**Lesson13 Azure App Configuration**

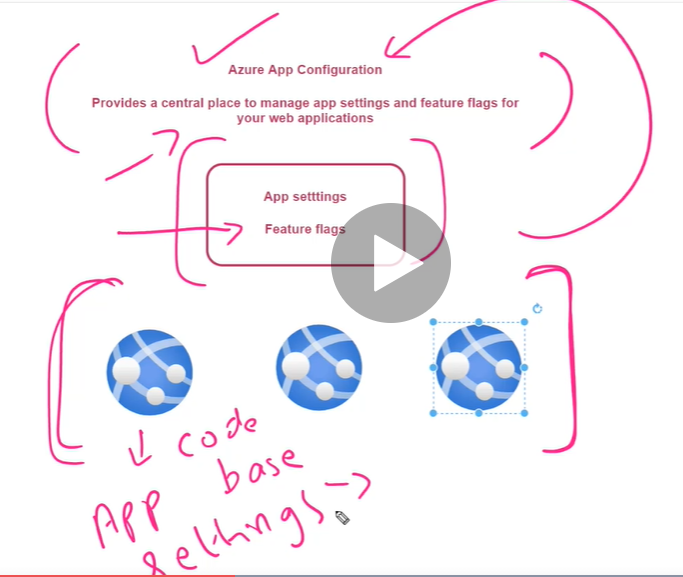
**Notes: -**

**1-by default each Azure Web App has its own Configuration, you can centerline these configurations through the Azure App Configuration resource**

