

SAKET COLLEGE OF ARTS, SCIENCE & COMMERCE



(Affiliated to University Of Mumbai)
2021 – 2021

M.Sc. (IT)
Semester-III Examination

SUBMITTED BY
NITESH SINGH

Roll No
217530

Subject
CLOUD APPLICATION DEVELOPMENT



**SAKET GYANPEETH'S
SAKET COLLEGE OF ARTS, SCIENCE & COMMERCE**
(Affiliated to University of Mumbai)

Saket Vidyanagari Marg, Chinchpada Road, Katelyan (East)-421 306, Dist. Thane(MAH.)

Department of Information Technology

CERTIFICATE

This is to certify that

**NITESH SINGH AS PRESCRIBED BY OF MSC INFORMATION
TECHNOLOGY**

Submitted the same in the Partial Fulfillment of **M.Sc (Information
Technology)** Degree of Mumbai University.

Project Code:psit3p2c

Head of the Department

External Examiner

Principal

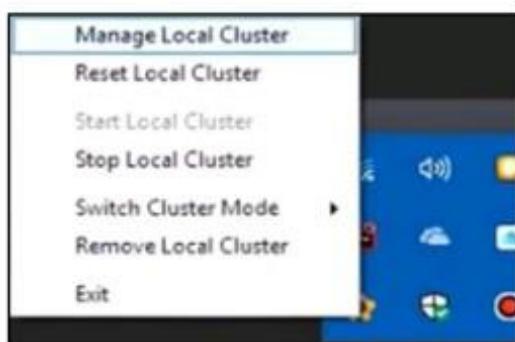
INDEX

Sr No.	Topic	Date	Sign
1.	Develop an Asp.net core MVC based state ful web application.		
2.	Create Asp.net basic web app.		
3.	Deploy on azure kubernet service cluster using Azure.		
4. (A)	Create on Api management services.		
(B)	Create on Api gateway service.		

Develop an ASP.NET
core MVC based
Stateful Web
Application.

Pre-Requisite:

1. Install Microsoft Visual Studio 2017 or any higher version.
2. Install the Microsoft Azure Service Fabric SDK.
3. Create a Microsoft Azure account.
4. Make sure that the Service Fabric local cluster is in a running state. Ensure this by browsing <https://localhost:19080/Explorer/index.html#> or by right clicking the service Fabric Icon in the system tray, as shown in below Figure.



Steps to create ASP.NET core MVC based Stateful Web Application:

1. Launch the Visual Studio as an administrator.
2. Create a project by selecting File > New > Project.
3. Select 'Service Fabric Application' from Project template and click on 'Next' button.
4. Enter Project Name and click on 'Next' button.
5. On the New Service Fabric Service page, choose Stateless ASP.NET Core, name your service **VotingWeb**, then click 'Create' button.
6. Select 'Web Application (Model-View-Controller)' and click on 'Create' button.
7. Visual Studio creates an application and a service project and displays them in Solution Explorer.
8. Update the site.js file

Open wwwroot/js/site.js. Replace its contents with the following JavaScript used by the Home views, then save your changes.

```
var app = angular.module('VotingApp', ['ui.bootstrap']);

app.run(function () {});

app.controller('VotingAppController', ['$rootScope', '$scope', '$http', '$timeout', function ($rootScope, $scope, $http, $timeout) {

    $scope.refresh = function () {
        $http.get('/api/Votes?c=' + new Date().getTime())
            .then(function (data, status) {
                $scope.votes = data;
            }, function (data, status) {
                $scope.votes = undefined;
            });
    };
}]);
```

```
};

$scope.remove = function (item) {
    $http.delete('api/Votes/' + item)
        .then(function (data, status) {
            $scope.refresh();
        })
};

$scope.add = function (item) {
    var fd = new FormData();
    fd.append('item', item);
    $http.put('api/Votes/' + item, fd, {
        transformRequest: angular.identity,
        headers: { 'Content-Type': undefined }
    })
        .then(function (data, status) {
            $scope.refresh();
            $scope.item = undefined;
        })
};

}};

});
```

9. Update the Index.cshtml file

Open Views/Home/Index.cshtml, the view specific to the Home controller. Replace its contents with the following, then save your changes.

```
@{

    ViewData["Title"] = "Service Fabric Voting Sample";
}

<div ng-controller="VotingAppController" ng-init="refresh()">
    <div class="container-fluid">
        <div class="row">
            <div class="col-xs-8 col-xs-offset-2 text-center">
                <h2>Service Fabric Voting Sample</h2>
            </div>
        </div>
    </div>
```

```
</div>

<div class="row">
  <div class="col-xs-8 col-xs-offset-2">
    <form class="col-xs-12 center-block">
      <div class="col-xs-6 form-group">
        <input id="txtAdd" type="text" class="form-control" placeholder="Add voting option" ng-model="item"/>
      </div>
      <button id="btnAdd" class="btn btn-default" ng-click="add(item)">
        <span class="glyphicon glyphicon-plus" aria-hidden="true"></span>
        Add
      </button>
    </form>
  </div>
</div>
<hr/>
<div class="row">
  <div class="col-xs-8 col-xs-offset-2">
    <div class="row">
      <div class="col-xs-4">
        Click to vote
      </div>
    </div>
    <div class="row top-buffer" ng-repeat="vote in votes.data">
      <div class="col-xs-8">
        <button class="btn btn-success text-left btn-block" ng-click="add(vote.key)">
          <span class="pull-left">
            {{vote.key}}
          </span>
          <span class="badge pull-right">
            {{vote.value}} Votes
          </span>
        </button>
      </div>
    </div>
  </div>
</div>
```

11. Update the VotingWeb.cs file

Open the VotingWeb.cs file, which creates the ASP.NET Core WebHost inside the stateless service using the WebListener web server.

Replace the content with the following code, then save your changes.

```
namespace VotingWeb
{
    using System;
    using System.Collections.Generic;
    using System.Fabric;
    using System.IO;
    using System.Net.Http;
    using Microsoft.AspNetCore.Hosting;
    using Microsoft.Extensions.Logging;
    using Microsoft.Extensions.DependencyInjection;
    using Microsoft.ServiceFabric.Services.Communication.AspNetCore;
    using Microsoft.ServiceFabric.Services.Communication.Runtime;
    using Microsoft.ServiceFabric.Services.Runtime;

    internal sealed class VotingWeb : StatelessService
    {
        public VotingWeb(StatelessServiceContext context)
            : base(context)
        {
        }

        protected override IEnumerable<ServiceInstanceListener> CreateServiceInstanceListeners()
        {
            return new ServiceInstanceListener[]
            {
                new ServiceInstanceListener(
                    serviceContext =>
                    new KestrelCommunicationListener(
                        serviceContext,
                        "ServiceEndpoint",
                        "ServicePort")
                )
            };
        }
    }
}
```

```

        (url, listener) =>
    {
        ServiceEventSource.Current.ServiceMessage(serviceContext, $"Starting Kestrel on
{url}");

        return new WebHostBuilder()
            .UseKestrel()
            .ConfigureLogging(logging =>
            {
                logging.ClearProviders();
                logging.AddApplicationInsights();
            })
            .ConfigureServices(
                services => services
                    .AddSingleton<HttpClient>(new HttpClient())
                    .AddSingleton<FabricClient>(new FabricClient())
                    .AddSingleton<StatelessServiceContext>(serviceContext)
                    .UseContentRoot(Directory.GetCurrentDirectory())
                    .UseStartup<Startup>()
                    .UseServiceFabricIntegration(listener, ServiceFabricIntegrationOptions.None)
                    .UseUrls(url)
                    .Build(),
            ))
    );
}

internal static Uri GetVotingDataServiceName(ServiceContext context)
{
    return new Uri($"{context.CodePackageActivationContext.ApplicationName}/VotingData");
}
}

```

12. Add the VotesController.cs file

Add a controller, which defines voting actions. Right-click on the Controllers folder, then select Add ->New item->Visual C#->ASP.NET Core->Class. Name the file VotesController.cs, then click Add.

Replace the VotesController.cs file contents with the following, then save your changes this file is modified to read and write voting data from the back-end service.

```
namespace VotingWeb.Controllers

{
    using System;
    using System.Collections.Generic;
    using System.Fabric;
    using System.Fabric.Query;
    using System.Linq;
    using System.Net.Http;
    using System.Net.Http.Headers;
    using System.Text;
    using System.Threading.Tasks;
    using Microsoft.AspNetCore.Mvc;
    using Newtonsoft.Json;
    [Produces("application/json")]
    [Route("api/[controller]")]
    public class VotesController : Controller
    {
        private readonly HttpClient httpClient;
        private readonly FabricClient fabricClient;
        private readonly string reverseProxyBaseUri;
        private readonly StatelessServiceContext serviceContext;

        public VotesController(HttpClient httpClient, StatelessServiceContext context, FabricClient fabricClient)
        {
            this.fabricClient = fabricClient;
            this.httpClient = httpClient;
            this.serviceContext = context;
            this.reverseProxyBaseUri = Environment.GetEnvironmentVariable("ReverseProxyBaseUrl");
        }

        // GET: api/Votes
        [HttpGet("")]
        public async Task<IActionResult> Get()
```

```
long partitionKey = this.GetPartitionKey(name);

string proxyUrl =
${proxyAddress}/api/VoteData/{name}?PartitionKey={partitionKey}&PartitionKind=Int64Range";

StringContent putContent = new StringContent(${'name': ${name}}", Encoding.UTF8,
"application/json");

putContent.Headers.ContentType = new MediaTypeHeaderValue("application/json");

using (HttpResponseMessage response = await this.httpClient.PutAsync(proxyUrl, putContent))
{
    return new ContentResult()
    {
        StatusCode = (int) response.StatusCode,
        Content = await response.Content.ReadAsStringAsync()
    };
}

// DELETE: api/Votes/name
[HttpDelete("{name}")]
public async Task<IActionResult> Delete(string name)
{
    Uri serviceName = VotingWeb.GetVotingDataServiceName(this.serviceContext);
    Uri proxyAddress = this.GetProxyAddress(serviceName);
    long partitionKey = this.GetPartitionKey(name);

    string proxyUrl =
${proxyAddress}/api/VoteData/{name}?PartitionKey={partitionKey}&PartitionKind=Int64Range";

    using (HttpResponseMessage response = await this.httpClient.DeleteAsync(proxyUrl))
    {
        if (response.StatusCode != System.Net.HttpStatusCode.OK)
        {
            return this.StatusCode((int) response.StatusCode);
        }
    }
}
```

```
        return new OkResult();
    }

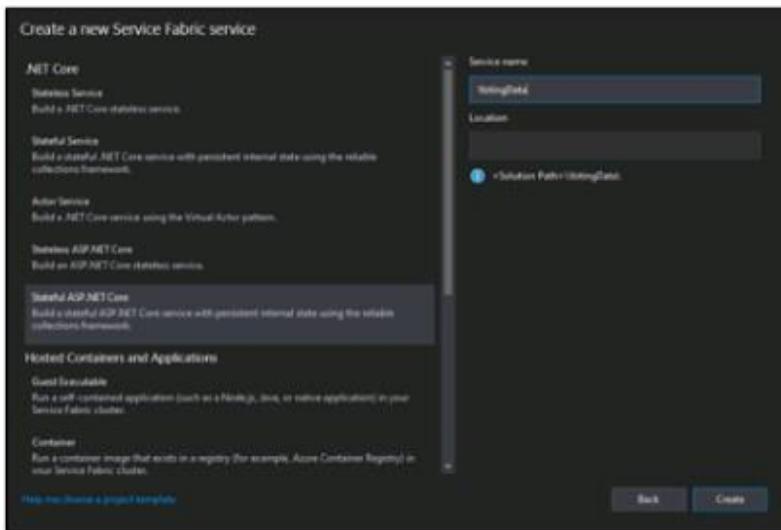
    /// <summary>
    /// Constructs a reverse proxy URL for a given service.
    /// Example: http://localhost:19081/VotingApplication/VotingData/
    /// </summary>
    /// <param name="serviceName"></param>
    /// <returns></returns>
    private Uri GetProxyAddress(Uri serviceName)
    {
        return new Uri($"{this.reverseProxyBaseUri}{serviceName.AbsolutePath}");
    }

    /// <summary>
    /// Creates a partition key from the given name.
    /// Uses the zero-based numeric position in the alphabet of the first letter of the name (0-25).
    /// </summary>
    /// <param name="name"></param>
    /// <returns></returns>
    private long GetPartitionKey(string name)
    {
        return Char.ToUpper(name.First()) - 'A';
    }
}
```

13. Add a stateful back-end service to your application

In Solution Explorer, right-click Services within the Voting application project and choose Add -> New Service Fabric Service....

In the New Service Fabric Service dialog, choose Stateful ASP.NET Core, name the service VotingData, then click 'Create' button.



14. Once your service project is created, you have two services in your application. As you continue to build your application, you can add more services in the same way. Each can be independently versioned and upgraded.

The next page provides a set of ASP.NET Core project templates. choose API. Visual Studio creates the VotingData service project and displays it in Solution Explorer.

15. Add the VoteDataController.cs file

In the VotingData project, right-click on the Controllers folder, then select Add->New item->Class. Name the file VoteDataController.cs and click Add.

