Microsoft Cloud Workshops

Agile continuous delivery hackathon

Lab Guide

February 2017

© 2017 Microsoft Corporation. All rights reserved. This document is confidential and proprietary to Microsoft. Internal use only. This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

This document is provided "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

Some examples are for illustration only and are fictitious. No real association is intended or inferred.

## Contents

[Exercise 0: Environment setup 2](#_Toc474136359)

[Agile continuous delivery hackathon 11](#_Toc474136360)

[Agile Continuous Delivery hackathon answers 14](#_Toc474136361)

## Overview

Contoso has asked you to automate their development process in two specific ways. First, they want you to define an Azure Resource Manager template that can deploy their application into the Microsoft Azure cloud using Platform-as-a-Service technology for their web application and their SQL database. Second, they want you to implement a continuous delivery process that will connect their source code repository into the cloud, automatically run their code changes through unit tests, and then automatically create new software builds and deploy them onto environment-specific deployment slots so that each branch of code can be tested and accessed independently..

## Requirements

* Microsoft Azure subscription
* Local machine or a virtual machine configured with:
  + Visual Studio 2015 Update 3 Community Edition
  + Azure SDK 2.9.6+ for Visual Studio
    - <https://www.microsoft.com/en-us/download/details.aspx?id=51657>
  + Azure PowerShell (installed as part of Azure SDK)
  + Reboot after installing the Azure SDK
  + Git command line interface (CLI)

## Lab structure

This lab has two sets of instructions. The first is a high-level set of instructions that is designed for students that have previous experience authoring templates in Azure. The second is a traditional hands-on lab guide that is designed for users that are new to Azure Resource Manger template authoring.

## Help references

|  |  |
| --- | --- |
| Switch deployment slots in Azure web apps | <https://www.visualstudio.com/docs/release/examples/azure/deployment-slots-webapps> |
| App service continuous deployment | <https://www.visualstudio.com/en-us/docs/release/examples/azure/continuous-to-app-service> |
| App service staging environments | <https://azure.microsoft.com/en-us/documentation/articles/web-sites-staged-publishing/> |

## Exercise 0: Environment setup

### Overview

In this lab, you will create a developer environment and download the required files for this course if you do not already have one that meets the requirements.

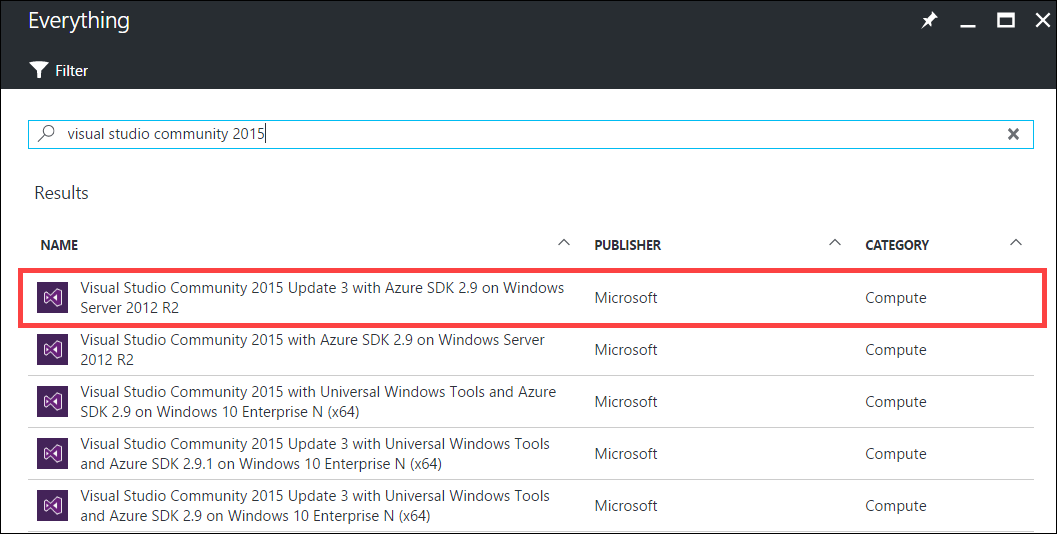
### Prerequisites

* Microsoft Azure subscription: <http://azure.microsoft.com/en-us/pricing/free-trial/>
* Client computer with Windows 7 or later with Visual Studio 2015 - SDK 2.9.6+
  + Ensure you reboot after installing the SDK or Azure PowerShell will not work correctly

#### Task 1: Configure a development environment

If you do not have a machine setup with Visual Studio 2015 Community and Azure SDK 2.9.6+, complete this task.

1. Create a virtual machine in Azure using the **Visual Studio Community 2015 Update 3 with Azure SDK 2.9 on Windows Server 2012 R2** image.



It is highly recommended to use a DS2\_V2 or D2\_V2 instance size for this VM

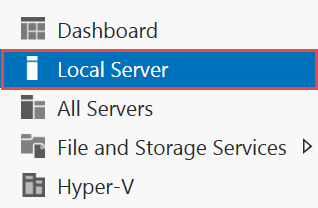
#### Task 2: Disable IE enhanced security

Note: Sometimes this image has IE ESC disabled, and sometimes it does not.

1. On the new VM, you just created click the Server Manager icon.



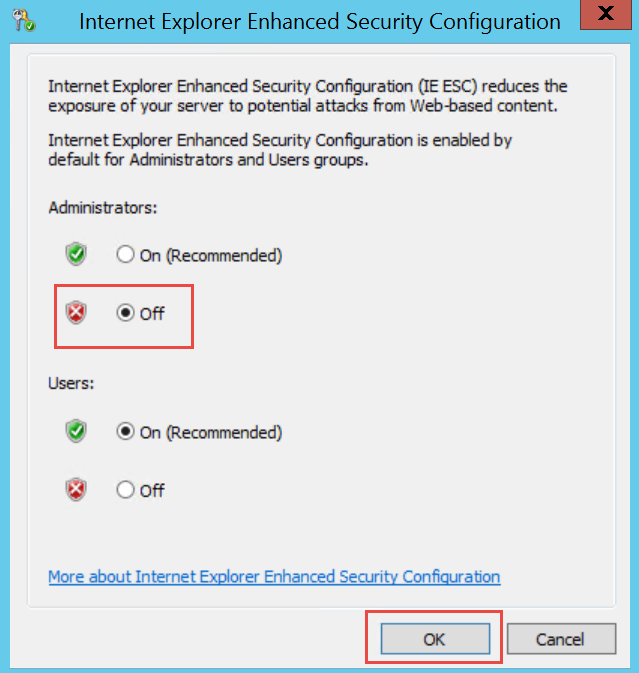
Click Local Server.



1. On the right side of the pane, click **On** by IE Enhanced Security Configuration.

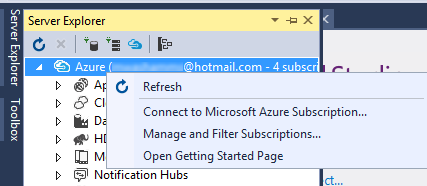


1. Change to **Off** for Administrators and click **OK**.



#### Task 3: Validate connectivity to Azure

1. From within the virtual machine, Launch Visual Studio 2015 and validate that you can login with your Microsoft Account when prompted.
2. Validate connectivity to your Azure subscription. Launch Visual Studio, open Server Explorer from the View menu, and ensure that you can connect to your Azure subscription.

