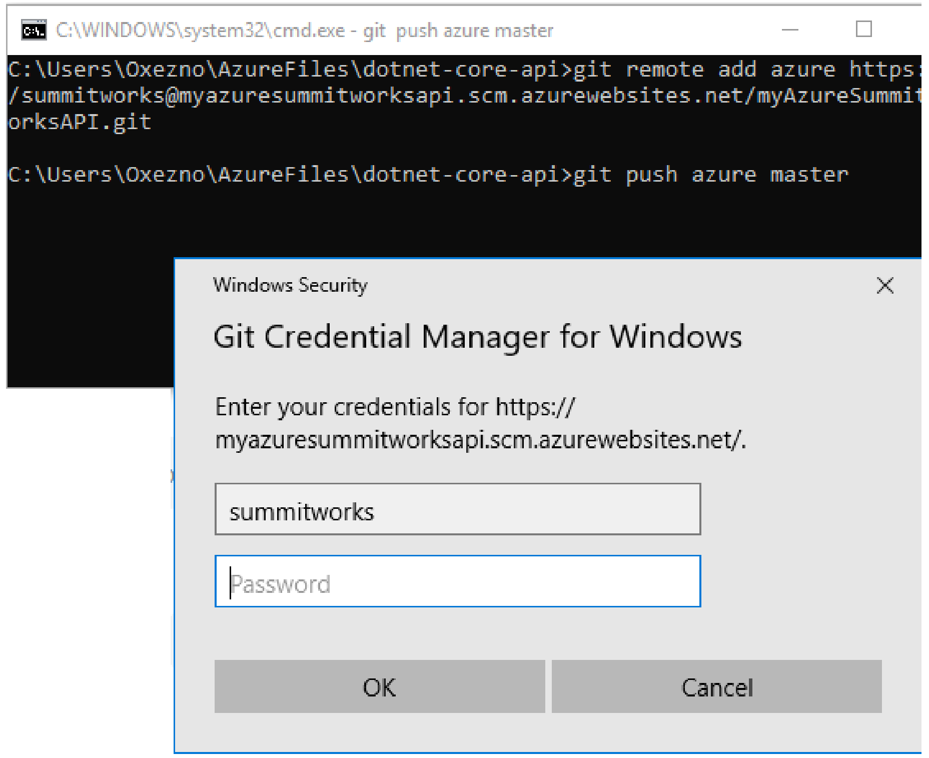


```
C:\Users\Oxezno\AzureFiles\dotnet-core-api>git remote add azure https:/
/summitworks@myazuresummitworksapi.scm.azurewebsites.net/myAzureSummitw
orksAPI.git

C:\Users\Oxezno\AzureFiles\dotnet-core-api>_
```

Host a RESTful API with CORS in Azure

Push to the Azure remote to deploy your app with the following command. When prompted for credentials by Git Credential Manager, make sure that you enter the credentials you created in Configure a deployment user, not the credentials you use to sign in to the Azure portal.

git push azure master



C:\WINDOWS\system32\cmd.exe - git push azure master

```
C:\Users\Oxezno\AzureFiles\dotnet-core-api>git remote add azure https:
/summitworks@myazuresummitworksapi.scm.azurewebsites.net/myAzureSummit
orksAPI.git

C:\Users\Oxezno\AzureFiles\dotnet-core-api>git push azure master
```

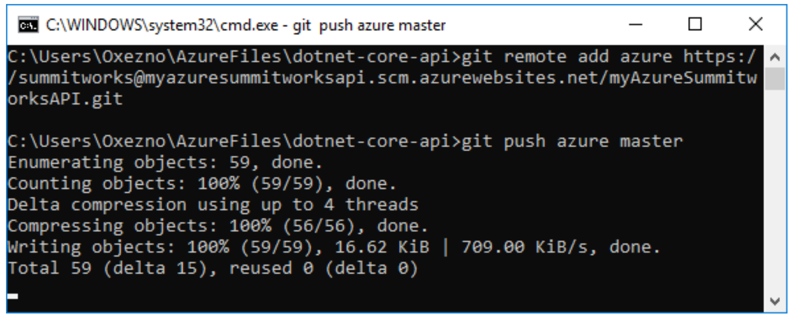
Windows Security

Git Credential Manager for Windows

Enter your credentials for https://myazuresummitworksapi.scm.azurewebsites.net/.

