12

Case Studies

In this chapter, we’ll going through the entire process of mobile DevOps from Mobile application development and integration to continuous testing and deployment.  
We’ll be using 2 applications as case studies to show the entire process.

1. Basic “Hello World” GUI
2. Button Widget

**Case Study 1: “Hello World” GUI**

In this case study, we’ll be covering the Mobile DevOps cycle with a simple Android application that would have a MainActivity with a “Hello World” Text Label on it.

This study is going to cover the entire process in brief and show you a step by step workflow of the same.

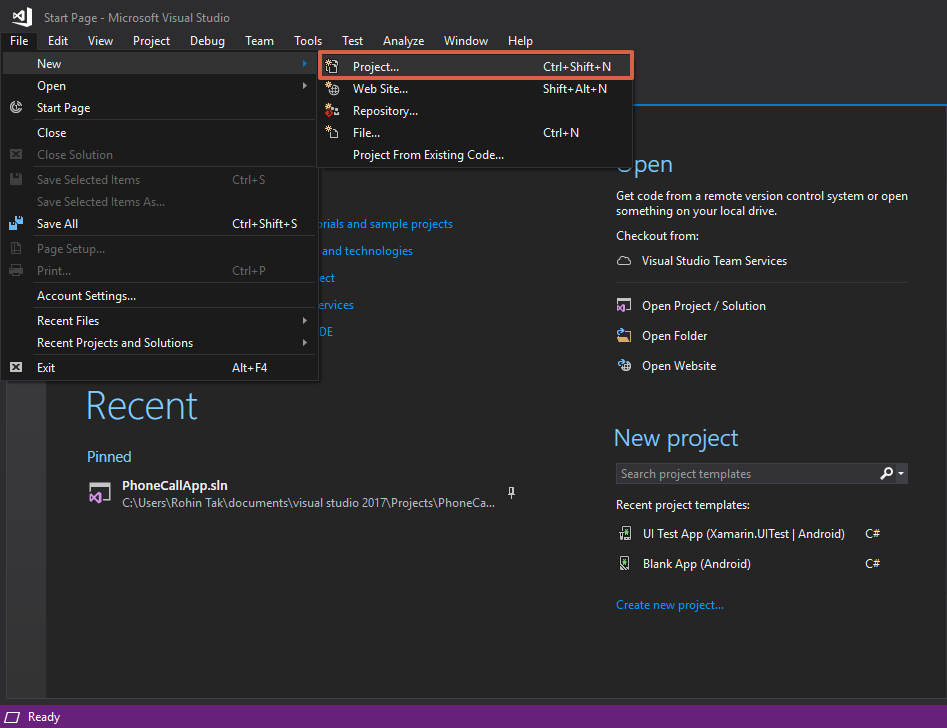
**Prerequisites**

Since these case studies will be covering all the steps involved in the life cycle, it would not be possible to cover these topics in detail and explain different parts of IDE and Android development fundamentals.   
Following are minimum prerequisites to follow this chapter smoothly. If you need to get more understanding of any of the below topics, please refer previous chapters for the same.

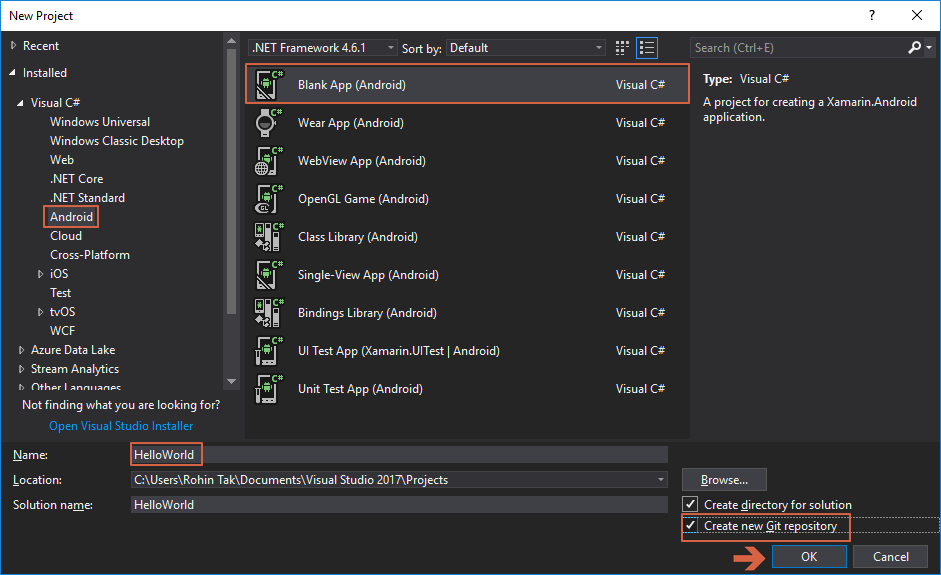
* It is assumed that you have Visual Studio and Xamarin installed on your computer and is configured and ready for Android application development.   
  If you do not have Visual Studio and Xamarin installed on your system, please refer to chapter 3 and install them first.
* Basic understanding of Visual Studio
* Basic understanding of Android development fundamentals
* You should have a working Git account are able to access the same.

Let’s get started with the following steps to get a complete practical workflow of mobile app development.

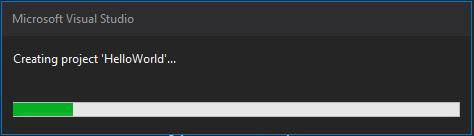
1. Open Visual Studio and go to File -> New -> Project



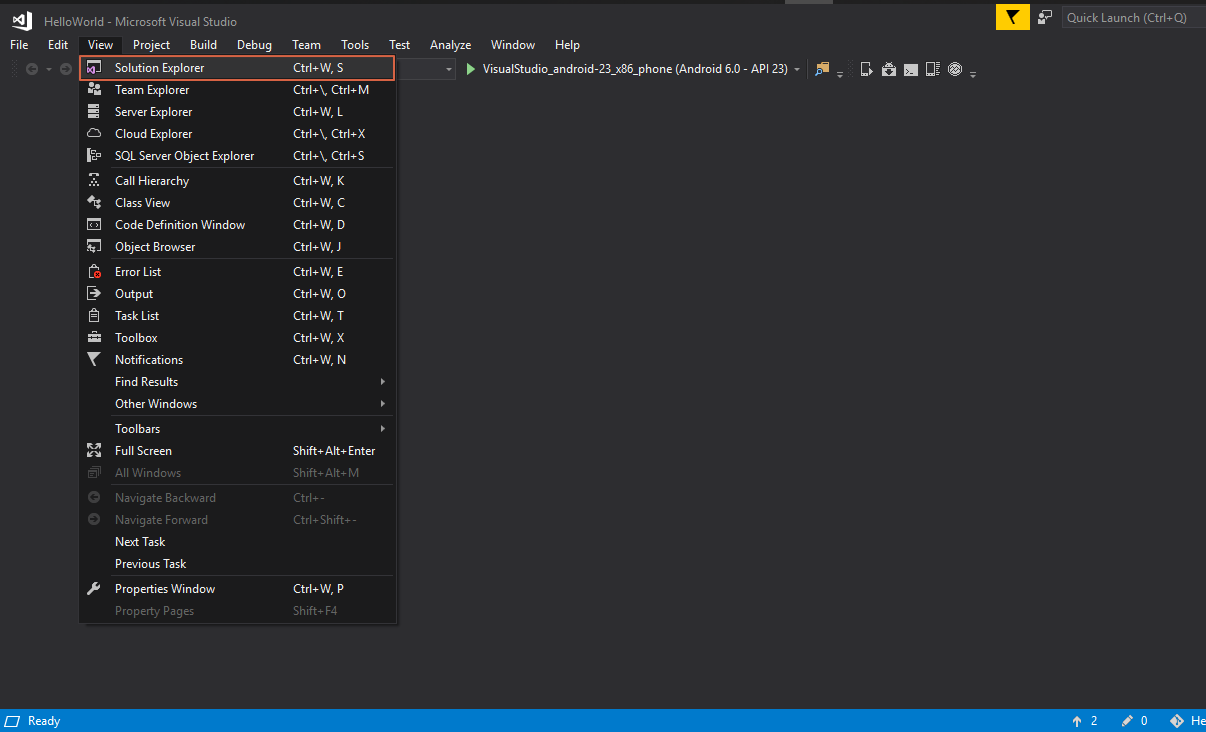
1. On the next window, select Android from the left pane and then Blank App (Android). Give your project a name and also tick the Create a new Git Repository checkbox (This will create a new Git Repository for your project) and click OK.