Replace the file contents with the following, then save your changes.

namespace VotingData.Controllers

```
{  
    using System.Collections.Generic;  
    using System.Threading;  
    using System.Threading.Tasks;  
    using Microsoft.AspNetCore.Mvc;  
    using Microsoft.ServiceFabric.Data;  
    using Microsoft.ServiceFabric.Data.Collections;  
  
    [Route("api/[controller]")]  
    public class VoteDataController : Controller  
    {  
        private readonly IReliableStateManager stateManager;  
  
        public VoteDataController(IReliableStateManager stateManager)  
        {  
            this.stateManager = stateManager;  
        }  
        // GET api/VoteData  
        [HttpGet]  
        public async Task< IActionResult> Get()  
        {
```

```

CancellationToken ct = new CancellationToken();
IReliableDictionary<string, int> votesDictionary = await
this.stateManager.GetOrAddAsync<IReliableDictionary<string, int>>("counts");
using (ITransaction tx = this.stateManager.CreateTransaction())
{
    Microsoft.ServiceFabric.Data.IAsyncEnumerable<KeyValuePair<string, int>> list = await
votesDictionary.CreateEnumerableAsync(tx);
    Microsoft.ServiceFabric.Data.IAsyncEnumerator<KeyValuePair<string, int>> enumerator =
list.GetAsyncEnumerator();
    List<KeyValuePair<string, int>> result = new List<KeyValuePair<string, int>>();
    while (await enumerator.MoveNextAsync(ct))
    {
        result.Add(enumerator.Current);
    }
    return this.Json(result);
}
// PUT api/VoteData/name
[HttpPut("{name}")]
public async Task<IActionResult> Put(string name)
{
    IReliableDictionary<string, int> votesDictionary = await
this.stateManager.GetOrAddAsync<IReliableDictionary<string, int>>("counts");

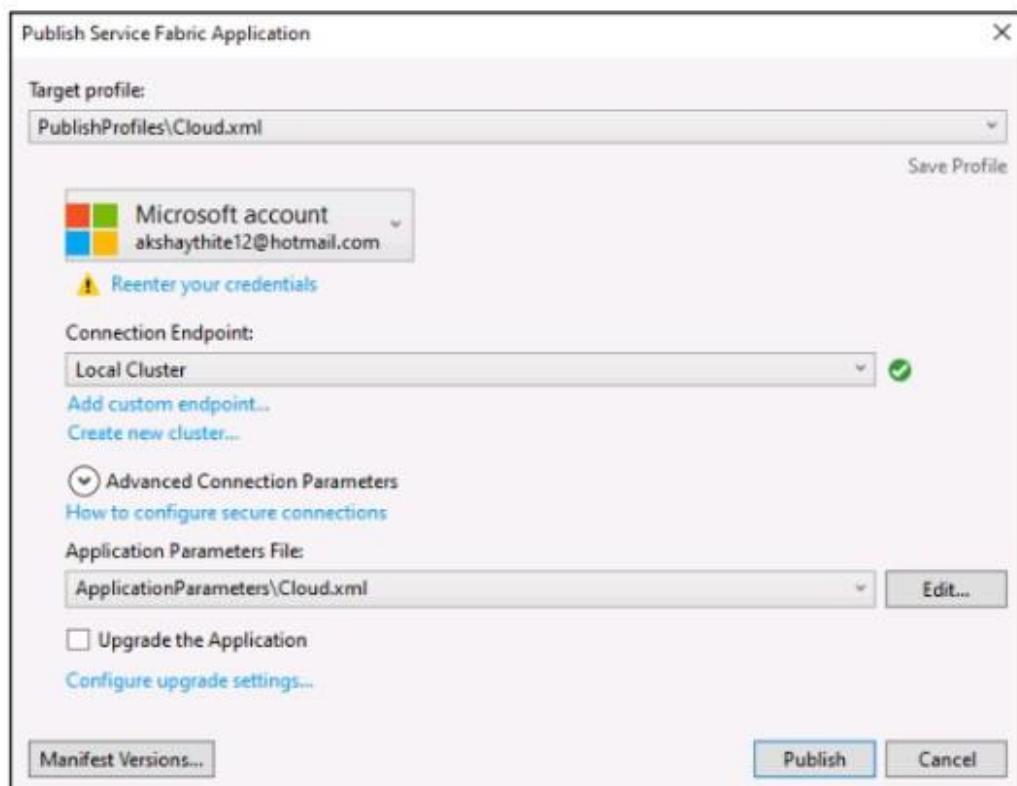
    using (ITransaction tx = this.stateManager.CreateTransaction())
    {
        await votesDictionary.AddOrUpdateAsync(tx, name, 1, (key, oldvalue) => oldvalue + 1);
        await tx.CommitAsync();
    }
    return new OkResult();
}
// DELETE api/VoteData/name
[HttpDelete("{name}")]
public async Task<IActionResult> Delete(string name)
{
    IReliableDictionary<string, int> votesDictionary = await
this.stateManager.GetOrAddAsync<IReliableDictionary<string, int>>("counts");
    using (ITransaction tx = this.stateManager.CreateTransaction())
    {
        if (await votesDictionary.ContainsKeyAsync(tx, name))
        {
            await votesDictionary.TryRemoveAsync(tx, name);
            await tx.CommitAsync();
            return new OkResult();
        }
        else { return new NotFoundResult(); } } } }

```

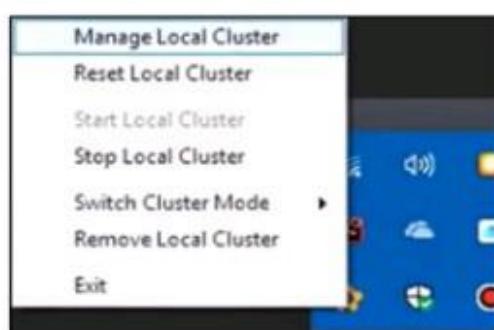
16. Build the solution by pressing the F5.

17. Once the solution is built successfully, Publish Service Fabric application

In the solution Explorer Right click on the Voting and select Publish. The Publish dialog box appears.



18. Once the application is ready, open the 'Service Fabric Explorer' from system tray.



19. Expand the web application which are present under 'Cluster' tab and double click on last node which present under cluster.

20. Click the URL displayed in 'Endpoints' section.

The screenshot shows the Service Fabric Explorer interface. On the left, there's a navigation tree with 'Cluster' expanded, showing 'Applications' (ServiceFabDemotype), 'Voting type' (fabric/Voting), and its sub-items 'Voting' and 'VotingWeb'. Under 'VotingWeb', the GUID '860931dd-f619-4be6-ba3f-0d7c41652ab1' is listed. On the right, the 'ESSENTIALS' tab is selected. It displays the node ID '132564075143738457', health state 'OK', and node name 'Node_0'. Below that is the 'ADDRESS' section with 'Endpoints' (empty) and the URL 'http://DESKTOP-G7NEKRT:8080'.

21. Enter the user name and one vote will be added.

The screenshot shows a web browser window titled 'Service Fabric Voting Sample'. The address bar indicates the URL is 'Not secure desktop-g7nekrt:8080'. The page content includes a text input field with the value 'Akshay Thite' and a 'Add' button. Below this, there's a heading 'Click to vote' followed by three voting items: 'Akshay Thite' (2 Votes, Remove), 'Amit Thite' (1 Vote, Remove), and 'Hiren Dand' (0 Votes, Remove).

Create a basic ASP.NET web app

1. Launch Visual Studio 2019.
2. Select **File > New > Project**.
3. Select **ASP.NET Web Application(.NET Framework) C#**.
4. Enter a project name > **Select Create**.
5. Select **MVC > Create**.

Add Application Insights automatically

Automatically adding Application Insights to a template-based ASP.NET web app within your ASP.NET web app project in Visual Studio:

1. Select **Add Application Insights Telemetry > Application Insights Sdk (local) > Next > Finish > Close**.
2. Open the ApplicationInsights.config file.
3. Before the closing `</ApplicationInsights>` tag add a line containing the instrumentation key for your Application Insights resource. Your instrumentation key can be found on the overview pane of your newly created Application Insights resource that you created as part of the prerequisites for this article.

XMLCopy

```
<InstrumentationKey>your-instrumentation-key-goes-here</InstrumentationKey>
```

4. Select **Project > Manage NuGet Packages > Updates > Update** each Microsoft.ApplicationInsights NuGet package to the latest stable release
5. Run your application by selecting **IIS Express**. A basic ASP.NET app will log information to Application Insights. As you navigate the pages on the site telemetry will be sent to Application Insights.

Add Application Insights manually

This section will guide you through manually adding Application Insights to a template-based ASP.NET web app. This section assumes you are using a web app based on the standard ASP.NET Framework MVC web app template.

1. Add the following NuGet packages and their dependencies to your project:
 - o [Microsoft.ApplicationInsights.WindowsServer](#)
 - o [Microsoft.ApplicationInsights.Web](#)
 - o [Microsoft.AspNet.TelemetryCorrelation](#)
2. In some cases, the ApplicationInsights.config file will be created for you automatically. If the file is already present, skip to step #4. If it is not created automatically, add the following code to the ApplicationInsights.config file:

```
<InstrumentationKey>your-instrumentation-key-goes-here</InstrumentationKey>
```

```
metadata:
  name: azure-vote-back
spec:
  replicas: 1
  selector:
    matchLabels:
      app: azure-vote-back
  template:
    metadata:
      labels:
        app: azure-vote-back
    spec:
      nodeSelector:
        "beta.kubernetes.io/os": linux
      containers:
        - name: azure-vote-back
          image: mcr.microsoft.com/oss/bitnami/redis:6.0.8
          env:
            - name: ALLOW_EMPTY_PASSWORD
              value: "yes"
          resources:
            requests:
              cpu: 100m
              memory: 128Mi
            limits:
              cpu: 250m
              memory: 256Mi
          ports:
            - containerPort: 6379
              name: redis
---
apiVersion: v1
kind: Service
metadata:
  name: azure-vote-back
spec:
  ports:
    - port: 6379
  selector:
    app: azure-vote-back
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: azure-vote-front
```

```
yProcessor, Microsoft.AI.ServerTelemetryChannel">
    <MaxTelemetryItemsPerSecond>5</MaxTelemetryItemsPerSecond>
    <IncludedTypes>Event</IncludedTypes>
</Add>
</TelemetryProcessors>
<TelemetryChannel
Type="Microsoft.ApplicationInsights.WindowsServer.TelemetryChannel.ServerTelemetryChanne
Microsoft.AI.ServerTelemetryChannel" />
</Add>
</TelemetrySinks>
<!--
    Learn more about Application Insights configuration with ApplicationInsights.config here:
    http://go.microsoft.com/fwlink/?LinkId=513840
-->
<InstrumentationKey>your-instrumentation-key-here</InstrumentationKey>
</ApplicationInsights>
```

4. Before the closing </ApplicationInsights> tag, add your instrumentation key for your Application Insights resource. Your instrumentation key can be found on the overview pane of your newly created Application Insights resource that you created as part of the prerequisites for this article.

XMLCopy

```
<InstrumentationKey>your-instrumentation-key-goes-here</InstrumentationKey>
```

5. At the same level of your project as the ApplicationInsights.config file, create a folder called ErrorHandler with a new C# file called AiHandleErrorAttribute.cs. The contents of the file will look as follows:

C#Copy

```
using System;
using System.Web.Mvc;
using Microsoft.ApplicationInsights;
```

```
namespace WebApplication10.ErrorHandler //namespace will vary based on your project name
{
    [AttributeUsage(AttributeTargets.Class | AttributeTargets.Method, Inherited = true, AllowMulti
    = true)]
    public class AiHandleErrorAttribute : HandleErrorAttribute
    {
        public override void OnException(ExceptionContext filterContext)
        {
            if (filterContext != null && filterContext.HttpContext != null && filterContext.Exception != nu
            {
                //If customError is Off, then AI HTTPModule will report the exception
            }
        }
    }
}
```

```
        if (filterContext.HttpContext.IsCustomErrorEnabled)
        {
            var ai = new TelemetryClient();
            ai.TrackException(filterContext.Exception);
        }
    }
    base.OnException(filterContext);
}
}
```

6. In the App_Start folder, open the FilterConfig.cs file and change it to match the sa

C#Copy

```
using System.Web;
using System.Web.Mvc;

namespace WebApplication10 //Namespace will vary based on project name
{
    public class FilterConfig
    {
        public static void RegisterGlobalFilters(GlobalFilterCollection filters)
        {
            filters.Add(new ErrorHandler.AiHandleErrorAttribute());
        }
    }
}
```

7. Update the Web.config file as follows:

XMLCopy

```
<?xml version="1.0" encoding="utf-8"?>
<!--
For more information on how to configure your ASP.NET application, please visit
https://go.microsoft.com/fwlink/?LinkId=301880
-->
<configuration>
    <appSettings>
        <add key="webpages:Version" value="3.0.0.0" />
        <add key="webpages:Enabled" value="false" />
        <add key="ClientValidationEnabled" value="true" />
        <add key="UnobtrusiveJavaScriptEnabled" value="true" />
    </appSettings>
    <system.web>
```

```
<compilation debug="true" targetFramework="4.7.2" />
<httpRuntime targetFramework="4.7.2" />
<httpModules>
  <add name="TelemetryCorrelationHttpModule"
    type="Microsoft.AspNet.TelemetryCorrelation.TelemetryCorrelationHttpModule,
    Microsoft.AspNet.TelemetryCorrelation" />
  <add name="ApplicationInsightsWebTracking"
    type="Microsoft.ApplicationInsights.Web.ApplicationInsightsHttpModule, Microsoft.AI.Web" />
</httpModules>
</system.web>
<runtime>
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
    <dependentAssembly>
      <assemblyIdentity name="Antlr3.Runtime" publicKeyToken="eb42632606e9261f" />
      <bindingRedirect oldVersion="0.0.0.0-3.5.0.2" newVersion="3.5.0.2" />
    </dependentAssembly>
    <dependentAssembly>
      <assemblyIdentity name="Newtonsoft.Json" publicKeyToken="30ad4fe6b2a6aeed" />
      <bindingRedirect oldVersion="0.0.0.0-12.0.0.0" newVersion="12.0.0.0" />
    </dependentAssembly>
    <dependentAssembly>
      <assemblyIdentity name="System.Web.Optimization" publicKeyToken="31bf3856ad364e3" />
      <bindingRedirect oldVersion="1.0.0.0-1.1.0.0" newVersion="1.1.0.0" />
    </dependentAssembly>
    <dependentAssembly>
      <assemblyIdentity name="WebGrease" publicKeyToken="31bf3856ad364e35" />
      <bindingRedirect oldVersion="0.0.0.0-1.6.5135.21930" newVersion="1.6.5135.21930" />
    </dependentAssembly>
    <dependentAssembly>
      <assemblyIdentity name="System.Web.Helpers" publicKeyToken="31bf3856ad364e35" />
      <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
    </dependentAssembly>
    <dependentAssembly>
      <assemblyIdentity name="System.Web.WebPages" publicKeyToken="31bf3856ad364e35" />
      <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
    </dependentAssembly>
    <dependentAssembly>
      <assemblyIdentity name="System.Web.Mvc" publicKeyToken="31bf3856ad364e35" />
      <bindingRedirect oldVersion="1.0.0.0-5.2.7.0" newVersion="5.2.7.0" />
    </dependentAssembly>
    <dependentAssembly>
      <assemblyIdentity name="System.Memory" publicKeyToken="cc7b13ffcd2ddd51" culture="neutral" />
      <bindingRedirect oldVersion="0.0.0.0-4.0.1.1" newVersion="4.0.1.1" />
    </dependentAssembly>
```

Requests to the following hostnames will not be modified by adding correlation headers.
Add entries here to exclude additional hostnames.

NOTE: this configuration will be lost upon NuGet upgrade.

-->

```
<Add>core.windows.net</Add>
<Add>core.chinacloudapi.cn</Add>
<Add>core.cloudapi.de</Add>
<Add>core.usgovcloudapi.net</Add>
</ExcludeComponentCorrelationHttpHeadersOnDomains>
<IncludeDiagnosticSourceActivities>
  <Add>Microsoft.Azure.EventHubs</Add>
  <Add>Microsoft.Azure.ServiceBus</Add>
</IncludeDiagnosticSourceActivities>
</Add>
<Add
```

Type="Microsoft.ApplicationInsights.Extensibility.PerfCounterCollector.PerformanceCollectorModule, Microsoft.AI.PerfCounterCollector">

<!--

Use the following syntax here to collect additional performance counters:

```
<Counters>
  <Add PerformanceCounter="\Process(??APP_WIN32_PROC??)\Handle Count"
ReportAs="Process handle count" />
  ...
</Counters>
```

PerformanceCounter must be either \CategoryName(InstanceName)\CounterName or
\CategoryName\CounterName

NOTE: performance counters configuration will be lost upon NuGet upgrade.

The following placeholders are supported as InstanceName:

??APP_WIN32_PROC?? - instance name of the application process for Win32 counters.

??APP_W3SVC_PROC?? - instance name of the application IIS worker process for IIS/ASP.NET counters.

??APP_CLR_PROC?? - instance name of the application CLR process for .NET counters.

