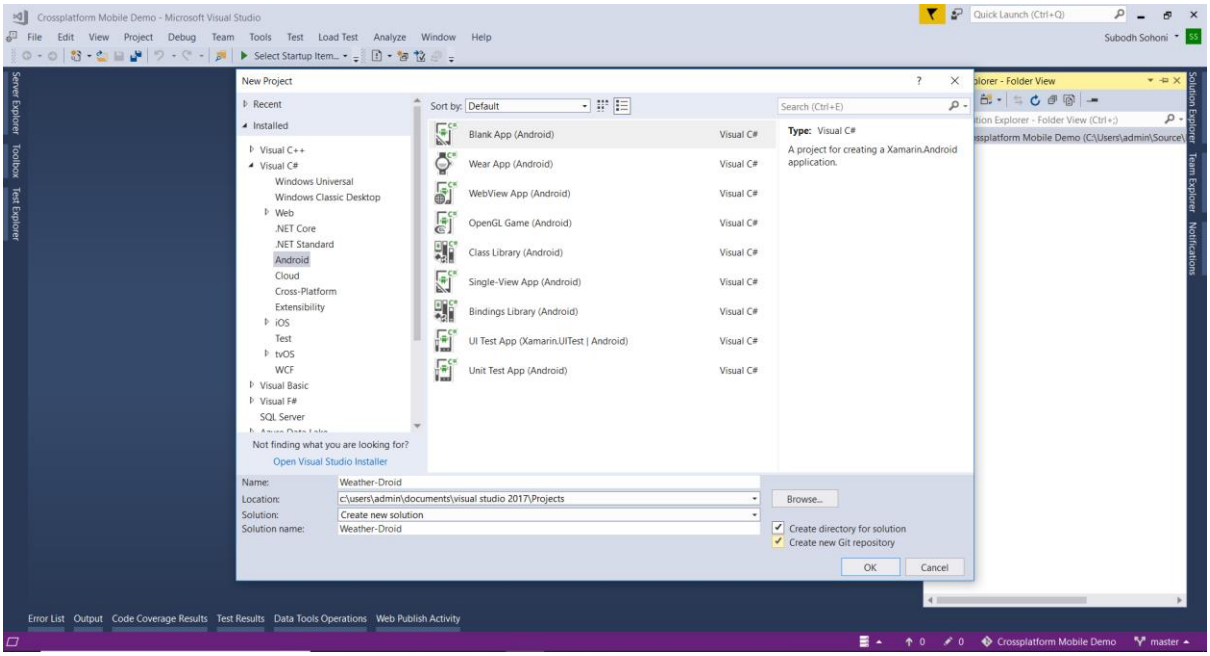
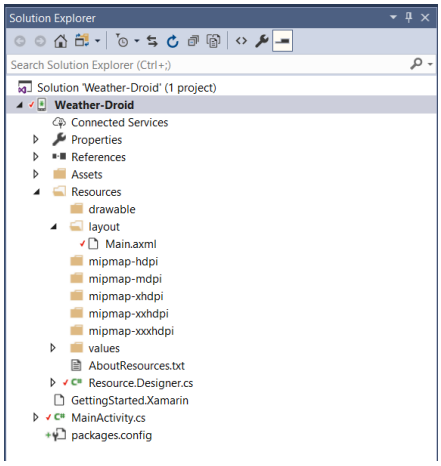
Hands on Lab: App Innovation - Cross Platform Mobile Application Development

Exercise 1: Create, Build and Distribute Xamarin Native App

1. Open Visual Studio 2017 and Create new project. Select the project type as Android – Blank Application. Name it as Weather.Droid.



2. From the folder Resources – layout select the file Main.axml which is created by the wizard.



3. Open the Source tab of Main.axml and replace the code with following code:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:minWidth="25px"
    android:minHeight="25px">
    <RelativeLayout
        android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:id="@+id/relativeLayout1"
    android:background="#545454">
    <TextView
        android:text="Search by Zip Code"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/ZipCodeSearchLabel"
        android:layout_marginStart="10dp"
        android:textColor="@android:color/white"
        android:textStyle="bold" />
    <TextView
        android:text="Zip Code"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/ZipCodeSearchLabel"
        android:id="@+id/ZipCodeLabel"
        android:layout_marginStart="10dp"
        android:layout_marginTop="6dp"
        android:textColor="@android:color/white" />
    <EditText
        android:inputType="number"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/ZipCodeLabel"
        android:id="@+id/zipCodeEntry"
        android:layout_marginStart="10dp"
        android:layout_marginBottom="10dp"
        android:width="165dp"
        android:textColor="@android:color/white" />
    <Button
        android:text="Get Weather"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/zipCodeEntry"
        android:id="@+id/weatherBtn"
        android:layout_alignBottom="@id/zipCodeEntry"
        android:layout_marginStart="20dp"
        android:width="165dp" />
</RelativeLayout>
<TextView
    android:text="Location"
    android:textAppearance="?android:attr/textAppearanceSmall"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/locationLabel"
    android:layout_marginStart="10dp"
    android:layout_marginTop="10dp" />
<TextView
    android:textAppearance="?android:attr/textAppearanceMedium"
```

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/locationText"
    android:layout_marginStart="20dp"
    android:layout_marginBottom="10dp" />
<TextView
    android:text="Temperature"
    android:textAppearance="?android:attr/textAppearanceSmall"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/tempLabel"
    android:layout_marginStart="10dp" />
<TextView
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/tempText"
    android:layout_marginBottom="10dp"
    android:layout_marginStart="20dp" />
<TextView
    android:text="Wind Speed"
    android:textAppearance="?android:attr/textAppearanceSmall"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/windLabel"
    android:layout_marginStart="10dp" />
<TextView
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/windText"
    android:layout_marginBottom="10dp"
    android:layout_marginStart="20dp" />
<TextView
    android:text="Humidity"
    android:textAppearance="?android:attr/textAppearanceSmall"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/humidtyLabel"
    android:layout_marginStart="10dp" />
<TextView
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/humidityText"
    android:layout_marginBottom="10dp"
    android:layout_marginStart="20dp" />
<TextView
    android:text="Visibility"
    android:textAppearance="?android:attr/textAppearanceSmall"
```

```
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/visibilityLabel"
            android:layout_marginStart="10dp" />
        <TextView
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/visibilityText"
            android:layout_marginBottom="10dp"
            android:layout_marginStart="20dp" />
        <TextView
            android:text="Time of Sunrise"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/sunriseLabel"
            android:layout_marginStart="10dp" />
        <TextView
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/sunriseText"
            android:layout_marginBottom="10dp"
            android:layout_marginStart="20dp" />
        <TextView
            android:text="Time of Sunset"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/sunsetLabel"
            android:layout_marginStart="10dp" />
        <TextView
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/sunsetText"
            android:layout_marginBottom="10dp"
            android:layout_marginStart="20dp" />
    </LinearLayout>
```

4. Build the project once. It will add some Ids in the project.Add a new .NET class library project named WeatherApp.
5. Sign up for a free weather API key at http://openweathermap.org/appid. This API key will allow the application to obtain the weather for any United States zip code. (It does not work for postal codes outside of the United States.)
6. Right-click the **WeatherApp** project and select **Add > Class...**. In the **Add New Item** dialog, name the file **Weather.cs**. You'll use this class to store data from the weather data service.

7. Replace the entire contents of **Weather.cs** with the following code:

```
namespace WeatherApp

{

    public class Weather

    {

        // Because labels bind to these values, set them to an empty
string to

        // ensure that the label appears on all platforms by default.

        public string Title { get; set; } = " ";

        public string Temperature { get; set; } = " ";

        public string Wind { get; set; } = " ";

        public string Humidity { get; set; } = " ";

        public string Visibility { get; set; } = " ";

        public string Sunrise { get; set; } = " ";

        public string Sunset { get; set; } = " ";

    }

}
```

8. Add another class to the .NET Standard project named **DataService.cs**. You'll use this class to process JSON data from the weather data service.
9. Replace the entire contents of **DataService.cs** with the following code:

```
using System.Net.Http;

using System.Threading.Tasks;

using Newtonsoft.Json;


namespace WeatherApp

{

    public class DataService
```

```
    {
        public static async Task<dynamic> getDataFromService(string
queryString)
        {
            HttpClient client = new HttpClient();
            var response = await client.GetAsync(queryString);

            dynamic data = null;
            if (response != null)
            {
                string json = response.Content.ReadAsStringAsync().Result;
                data = JsonConvert.DeserializeObject(json);
            }

            return data;
        }
    }
}
```

10. Add a third class to the .NET Standard library named **Core.cs**. You'll use this class to form a query string with a zip code, call the weather data service, and populate an instance of the **Weather** class.
11. Replace the contents of **Core.cs** with the following code:

```
using System;
using System.Threading.Tasks;

namespace WeatherApp
{
```

```
    public class Core

    {

        public static async Task<Weather> GetWeather(string zipCode)

        {

            //Sign up for a free API key at
http://openweathermap.org/appid

            string key = "YOUR API KEY HERE";

            string queryString =
"http://api.openweathermap.org/data/2.5/weather?zip="

                + zipCode + ",us&appid=" + key + "&units=imperial";



            //Make sure developers running this sample replaced the API
key

            if (key == "YOUR API KEY HERE")

            {

                throw new ArgumentException("You must obtain an API key
from openweathermap.org/appid and save it in the 'key' variable.");

            }



            dynamic results = await
DataService.GetDataFromService(queryString).ConfigureAwait(false);



            if (results["weather"] != null)

            {

                Weather weather = new Weather();

                weather.Title = (string)results["name"];

                weather.Temperature = (string)results["main"]["temp"] + "
F";

                weather.Wind = (string)results["wind"]["speed"] + " mph";
```

```
                weather.Humidity = (string)results["main"]["humidity"] + "
%";

                weather.Visibility =
(string)results["weather"][0]["main"];



                DateTime time = new System.DateTime(1970, 1, 1, 0, 0, 0,
0);

                DateTime sunrise =
time.AddSeconds((double)results["sys"]["sunrise"]);

                DateTime sunset =
time.AddSeconds((double)results["sys"]["sunset"]);

                weather.Sunrise = sunrise.ToString() + " UTC";

                weather.Sunset = sunset.ToString() + " UTC";

                return weather;

            }

            else

            {

                return null;

            }

        }

    }

}
```

12. Replace the first occurrence of *YOUR API KEY HERE* with the API key you obtained in step 1. It still needs quotes around it!
13. Delete **MyClass.cs** in the .NET Standard library because it won't be used.
14. Build the **WeatherApp** project to make sure the code is correct.
15. Open the **MainActivity.cs** file of the **Weather** project in the code editor and replace its contents with the code below. This code calls the `GetWeather` method that you defined in your shared code. Then, in the UI of the app, it shows the data that is retrieved from that method.

```
using System;
using Android.App;
using Android.Widget;
using Android.OS;

namespace WeatherApp.Droid
{
    [Activity(Label = "Sample Weather App",
              Theme = "@android:style/Theme.Material.Light",
              MainLauncher = true)]
    public class MainActivity : Activity
    {
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);

            SetContentView(Resource.Layout.Main);

            Button button = FindViewById<Button>(Resource.Id.weatherBtn);

            button.Click += Button_Click;
        }

        private async void Button_Click(object sender, EventArgs e)
        {
            EditText zipCodeEntry =
FindViewById<EditText>(Resource.Id.zipCodeEntry);

            if (!String.IsNullOrEmpty(zipCodeEntry.Text))
            {
                Weather weather = await
Core.GetWeather(zipCodeEntry.Text);
                FindViewById<TextView>(Resource.Id.locationText).Text =
weather.Title;
                FindViewById<TextView>(Resource.Id.tempText).Text =
weather.Temperature;
                FindViewById<TextView>(Resource.Id.windText).Text =
weather.Wind;
                FindViewById<TextView>(Resource.Id.visibilityText).Text =
weather.Visibility;
                FindViewById<TextView>(Resource.Id.humidityText).Text =
weather.Humidity;
                FindViewById<TextView>(Resource.Id.sunriseText).Text =
weather.Sunrise;
                FindViewById<TextView>(Resource.Id.sunsetText).Text =
weather.Sunset;
            }
        }
```

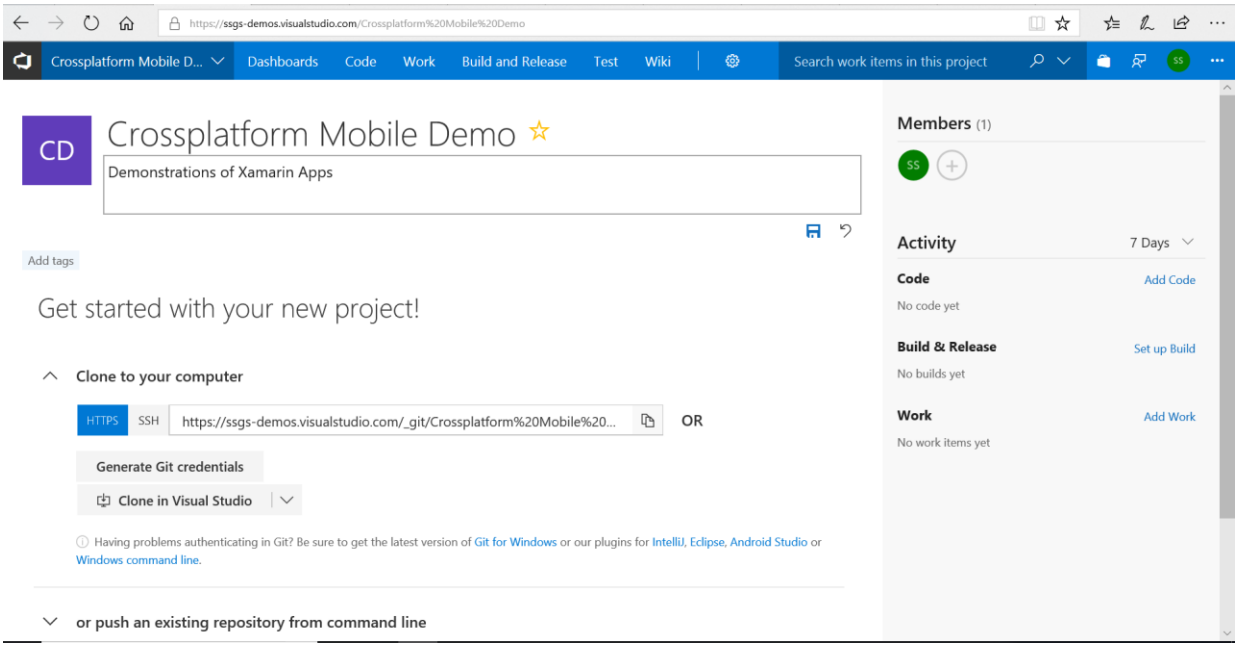
```
    }
}
```

Notice that the activity has been given a theme for a light background.