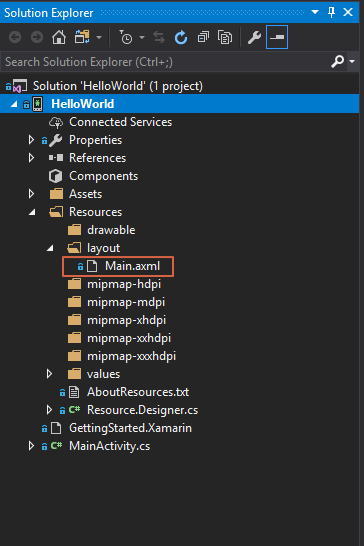
1. Visual Studio will create a new project called HelloWorld for you



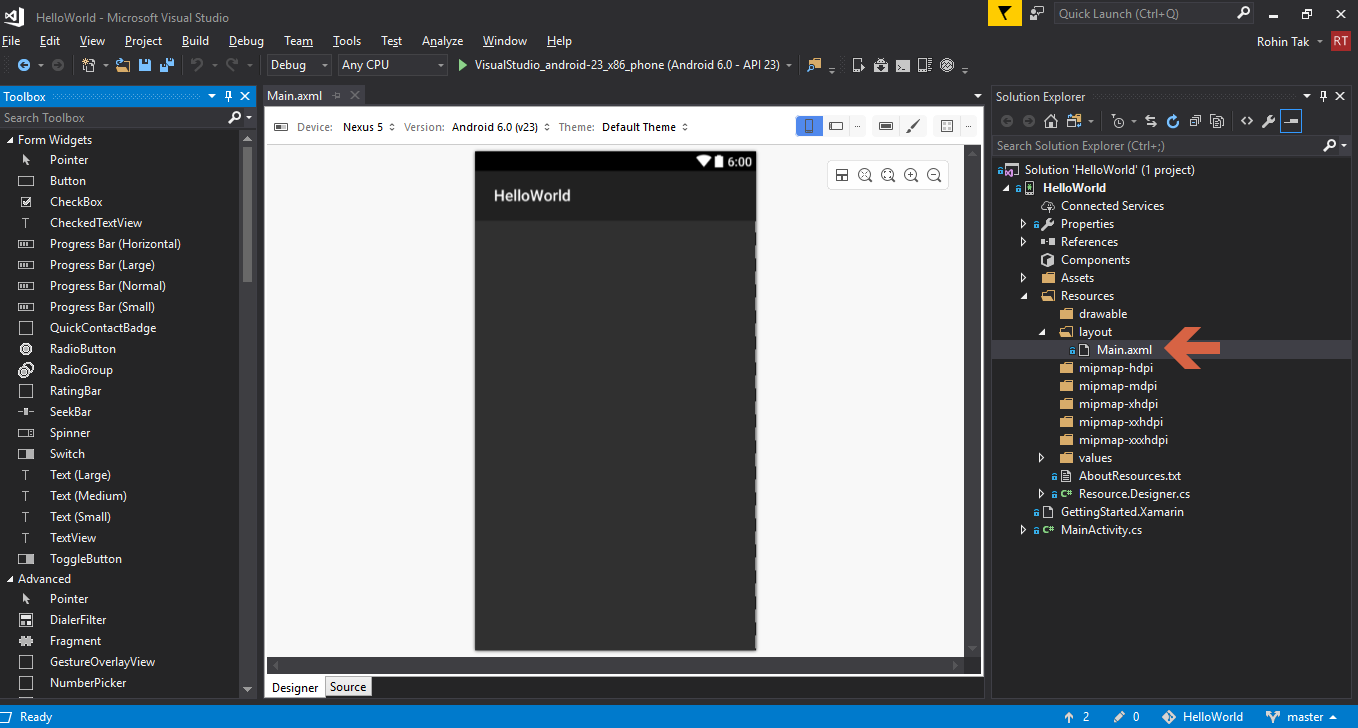
1. Once this is done, open solution explorer to see the project structure. Go to View -> Solution Ecplorer.



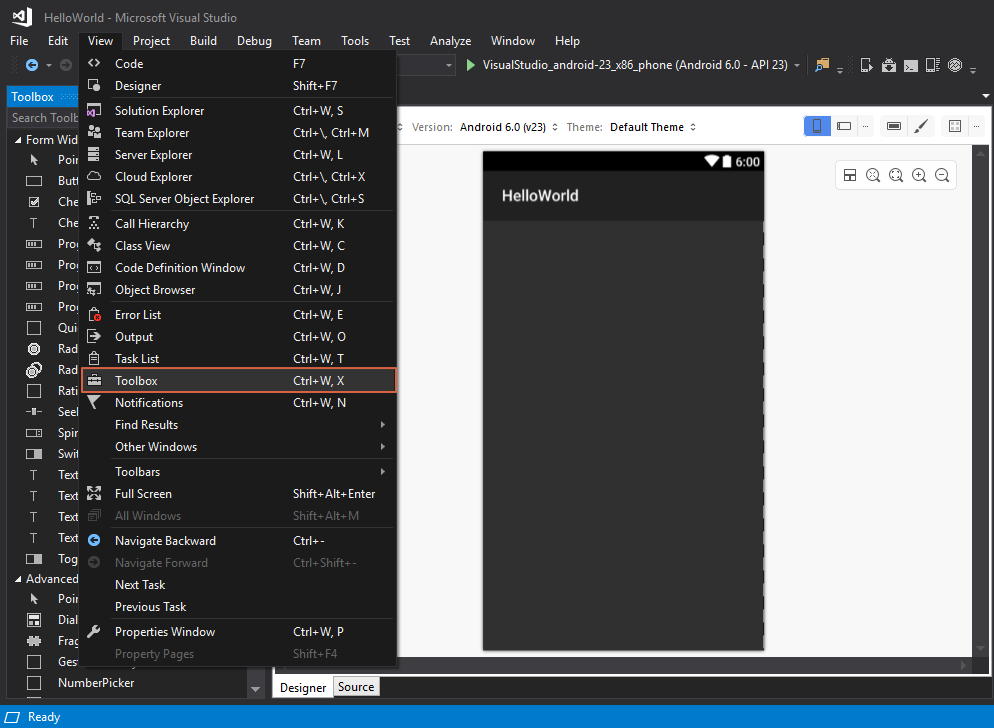
1. In the solution explorer, expand the Resources folder and then layout folder and find a file called Main.axml. This is the layout file or you can say, view of our MainActivity.



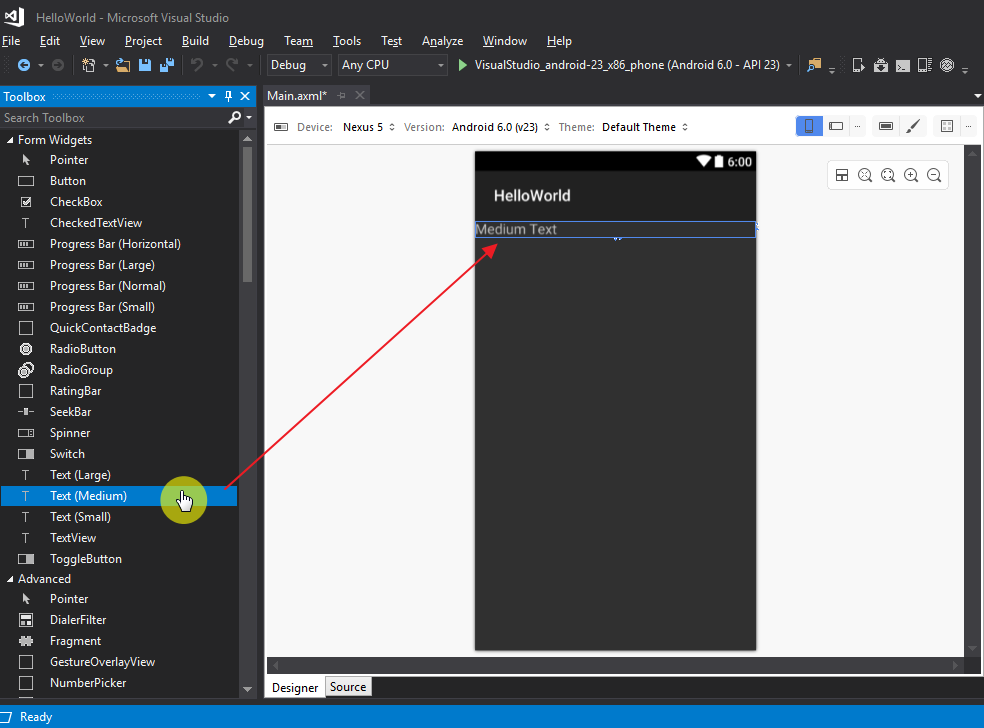
1. Open Main.axml by double clicking on it, this should open the layout designer for you.