summitworks

Password



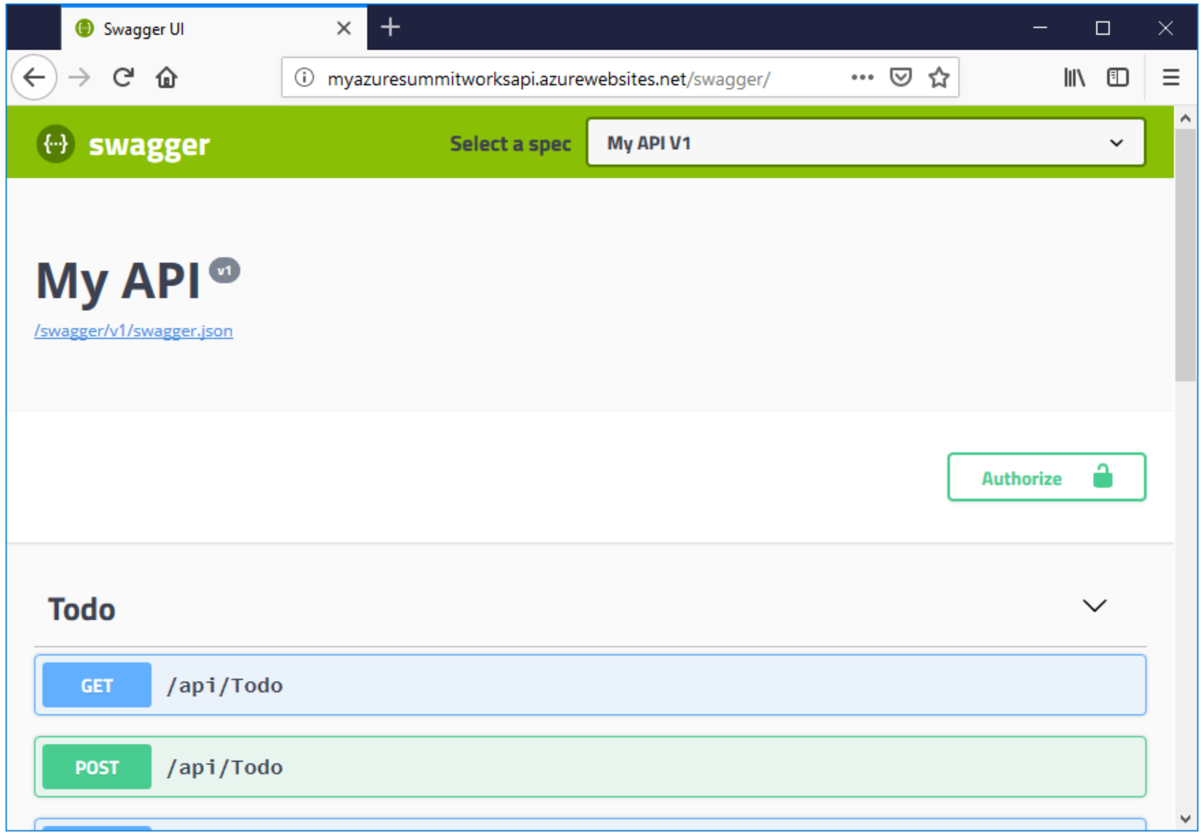
C:\WINDOWS\system32\cmd.exe - git push azure master

```
C:\Users\Oxezno\AzureFiles\dotnet-core-api>git remote add azure https:/
/summitworks@myazuresummitworksapi.scm.azurewebsites.net/myAzureSummitw
orksAPI.git

C:\Users\Oxezno\AzureFiles\dotnet-core-api>git push azure master
Enumerating objects: 59, done.
Counting objects: 100% (59/59), done.
Delta compression using up to 4 threads
Compressing objects: 100% (56/56), done.
Writing objects: 100% (59/59), 16.62 KiB | 709.00 KiB/s, done.
Total 59 (delta 15), reused 0 (delta 0)
```

