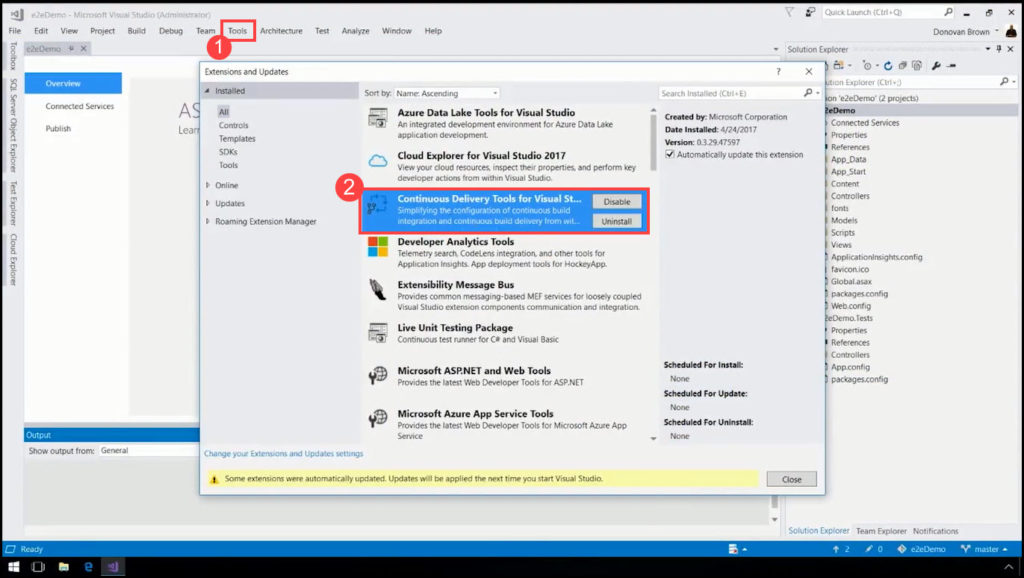
**Step1: Enable the Continuous Delivery Extension for Visual Studio**

In order to use Continuous Delivery Tools for Visual Studio extension you just need to enable it. The Continuous Delivery Tools for Visual Studio extension makes it simple to automate and stay up to date on your [DevOps](https://www.visualstudio.com/learn/what-is-devops/) pipeline for and other projects targeting Azure. The tools also allow you to improve your code quality and security.

1. Go to **Tools**, choose **Extensions and Updates**.
2. From the prompted window, select **Continuous Delivery Tools for Visual Studio** and click **Enable**.

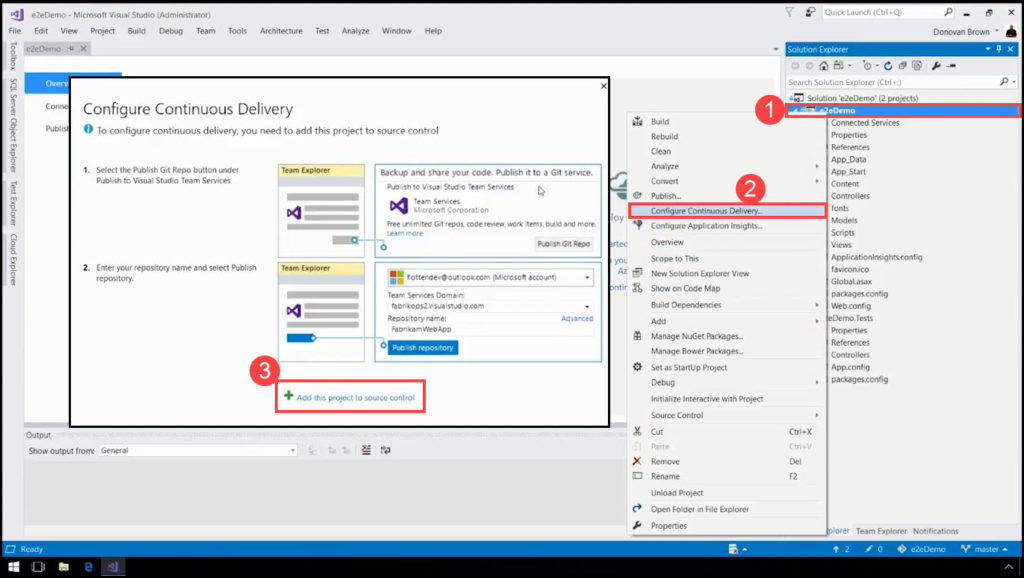
\*If you don't have installed Continuous Delivery Tools go to [**Online Visual Studio Marketplace**](https://marketplace.visualstudio.com/), search for "Continuous" and download it.

[](http://mohamedradwan.com/wp-content/uploads/2017/12/1-Enable-Continuous-Delivery-Tools-for-Visual-Studio.jpg)

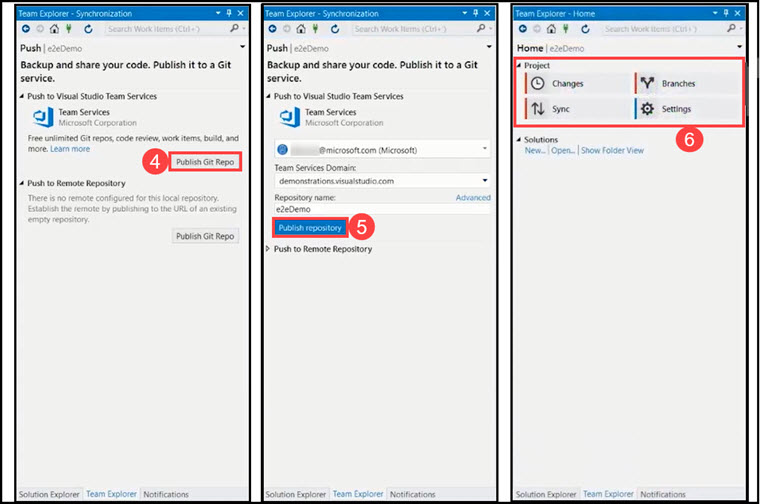
**Step 2: Create a Project in Team Services**

In this step, you are going to create a project in [Team Services](http://mohamedradwan.com/2016/07/13/release-management-overview-for-tfs-and-vsts/) and put your project code there without leaving your IDE. Team Services is a tool that allows you to build a continuous integration and continuous delivery.

1. Go in the **Solution Explorer**, right-click on your web-based project.
2. Click on the new context menu **Configure Continuous Delivery.**
3. A new window is displayed **Configure Continuous Delivery.**Click on the **Add this project to source control** plus button.

