DevOps Kata

**Continuous deployment / integration**

Last updated: 10/30/2016



1. **TABLE OF CONTENT**

[Overview 3](#_Toc465665993)

[Prerequisites 3](#_Toc465665994)

[Exercises 3](#_Toc465665995)

[Exercise 1: Create a team project 5](#_Toc465665996)

[Task 1: Create a team project 5](#_Toc465665997)

[Exercise 2: Create and deploy with VS 7](#_Toc465665998)

[Task 1: Connect to your Azure account and VSTS instance in Visual Studio 7](#_Toc465665999)

[Task 2: Create a new app and publish it 8](#_Toc465666000)

[Exercise 3: Add your code to VSTS 17](#_Toc465666001)

[Task 1: Create a local Git repo for your Visual Studio project 17](#_Toc465666002)

[Task 2: Publish your code 18](#_Toc465666003)

[Task 3: Commit updates 20](#_Toc465666004)

[Task 4: Sync changes 20](#_Toc465666005)

[Exercise 4: Create a build and deploy to Azure 22](#_Toc465666006)

[Task 1: Create the definition 22](#_Toc465666007)

[Task 2: Specify the web app to deploy 23](#_Toc465666008)

[Task 3: Finish and test the definition 25](#_Toc465666015)

## Overview

Continuous Deployment, or continous delivery (CD), is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time. It aims at building, testing, and releasing software faster and more frequently. The approach helps reduce the cost, time, and risk of delivering changes by allowing for more incremental updates to applications in production. A straightforward and repeatable deployment process is important for continuous delivery. (<https://en.wikipedia.org/wiki/Continuous_delivery>).

Continuous Integration is the practice of merging all developer working copies to a shared mainline several times a day (<https://en.wikipedia.org/wiki/Continuous_integration>).

These practices are important steps to start realizing a dev ops vision. They may be uilized to varying degrees of tolerance within an organization, based primarily on the criticality of the project and application.

### Prerequisites

1. In order to complete the lab
2. 1. Log on to your Visual Studio / MSDN subscription and create a VSTS instance. Alternatively, you can use a team sandbox VSTS instance if you have one.
3. 2. Have an azure subscription available that you can deploy resource to.
4. 3. Visual Studio Enterprise

### Exercises

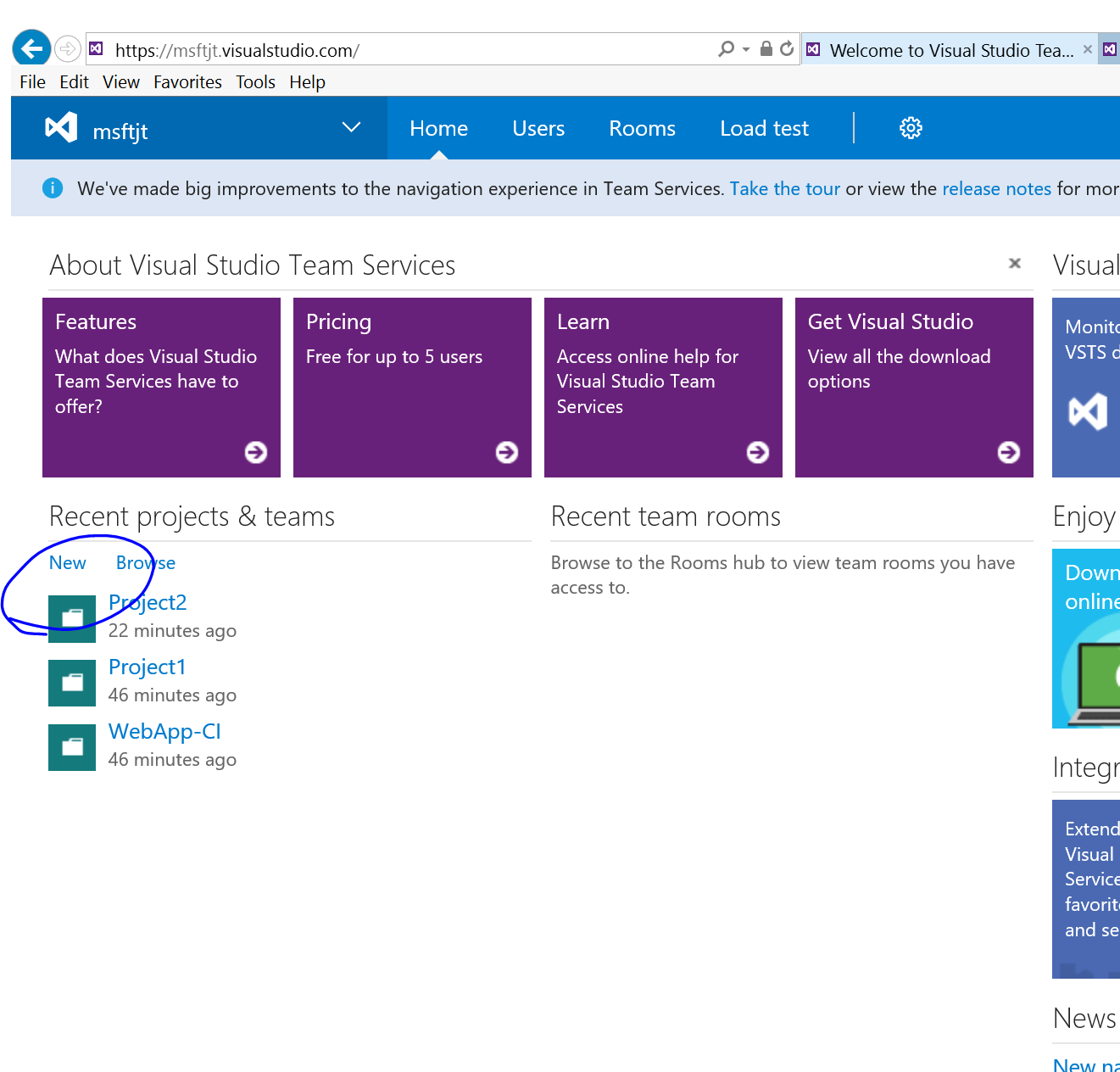
* 1. This hands-on lab includes the following exercises:
  2. Create a team project/repository and a local directory
  3. Create and deploy a basic web app to azure
  4. Check your code into VSTS
  5. Create a build and deploy to azure
  6. Homework: send me your url with a nice message or screen shot
  7. Estimated time to complete this exersize: **15 minutes**.

