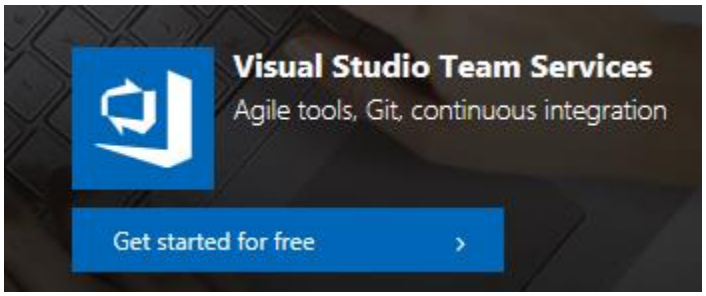
DevOps Hands on Lab

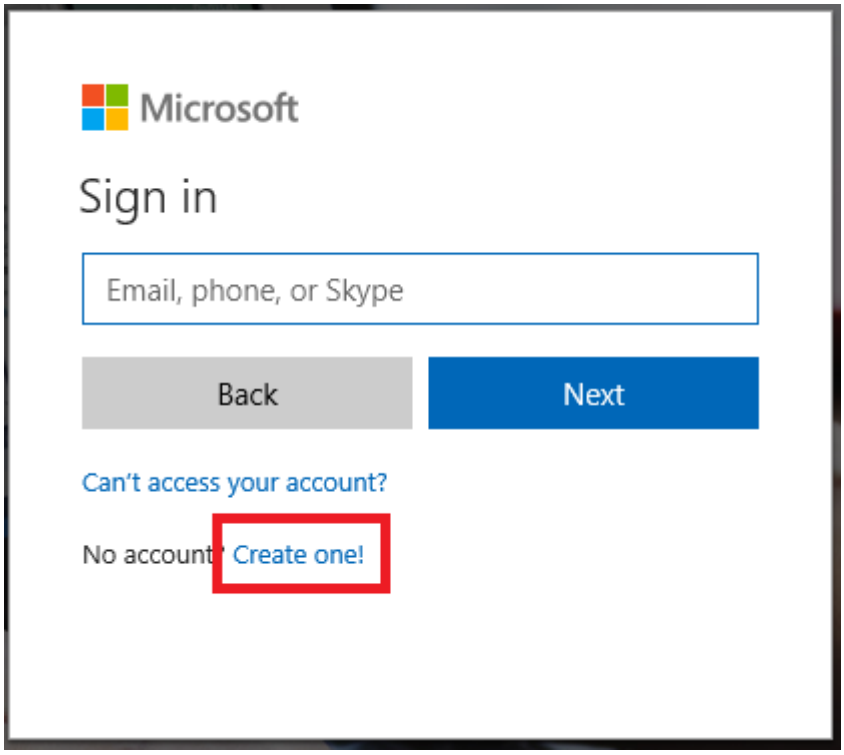
Pre-requisite: Azure Account

Exercise 1: Create VSTS account, add a Team Project in it and Resource Planning

1. Start browser and enter URL http://www.visualstudio.com
2. Click on Visual Studio Team Services - Get Started for Free



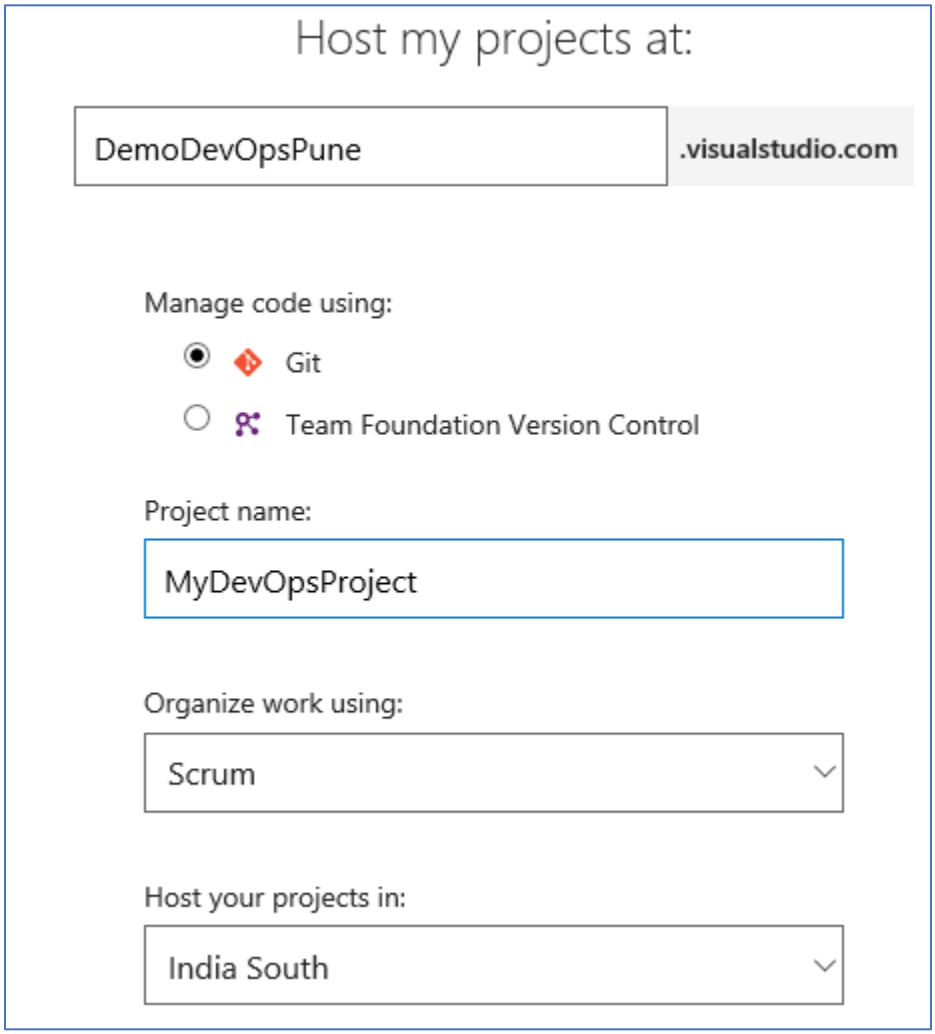
3. Sign in with any Microsoft Account (Please do not use your company id, use a personal one, preferably the one you used while creating Azure Account). If you do not have any id click on Create one!



Follow instruction for creating a new or enter email and password.

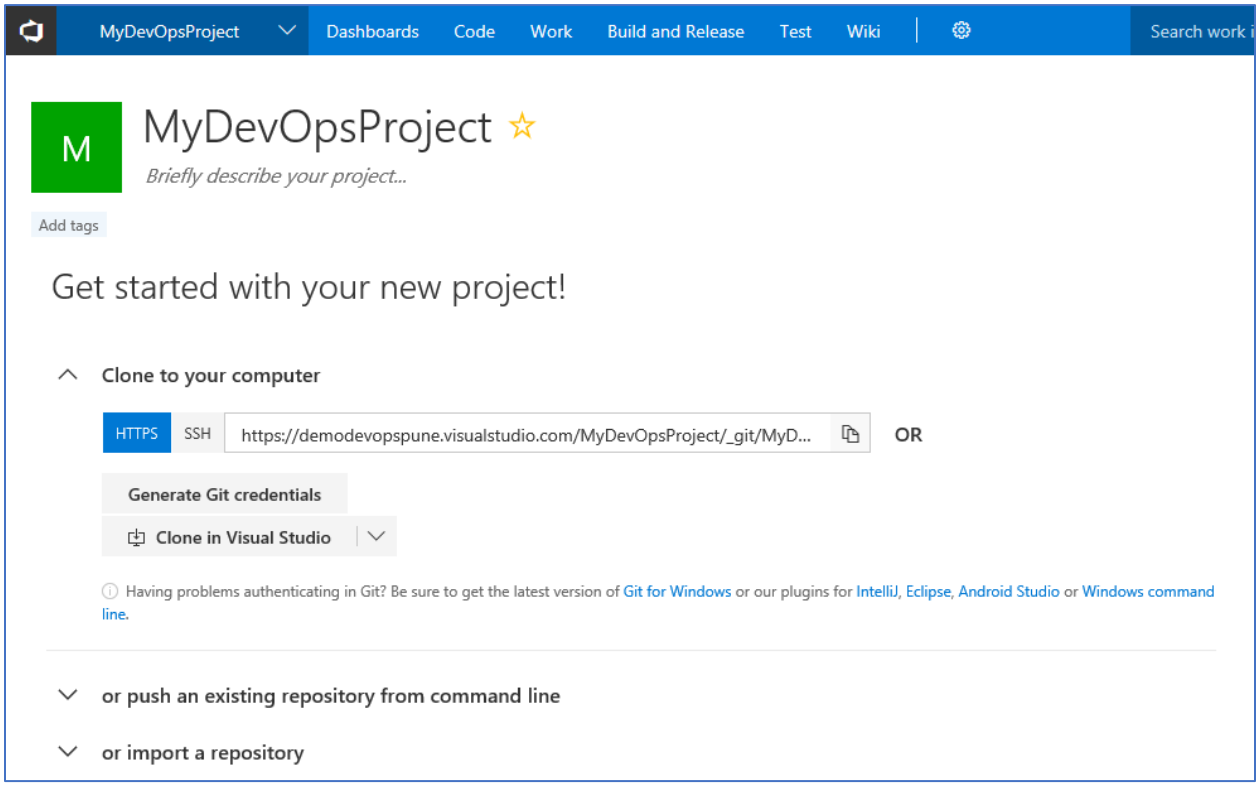
4. Click on Create new Account

5. Provide unique name for the VSTS account, provide any name of your choice. Click on change details. Make sure that you have selected Git as source control and Scrum as process template as follows



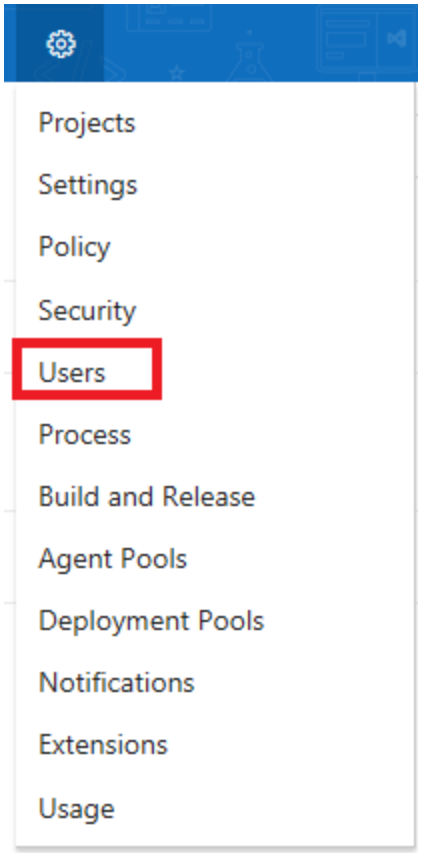
Click on Continue button. after a short while a new account with the specified project name will be created.