-->

```
</Add>
<Add
```

Type="Microsoft.ApplicationInsights.Extensibility.PerfCounterCollector.QuickPulse.QuickPulseTelemetryModule, Microsoft.AI.PerfCounterCollector" />

```
<Add
```

Type="Microsoft.ApplicationInsights.WindowsServer.AppServicesHeartbeatTelemetryModule, Microsoft.AI.WindowsServer" />

```
<Add
```

```
</Add>
<Add Type="Microsoft.ApplicationInsights.Web.RequestTrackingTelemetryModule,
Microsoft.AI.Web">
<Handlers>
<!--
    Add entries here to filter out additional handlers:
-->

    NOTE: handler configuration will be lost upon NuGet upgrade.

<Add>Microsoft.VisualStudio.Web.PageInspector.Runtime.Tracing.RequestDataHttpHandler</
>
<Add>System.Web.StaticFileHandler</Add>
<Add>System.Web.Handlers.AssemblyResourceLoader</Add>
<Add>System.Web.Optimization.BundleHandler</Add>
<Add>System.Web.Script.Services.ScriptHandlerFactory</Add>
<Add>System.Web.Handlers.TraceHandler</Add>
<Add>System.Web.Services.Discovery.DiscoveryRequestHandler</Add>
<Add>System.Web.HttpDebugHandler</Add>
</Handlers>
</Add>
<Add Type="Microsoft.ApplicationInsights.Web.ExceptionTrackingTelemetryModule,
Microsoft.AI.Web" />
<Add Type="Microsoft.ApplicationInsights.Web.AspNetDiagnosticTelemetryModule,
Microsoft.AI.Web" />
</TelemetryModules>
<ApplicationIdProvider
Type="Microsoft.ApplicationInsights.Extensibility.Implementation.ApplicationId.ApplicationIns
ApplicationIdProvider, Microsoft.ApplicationInsights" />
<TelemetrySinks>
<Add Name="default">
<TelemetryProcessors>
<Add
Type="Microsoft.ApplicationInsights.Extensibility.PerfCounterCollector.QuickPulse.QuickPulse
metryProcessor, Microsoft.AI.PerfCounterCollector" />
<Add Type="Microsoft.ApplicationInsights.Extensibility.AutocollectedMetricsExtractor,
Microsoft.ApplicationInsights" />
<Add
Type="Microsoft.ApplicationInsights.WindowsServer.TelemetryChannel.AdaptiveSamplingTele
yProcessor, Microsoft.AI.ServerTelemetryChannel">
<MaxTelemetryItemsPerSecond>5</MaxTelemetryItemsPerSecond>
<ExcludedTypes>Event</ExcludedTypes>
</Add>
<Add
Type="Microsoft.ApplicationInsights.WindowsServer.TelemetryChannel.AdaptiveSamplingTele
```

```
Type="Microsoft.ApplicationInsights.WindowsServer.AzureInstanceMetadataTelemetryModule, Microsoft.AI.WindowsServer">
    <!--
        Remove individual fields collected here by adding them to the
        ApplicationInsights.HeartbeatProvider
        with the following syntax:

    <Add
        Type="Microsoft.ApplicationInsights.Extensibility.Implementation.Tracing.DiagnosticsTelemetryModule, Microsoft.ApplicationInsights">
        <ExcludedHeartbeatProperties>
            <Add>osType</Add>
            <Add>location</Add>
            <Add>name</Add>
            <Add>offer</Add>
            <Add>platformFaultDomain</Add>
            <Add>platformUpdateDomain</Add>
            <Add>publisher</Add>
            <Add>sku</Add>
            <Add>version</Add>
            <Add>vmId</Add>
            <Add>vmSize</Add>
            <Add>subscriptionId</Add>
            <Add>resourceGroupName</Add>
            <Add>placementGroupId</Add>
            <Add>tags</Add>
            <Add>vmScaleSetName</Add>
        </ExcludedHeartbeatProperties>
    </Add>
```

NOTE: exclusions will be lost upon upgrade.

```
    -->
</Add>
<Add
    Type="Microsoft.ApplicationInsights.WindowsServer.DeveloperModeWithDebuggerAttachmentModule, Microsoft.AI.WindowsServer" />
    <Add Type="Microsoft.ApplicationInsights.WindowsServer.UnhandledExceptionTelemetryModule, Microsoft.AI.WindowsServer" />
    <Add
        Type="Microsoft.ApplicationInsights.WindowsServer.UnobservedExceptionTelemetryModule, Microsoft.AI.WindowsServer">
            <!--</Add>
            <Add
                Type="Microsoft.ApplicationInsights.WindowsServer.FirstChanceExceptionStatisticsTelemetryModule, Microsoft.AI.WindowsServer">-->
```

To add client-side monitoring, open the _Layout.cshtml file

2. Deploy an Azure Kubernetes Service (AKS) cluster using the Azure portal

Azure Kubernetes Service (AKS) is a managed Kubernetes service that lets you quickly deploy and manage clusters. In this quickstart, you deploy an AKS cluster using the Azure portal. A multi-container application that includes a web front end and a Redis instance is run in the cluster.

Sign in to Azure

Sign in to the Azure portal at <https://portal.azure.com>.

Create an AKS cluster

To create an AKS cluster, complete the following steps:

1. On the Azure portal menu or from the **Home** page, select **Create a resource**.
2. Select **Containers > Kubernetes Service**.
3. On the **Basics** page, configure the following options:
 - o **Project details:** Select an Azure Subscription, then select or create an Azure Resource group, such as *myResourceGroup*.
 - o **Cluster details:** Enter a **Kubernetes cluster name**, such as *myAKSCluster*. Select a Region and Kubernetes version for the AKS cluster.
 - o **Primary node pool:** Select a VM Node size for the AKS nodes. The VM size *can't* be changed once an AKS cluster has been deployed. - Select the number of nodes to deploy into the cluster. For this quickstart, set **Node count** to 1. Node count *can* be adjusted after the cluster has been deployed.

Create Kubernetes cluster

Basics

[Node pools](#)[Authentication](#)[Networking](#)[Integrations](#)[Tags](#)

Azure Kubernetes Service (AKS) manages your hosted Kubernetes environment, making it easy to manage containerized applications without container orchestration expertise. It also eliminates operations and maintenance by provisioning, upgrading, and scaling resources on demand, even when your cluster is offline. [Learn more about Azure Kubernetes Service](#)

Project details

Select a subscription to manage deployed resources and costs. Use resource groups like to group your resources.

Subscription * ⓘ

Resource group * ⓘ

(New) myResourceGroup

[Create new](#)

Cluster details

Kubernetes cluster name * ⓘ

myAKSCluster

Region * ⓘ

(US) East US

Availability zones ⓘ

Zones 1,2,3

Kubernetes version * ⓘ

1.17.11 (default)

Primary node pool

The number and size of nodes in the primary node pool in your cluster. For production workloads, we recommend using at least two nodes per node pool. For development or test workloads, only one node is required. To add additional node pools or to see additional configuration options for this node pool, go to the 'Node pools' tab after creating your cluster. [Learn more about node pools in AKS](#)

Node size * ⓘ

Standard DS2 v2

[Change size](#)

Node count * ⓘ

1

[Review + create](#)

< Previous

Next : Node pools >

The screenshot shows the Azure portal interface for managing a Kubernetes cluster named 'myAKSCluster'. The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Node pools, Upgrade, Scale, Networking, Dev Spaces, Deployment center (preview), Policies (preview), Properties, and Locks. The main content area displays basic cluster details: Resource group (myResourceGroup), Status (Succeeded), Location (East US), Subscription (My Subscription Name), and Subscription ID (00000000-0000-0000-0000-000000000000). It also includes a 'Tags (change)' section with a link to add tags. A callout box titled 'Monitor containers' provides a link to Azure Monitor insights.

Connect to the cluster

To manage a Kubernetes cluster, you use [kubectl](#), the Kubernetes command-line client. The kubectl client is pre-installed in the Azure Cloud Shell.

Open Cloud Shell using the >_ button on the top of the Azure portal.

Select **Next: Node pools** when complete.

4. On the **Node pools** page, keep the default options. At the bottom of the screen click **Next: Authentication**.

Caution

Creating new AAD Service Principals may take multiple minutes to propagate become available causing Service Principal not found errors and validation fail in Azure portal. If you hit this please visit [here](#) for mitigation.

5. On the **Authentication** page, configure the following options:
 - o Create a new service principal by leaving the **Service Principal** field with **(new) default service principal**. Or you can choose *Configure service principal* to use existing one. If you use an existing one, you will need to provide the SPN client ID and secret.
 - o Enable the option for Kubernetes role-based access control (Kubernetes RBAC). This will provide more fine-grained control over access to the Kubernetes resources deployed in your AKS cluster.

Alternatively, you can use a managed identity instead of a service principal. See [use managed identities](#) for more information.

By default, *Basic* networking is used, and Azure Monitor for containers is enabled. Click **Review + create** and then **Create** when validation completes.

It takes a few minutes to create the AKS cluster. When your deployment is complete click **Go to resource**, or browse to the AKS cluster resource group, such as *myResourceGroup*, and select the AKS resource, such as *myAKSCluster*. The AKS cluster dashboard is shown, as in this example:

The following example output shows the Deployments and Services created successfully:

OutputCopy

```
deployment "azure-vote-back" created
service "azure-vote-back" created
deployment "azure-vote-front" created
service "azure-vote-front" created
```

Test the application

When the application runs, a Kubernetes service exposes the application front end to the internet. This process can take a few minutes to complete.

To monitor progress, use the [kubectl get service](#) command with the --watch argument.

ConsoleCopy

```
kubectl get service azure-vote-front --watch
```

Initially the *EXTERNAL-IP* for the *azure-vote-front* service is shown as *pending*.

OutputCopy

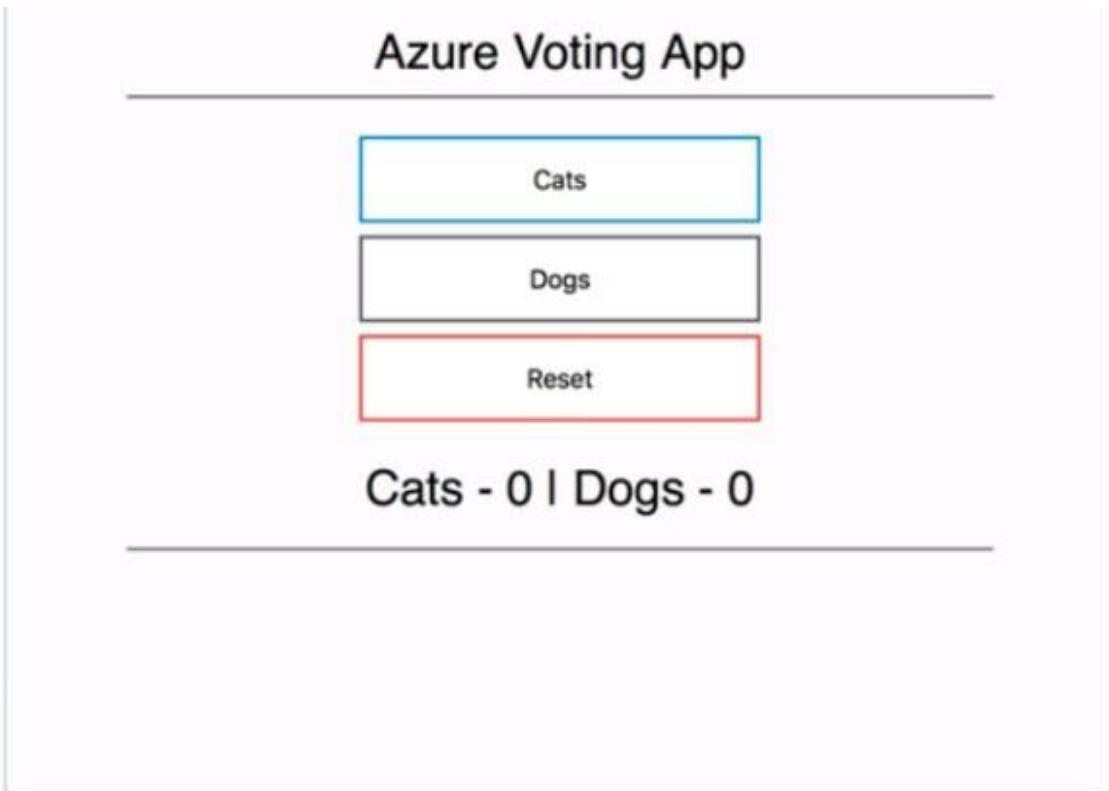
NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
azure-vote-front	LoadBalancer	10.0.37.27	<pending>	80:30572/TCP	6s

When the *EXTERNAL-IP* address changes from *pending* to an actual public IP address, use CTRL-C to stop the kubectl watch process. The following example output shows the valid public IP address assigned to the service:

OutputCopy

```
azure-vote-front LoadBalancer 10.0.37.27 52.179.23.131 80:30572/TCP 2m
```

To see the Azure Vote app in action, open a web browser to the external IP address of your service.



Monitor health and logs

When you created the cluster, Azure Monitor for containers was enabled. This monitoring feature provides health metrics for both the AKS cluster and pods running on the cluster.

It may take a few minutes for this data to populate in the Azure portal. To see current status, uptime, and resource usage for the Azure Vote pods, browse back to the AKS resource in the Azure portal, such as *myAKSCluster*. You can then access the health status as follows:

1. Under **Monitoring** on the left-hand side, choose **Insights**
2. Across the top, choose to **+ Add Filter**
3. Select *Namespace* as the property, then choose **<All but kube-system>**
4. Choose to view the **Containers**.

The *azure-vote-back* and *azure-vote-front* containers are displayed, as shown in the following example:

The screenshot shows the Azure Log Analytics workspace interface. On the left, there's a sidebar with sections for 'Active' workspaces (one selected: 'defaultworkspace-19da35d3-9a1a...'), 'Favorite workspaces', and a search bar. The main area has a 'Logs' tab selected, showing a query editor with a complex Kusto query and its results.

```

let startDateTime = datetime('2018-12-18T12:45:00.000Z');
let endDateTime = date
ContainerIdList = KubePodInventory
| where TimeGenerated >= startDateTime and T
ContainerName == '37d12cec-#2f6-11e9-8c58-8a43b98784de/azure-vote-back'
| where
| distinct ContainerID;
ContainerLog
| where TimeGenerated >=
| where ContainerID in (ContainerIdList)
| project LogEntrySource, LogEntr
ContainerID
| order by TimeGenerated desc
| render table

```

Completed

LogEntrySource	LogEntry	TimeGenerated [UT]
> stdout	1:M 18 Dec 2018 18:54:08.774 # Server initialized	2018-12-18T18:54:
> stdout	1:M 18 Dec 2018 18:54:08.774 * Ready to accept connections	2018-12-18T18:54:
> stdout	1:M 18 Dec 2018 18:54:08.774 # WARNING you have Transparent Hu...	2018-12-18T18:54:
> stdout	1:M 18 Dec 2018 18:54:08.774 # WARNING: The TCP backlog setting ...	2018-12-18T18:54:
> stdout	1:M 18 Dec 2018 18:54:08.774 * Running mode=standalone, port=63...	2018-12-18T18:54:
> stdout	1:C 18 Dec 2018 18:54:08.773 # Warning: no config file specified, usl...	2018-12-18T18:54:
> stdout	1:C 18 Dec 2018 18:54:08.773 # Redis version=5.0.3, bits=64, comm...	2018-12-18T18:54:
> stdout	1:C 18 Dec 2018 18:54:08.773 # c0000e0300e0000e Redis is starting o...	2018-12-18T18:54:

14 ← Page 1 of 1 → ⏪ ⏩ 50 ▾ Items per

Delete cluster

When the cluster is no longer needed, delete the cluster resource, which deletes associated resources. This operation can be completed in the Azure portal by clicking the **Delete** button on the AKS cluster dashboard. Alternatively, the [az aks delete](#) command can be used in the Cloud Shell:

Azure CLICopy

```
az aks delete --resource-group myResourceGroup --name myAKSCluster --no-wait
```

3. Create an ASP.NET Core Web API and configure monitoring.

Page No. 58

Create a basic ASP.NET web app

1. Launch Visual Studio 2019.
2. Select File > New > Project.
3. Select ASP.NET Web Application(.NET Framework) C#.
4. Enter a project name > Select Create.
5. Select MVC > Create.