Host a RESTful API with CORS in Azure

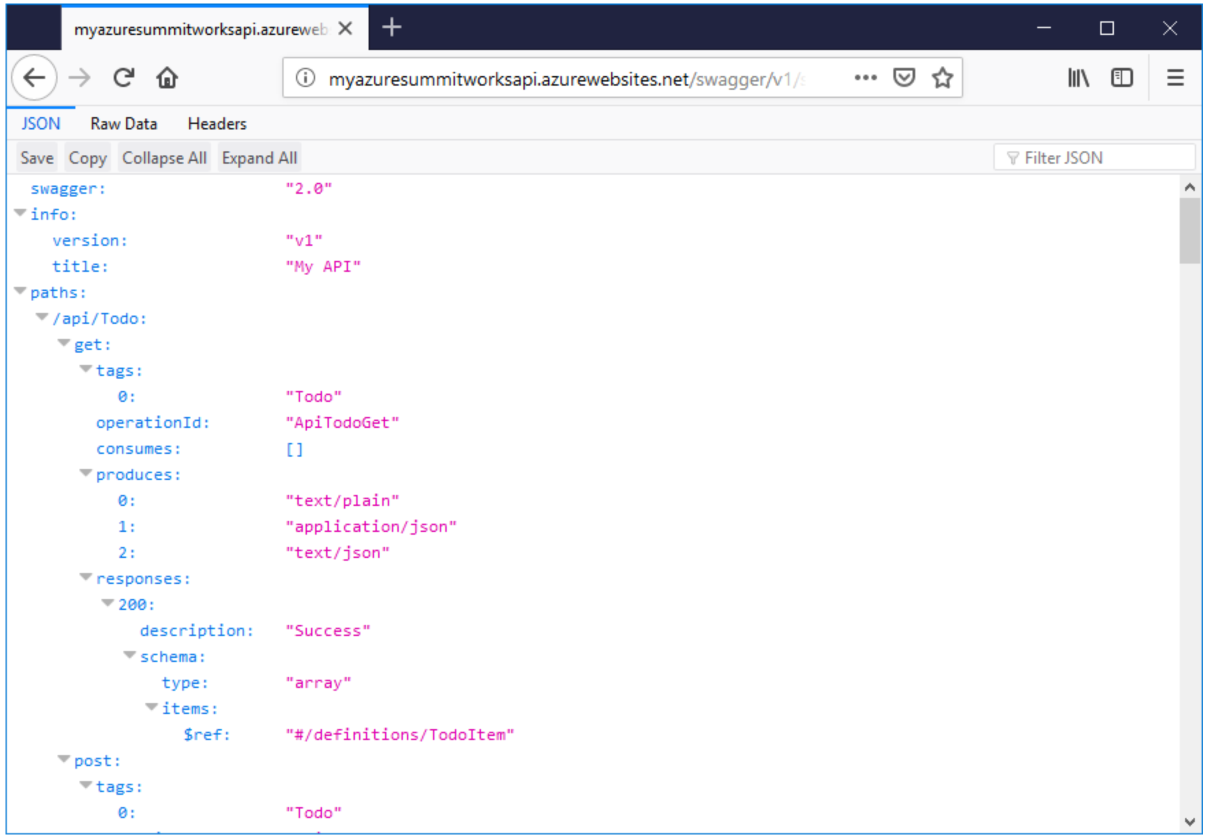
Browse to the Azure web app

Navigate to *http://<app_name>.azurewebsites.net/swagger* in a browser and play with the Swagger UI.