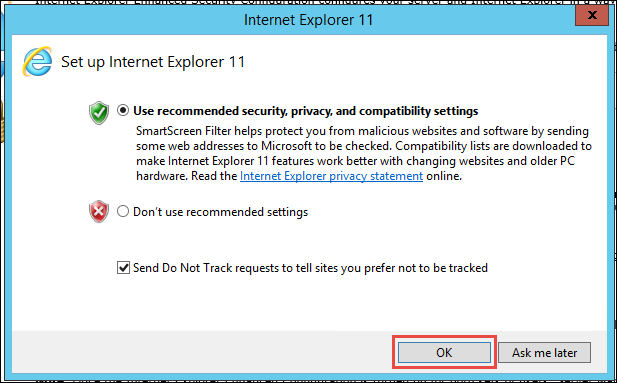
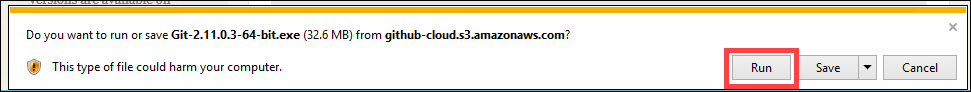
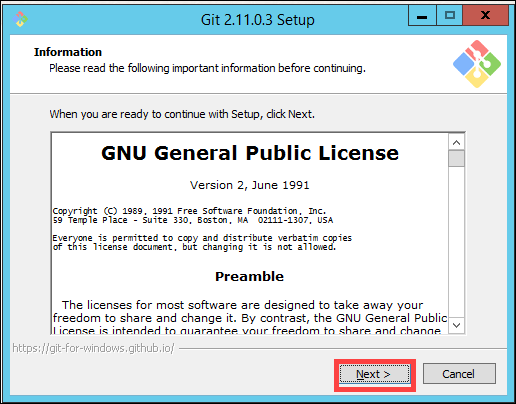
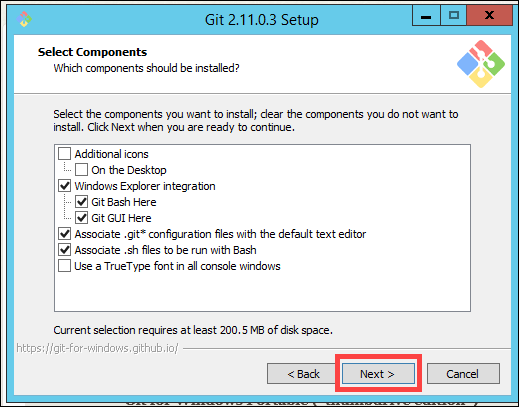
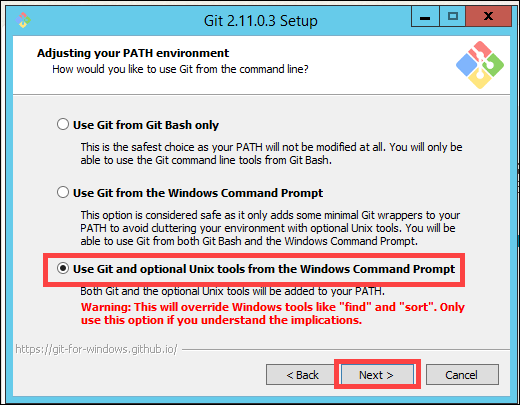
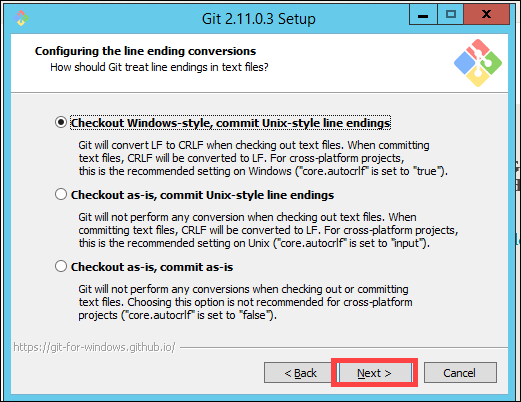
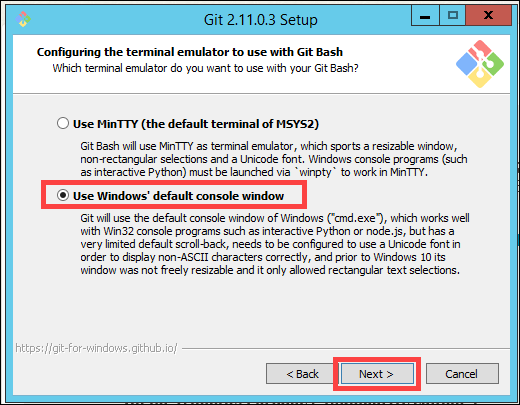
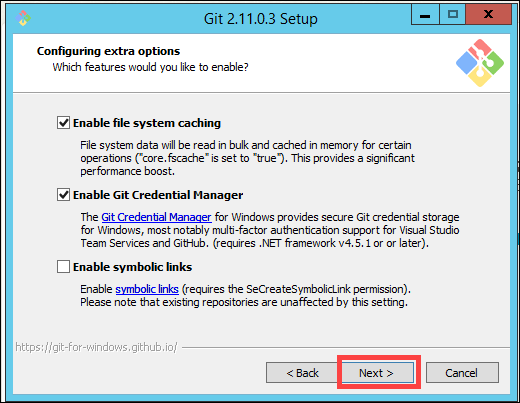
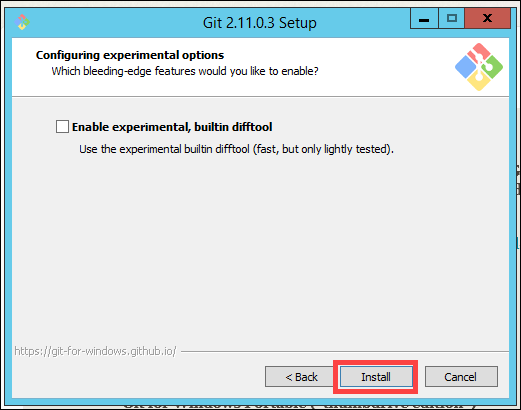
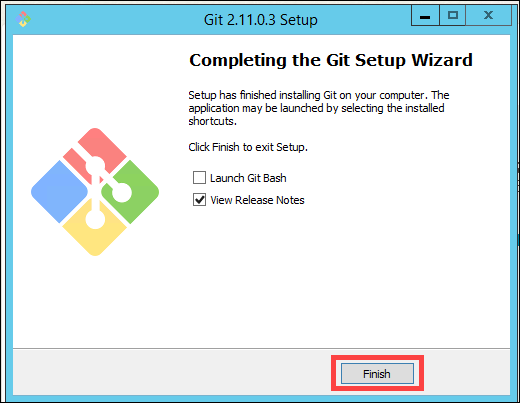


**Download the exercise files**

1. Download the exercise files for the training (from within the virtual machine).
   1. Create a new folder on your computer named **C:\Hackathon**
   2. Download the support files (.zip format), https://opsgilityweb.blob.core.windows.net/cebootcamp-feb-2017/Agile%20Continuous%20Delivery%20Hackathon%20-%20Student%20Files.zip to the new folder.
   3. Extract the contents to the same folder.

Tip: In the labs when adding resources to the Azure Resource Manager template, ensure there are no spaces at the end of the resource names. Visual Studio will include the space in all of the code it generates.

#### Task 3: Download and install Git

1. Open a web browser and navigate to <https://git-scm.com>
2. If you get a prompt about Internet Explorer defaults, just click OK.  
   
3. Find the computer screen icon on the left that says “Latest source Release” and click “Downloads for Windows.” **NOTE: In this screenshot, it shows version 2.11. 0 but you might see a more recent version. Use whatever version is listed on the site.**  
   
4. The download should start; click **Run** in the security prompt.  
   
5. Click **Next**  
   
6. Click **Next**  
   
7. Click “**Use Git and optional Unix tools from the Windows Command Prompt**”, and click **Next**.  
   
8. Leave the next screen at “**Checkout Windows-style, commit Unix-style line endings**”. Click **Next**.  
   
9. On the next screen, click “**Use Windows’ default console window**”, then click **Next**.  
   
10. Leave box check boxes checked and click Install.  
    
11. Click **Install**
12. The Git install should complete and then click **Finish** on the final screen.  
    
13. Open a **command prompt** and type these commands on the command line.

git config --global user.name "“<your name>”"

git config --global user.email <your email>

# Agile continuous delivery hackathon

## Exercise 1: Create an Azure Resource Manager (ARM) template that can provision the web application, SQL database, and deployment slots in a single automated process.

Tailspin Toys has requested an Azure environment consisting of the following resources:

* App Service
  + Web App
    - Auto-scale rule
  + 3 deployment slots (production, test, and development)
* SQL Server
  + SQL Database
* Application Insights

Since this solution is based on Azure Platform-as-a-Service (PaaS) technology, it should take advantage of that platform by utilizing automatic scale for the web app and the SQL Database PaaS service instead of SQL Server virtual machines.

*Tasks to complete*

* Create an Azure Resource Manager (ARM) template file using Visual Studio
  + Add an Azure SQL Database and Server to the template
  + Add a web hosting plan (Azure App Service) to the template
  + Add a web app to the web hosting plan in the template
  + Configure automatic scale for the web app in the template
  + Configure a web app deployment slot called “test” in the template
  + Configure a web app deployment slot called “development” in the template
  + Add Application Insights as a resource to the template
* Create an ARM template parameters file to facilitate the template deployment
* Save the completed template to source control to enable future versioning
* Deploy the completed template to Azure and verify that it is complete

*Exit criteria*

* You have a completed ARM template and parameters file saved and committed to source control
* You have deployed the completed template to Azure

## Exercise 2: Create a Visual Studio team services team project with Git source control

Tailspin Toys has asked you to create a continuous delivery process for their development team. To do that without disrupting their current production application, you are tasked with creating an environment in Azure where this new process and workflow can be proven. Your first task is to create a Visual Studio Team Services Team Project with Git source control where their source code can be maintained.

*Tasks to complete*

* Create Visual Studio Team Services account
* Create Agile Team Project with Git source control
* Upload Tailspin Toys current web app source code with all branches

*Exit criteria*

* You can connect Visual Studio to the Visual Studio Team Services account, clone and view the web app repository and branches (dev, test, master)

## Exercise 3: Configure Azure Application Insights for the App Service

In this exercise, you will connect the Azure Application Insights resource that you created previously with the web app so that it can record diagnostic and analytics information.

*Tasks to complete*

* Collect Application Insights instrumentation key
* Add the instrumentation key to the web app’s configuration
* Commit the changes to source control

*Exit criteria*

* The Application Insights settings in the portal are properly connected to the web app

## Exercise 4: Configure continuous delivery for the App Service and enable automated unit testing

In this exercise you will continue the work you started in the previous tasks by configuring the continuous delivery feature for the App Service web app. You will then propagate code changes to the dev branch and promote them all the way to the test branch through continuous deployment and then to production through a deployment slot swap.

*Tasks to complete*

* Link Visual Studio Team Services to the App Service for the web app
* Enable continuous deployment from Visual Studio Team Services to each App Service deployment slot for its respective source control branch
  + Master deploys to the staging deployment slot
  + Test deploys to the test deployment slot
  + Dev deploys to the dev deployment slot
* Propagate new code across environments and swap staging with production

*Exit criteria*

You have continuous deployment configured for each source control branch to specific deployment slots in the App Service. You can push a change from source control and see that it is published to a live deployment slot. You can swap staging with production to propagate changes all the way into the production App Service deployment slot.

# Agile continuous delivery hackathon answers

## Overview

This portion of the lab is designed to help you if you are blocked or have limited experience with Visual Studio Team Services, App Service, or other required technologies.