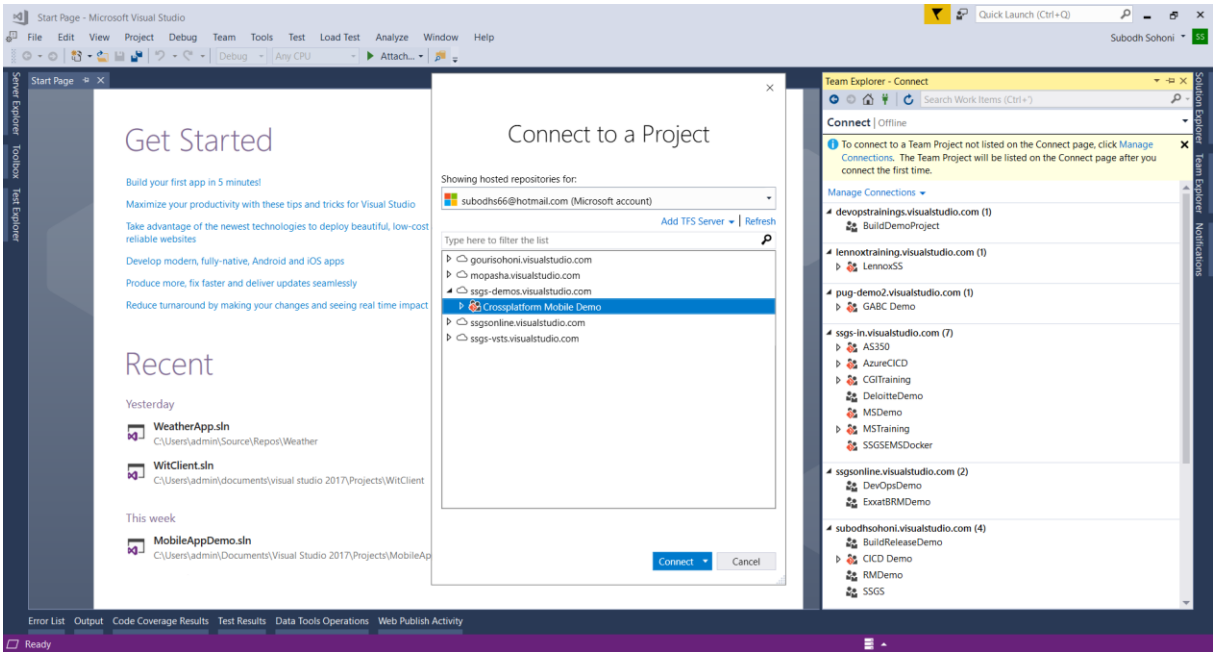
You can run the project now locally in an emulator by F5 or hit the debug button.

Exercise 2: Create a new Team Project and clone its repository

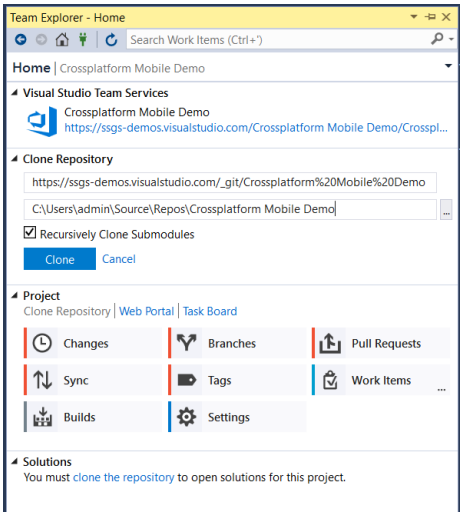
1. Create a new Team Project in your VSTS Account named Crossplatform Mobile Demo.



2. From Team Explorer in Visual Studio Connect to that team project.



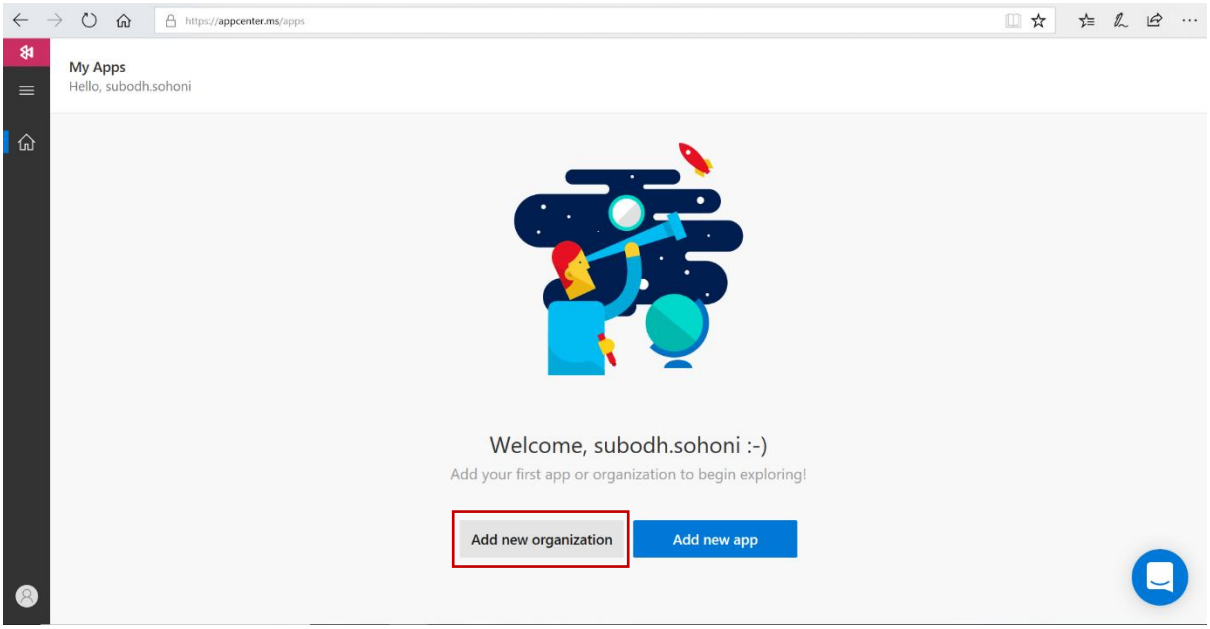
3. Click the button Clone to clone the repository and map it to a convenient location.



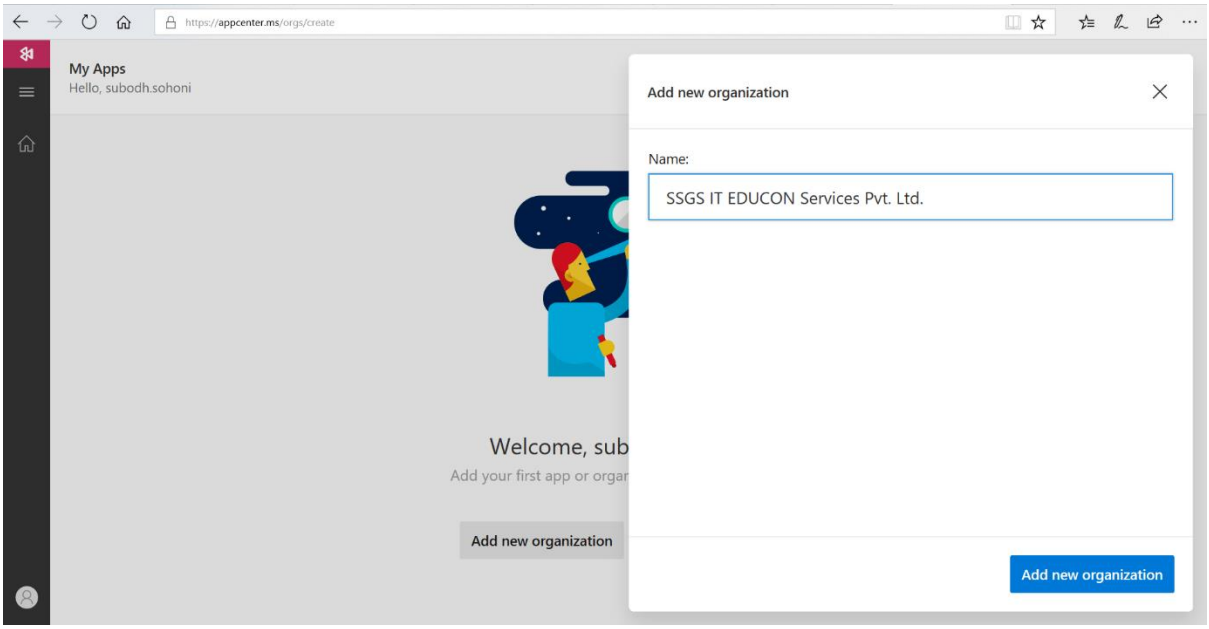
4. Cut and paste the entire solution that was created in earlier exercise in the folder mapped to the repository.
5. Commit and Push the code.