Host a RESTful API with CORS in Azure

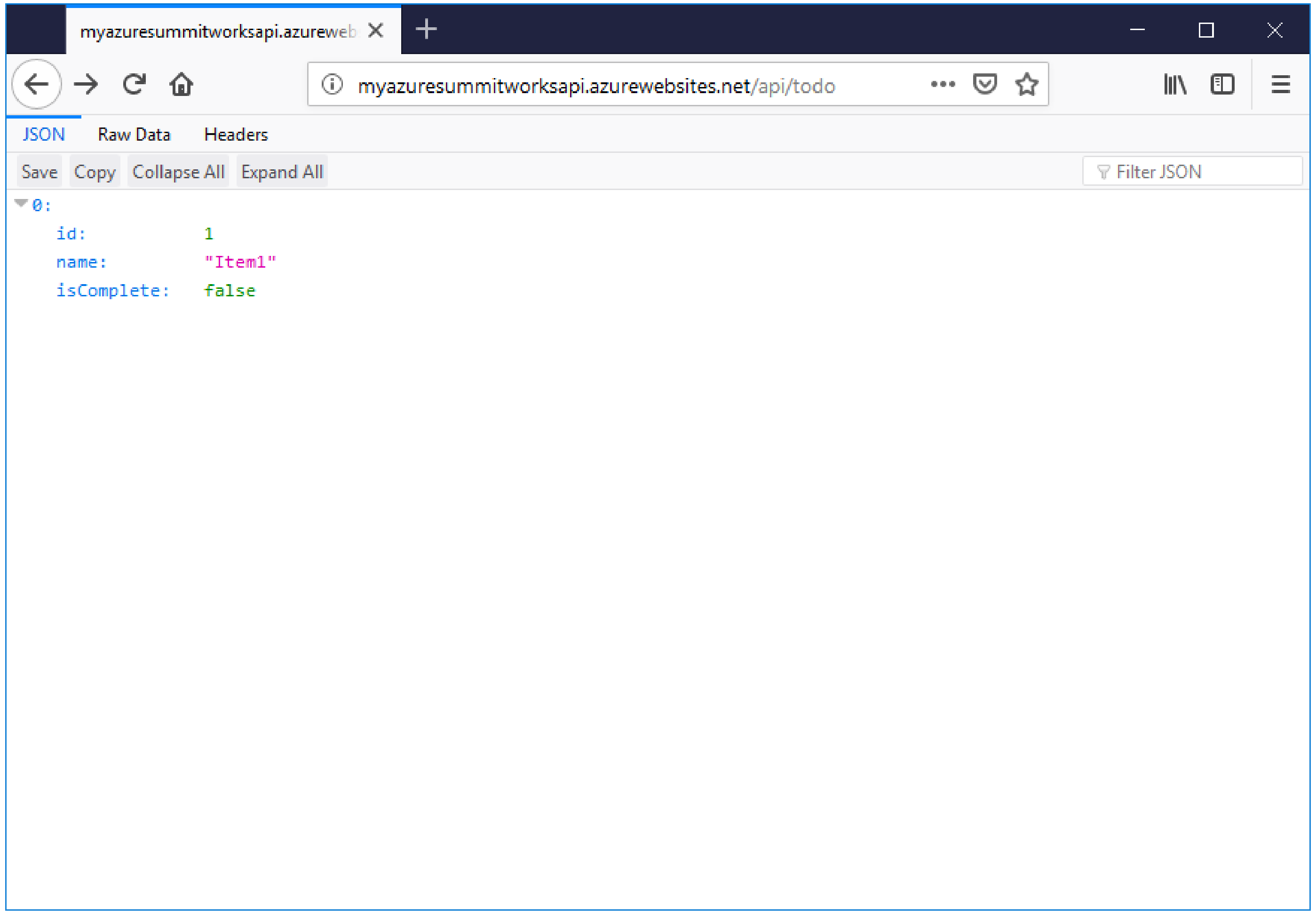
Navigate *to http://<app_name>.azurewebsites.net/swagger/v1/swagger.json* to see the swagger.json for your deployed API.


myazuresummitworksapi.azureweb
myazuresummitworksapi.azurewebsites.net/swagger/v1/
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON

```
swagger:                    "2.0"
info:
  version:                  "v1"
  title:                    "My API"
paths:
  /api/Todo:
    get:
      tags:
        0:                  "Todo"
      operationId:          "ApiTodoGet"
      consumes:             []
      produces:
        0:                  "text/plain"
        1:                  "application/json"
        2:                  "text/json"
      responses:
        200:
          description:      "Success"
          schema:
            type:           "array"
            items:
              $ref:         "#/definitions/TodoItem"
    post:
      tags:
        0:                  "Todo"
```

Host a RESTful API with CORS in Azure

Navigate to *http://<app_name>.azurewebsites.net/api/todo* to see your deployed API working.


myazuresummitworksapi.azureweb
myazuresummitworksapi.azurewebsites.net/api/todo
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON

```
0:
  id:          1
  name:        "Item1"
  isComplete:  false
```

Host a RESTful API with CORS in Azure

Add CORS functionality

Next, you enable the built-in CORS support in App Service for your API.