Add Application Insights automatically

Automatically adding Application Insights to a template-based ASP.NET web app. |
within your ASP.NET web app project in Visual Studio:

1. Select Add Application Insights Telemetry > Application Insights Sdk
(local) > Next > Finish > Close.
2. Open the ApplicationInsights.config file.
3. Before the closing </ApplicationInsights> tag add a line containing the instrumentation key for your Application Insights resource. Your instrumentation key can be found on the overview pane of your newly created Application Insights resource that you created as part of the prerequisites for this article.

XMLCopy

```
<InstrumentationKey>your-instrumentation-key-goes-here</InstrumentationKey>
```

4. Select Project > Manage NuGet Packages > Updates > Update each Microsoft.ApplicationInsights NuGet package to the latest stable release.
5. Run your application by selecting IIS Express. A basic ASP.NET app will launch. You navigate the pages on the site telemetry will be sent to Application Insights.

Add Application Insights manually

This section will guide you through manually adding Application Insights to a template-based ASP.NET web app. This section assumes you are using a web app based on standard ASP.NET Framework MVC web app template.

1. Add the following NuGet packages and their dependencies to your project:
 - o Microsoft.ApplicationInsights.WindowsServer
 - o Microsoft.ApplicationInsights.Web
 - o Microsoft.AspNet.TelemetryCorrelation
2. In some cases, the ApplicationInsights.config file will be created for you automatically. If the file is already present, skip to step #4. If it is not created automatically,

you will need to create it yourself. At the same level in your project as the Global.asax file, create a new file called ApplicationInsights.config

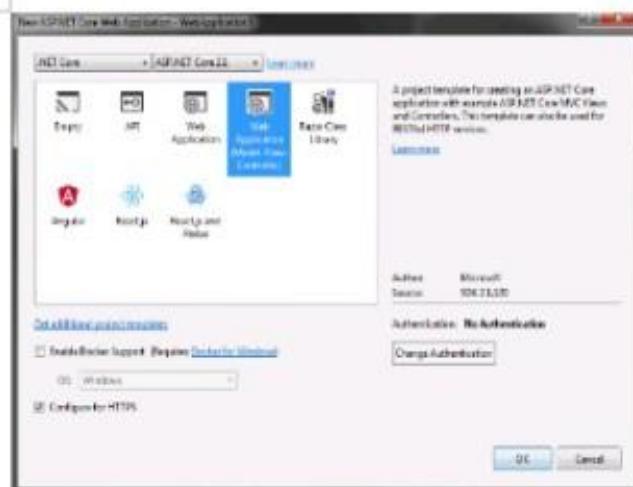
3. Copy the following XML configuration into your newly created file:

Practical No.4A - Create an Azure Kubernetes Service Cluster

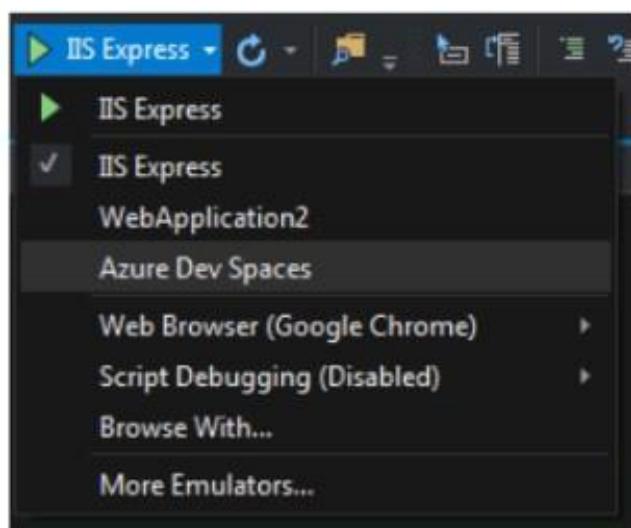
Steps –

1. Sign in Azure Portal (<https://portal.azure.com/>)
2. Create Resource by clicking on Create Resource option.
3. Select Kubernetes Services
4. Enter the subscription, resource group, kubernetes, Cluster name, Region, KI DNS Name prefix
5. Click on Review + Create Button
6. Click on Create Button

The screenshot shows the Azure portal interface for creating a Kubernetes cluster. On the left, the 'Create Kubernetes cluster' wizard is displayed with the 'Basics' tab selected. It includes fields for Subscription (Free Trial), Resource group (View CAD), Cluster details (Cluster name: CAD, Region: US East, Availability zones: Zones 1,2,3, Kubernetes version: 1.18.14 (default)), and a note about primary node pool. At the bottom are 'Review + create' and 'Next: Node pools >' buttons. On the right, the 'Kubernetes service' blade for the 'CAD' cluster is open, showing tabs for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Security, Namespaces, Workloads, Services and ingresses, Storage, Configuration, Settings, Node pools, Cluster configuration, Scale, Networking, Dev Spaces, Deployment center (preview), Policies, Properties, Locks, and Monitoring. The 'Node pools' tab is currently selected.



11. Click On Down Arrow of IIS Express
12. We can see option of Azure Dev Space



Practical No.4D - Configure Visual Studio Code to work with an azure kubernetes services Cluster

Steps –

1. Install visual studio code
2. Open extension window
3. Search for Azure Dev Space
4. Click Install

The screenshot shows the Microsoft Visual Studio interface with the title bar "EMPLOYEE - Microsoft Visual Studio (Administrator)". The menu bar includes File, Edit, View, Project, Build, Debug, Team, Tools, Test, Analyze, Window, Help. The toolbar has icons for Save, Undo, Redo, Cut, Copy, Paste, Find, Replace, and others. The status bar at the bottom shows "Any CPU" and "Azure Dev Spaces". The main code editor window displays the "EMPLOYEE.Controllers.ValuesController" file. The code is as follows:

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using Microsoft.AspNetCore.Mvc;
6
7  namespace EMPLOYEE.Controllers
8  {
9      [Route("api/{controller}")]
10     [ApiController]
11     public class ValuesController : ControllerBase
12     {
13         // GET api/values
14         [HttpGet]
15         [ProducesResponseType(typeof(IEnumerable<string>), StatusCodes.Status200OK)]
16         public ActionResult<IEnumerable<string>> Get()
17         {
18             return new string[] { "Azure", "Kubernetes", "Service" };
19         }
20     }
21 }
```

9. Save changes & press F5 button to run the application
10. Select cluster name & space > click on OK button

The screenshot shows the "Azure Dev Spaces" dialog box and the "Output" window.

Azure Dev Spaces Dialog:

- Microsoft account: deepaliwalanju9920@gmail.com
- Subscription: Free Trial (1f62cf0-e8a8-4d07-8dc7-43c1d565c449)
- Azure Kubernetes Service cluster: CAD
- Space: default

Output Window:

```
Output
Show output from: Azure Dev Spaces
NAMESPACE: default
STATUS: DEPLOYED
RESOURCES:
--> v1/Service
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
employee   ClusterIP  10.6.192.15    <none>        80/TCP      0s
--> v1beta2/Deployment
NAME      DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
employee  1        0        0          0          0s
--> v1beta1/Ingress
NAME      HOSTS      ADDRESS      PORTS      AGE
employee  default.employee.jpksgnkw2j.eus.azds.io  13.68.220.220  80      0s
--> v1/Pod(related)
NAME      READY  STATUS      RESTARTS  AGE
employee-6cf9fb99-z1v4r  0/2  Pending  0          0s
NOTES:
1. Get the application URL by running these commands:
  http://default.employee.jpksgnkw2j.eus.azds.io/
Building container image...
Sending build context to Docker daemon 17.92kB
Step 1/13 : FROM microsoft/aspnetcore-build:2.0
--> 06a6525397c2
```

2.Node.js API

Steps -

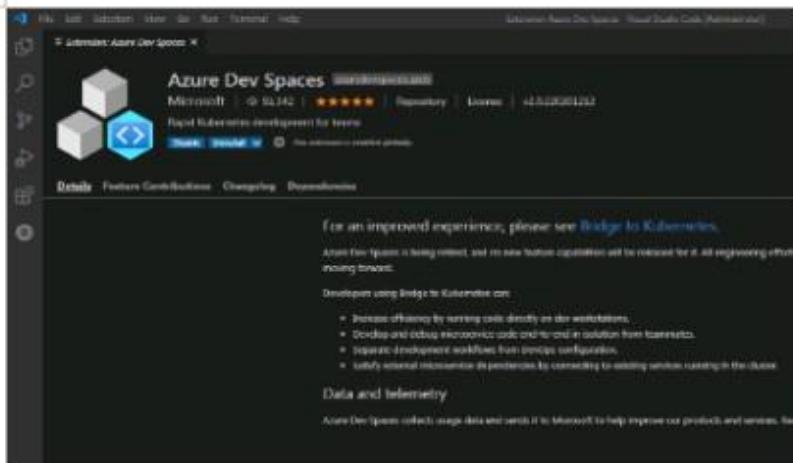
1. Create MYAPP folder
2. Open visual studio code
3. Open folder MYAPP in visual studio code
4. Create new file server.js & pacakge.json
5. From windows explorer create PUBLIC folder in MYAPP folder
6. Now create new file using public folder by using visual studio code
7. Create file index.html & app.css
8. Open command palette from view menu (ctrl+shift+p)
9. Enter azure dev space & click on it
10. Click on configure
11. Click DEBUG icon on left & then click on LAUNCH SERVER(AZDS)
12. OUTPUT will display in Output window

Server.js

```
var express = require('express');
var app = express();
app.use(express.static(__dirname + '/public'));
app.get('/', function (req, res) {
  res.sendFile(__dirname + '/public/index.html');
});
app.get('/api', function (req, res) {
  res.send('Hello from webfrontend');
});
var port = process.env.PORT || 80;
var server = app.listen(port, function () {
  console.log('Listening on port ' + port);
});
process.on("SIGINT", () => {
  process.exit(130 /* 128 + SIGINT */);
});
process.on("SIGTERM", () => {
  console.log("Terminating...");
  server.close();
});
```

Package.json

```
{
  "name": "webfrontend",
  "version": "0.1.0",
  "devDependencies": {
    "nodemon": "^1.18.10"
  },
  "dependencies": {
    "express": "^4.16.2",
    "request": "2.83.0"
  },
  "main": "server.js",
  "scripts": {
    "start": "node server.js"
```



5. Install Azure CLI (<https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest>)
 6. Open cmd
 7. Enter command az Login
 8. Enter Command to install AZDS utility
`az aks use-dev-spaces -n CAD -g CAD`

```
Administrator:~ C:\Windows\system32\cmd.exe -e az aks use-dev-spaces -n CLOUDAPP -g CLOUDAPP

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>az login
The default web browser has been opened at https://login.microsoftonline.com/con
nections/oauth2/authorize. Please continue the login in the web browser. If no web br
owser is available or if the web browser fails to open, use device code flow wit
h 'az login --use-device-code'.
You have logged in. Now let us find all the subscriptions to which you have acc
ess...
{
  "cloudName": "AzureCloud",
  "homeTenantId": "aaabfd46-4740-4963-abed-282dabe77cde",
  "id": "ff62cfdb-ea8-4d87-8dc7-43c1d565c449",
  "isDefault": true,
  "managedByTenants": [],
  "name": "Free Trial",
  "state": "Enabled",
  "tenantId": "aaabfd46-4740-4963-abed-282dabe77cde",
  "user": {
    "name": "deepalivalanju9920@gmail.com",
    "type": "user"
  }
}

C:\Users\Administrator>az aks use-dev-spaces -n CAD -g CAD
This command has been deprecated and will be removed in a future release.
For more information, please see https://github.com/Azure/dev-spaces/issues/410
Installing Dev Spaces commands...
A separate window will open to guide you through the installation process.
```

Practical No.4E - Deploy Application on AKS

1. Core web API

Steps -

- Steps**

 1. Open Visual Studio
 2. Click on New project
 3. Select Cloud Container Application for Kubernetes

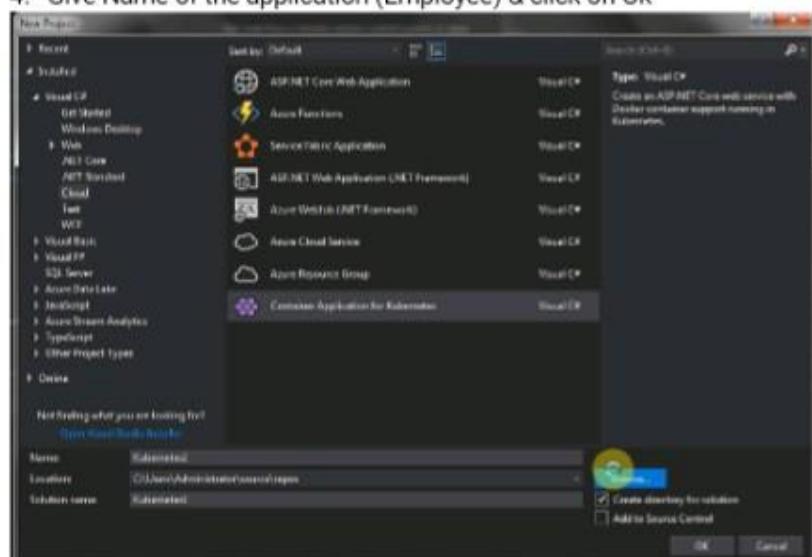
Practical No.4E - Deploy Application on AKS

1.Core web API

Steps –

1. Open Visual Studio
2. Click on New project
3. Select Cloud Container Application for Kubernetes

4. Give Name of the application (Employee) & click on Ok



5. Select API option & click on OK button



6. Make sure Azure Dev Space is selected in menu.
7. Open ValuesController.cs file
8. Edit line number 16 as shown in the screen shot

OUTPUT

The Debug console shows the log output.