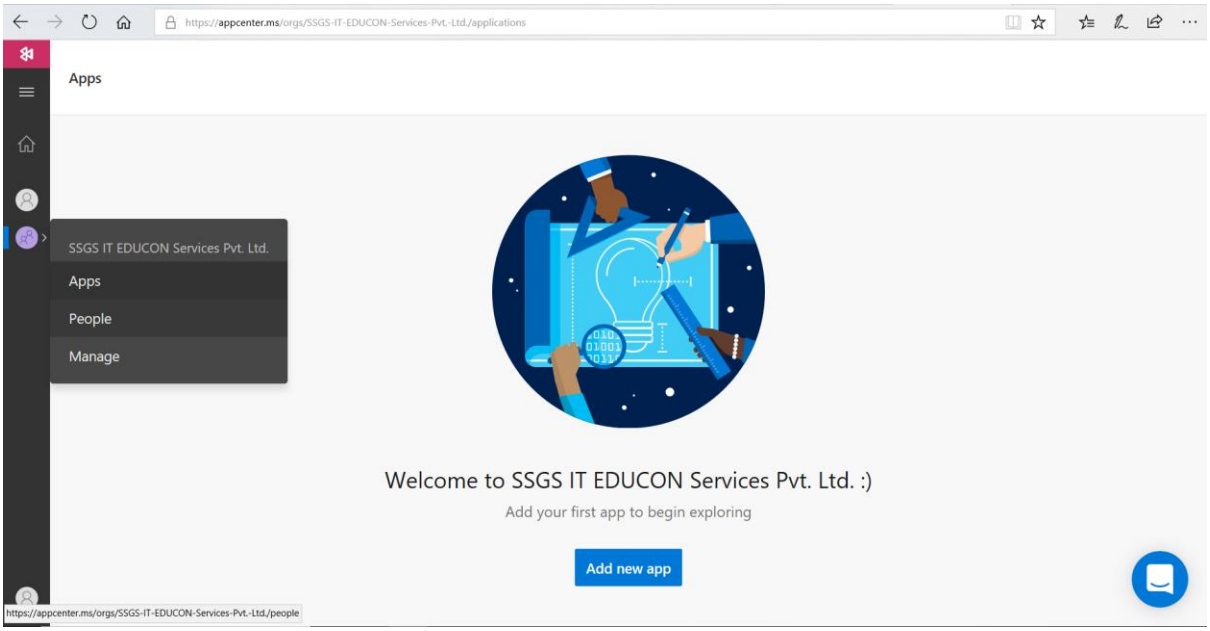
Exercise 3: Work with App Center

1. Open App Center (https://appcenter.ms) and login with your account.
2. Click button to Add new organization.

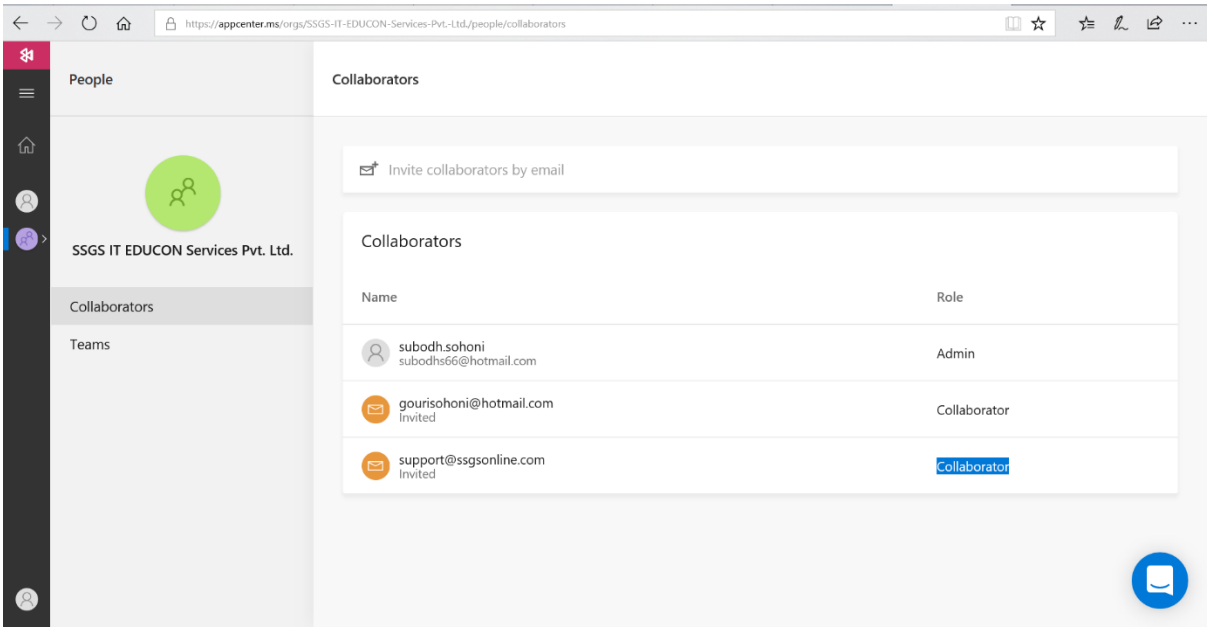


3. Provide the name of your organization.

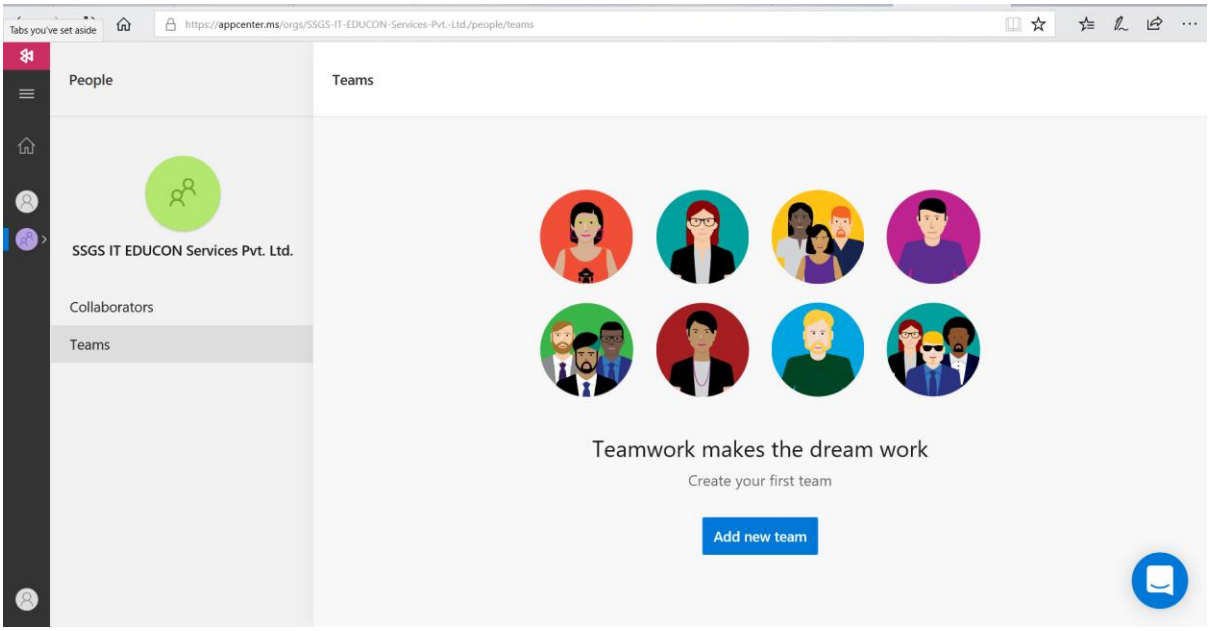


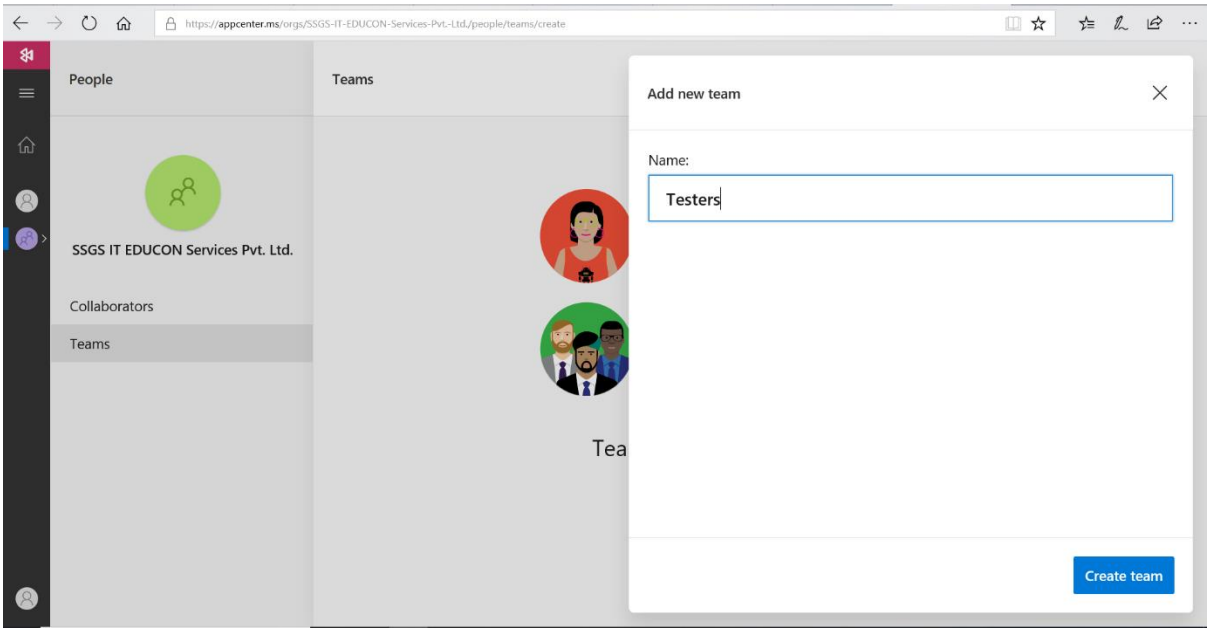


4. In your organization add emails of your collaborators.

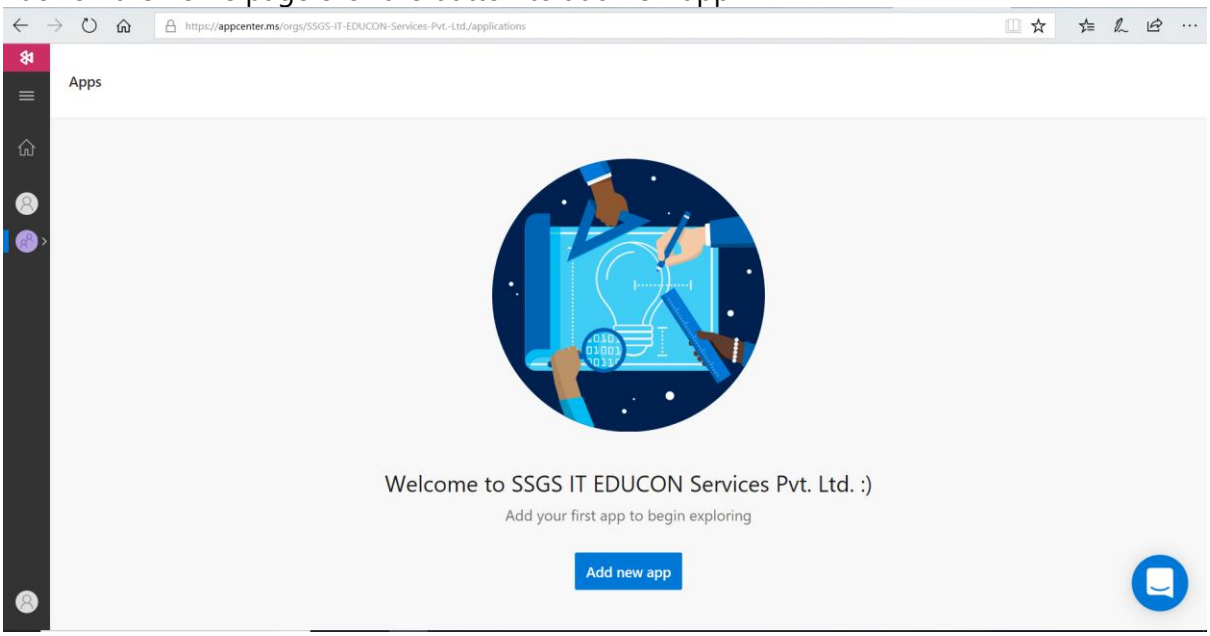


5. Create a team of Testers

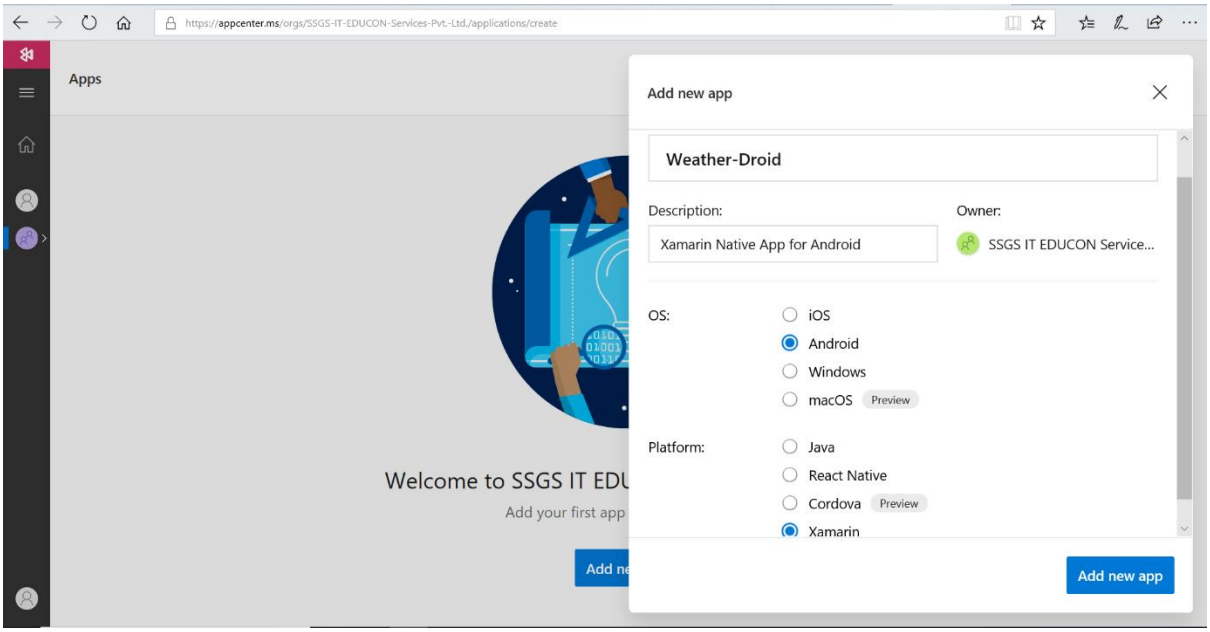




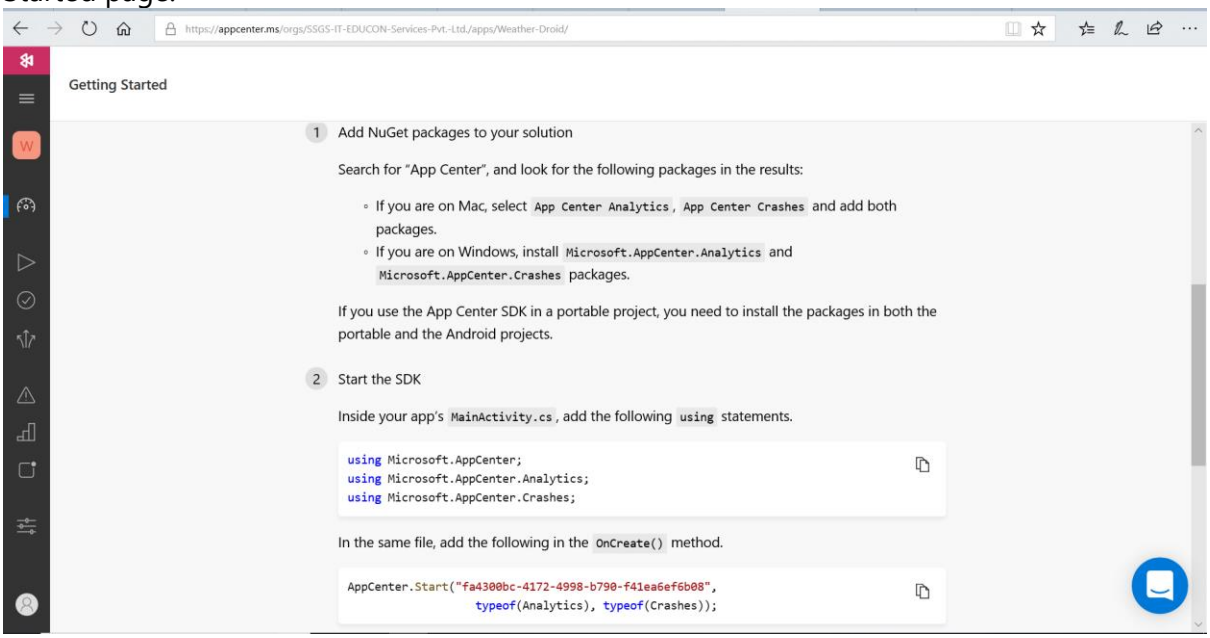
6. Back on the home page click the button to add new app



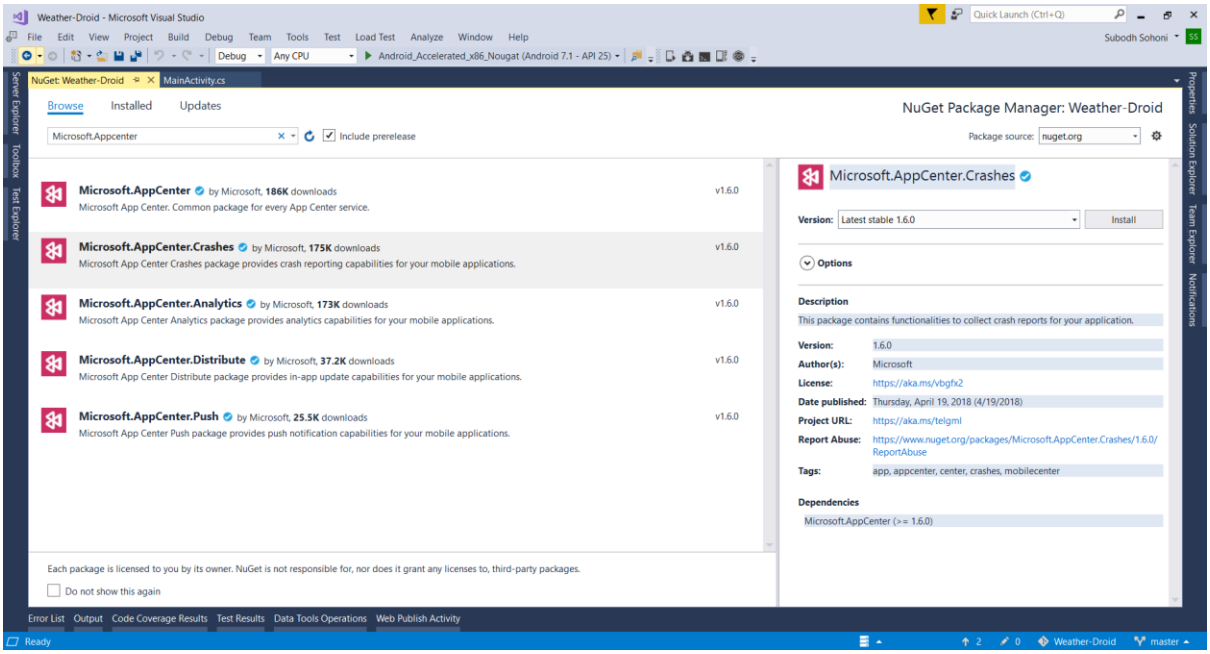
7. Give a name to the app as Weather-Droid and select the radio buttons of Android in OS group and Xamarin in the platform group.