6. You can see following on the screen



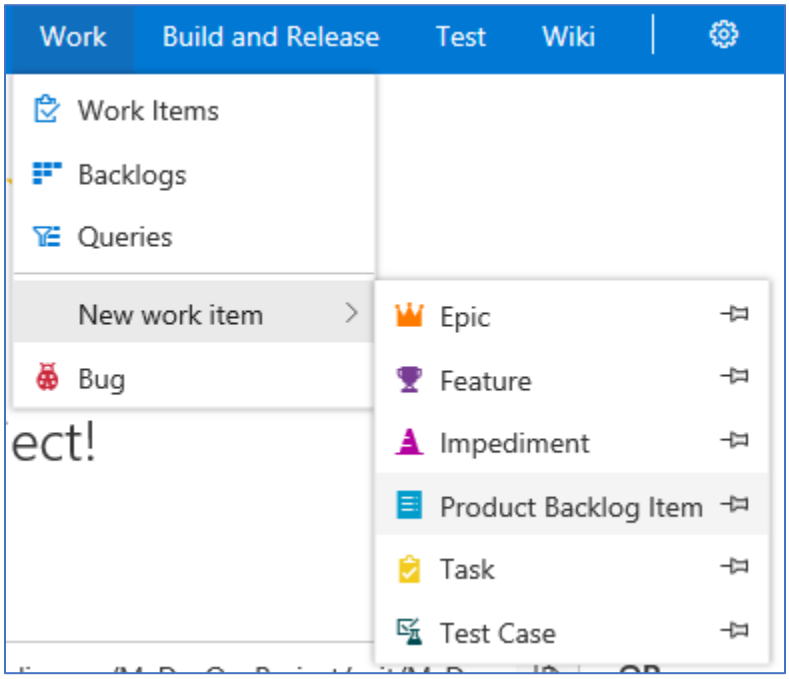
7. Let us add a couple of users to out new account. go to home for account by clicking on left most icon 

8. Select Users from Gear icon

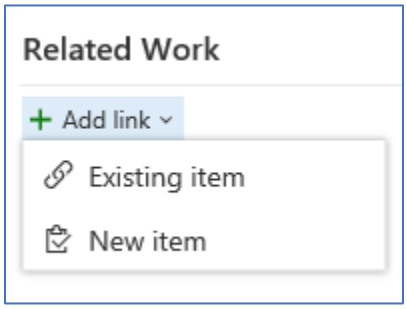


9. Provide user name, select that he/she has access to the new project created and click on add. You can add multiple users at a time by separating the list with ;
   You can see the users added and their access levels. we can add up to 5 users to the account, Browse to the project name

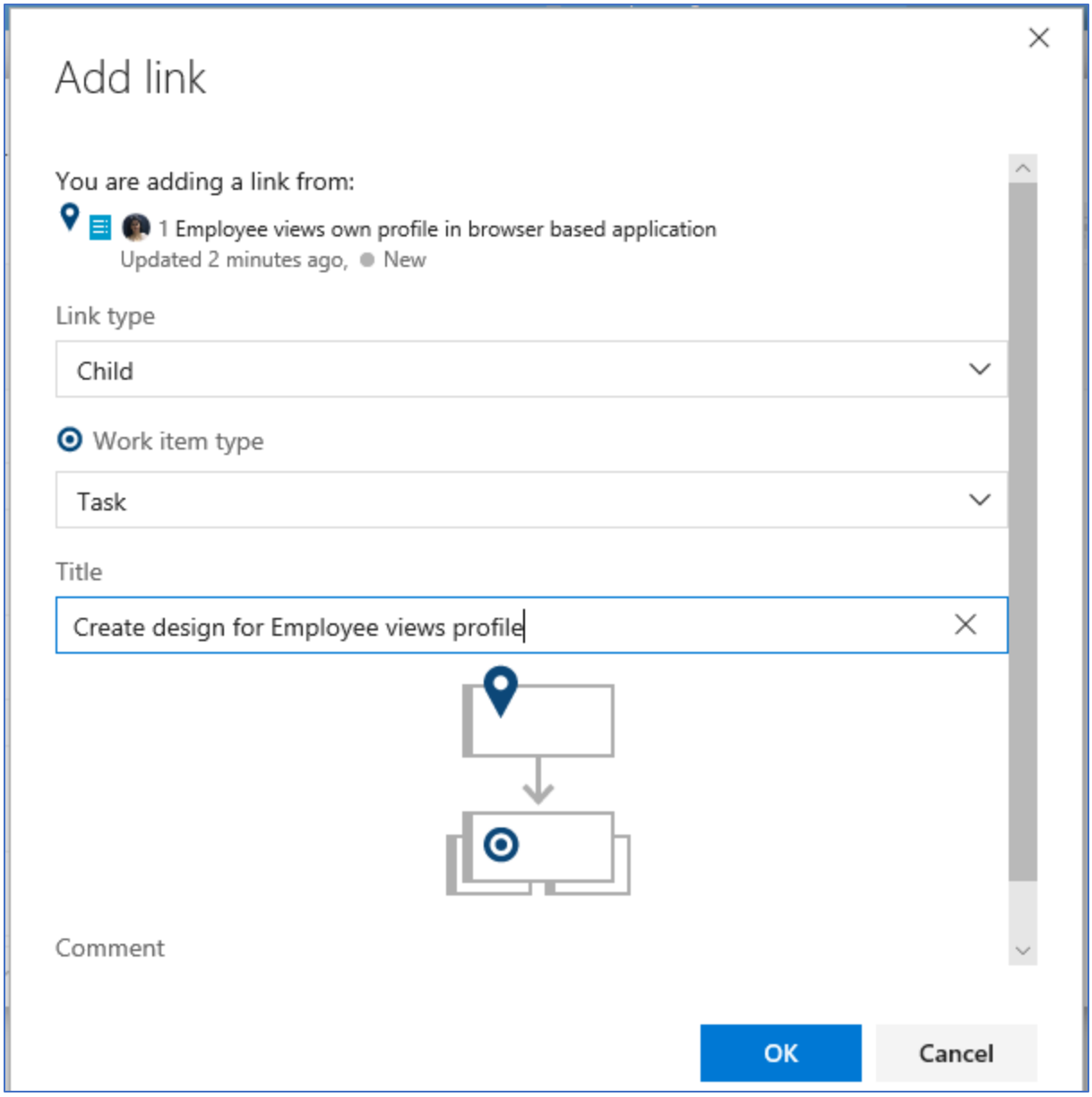
10. Let us add work items to the project. Select Work – New Work Item – Product Backlog Item. You will get UI for Product Backlog Item



11. Enter "Employee views own profile in browser-based application" as Title for PBI, assign to yourself, click on the down going arrow for Save and Close and select Save so as the control remains in the work item itself. Enter Business Value as 5 and Effort as 1.
12. Click on Add Link for Related Work and select New item



13. Make sure that the Link Type is Child, Work Item Type is Task and provide "Create design for Employee views profile" as Title and click on Ok



the UI for Task is opened, assign this task to one of your team members and enter remaining work as 5. Provide Activity for task. Make sure that you Save and Close this work item so as to return back to PBI. Repeat the steps for adding child to PBI and add 3 more tasks as follows with Activity also set

Write code for Employee views profile -Remaining work 8

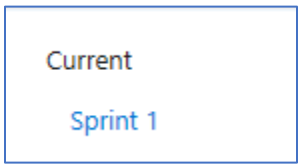
Create PBI for Employee views profile – Remaining" Employee edits own profile" with the similar 4 tasks. (Provide Effort and Business Value)

Create another PBI with title "HR Person creates Employee Profile" and add 4 tasks to it also. Provide Activity for each task (Provide Effort and Business Value)

14. If you happen to have Excel with Team tab in it, you can easily do the same by directly connecting to VSTS from Excel. There is a file named DevOpsData.xlsx
15. Click on Backlog under Work tab for the project to see the work items added.

16. Click on Forecast toggle switch and make sure that it is On

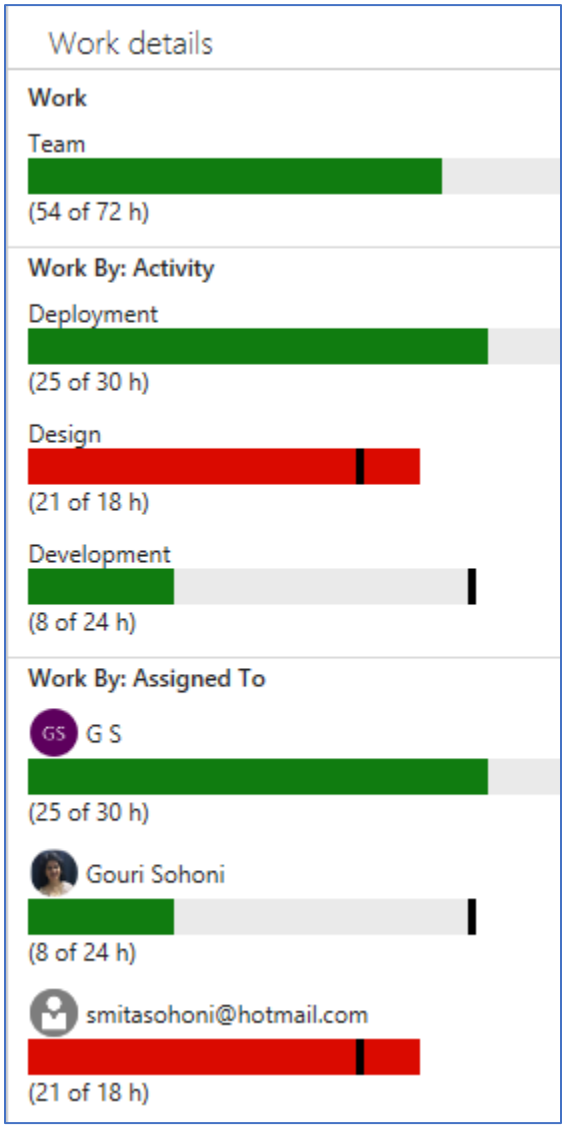
17. Automatically the velocity will be considered as 10 and you will see the work items getting split into sprints. (You can try changing the velocity)
18. Drag and drop first 2 PBIs to Sprint 1. You can see the Iteration Path changed for them. Select Sprint 1 from left hand column below Current



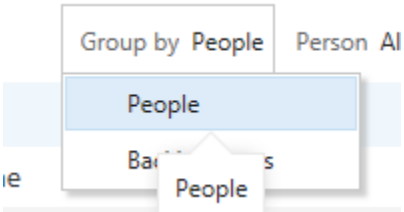
Click on Set Date for the sprint

19. Set Today's date as start date and provide 4 weeks sprint duration.
20. Select Capacity tab, make sure that all team members who have been assigned work items are listed (add them if they are not shown). Provide their activity and capacity per day. Do not forget to Save. Immediately you will start getting charts with green or red colors on the

right-hand side. Remember green is good.



21. Click on Board to view all the work items in different states. Move a task from To do state to In Progress State. Try changing states for tasks.
22. The grouping of Board can either be by People or Backlog Items

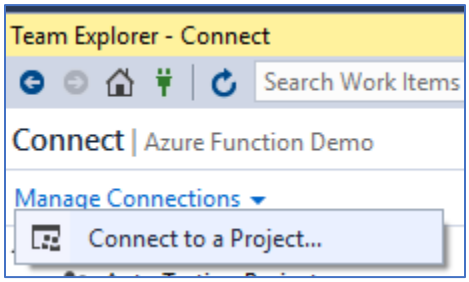


23. You can also view data for a particular team member.

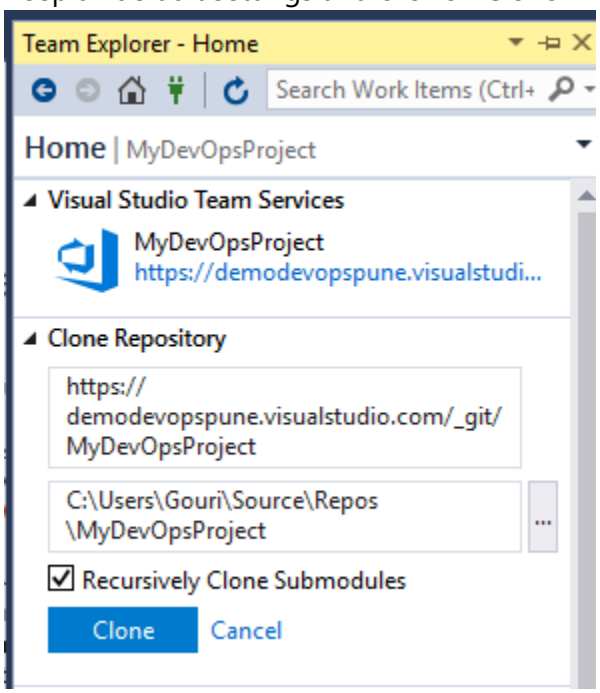
Exercise 2: Add Code to your project, add it to Source Control.

