Continuous Integration with Visual Studio Team Services

Lab Tasks

Task 1: Create a Team Project

1. Go to visual Studio Team Services account.
2. Click on <New Project>.
   - Add the Project name.
   - Add a Description[1].
   - Select <Git> in Version control.
   - Select Agile in Work item process.
   - Click on <Create>.
3. It will redirect to the new project Dashboard.

Task 2: Create Web Application in VS[2]

1. Create a New project.
   - Select File -> New -> Project.
2. In the new project wizard.
   - Select File -> New -> Project.
   - Go to Installed -> Visual C# -> Web.
   - Select Web Application ASP.NET (.NET Framework).
   - Add project name.
   - Select path.
   - Add solution name.
   - Click on <Accept>.
3. In the New Application wizard.
   - Select Empty template.
   - Select MVC in the Add folders and core references section.
   - Click on <OK>.
4. Create Home page.
   - Right click on Controllers folder from the Solution Explorer.
   - Click on Add -> Controller.
   - Select <MVC 5 Controller - Empty>.
   - Click on <Add>.
   - Add the controller name (HomeController).
   - Click on <Add>.
   - Right click on the Views -> Home folder.
   - Click on Add -> View...
   - Add the View name (Index).
   - Select Empty (without model) Template.
   - Check <Use a layout page:>.
   - Click <Add>.
   - Replace the new file created with the next lines of code:

```
@{
    ViewBag.Title = "Dev Ops";
}
<h2>Index</h2>
<div class="jumbotron">
```

```
  <h1>DevOps development</h1>
</div>
```

- Save the Index view.
- Open the Views -> Shared -> _Layout.cshtml partial view.
- Add the next line, after the <ul class="nav navbar-nav">.

```
<li>@Html.ActionLink("Home", "Index", "Home")</li>
```

- Change the tittle of the application, replace the next lines.
  - Line 6.

```
<title>@ViewBag.Title – DevOps Application</title>
```

  - Line 20.

```
@Html.ActionLink("DevOps development", "Index", "Home", new { area = "" }, new { @class =
"navbar-brand" })
```

  - Line 34.

```
<p>&copy; @DateTime.Now.Year – DevOps Application</p>
```

- Save the _layout partial view.

5. Test application
   - Right click on the Solution from the Solution Explorer.
   - Click on <Build Solution>.
   - Once the Solution is builded, click on Debug-> Start Debugging menu or press or F5.
   - Check the Web Application runs without problem.
   - Close the internet Explorer.

Task 3: Import Source Code into your VSTS Account

1. Select GIT as source control.
   - On the Tools menu
   - Click in Options.
   - In the Options dialog box, select Source Control.
   - Select <GIT> from the Current source control plug-in list.
   - Click on <Accept>.
2. Add solution to Source Control.
   - In Solution Explorer.
   - Right-click the solution
   - Click on <Add Solution to Source Control>.
3. Connect to GitHub.
   - Go to Team Explorer.
   - Click on <Sync>.
   - Click on <Publish to GitHub> in the Publish to GitHub section.
   - Login with your GitHub account.
   - Click on Publish.
4. Commit code to GitHub.
   - Click on <Changes>.
   - Add a message.
   - Select <Commit All and Push>.
5. Upload code from GitHub to VSTS.
   - In the VSTS project Dashboard.

   - Select the option <or import a repository>.
   - Click on <Import>.
   - Select <Git> on Source type.
   - Add the project URL.
   - Click on <Import>.
6. At this time, you will have your code in GitHub and VSTS.
7. Explore Commits.
   - Click on Code -> Commits menu.
   - Click on one message from the list.
   - You will see the list of files with changes.
8. Explore Pushes.
   - Click on Code -> Pushes menu.
   - You will see the list of publishes with the commits in each one.

Task 4: Create Work Items

1. Create Epic.
   - Click on Work -> Work Items.
   - Click on New Work Item -> Epic.
   - In the new window, provide the details.
     - Title.
     - Assigned to.
     - Description.
     - The rest of the fields are optional.
     - Click on <Save>.
   - Save the ID for future usage.
2. Create Feature.
   - Click on Work -> Work Items.
   - Click on New Work Item -> Feature.
   - In the new window, provide the details.
     - Title.
     - Assigned to.
     - Description.
     - In Related Work.
       - Click Add link -> Existing Item.
         - Link type: Child.
         - Work item to link: Enter the Epic Id.
         - Comment: Optional.
         - Click on <OK>.
     - The rest of the fields are optional.
   - Click on <Save>.
   - Save the ID for future usage.
3. Create User Story.
   - Click on Work-> Work Items.

- Click on New Work Item -> User Story.
- In the new window, provide the details.
  - Title.
  - Assigned to.
  - Description.
  - In Related Work.
    - Click Add link -> Existing Item.
      - Link type: Child.
      - Work item to link: Enter the Feature Id.
      - Comment: Optional.
      - Click on <OK>.
  - The rest of the fields are optional.
- Click on <Save>.