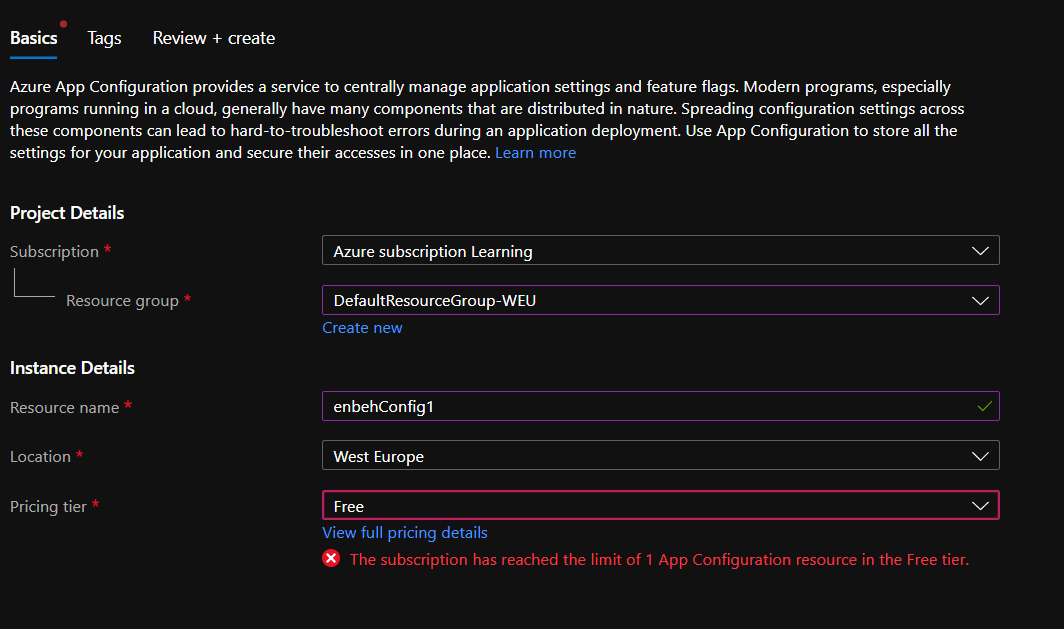
**2-there are two types of settings**

**A-App Settings**

**B-Feature Flags**



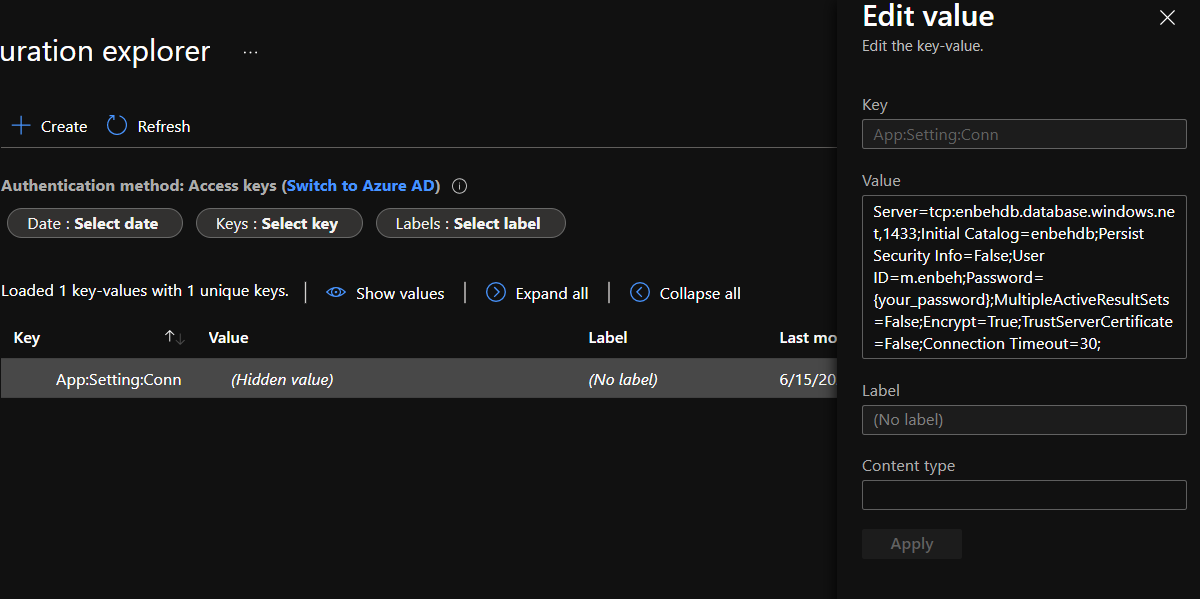
3-on Azure > create Azure App Configuration >