1. Open Visual Studio 2017 Select Team Explorer (if it not shown select from View)

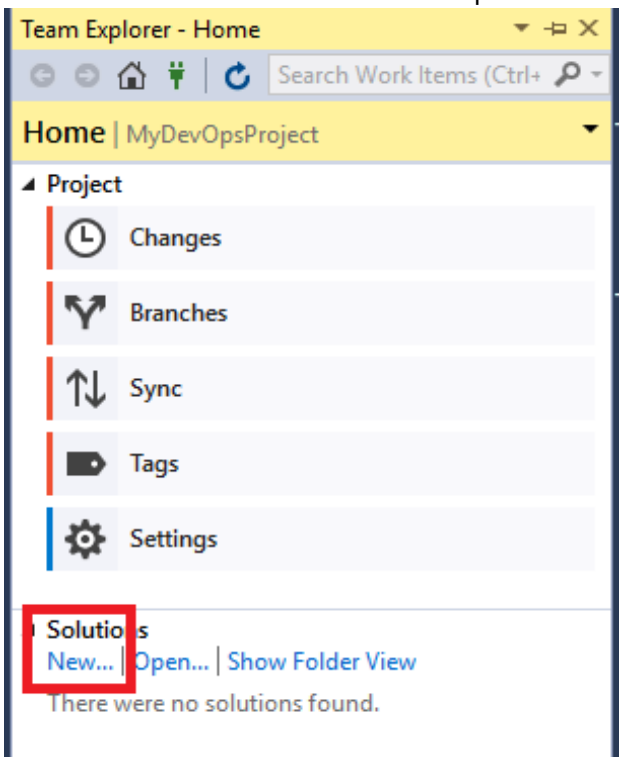
2. Click on Manage Connections and Connect to a Project



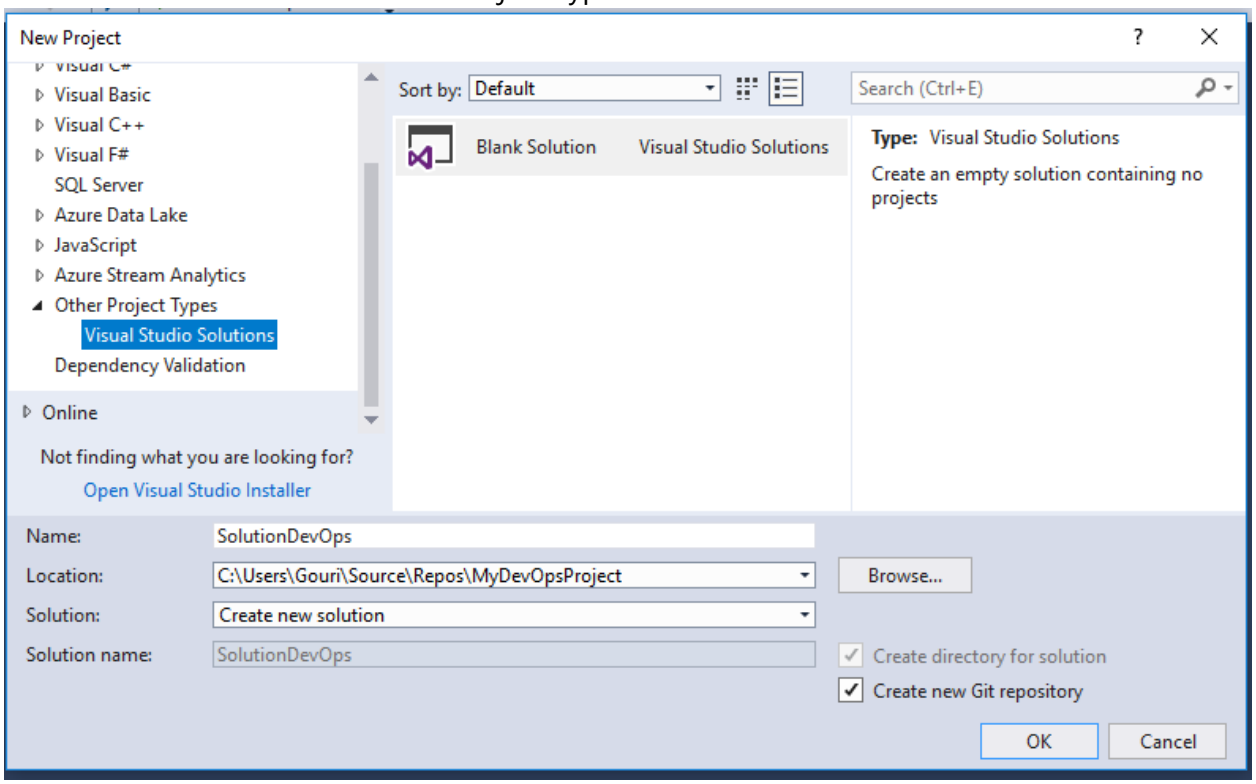
3. Make sure that the account with which you have created VSTS account is selected and Connect to the Team Project
4. Keep all default settings and click on Clone



5. Click on New Solution from Team Explorer

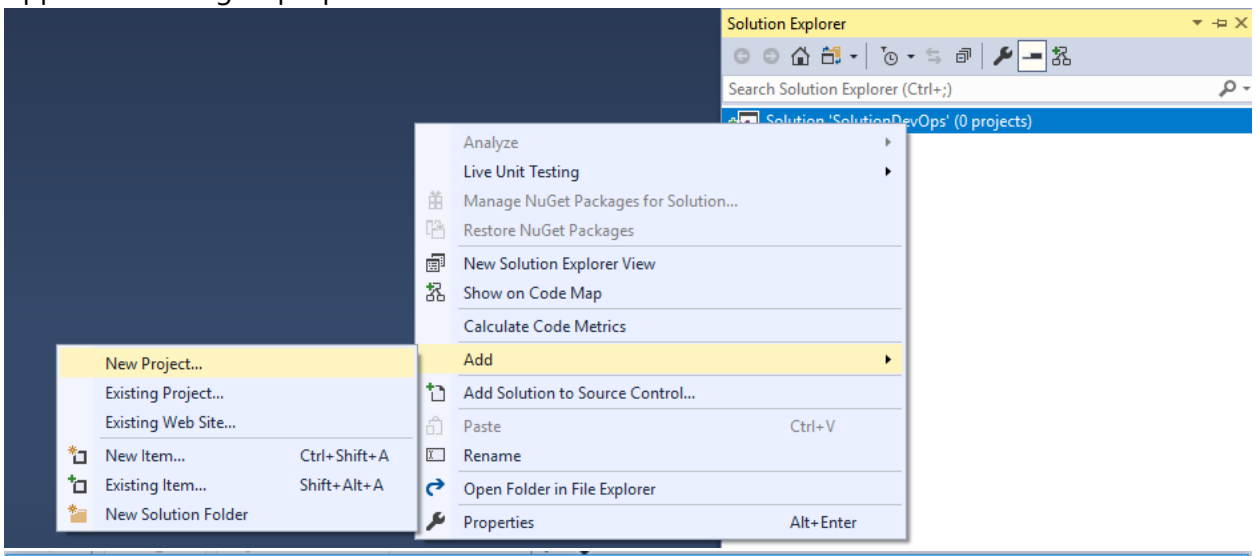


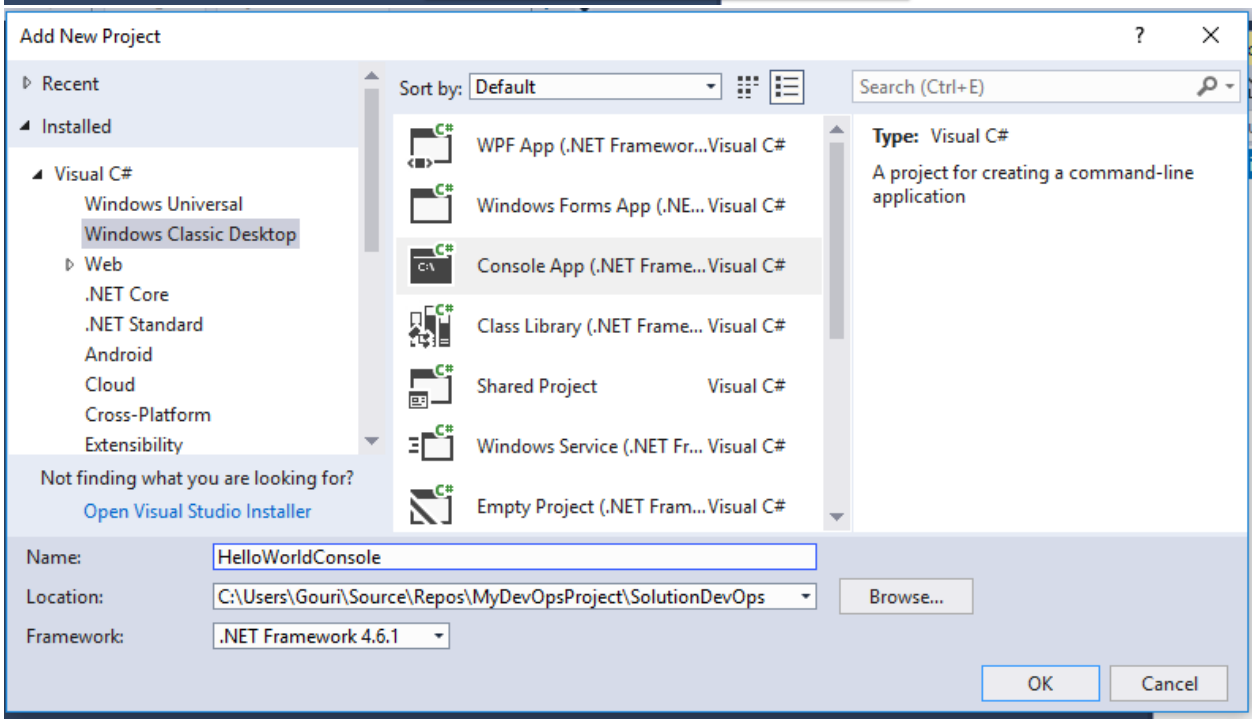
6. Select a blank solution from Other Project Types – Visual Studio Solutions.



Provide name for the solution and click on Ok button.

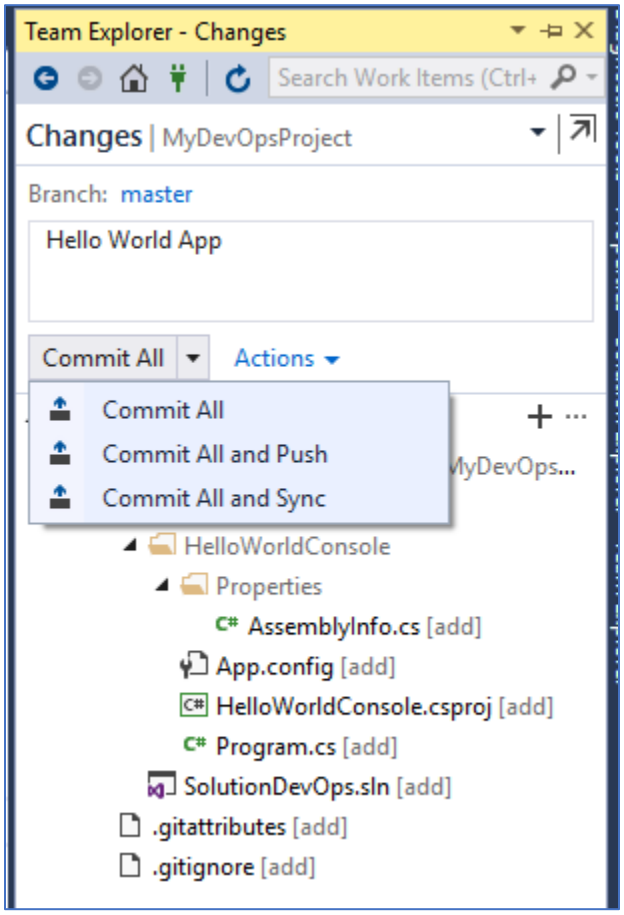
7. A blank solution will be added. Let us add our favorite Hello World to it. Right click on Solution name in Solution Explorer, select Add – New Project. Select project of type Console Application and give proper name.



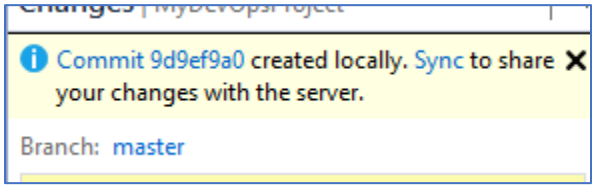
8. 

9. Click on Ok to create the application
10. Add 2 lines of code to display Hello World! and wait on next line.
11. Execute the application to make sure that it works.

12. Select Team Explorer – Changes. Provide a comment and click on down going arrow for Commit to see 3 options



13. Select the option of Commit All. You will see that the commit was done locally.



14. You can verify that the code is not shown in source control in VSTS. Click on sync followed by push so that the code is added in Source Control.