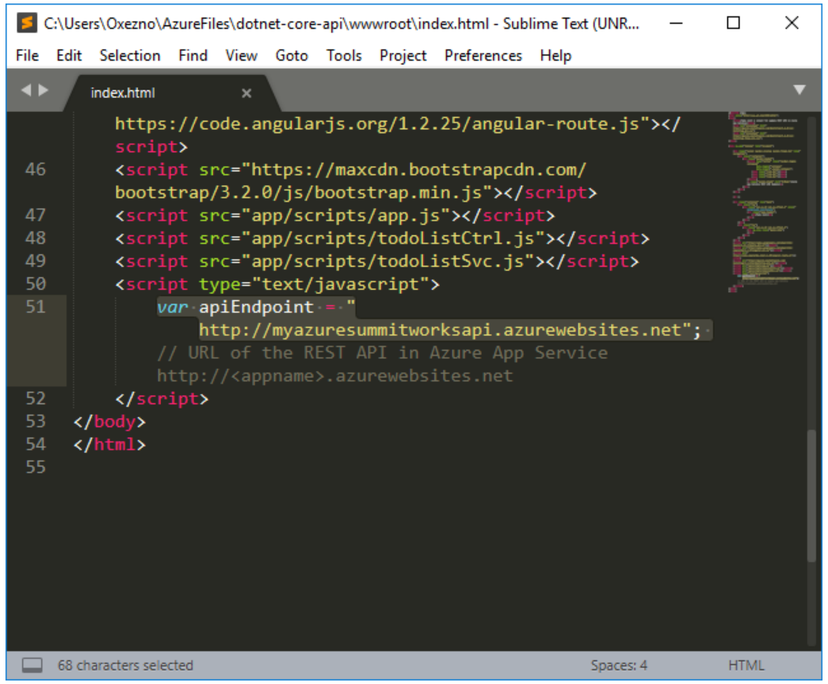
Test CORS in sample app

In your local repository, open wwwroot/index.html.

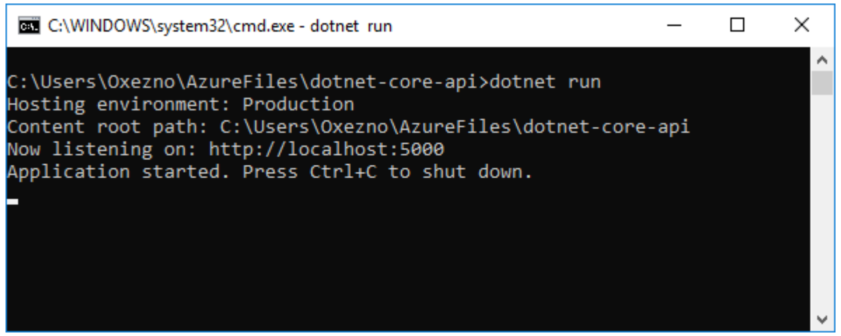
In Line 51, set the apiEndpoint variable to the URL of your deployed API (http://<app_name>.azurewebsites.net). Replace <appname> with your app name in App Service.