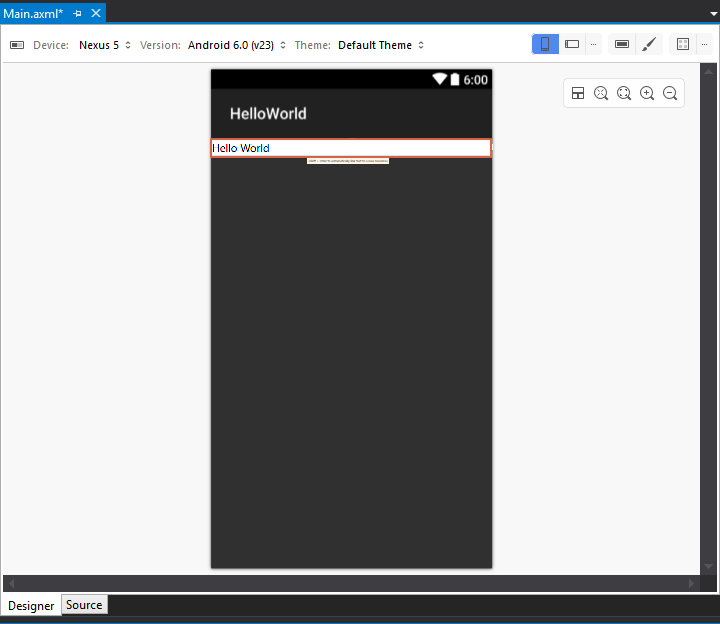
1. If you are not able to see the toolbox on the left side, go to View -> Toolbox to make it appear.



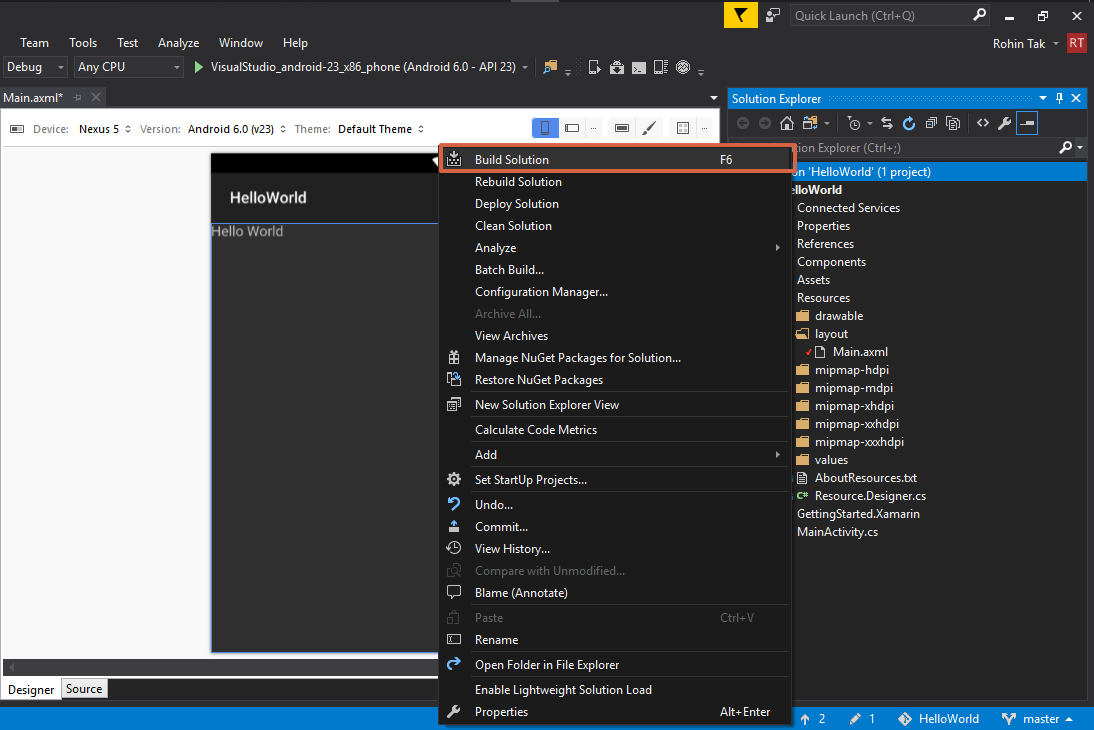
1. Now we will just add a Text View on the Activity which says Hello World.
2. From the toolbox on the left, Select Text (Medium) from the Form Widgets section and drag and drop it to the Activity View.



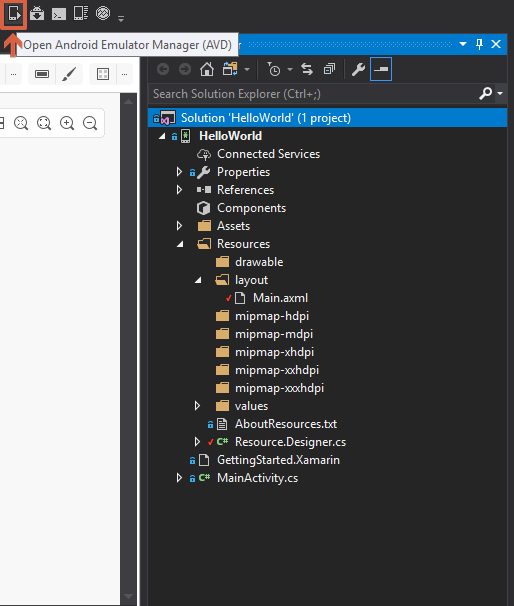
1. Double Click on the Text View and change its text to “Hello World”



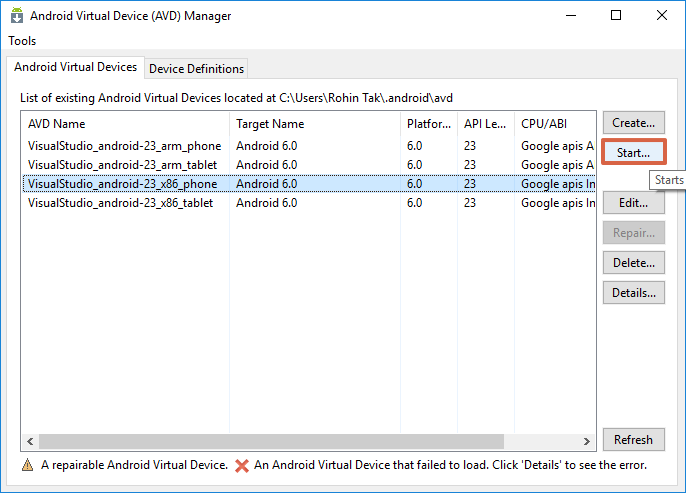
1. Awesome, Hello World app is done, now we just need to build the solution to make sure everything is fine and is ready to be deployed on an Android device or emulator.
2. Right click on the solution and click Build.



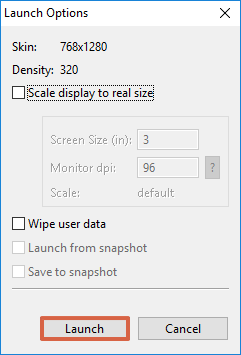
1. This will build the solution for you and it should say Build Succeeded on the bottom left corner on the blue line when it’s done.
2. To deploy and test the application on an emulator, click on Android Emulator Manager (AVD) icon from the toolbar on top.