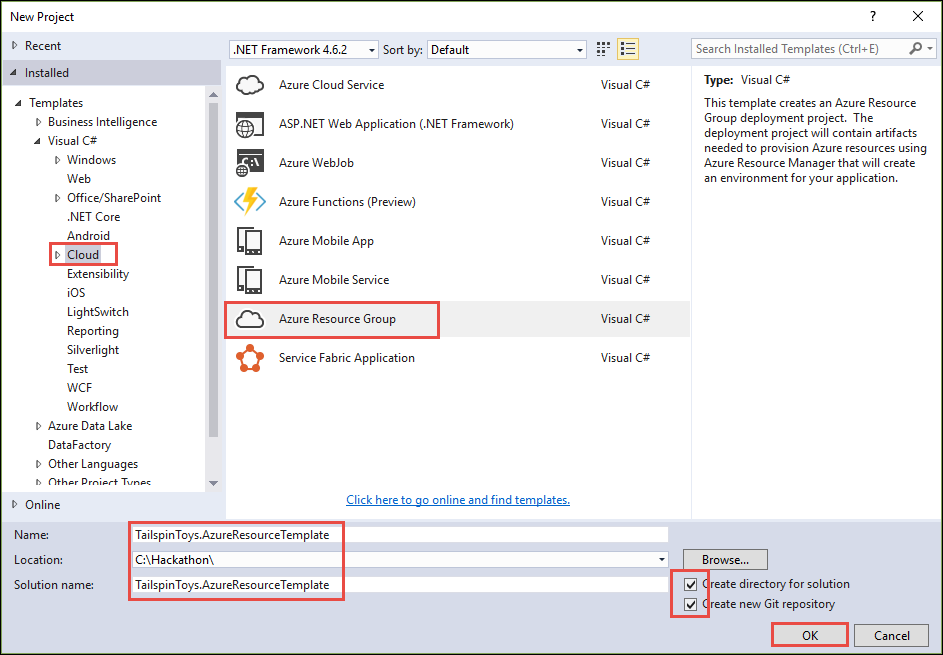
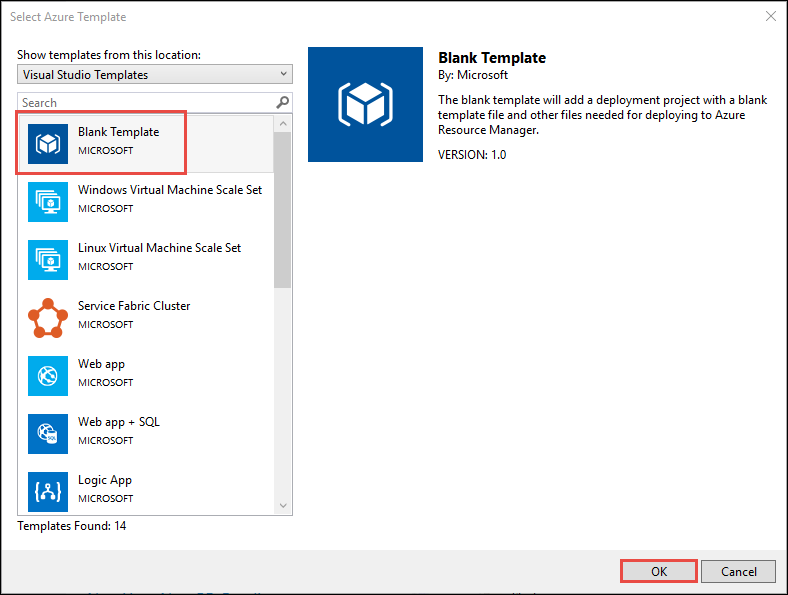
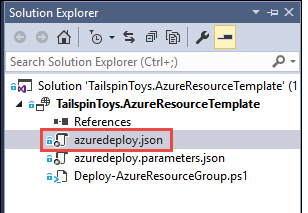
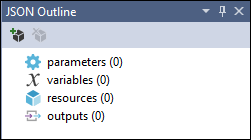
## Exercise 1: Create an Azure Resource Manager (ARM) template that can provision the web application, SQL database, and deployment slots in a single automated process.

Tailspin Toys has requested an Azure environment consisting of the following resources:

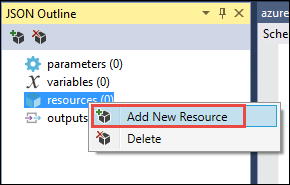
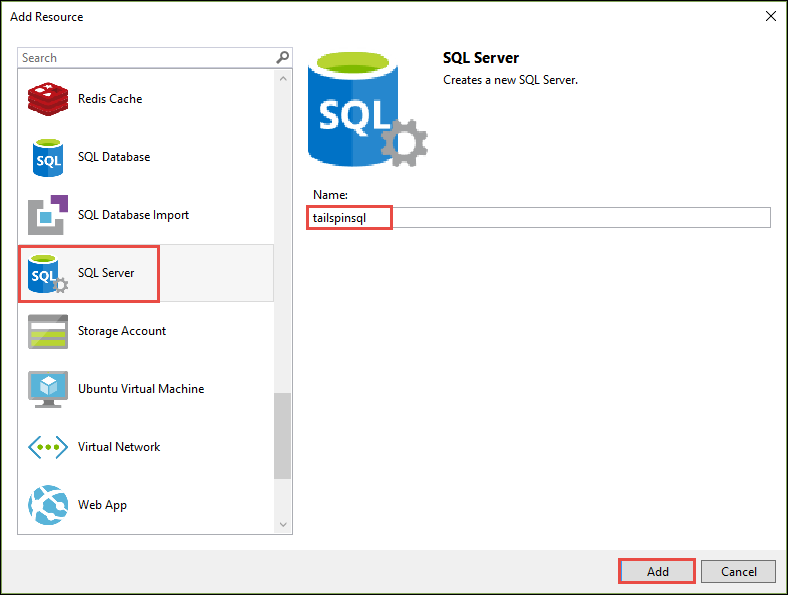
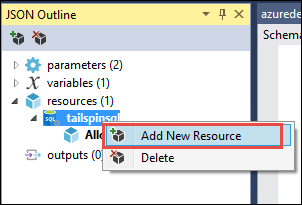
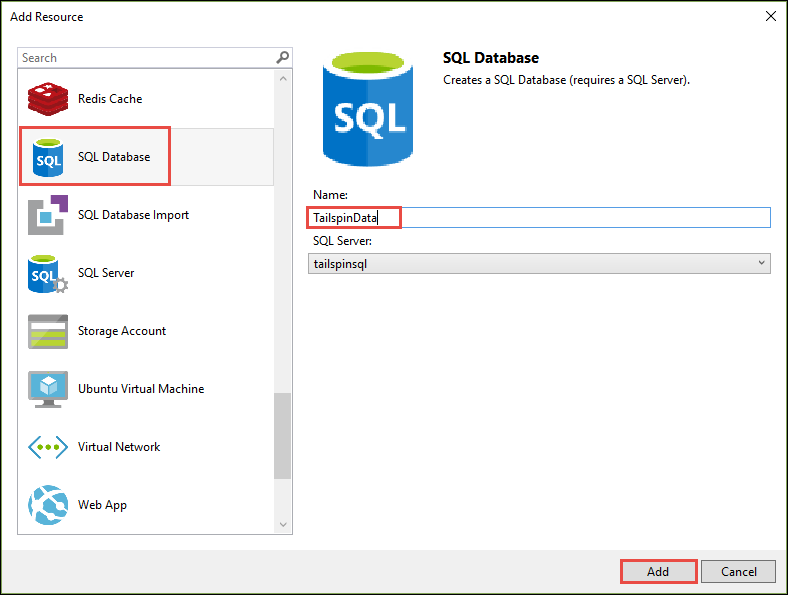
* App Service
  + Web App
    - Auto-scale rule
  + 3 deployment slots (production, test, and development)
* SQL Server
  + SQL Database
* Application Insights

Since this solution is based on Azure Platform-as-a-Service (PaaS) technology, it should take advantage of that platform by utilizing automatic scale for the web app and the SQL Database PaaS service instead of SQL Server virtual machines.

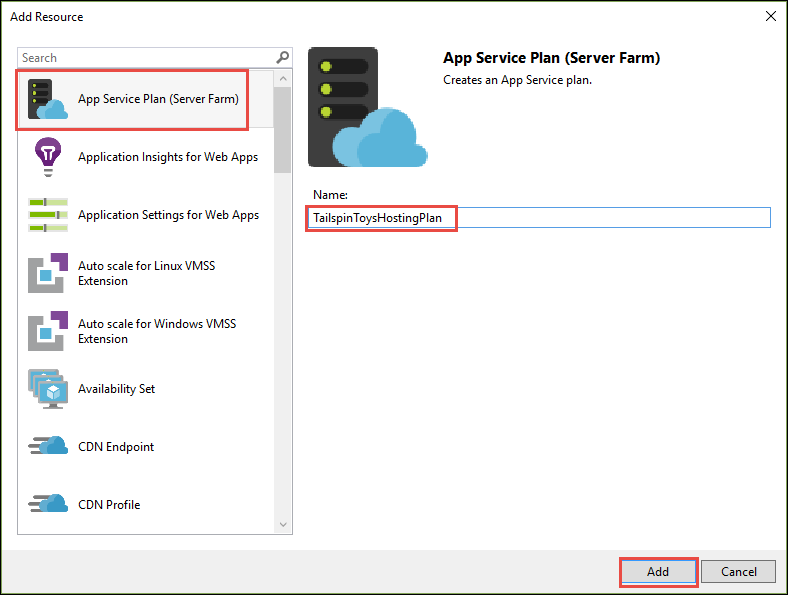
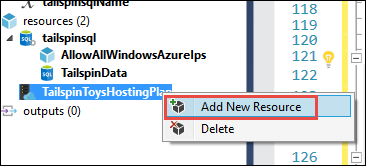
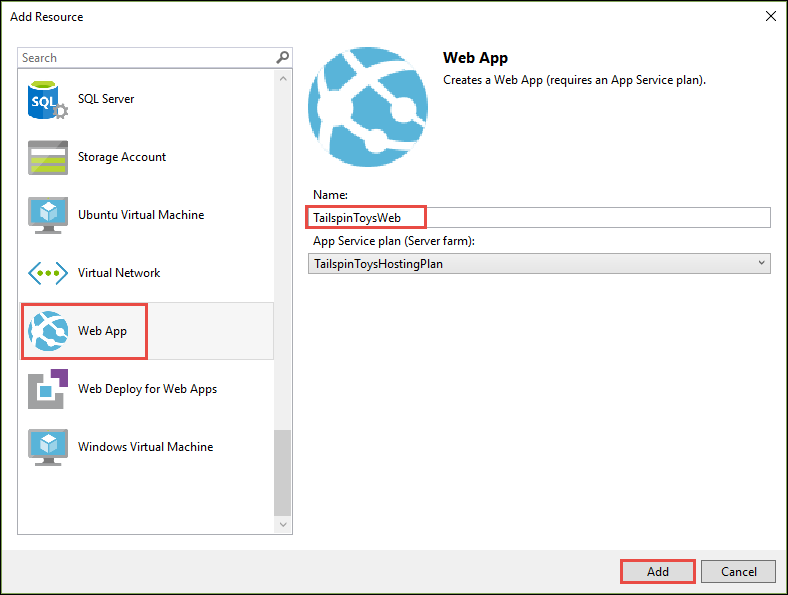
### Task 1: Create an Azure Resource Manager (ARM) template using Visual Studio

1. Open Visual Studio and create a new project of the type Cloud – Azure Resource Group. Name the new project “TailspinToys.AzureResourceTemplate” and save it to C:\Hackathon. Also, make sure that both check boxes are checked on the lower right, as in the screen shot below. When finished, click **OK**.  
   