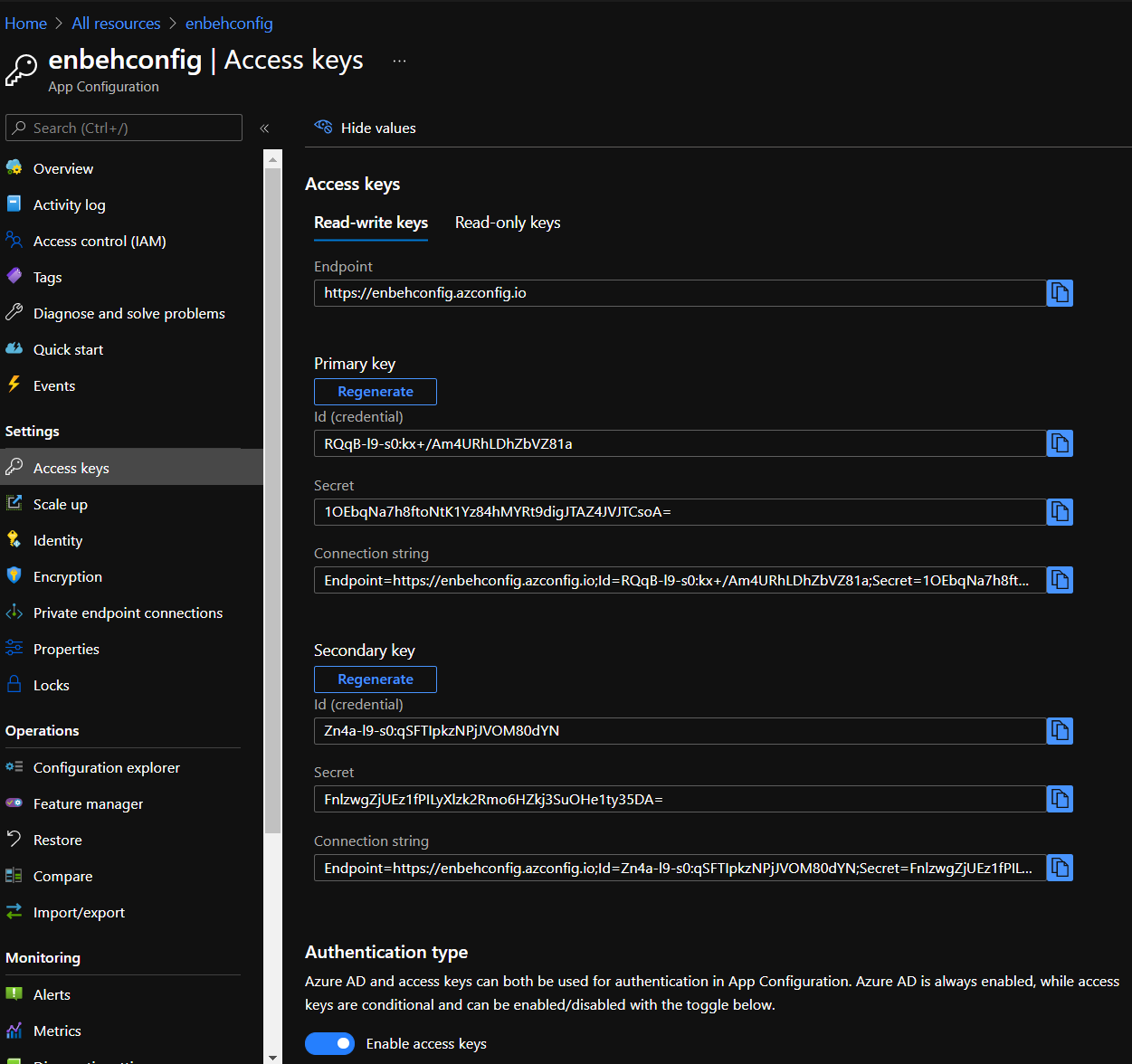
**App Settings Configuration**

**1-on Azure App Configuration > Configuration Explorer > we set key value pair as below**

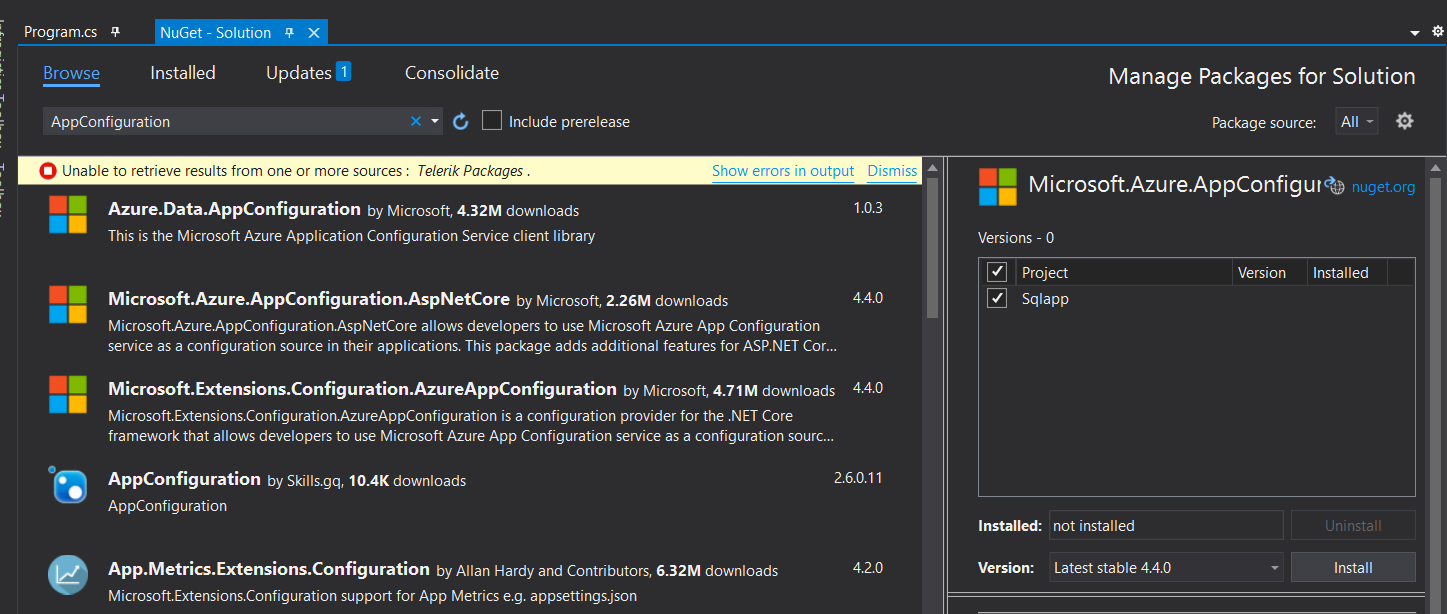
**(Value is the connection string that exist in Azure SQL server)**



**2-on Azure App Configuration > Access Keys > show values and copy the primary key**



**3-on VS2019 > install the NUGET package Microsoft.Azure.AppConfiguration.AspNetCore**



**4-on program.cs we set the following code to configure connect to the Azure App Config by copy the primary connection string into the connection configuration as below**