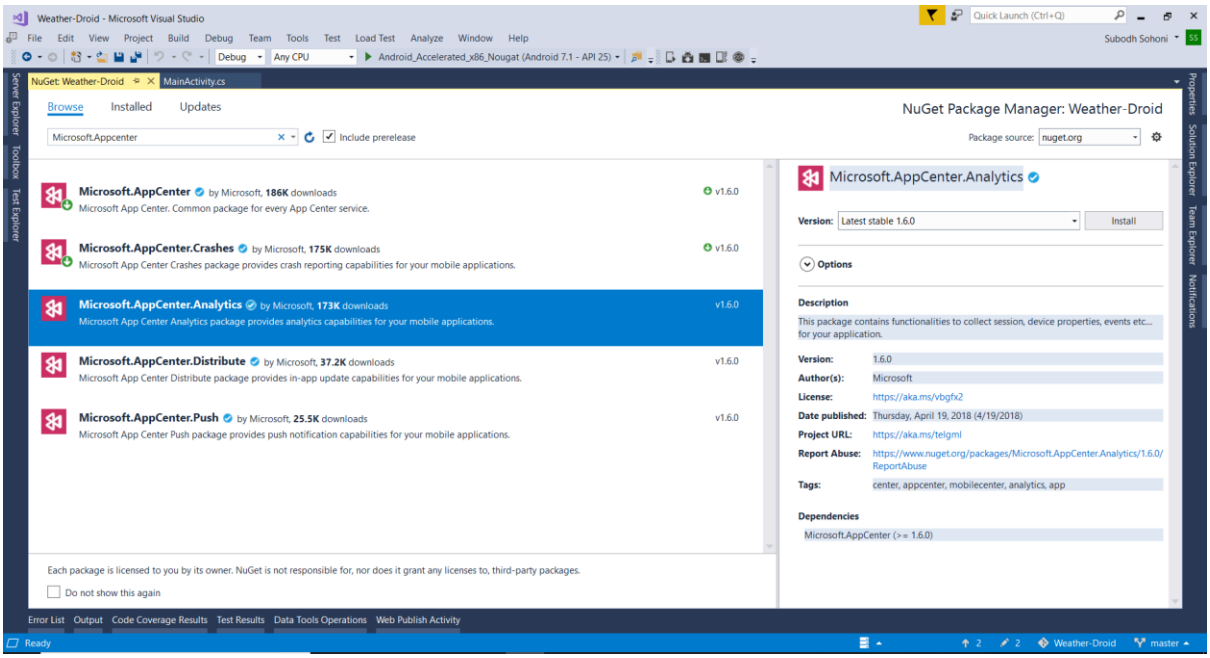


8. Once the app creation is complete, you will find further instructions in the Getting Started page.

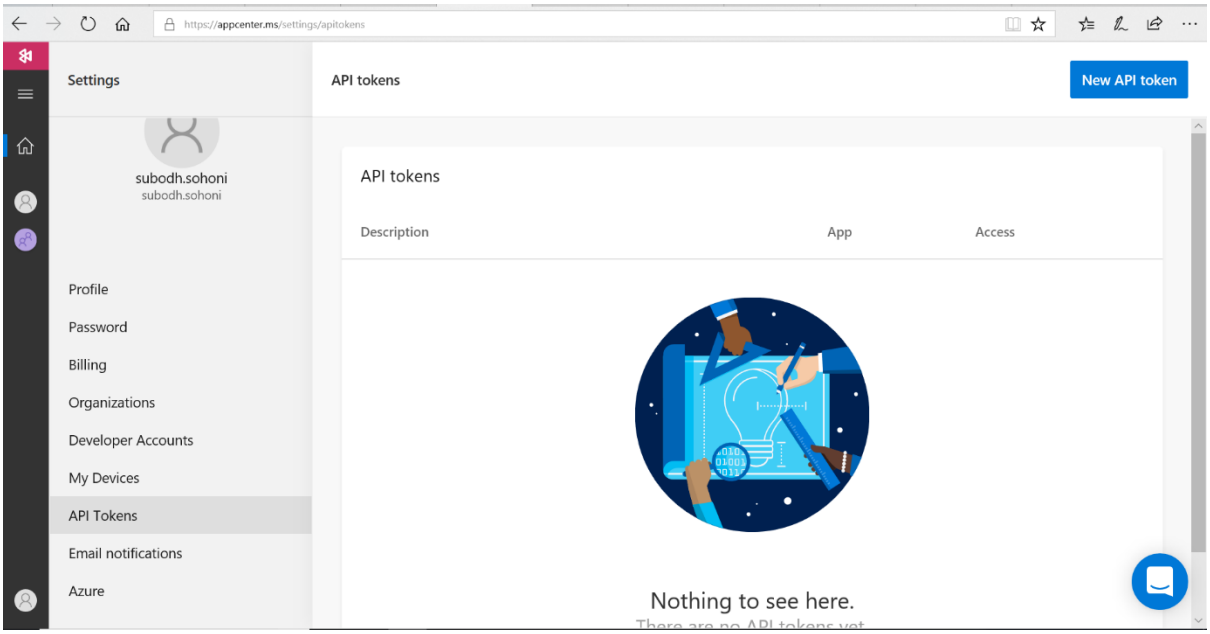


9. Open your project in Visual Studio and add the NuGet packages for Microsoft.AppCenter.Analytics and Microsoft.AppCenter.Crashes to your project.

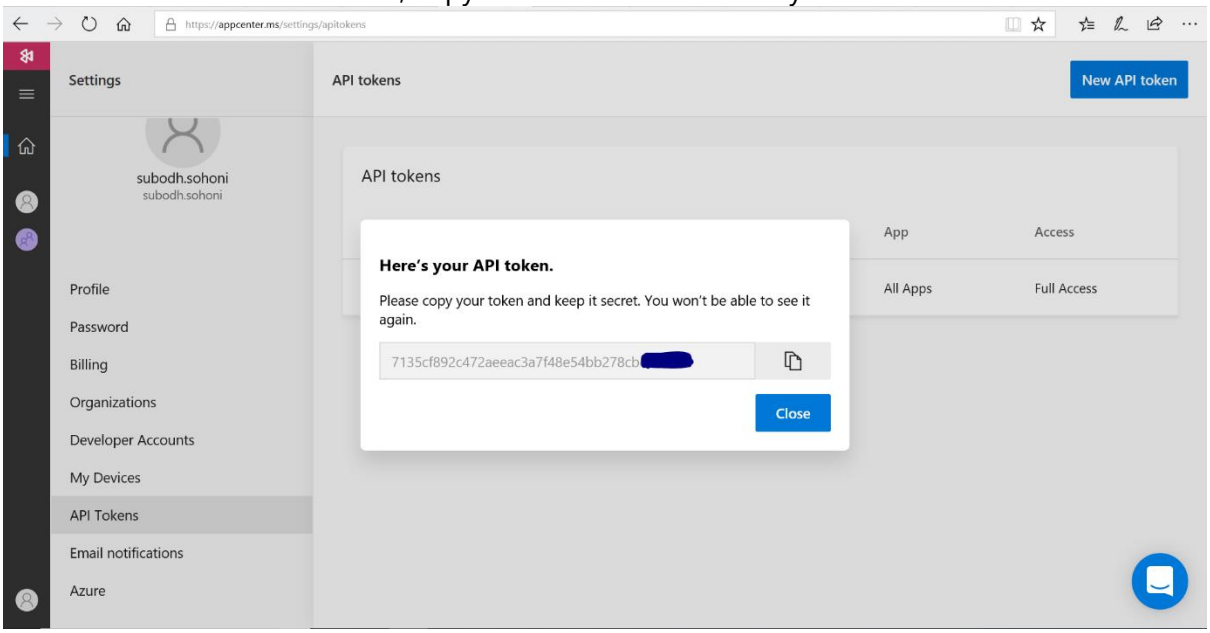




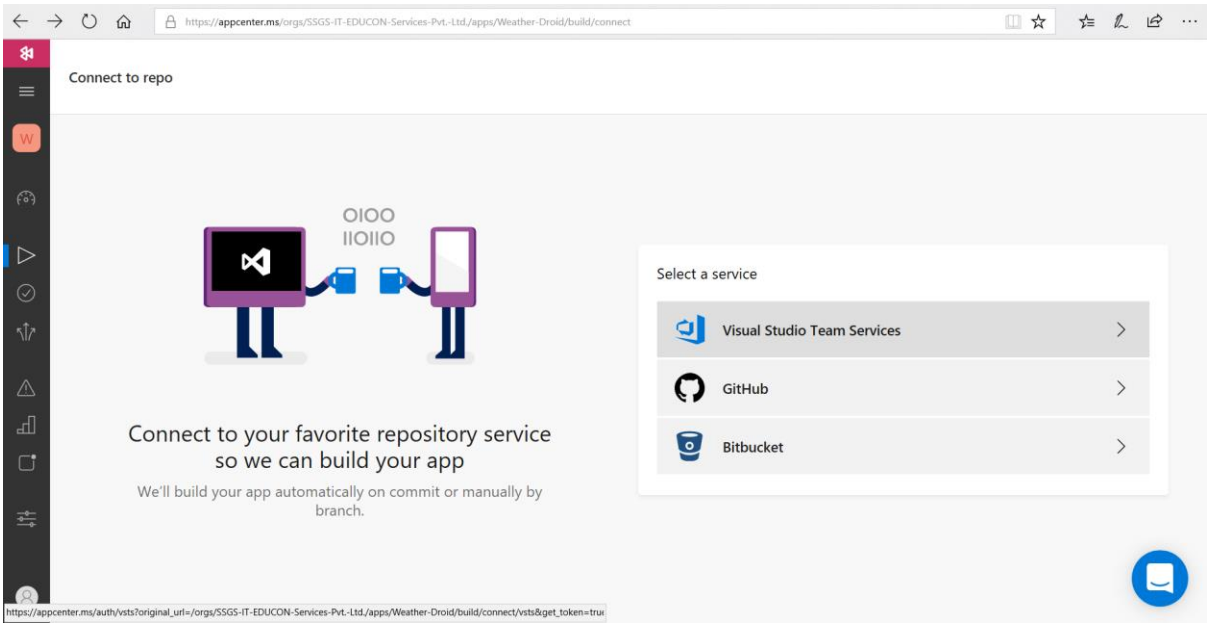
10. Add the using statements and the line of code to provide the authentication key to the file MainActivity.cs
11. Open your profile under Settings and select the option of API Tokens. We will require the API Token if we want to do deployment from VSTS.



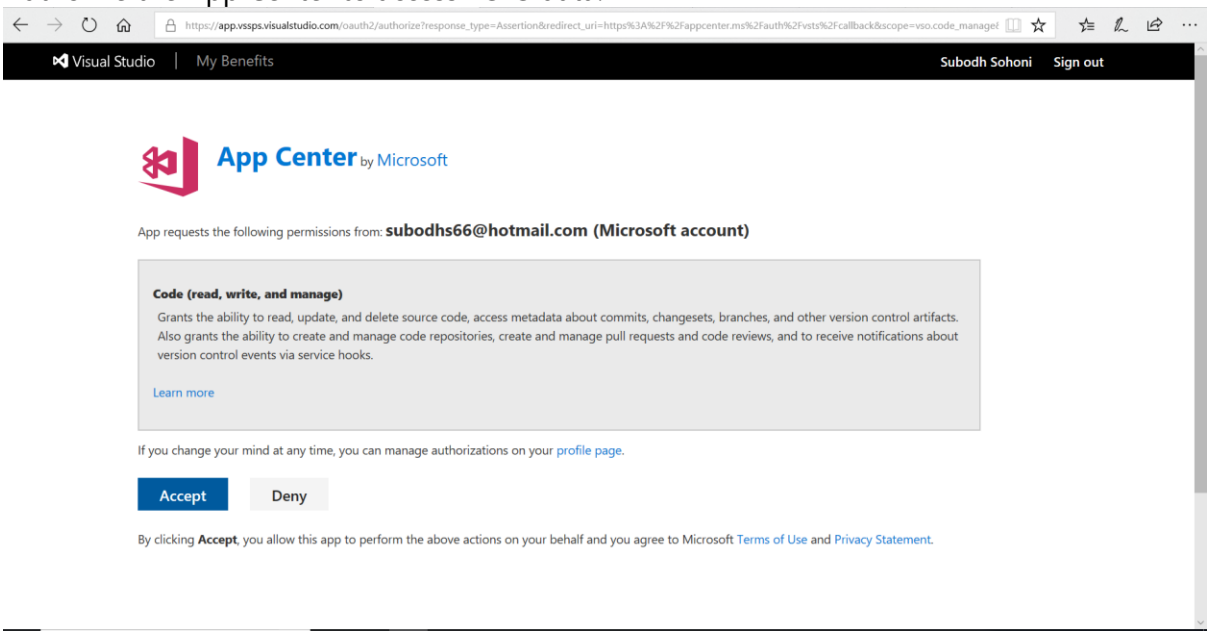
12. Click the button to create New API Token. Provide a description and select the radio button of Full Access.
13. When the API Token is created, copy that and store it securely for later use.