2. On the next window, click **Blank Template** and then click **OK**.  
   .  
   
3. In the Solution Explorer window, open the azuredeploy.json file by double-clicking it.  
   
4. Then, probably on the left side of the Visual Studio window, open the window called JSON Outline. It will look like this screen shot.  
   
5. Save your files.

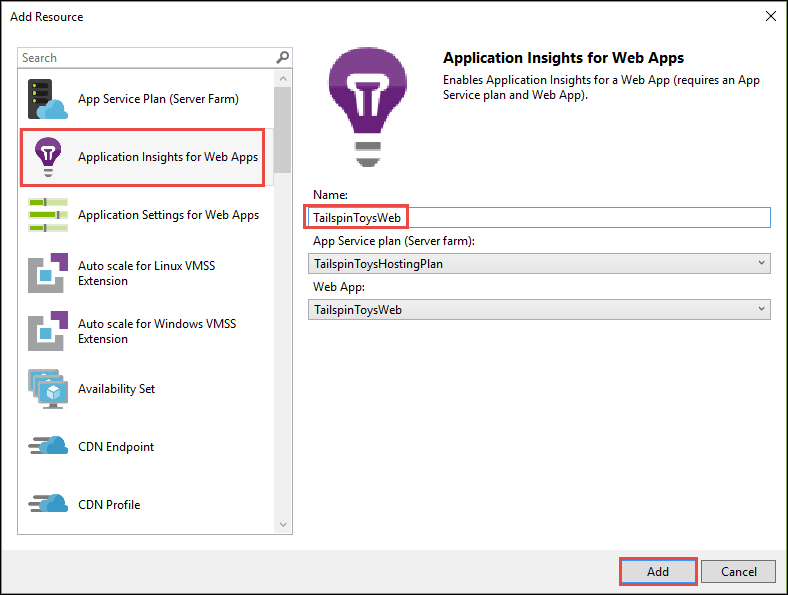
### Task 2: Add an Azure SQL database and server to the template

1. Right-click on the **resources** item in the **JSON Outline** and click **Add New Resource**.  
   
2. Select **SQL Server** and give it a name like “tailspinsql”, then click **Add**.  
     
   
3. Now that the SQL Server has been created as a resource, right-click that SQL Server resource and choose **Add New Resource** so that you can add a database.  
   
4. Choose SQL Database, call it “TailspinData”, make sure that your server is selected in the drop down list below and then click Add.  
   

### Task 3: Add a web hosting plan and web app to the template

1. Add another resource, this time choose **App Service Plan** and call it “TailspinToysHostingPlan”, then click **Add**.  
   
2. Right-click the hosting plan resource and add a new resource underneath it.  
   
3. Choose **Web App**, name it “TailspinToysWeb”, make sure your hosting plan is selected in the drop down list, and then click **Add**.  
   

### Task 4: Add Application Insights to the template

1. Add a new resource to the template, this time choose **Application Insights for Web Apps**. Make sure your correct hosting plan and web app are selected in the boxes. Name the Application Insights resource “TailspinToysWeb” and then click **Add**.  
   
2. Next, you need to add the Application Insights extension to the App Service so that it will be running automatically once the site is deployed. This is going to require some manual code because there is not a wizard for this resource type. Click on the TailspinToysWeb web app resource to locate its JSON code. Then, just below the “properties” property, paste or type in this block of JSON code.

"resources": [

{

"apiVersion": "2015-08-01",

"name": "Microsoft.ApplicationInsights.AzureWebSites",

"type": "siteextensions",

"tags": {

"displayName": "Application Insights Extension"

},

"dependsOn": [

"[resourceId('Microsoft.Web/Sites/', variables('TailspinToysWebName'))]",

"[resourceId('Microsoft.Insights/components/', 'TailspinToysWeb')]"

],

"properties": {

}

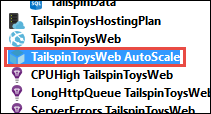
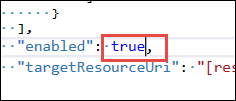
}

]

It will look something like this screen shot.



### Task 5: Configure automatic scale for the web app in the template

1. Click on the resource called “TailspinToysWeb AutoScale” to see its JSON value.  
   
2. In the main window, scroll down a little to find the “enabled” property of the auto scale rule. Change it from “false” to “true”. You can examine the other settings in this JSON value to understand the setting. It defaults to increasing the instance count if the CPU goes above 80% for a while and reduces the instance count if the CPU falls below 60% for a while.  
   

### Task 6: Add a deployment slot for the “test” version of the site

1. Next, you need to add the “test” deployment slot to the web app. This is going to require some manual code because there is not a wizard for this resource type. Click on the TailspinToysWeb web app resource to locate its JSON code. Then, add this code to the “resources” array, just below the element for the application insights extension.

{

"apiVersion": "2015-08-01",

"name": "test",

"type": "slots",

"tags": {

"displayName": "Deployment Slot: test"

},

"location": "[resourceGroup().location]",

"dependsOn": [

"[resourceId('Microsoft.Web/Sites/', variables('TailspinToysWebName'))]"

],

"properties": {

},

"resources": []

}

It will look something like this screen shot.



### Task 7: Add a deployment slot for the “development” version of the site

1. Next, you need to add the “development” deployment slot to the web app. This is going to require some manual code because there is not a wizard for this resource type. Click on the TailspinToysWeb web app resource to locate its JSON code. Then, add this code to the “resources” array, just below the element for the “test” deployment slot.

{

"apiVersion": "2015-08-01",

"name": "development",

"type": "slots",

"tags": {

"displayName": "Deployment Slot: development"

},

"location": "[resourceGroup().location]",

"dependsOn": [

"[resourceId('Microsoft.Web/Sites/', variables('TailspinToysWebName'))]"

],

"properties": {

},

"resources": []

}

It will look something like this screen shot.



### Task 8: Deploy the template to Azure

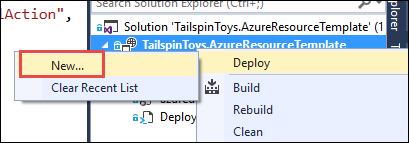
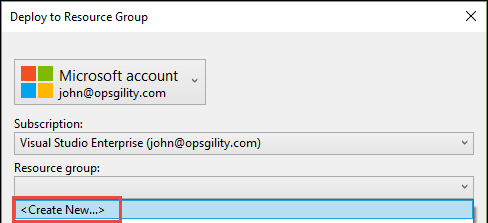
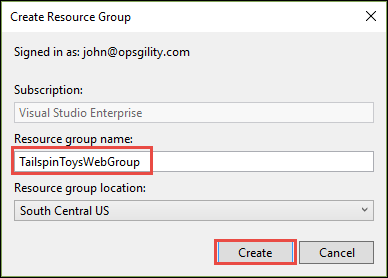
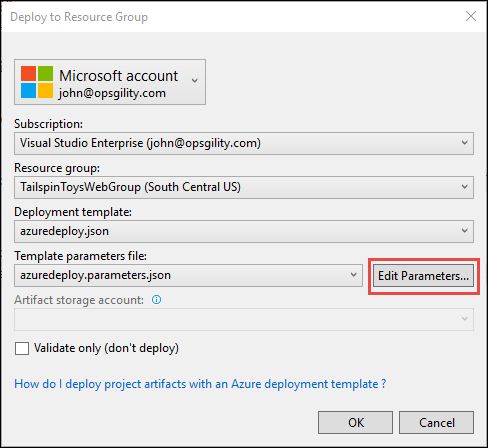
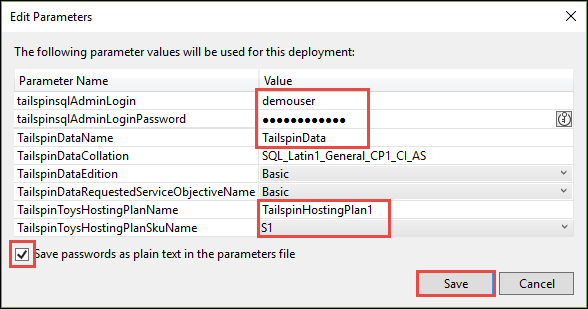
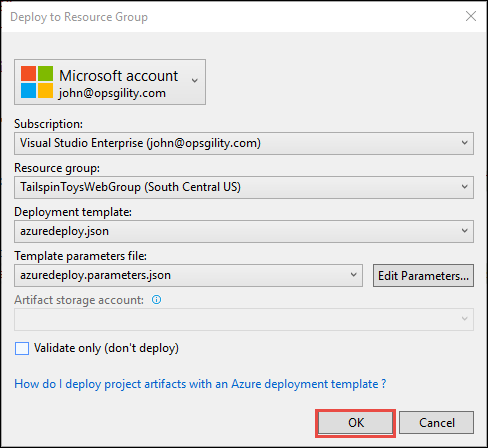
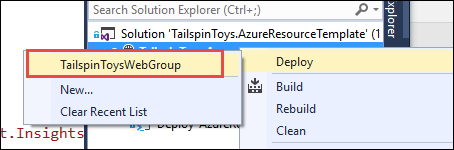
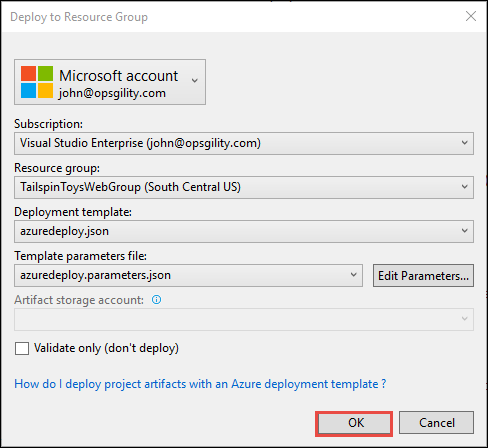
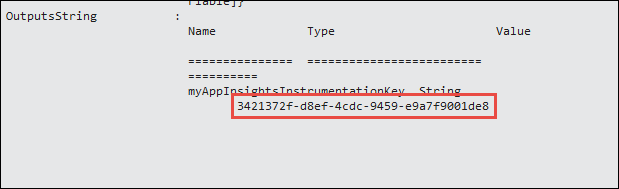
1. First, before you deploy the template, you need to make sure that you can get the instrumentation key from the Application Insights resource because you will need it later. To do this, you can add an output property to the template. Go the “outputs” area of the template and paste or type in this JSON code.