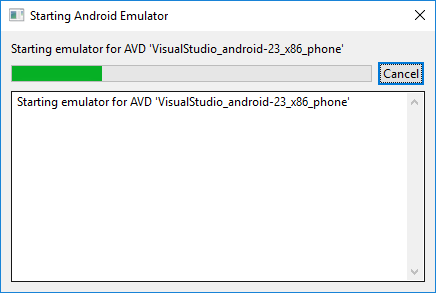
1. This will open Android Emulator Manager where you can select any existing virtual device from the list provided by Visual Studio and hit Start.



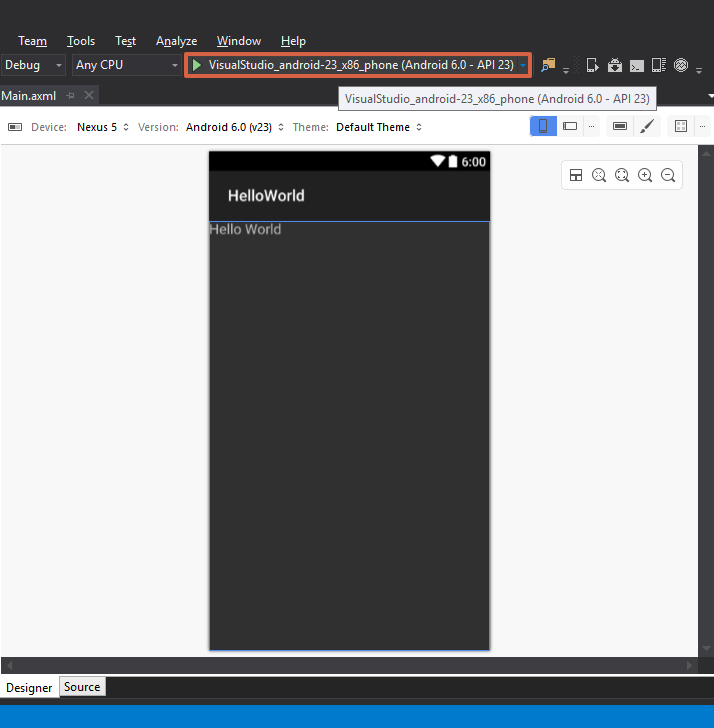
1. Then Click on Launch without changing any configuration on the next window.



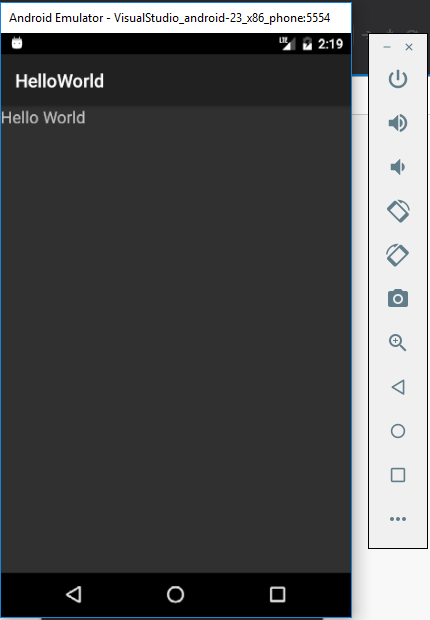
1. This should start a new Android Virtual Device (AVD) on your machine.



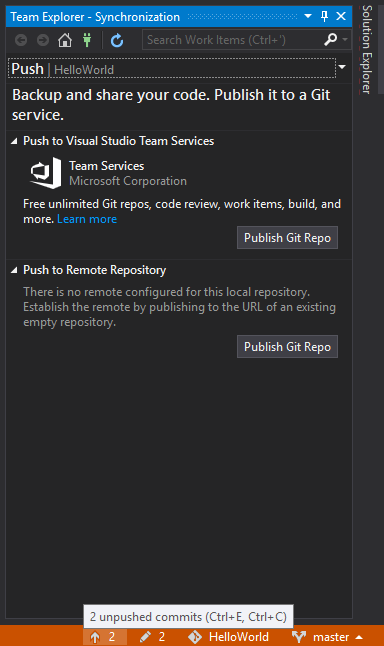
1. Now, come back to Visual Studio after the AVD is started and hit play button by selecting your device from the list to deploy and start you application on the AVD.



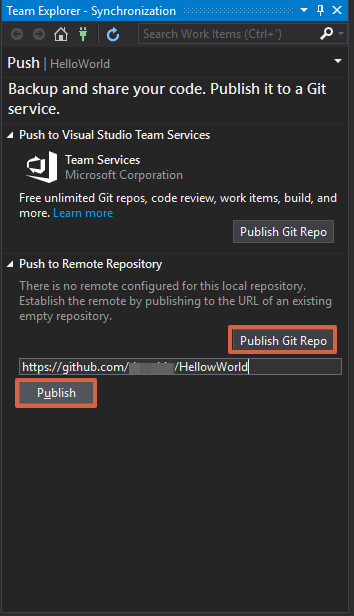
1. Once the application is deployed, it will get opened on the AVD and you should be able to see your Hello World Text on the Main Activity Screen.



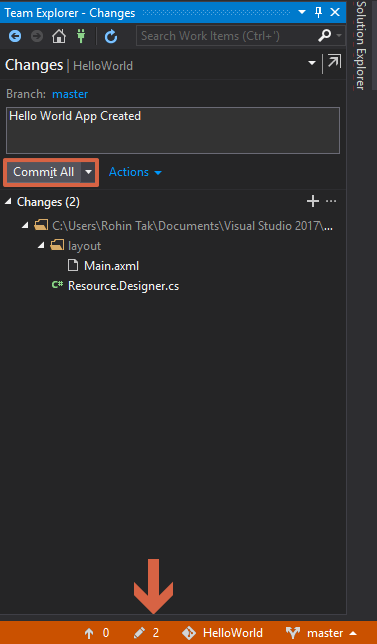
1. Congratulations, Hello World app is up and running on the emulator.
2. It is time to push our newly create project to Git remote repository. Remember that we have created a local repository already while creating the project. So now we need to connect this local repository to a remote Git repository and then push the code.
3. Click on the Push logo on the bottom right corner in Visual Studio. It will open Team Explorer as shown below.



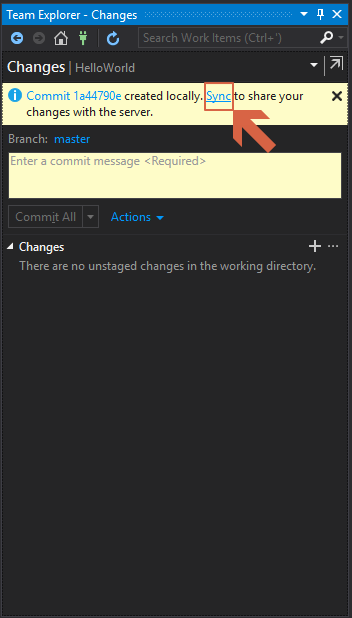
1. Now before publishing we need to create a repository in GitHub to connect this local repository to.
2. Head over to GitHub and login to your account.
3. Create a new repository called HelloWorld and copy the URL to that repository.
4. Once done, come back to Visual Studio and hit “Publish Git Repo” as shown in the previous image and copy the link to the repository and hit Publish.



1. Visual Studio might ask you to provide your Credentials for the first time to connect to Git, once done it should configure the remote repository with the local one.
2. After this, click on the edit icon that says “2 Changes”, to commit your changes locally.
3. Give some commit comments and then hit Commit All.



1. Next, click on the sync link to share your committed changes with remote.

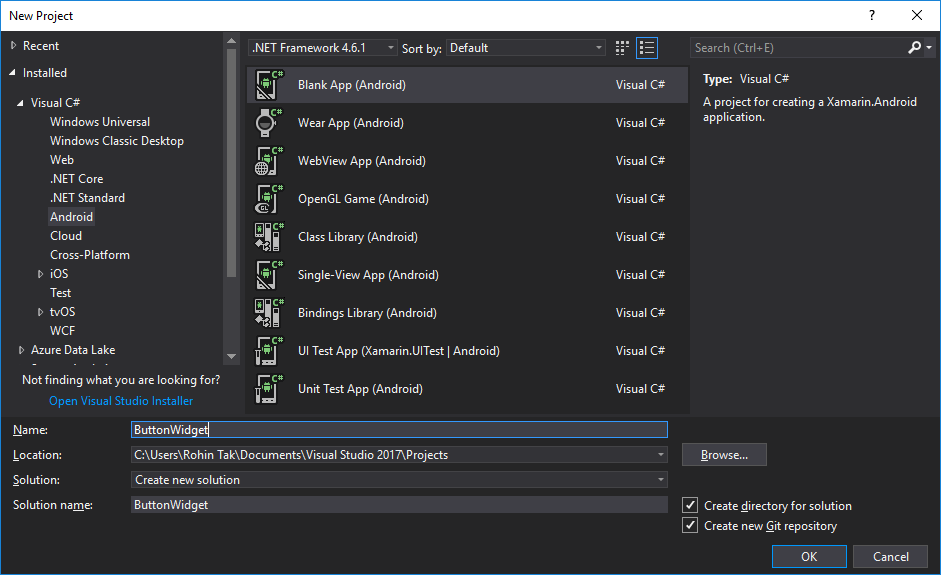


1. On the next page, hit Push to push your changes to GitHub remote repository.
2. Because there isn’t much to test in this application, we’ll be covering that in the next case study.