[](http://mohamedradwan.com/wp-content/uploads/2017/12/2.1-Add-project-to-Source-Control-and-Configure-Continuous-Delivery.jpg)

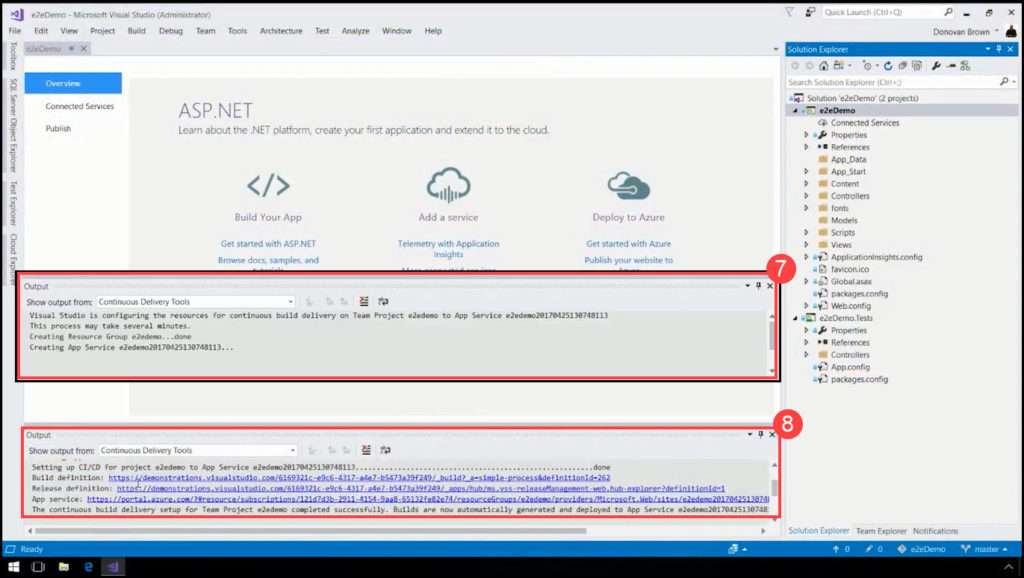
1. Click on the **Publish Git Repo** button located in the **Publish to Visual Studio Team Services**section in **Team Explorer.**
2. Your **Microsoft Account** is automatically fetched from your IDE. Also is displayed the**Team Services Domain** which will be used and your **Repository Name**. Click on the **Publish Repository**button in order to create a project in **Team Services**.
3. After the synchronization is finished you will see that your project is created in the Team Explorer.

[](http://mohamedradwan.com/wp-content/uploads/2017/12/2.2-Create-project-in-Team-Services.jpg)

Now your project is created into Team Services account (the source code is uploaded, there is a Git Repository and it is generating continuous delivery pipeline automatically).

1. In the output window, you can see that your **CI/CD** are setting up for your project.
2. After a while, you are going to get 3 different links:

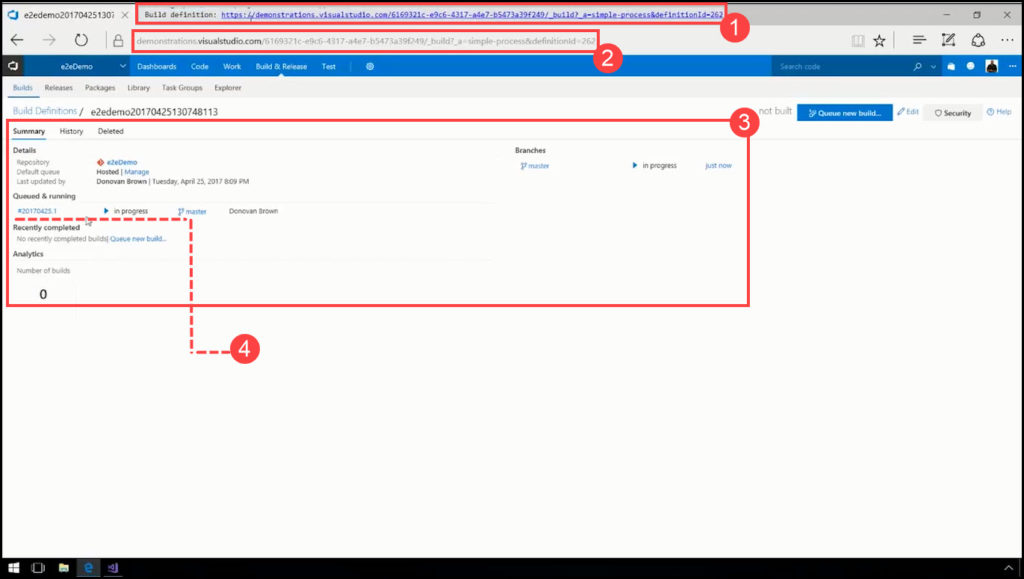
* Link to the **build.**
* Link to the **release.**
* Link to the assets created in **Azure** which is going to be the target for your deployment. (application service).

[](http://mohamedradwan.com/wp-content/uploads/2017/12/2.3-Linking-Continiuos-Delivery-to-the-Build-Release-and-Azure-Assets.jpg)

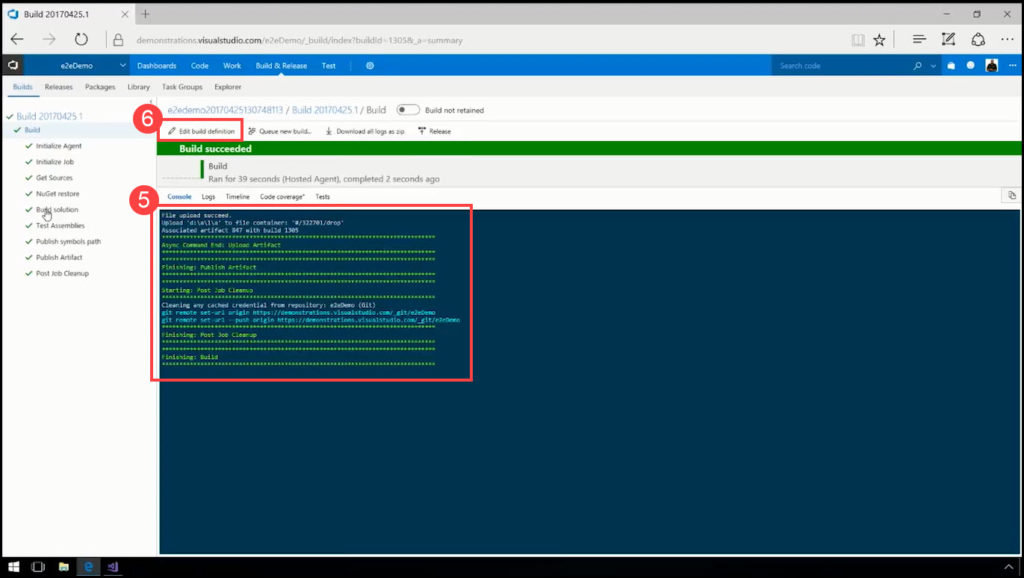
**Step 3: Open the Project in Team Services**

A [**build**](http://mohamedradwan.com/2015/06/16/changing-the-connection-string-during-the-tfs-build-for-the-publish-profile-of-the-ssdt-sql-server-data-tool/)**definition** is the entity through which you define your automated build process. In the build definition, you compose a set of tasks, each of which performs a step in your build.