"MyAppInsightsInstrumentationKey": {

"value": "[reference(resourceId('Microsoft.Insights/components', 'TailspinToysWeb'), '2014-04-01').InstrumentationKey]",

"type": "string"

}

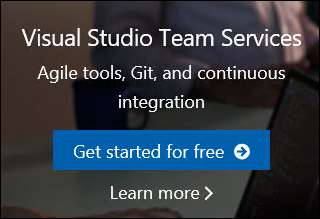
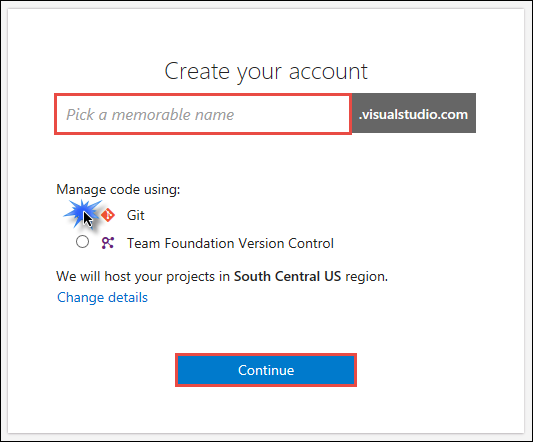
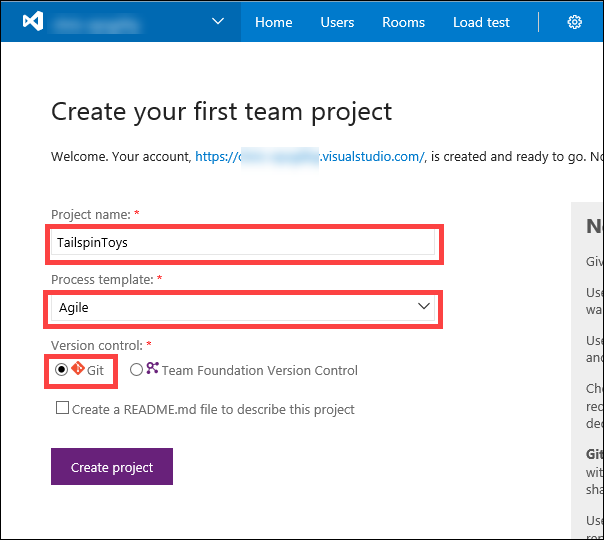
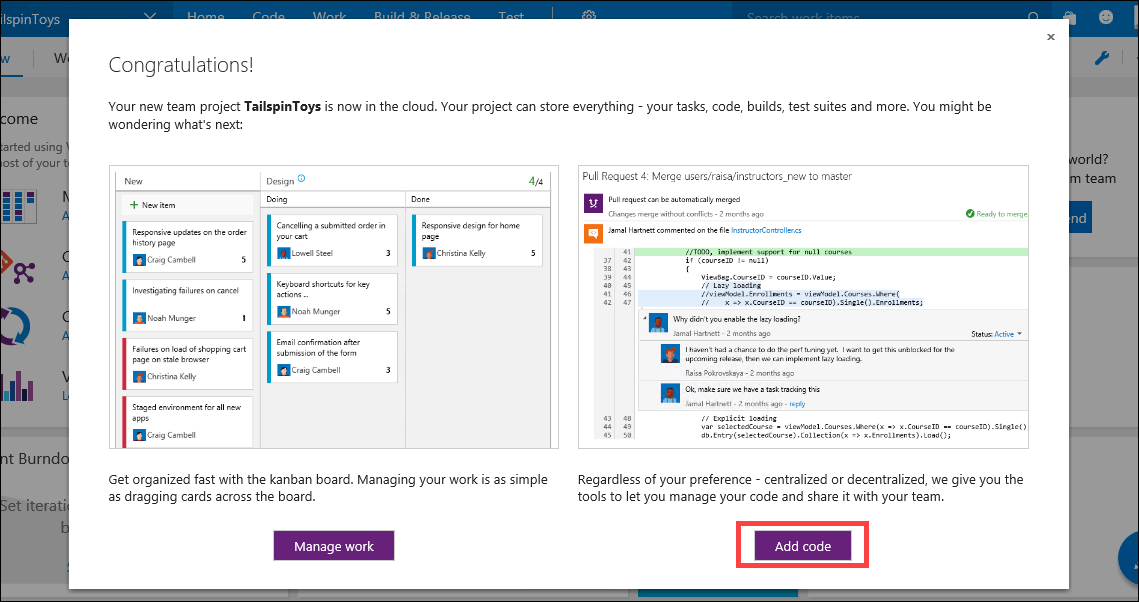
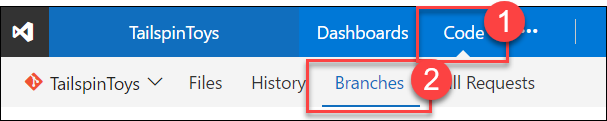
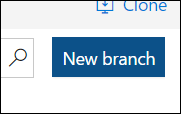
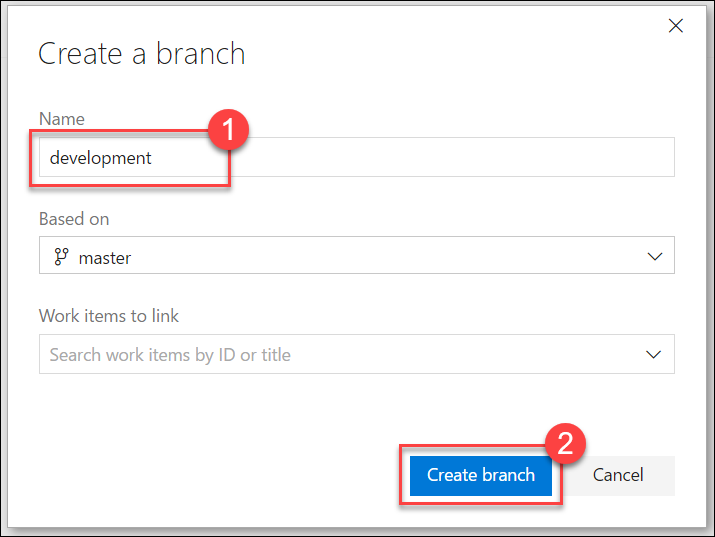
1. Now, save all of your files.
2. Right-click the project in Solution Explorer and choose “Deploy” and then “New…”.  
   
3. Sign in to your Azure account if necessary, and then choose your correct subscription. Under Resource group, choose “Create New…” and create a new resource group for this deployment. You can name it whatever you want, such as “TailspinToysWebGroup”. Choose a location near you.  
     
     
   
4. Once you have the resource group created, click the **Edit Parameters** button.  
   
5. In the next window, pick an admin username, and password for the database, it does not matter what you choose. Then use “TailspinData” for the TailspinDataName value. Call the hosting plan “TailspinHostingPlan1” and choose “S1” for the Sku. See this screen shot for help. When finished, click Save.  
   
6. Then, click the OK button on the deployment window.  
     
   
7. You will probably get a deployment error at the end of the process, in the output window, due to an issue where the Application Insights extension is not deployed in time. This is OK, it most likely did deploy successfully. In this case, simply wait about 2 or 3 minutes and run the deploy again by right-clicking the project again, choosing “Deploy”, but this time choose the existing deployment from the list like this:  
   
8. Then, just click OK because all of the settings are the same.  
     
   
9. This time through, you should see success and you should see an instrumentation key be written out in the output window. Copy this down for a future step in this process. Note that your key will be different from the one shown in this screen shot.  
   

## Exercise 2: Create Visual Studio Team Services Team Project and Git Repository

### Overview

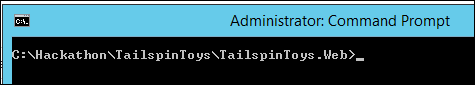
In this exercise, you will create and configure a Visual Studio Team Services account along with an Agile Team Project.

### Task 1: Create Visual Studio Team Services Account

1. Browse to the Visual Studio site at <http://visualstudio.com>
2. If you do not already have an account, click **Get started for free**  
   
3. Authenticate with a Microsoft account.
4. Choose a name for your visualstudio.com account. Choose **Git** for the source code and then click Continue.  
   
5. Close the congratulations window if it appears.  
   
6. On the **Create your first team** project page, set the following values:
   1. Project name: **TailspinToys**
   2. Process template: **Agile**
   3. Version control: **Git**  
       
7. Click **Create project**
8. On the **Congratulations!** window, click **Add Code**.  
     
   
9. Within VTST, on the **Code** screen, click on **Branches.**
10. Click on **New branch**
11. Enter **development** into the name field and click **Create branch**.  
    
12. Repeat steps 10 – 11 to add another branch named **test**

### Task 2: Add the Tailspin Toys source code repositories to Visual Studio Team Services

In this Task, you will configure the Visual Studio Team Services Git repository. You will configure the remote repository using Git and then push the source code up to Visual Studio Team Services through the command line tools.