```
> Executing task: C:\Program Files\Microsoft SDKs\Azure\Azure Dev Spaces CLI (Preview)\azds.exe up --port=50521:9229 --await-exec --keep-alive <
Synchronizing files...4s
Using dev space 'new01' with target 'kuber01'
Installing Helm chart...2s
Waiting for container image build...29s
Building container image...
Step 1/8 : FROM node:lts
Step 2/8 : ENV PORT 80
Step 3/8 : EXPOSE 80
Step 4/8 : WORKDIR /app
Step 5/8 : COPY package.json .
Step 6/8 : RUN npm install
Step 7/8 : COPY .
Step 8/8 : CMD ["npm", "start"]
Built container image in 45s
Waiting for container...52s
Service 'myapp' port 80 (http) is available via port forwarding
at http://localhost:50764
Terminal will be reused by tasks, press any key to close it.
```

```
    }
}

Index.html
<!doctype html>
<html ng-app="myApp">
<head>
<script src="https://ajax.googleapis.com/ajax/libs/
angularjs/1.5.3/angular.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/
libs/angular.js/1.5.3/angular-route.js"></script>
<script src="app.js"></script>
<link rel="stylesheet" href="app.css">
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-1q8mTJOASx8j1Au
+a5WDVnPi2lkFfwwEAa8hDDjZlpLegxhjVME1fgjWPGmkzs7"
crossorigin="anonymous">
<!-- Uncomment the next line -->
<!-- <meta name="viewport" content="width=device-width,
initial-scale=1"> -->
</head>
<body style="margin-left:10px; margin-right:10px;">
<div ng-controller="MainController">
<h2>Server Says</h2>
<div class="row">
<div class="col-xs-8 col-md-10">
<div ng-repeat="message in messages
track by $index">
<span class="message">{{message}}</
span>
</div>
</div>
<div class="col-xs-4 col-md-2">
<button class="btn btn-primary"
ng-click="
sayHelloToServer()">Say It
Again</button>
</div>
</div>
</div>
</body>
</html>
```

APP.css