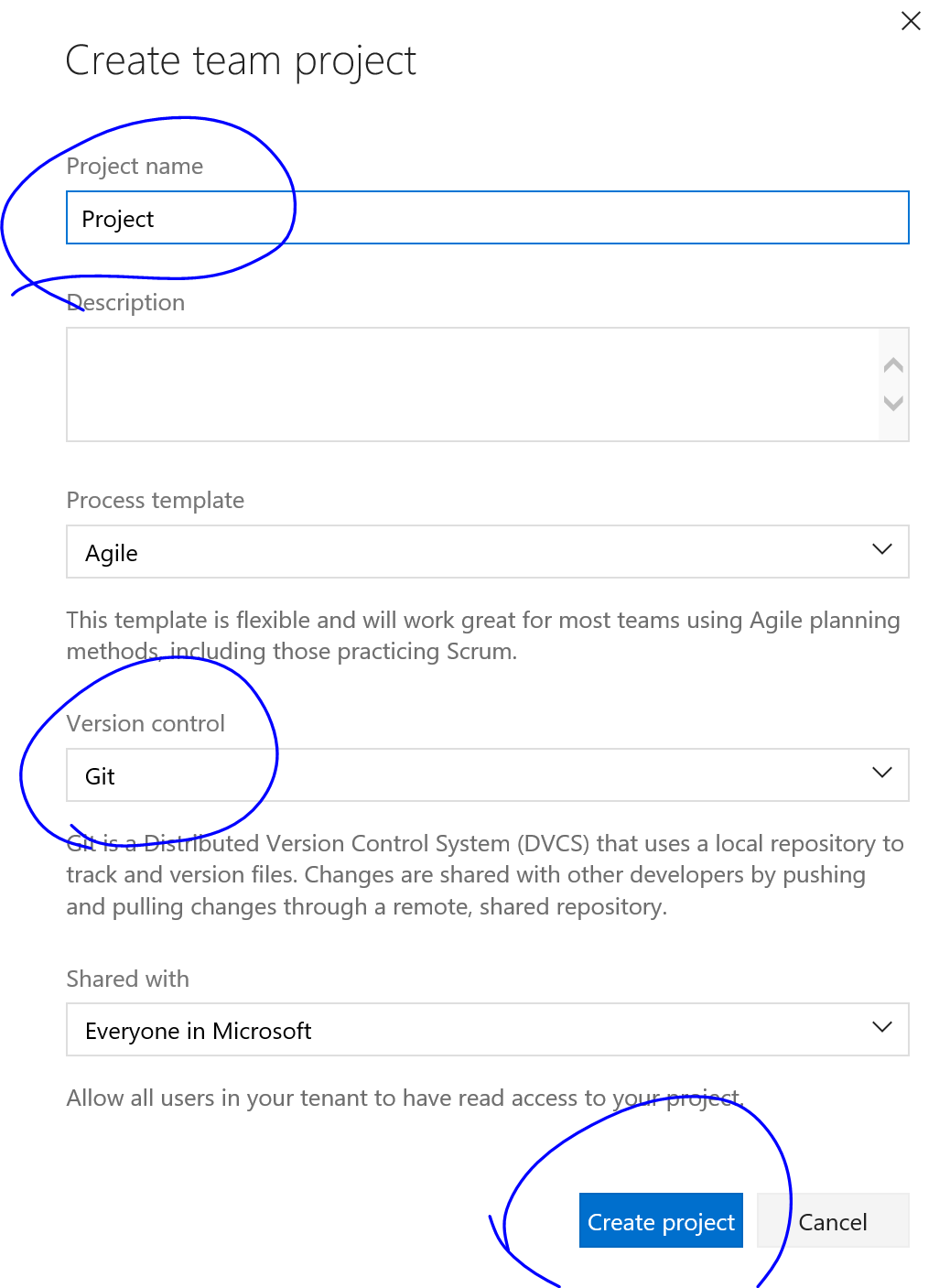
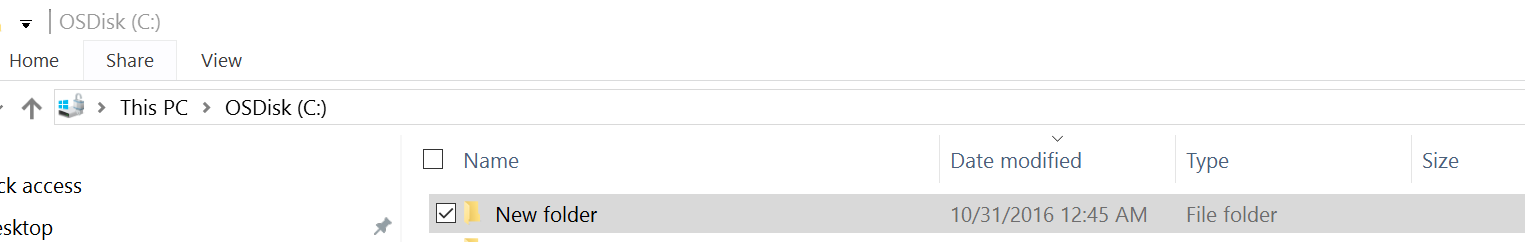
Exercise 1: Create a team project