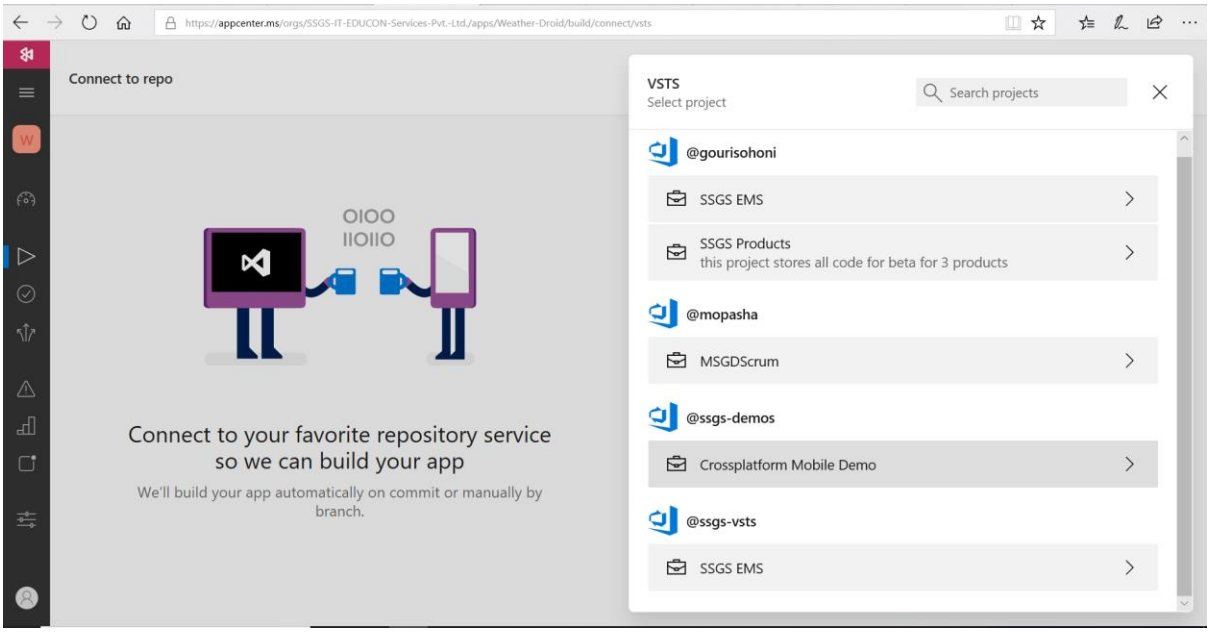
16. Now we can build our app on App Center. To do that under the Build activity select the version control that we are using – Visual Studio Team Services.

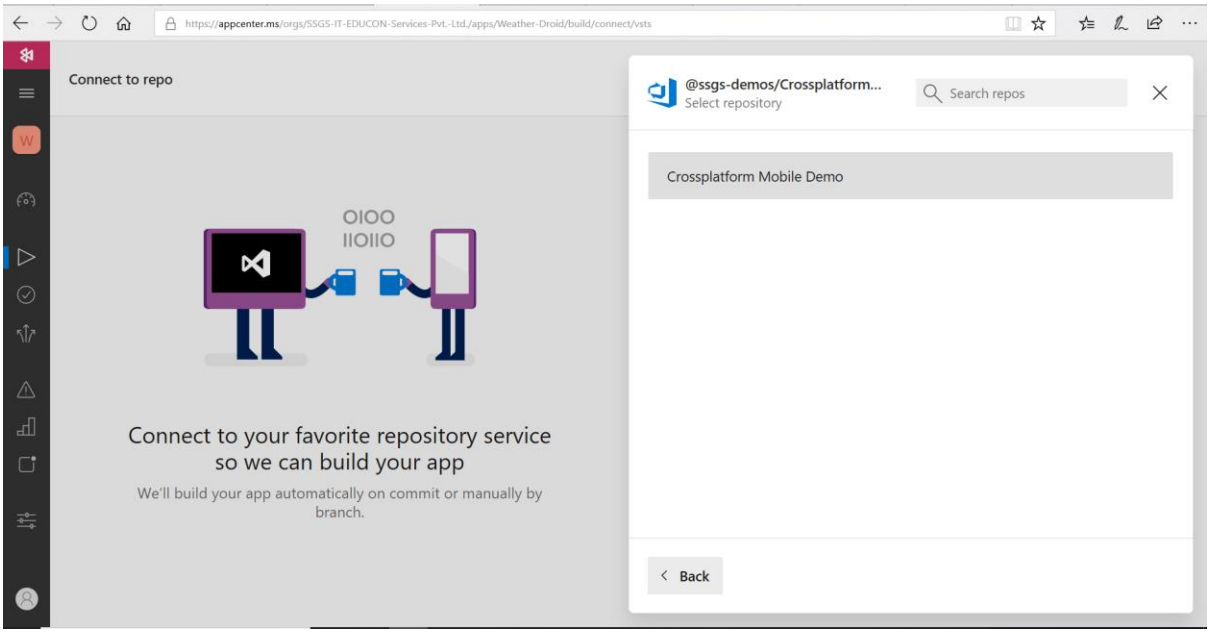


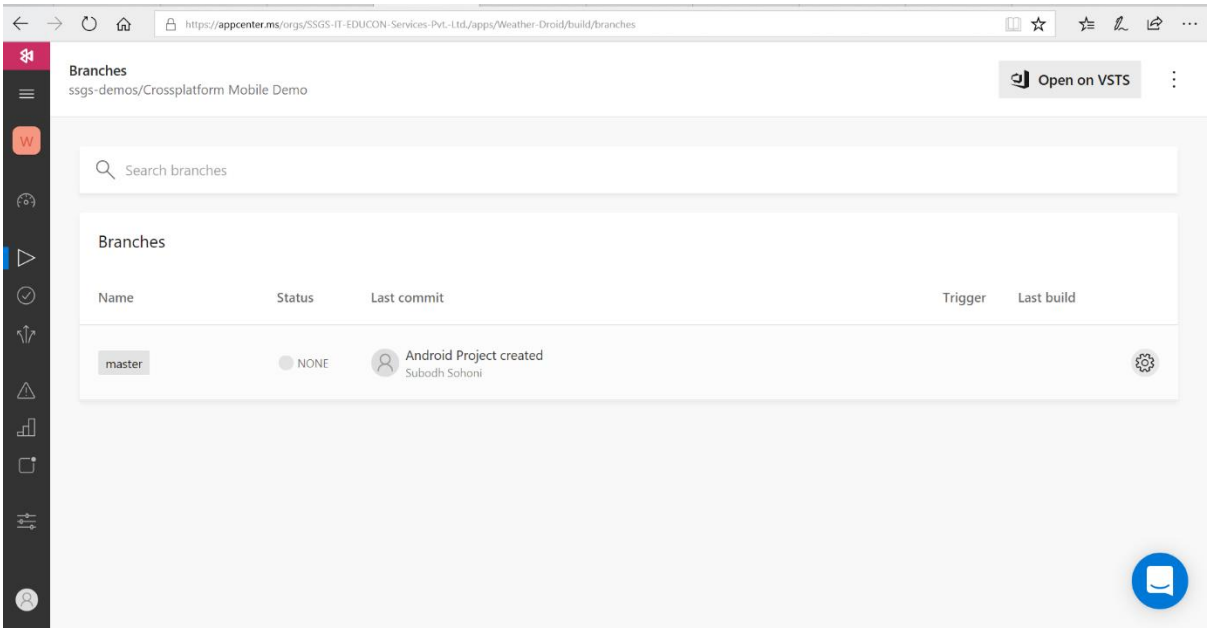
17. Authorize the App Center to access VSTS data.



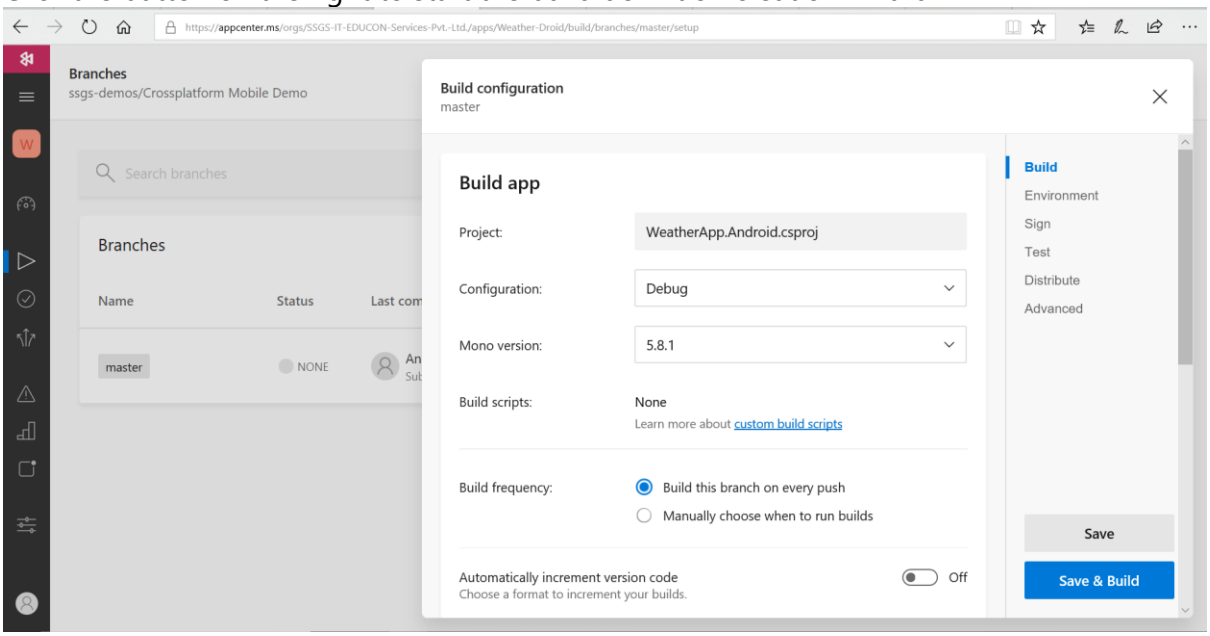
18. Select the VSTS Account, team project and then the git repository in which we have pushed our code.

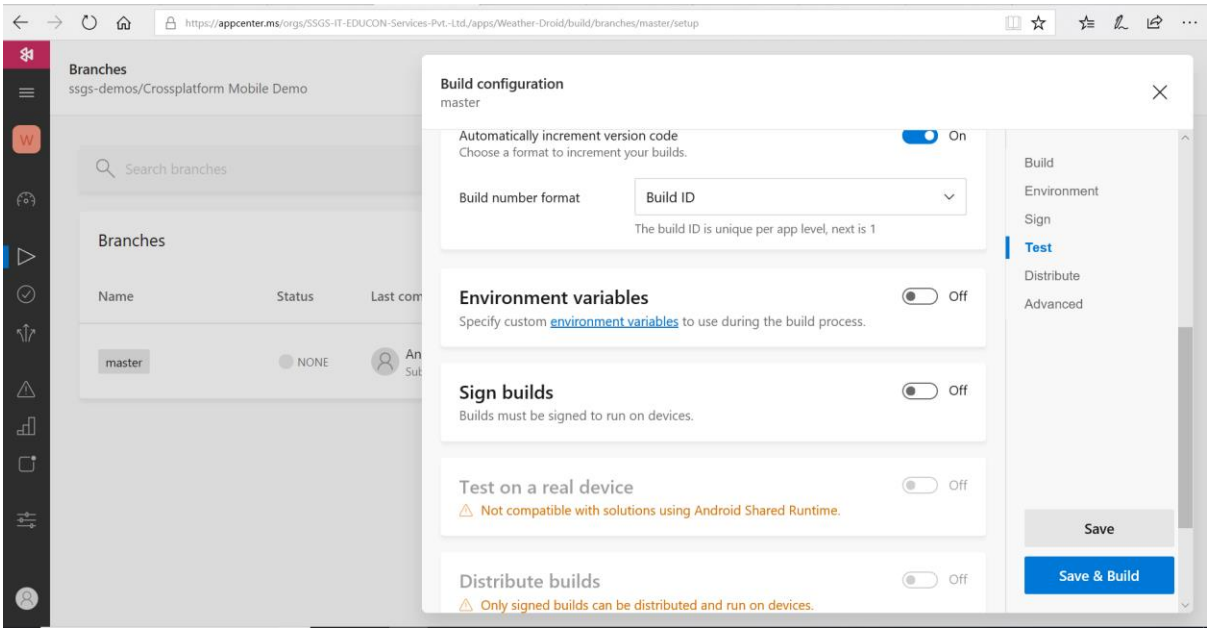

https://appcenter.ms/orgs/SSGS-IT-EDUCON-Services-Pvt.-Ltd./apps/Weather-Droid/build/connect/vsts
Connect to repo
Connect to your favorite repository service so we can build your app
We'll build your app automatically on commit or manually by branch.
VSTS
Select project
Search projects
@gourisohoni
SSGS EMS
SSGS Products
this project stores all code for beta for 3 products
@mopasha
MSGDScrum
@ssgs-demos
Crossplatform Mobile Demo
@ssgs-vsts
SSGS EMS


https://appcenter.ms/orgs/SSGS-IT-EDUCON-Services-Pvt.-Ltd./apps/Weather-Droid/build/connect/vsts
Connect to repo
Connect to your favorite repository service so we can build your app
We'll build your app automatically on commit or manually by branch.
@ssgs-demos/Crossplatform...
Select repository
Search repos
Crossplatform Mobile Demo
Back