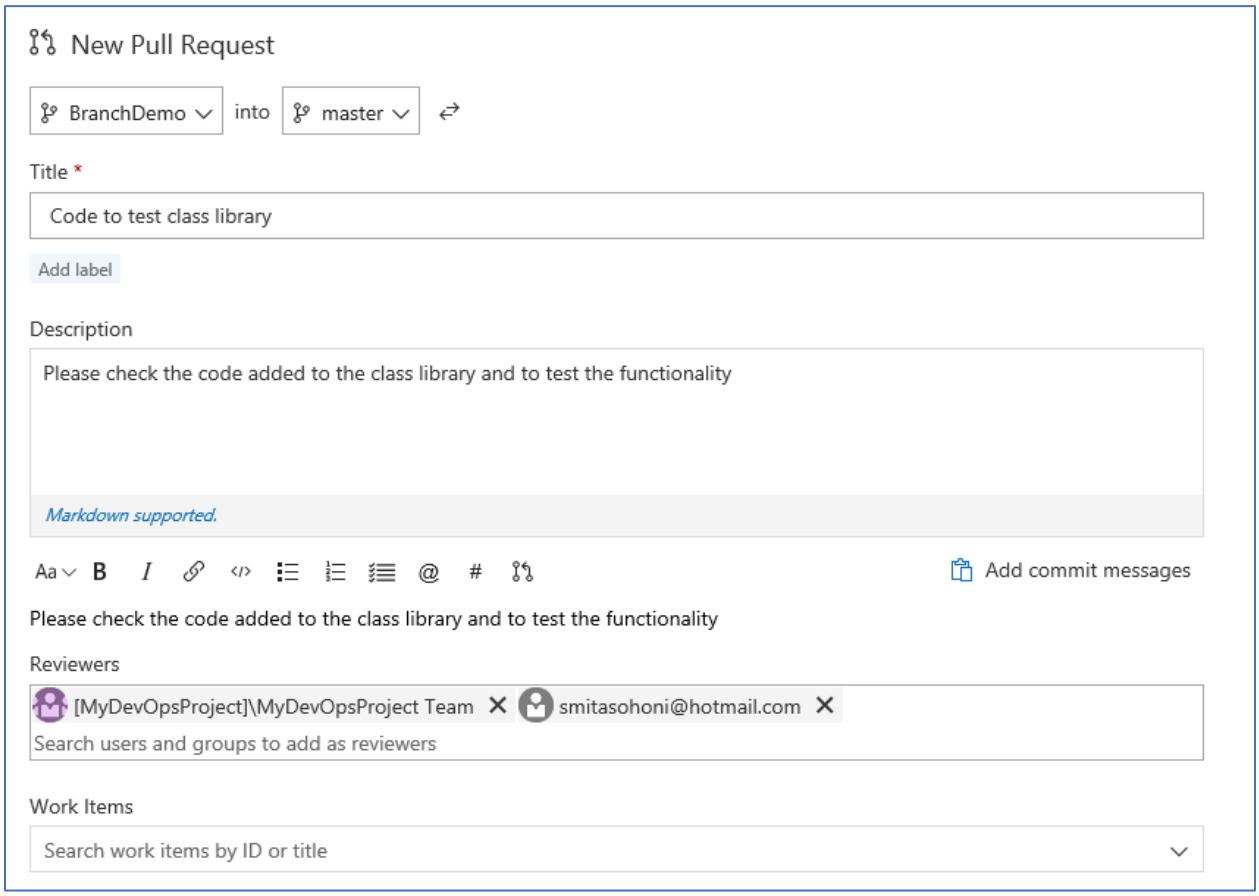
Exercise 3: Work with Pull requests, merge and create Build

1. Go to Home in Team Explorer and select branches. Create a new branch, provide name and click on Create Branch. Make sure that check box for Checkout branch is clicked.
2. Add another project to the solution of type class library, provide appropriate name for it and the class in it. Add 2 methods for addition and subtraction of 2 numbers to the class as follows

```
public int Sum(int i, int j)
```

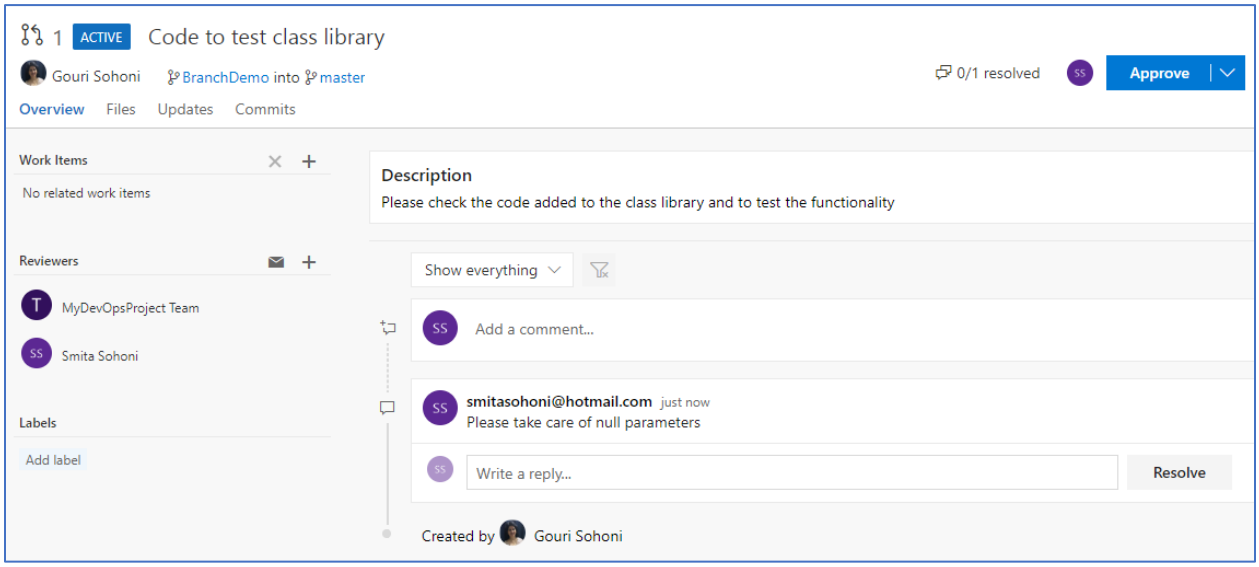
```
{
    return i + j;
}
public int Subt(int i, int j)
{
    return i - j;
}
```

3. Commit this new code to local branch.
4. Add code to use the method sum and again commit it with comment.
5. Push
6. View branches in VSTS
7. Create a new pull request, click on Code – Pull Requests – New Pull Request. Select the branch name, provide Title and name for the reviewer.

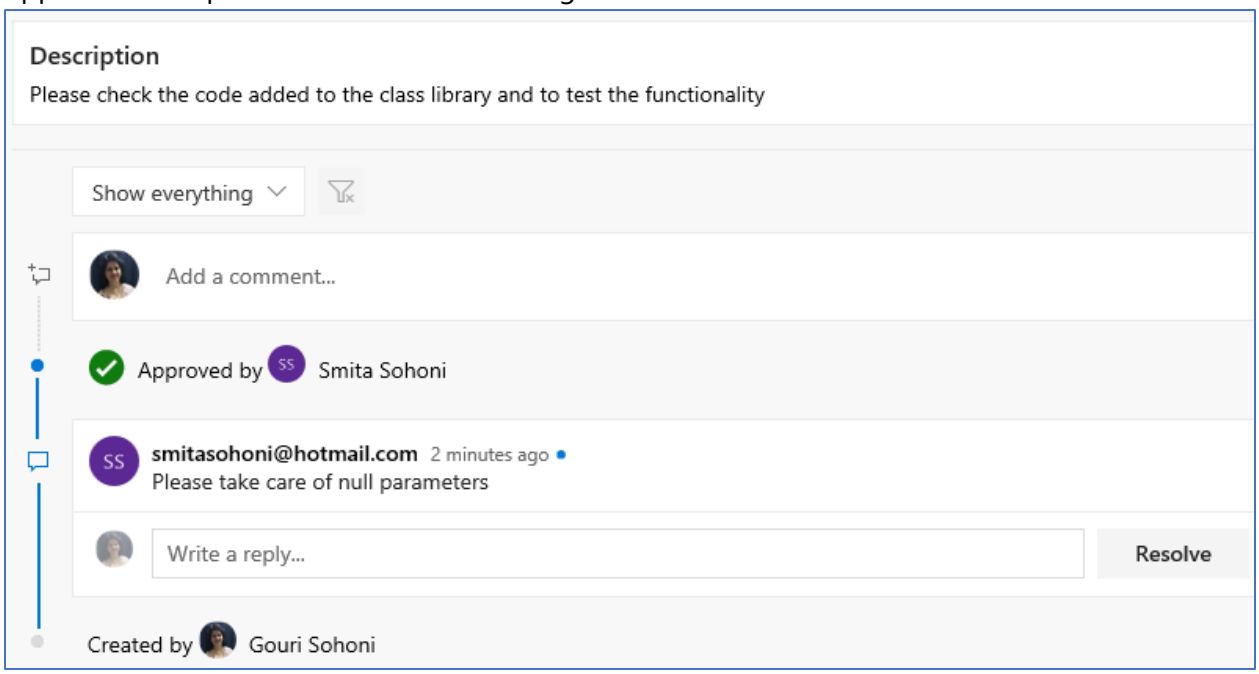


8. Go down to view differences in the 2 branches and click on Create

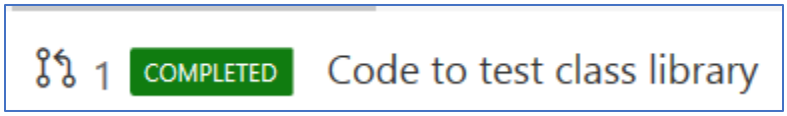
9. If possible login with another user who has been added as reviewer and provide comment.



10. Approve the request and switch back to original team member.

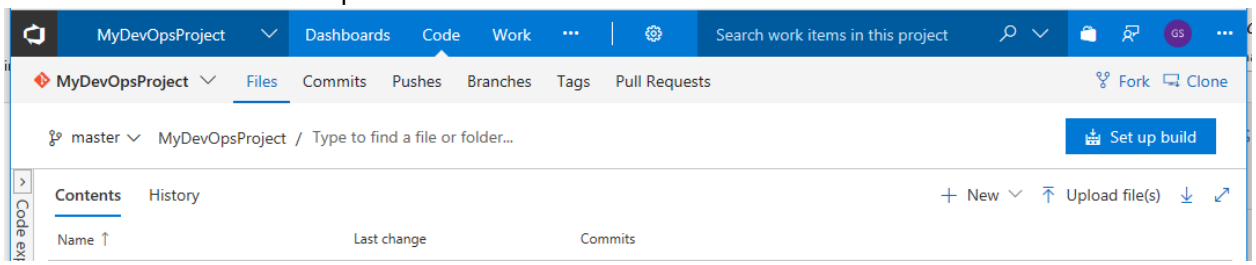


11. Click on Complete Pull request but do not delete branch.

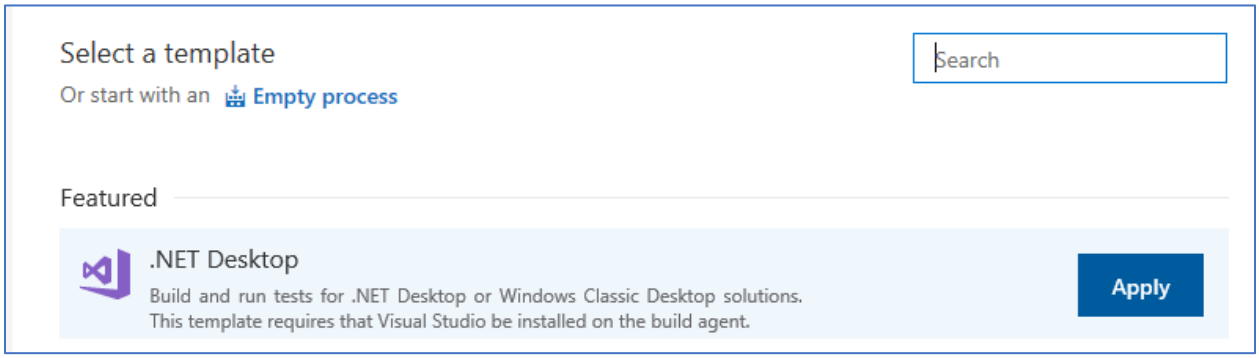
12. 

13. Verify that both the branches have same code.

14. Select Code – Files – Set up Build



15. Select the template of .NET desktop and click on Apply

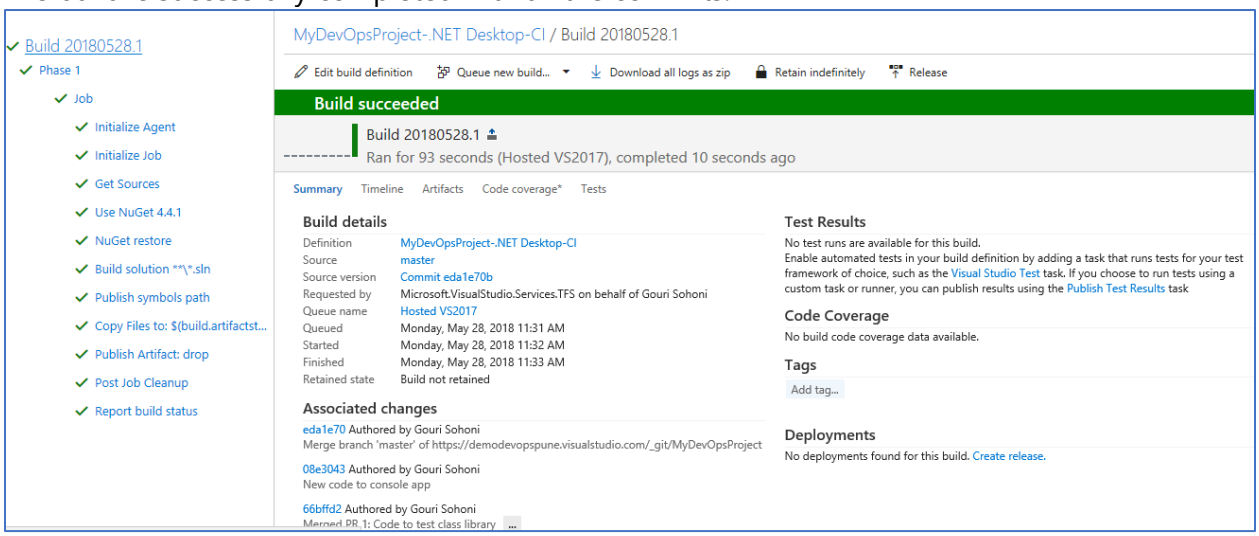


16. All required build tasks will be automatically added. Right click on the task for VsTest – Assemblies and disable it. Keep all with their default. (We will enable and use it later)
17. Click on Triggers and enable the trigger for Continuous Integration. Ensure that master branch is selected.
18. Save the build definition, do not Queue it yet.
19. Go back to Visual Studio 2017 and add another statement to the console application to execute the other method. The complete code looks as below