In your local terminal window, run the sample app again.

dotnet run


C:\Users\Oxezno\AzureFiles\dotnet-core-api\wwwroot\index.html - Sublime Text (UNR...
File Edit Selection Find View Goto Tools Project Preferences Help
index.html
68 characters selected
Spaces: 4
HTML

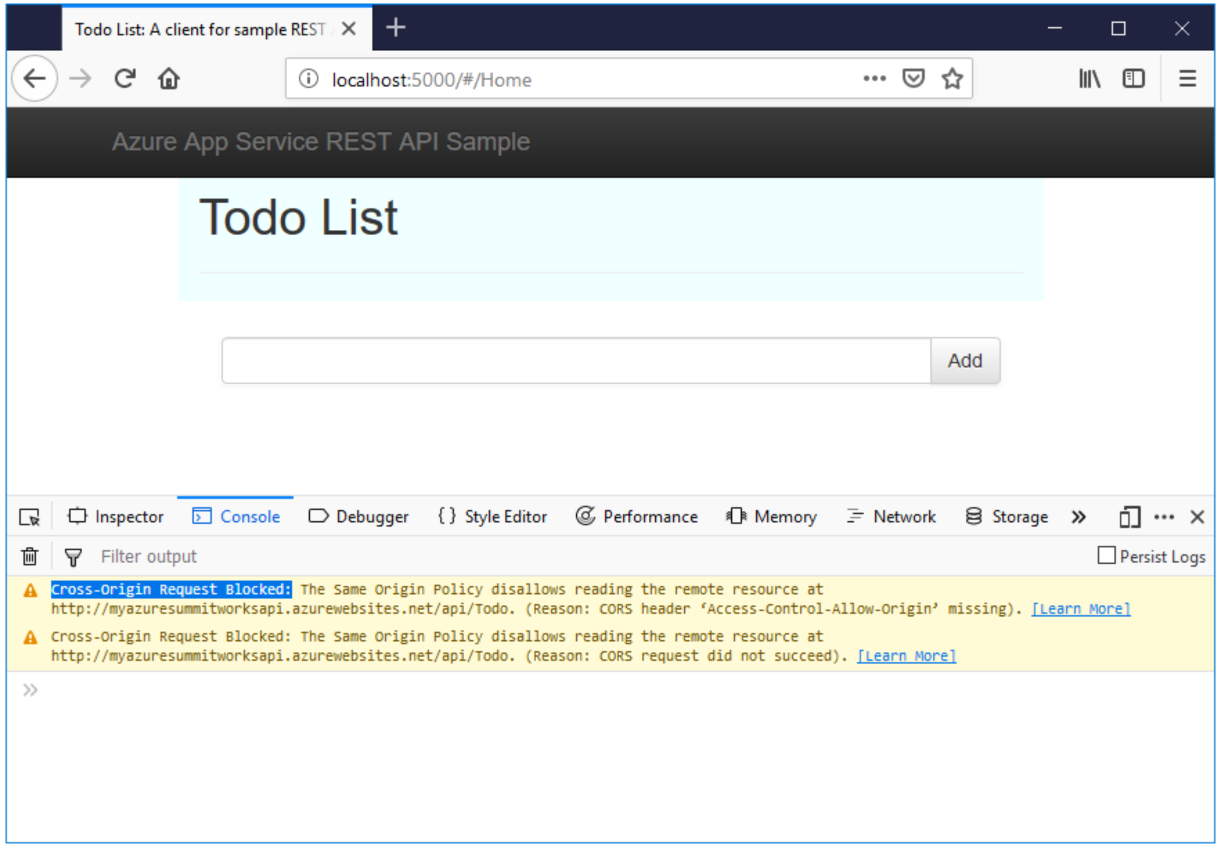
```
        https://code.angularjs.org/1.2.25/angular-route.js"></
        script>
46      <script src="https://maxcdn.bootstrapcdn.com/
        bootstrap/3.2.0/js/bootstrap.min.js"></script>
47      <script src="app/scripts/app.js"></script>
48      <script src="app/scripts/todoListCtrl.js"></script>
49      <script src="app/scripts/todoListSvc.js"></script>
50      <script type="text/javascript">
51          var apiEndpoint = "
            http://myazuresummitworksapi.azurewebsites.net";
            // URL of the REST API in Azure App Service
            http://<appname>.azurewebsites.net
52      </script>
53  </body>
54  </html>
55
```



```
C:\Users\Oxezno\AzureFiles\dotnet-core-api>dotnet run
Hosting environment: Production
Content root path: C:\Users\Oxezno\AzureFiles\dotnet-core-api
Now listening on: http://localhost:5000
Application started. Press Ctrl+C to shut down.
```