```
.message {
font-family: Courier New, Courier, monospace;
font-weight: bold;
}
```

- a) Create an API Management Service.
- b) Create an API Gateway Service.

a) Create an API Management Service.

Steps for creating an API management service are mentioned below-

Step 1- Sign-in to your Azure Subscription Portal

Step 2- Search for “API Management”, then select API Management in order to create a service instance.

Step 3- As shown in Image 1 Select “API Management service”

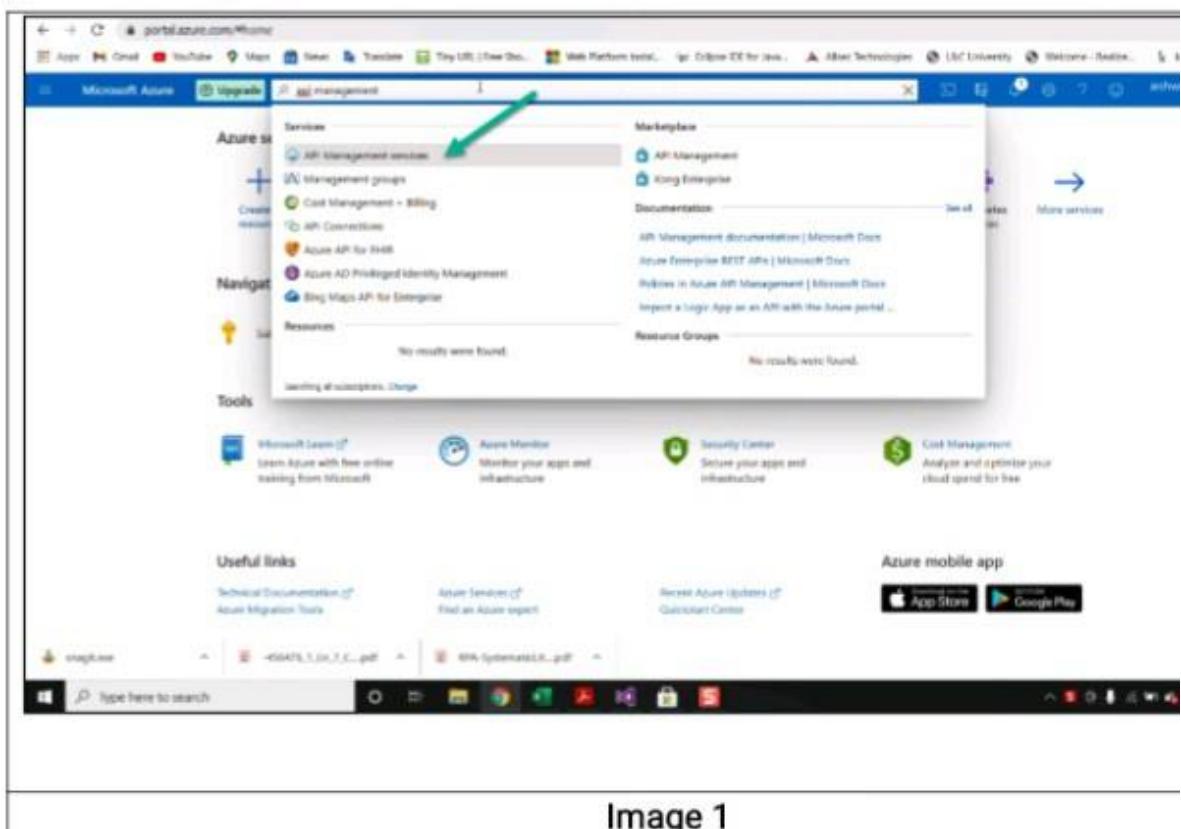


Image 1

Step 4- As API management service page is loaded, select "Create API management service" button. As shown in the image 2.

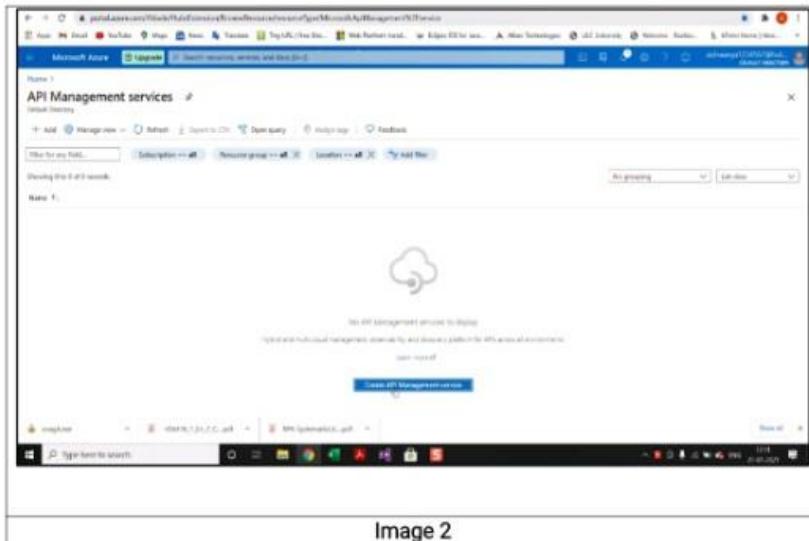


Image 2

Step 5- We will be able to see the Azure API Management service creation blade as shown in Image after clicking on the "Create API management service" button

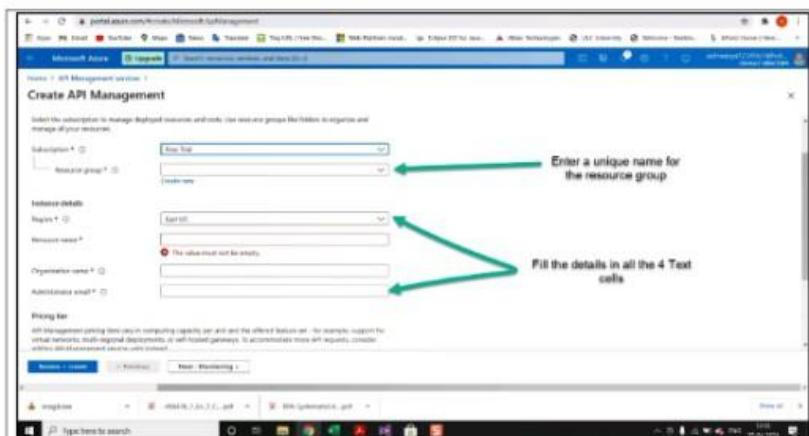


Image 3

Step 6- Provide a **name**. This name sets the URL of the API gateway and portal [Name- AzureManagementService]

Step 7-Select the subscription and **resource group** (or create a new resource group), and select the location. [Resource Name- AzureManagementService, Location- SouthEast Asia]

Step 8-Specify the **organization name**. This name will appear in the developer portal as the organization that publishes the API. [Name- CloudApplicationPractical]

Step 9- Specify the **email address** of the administrator. The user who creates the service instance will be the default administrator, so it's best

Step 13-Once the Validation is done and the label displays "Validation Passed", Click on **create** button to create the instance of the service.

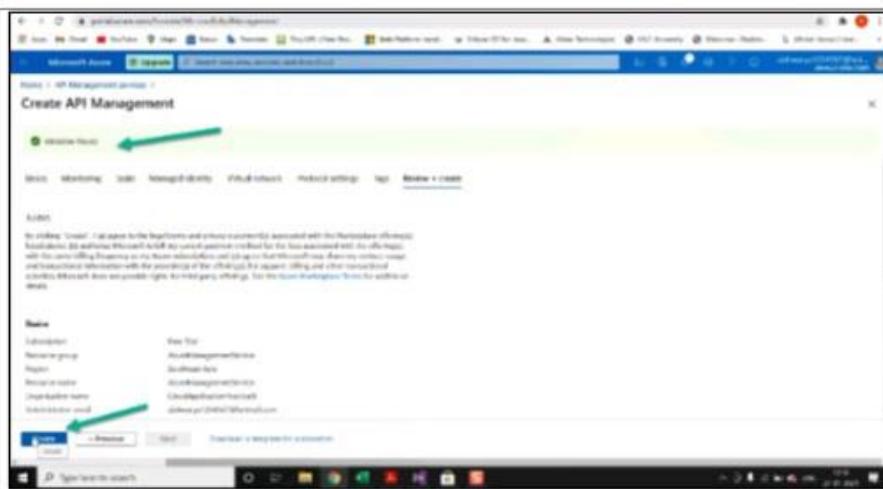


Image 6

Step 14- After clicking on the create button the API management service instance will go to deployment stage as shown in image 7. After the deployment is complete the instance will be shown as in image 8

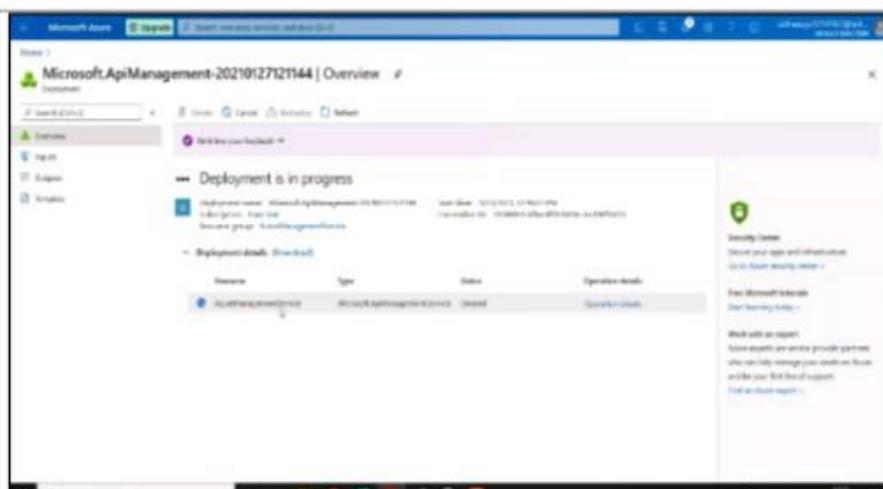


Image 7

Image 6

Step 14- After clicking on the create button the API management service instance will go to deployment stage as shown in image 7. After the deployment is complete the instance will be shown as in image 8

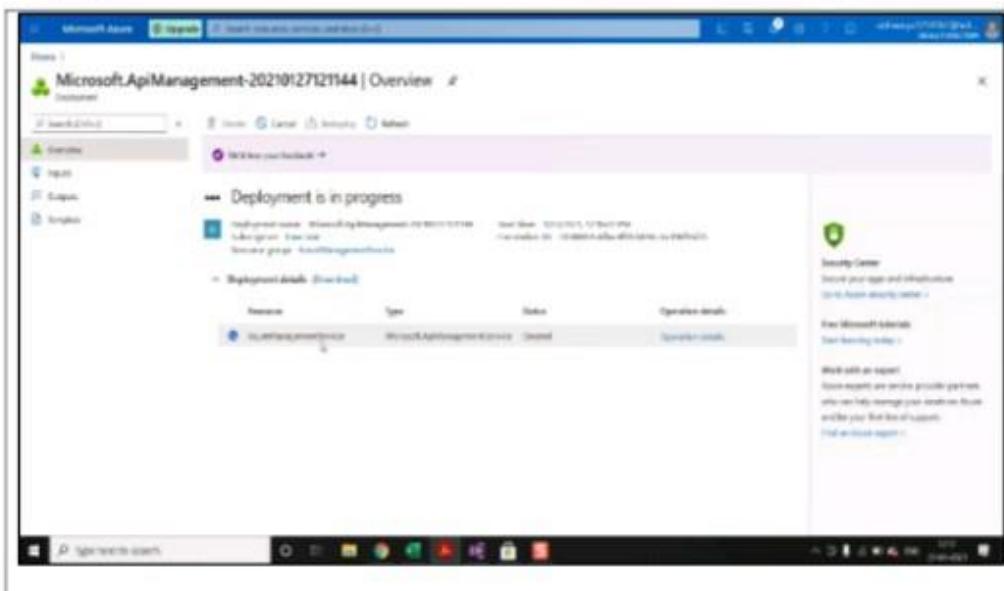


Image 7

Next step is Import an API into API Management and test the API in the Azure portal

Step 15- In the Azure portal, search for and select API Management services.

Step 16- On the API Management services page, select your API Management instance as in image 9.

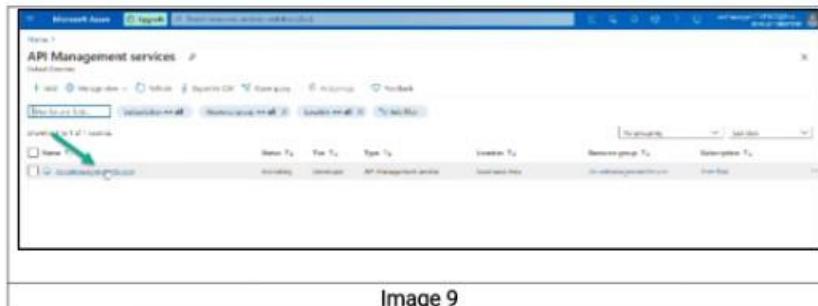


Image 9

Step 17- In the left navigation of API Management instance, select APIs.

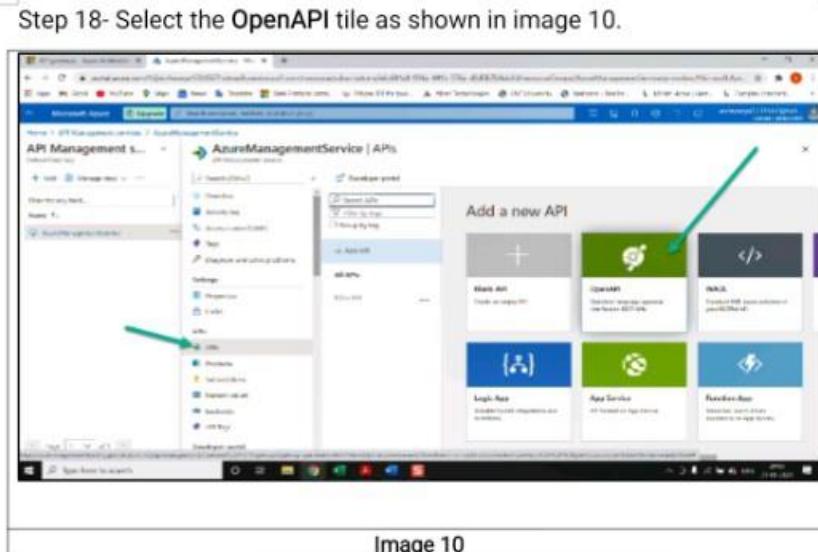


Image 10

Step 18- Select the OpenAPI tile as shown in image 10.

Setting	Value
OpenAPI specification	https://conferenceapi.azurewebsites.net?format=json
Display name	After you enter the preceding service URL, API Management fills out this field based on the JSON. .(automatic)
Name	After you enter the preceding service URL, API Management fills out this field based on the JSON. .(automatic)
Description	After you enter the preceding service URL, API Management fills out this field based on the JSON.(automatic)

Gateways Managed

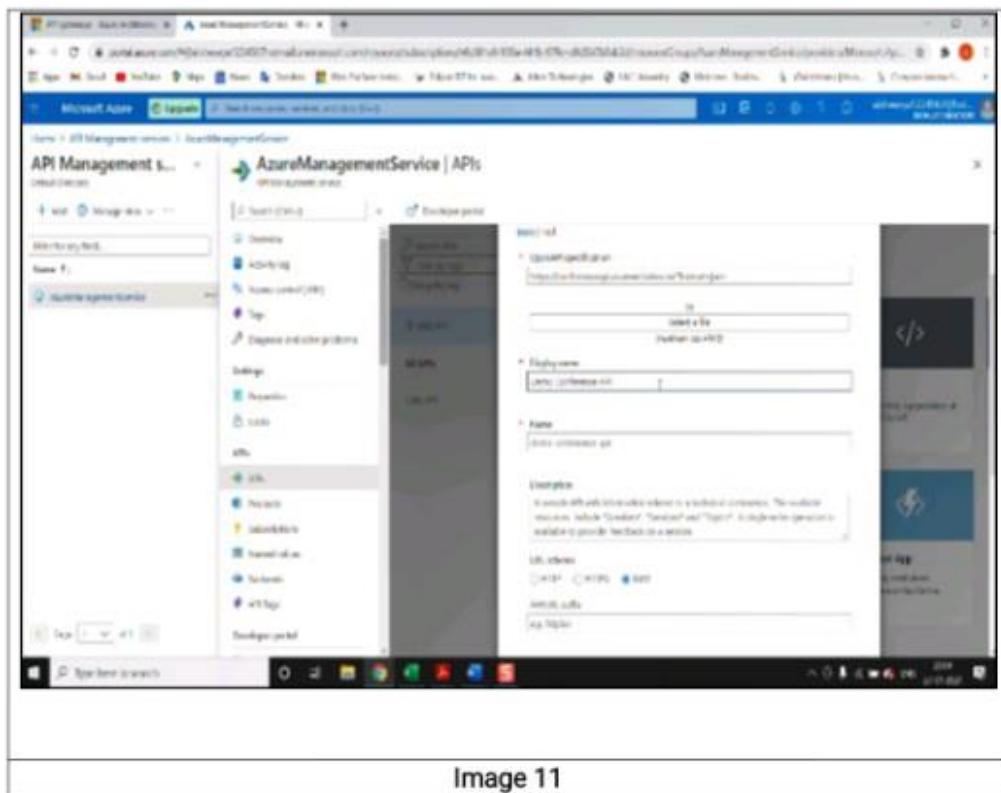


Image 11

Next steps are to Test the new API in the Azure portal

Step 21- In the left navigation of your API Management instance, select APIs > **Demo Conference API**.

Step 22- Select the Test tab, and then select **GetSpeakers**. The page shows Query parameters and Headers, if any. The Ocp-Apim-Subscription-Key is filled in automatically for the subscription key associated with this API.

Step 22- Select **Send** as shown in image 12.

Output -

b) Create an API Gateway Service.

Steps for creating an API gateway service are mentioned below-

Step 1- Sign in to your AWS account.

Step 2- search for API Gateway, and create an AWS Gateway instance as shown in Image 1.

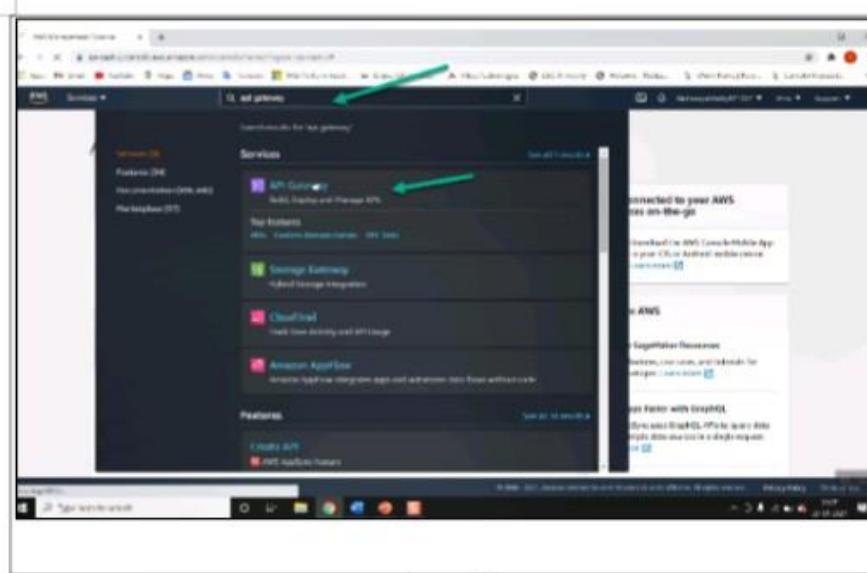


Image 1

Step 3- Select Import from the REST API selection as shown in image 2.

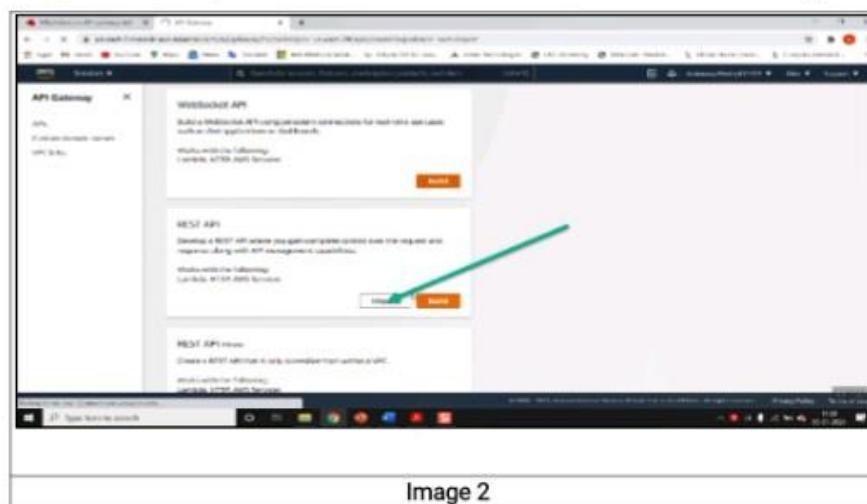


Image 2

Step 4- Select Import from swagger and copy or select the swagger file as shown in image 3.

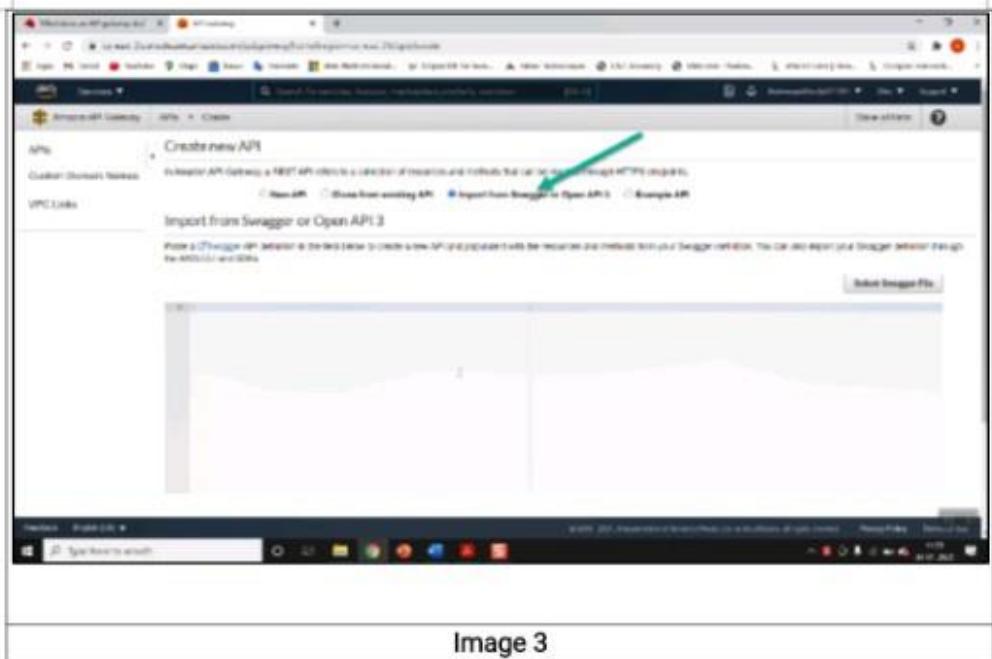


Image 3

Step 5- To Create an API definition in Swagger. First login to the swagger account. As depicted in image 4.

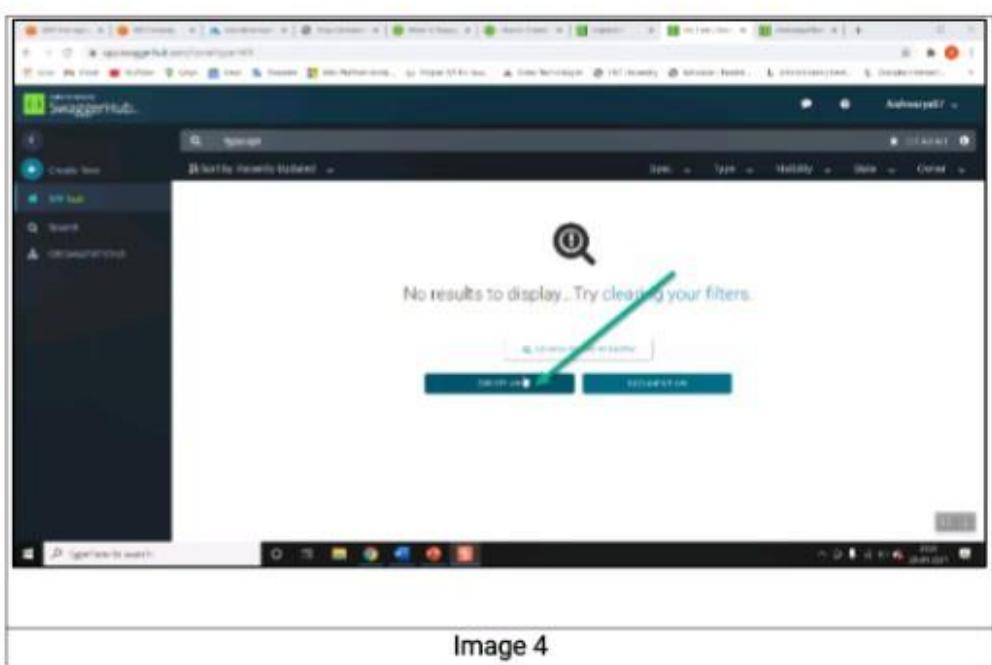


Image 4

Step 6- Then fill the Form which includes the version, name and other details click on Create API button. As depicted in image 5.

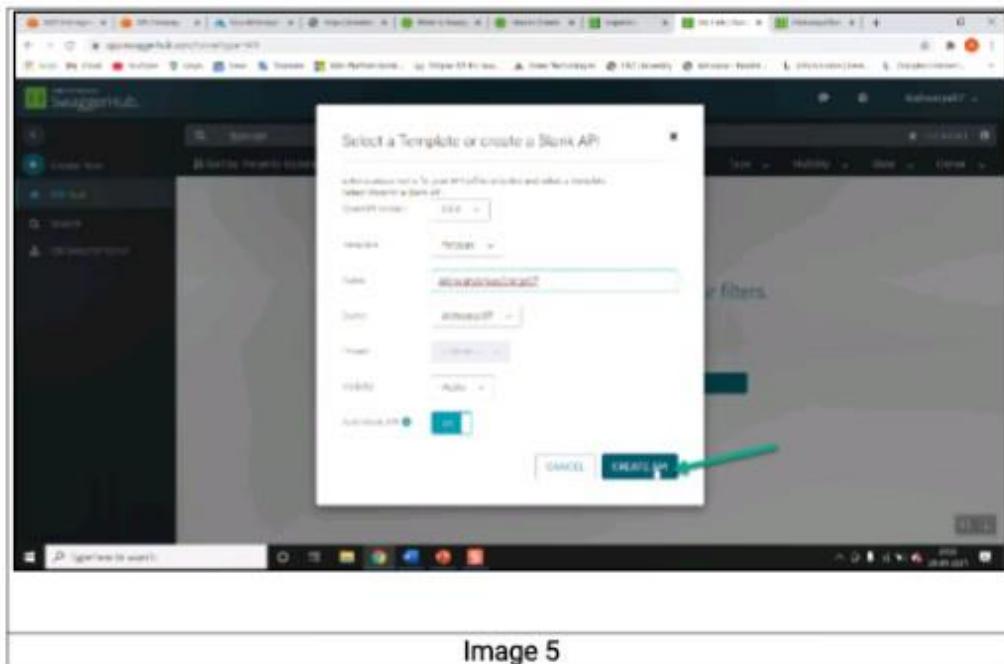
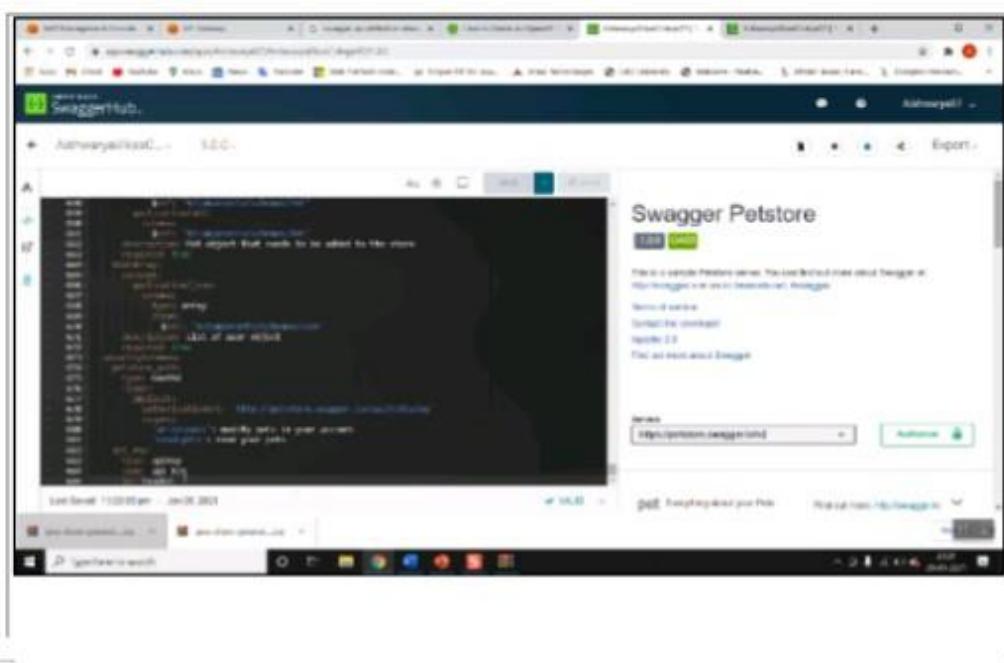


Image 5

Step 7- Copy the API and definition from swagger as shown in image 6



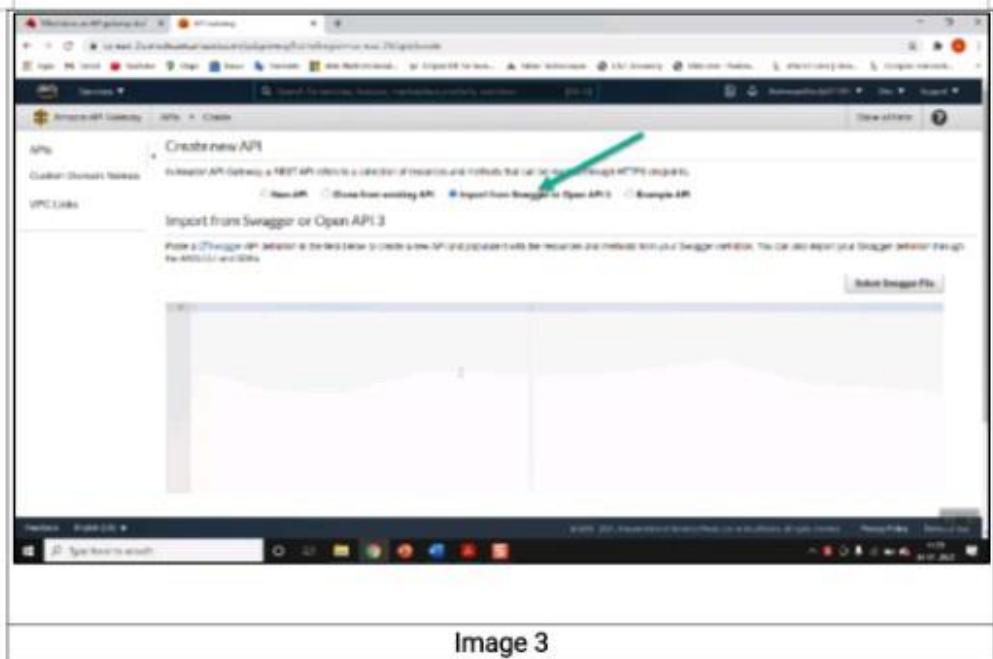


Image 3

Step 5- To Create an API definition in Swagger. First login to the swagger account. As depicted in image 4.

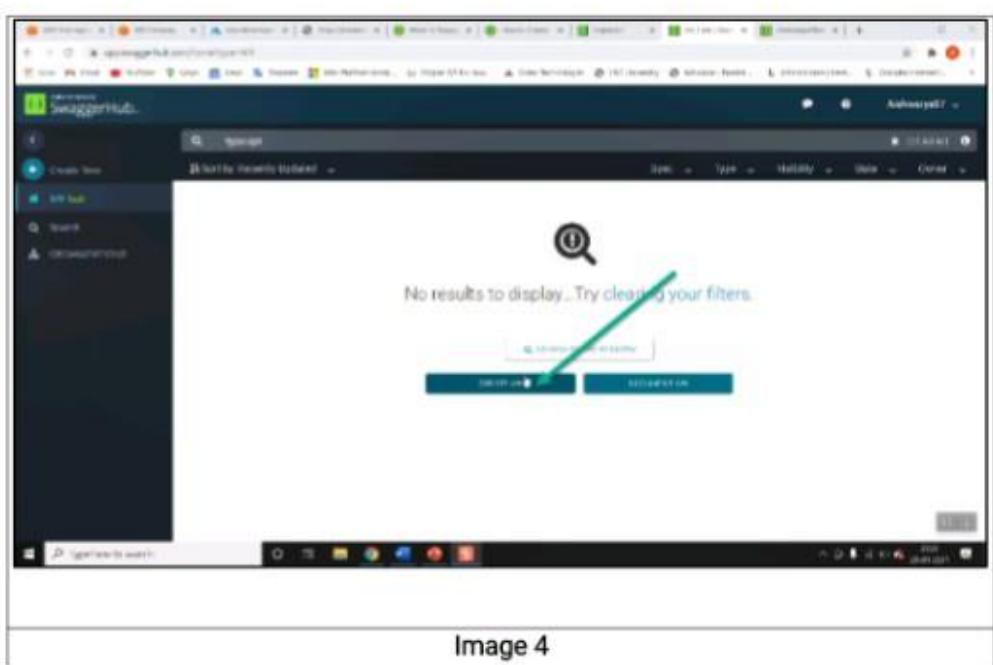


Image 4

Step 6- Then fill the Form which includes the version, name and other details click on Create API button. As depicted in image 5.

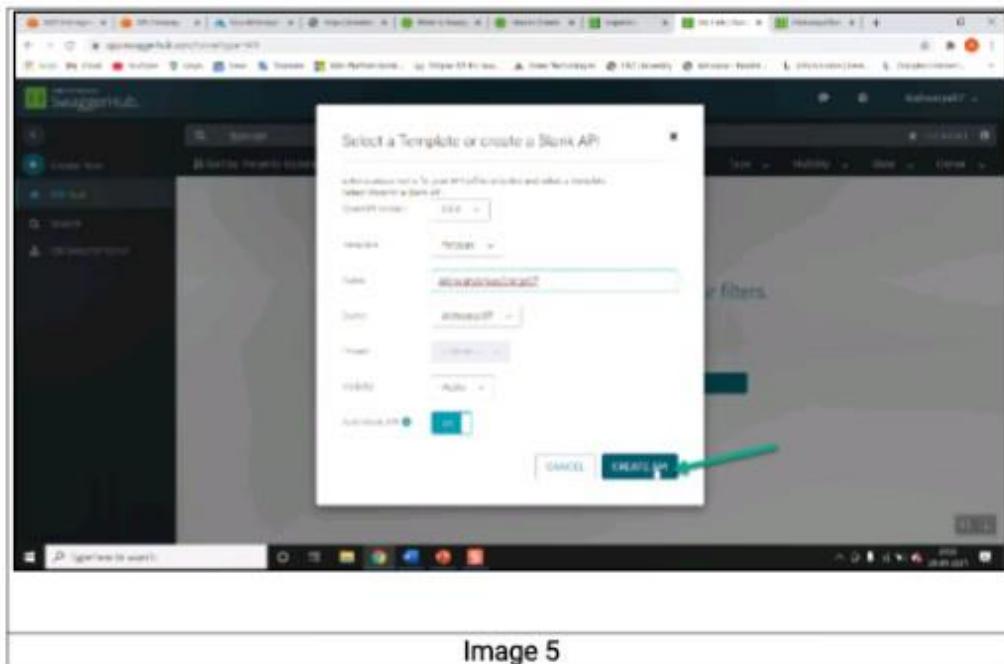


Image 5

Step 7- Copy the API and definition from swagger as shown in image 6

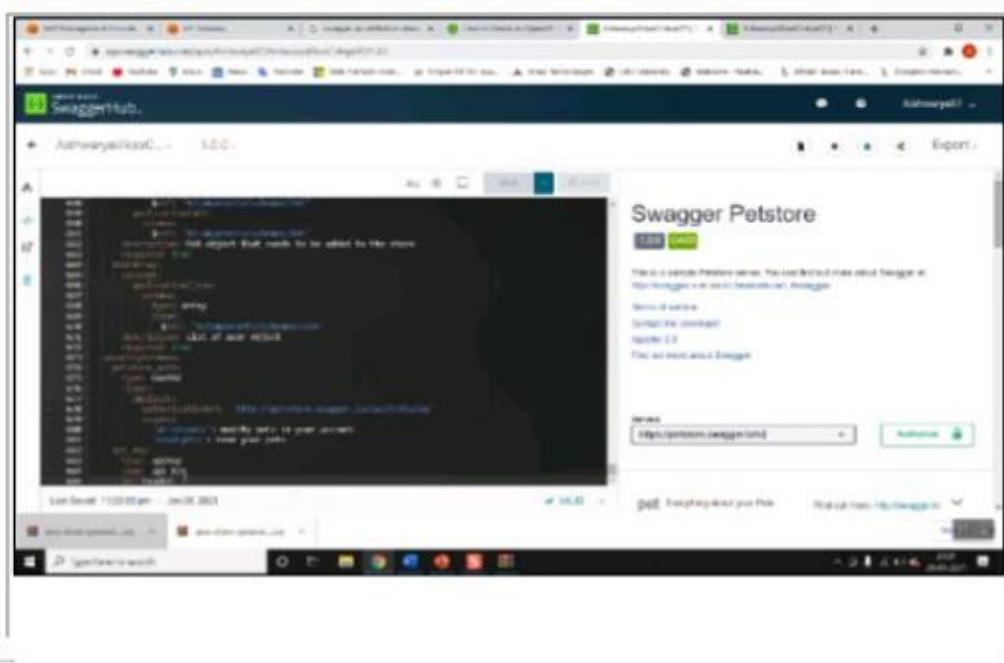


Image 6

Step 8-Paste the definition and in In the Settings section, select the endpoint type. Select “Edge Optimized” option and then click on import button as shown in image 7

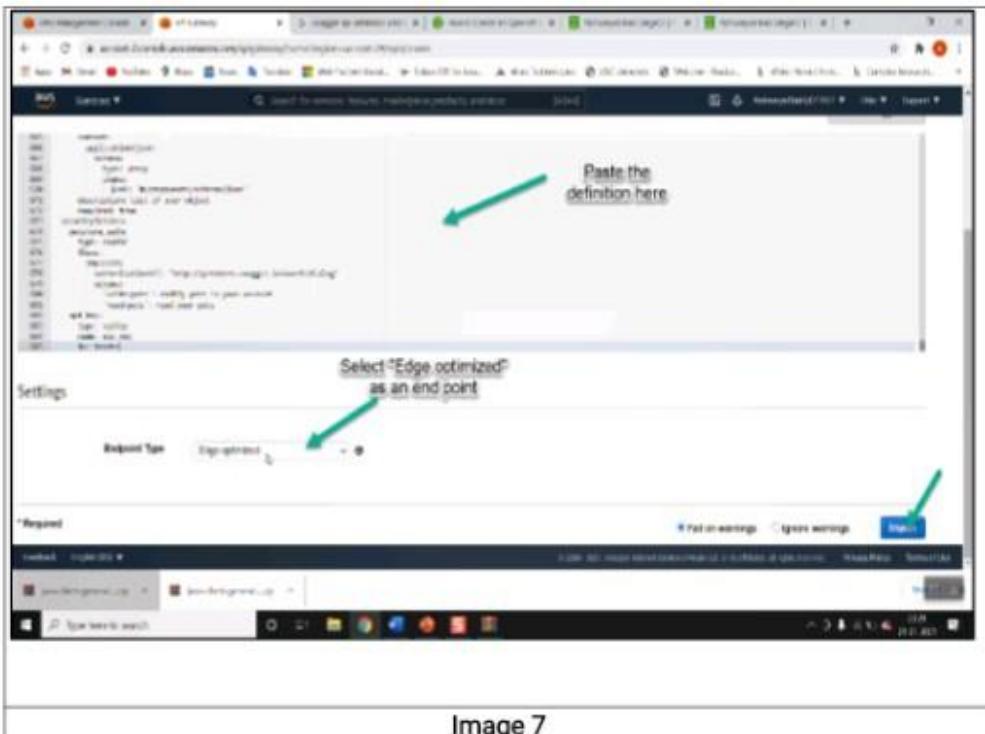


Image 7

API definition from swagger is mentioned below-

openapi: 3.0.0

info:

description: |

This is a sample Petstore server. You can find

out more about Swagger at

[<http://swagger.io>](http://swagger.io) or on

[irc.freenode.net, #swagger](http://swagger.io/irc/).

version: "1.0.0"

title: Swagger Petstore