4. Create Task.
   - Go to Work -> Backlogs.
   - Click on <+> beside the User Story created previously.
   - Click on <Task>.
   - In the new window, provide the details.
     - Title.
     - Assigned to.
     - Description.
     - Assign Original Estimate in the Effort (Hours).
     - The rest of the fields are optional.
     - Check the Task is linked to the user story.
     - Click Save & Close.

Task 5. Create Continuous Integration Build

1. Click on Pipelines -> Builds at the top menu of the page.
2. Create a new pipeline.
   - Click on <New pipeline>.
   - Under the Select sources you need to make sure to select Azure Repos Git.
   - In the Team project select the one you created.
   - In the Repository select the project you created.
   - In the Default branch, select Master.
   - Click on <Continue>.
3. Select the <Empty pipeline> template.
   - Click on <Apply>.
4. Once the new empty build has been created.
   - Modify the Name[1].
   - Select <Hosted VS2017> in the Agent pool.
5. To start adding tasks.
   - Click on <+> beside the Agent job1 on the left panel.
6. Add tasks.
   - Select the Tool tab.

- Select <NuGet Tool Installer> task.
- Click on <Add>.
- Select the Package tab.
- Select <NuGet> task.
- Click on <Add>.
- Select the Build tab.
- Select <Visual Studio Build> task.
- Click on <Add>.
- Select <Index Sources & Publish symbols> task.
- Click on <Add>.
- Select the Utility tab.
- Select <Publish Build Artifacts> task.
- Click on <Add>.

7. Modify <NuGet restore> task.
   - Select <NuGet restore> task from the left panel.
   - Click on <… > from the Path to solution.
   - Select the .sln solution.
   - Click on <OK>.
8. Modify <Build solucion ***.sln> task.
   - Select <Build solucion ***.sln> task from the left panel.
   - Click on <…> from Solution.
   - Select the .sln solution.
   - Click on <OK>.
   - Add the next arguments in MSBuild Arguments

/p:DeployOnBuild=true /p:WebPublishMethod=Package /p:PackageAsSingleFile=true
/p:SkipInvalidConfigurations=true
/p:DesktopBuildPackageLocation="$(build.artifactstagingdirectory)\WebApp.zip"
/p:DeployIisAppPath="Default Web Site"

   - Add the next variable in Platform

$(BuildPlatform)

   - Add the next variable in Configuration

$(BuildConfiguration)

9. Modify <Publish symbols path> task.
   - Disable Publish symbols
   - Disable Continue on error, on the Control Options section.
10. Modify <Publish Artifact> task.
    - Change the Path to publish.

$(build.ArtifactStagingDirectory)

    - Change the Artifact name to <drop>.
11. Add variables.
    - Click on the Variables tab.
    - Click on <+Add>.
      - Name: BuildConfiguration.
      - Value: release.
      - Settable at queue time: enabled.
    - Click on <+Add>.

        ○ Name: BuildPlatform.
        ○ Value: any cpu.
        ○ Settable at queue time: enabled.
    - Enable the Settable at queue time to the variable system.debug

12. Modify triggers.
    - Click on the Triggers tab.
    - Check Enable continuous integration (CI) option for your project to build the solution every time a change is pushed.
    - Make sure the filter includes the appropriate branch, in this case master.
    - Uncheck Batch changes while a build is in progress option.
13. Click <Save & queue> from the top menu of the Pipeline.
14. Click on <Save> from the list.
    - Add a Comment.
    - Click on <Save>.

Task 6: Test the CI trigger in VSTS

1. Go to your solution in VS.
2. Navigate to SolutionName -> ProjectName -> Controllers.
3. Open HomeController.cs.
4. Add after the using section a comment (Example: //CI Test).
5. Save the file (File->Save).
6. In the Solution Explorer right click on the solution.
7. Click on <Commit…>.
8. In the Team Explorer add a comment.
9. Make sure the controller where added the comment is listed in Changes.
10. Click on <Commit All>.
11. From the home in the Team Explorer or the commit message.
    - Click on <Sync> to upload the changes in the server.
    - Click on <Push> from the Outgoing commits list.
12. Go back to VSTS.
    - Click on Pipelines -> Build menu.
    - The previous push should have triggered the build we previously created.
13. Click on <…> from the new build that has been created.
    - Click on <View build results> from the list.
    - You should see progress of the build.
    - Here you can also see the commands being logged to console and the current steps that the build is on.
14. You can see in the Artifacts menu, there is the drop folder with the output configured (the build project).

[1]Optional.

[2]Optional, you can use the application created in <Developing Microsoft Azure Infrastructure Solutions> Module 3.

References

- https://msdn.microsoft.com/en-us/library/ms181374(v=vs.80).aspx
- http://microsoft.github.io/PartsUnlimited/cicd/200.3x-CICD-M01-CIwithVSTS.html