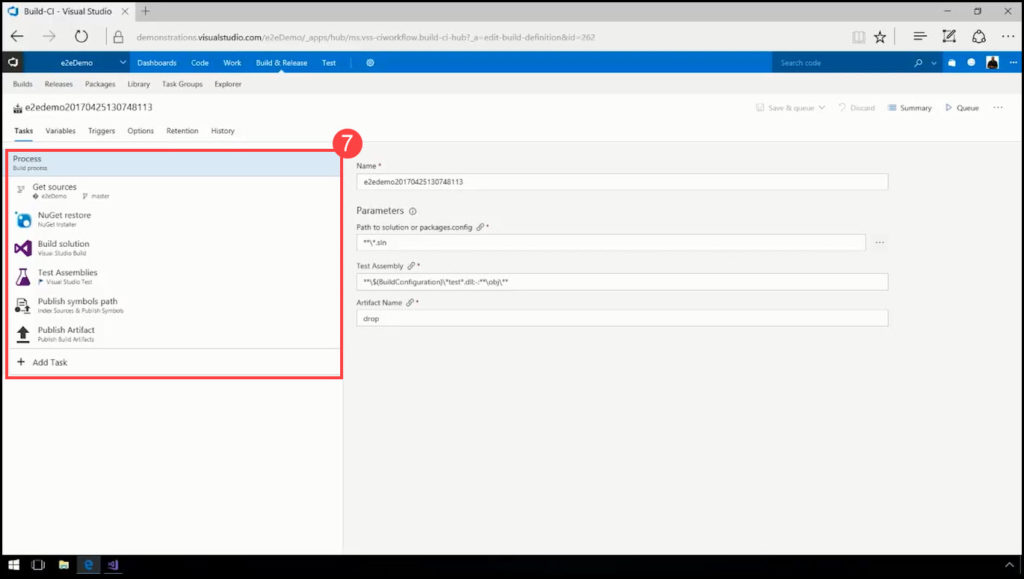
1. Choose the **Build Definition** link provided in Output window and copy.
2. Paste it into a browser in order to open the project containing your application in Team Services.
3. The summary for the build definition is displayed. You can see that the build is already running.
4. Click on the build link.

[](http://mohamedradwan.com/wp-content/uploads/2017/12/3.1-Build-definition-summary-in-Team-Services.jpg)

1. It is shown an output of your build server which is running your build automatically.
2. Click on the **Edit build definition.**

[](http://mohamedradwan.com/wp-content/uploads/2017/12/3.2-Editing-Build-Definition-in-Team-Services.jpg)

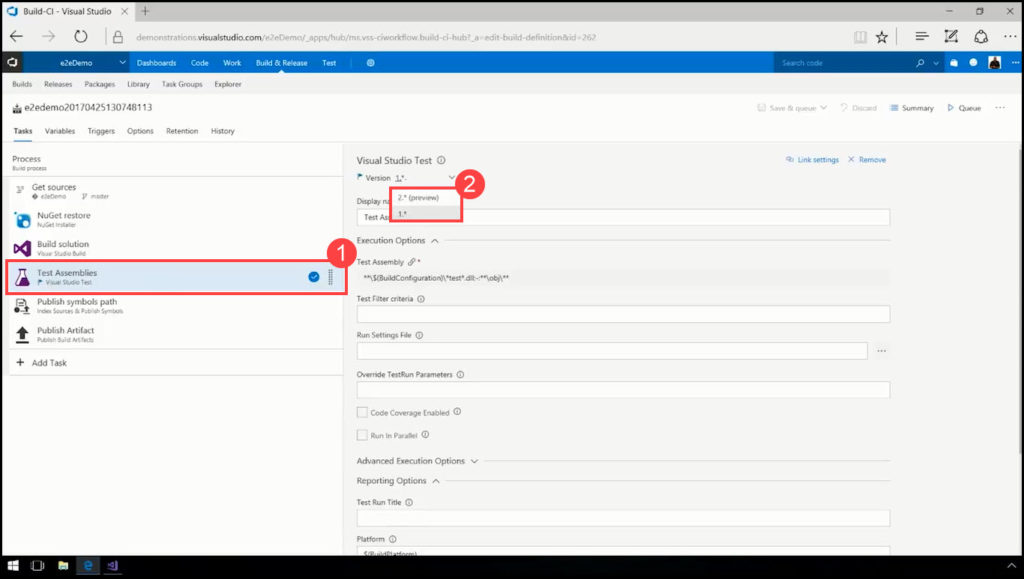
* Add an additional task.
* Customize the tasks that are already there.

[](http://mohamedradwan.com/wp-content/uploads/2017/12/3.3-Customizing-the-tasks-of-Build-Process.jpg)