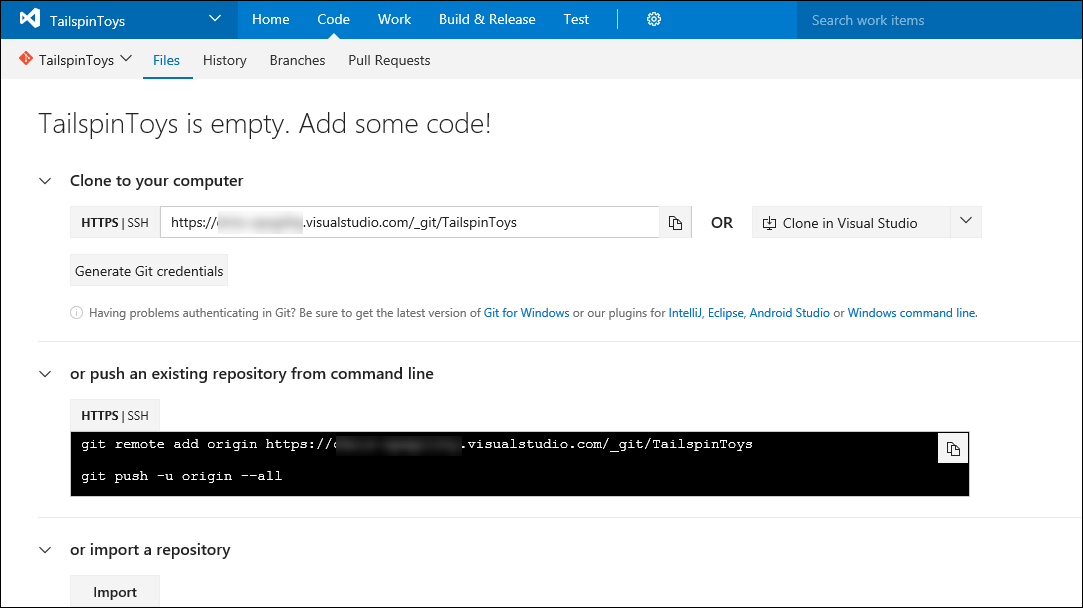
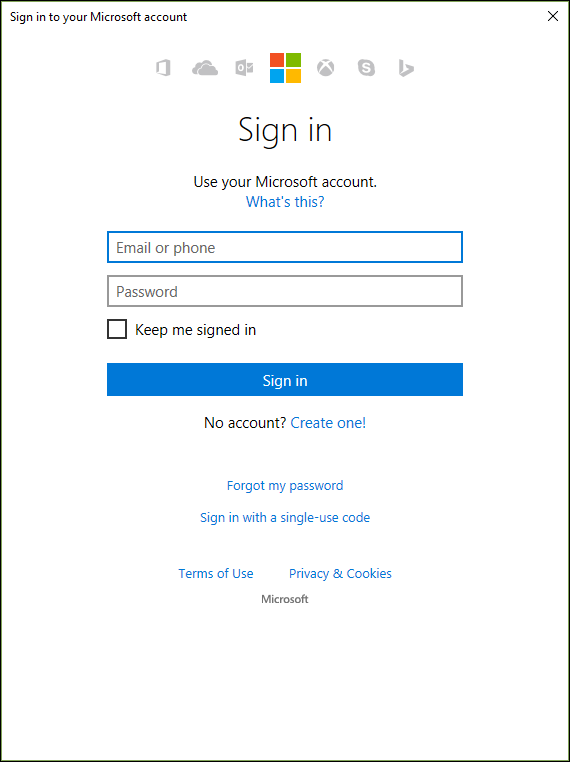
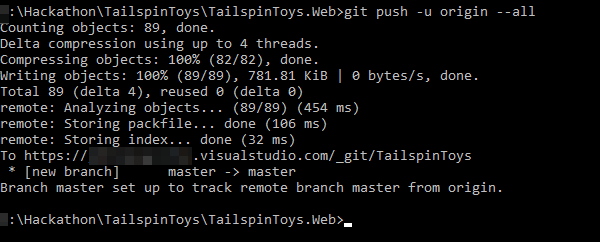
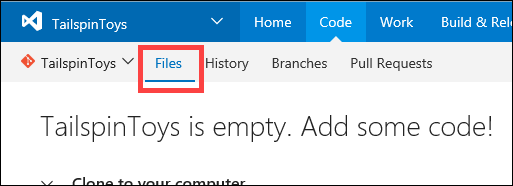
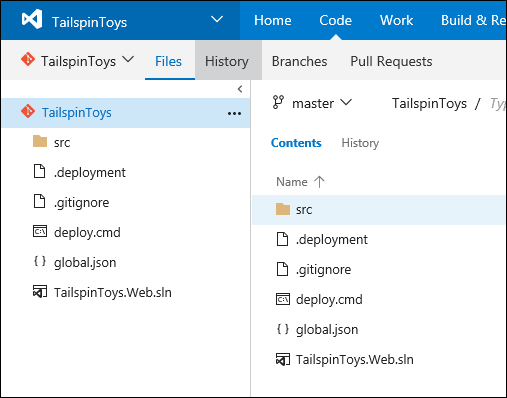
1. In the support files, open a command prompt in the **C:\Hackathon\TailspinToys\TailspinToys.Web** folder.  
   
2. Initialize a local Git repository by running the following commands at the command prompt:

git init

git add \*

git commit -m "adding files"

If a “.git” folder and local repository already exists in the TailspinToys.Web folder, then you’ll need to delete the “.git” folder first before running the above commands to initialize the Git repository.

1. Leave that command prompt window open and switch back to the web browser window for Visual Studio Team Services from the previous Task.  
   
2. Scroll down the page and find the section that provides Git command line instructions.  
     
   
3. Copy the first command under the **Push an existing repository** heading and then paste it in your command window and press the Enter key.  
     
   
4. Next, copy the second command under the **Push an existing repository** heading, then paste it in your command window, and press the Enter key.  
     
   
5. You should get a prompt to login with your Visual Studio Team Services credentials, which should be the same Microsoft account you used to create it.  
   
6. You should see some output similar to this after you have logged in.  
     
   
7. Go back to the web browser window for Visual Studio Team Services and click on the **Files** link.  
   
8. You should see your source code now appearing inside of Visual Studio Team Services.  
     
   

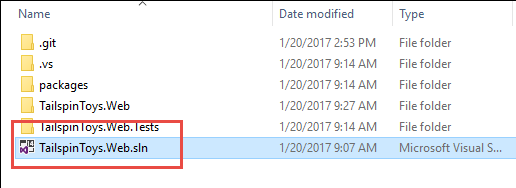
### Summary

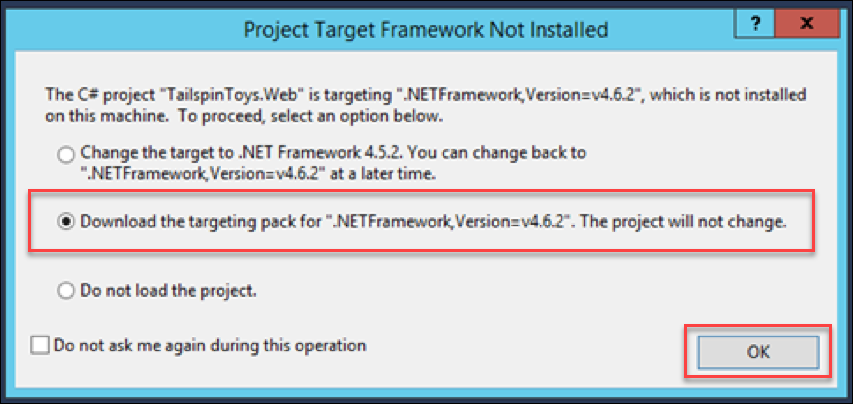
In this exercise, you created a Visual Studio Team Services account where the Tailspin Toys repositories will be stored online. You then configured Git source control for the Tailspin Toys source code and pushed this source code into Visual Studio Team Services.

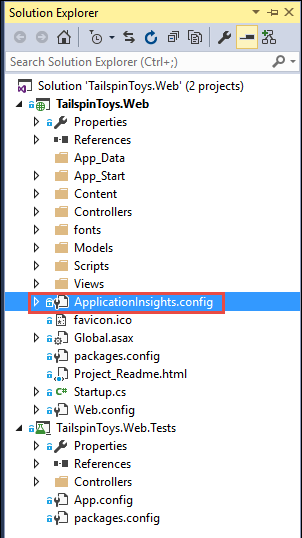
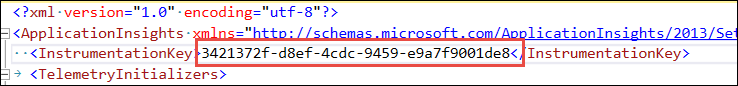
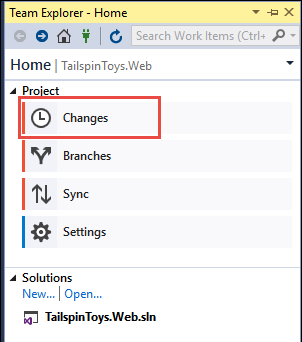
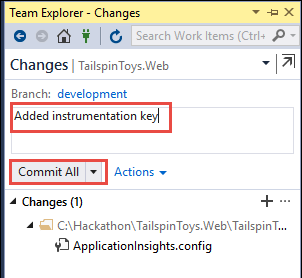
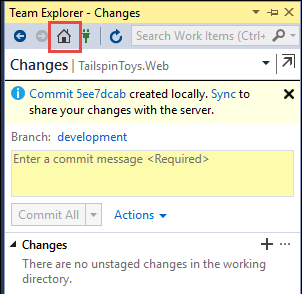
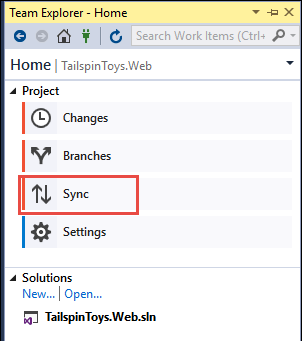
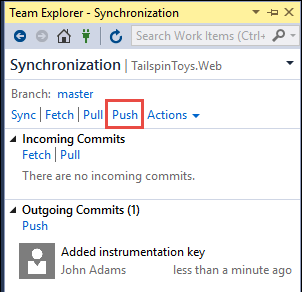
## Exercise 3: Configure Azure Application Insights for the App Service

In this exercise, you will connect the Azure Application Insights resource that you created previously with the web app so that it can record diagnostic and analytics information.

