Guide to Run Stateless Web API Services on Service Fabric

# Prerequisites:

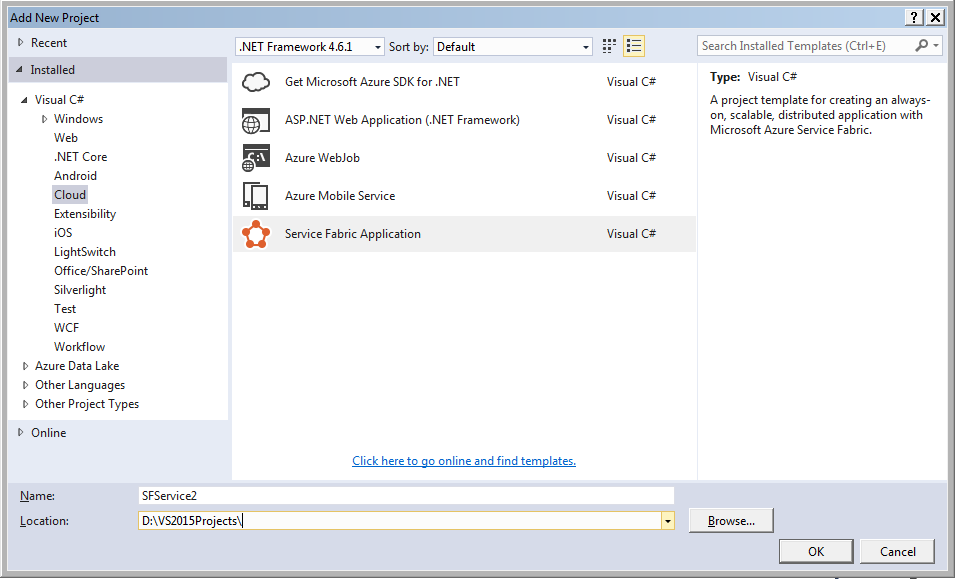
* Install [Microsoft Azure Service Fabric SDK](https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-get-started)
* Install [Powershell 5.x](https://chocolatey.org/packages/powershell)

# Development Steps:

## Step 1: Launch Visual Studio as an administrator

## Step 2: Create a Service Fabric Application

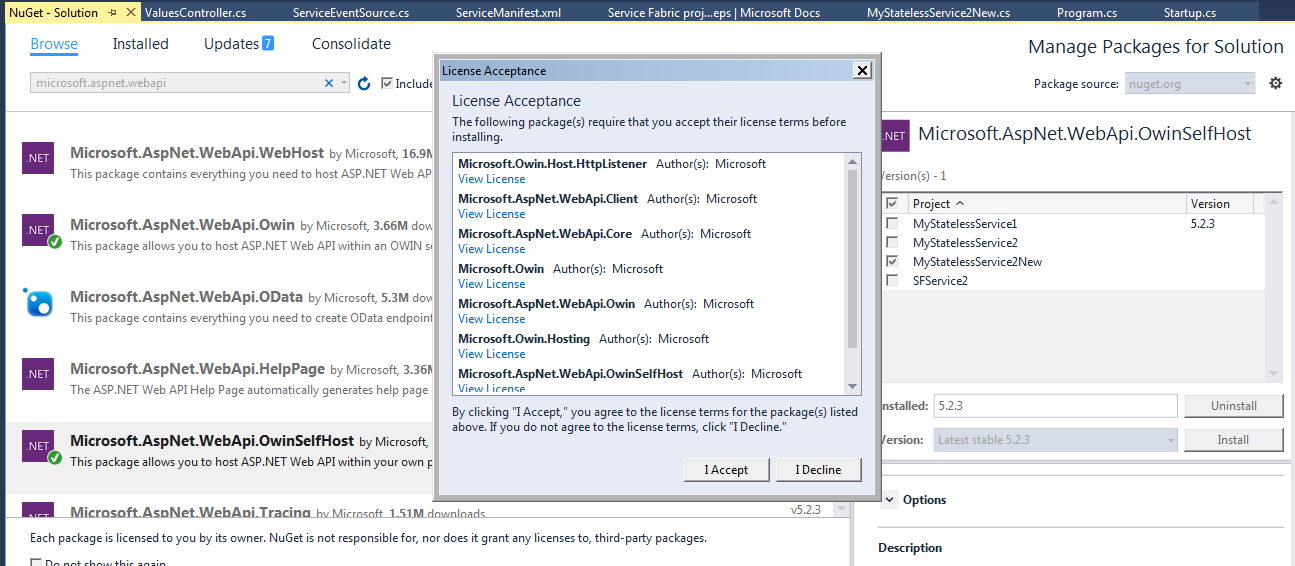
Click **File > New Project > Cloud > Service Fabric Application**