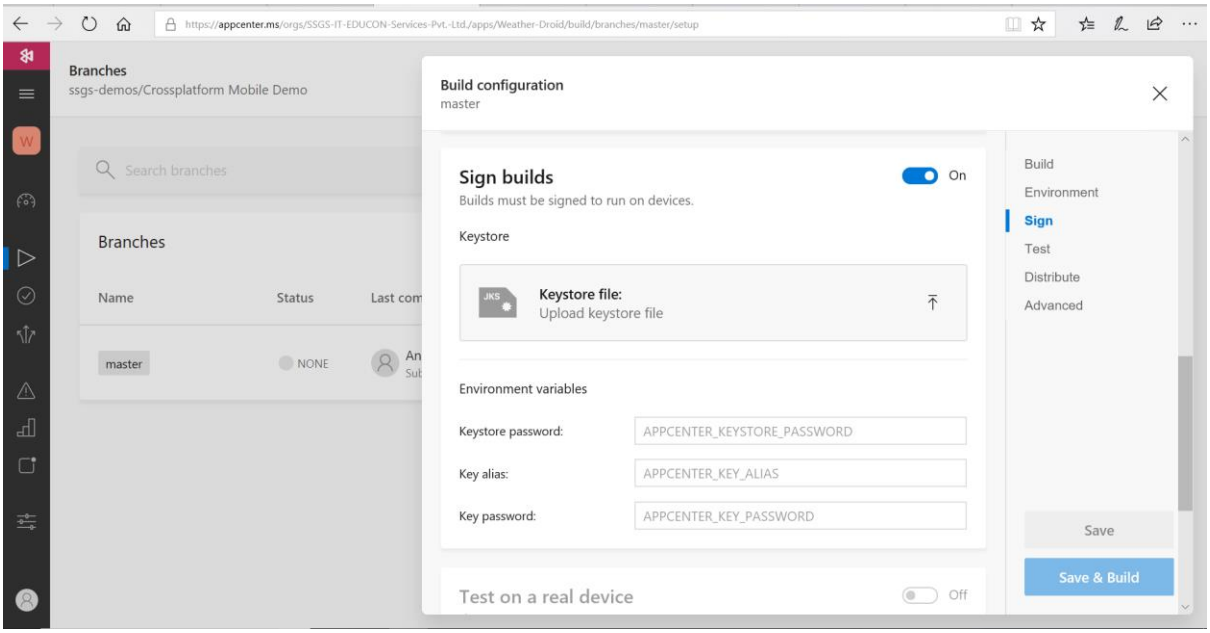


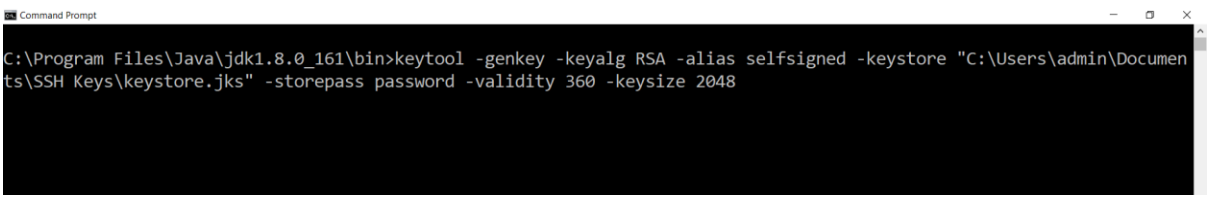
19. Click the button on the right to start the build definition creation wizard



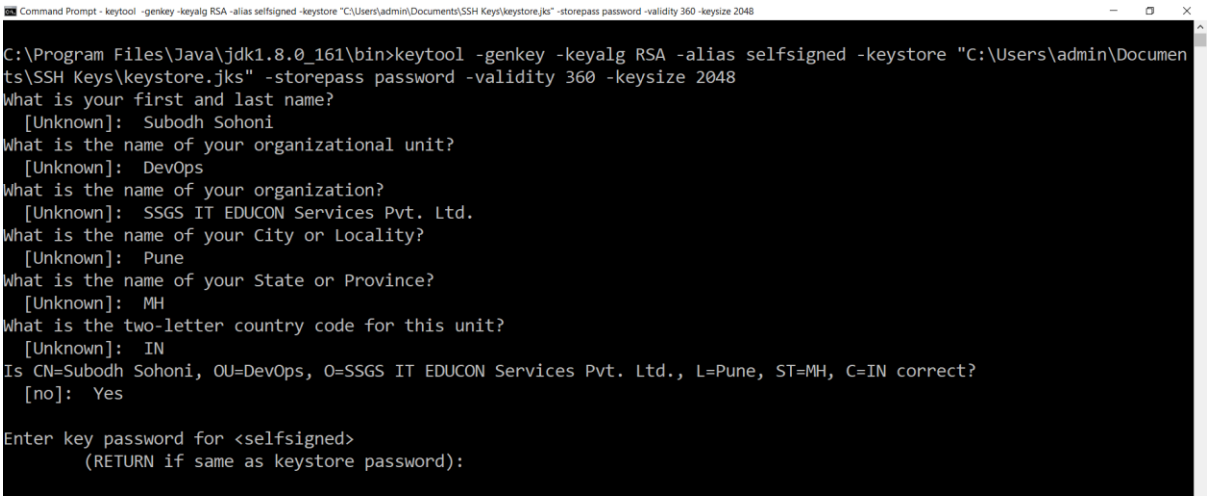


20. To deploy the app on a device or to Google Play Store, we need to sign the .apk that is produced by the build. App Center Build can sign the .apk if we provide the Key Store to the build. We can create Key Store by using keytool that is included in the JDK.

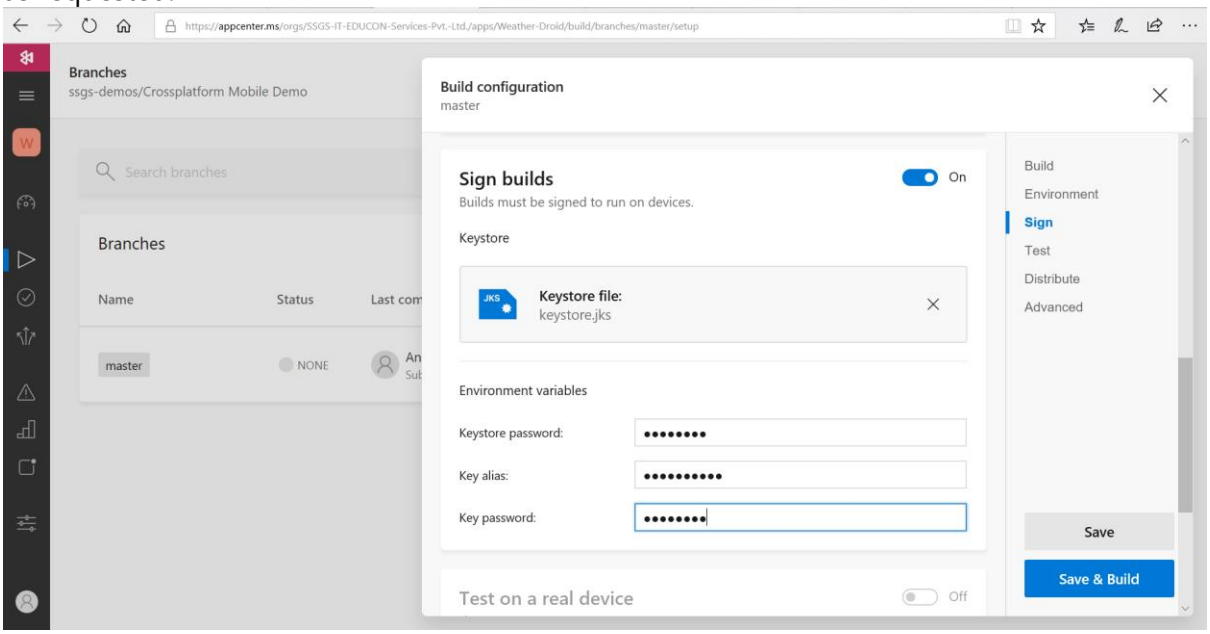




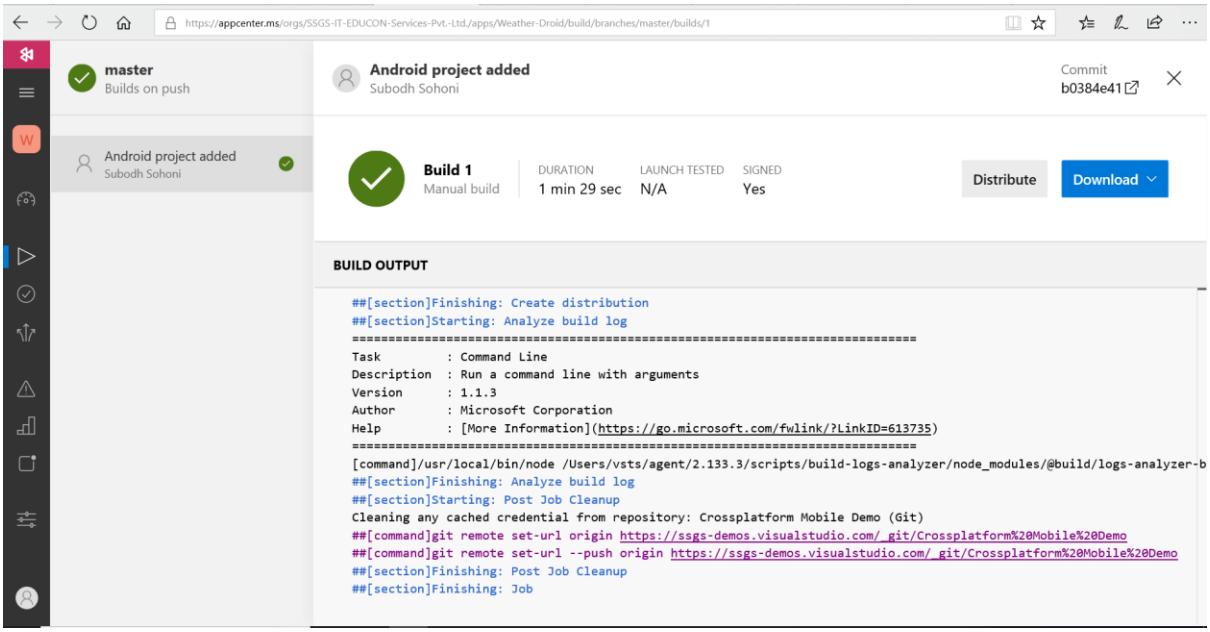
Execute the keytool -genkey command as shown from the Admin command prompt. Provide required data and the password for key. Note the alias and passwords.



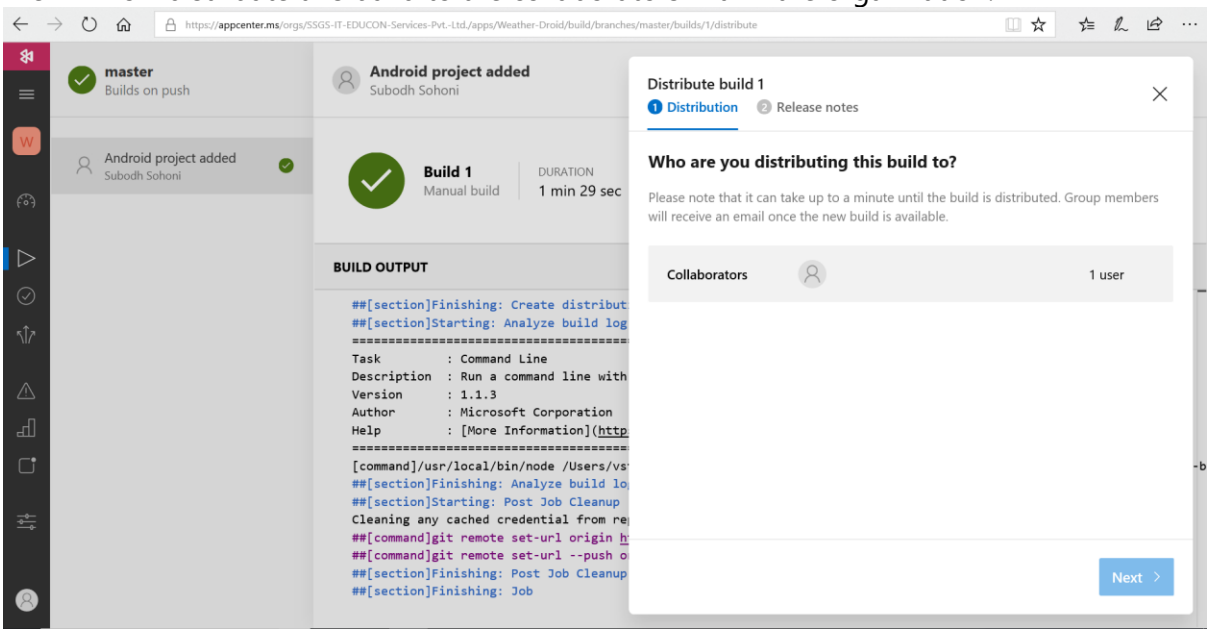
21. Upload the created Key Store and enter the passwords of Key Store, key and the alias as requested.