### Task 1: Add the instrumentation key to the web app’s configuration

1. Make sure you have the instrumentation key from the output in the first exercise.
2. Open Visual Studio and then open the solution found in **C:\Hackathon\TailspinToys.Web\TailspinToys.Web.sln**  
   

If a “**Project Target Framework Not Installed**” prompt is displayed, then choose the *“Download the targeting pack for “.NETFramework,Version=v4.6.2”. The project will not change”* option, and click **OK**.  
  
After clicking OK, a browser window will popup where you’ll need to download and install the **.NET Framework 4.6.2 Developer Pack**. After installed, Windows will need to be restarted, then you can re-open the TailspinToys.Web project and continue with the next step.  

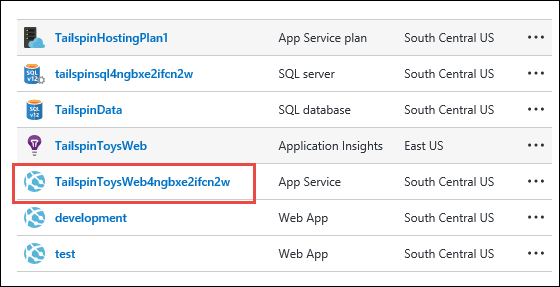
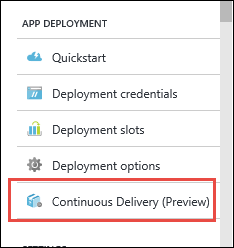
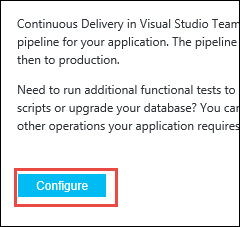
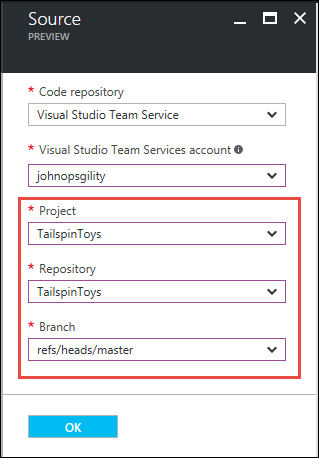
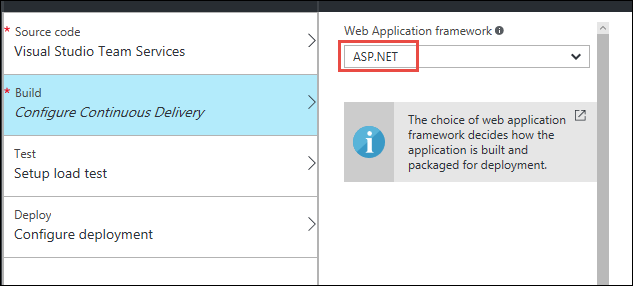
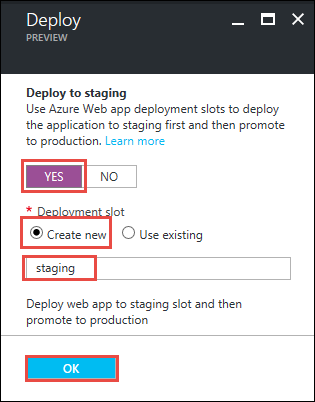
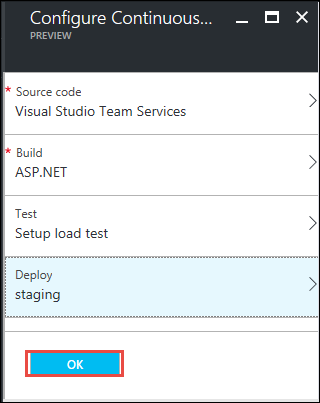
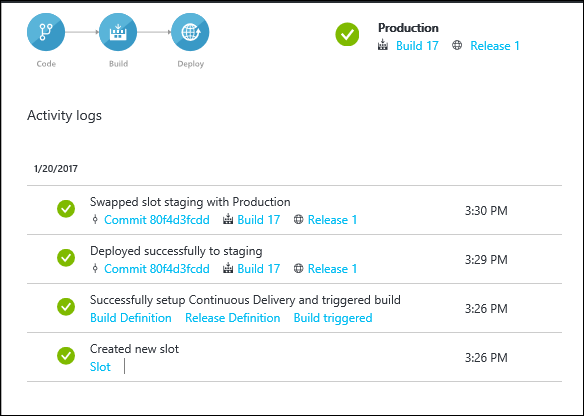

1. Once it is open, use the Solution Explorer to open the file called ApplicationInsights.config.  
   
2. Near the top of the file, you will see a location to paste in the instrumentation key. Paste it in and save your files.  
   
3. Now, you need to push this change up to the repository. Click on the **Team Explorer** window, and then click on the **Changes** button.  
   
4. You will see the ApplicationInsights.config file listed under the pending changes. In the comment box, type “Added instrumentation key” and then click on Commit All.  
   
5. Then, click the little house icon near the top.  
     
   
6. Then, click the **Sync** button.  
   
7. Click on the Push link to push your changes up to the remote repository. If you are prompted to log in, log in with your Azure account.  
     
   

## Exercise 4: Configure Continuous Delivery with unit tests for the App Service

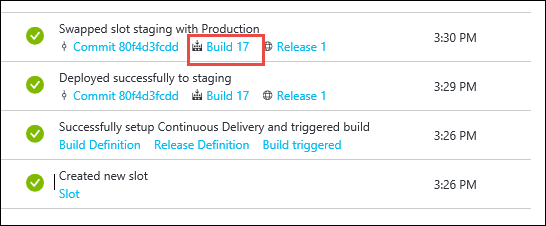
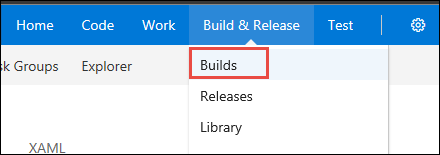
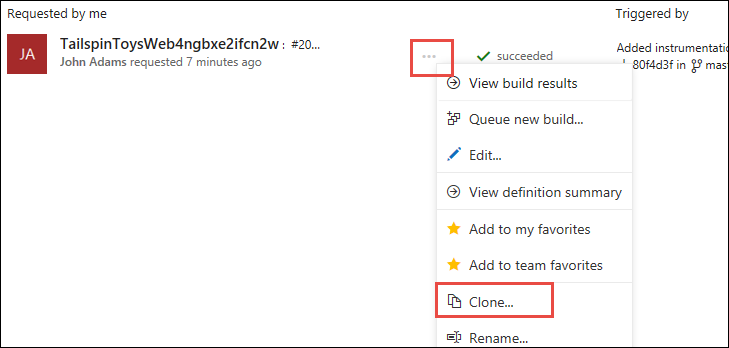
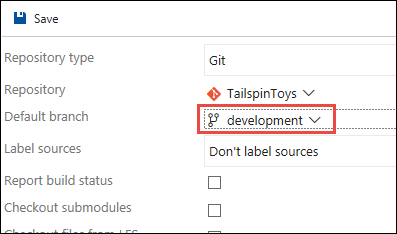
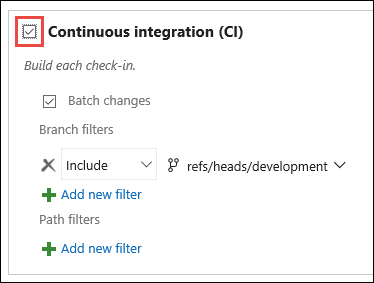
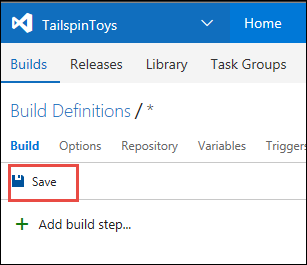
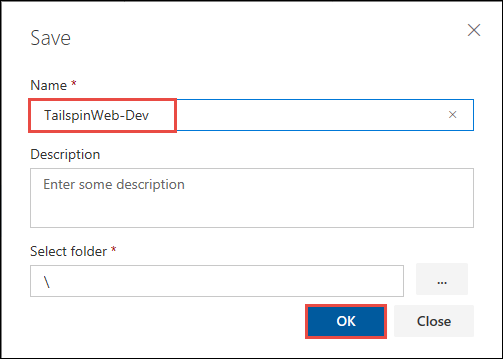
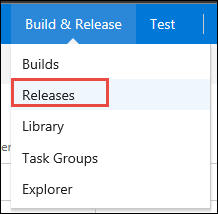
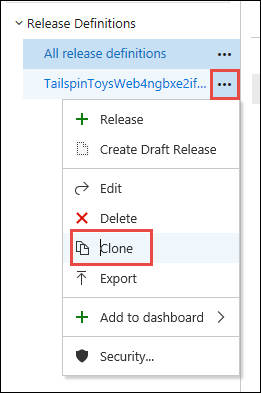
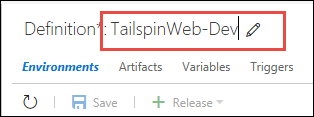
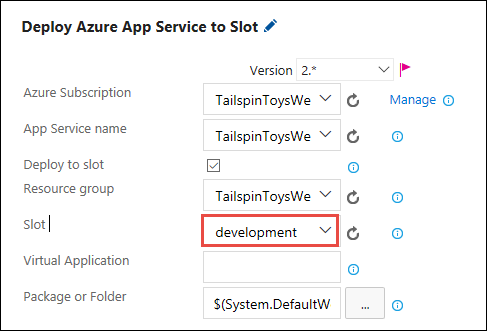
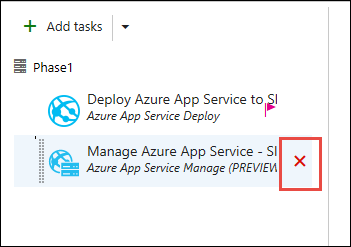
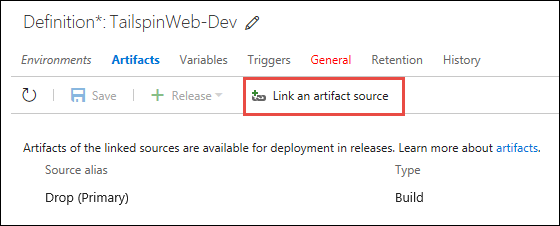
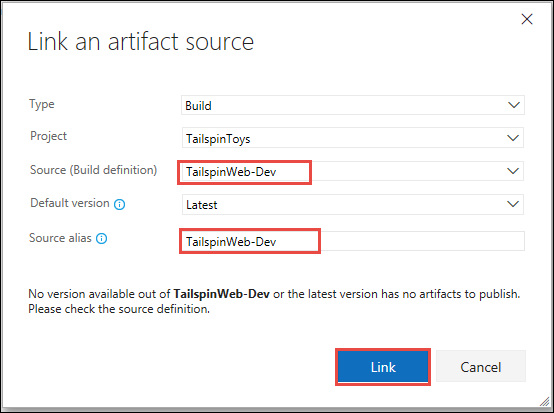
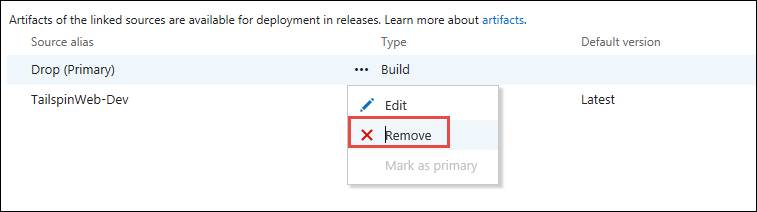
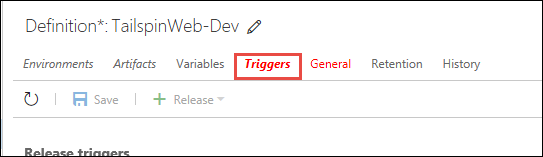
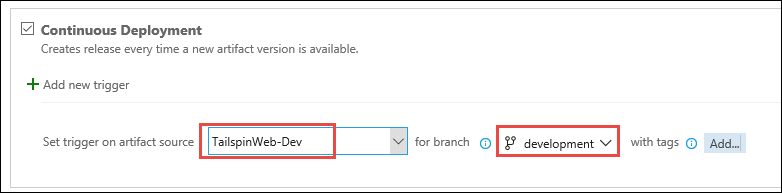
### Overview