**using Microsoft.AspNetCore.Hosting;**

**using Microsoft.Extensions.Configuration;**

**using Microsoft.Extensions.Hosting;**

**namespace Sqlapp{**

**public class Program{**

**public static void Main(string[] args){**

**CreateHostBuilder(args).Build().Run();}**

**public static IHostBuilder CreateHostBuilder(string[] args) =>**

**Host.CreateDefaultBuilder(args)**

**.ConfigureWebHostDefaults(webBuilder =>**

**webBuilder.ConfigureAppConfiguration(config => {**

**var settings = config.Build();**

**//we set configuration by connect to the Azure App Configuration primary connection string**

**config.AddAzureAppConfiguration("Endpoint=https://enbehconfig.azconfig.io;Id=RQqB-l9-s0:kx+/Am4URhLDhZbVZ81a;Secret=1OEbqNa7h8ftoNtK1Yz84hMYRt9digJTAZ4JVJTCsoA=");**

**}).UseStartup<Startup>());}}**

**5-we only need to set the connection string on the CourseService.cs as below**

**private SqlConnection GetConnection(){**

**// Here we are creating the SQL connection**

**//return new SqlConnection(\_configuration.GetConnectionString("SQLConnection"));**

**return new SqlConnection(\_configuration["App:Setting:Conn"]);}**

**Feature Flags**

**Notes: -**

**feature flag is just flag that used to turn on / off feature on your web app**

**(By add feature flag in azure app config and manipulate connect to web app, you can show / hide menus from enable / disable options in Azure app config)**

**Steps: -**

**1-VS 2019 > create new project asp.net core web application > install the packages**

**Microsoft.Azure.AppConfiguration.AspNetCore**

**Microsoft.FeatureManagement.AspNetCore**

**2-on Program.cs we set the following code as below**

**using Microsoft.AspNetCore.Hosting;**

**using Microsoft.Extensions.Configuration;**

**using Microsoft.Extensions.Hosting;**

**namespace FeatureApp{**

**public class Program{**

**public static void Main(string[] args){CreateHostBuilder(args).Build().Run();}**

**public static IHostBuilder CreateHostBuilder(string[] args) =>**

**Host.CreateDefaultBuilder(args)**

**.ConfigureWebHostDefaults(webBuilder =>**

**webBuilder.ConfigureAppConfiguration(config =>{**

**var settings = config.Build();**

**//we enable using feature flags as below**

**config.AddAzureAppConfiguration(**

**opt => opt.Connect("Endpoint=https://enbehconfig.azconfig.io;Id=RQqB-l9-s0:kx+/Am4URhLDhZbVZ81a;Secret=1OEbqNa7h8ftoNtK1Yz84hMYRt9digJTAZ4JVJTCsoA=")**

**.UseFeatureFlags());**

**}).UseStartup<Startup>());}}**

**3-on startup.cs we add feature manager middleware as below**

**public void ConfigureServices(IServiceCollection services){**

**services.AddControllersWithViews();**

**services.AddFeatureManagement();}**

**4-on FeatureController.cs we set the following code as below**

**using FeatureApp.Models;**

**using Microsoft.AspNetCore.Mvc;**

**using Microsoft.FeatureManagement;**

**using Microsoft.FeatureManagement.Mvc;**

**namespace FeatureApp.Controllers{**

**public class FeatureController : Controller{**

**private readonly IFeatureManager \_featureManager;**

**public FeatureController(IFeatureManager featureManager){**

**\_featureManager = featureManager;}**

**//it will check if the feature flag is enable it will allow access**

**[FeatureGate(FeatureFlag.Staging)]**

**public IActionResult Index(){return View();}}}**

**5-on Models > add enum called FeatureFlag**

**namespace FeatureApp.Models{**

**public enum FeatureFlag{**

**Staging = 1}}**

**6-on \_Layout.cshtml we set the following code as below**

**<feature name="staging">**

**<li class="nav-item">**

**<a class="nav-link text-dark" asp-area="" asp-controller="Feature" asp-action="Index">**

**Feature Option</a></li>**

**</feature>**

**7-on \_viewImport we have to import the FeatureManagment.AspNetCore**

@using FeatureApp

@using FeatureApp.Models

@addTagHelper \*, Microsoft.AspNetCore.Mvc.TagHelpers

@addTagHelper \*, Microsoft.FeatureManagement.AspNetCore

**8-on the Azure App Configuration > Feature Flag > add flag called staging as below**