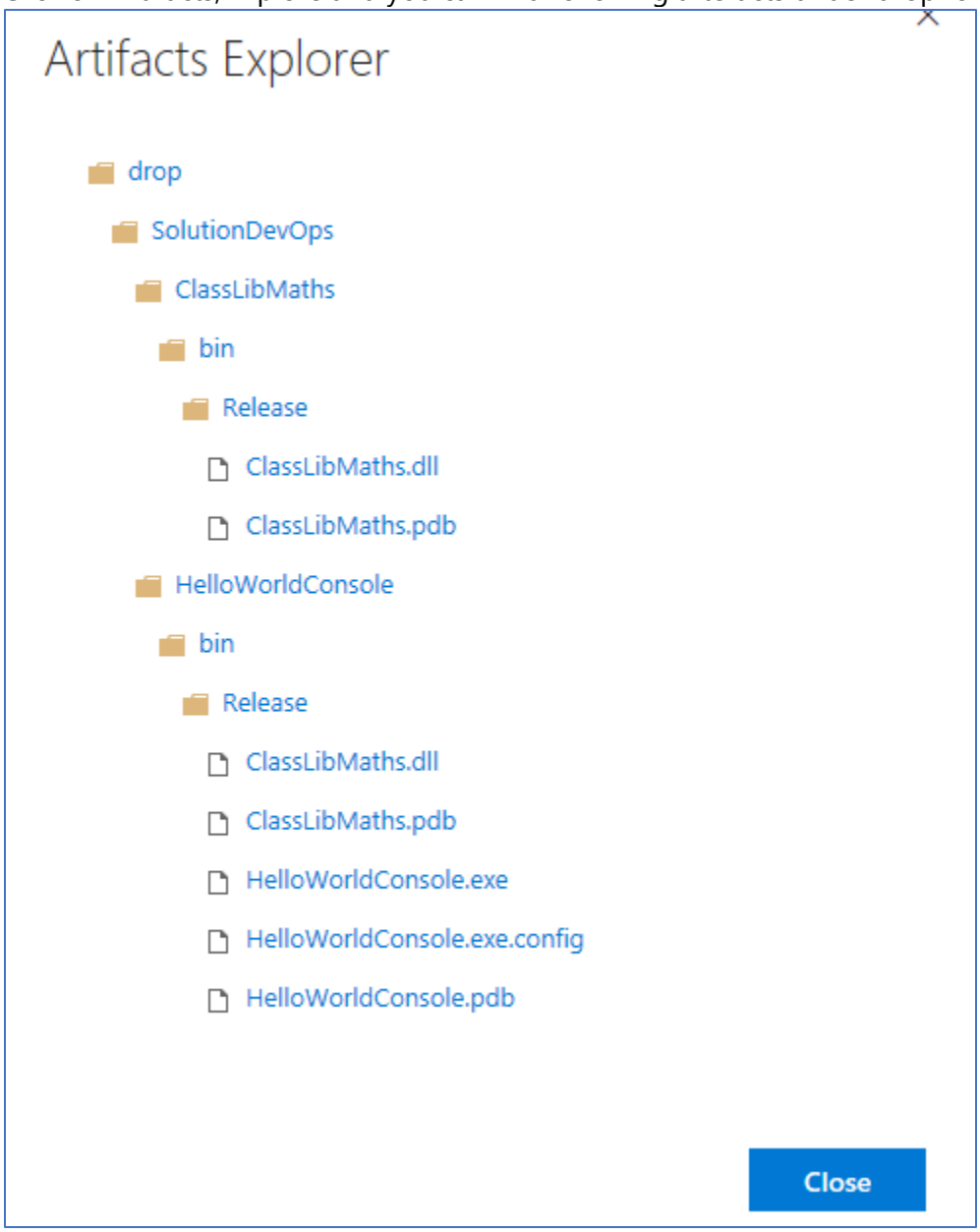
```
static void Main(string[] args)
{
    Console.WriteLine("Hello World!");
    MathsCls obj = new MathsCls();
    Console.WriteLine("Sum is: " + obj.Sum(10, 24));
    Console.WriteLine("Subtraction is: " + obj.Subt(90, 35));
    Console.ReadLine();
}
```

20. Create the desktop build to ensure there are no compilation errors. save the code, go to changes in Team Explorer and directly Commit and Push the code after providing comment.
21. Select Build tab and you will observe that the build is not Queued. This is because we are working in another branch in Visual Studio.
22. Make sure that master is selected, merge the 2 branches and click on push.
23. Now you can see that the build is automatically triggered. It is using the hosted agent.

24. The build is successfully completed with all the commits.

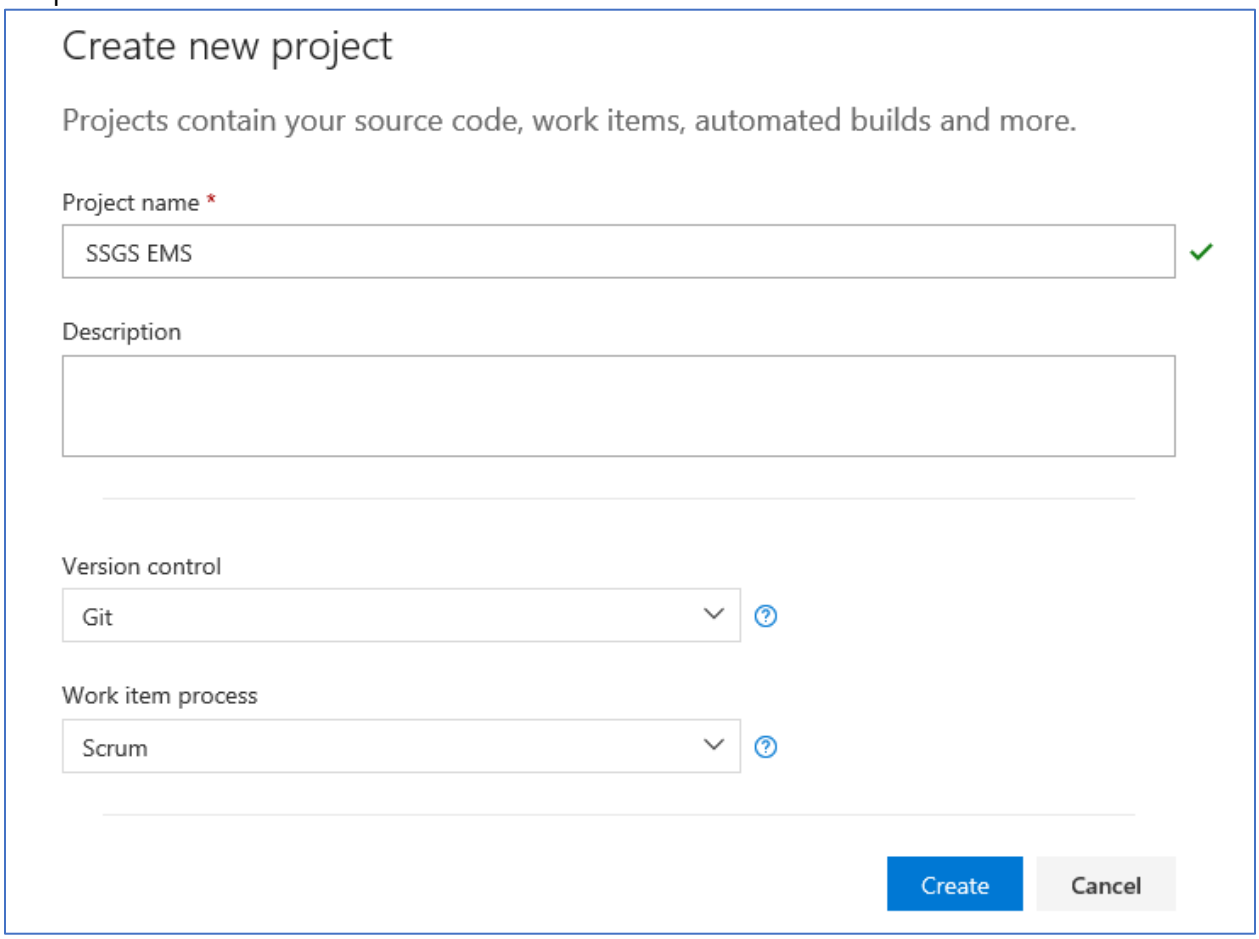


25. Click on Artifacts, Explore and you can find following artefacts under drop folder



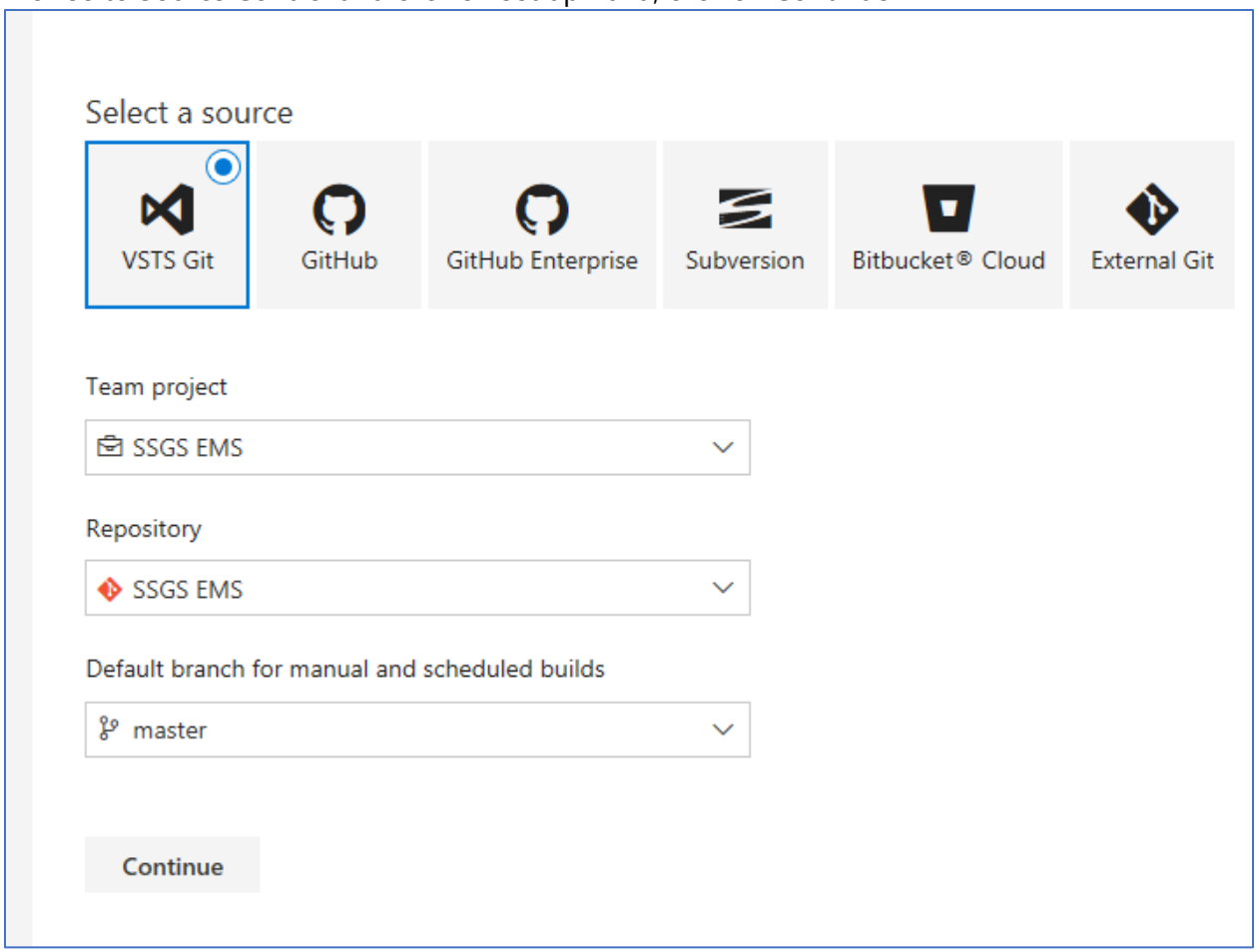
Exercise 4: Create Web App Service using Azure Portal, deploy web app to it