In this exercise, you will continue the work you started in the previous tasks by configuring the continuous delivery feature for the App Service web app.

### Task 1: Configure continuous delivery from Visual Studio Team Services to the web app

1. Open the resource group in Azure that contains your web app and then click on the web app that you created in the first exercise.  
   
2. When it opens, click on the Continuous Delivery item in the list of links.  
   
3. On the page that opens, click Configure.  
     
   
4. Click the Source code link at the top. In the **Code repository**, chose “Visual Studio Team Service”. Then, choose your account name, the TailspinToys project, and repository, and choose the master branch. Then, click **OK** at the bottom.  
   
5. Click on the Build link, and choose “ASP.NET”. Then click **OK**.  
     
   
6. Skip the test link, and click on the Deploy link instead. Click **YES** on the deploy to staging option. Choose “create new” and create a slot called “staging”.  
   
7. Then, click OK at the bottom of the continuous delivery window.  
     
   
8. The system will automatically connect to source control, build the code in the master branch, run the tests, and deploy the code to the production deployment slot. After a few minutes, you will see some output like this. In this case, with the default settings, since all the builds and tests passed, it automatically swapped the staging and production slots.  
   

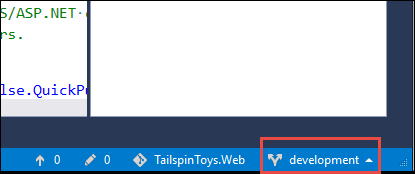
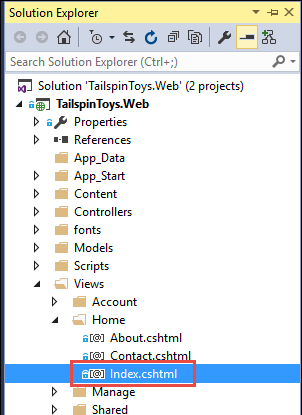
### Task 2: Configure continuous delivery for the development and test slots.

1. From the same screen as the last step, click on the link with the build number. This will take you to the builds section of Visual Studio Team Services.  
   
2. Near the top of the page, hover over “Build & Release” and then choose “Builds”.  
     
   
3. Click the ellipsis next to the existing build definition, and choose “Clone”.  
   
4. Click on the “Repository” link, and change the Default branch from “master” to “development”.  
   
5. Click on “Triggers” and click on “Continuous integration (CI)”.  
     
   
6. Then, click on the Save button near the top-left.  
   
7. Give it a name like “TailspinWeb-Dev” and click OK.  
     
   
8. Next, go back to the “Build & Release” link and this time choose **Releases**.  
   
9. Find the release definition on the left, click the ellipsis again, and choose **Clone**.  
     
   
10. Click the name at the top and change it to “TailspinWeb-Dev”.  
    
11. On the main screen, change the **Slot** from “staging” to “development”.  
      
    
12. Delete the second step of the build by clicking the red “X” symbol next to it. We do not want an auto-swap for the development build process.  
    
13. Click on the Artifacts link near the top.  
      
    
14. Then, click “Link an artifact source”.  
    
15. On the window that pops up, change the build definition to your development build, then click **Link**.  
    
16. Next, remove the “Drop (Primary”) artifact from the list.  
    
17. Then, choose the “Triggers” link at the top.  
      
    
18. Then, choose the new “TailspinWeb-Dev” artifact and the “development” source control branch.  
    
19. Next, click on the “General” link at the top.  
    
20. In the “Release name format” box type this value:

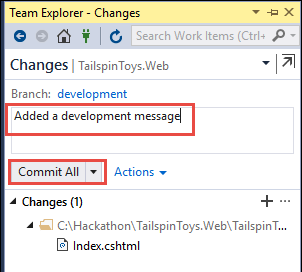
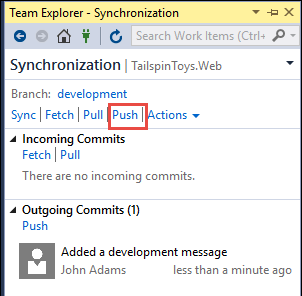
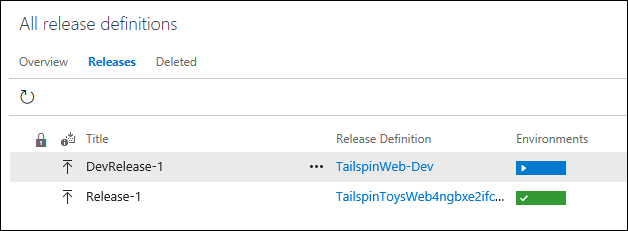
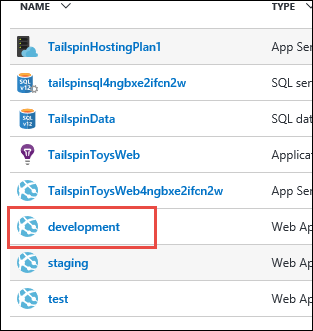
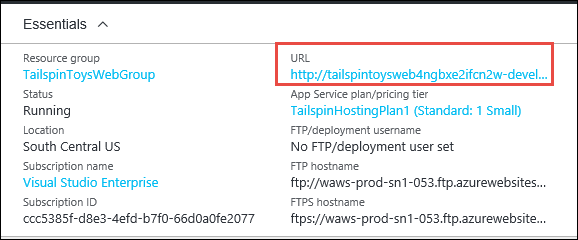
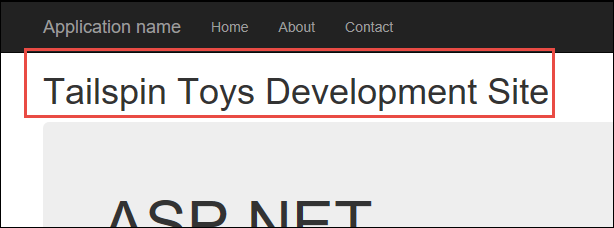
DevRelease-$(rev:r)

1. Then, click the **Save** button. Click **OK** at the next prompt.  
   
2. Repeat steps 1-22 for the “test” deployment slot.
3. Now, with all of these configured, you can push code changes to master, which will trigger a build, with unit tests, deploy to the staging slot and automatically swap with the production slot if everything was successful. You can push code to the development branch, which will automatically build, with unit tests, and deploy to the development deployment slot. You can also push code to the test branch, which will automatically build, with unit tests, and deploy to the test deployment slot.

### Task 3: Test the deployment process

1. Open Visual Studio and the TailspinToys.Web solution.
2. At the bottom-right of the window, click the branch and change it to “development”.  
   
3. Then, in Solution Explorer, open the Index.cshtml view under the Views/Home directory.  
     
   
4. Above the existing <div> element, add this code. Save your files.

<h1>Tailspin Toys Development Site</h1>

1. Click on Team Explorer, click Changes, and commit the change with a brief description.  
   
2. Then, go to the Sync section and push your changes.  
     
   
3. Now, navigate back to the Visual Studio Team Services web portal and go the builds page. You should see something like this, showing that it found your change and started a build.  
   
4. Once it succeeds, go to the releases section of this Visual Studio Team Services portal, and check the release process. You should see the DevRelease in progress like in this screen shot.  
   
5. Once it completes, go back to the Azure portal, find your resource group, and click on the development slot.  
   
6. Click the hyperlink at the top to see the site. Your changes should be live.  
     
   
7. You can see your text, which is only in this branch, at the top.  
   

### Summary

In this hackathon lab, you created deployment slots inside of an Azure App Service web app that were designed to accommodate multiple branches of source code. You then configured continuous deployment for each slot from a specific branch of a Git repository in Visual Studio Team Services.