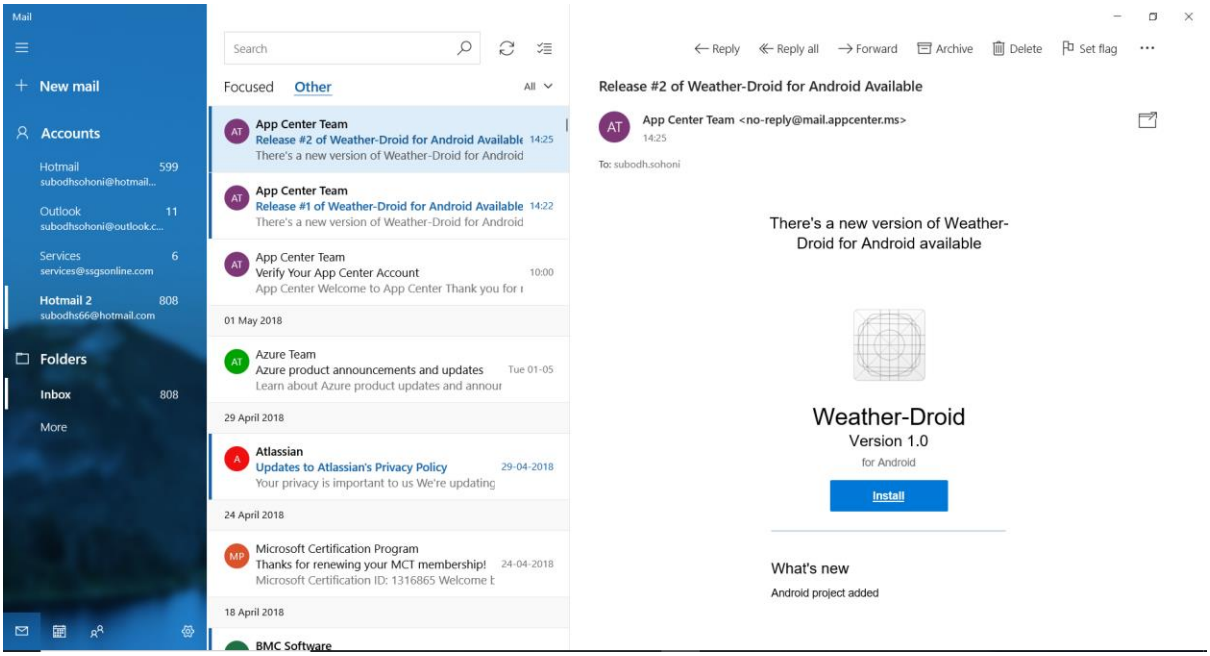
Now the built .apk will be signed. Save and Build.



22. We will now distribute this build to the collaborators within the organization.

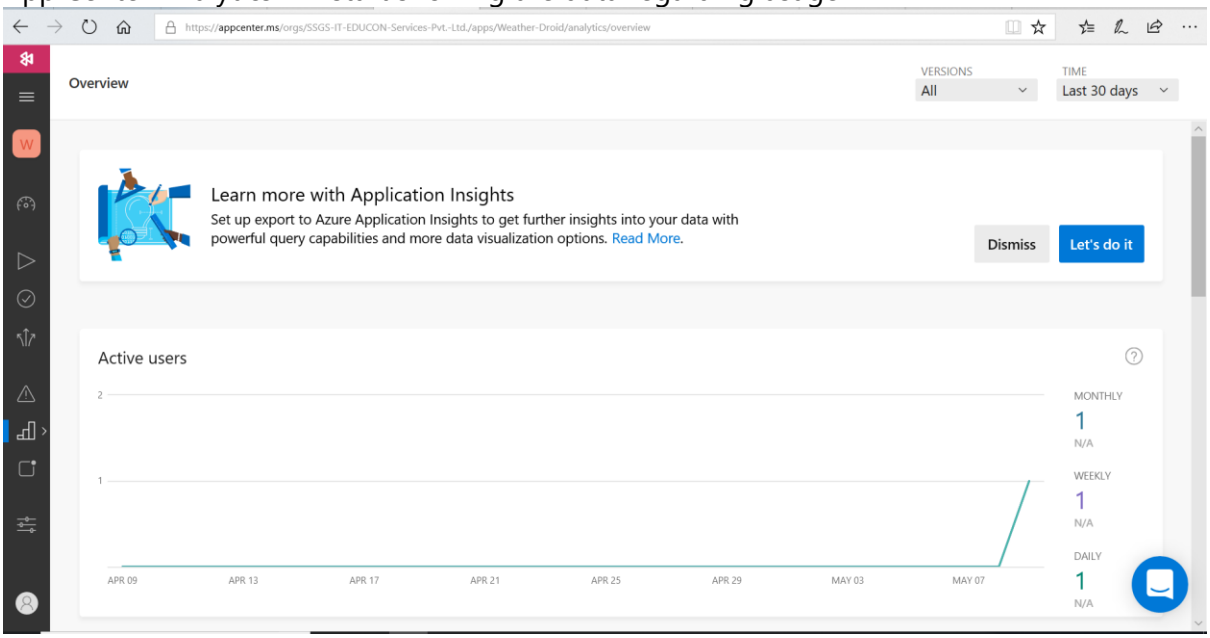


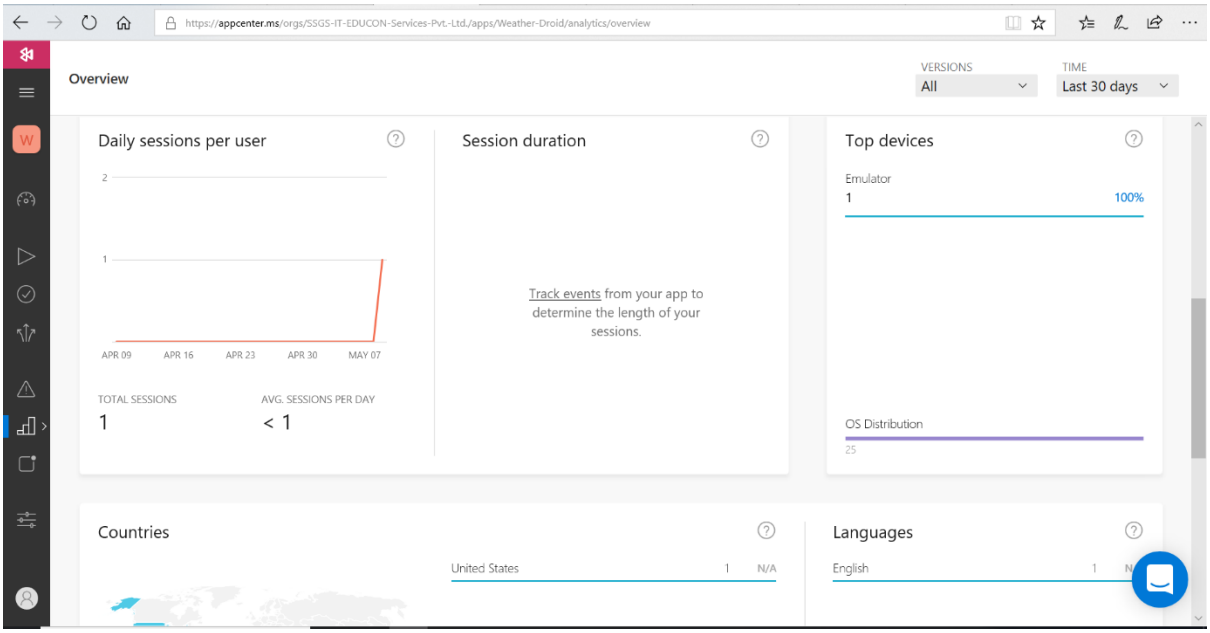
23. Once the distribution is done, each collaborator will receive and email

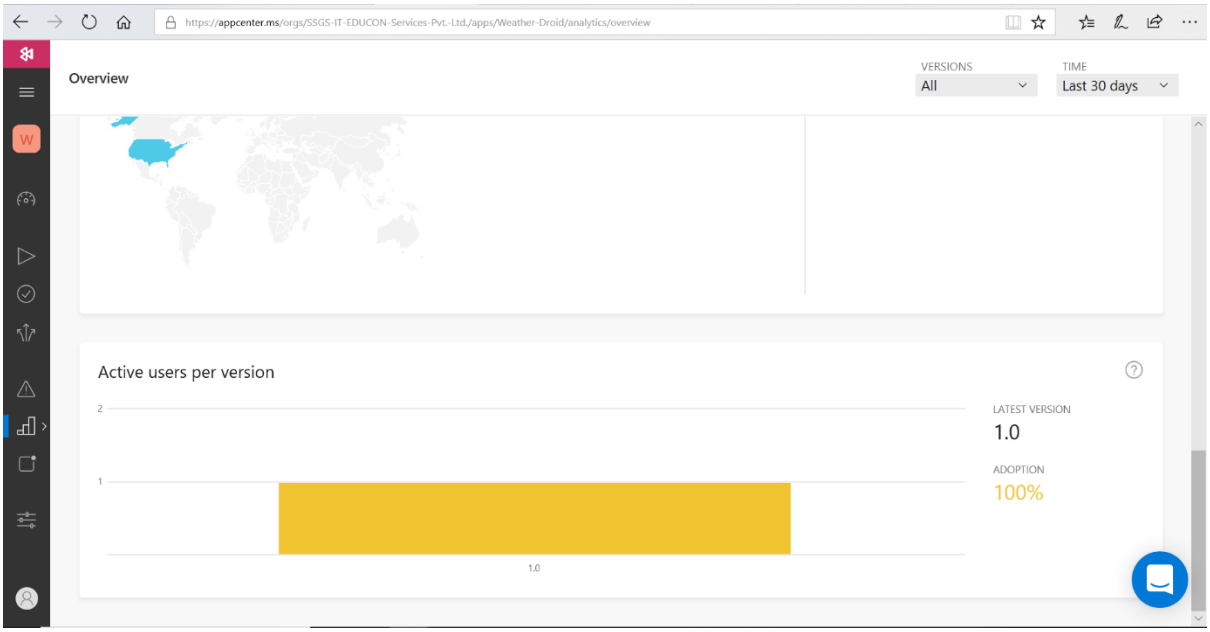


They can view it on the device and click the Install button provided in the email to install it.

24. Once the installation is over, we can execute the application on the device and the App Center Analytics will start showing the data regarding usage