1. Create another Team Project in VSTS account with Git as source control, scrum as process template.

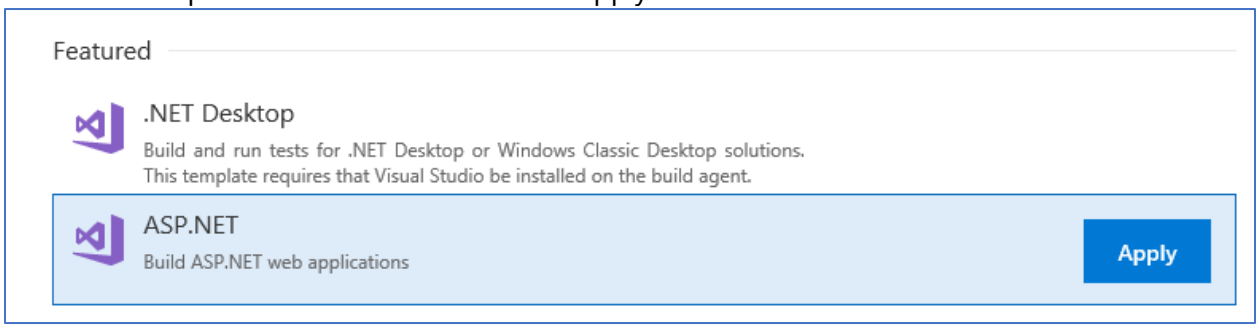


2. Connect to this Team Project from Visual Studio, clone the project. Create a new blank solution provide appropriate name
3. Add a New project to the solution of type Web application, select Web Forms and click on Ok
4. Keep all default pages, view the web app in browser, commit and push after giving comment

5. Browse to Source Control and click on set up Build, click on Continue

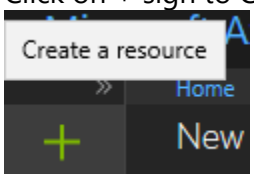


6. Select the template of ASP.NET and click on Apply

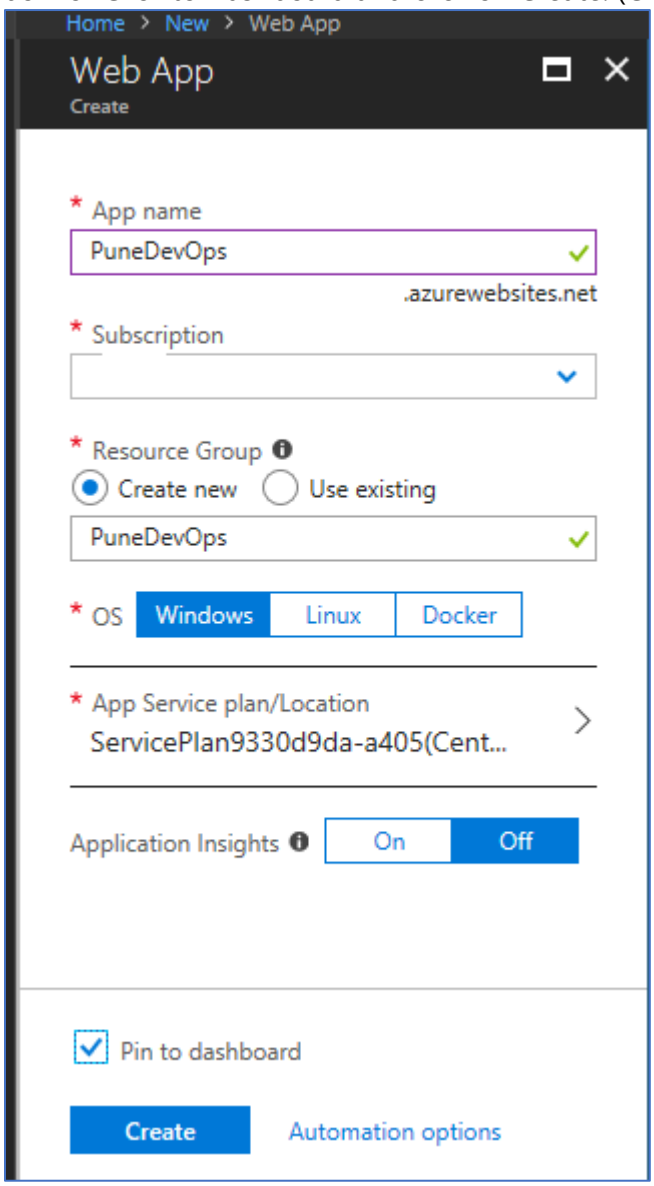


7. Disable testing assembly and select task for Build Solution. Observe how arguments are given. Make sure that CI is triggered.
8. Save and Queue the build. Observe drop folder from artifacts after build succeeds. (You will see zip file created)
9. Let us create Web App Service using Azure Portal
10. Login to azure account using http://portal.azure.com

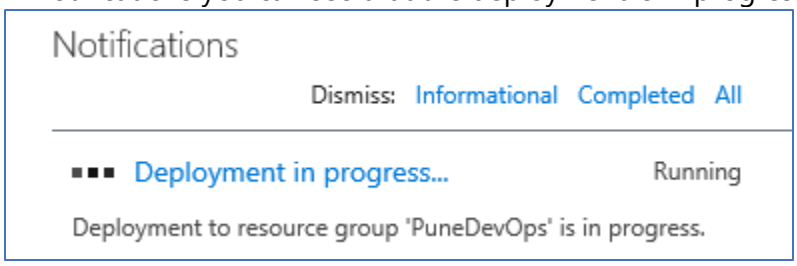
11. Click on + sign to Create a resource



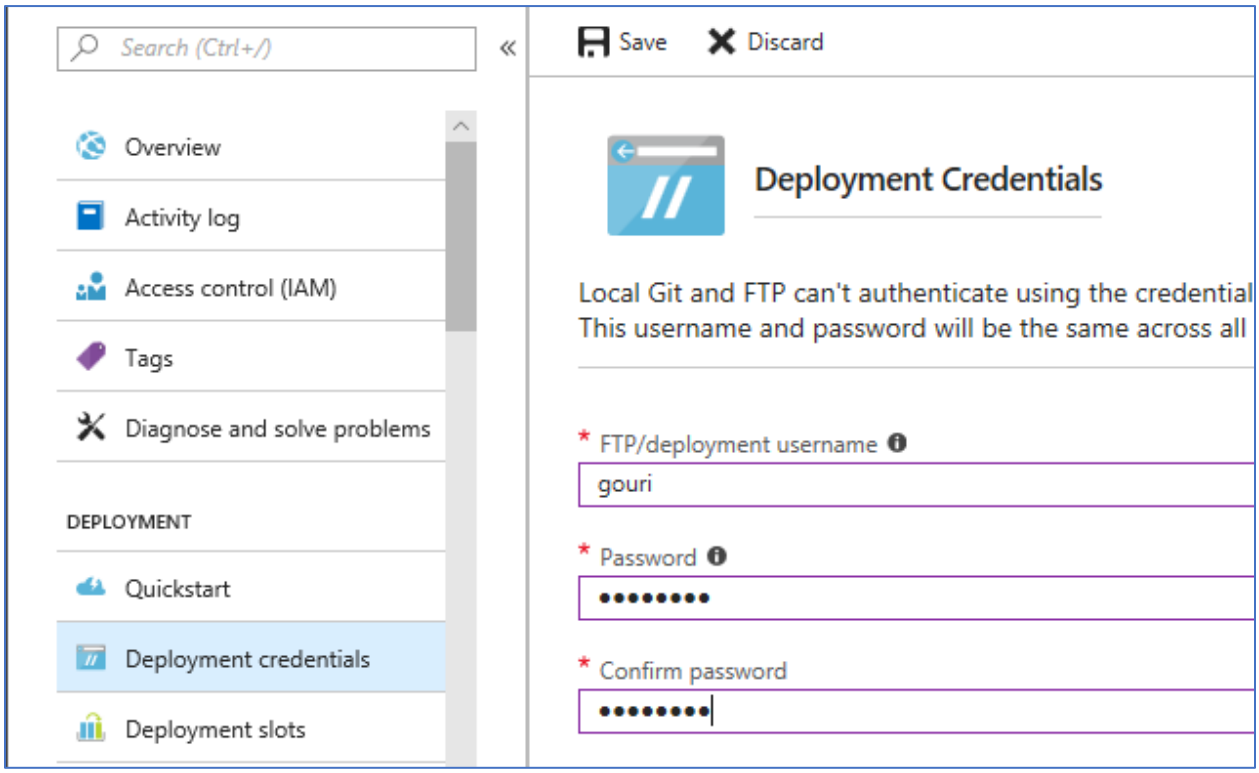
12. Select Web App, provide name to app, select subscription (if you have multiple), select check box for Click to Dashboard and click on Create. (Create New Resource Group)



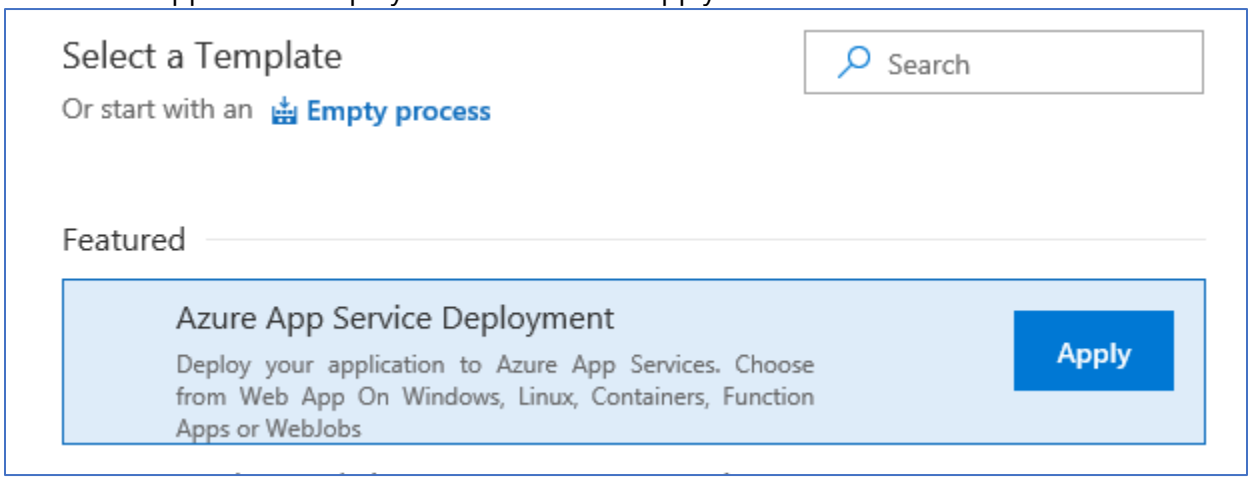
In notifications you can see that the deployment is in progress.



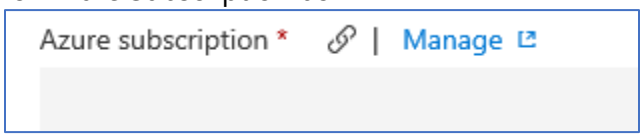
13. Click on Go to Resource and close the notifications.
14. Select the option for Deployment Credentials and provide user name and password of your choice and click on Save



15. Select Releases tab and click on New Definition in VSTS account, select the template of Azure Web App Service Deployment and click on Apply