```
  termsOfService: 'http://swagger.io/terms/'

  contact:
    email: apiteam@swagger.io

  license:
    name: Apache 2.0
    url: 'http://www.apache.org/licenses/LICENSE-2.0.html'

  servers:
    # Added by API Auto Mocking Plugin
    - description: SwaggerHub API Auto Mocking
      url:
        https://virtserver.swaggerhub.com/Aishwarya07/AishwaryaVikasColege
        07/1.0.0
      - url: 'https://petstore.swagger.io/v2'

  tags:
    - name: pet
      description: Everything about your Pets
      externalDocs:
        description: Find out more
        url: 'http://swagger.io'
    - name: store
      description: Access to Petstore orders
    - name: user
      description: Operations about user
      externalDocs:
        description: Find out more about our store
        url: 'http://swagger.io'

  paths:
    /pet:
      post:
```

```
'404':  
    description: Pet not found  
    security:  
        - api_key: []  
post:  
tags:  
    - pet  
summary: Updates a pet in the store with form data  
operationId: updatePetWithForm  
parameters:  
    - name: petId  
        in: path  
        description: ID of pet that needs to be updated  
        required: true  
        schema:  
            type: integer  
            format: int64  
responses:  
'405':  
    description: Invalid input  
    security:  
        - petstore_auth:  
            - 'write:pets'  
            - 'read:pets'  
requestBody:  
content:  
    application/x-www-form-urlencoded:  
        schema:
```

```
    type: object
    properties:
        name:
            description: Updated name of the pet
            type: string
        status:
            description: Updated status of the pet
            type: string
    delete:
        tags:
            - pet
        summary: Deletes a pet
        operationId: deletePet
        parameters:
            - name: api_key
              in: header
              required: false
              schema:
                  type: string
            - name: petId
              in: path
              description: Pet id to delete
              required: true
              schema:
                  type: integer
                  format: int64
        responses:
            '400':
```

```
tags:
- pet
summary: Add a new pet to the store
operationId: addPet
responses:
'405':
  description: Invalid input
security:
- petstore_auth:
  - 'write:pets'
  - 'read:pets'
requestBody:
$ref: '#/components/requestBodies/Pet'
put:
tags:
- pet
summary: Update an existing pet
operationId: updatePet
responses:
'400':
  description: Invalid ID supplied
'404':
  description: Pet not found
'405':
  description: Validation exception
security:
- petstore_auth:
  - 'write:pets'
```

```
  deprecated: true
  '/pet/{petId}':
    get:
      tags:
        - pet
      summary: Find pet by ID
      description: Returns a single pet
      operationId: getPetById
      parameters:
        - name: petId
          in: path
          description: ID of pet to return
          required: true
          schema:
            type: integer
            format: int64
      responses:
        '200':
          description: successful operation
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Pet'
            application/xml:
              schema:
                $ref: '#/components/schemas/Pet'
        '400':
          description: Invalid ID supplied
```