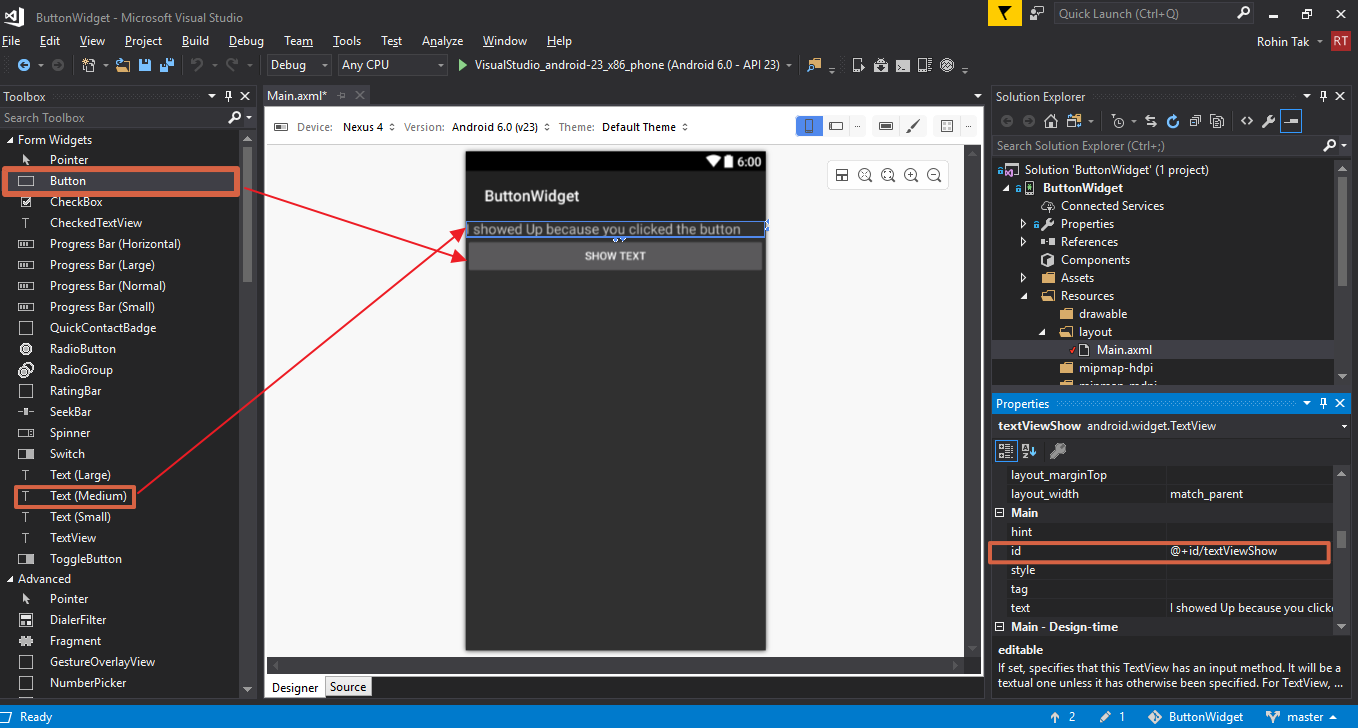
**Case Study 2: Button Widget**

In this case study, we’ll be creating a new Android application which will have a button and it will show a new TextView when clicked. We’ll also be writing UITest for this application.

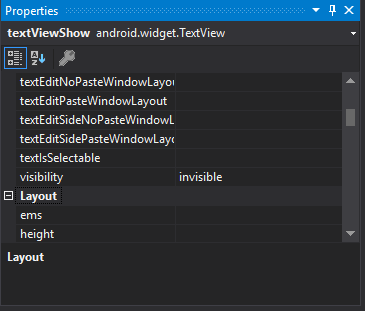
1. Create a new Blank Android Application Project in Visual Studio and name it “ButtonWidget” and click OK.



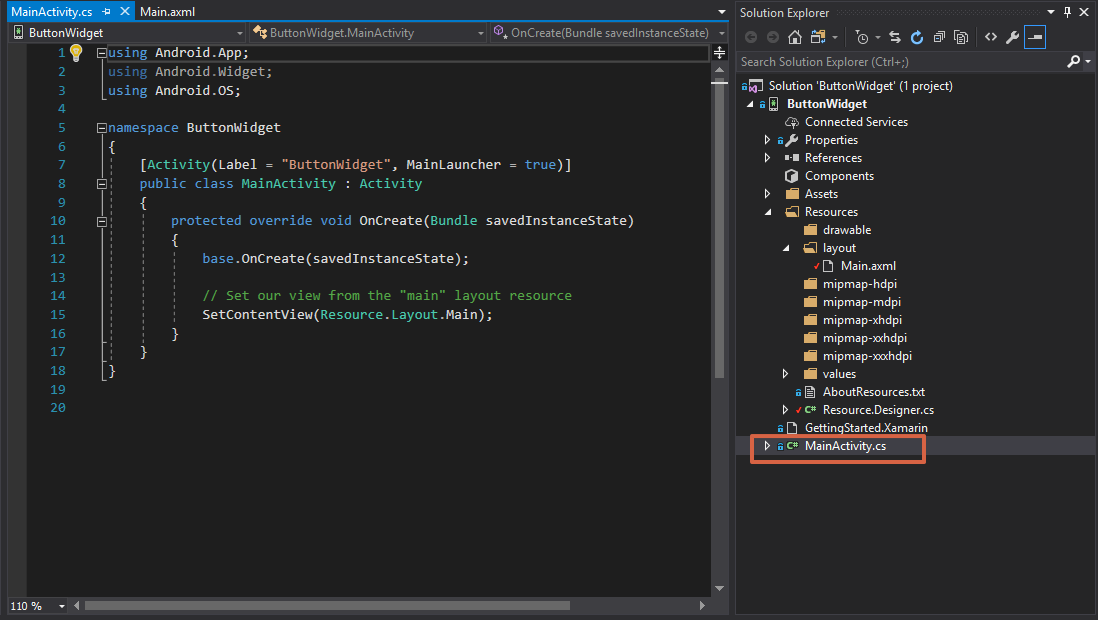
1. After creating the project, open Main.axml file from Resouces -> layout in the solution explorer.
2. Then Add A textView and a Button to the view from the toolbox on the left.
3. Give each of these ID’s to identify them in the code. You can select them and then show property window and give ID’s there.