VSTS or TFS is your home for new projects as you develop and share code

#### Task 1: Create a team project

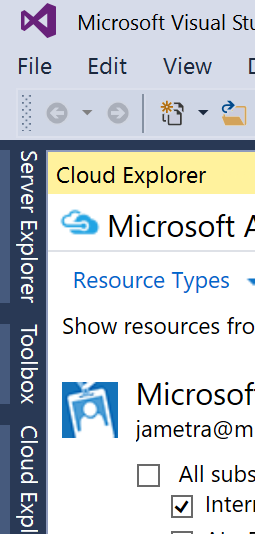
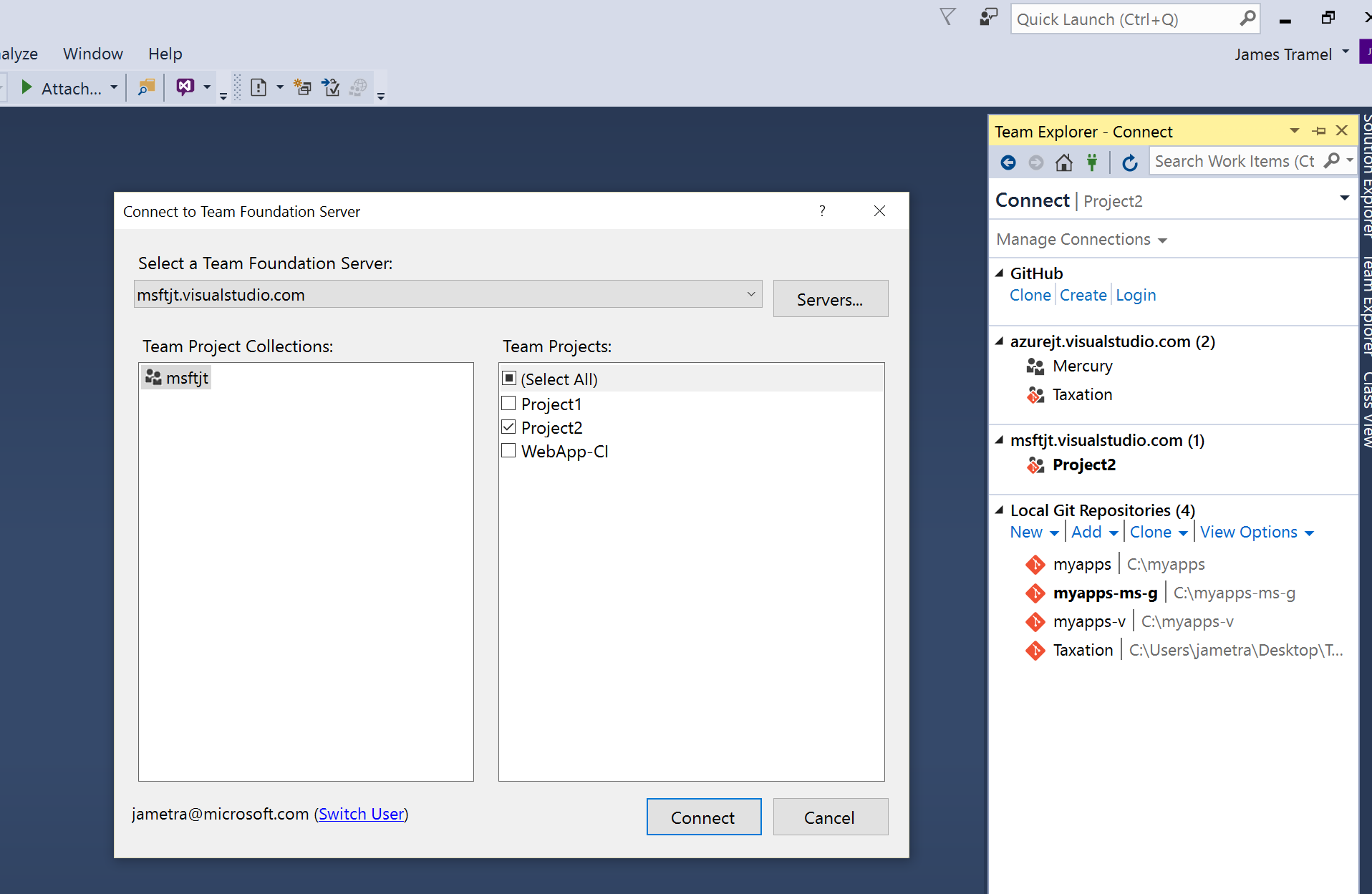
1. From a web browser, connect to the team project that you want to work in. For example, the Fabrikam, Inc. team navigates to http://fabrikamprime:8080/tfs/DefaultCollection/Fabrikam%20Fiber%20Website/.  
   If you haven’t been added as a team member, [get added now](https://www.visualstudio.com/en-us/docs/work/scale/multiple-teams#add-team-members).

Click the new project button

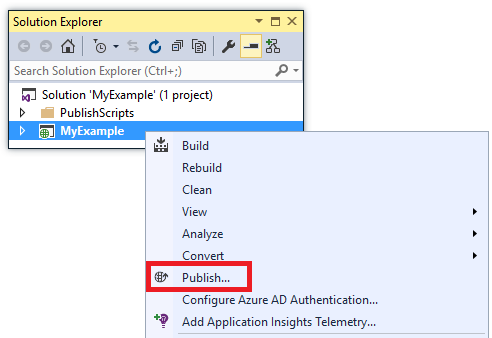
1. Enter a title, use Git or TFVC, and click create project.
2. Create a new local directory on your local computer