****

## ****Step 3: Select Service Template****

**It will prompt for selecting a “Service Template”. Select “Stateless Service” -> name the service (say *MyStatelessService1*).**

## ****Step 4: OwinSelfHost Nuget install****

**Install the nuget package for *Microsoft.AspNet.WebApi.OwinSelfHost* in the stateless service application created in Step 3**

## ****Step 5: Create the Self-hosting server****

**In *MyStatelessService1* project -> Add ->New ->Class Library. Name the class file as *OwinCommunicationListener.cs*. This is the code that contains Owin self-hosting code**

*Ref:* [*https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-reliable-services-communication-webapi*](https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-reliable-services-communication-webapi)

using Microsoft.Owin.Hosting;

using Microsoft.ServiceFabric.Services.Communication.Runtime;

using Owin;

using System;

using System.Fabric;

using System.Fabric.Description;

using System.Globalization;

using System.Threading;

using System.Threading.Tasks;

namespace MyStatelessService1

{

internal class OwinCommunicationListener : ICommunicationListener

{

private readonly ServiceEventSource eventSource;

private readonly Action<IAppBuilder> startup;

private readonly ServiceContext serviceContext;

private readonly string endpointName;

private readonly string appRoot;

private IDisposable webApp;

private string publishAddress;

private string listeningAddress;

public OwinCommunicationListener(Action<IAppBuilder> startup, ServiceContext serviceContext, ServiceEventSource eventSource, string endpointName)

: this(startup, serviceContext, eventSource, endpointName, null)

{

}

public OwinCommunicationListener(Action<IAppBuilder> startup, ServiceContext serviceContext, ServiceEventSource eventSource, string endpointName, string appRoot)

{

if (startup == null)

{

throw new ArgumentNullException(nameof(startup));

}

if (serviceContext == null)

{

throw new ArgumentNullException(nameof(serviceContext));

}

if (endpointName == null)

{

throw new ArgumentNullException(nameof(endpointName));

}

if (eventSource == null)

{

throw new ArgumentNullException(nameof(eventSource));

}

this.startup = startup;

this.serviceContext = serviceContext;

this.endpointName = endpointName;

this.eventSource = eventSource;

this.appRoot = appRoot;

}

public Task<string> OpenAsync(CancellationToken cancellationToken)

{

//TODO:

//EndpointResourceDescription endpoint =

// serviceContext.CodePackageActivationContext.GetEndpoint("WebEndpoint");

//string uriPrefix = $"{endpoint.Protocol}://+:{endpoint.Port}/myapp/";

var serviceEndpoint = this.serviceContext.CodePackageActivationContext.GetEndpoint(this.endpointName);

var protocol = serviceEndpoint.Protocol;

int port = serviceEndpoint.Port;

if (this.serviceContext is StatefulServiceContext)

{

StatefulServiceContext statefulServiceContext = this.serviceContext as StatefulServiceContext;

this.listeningAddress = string.Format(

CultureInfo.InvariantCulture,

"{0}://+:{1}/{2}{3}/{4}/{5}",

protocol,

port,

string.IsNullOrWhiteSpace(this.appRoot)

? string.Empty

: this.appRoot.TrimEnd('/') + '/',

statefulServiceContext.PartitionId,

statefulServiceContext.ReplicaId,

Guid.NewGuid());

}

else if (this.serviceContext is StatelessServiceContext)

{

this.listeningAddress = string.Format(

CultureInfo.InvariantCulture,

"{0}://+:{1}/{2}",

protocol,

port,

string.IsNullOrWhiteSpace(this.appRoot)

? string.Empty

: this.appRoot.TrimEnd('/') + '/');

}

else

{

throw new InvalidOperationException();

}

this.publishAddress = this.listeningAddress.Replace("+", FabricRuntime.GetNodeContext().IPAddressOrFQDN);

try

{

this.eventSource.Message("Starting web server on " + this.listeningAddress);

this.webApp = WebApp.Start(this.listeningAddress, appBuilder => this.startup.Invoke(appBuilder));

this.eventSource.Message("Listening on " + this.publishAddress);

return Task.FromResult(this.publishAddress);

}

catch (Exception ex)

{

this.eventSource.Message("Web server failed to open endpoint {0}. {1}", this.endpointName, ex.ToString());

this.StopWebServer();

throw;

}

}

public Task CloseAsync(CancellationToken cancellationToken)