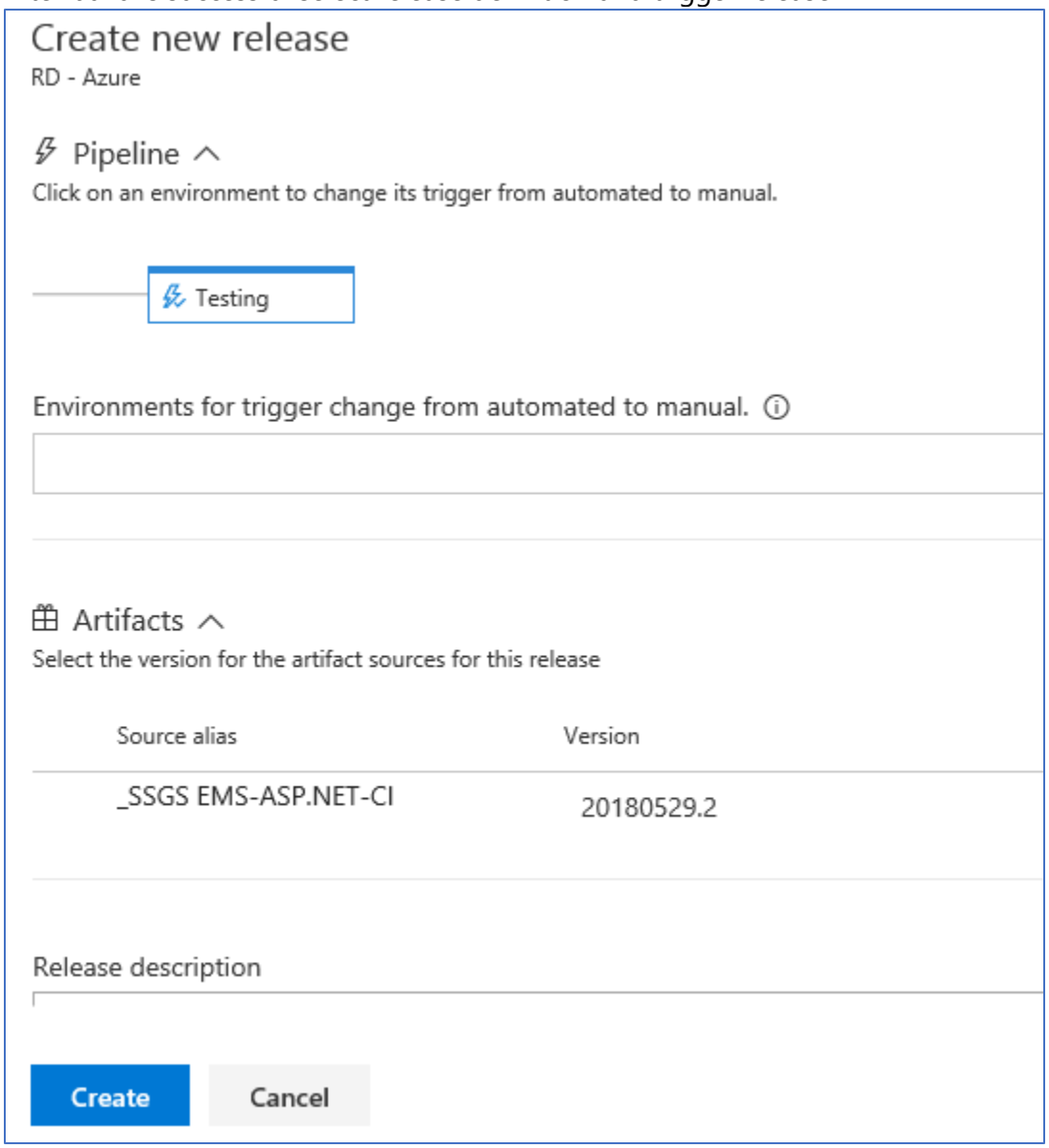


16. Provide proper name for the environment. Click on artifacts add, select the previously created build and click on Add
17. Click on environment, select the task added for Deploy Azure App Service, click on Manage for Azure Subscription box

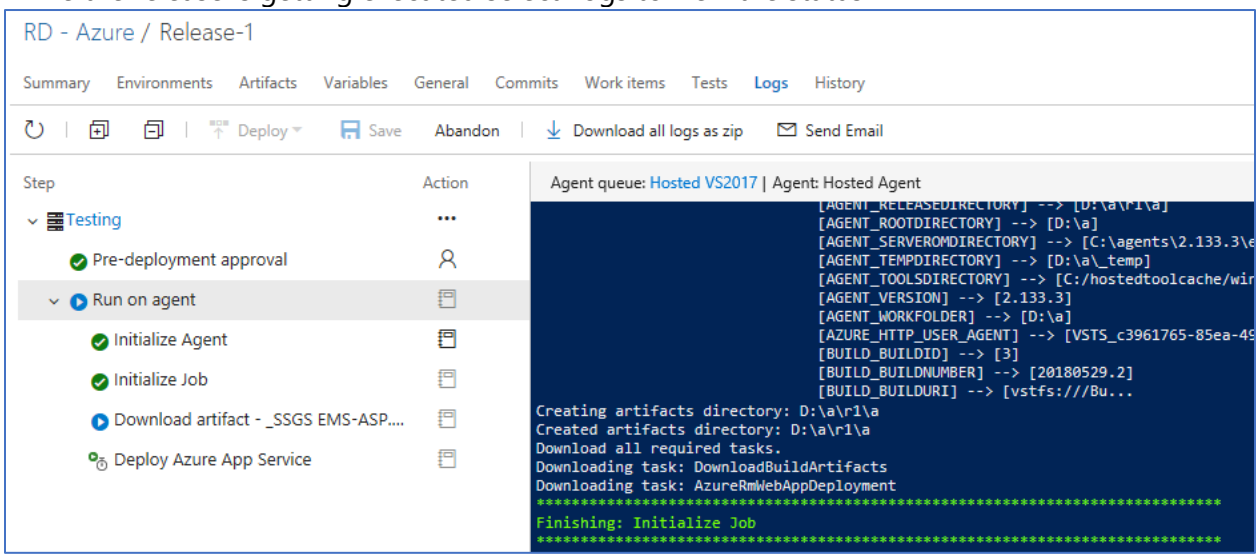


18. Create a new Azure Resource Manager Endpoint or use your Azure Subscription, click on Authorize.
19. Click on refresh button for Web Service Name after authorization is done and select the Web App Service.
20. Save the Release Definition after providing name. Let us go back to Visual Studio and do some changes to the application.
21. Commit and push all changes and observe the build getting triggered.

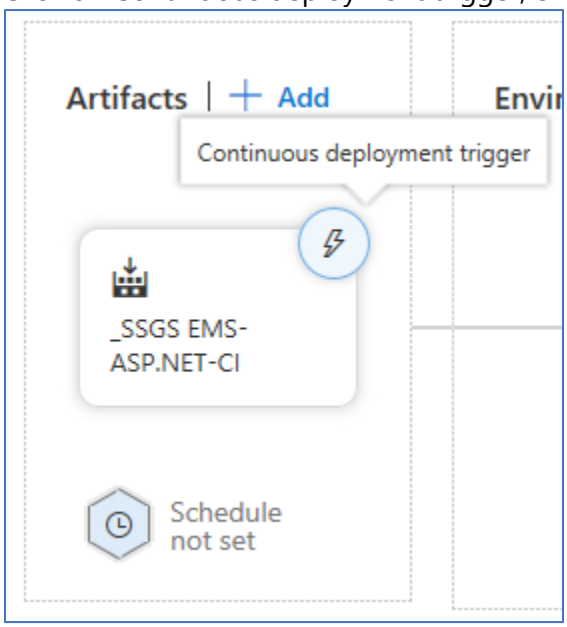
22. After build is successful select release definition and trigger release



23. While the release is getting executed select Logs to view the status



24. After successful release go to azure portal and click on the URL for Web App Service (you should the changed contents appearing)
25. Click on Continuous deployment trigger, enable it, save release definition

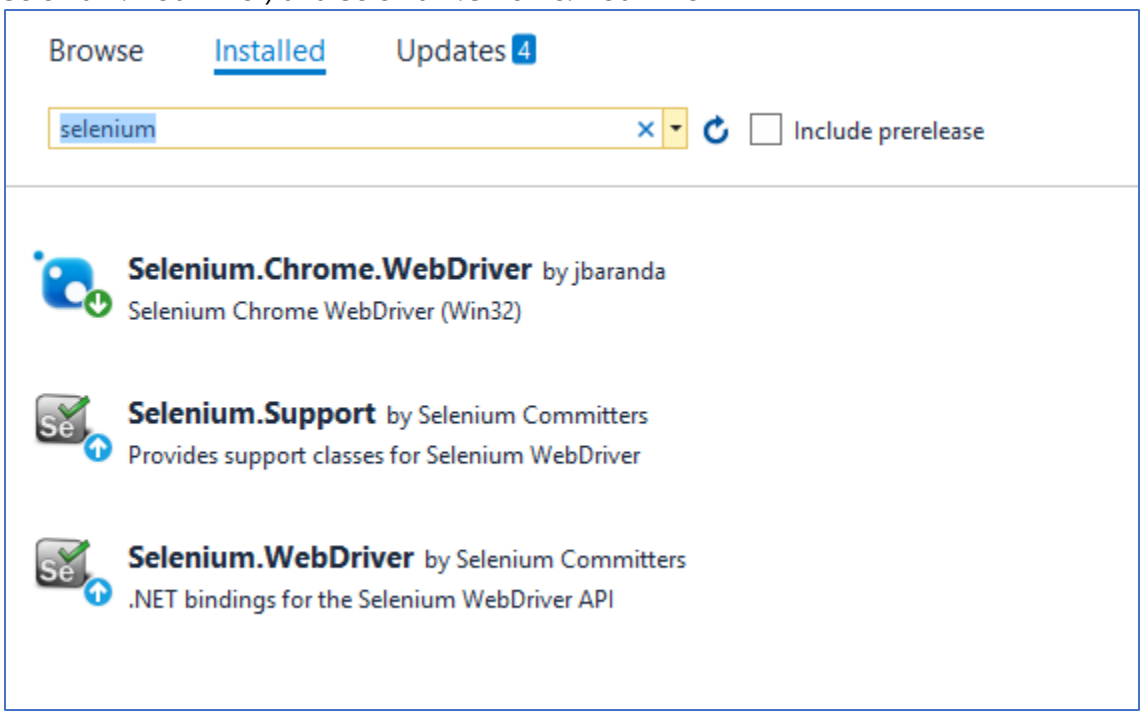


26. In order to observe the complete CI CD process, go and do some changes in web app, commit and push and observe build as well as release getting triggered automatically.

Exercise 5: Complete the cycle with CT (Continuous Testing)

1. Open the solution using Visual Studio 2017 for Web Application.
2. Add a Unit Test Project to the solution, provide proper name. (Please keep 'test' in name of the project)
3. Right click on project and select Manage NuGet Packages. Click on Browse tab, search for Selenium and add packages for Selenium.Support (this automatically adds support to

Selenium.WebDriver) and Selenium.Chrome.WebDriver



4. Add following using statements

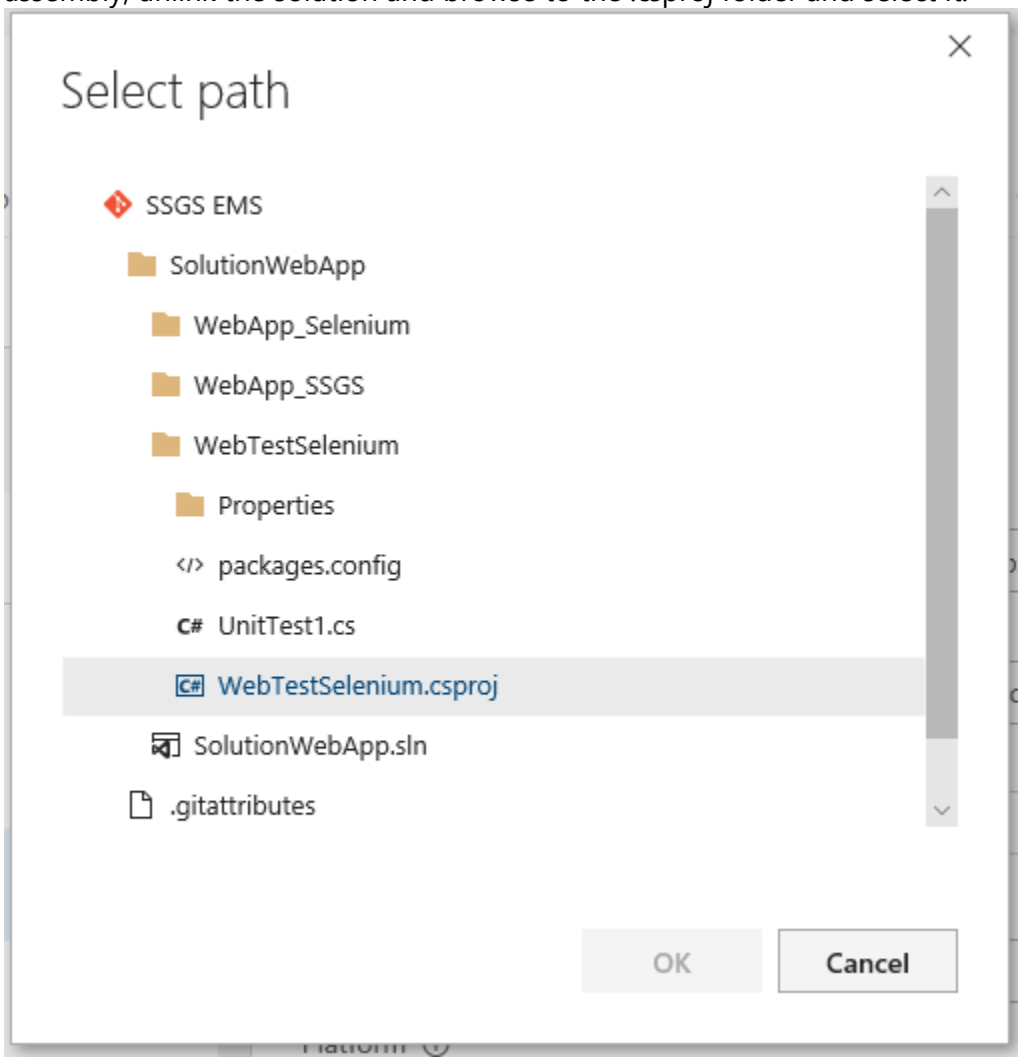
```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
```