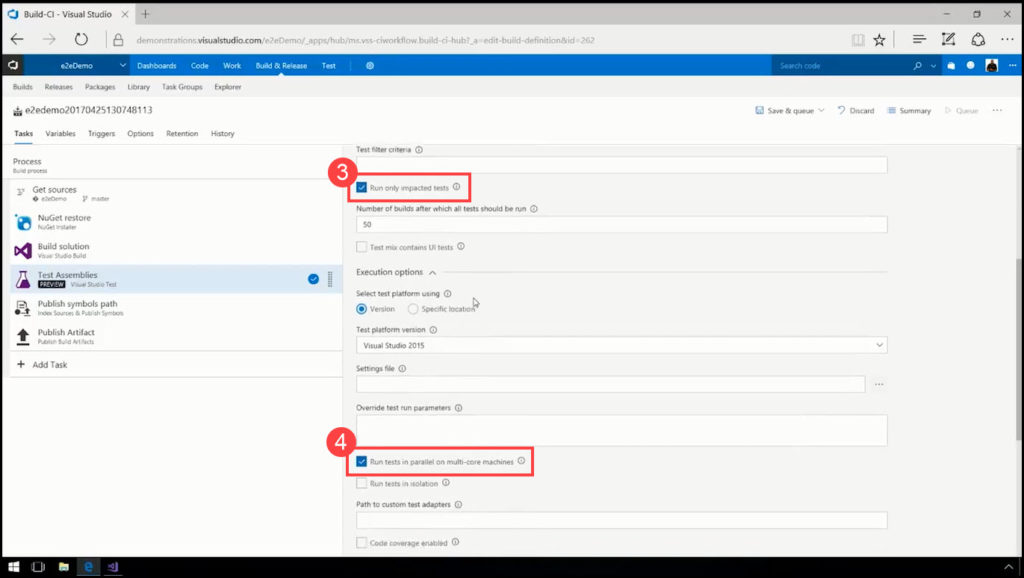
**Step 4: Test Assemblies Task**

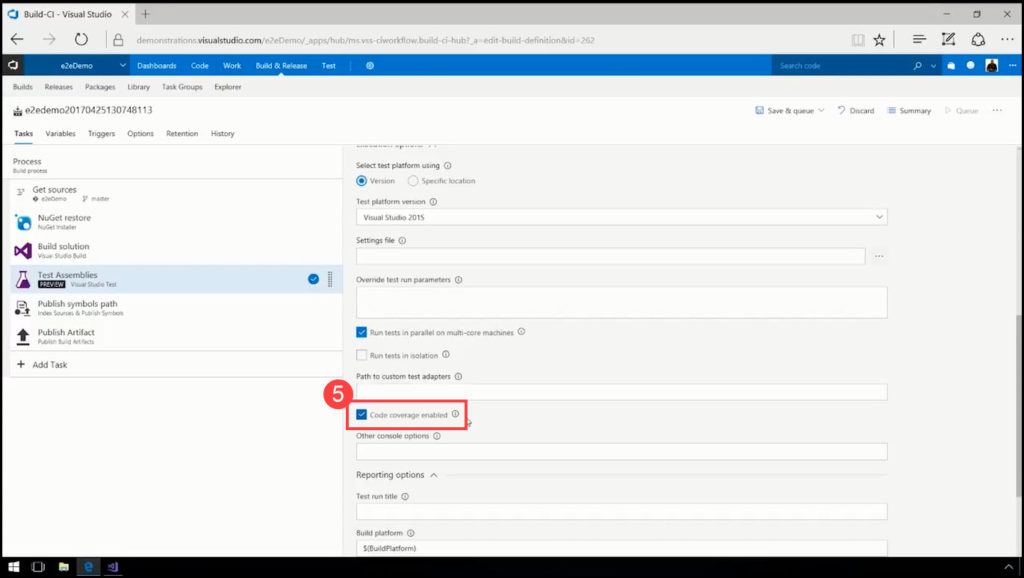
Each task has a **Version** selector that enables you to specify the major version of the task used in your build or deployment. When a new minor version is released (for example, 1.2 to 1.3), your build or release will automatically use the new version. However, if a new major **version** is released (for example 2.0), your build or release will continue to use the major version you specified until you edit the definition and manually change to the new major version.

1. Click on the **Test Assemblies.**
2. You can see a little flag icon which means that there is a new available preview version of this task. Click on the Flag Icon and choose version 2\* in order to preview.

[](http://mohamedradwan.com/wp-content/uploads/2017/12/4.1-Visual-Studio-Test-Asesemblies-Tasks.jpg)

1. There are several new items shown for the **Test Assemblies** One of them is **Run only impacted tests.**This is an item which allows to tools to analyze which lines of code were changed against the tests that were run in the past and you will know which tests execute which lines of code (you will not have to run all of your tests, you are able to run only the tests that were impacted by the changes).
2. **Run tests in parallel on multi-core machines** is an item which allows your tests to run in such a way to use all the cores you have available. Using this item you will effectively increase the number of tests running at the same time, which will reduce the time to run all the tests.

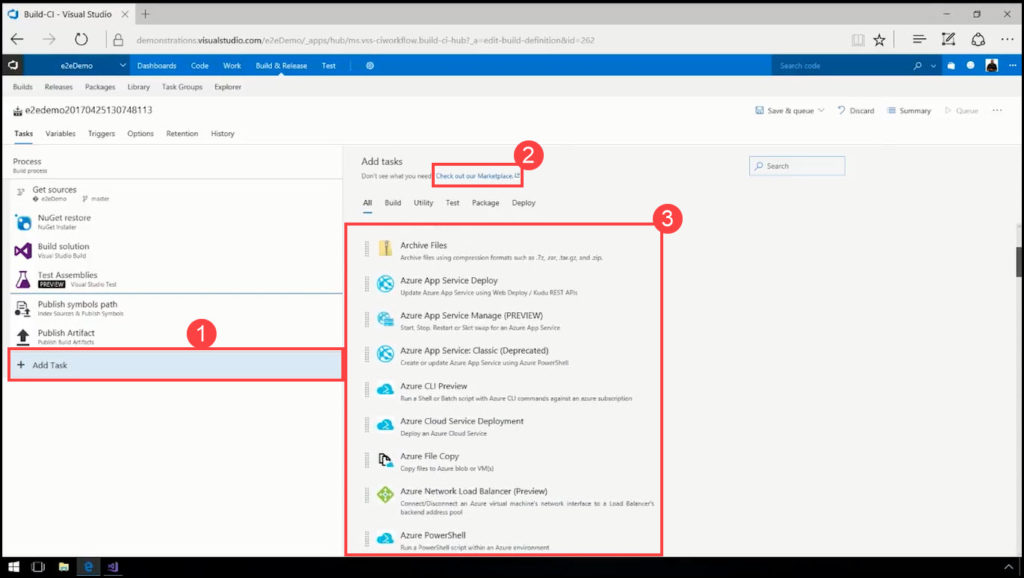
[](http://mohamedradwan.com/wp-content/uploads/2017/12/4.2-Visual-Studio-Test-Assemblies-Items.jpg)