## Exercise 2: Create and deploy with VS

#### Task 1: Connect to your Azure account and VSTS instance in Visual Studio

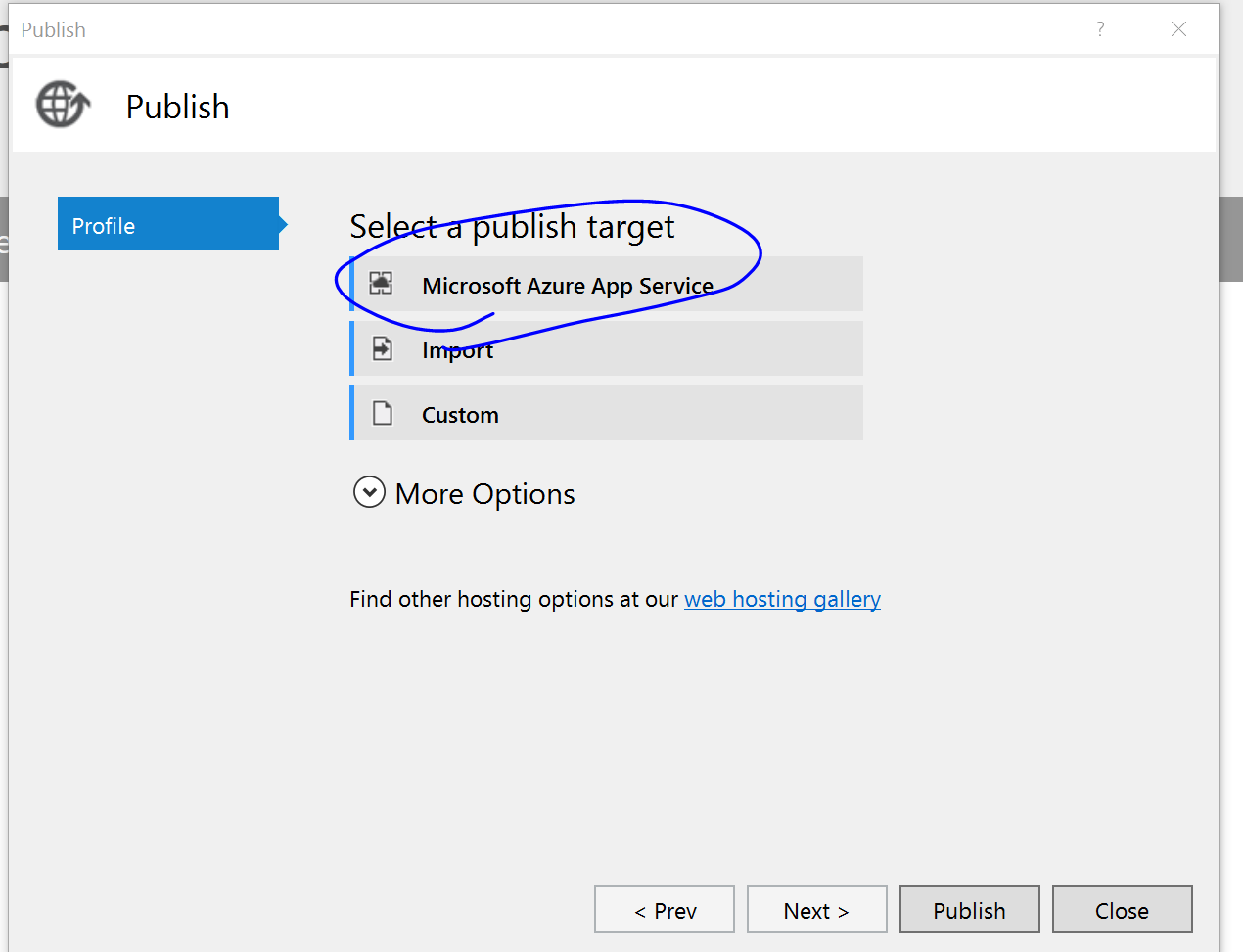
1. Open Visual Studio 2015.
2. Connect using Cloud explorer to your Azure Subscription 
3. Open Team explorer and connect to your existing project 

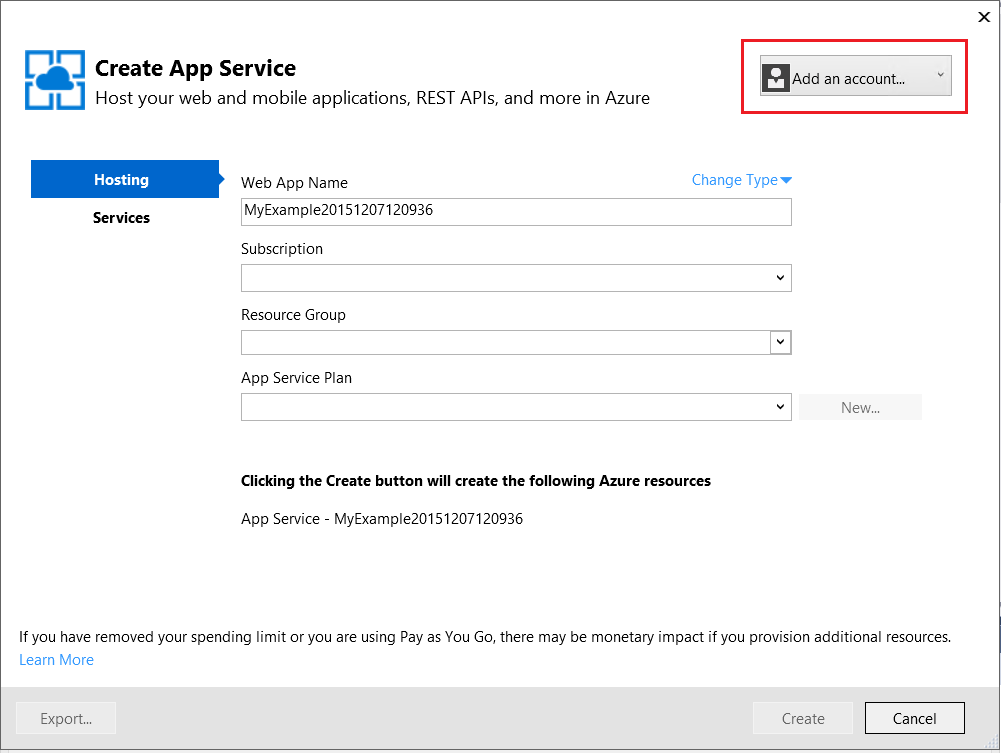
#### Task 2: Create a new app and publish it

1. Click **File > New > Project**.
2. In the **New Project** dialog box, click **Visual C# > Cloud > ASP.NET Web Application**.
3. Name the application and then click **OK**.
4. In the **New ASP.NET Project** dialog box, select the **MVC** template and click **OK**.
5. In **Solution Explorer**, right-click the project, and choose **Publish**.