5. We will just add code to view the site for 5 seconds, if you want you can add functionality for clicking on links and other details (the purpose of this is to find out how to automatically test after deployment to Azure Web Service and not Selenium coding). This code assumes you are having Chrome browser installed if you want to run the test locally

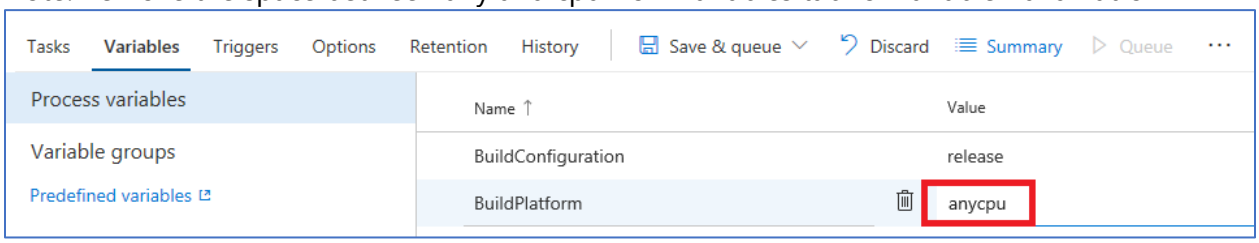
```
[TestClass]
public class UnitTest1
{
    [TestMethod]
    public void TestSSGS()
    {
        IWebDriver driver = null;
        driver = new ChromeDriver();
        driver.Manage().Window.Maximize();
        driver.Navigate().GoToUrl("https://punedevops.azurewebsites.net");
        System.Threading.Thread.Sleep(5000);
        driver.Quit();
    }
}
```

6. Build the application and right click on the test name in Test Explorer and select Run selected test to run the test locally.

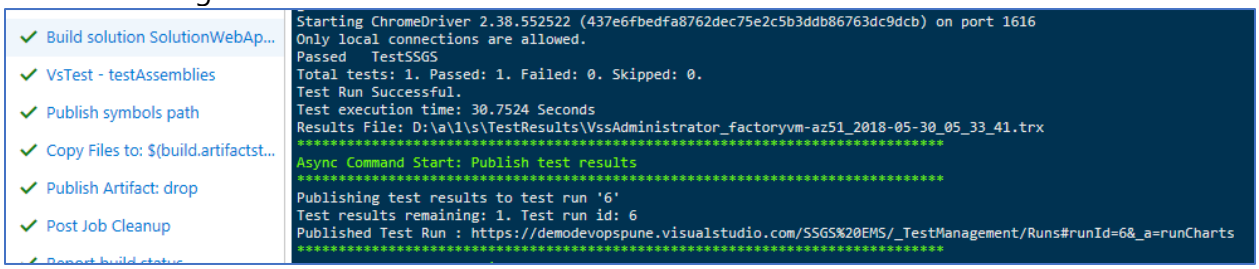
7. As we had selected the ASP.NET template if we enable the test assembly task the test will get executed but we will not get test assembly in drop folder. You can add a task for doing the same but I also want to stress how multiple build artifacts can be used with a single Release Definition so we will create a new Build definition. You can disable the CI trigger for previous build for some time.
8. Commit and push the test project and click on New Build definition. Select .NET Desktop as the template, keep all default task except for Build Solution. As we want to create only build assembly, unlink the solution and browse to the .csproj folder and select it.



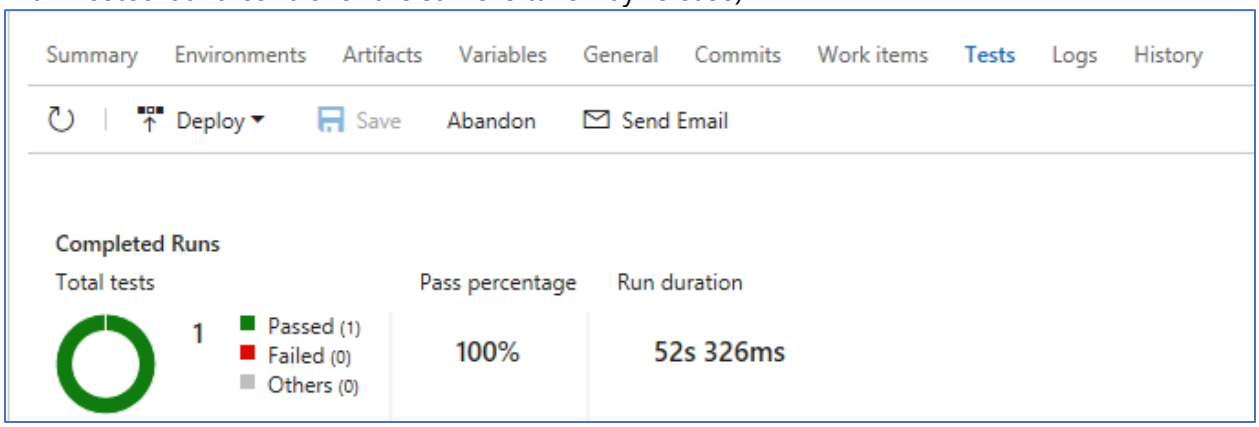
Note: Remove the space between any and cpu from variables tab for variable BuildPlatform



9. Trigger the build and observe that the test gets executed but we cannot see as we are using Hosted Build Agent.



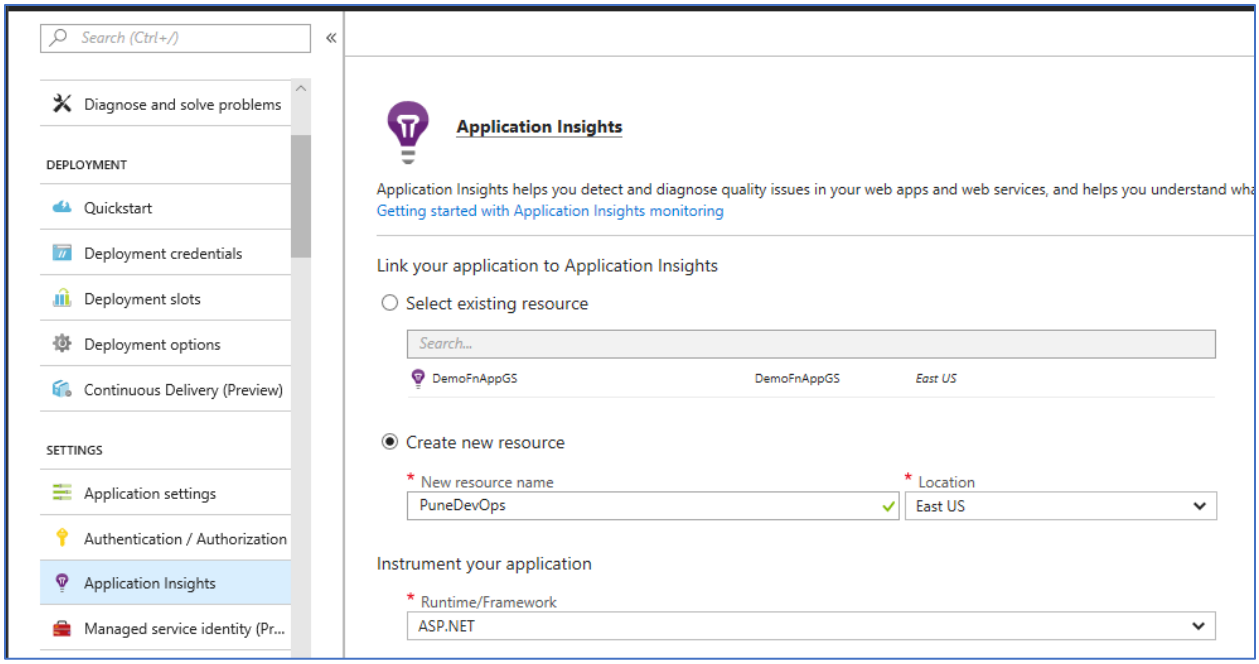
10. Edit the previous release definition. As there is no change in the web site contents we can temporarily disable the task of azure deployment. Add another task of "VsTest – testAssemblies".
11. Add another artifact to the new build definition by selecting Pipeline tab. Save and create release.
12. You can view that the test gets executed and the result can be seen as follows (if the build is with Hosted build controller the same is taken by release)



13. Reapply the trigger of CI for build definition, enable the task for Azure deployment in Release Definition , do some change in site contents from Visual Studio, commit and push the changes and observe the complete cycle of CI-CD-CT (Continuous Integration – Continuous Deployment – Continuous Testing)

Exercise 6: Application Monitoring (Continuous Monitoring)

1. Go to Azure Portal and select the Web Service. Select the tab for Application Insights



Provide the resource and click on Ok.

2. You can also configure App Insights from Visual Studio. Right click on the web application and select Configure Application Insight Settings

Application Insights Configuration | UnitTest1.cs | Contact.aspx | Default.aspx | WebApp_SSG

SDK

A new version of Microsoft.ApplicationInsights.Web (v2.4.1) is available. Your project is using v2.2.0.

Update SDK

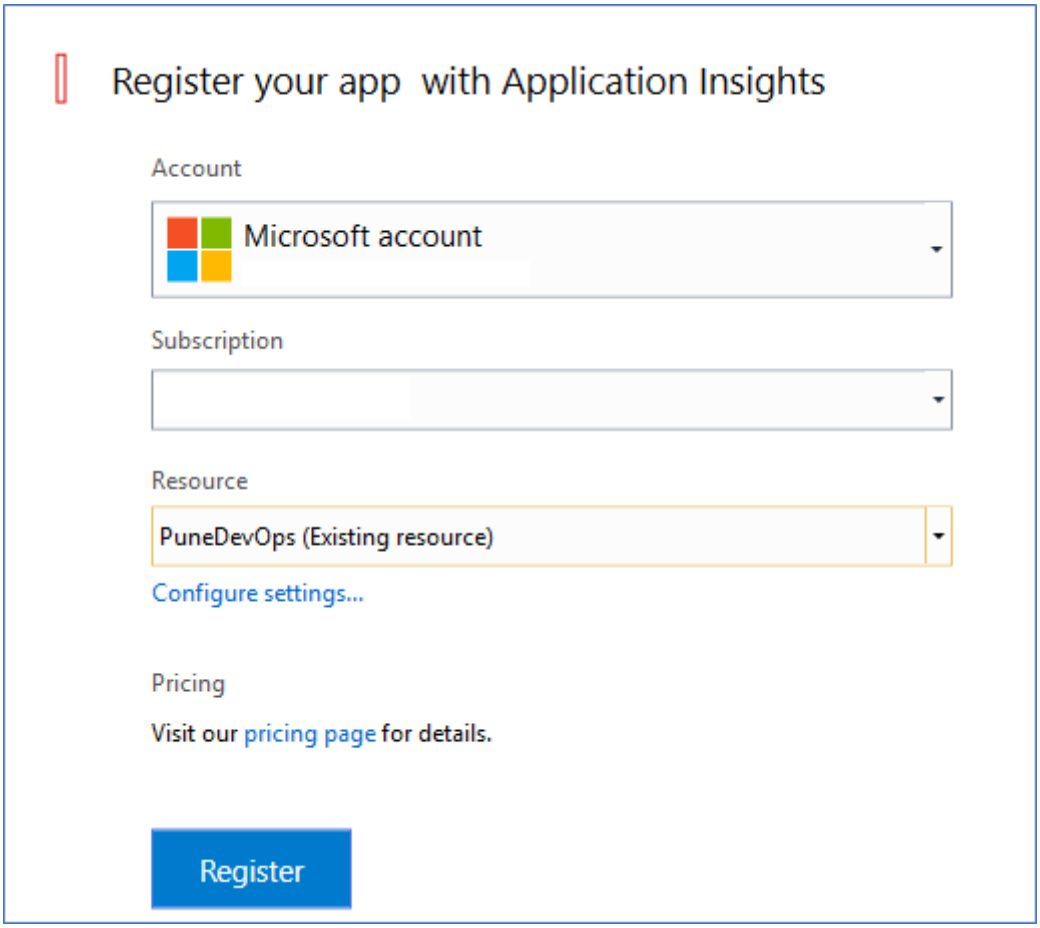
Gain insights through telemetry, analytics and smart detection

Detect
and diagnose exceptions and application performance issues

Monitor
websites on Azure, hosted containers, on-premises and with other cloud providers

Integrate
with your DevOps pipeline using Visual Studio, VSTS, GitHub, and web hooks

Get Started



You can provide the same resource which you have created in Web App
You can observe similar data as follows for App Insights