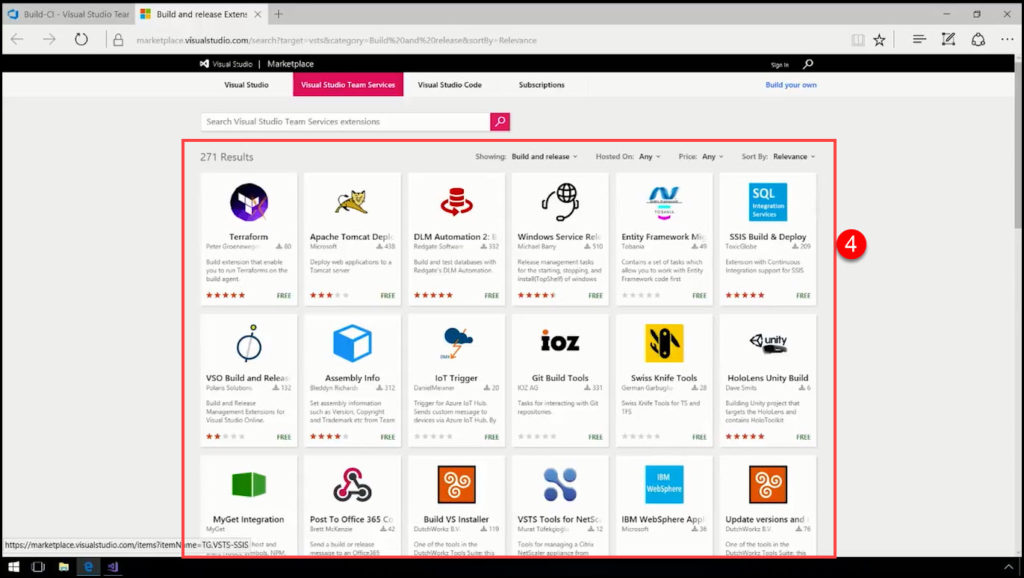
[](http://mohamedradwan.com/wp-content/uploads/2017/12/4.3-Visual-Stuido-Coverage-of-Test-Assemblies-Items.jpg)

**Step 5: Add an Additional Task**

A **task** is the building block for defining automation in a build definition, or in an environment of a release definition. A task is simply a packaged script or procedure that has been abstracted with a set of inputs. There are some built-in tasks in order to enable fundamental build and deployment scenarios.

1. Click on the **Add Task** plus button in order to create a new additional task.
2. An enormous list of tasks is displayed that can be run out of the box that allows you to target any language/platform. (Chef support, CocoaPods, Docker, NodeJS, Java).
3. If you want to install another feature or extension which is not listed, simply click on the link **Check out our Marketplace**which is displayed above the list of tasks**.**

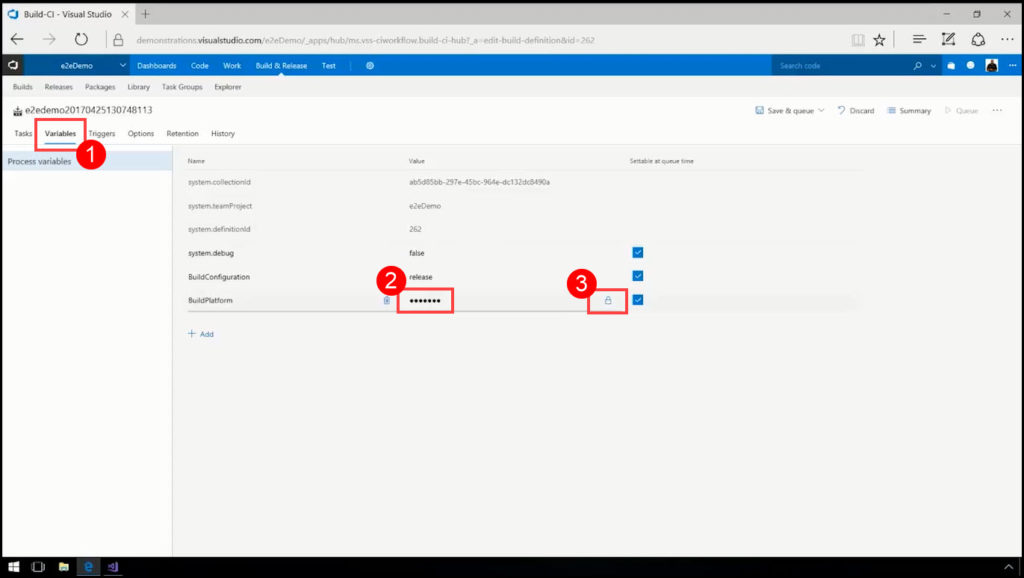
[](http://mohamedradwan.com/wp-content/uploads/2017/12/5.1-Adding-additional-task-in-build-definition.jpg)

[](http://mohamedradwan.com/wp-content/uploads/2017/12/5.2-Visual-Studio-Marketplace-extensions.jpg)

**Step 6: Setting Encrypted and Non-Encrypted Variables**

**Variables**are a great way to store and share key bits of data in your build definition. Some build templates automatically define some variables for you.

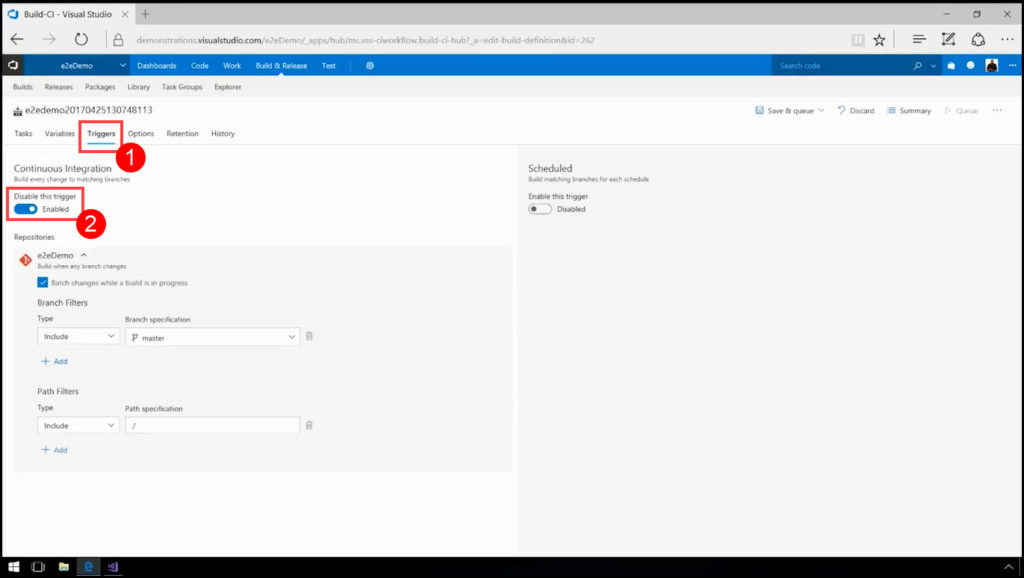
1. Go and click on the second tab named **Variables** (next to the tab Tasks).
2. Click on the padlock located next to the variable value, in order to encrypt it.
3. After encrypting, the value of the variable is displayed with asterisks, and no one can see this value except the person who encrypted it.

[](http://mohamedradwan.com/wp-content/uploads/2017/12/6-Encrypting-Variables-in-Build-Definitions.jpg)

**Step 7: Turn on the Continuous Integration (CI) Trigger**

On the **Triggers** tab, you specify the events that will trigger the build. You can use the same build definition for both CI and scheduled builds.