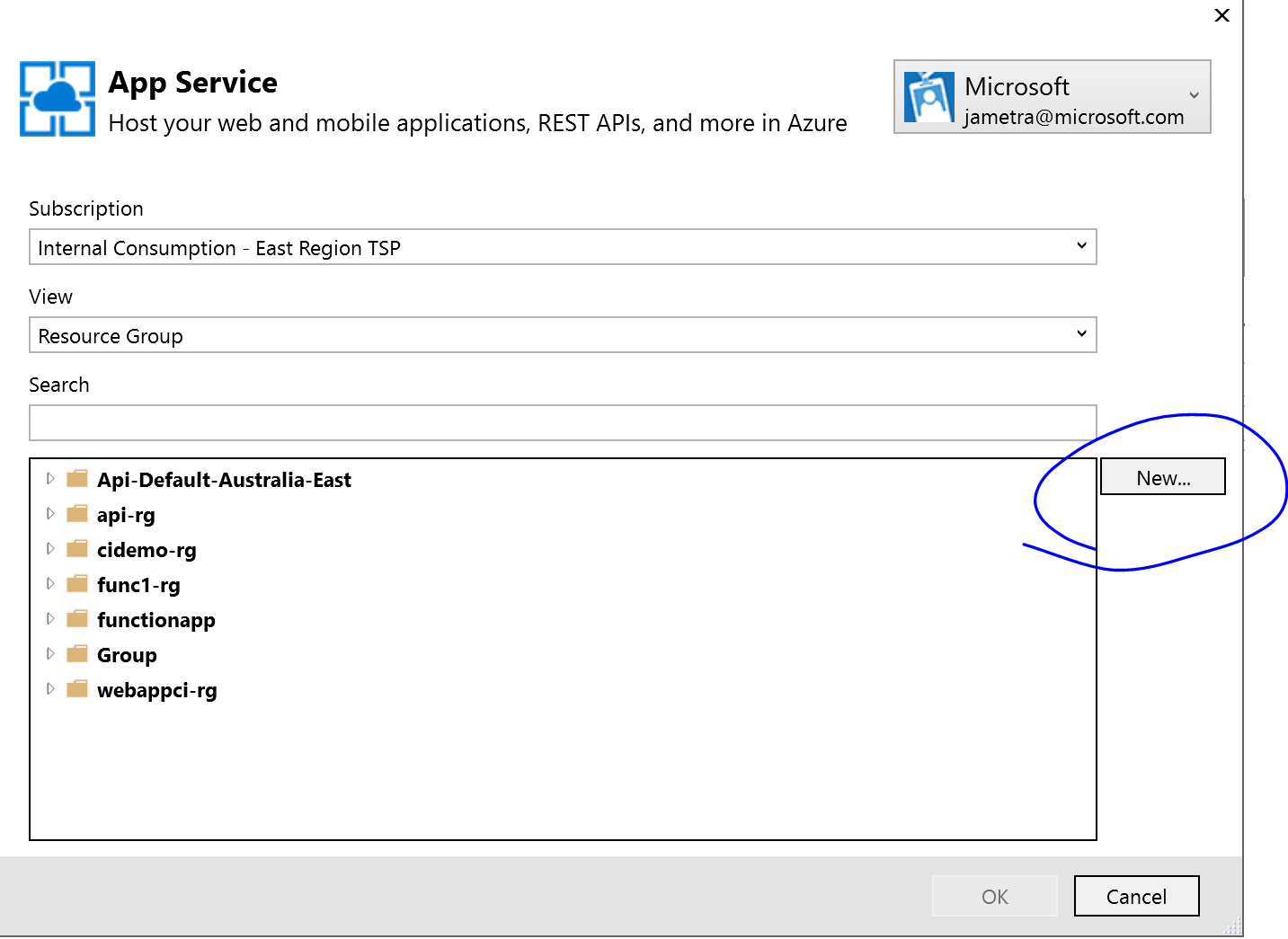
In a few seconds, the **Publish Web** wizard appears. The wizard opens to a publish profile that has settings for deploying the web project to the new web app.

The publish profile includes a user name and password for deployment. These credentials have been generated for you, and you don't have to enter them. The password is encrypted in a hidden user-specific file in the Properties\PublishProfiles folder.

1. Select Microsoft Azure App Service as a publish target
2. In the **Create App Service** dialog, click **Add an account (if necessary)**, and then sign in to Azure with the ID and password of the account that you use to manage your Azure subscription.



If you already signed in earlier on the same computer, you might not see the **Add an account** button. In that case, you can skip this step or you might need to reenter your credentials.

1. Click New  
   
2. Enter a **Web App Name** that is unique in the azurewebsites.net domain. For example, you can name it MyExample with numbers to the right to make it unique, such as MyExample810. If a default web name is created for you, it will be unique and you can use that.

If someone else has already used the name that you enter, you see a red exclamation mark to the right instead of a green check mark, and you have to enter a different name.

The URL for your application is this name plus .azurewebsites.net. For example, if the name is MyExample810, the URL is myexample810.azurewebsites.net.

You can also use a custom domain with an Azure web app. For more information, see [Configure a custom domain name in Azure App Service](https://azure.microsoft.com/en-us/documentation/articles/web-sites-custom-domain-name/).

1. Click the **New** button next to the **Resource Group** box, and then enter "MyExample" or another name if you prefer.

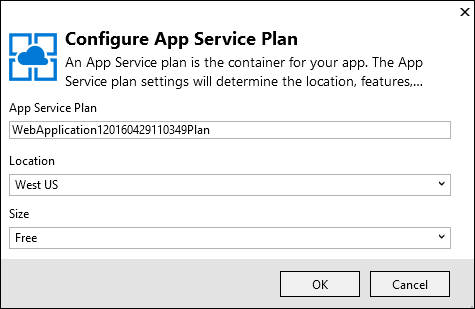
Create App Service dialog

A resource group is a collection of Azure resources such as web apps, databases, and VMs. For a tutorial, it's generally best to create a new resource group because that makes it easy to delete in one step any Azure resources that you create for the tutorial. For more information, see [Azure Resource Manager overview](https://azure.microsoft.com/en-us/documentation/articles/resource-group-overview/).