Host a RESTful API with CORS in Azure

Navigate to the browser app at *http://localhost:5000*. Open the developer tools window in your browser (Ctrl+Shift+i in Chrome for Windows) and inspect the Console tab. You should now see the error message, No 'Access-Control-Allow-Origin' header is present on the requested resource.



```
Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at
http://myazuresummitworksapi.azurewebsites.net/api/Todo. (Reason: CORS header ‘Access-Control-Allow-Origin’ missing). [Learn More]
Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at
http://myazuresummitworksapi.azurewebsites.net/api/Todo. (Reason: CORS request did not succeed). [Learn More]
```

Host a RESTful API with CORS in Azure

Because of the domain mismatch between the browser app (http://localhost:5000) and remote resource (http://<app_name>.azurewebsites.net), and the fact that your API in App Service is not sending the Access-Control-Allow-Origin header, your browser has prevented cross-domain content from loading in your browser app.

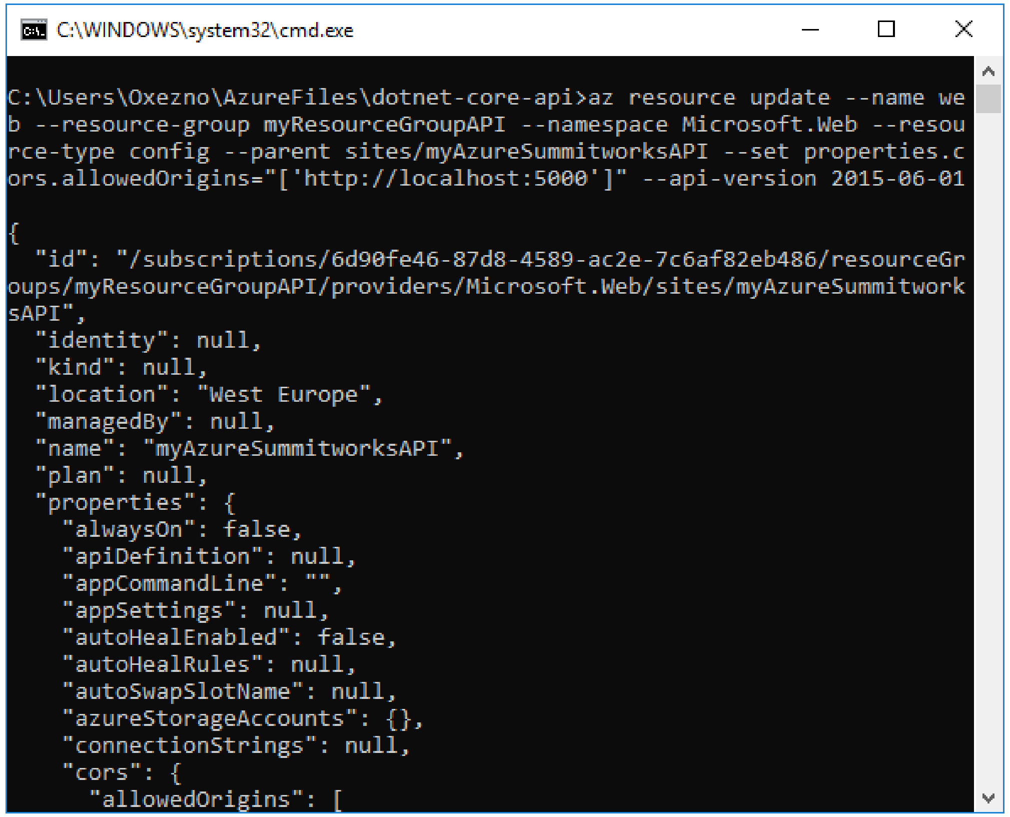
In production, your browser app would have a public URL instead of the localhost URL, but the way to enable CORS to a localhost URL is the same as a public URL.

Host a RESTful API with CORS in Azure

Enable CORS

In the Cloud Shell, enable CORS to your client's URL by using the az resource update command. Replace the <appname> placeholder.