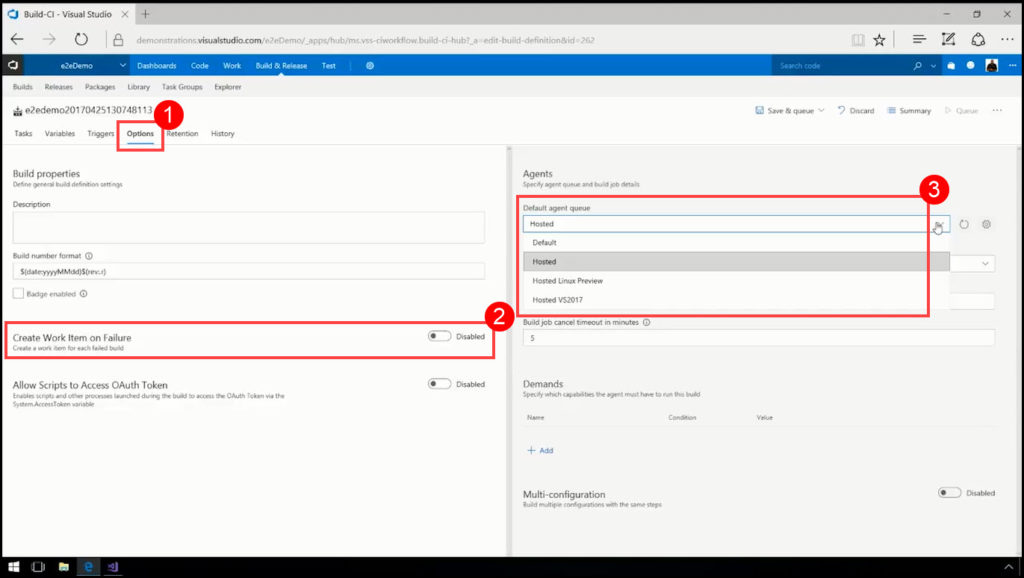
1. Go and click on the third tab named **Triggers**, where actually you can set up your **Continuous Integration**.
2. Enable the box **Disable this trigger**means that this build will run automatically whenever someone checks in code or with other words when a new version of the source artifacts is available.

[](http://mohamedradwan.com/wp-content/uploads/2017/12/7-Setting-up-Continuous-Integration-Triggers.jpg)

**Step 8: Build Definition Options**

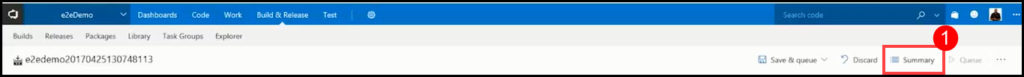
If the build process fails, you can automatically create a work item to track getting the problem fixed. You can specify the work item type. You can also select if you want to assign the work item to the requestor. For example, if this is a CI build, and a team member checks in some code that breaks the build, then the work item is assigned to that person.

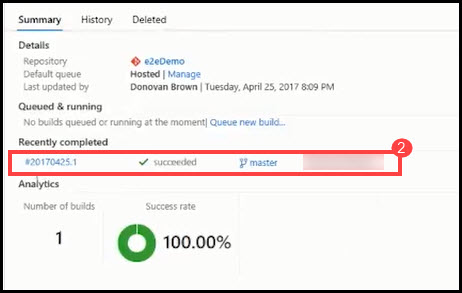
1. Go and click on the fourth tab named **Options**.
2. Enable the box **Create Work Item on Failure.**CI builds are supposed to build at every check-in, and if some of them fail because the developer made an error, you can automatically create a work item in order to track getting the problem fixed.
3. **Default agent queue**option is displayed in the second half of the **Options** In the drop-down list are listed all available pools:
   * **Default** (if your team uses private agents set up by your own)
   * [**Hosted**](https://docs.microsoft.com/en-us/vsts/build-release/concepts/agents/hosted)(Windows-based machine, if your team uses VS2017 or VS2015)
   * [**Hosted Linux Preview**](https://docs.microsoft.com/en-us/vsts/build-release/concepts/agents/hosted) (if your team uses development tools on Ubuntu)
   * [**Hosted VS2017**](https://docs.microsoft.com/en-us/vsts/build-release/concepts/agents/hosted) (if your team uses Visual Studio 2017)

[](http://mohamedradwan.com/wp-content/uploads/2017/12/8-Creating-a-work-item-on-failure-of-Build-Definition.jpg)

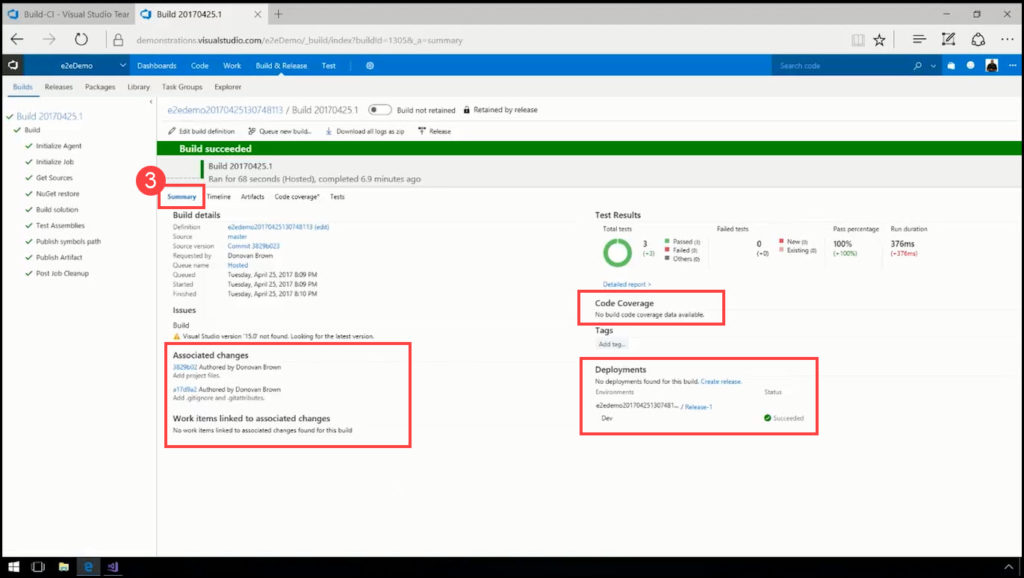
**Step 9: Build Summary**

You can see the summary of the build, in other words, everything that happened during the build, following the next steps:

[](http://mohamedradwan.com/wp-content/uploads/2017/12/9.1-Opening-Build-Summary.jpg)