1. Click the **New** button next to the **App Service Plan** drop-down.



The **Configure App Service Plan** dialog appears.



In the following steps, you configure an App Service plan for the new resource group. An App Service plan specifies the compute resources that your web app runs on. For example, if you choose the free tier, your API app runs on shared VMs, while for some paid tiers it runs on dedicated VMs. For more information, see [App Service plans overview](https://azure.microsoft.com/en-us/documentation/articles/azure-web-sites-web-hosting-plans-in-depth-overview/).

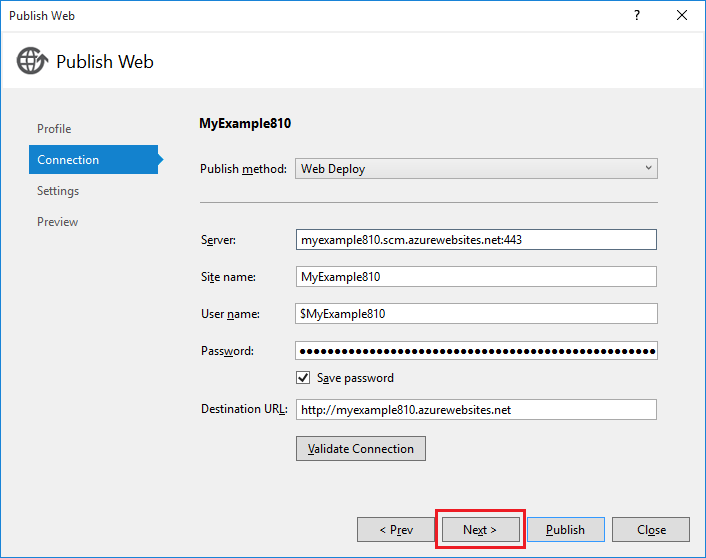
1. In the **Configure App Service Plan** dialog, enter "MyExamplePlan" or another name if you prefer.
2. In the **Location** drop-down list, choose the location that is closest to you.

This setting specifies which Azure datacenter your app will run in. For this tutorial, you can select any region and it won't make a noticeable difference. But for a production app, you want your server to be as close as possible to the clients that are accessing it, to minimize [latency](http://www.bing.com/search?q=web%20latency%20introduction&qs=n&form=QBRE&pq=web%20latency%20introduction&sc=1-24&sp=-1&sk=&cvid=eefff99dfc864d25a75a83740f1e0090).

1. In the **Size** drop-down, click **Free**.

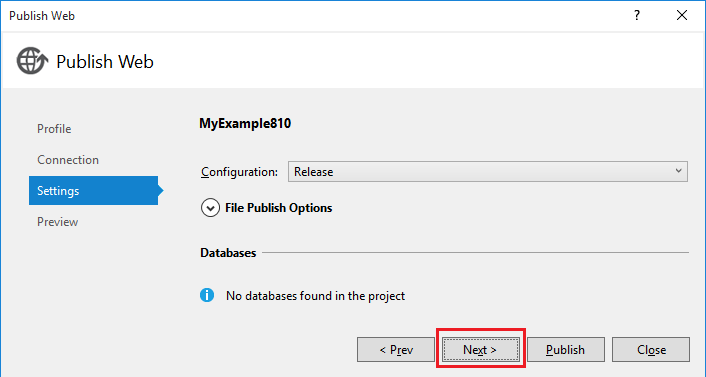
For this tutorial, The free pricing tier will provide good enough performance.

1. In the **Configure App Service Plan** dialog, click **OK**.
2. In the **Create App Service** dialog box, click **Create**.
3. On the **Connection** tab of the **Publish Web** wizard, click **Next**.



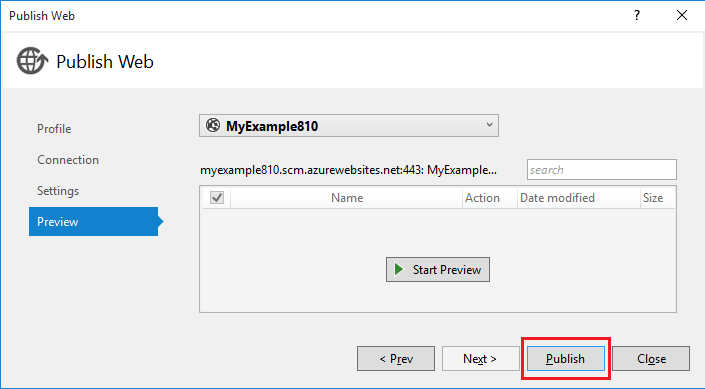
Next is the **Settings** tab. Here you can change the build configuration to deploy a debug build for [remote debugging](https://azure.microsoft.com/en-us/documentation/articles/web-sites-dotnet-troubleshoot-visual-studio/#remotedebug). The tab also offers several [File Publish Options](https://msdn.microsoft.com/library/dd465337.aspx#Anchor_2).

1. On the **Settings** tab, click **Next**.



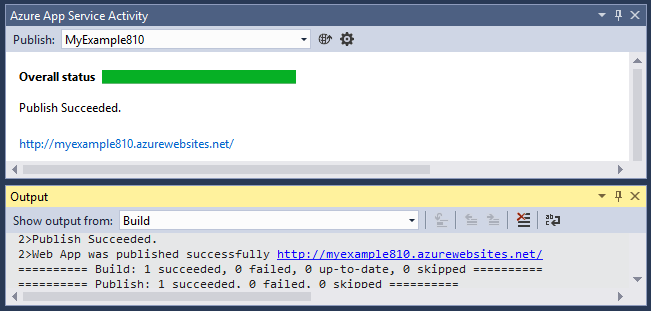
The **Preview** tab is next. Here you have an opportunity to see what files are going to be copied from your project to the API app. When you're deploying a project to an API app that you already deployed to earlier, only changed files are copied. If you want to see a list of what will be copied, you can click the **Start Preview** button.