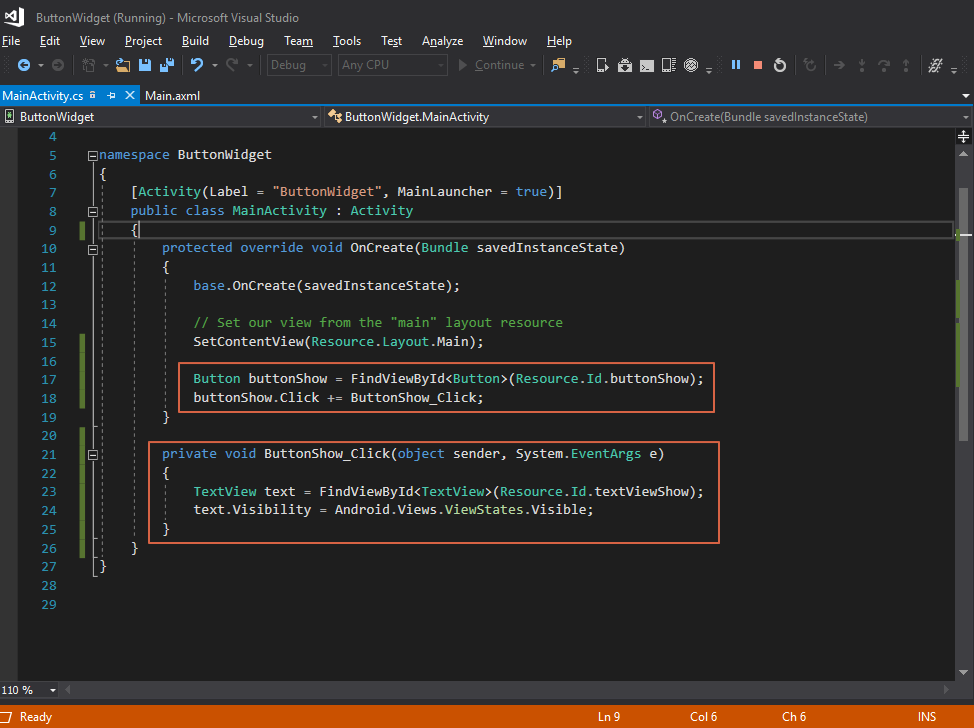
1. Also set the Visibility of the textView to hidden. Because we’ll be showing this text only on button click.



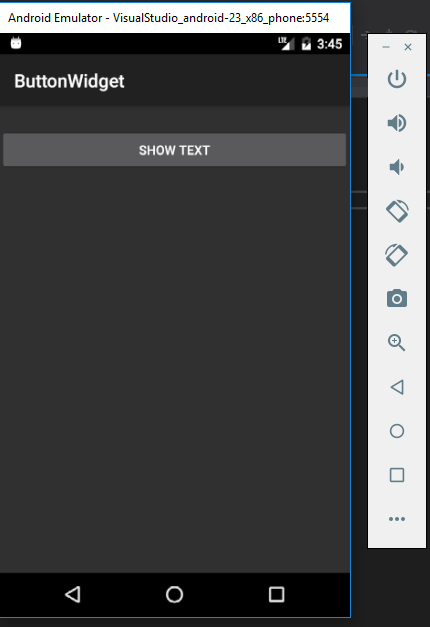
1. Now open MainActivity.cs file from the solution explorer.



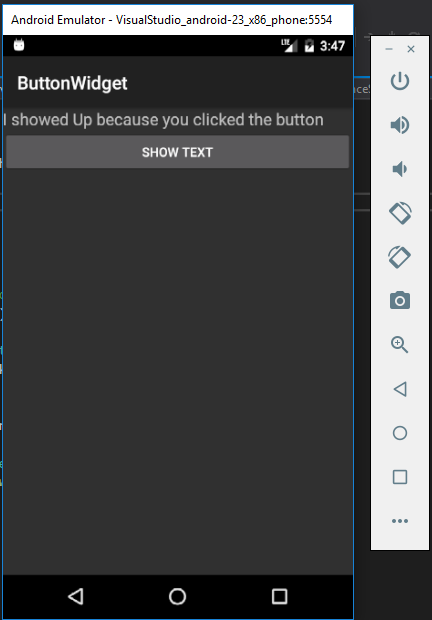
1. In MainActivtiy.cs Change add code to show the textView when the button is clicked.  
   Change your code to match as shown in below image.



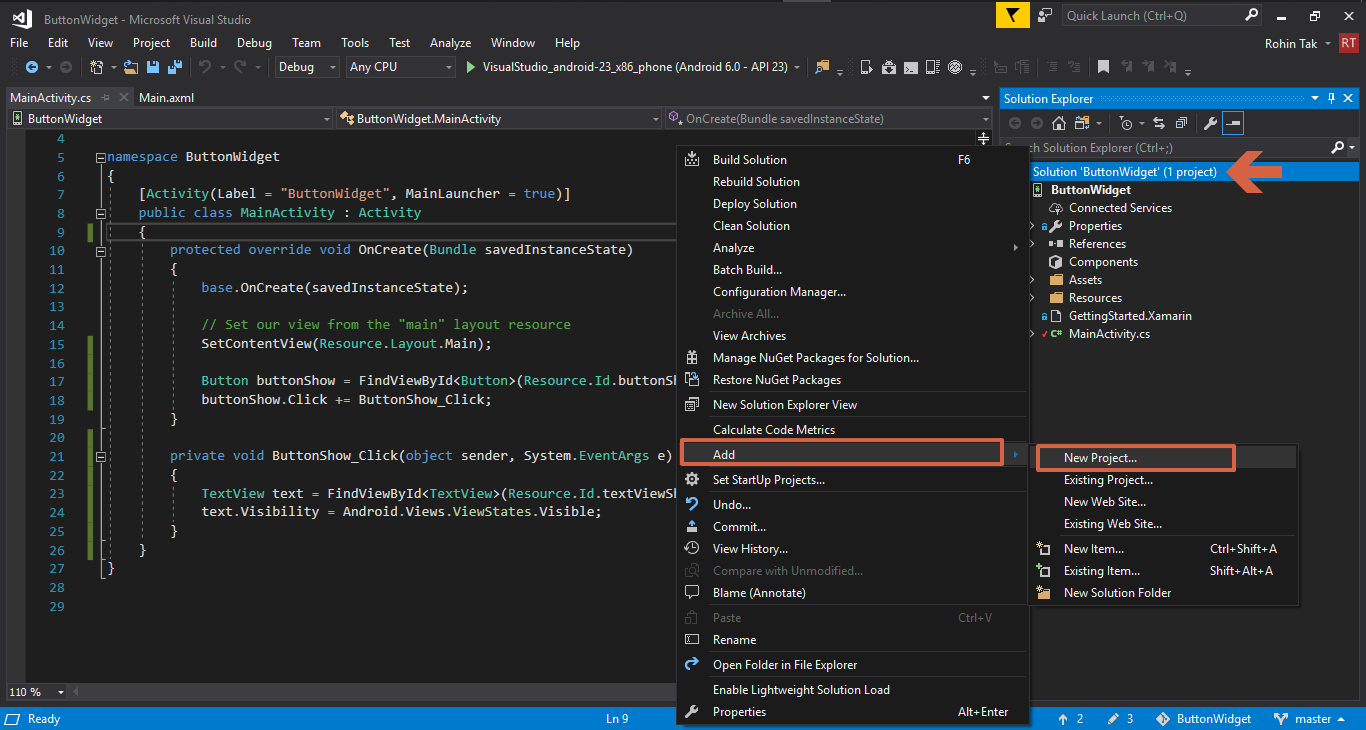
1. That’s it. The coding part is done for the application. Now the TextView added will not be shown in the application until the button is clicked.
2. Build your application and hit Run, You’ll see that the textView is not visible when the app loads.



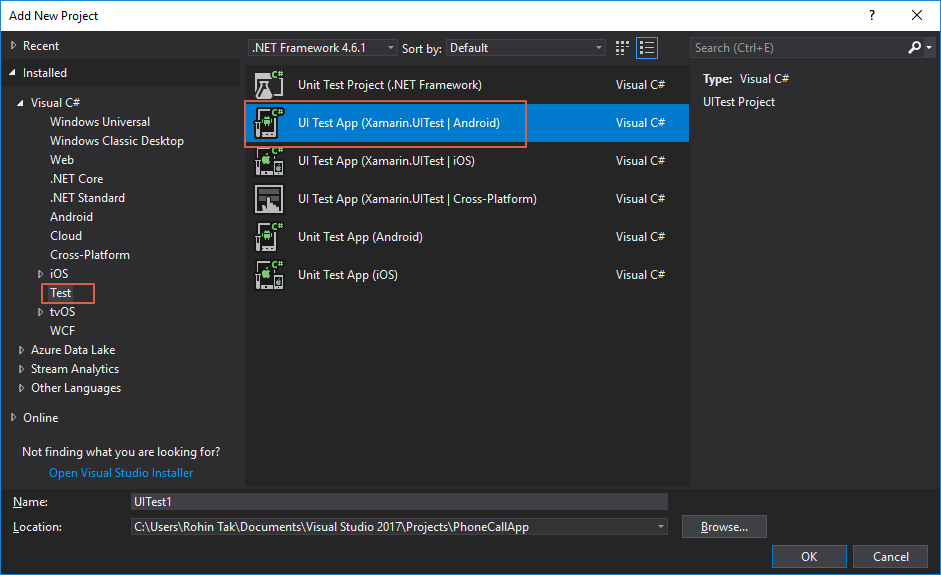
1. Now click on the Button and see that the textView appears.