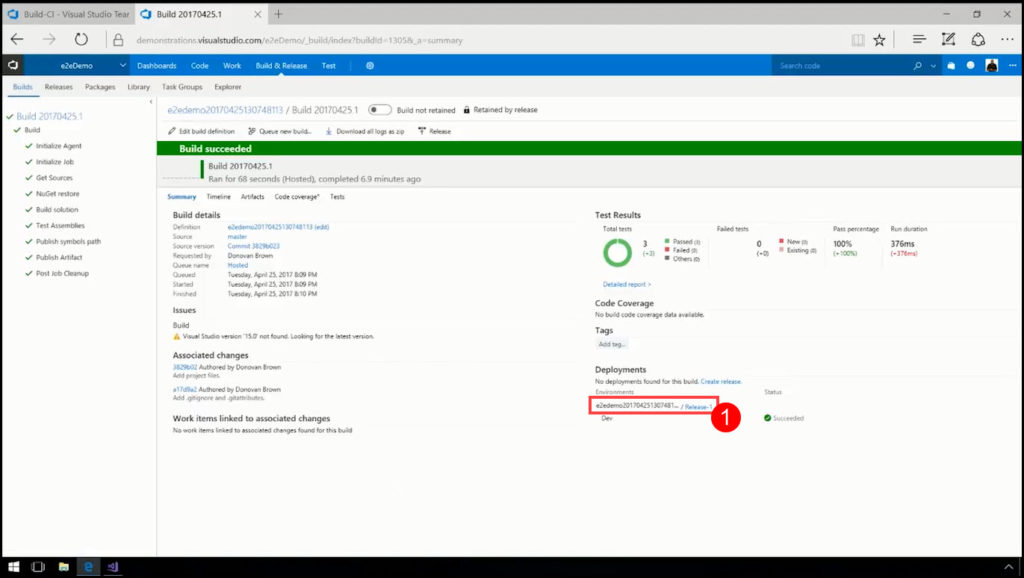
[](http://mohamedradwan.com/wp-content/uploads/2017/12/9.2-Short-Build-Summary.jpg)

* Code coverage
* All work items and tasks
* Deployments

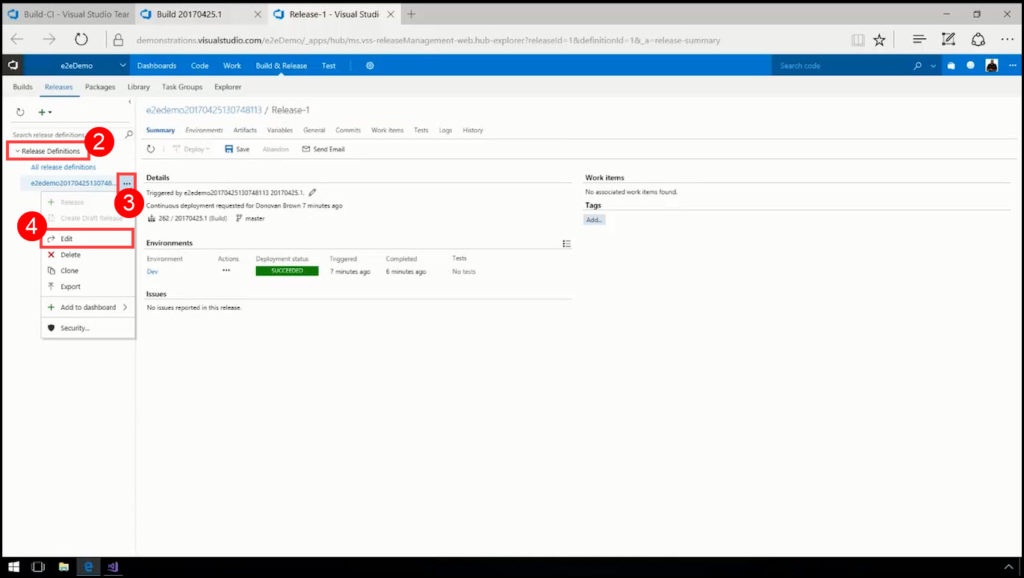
[](http://mohamedradwan.com/wp-content/uploads/2017/12/9.3-Build-Summary-with-Associated-changes-Code-Coverage-and-Deployments.jpg)

**Step 10: Release Definition**

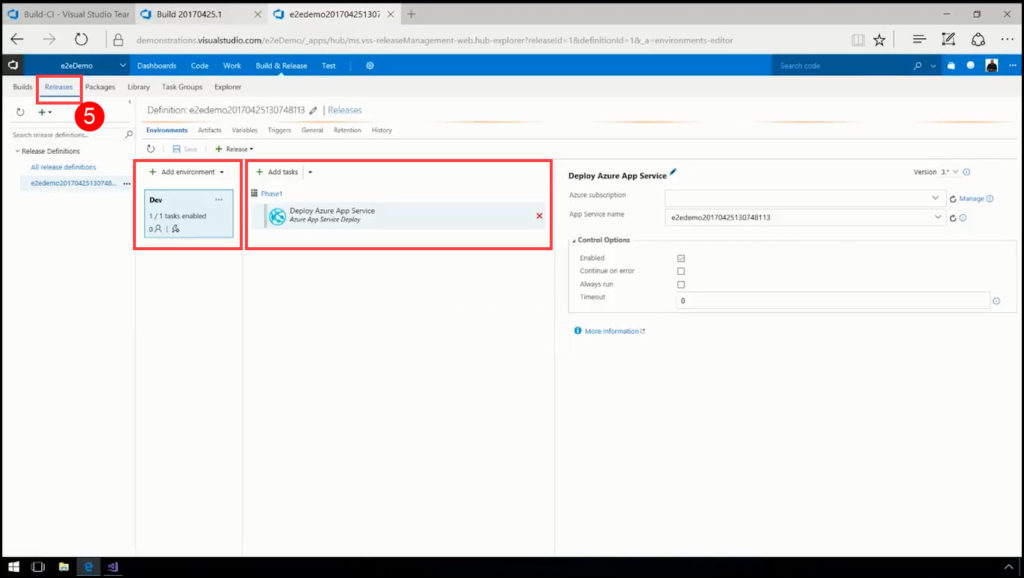
A **release definition** is one of the fundamental concepts in Release Management for VSTS and TFS. It defines the end-to-end release process for an application to be deployed across various environments. Remember that you as a developer, never have to leave VS in order to deploy the application from VS into Azure.

[](http://mohamedradwan.com/wp-content/uploads/2017/12/10.1-Opening-Release-Definition.jpg)

1. A **release definition** is displayed that deployed the code into **Azure**.
2. Click on the three dots located next to the particular release definition.
3. From the displayed context menu, select **Edit**.