- *az resource update --name web --resource-group myResourceGroup --namespace Microsoft.Web --resource-type config --parent sites/<app_name> --set properties.cors.allowedOrigins="['http://localhost:5000']" --api-version 2015-06-01*

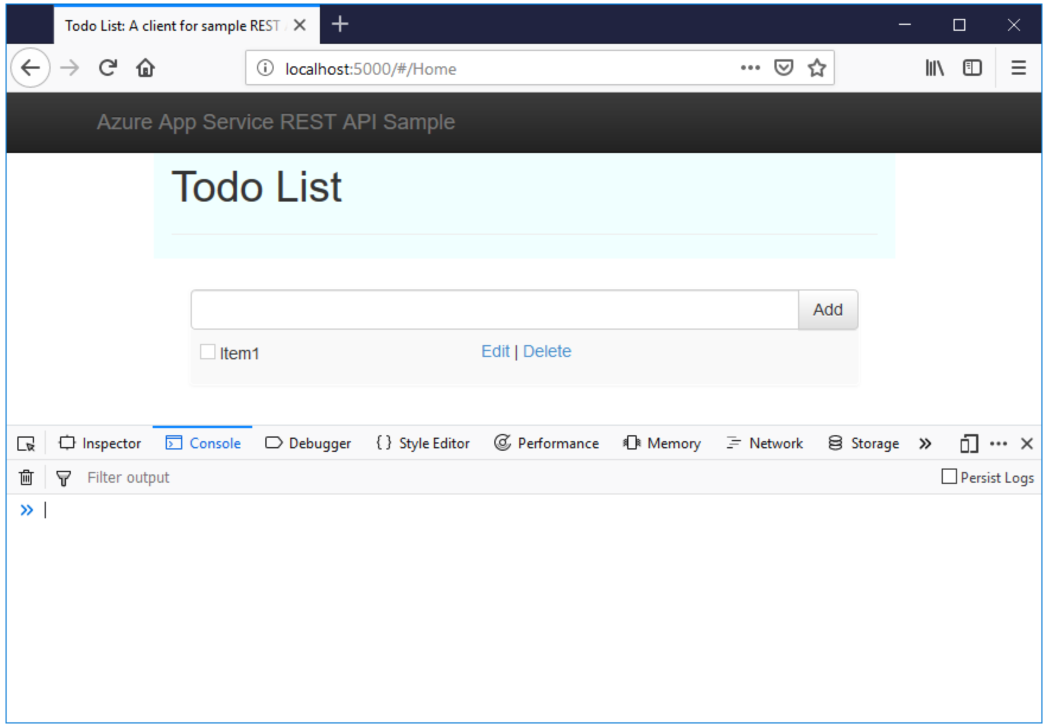


```
C:\Users\Oxezno\AzureFiles\dotnet-core-api>az resource update --name we
b --resource-group myResourceGroupAPI --namespace Microsoft.Web --resou
rce-type config --parent sites/myAzureSummitworksAPI --set properties.c
ors.allowedOrigins="['http://localhost:5000']" --api-version 2015-06-01

{
  "id": "/subscriptions/6d90fe46-87d8-4589-ac2e-7c6af82eb486/resourceGr
oups/myResourceGroupAPI/providers/Microsoft.Web/sites/myAzureSummitwork
sAPI",
  "identity": null,
  "kind": null,
  "location": "West Europe",
  "managedBy": null,
  "name": "myAzureSummitworksAPI",
  "plan": null,
  "properties": {
    "alwaysOn": false,
    "apiDefinition": null,
    "appCommandLine": "",
    "appSettings": null,
    "autoHealEnabled": false,
    "autoHealRules": null,
    "autoSwapSlotName": null,
    "azureStorageAccounts": {},
    "connectionStrings": null,
    "cors": {
      "allowedOrigins": [
```

Host a RESTful API with CORS in Azure

Test CORS again

Refresh the browser app at *http://localhost:5000*. The error message in the Console window is now gone, and you can see the data from the deployed API and interact with it. Your remote API now supports CORS to your browser app running locally.


Todo List: A client for sample REST
localhost:5000/#/Home
Azure App Service REST API Sample
Todo List
Add
Item1
Edit | Delete
Inspector
Console
Debugger
Style Editor
Performance
Memory
Network
Storage
Filter output
Persist Logs