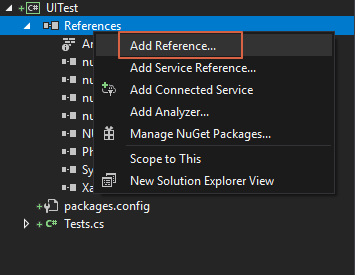
1. Now that the application is working, let’s write Xamarin.UITest for the same and upload it to Xamarin Test Cloud.
2. Add a new test project to the solution.



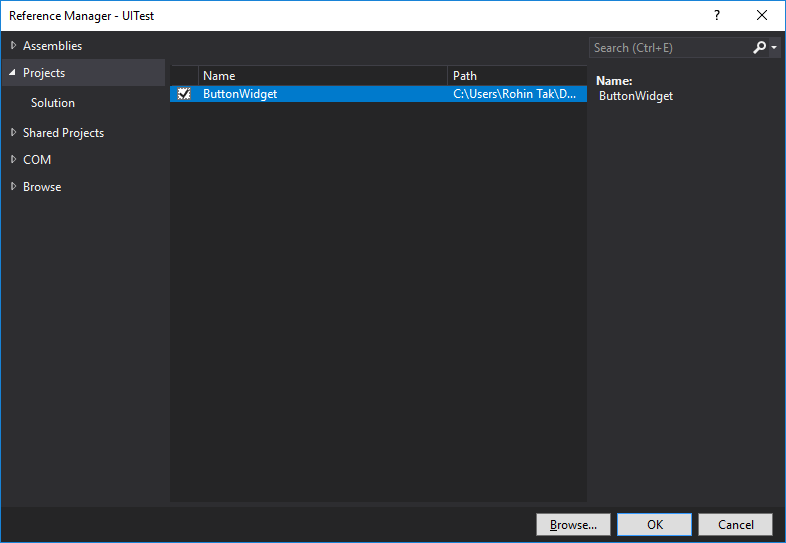
1. In the Add New Project window, click on Test from the left pane and then select “UI Test App (Xamarin.UITest | Android)” , Give the project a name and click OK.



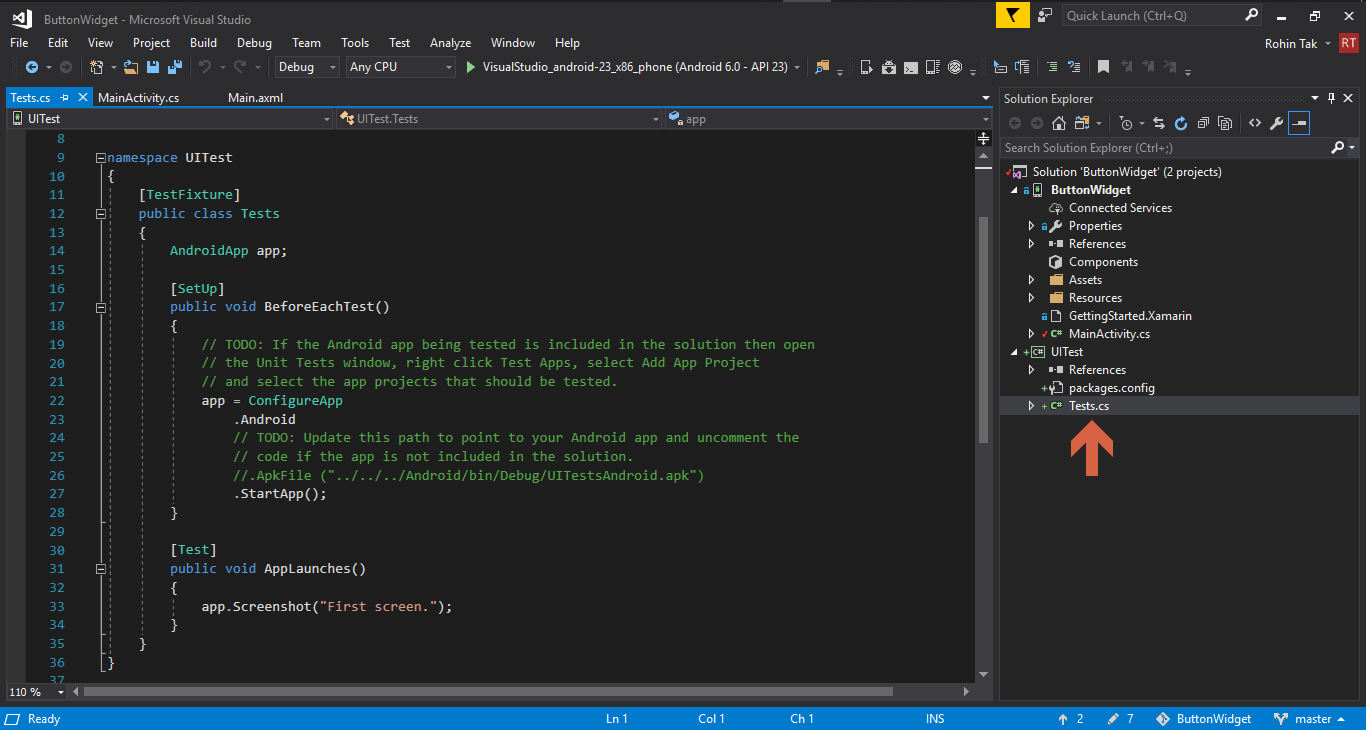
1. Next we need to add a reference to the application project, so UITest project can build and run the application.
2. Right click on References under UITest project and click on Add Reference.



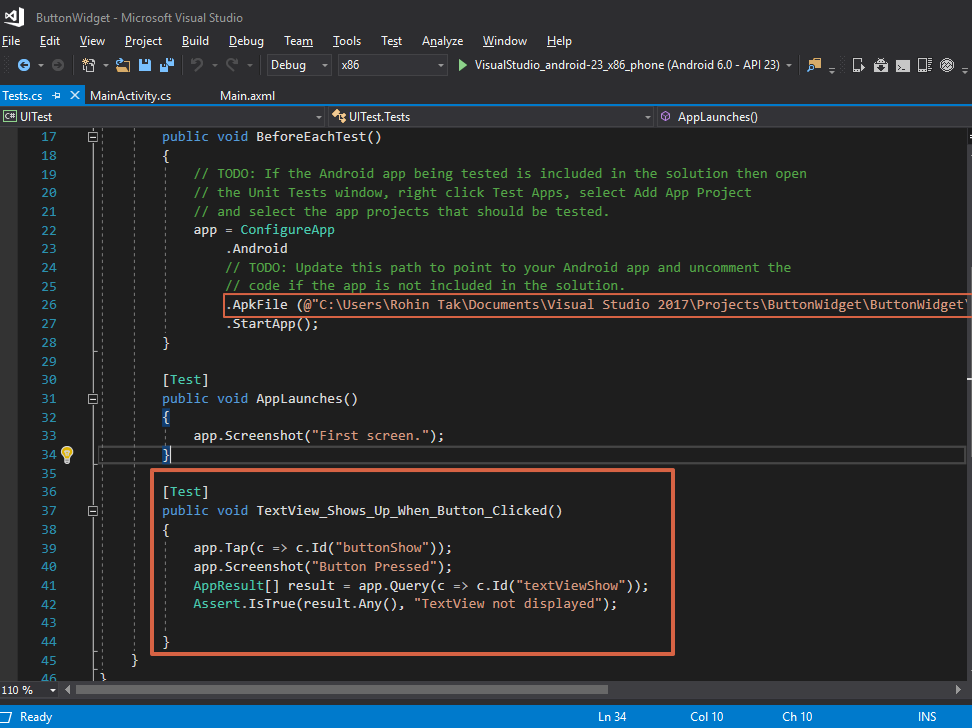
1. On the next screen, select Projects from the left section and then select the ButtonWidget (the application project we want to test) and click OK.