{

this.eventSource.Message("Closing web server on endpoint {0}", this.endpointName);

this.StopWebServer();

return Task.FromResult(true);

}

public void Abort()

{

this.eventSource.Message("Aborting web server on endpoint {0}", this.endpointName);

this.StopWebServer();

}

private void StopWebServer()

{

if (this.webApp != null)

{

try

{

this.webApp.Dispose();

}

catch (ObjectDisposedException)

{

// no-op

}

}

}

}

}

## ****Step 6: Wire up the Owin listener to API service instance****

**Make sure Service uses the OwinListener we just created. To make that happen, change the MySatelessService1.cs file as following:**

protected override IEnumerable<ServiceInstanceListener> CreateServiceInstanceListeners()

{

var endpoints = Context.CodePackageActivationContext.GetEndpoints()

.Where(endpoint => endpoint.Protocol == EndpointProtocol.Http || endpoint.Protocol == EndpointProtocol.Https)

.Select(endpoint => endpoint.Name);

return endpoints.Select(endpoint => new ServiceInstanceListener(

serviceContext => new OwinCommunicationListener(Startup.ConfigureApp, serviceContext, ServiceEventSource.Current, endpoint), endpoint));

}

**Step 7: go to *ServiceManifest.xml* in Project *MyStatelessService1* and make proper entry for endpoint as following:**

<Resources>

<Endpoints>

<!-- This endpoint is used by the communication listener to obtain the port on which to

listen. Please note that if your service is partitioned, this port is shared with

replicas of different partitions that are placed in your code. -->

<Endpoint Name="ServiceEndpoint" Protocol="http" Port="9091" />

</Endpoints>

</Resources>

## ****Step 8: Add the Startup.cs****

**Again, Add-> new -> Class. Name the class as Startup.cs. Put the following code therein:**

using System.Web.Http;

using Owin;

namespace MyStatelessService1

{

public static class Startup

{

public static void ConfigureApp(IAppBuilder appBuilder)

{

// Configure Web API for self-host.

HttpConfiguration config = new HttpConfiguration();

config.Routes.MapHttpRoute(

name: "DefaultApi",

routeTemplate: "api/{controller}/{id}",

defaults: new { id = RouteParameter.Optional }

);

appBuilder.UseWebApi(config);

}

}

}

## ****Step 9: Add the API Controller****

**In the stateless web API project, create new folder labelled “Controller” and add a Controller as following therein:**

using System.Collections.Generic;

using System.Web.Http;

namespace MyStatelessService1.Controllers

{

public class ValuesController : ApiController

{

// GET api/values

public IEnumerable<string> Get()

{

return new string[] { "value1", "value2" };

}

// GET api/values/5

public string Get(int id)

{

return "value";

}

// POST api/values

public void Post([FromBody]string value)

{

}

// PUT api/values/5

public void Put(int id, [FromBody]string value)

{

}

// DELETE api/values/5

public void Delete(int id)

{

}

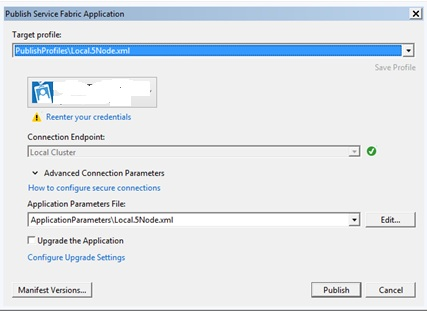
}

}

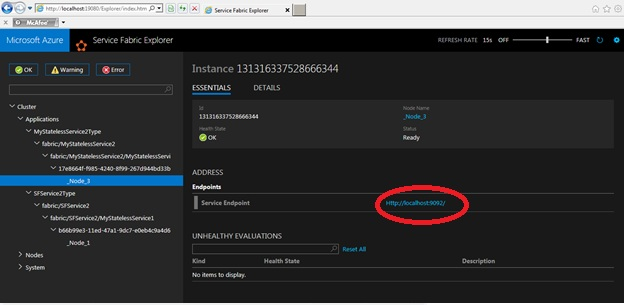
**Step 10: Build and run**

* **ReBuild the solution**
* **Right-click SFService2 -> *Deploy***
* **Once deploy is successful, right-click SFService2 *-> Publish***

**Select *Local.5 Node* as target profile**

****

* **In the Notification tray, locate the Service fabric icon, right click->*Manage local cluster***
* **In *Service Fabric Explorer*, monitor the applications and services therein**
* **Get the Service URL endpoint from Service Fabric Explorer and use that to test the services:**

****

**Hope that helps!**