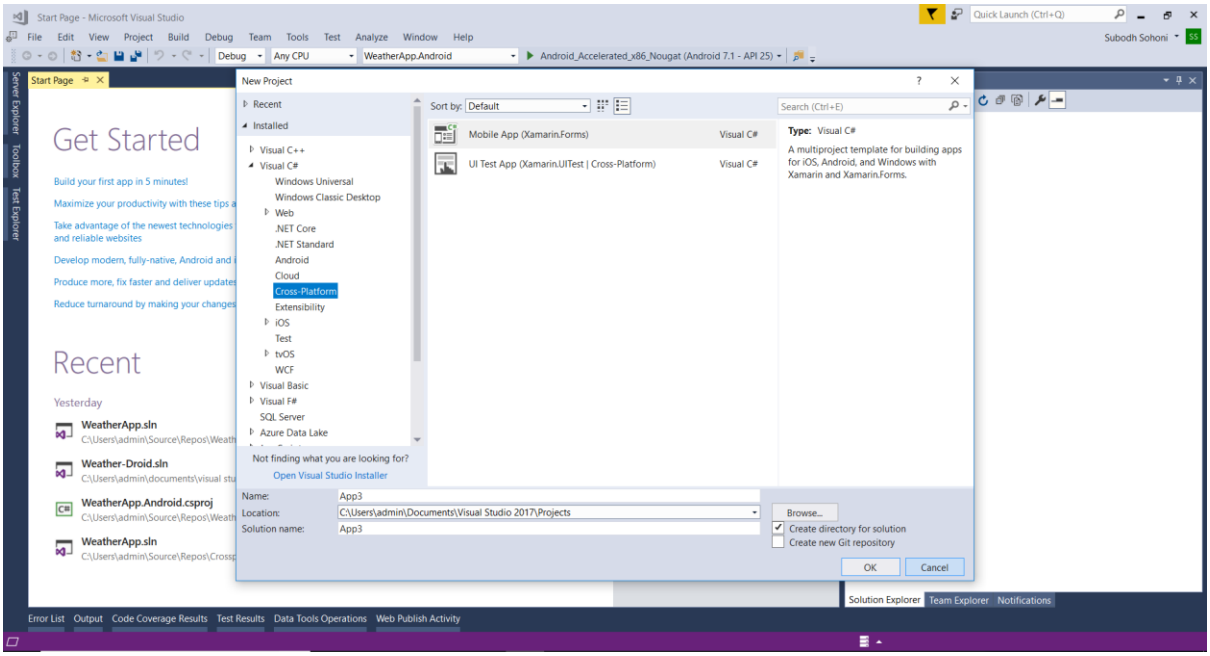




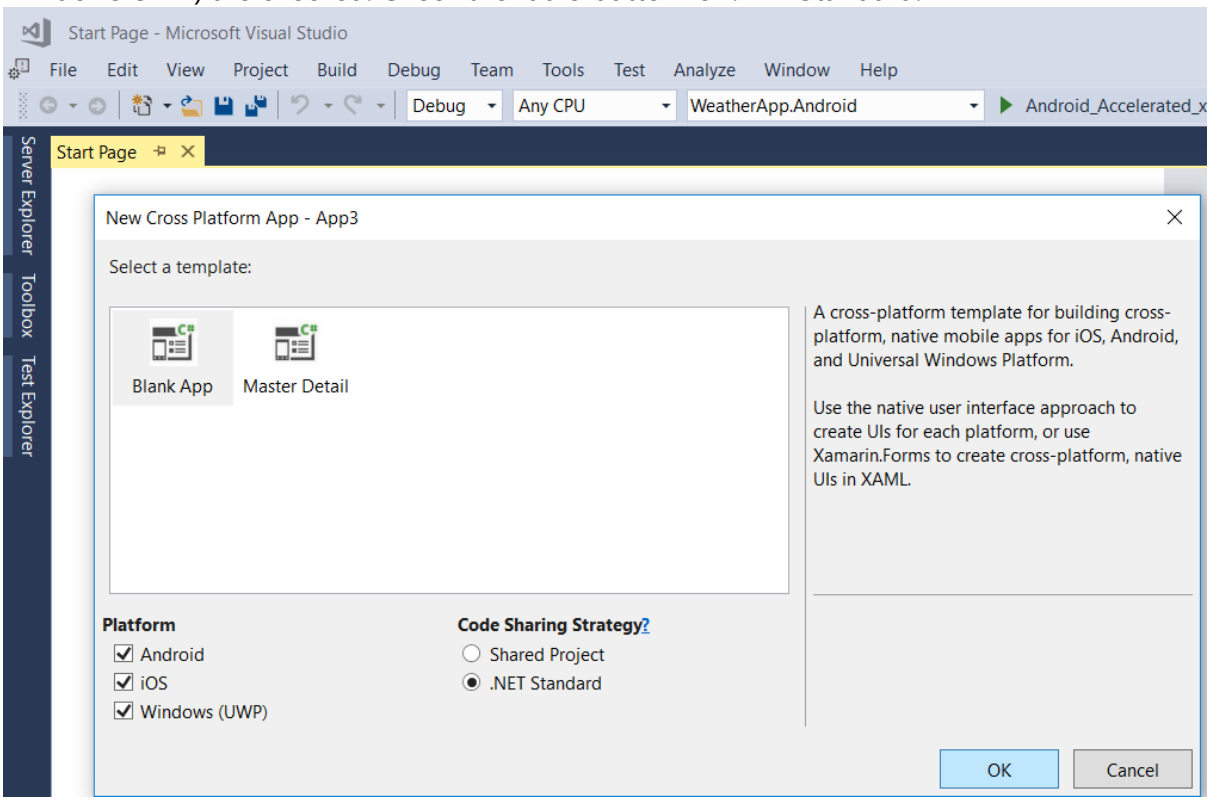


Exercise 3: Create, Build and Distribute Xamarin Forms App

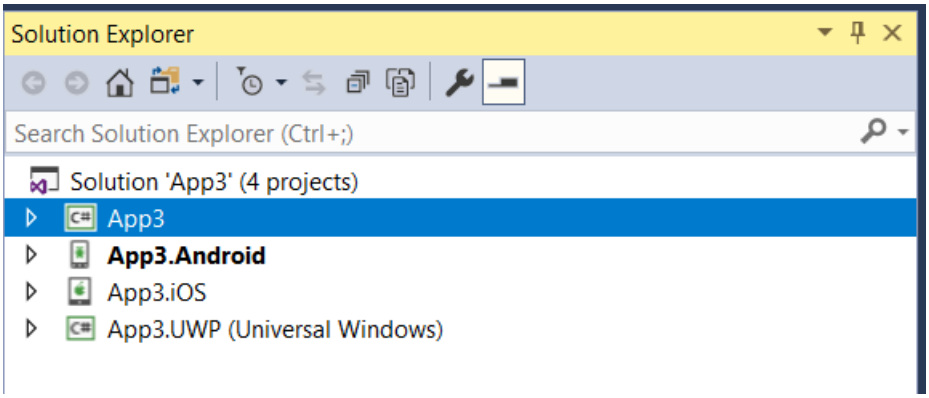
1. In Visual Studio 2017 create a new project. Select the template of Cross Platform – Mobile App (Xamarin.Forms). Do not add this project to any git repository.



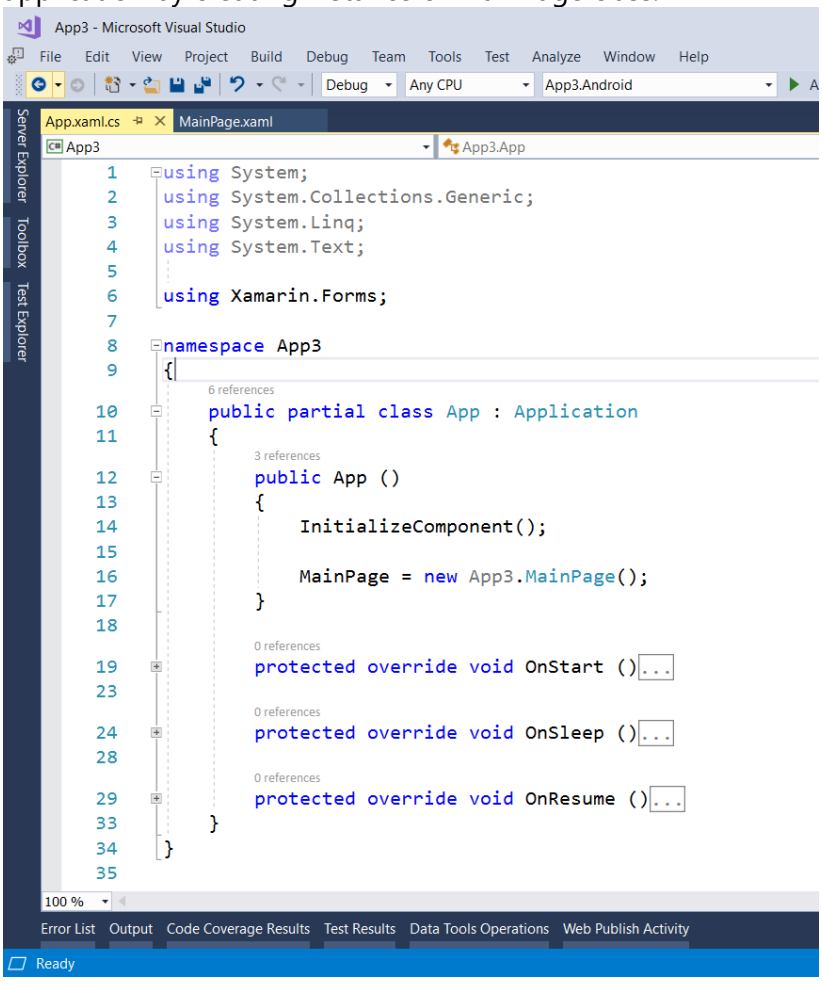
2. From the Platform options ensure that all check-boxes (for Android, iOS and Windows UWP) are checked. Check the radio button for .NET Standard.



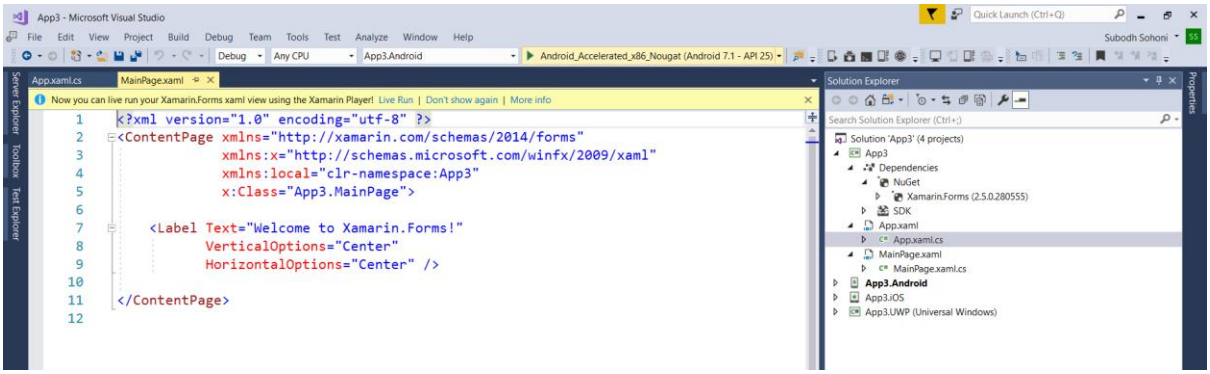
3. When the solution is created, you will find that 4 projects are created. First one is a class library type of project that contains the code that is shared by all the other projects. It will compile into a PCL (Portable Class Library) that is referenced by all other projects. Other projects are specific to OS – Android, iOS and Windows (UWP).



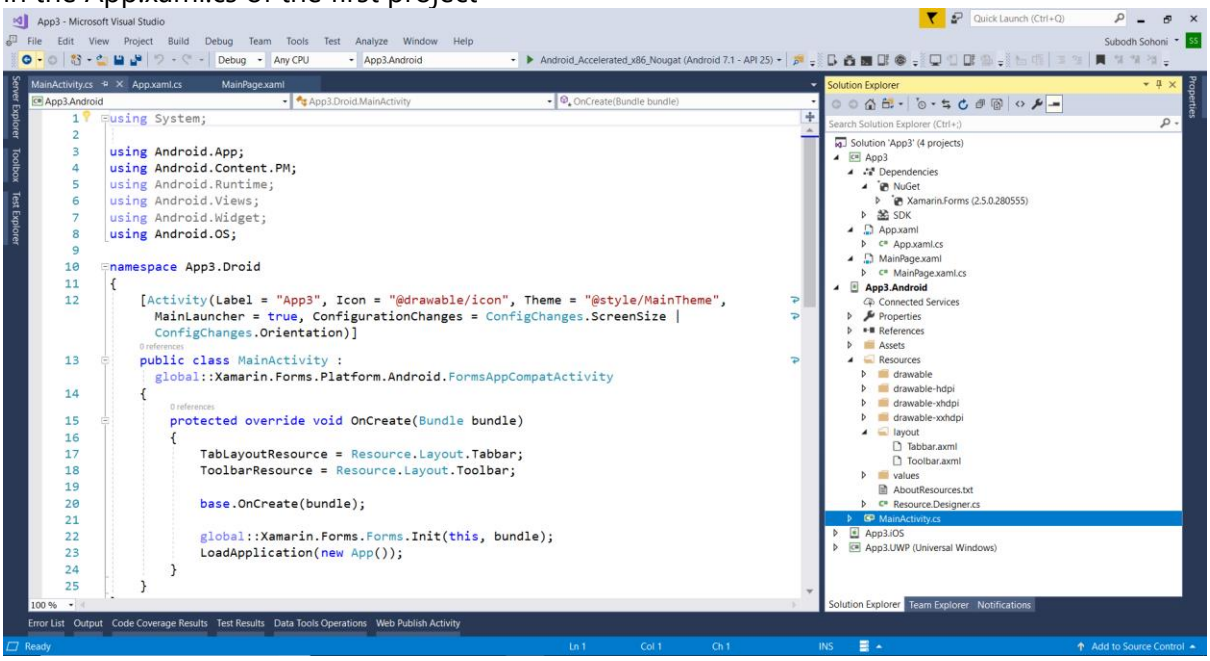
4. The PCL project has a NuGet packaged added for Xamarin.Forms. That project contains two most significant files: First one is App.xaml.cs which initializes the application by creating instance of MainPage class.



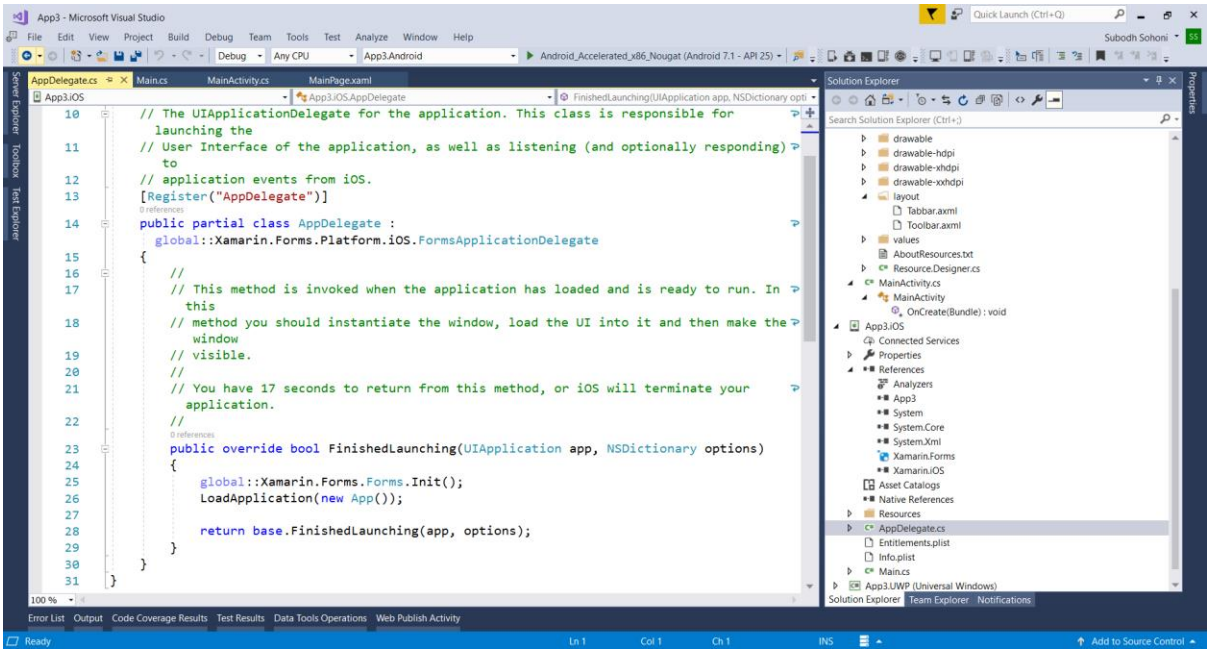
The second important file is MainPage.xaml. This is the file which contains all the common layout which will be converted to OS specific UI Elements at compile time.



5. The Android specific project contains a file named MainActivity.cs which has a OnCreate method that initializes minor layouts and then Loads the App class that is in the App.xaml.cs of the first project



6. Similarly iOS specific project has a file named AppDelegate.cs that loads the App class.



7. Windows (UWP) specific project has Main.xaml.cs file that loads the App class.

8. After going through the details / anatomy of the projects created by the project templates, let us make some changes in it.

9. Change the code of MainPage.xaml as follows:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:local="clr-namespace:App3"
             x:Class="App3.MainPage">
    <StackLayout VerticalOptions="Center"
                 HorizontalOptions="Center">
        <Label Text="Welcome to Xamarin.Forms!" />
        <Button x:Name="Btn" AutomationId="Btn" Text="Click Me"
                Clicked="BtnClick"/>
        <Label x:Name="LblResult" AutomationId="LblResult" Text="" />
    </StackLayout>
</ContentPage>
```

10. In the MainPage.xaml.cs add the event handler code as follows:

```
void BtnClick(object sender, EventArgs args)
{
    LblResult.Text = "Button Clicked";
}
```

11. Run the app in the emulator.

12. In the App Center create 3 apps, 1 for each OS. Get the App Secrets from each app and store them securely.
13. From http://openweathermap.org download the Weather app for Xamarin.Forms. Unzip that into the same folder that is mapped to your git repository. Open the solution in the Visual Studio.
14. Go through the important files as were mentioned in the above steps for projects under the opened solution.
15. In the OnStart method of App.xaml.cs file add the line of code:

    AppCenter.Start("ios={Your App Secret};android={Your App Secret};uwp={Your App Secret}", typeof(Analytics), typeof(Crashes));

16. Replace the {Your App Secret} with the respective app secret.
17. Commit and Push the code to the repository.
18. Create new build for each app and execute those builds to create apps.
19. Distribute builds to your teams.