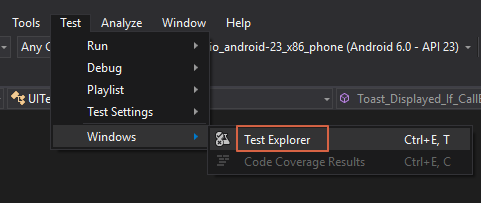
1. We are all set to start writing our tests for the ButtonWidget app.  
   Open Tests.cs file from solution explorer under “TestProjectName” -> Tests.cs



1. Now change the code to add a new Test in the Tests.cs file to test of the TextView is displayed when the button is pressed.



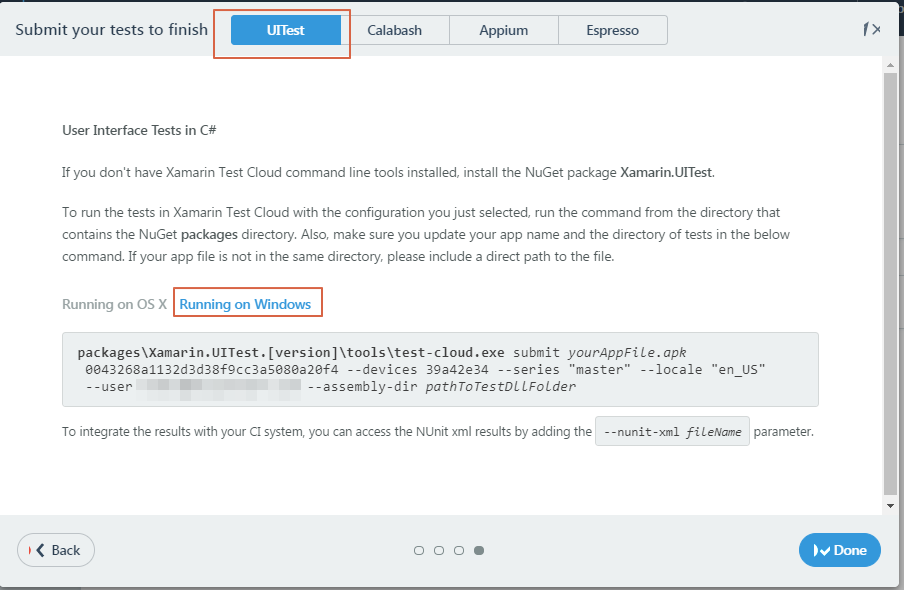
1. Now rebuild and deploy the solution, then click Test -> Windows -> Test Explorer



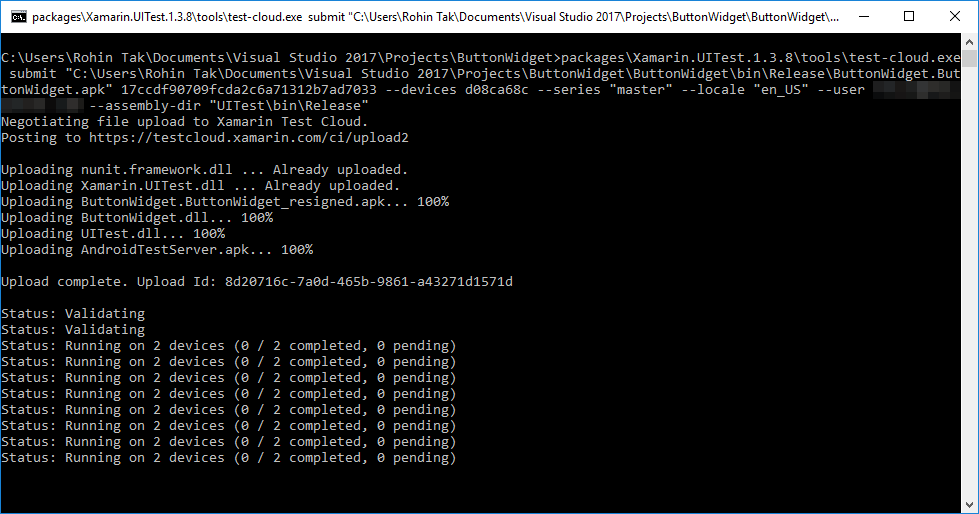
1. You should be able to see the tests written in the Test Explorer.

# 

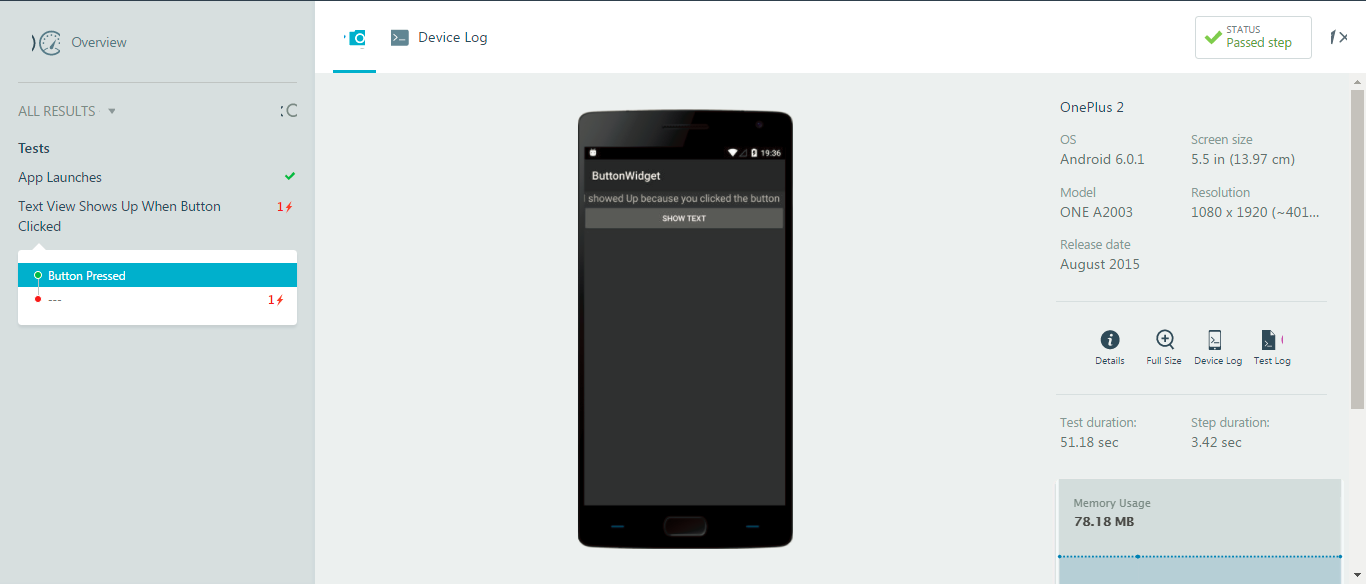
1. Click on Run All to run the tests.
2. Now to upload these Tests to Xamarin Test Cloud, login to your Xamarin Test Cloud account.
3. Go to Account Settings -> Teams & Apps
4. Click on New Team button to create a new team.
5. Add members to the team and then Click on New Test Run.
6. This will open a self-guiding dialog box, where we can select the platform and chose devices etc.
7. Select the operating system as Android and then devices of your choice and go to the last step.
8. You’ll find a screen like below where you get a command to upload the Tests to Xamarin Test Cloud.



1. Before you upload your application to Xamarin Test Cloud, it is important to Build your application in **Release build configuration**.
2. Add Internet permission to the project in the project’s manifest file.
3. Once, you have built the project with Release, you are ready to upload your application on Xamarin and Run the UI Tests there. Use the command from previous step and modify the Xamarin.UITest.[version] to your UITest version and then APK file name with the full path to the apk and relative path to the UITest folder as shown in the below image and then run it on the root directory of your project.
4. Once you have made these changes to the command given, go to the root directory and open command prompt windows there and run the command to upload UITests to Xamarin.



1. And with this, The Application is being deployed and tested on Xamarin Test Cloud on real physical devices.
2. You can use this command with your CI tool to automate this process as part of continuous integration and continuous testing.
3. On checking back in Xamarin Test Cloud’s web application, we can see that the test is passed on selected device.



**Summary**

In this chapter we have gone through the entire process of application development, deployment, writing test cases and testing application on using continuous testing on Xamarin Test Cloud. There were 2 case studies used in this chapter to explain the process step by step from creating a simple android project to writing UITests and using Test Cloud for continuous testing.