1. On the **Preview** tab, click **Publish**.

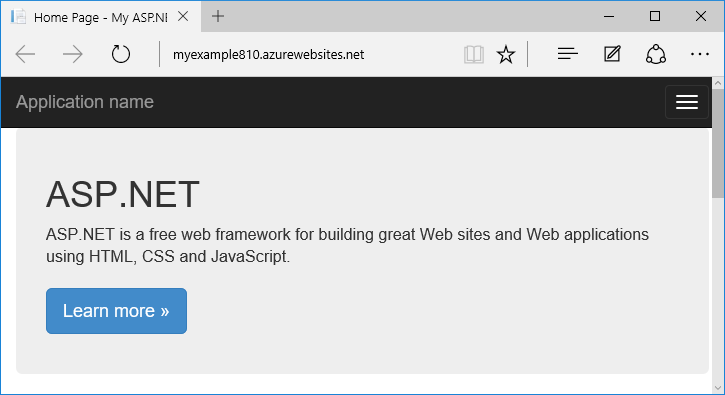


When you click **Publish**, Visual Studio begins the process of copying the files to the Azure server. This may take a minute or two.

The **Output** and **Azure App Service Activity** windows show what deployment actions were taken and report successful completion of the deployment.



Upon successful deployment, the default browser automatically opens to the URL of the deployed web app, and the application that you created is now running in the cloud. The URL in the browser address bar shows that the web app is loaded from the Internet.



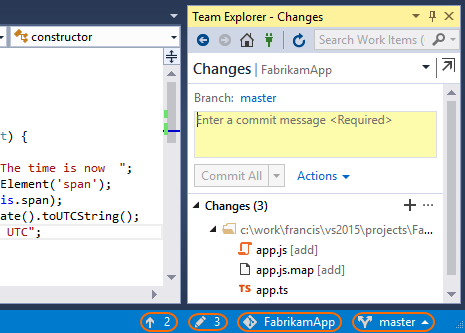
<https://azure.microsoft.com/en-us/documentation/articles/web-sites-dotnet-get-started/>

## Exercise 3: Add your code to VSTS

#### Task 1: Create a local Git repo for your Visual Studio project

Create a new local Git repo for your project by selecting Publish to Git from the Visual Studio Status baron the status bar in the lower right hand corner of Visual Studio. This will create a new repo in the folder the solution is in and commit your code into that repo.

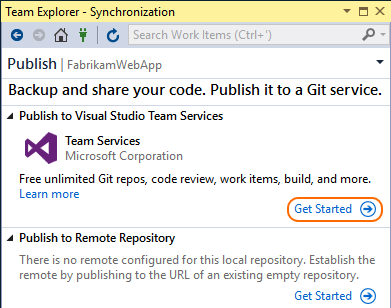
Once you have a local repo, select items in the status bar to quickly navigate between Git tasks in Team Explorer.



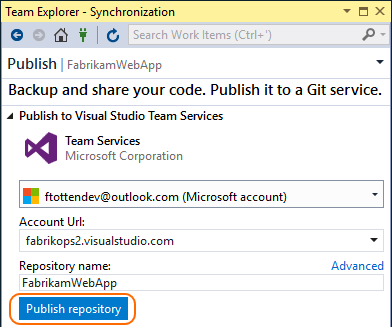
* Visual Studio Unpublished Changes Status Bar iconshows the number of outgoing commits. Selecting this will open the **Synchronization** view in Team Explorer.
* Visual Studio Pending Changes Status Bar iconshows the number of uncommitted file changes. Selecting this will open the **Changes** view in Team Explorer.
* Visual Studio Repo Status Bar iconshows the current Git repo. Selecting this will open the **Connect** view in Team Explorer.
* Visual Studio branch status bar iconshows your current Git branch. Selecting this displays a branch picker to quickly switch between Git branches or create new branches.

#### Task 2: Publish your code

1. In the **Sync** view in Team Explorer, select the **Get Started** link under **Publish to Team Services**.

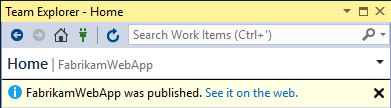


1. Verify your email and select your account in the **Account Url** drop down.
2. Enter your repository name and select **Publish Repository**.



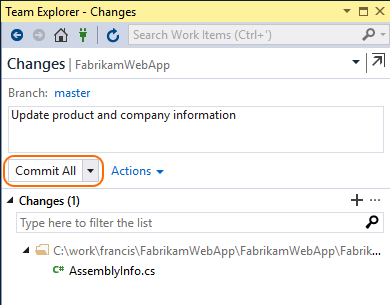
This creates a new Team Project in your account with the same name as the repository. To create the repo in an existing Team Project, click **Advanced** next to **Repository name** and select a team project.

Your code is now in a Team Services repo and your local repository can push and pull updates. You can view your code on the web by selecting **See it on the web** .

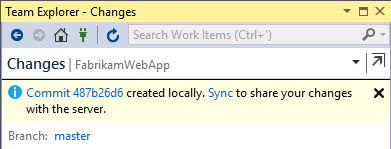


#### Task 3: Commit updates

1. As you write your code, your changes are automatically tracked by Visual Studio. You can [commit](https://www.visualstudio.com/en-us/docs/git/tutorial/commits) changes to your local Git repository by selecting the pending changes icon Visual Studio Pending Changes Status Bar iconfrom the status bar.
2. On the **Changes** view in Team Explorer, add a message describing your update and commit your changes.



1. Your changes are now committed into the local Git repository.

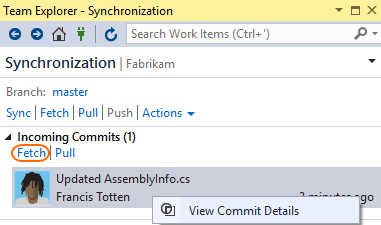


#### Task 4: Sync changes

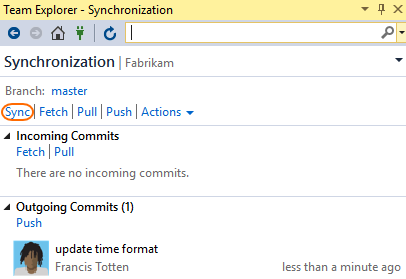
As other team members update the code in the repo, you'll need to sync their changes with your own code.