[](http://mohamedradwan.com/wp-content/uploads/2017/12/10.2-Editing-Release-Definition.jpg)

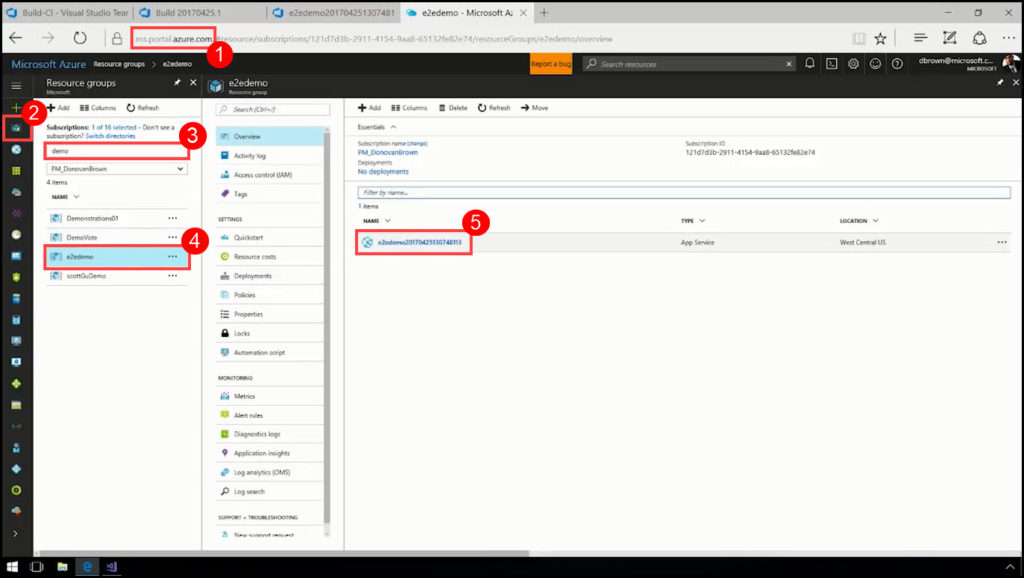
* Series of environments
* Tasks that you want to perform in each environment

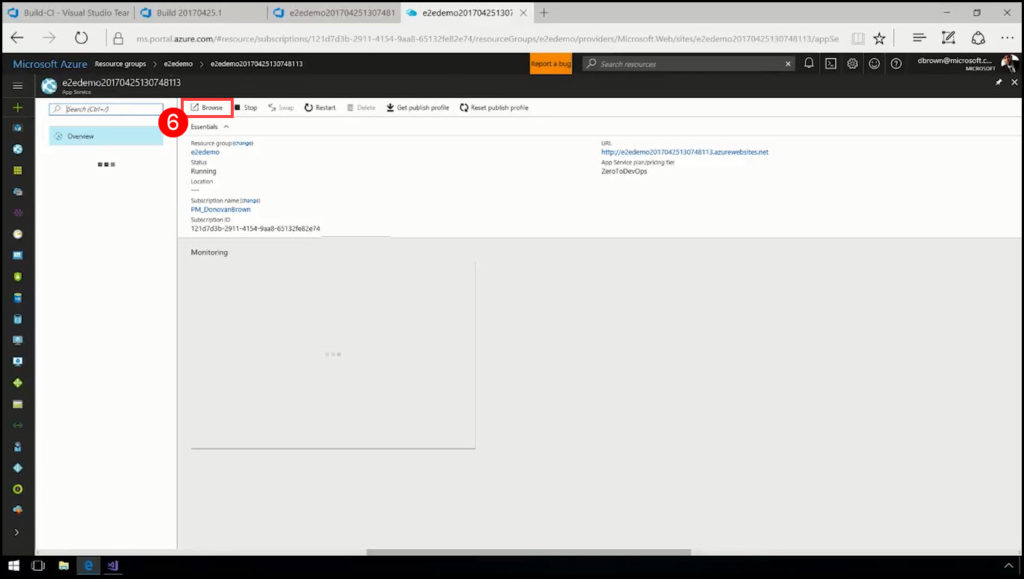
[](http://mohamedradwan.com/wp-content/uploads/2017/12/10.3-Deploying-result-of-CI-into-Azure-using-CD.jpg)

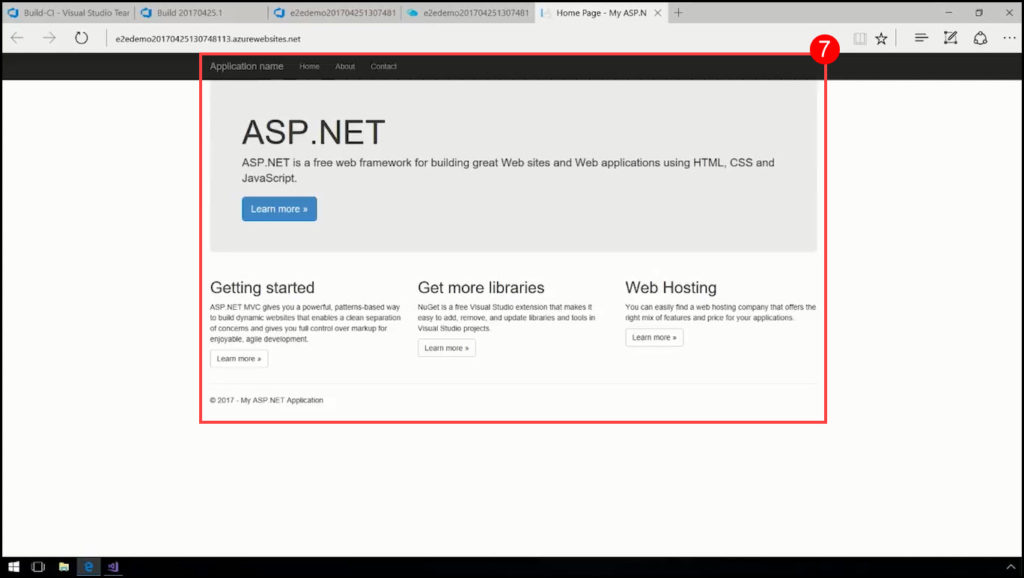
**Step 11: Check if the Application Is Really Deployed From Visual Studio Into Azure**

**MicrosoftAzure** is a cloud computing service for building, testing, deploying, and managing applications and services through a global network of Microsoft-managed data centers. In this step you will verify if your web application is deployed in Azure, following the next steps:

1. Go to your **Azure portal.**
2. Click on the **Resource Groups.**
3. Search for the "demo."
4. Click In the search results on your web project "e2edemo."
5. Open the web application link.

[](http://mohamedradwan.com/wp-content/uploads/2017/12/11.1-Logging-into-Azure-portal.jpg)

[](http://mohamedradwan.com/wp-content/uploads/2017/12/11.2-Browsing-for-deployed-application-in-Azure.jpg)

[](http://mohamedradwan.com/wp-content/uploads/2017/12/11.3-Deployed-application.jpg)