```
tags:
  - user
summary: Delete user
description: This can only be done by the logged in user.
operationId: deleteUser
parameters:
  - name: username
    in: path
    description: The name that needs to be deleted
    required: true
    schema:
      type: string
responses:
  '400':
    description: Invalid username supplied
  '404':
    description: User not found
externalDocs:
  description: Find out more about Swagger
  url: 'http://swagger.io'
components:
  schemas:
    Order:
      type: object
      properties:
        id:
          type: integer
          format: int64
```

```
responses:  
  default:  
    description: successful operation  
  '/user/{username}':  
    get:  
      tags:  
        - user  
      summary: Get user by user name  
      operationId: getUserByName  
      parameters:  
        - name: username  
          in: path  
          description: The name that needs to be fetched. Use user1 for testing.  
          required: true  
        schema:  
          type: string  
      responses:  
        '200':  
          description: successful operation  
          content:  
            application/json:  
              schema:  
                $ref: '#/components/schemas/User'  
            application/xml:  
              schema:  
                $ref: '#/components/schemas/User'  
        '400':  
          description: Invalid username supplied
```

```
'404':
  description: User not found
put:
  tags:
    - user
  summary: Updated user
  description: This can only be done by the logged in user.
  operationId: updateUser
  parameters:
    - name: username
      in: path
      description: name that need to be updated
      required: true
      schema:
        type: string
  responses:
    '400':
      description: Invalid user supplied
    '404':
      description: User not found
requestBody:
  content:
    application/json:
      schema:
        $ref: '#/components/schemas/User'
      description: Updated user object
      required: true
delete:
```

```
- user
  summary: Creates list of users with given input array
  operationId: createUsersWithListInput
  responses:
    default:
      description: successful operation
  requestBody:
    $ref: '#/components/requestBodies/UserArray'
/user/login:
  get:
    tags:
      - user
    summary: Logs user into the system
    operationId: loginUser
    parameters:
      - name: username
        in: query
        description: The user name for login
        required: true
        schema:
          type: string
      - name: password
        in: query
        description: The password for login in clear text
        required: true
        schema:
          type: string
    responses:
```

```
        '200':  
            description: successful operation  
            headers:  
                X-Rate-Limit:  
                    description: calls per hour allowed by the user  
                    schema:  
                        type: integer  
                        format: int32  
                X-Expires-After:  
                    description: date in UTC when token expires  
                    schema:  
                        type: string  
                        format: date-time  
            content:  
                application/json:  
                    schema:  
                        type: string  
                application/xml:  
                    schema:  
                        type: string  
        '400':  
            description: Invalid username/password supplied  
/user/logout:  
    get:  
        tags:  
            - user  
        summary: Logs out current logged in user session  
        operationId: logoutUser
```

```
        description: Invalid ID supplied
        '404':
            description: Order not found
    delete:
        tags:
            - store
        summary: Delete purchase order by ID
        description: >-
            For valid response try integer IDs with positive integer value.\\ \
            Negative or non-integer values will generate API errors
        operationId: deleteOrder
        parameters:
            - name: orderId
              in: path
              description: ID of the order that needs to be deleted
              required: true
              schema:
                  type: integer
                  format: int64
                  minimum: 1
        responses:
            '400':
                description: Invalid ID supplied
            '404':
                description: Order not found
    /user:
        post:
            tags:
```

```
requestBodies:  
  Pet:  
    content:  
      application/json:  
        schema:  
          $ref: '#/components/schemas/Pet'  
      application/xml:  
        schema:  
          $ref: '#/components/schemas/Pet'  
    description: Pet object that needs to be added to the store  
    required: true  
  UserArray:  
    content:  
      application/json:  
        schema:  
          type: array  
          items:  
            $ref: '#/components/schemas/User'  
    description: List of user object  
    required: true  
  securitySchemes:  
    petstore_auth:  
      type: oauth2  
      flows:  
        implicit:  
          authorizationUrl: 'http://petstore.swagger.io/oauth/dialog'  
          scopes:  
            'write:pets': modify pets in your account
```

```
- user

summary: Create user
description: This can only be done by the logged in user.
operationId: createUser
responses:
  default:
    description: successful operation
requestBody:
  content:
    application/json:
      schema:
        $ref: '#/components/schemas/User'
        description: Created user object
        required: true
/user/createWithArray:
  post:
    tags:
      - user
    summary: Creates list of users with given input array
    operationId: createUsersWithArrayInput
    responses:
      default:
        description: successful operation
    requestBody:
      $ref: '#/components/requestBodies/UserArray'
/user/createWithList:
  post:
    tags:
```

```
- api_key: []  
/store/order:  
  post:  
    tags:  
      - store  
    summary: Place an order for a pet  
    operationId: placeOrder  
    responses:  
      '200':  
        description: successful operation  
        content:  
          application/json:  
            schema:  
              $ref: '#/components/schemas/Order'  
          application/xml:  
            schema:  
              $ref: '#/components/schemas/Order'  
      '400':  
        description: Invalid Order  
    requestBody:  
      content:  
        application/json:  
          schema:  
            $ref: '#/components/schemas/Order'  
        description: order placed for purchasing the pet  
        required: true  
  '/store/order/{orderId}':  
    get:
```

```
    type: object
    properties:
      id:
        type: integer
        format: int64
      name:
        type: string
      xml:
        name: Tag
    Pet:
      type: object
      required:
        - name
        - photoUrls
      properties:
        id:
          type: integer
          format: int64
        category:
          $ref: '#/components/schemas/Category'
        name:
          type: string
          example: doggie
        photoUrls:
          type: array
          xml:
            name: photoUrl
        wrapped: true
```

```
tags:
  - store

summary: Find purchase order by ID
description: >
  For valid response try integer IDs with value >= 1 and <= 10.\ \ Other
  values will generated exceptions
operationId: getOrderByld
parameters:
  - name: orderld
    in: path
    description: ID of pet that needs to be fetched
    required: true
    schema:
      type: integer
      format: int64
      minimum: 1
      maximum: 10
responses:
  '200':
    description: successful operation
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/Order'
      application/xml:
        schema:
          $ref: '#/components/schemas/Order'
  '400':
```

```
    items:
      type: string
    tags:
      type: array
    xml:
      name: tag
      wrapped: true
  items:
    $ref: '#/components/schemas/Tag'
  status:
    type: string
    description: pet status in the store
  enum:
    - available
    - pending
    - sold
  xml:
    name: Pet
  ApiResponse:
    type: object
    properties:
      code:
        type: integer
        format: int32
      type:
        type: string
      message:
        type: string
```

```
        description: Invalid ID supplied  
        '404':  
            description: Pet not found  
        security:  
            - petstore_auth:  
                - 'write:pets'  
                - 'read:pets'  
    '/pet/{petId}/uploadImage':  
        post:  
            tags:  
                - pet  
            summary: uploads an image  
            operationId: uploadFile  
            parameters:  
                - name: petId  
                in: path  
                description: ID of pet to update  
                required: true  
                schema:  
                    type: integer  
                    format: int64  
            responses:  
                '200':  
                    description: successful operation  
                    content:  
                        application/json:  
                            schema:  
                                $ref: '#/components/schemas/ApiResponse'
```

```
petId:  
    type: integer  
    format: int64  
  
quantity:  
    type: integer  
    format: int32  
  
shipDate:  
    type: string  
    format: date-time  
  
status:  
    type: string  
    description: Order Status  
  
enum:  
    - placed  
    - approved  
    - delivered  
  
complete:  
    type: boolean  
    default: false  
  
xml:  
    name: Order  
  
Category:  
    type: object  
    properties:  
        id:  
            type: integer  
            format: int64  
        name:
```

```
        security:
          - petstore_auth:
              - 'write:pets'
              - 'read:pets'
        requestBody:
          content:
            application/octet-stream:
              schema:
                type: string
                format: binary
        /store/inventory:
          get:
            tags:
              - store
            summary: Returns pet inventories by status
            description: Returns a map of status codes to quantities
            operationId: getInventory
            responses:
              '200':
                description: successful operation
                content:
                  application/json:
                    schema:
                      type: object
                      additionalProperties:
                        type: integer
                        format: int32
            security:
```

```
    type: string
    xml:
      name: Category
  User:
    type: object
    properties:
      id:
        type: integer
        format: int64
      username:
        type: string
      firstName:
        type: string
      lastName:
        type: string
      email:
        type: string
      password:
        type: string
      phone:
        type: string
      userStatus:
        type: integer
        format: int32
        description: User Status
      xml:
        name: User
  Tag:
```

'read:pets': read your pets

```
api_key:  
  type: apiKey  
  name: api_key  
  in: header
```

Output-

If the import is successful, we can view the "*API Structure in AWS API Gateway*" where we can see methods of the API. As shown in image 8

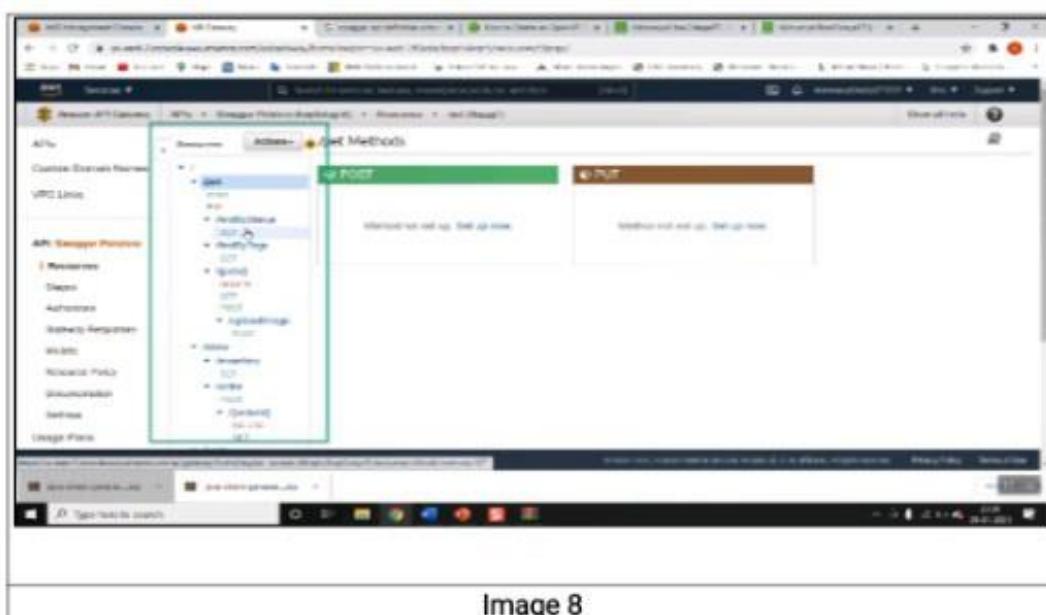


Image 8

Create a basic ASP.NET web app

1. Launch Visual Studio 2019.
2. Select **File > New > Project**.
3. Select **ASP.NET Web Application(.NET Framework) C#**.
4. Enter a project name > **Select Create**.
5. Select **MVC > Create**.

Add Application Insights automatically

Automatically adding Application Insights to a template-based ASP.NET web app within your ASP.NET web app project in Visual Studio:

1. Select **Add Application Insights Telemetry > Application Insights Sdk (local) > Next > Finish > Close**.
2. Open the ApplicationInsights.config file.
3. Before the closing `</ApplicationInsights>` tag add a line containing the instrumentation key for your Application Insights resource. Your instrumentation key can be found on the overview pane of your newly created Application Insights resource that you created as part of the prerequisites for this article.

XMLCopy

```
<InstrumentationKey>your-instrumentation-key-goes-here</InstrumentationKey>
```

4. Select **Project > Manage NuGet Packages > Updates > Update** each Microsoft.ApplicationInsights NuGet package to the latest stable release.
5. Run your application by selecting **IIS Express**. A basic ASP.NET app will log information to Application Insights. As you navigate the pages on the site telemetry will be sent to Application Insights.

Add Application Insights manually

This section will guide you through manually adding Application Insights to a template-based ASP.NET web app. This section assumes you are using a web app based on the standard ASP.NET Framework MVC web app template.

1. Add the following NuGet packages and their dependencies to your project:
 - o [Microsoft.ApplicationInsights.WindowsServer](#)
 - o [Microsoft.ApplicationInsights.Web](#)
 - o [Microsoft.AspNet.TelemetryCorrelation](#)
2. In some cases, the ApplicationInsights.config file will be created for you automatically. If the file is already present, skip to step #4. If it is not created automatically, add the following code to the ApplicationInsights.config file:

```
<InstrumentationKey>your-instrumentation-key-goes-here</InstrumentationKey>
```

'read:pets': read your pets

```
api_key:  
  type: apiKey  
  name: api_key  
  in: header
```

Output-

If the import is successful, we can view the "*API Structure in AWS API Gateway*" where we can see methods of the API. As shown in image 8

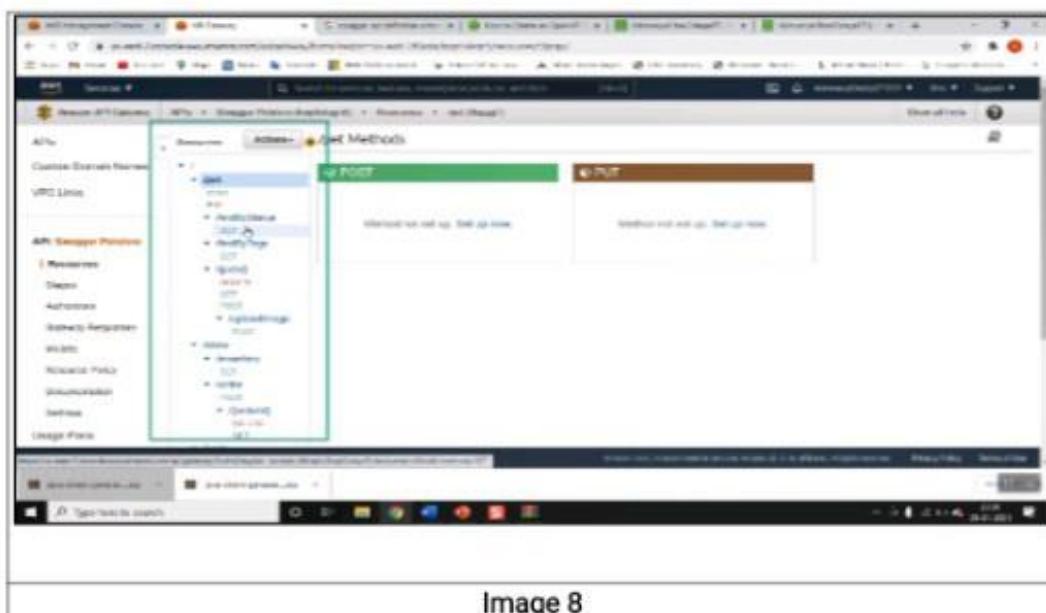


Image 8

PDF Created Using



Camera Scanner

Easily Scan documents & Generate PDF



<https://play.google.com/store/apps/details?id=photo.pdf.maker>