1. From the **Sync** view in Team Explorer, fetch the commits to view changes that your team has made. Double-click the commit to view the list of files changed.

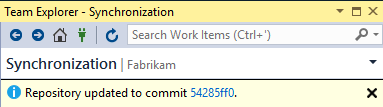
Fetching changes does not merge the commits into your local repository. You will need to [pull](https://www.visualstudio.com/en-us/docs/git/tutorial/pulling) the changes to merge them into your local repository.



1. When you're ready to merge the changes with your local repo, **Sync** the changes to [push](https://www.visualstudio.com/en-us/docs/git/tutorial/pushing) your commits and pull changes from others.



1. The changes from your team are now in your local repository.

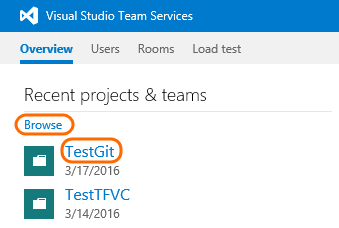


1. https://www.visualstudio.com/en-us/docs/git/share-your-code-in-git-vs

## Exercise 4: Create a build and deploy to Azure

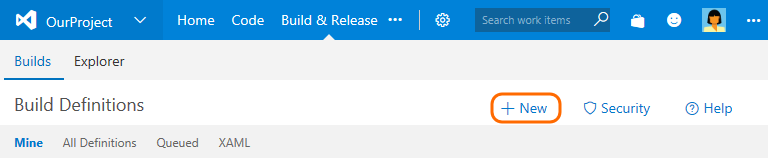
#### Task 1: Create the definition

1. [Open your team project in your web browser ▼](https://www.visualstudio.com/en-us/docs/build/apps/aspnet/aspnet4-to-azure#expando-begin-create-build-definition-open-team-project)



* + On-premises http://{your\_server}:8080/tfs/DefaultCollection/{your\_team\_project}
  + Visual Studio Team Services https://{your\_account}.visualstudio.com/DefaultCollection/{your\_team\_project}

1. [Create a build definition ▼](https://www.visualstudio.com/en-us/docs/build/apps/aspnet/aspnet4-to-azure#expando-begin-create-build-definition-create)



1. On the Definition Templates dialog box, click **Deployment**, and then select Deploy: Azure Web App Deployment**Azure WebApp**.

#### Task 2: Specify the web app to deploy

On the Build tab:

|  |  |
| --- | --- |
| Deploy: Azure Web App Deployment**Deploy: Azure Web App Deployment**  Deploy the app. Azure Subscription Make sure the subscription you want to use is selected.  If a subscription is not available, then add a service endpoint:   1. Manage 2. Click New Service Endpoint, and then click Azure. 3. On the Add Azure Subscription dialog box:    1. Select Certificate.    2. Click the link to download your publishsettings xml file and then open the file.    3. Copy the ID and certificate values from the file and paste them into the Add Azure Subscription dialog box.   <?xml version="1.0" encoding="utf-8"?>  <PublishData>    <PublishProfile      SchemaVersion="2.0"      PublishMethod="AzureServiceManagementAPI">      <Subscription        ServiceManagementUrl="https://management.core.windows.net"        Id="{Copy\_and\_Paste\_into\_Subscription\_id\_field}"        Name="{Copy\_and\_paste\_into\_Name\_field\_or\_use\_another\_name}"  ManagementCertificate="{Copy\_and\_paste\_into\_Subscription\_certificate\_field}" />    </PublishProfile>  </PublishData> Web App Name Enter the name.  **Tip:** The name you enter must be unique or the name of an Azure Web App you have already created. Azure will create the Web App for you and add it to your subscription if it does not already exist. Web Deploy Package $(build.stagingDirectory)\\*\*\\*.zip | |
| Deploy: Azure Web App Deployment**Deploy: Azure Web App Deployment** | Deploy the app. Azure Subscription Make sure the subscription you want to use is selected.  If a subscription is not available, then add a service endpoint:   1. Manage 2. Click New Service Endpoint, and then click Azure. 3. On the Add Azure Subscription dialog box:    1. Select Certificate.    2. Click the link to download your publishsettings xml file and then open the file.    3. Copy the ID and certificate values from the file and paste them into the Add Azure Subscription dialog box.   <?xml version="1.0" encoding="utf-8"?>  <PublishData>    <PublishProfile      SchemaVersion="2.0"      PublishMethod="AzureServiceManagementAPI">      <Subscription        ServiceManagementUrl="https://management.core.windows.net"        Id="{Copy\_and\_Paste\_into\_Subscription\_id\_field}"        Name="{Copy\_and\_paste\_into\_Name\_field\_or\_use\_another\_name}"  ManagementCertificate="{Copy\_and\_paste\_into\_Subscription\_certificate\_field}" />    </PublishProfile>  </PublishData> Web App Name Enter the name.  **Tip:** The name you enter must be unique or the name of an Azure Web App you have already created. Azure will create the Web App for you and add it to your subscription if it does not already exist. Web Deploy Package $(build.stagingDirectory)\\*\*\\*.zip |

#### Task 3: Finish and test the definition

1. On the Repository tab:
   * Git: Make sure the repository and branch containing your application are selected. (By default the repository with the same name as the team project is selected.)
   * TFVC: Make sure the folder that contains your app is mapped.
2. On the Triggers tab select continuous integration (CI). If your code is in Git, specify the branches you want to build.
3. Save your definition and queue the build.

After a successful build, check your site: http://{web\_app\_name}.azurewebsites.net

https://www.visualstudio.com/en-us/docs/build/apps/aspnet/aspnet4-to-azure