DevOps Kata

**Infrastucture as code**

Last updated: 11/06/2016



1. **TABLE OF CONTENT**

[Overview 3](#_Toc466271646)

[Prerequisites 3](#_Toc466271647)

[Exercises 3](#_Toc466271648)

[Exercise 1: Create ARM Project 5](#_Toc466271649)

[Task 1: Create a team project 5](#_Toc466271650)

[Task 2: Create an ARM Project 6](#_Toc466271651)

[Task 2: Customize the template 9](#_Toc466271652)

[Task 3: Deploy to Azure 13](#_Toc466271653)

[Exercise 2: Deploy code with infrastructure 19](#_Toc466271654)

[Task 1: Add code project and redeploy 19](#_Toc466271655)

## Overview

The infrastructure for your application is typically made up of many components – maybe a virtual machine, storage account, and virtual network, or a web app, database, database server, and 3rd party services. You do not see these components as separate entities, instead you see them as related and interdependent parts of a single entity. You want to deploy, manage, and monitor them as a group. Azure Resource Manager enables you to work with the resources in your solution as a group. You can deploy, update, or delete all the resources for your solution in a single, coordinated operation. You use a template for deployment and that template can work for different environments such as testing, staging, and production. Resource Manager provides security, auditing, and tagging features to help you manage your resources after deployment

### Resource Manager provides several benefits:

### You can deploy, manage, and monitor all the resources for your solution as a group, rather than handling these resources individually.

### You can repeatedly deploy your solution throughout the development lifecycle and have confidence your resources are deployed in a consistent state.

### You can manage your infrastructure through declarative templates rather than scripts.

### You can define the dependencies between resources so they are deployed in the correct order.

### You can apply access control to all services in your resource group because Role-Based Access Control (RBAC) is natively integrated into the management platform.

### You can apply tags to resources to logically organize all the resources in your subscription.

### You can clarify your organization's billing by viewing costs for a group of resources sharing the same tag.

### Resource Manager provides a new way to deploy and manage your solutions. If you used the earlier deployment model and want to learn about the changes, see Understanding Resource Manager deployment and classic deployment.

### Prerequisites

1. In order to complete the lab
2. 1. Log on to your Visual Studio / MSDN subscription and create or use your own 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

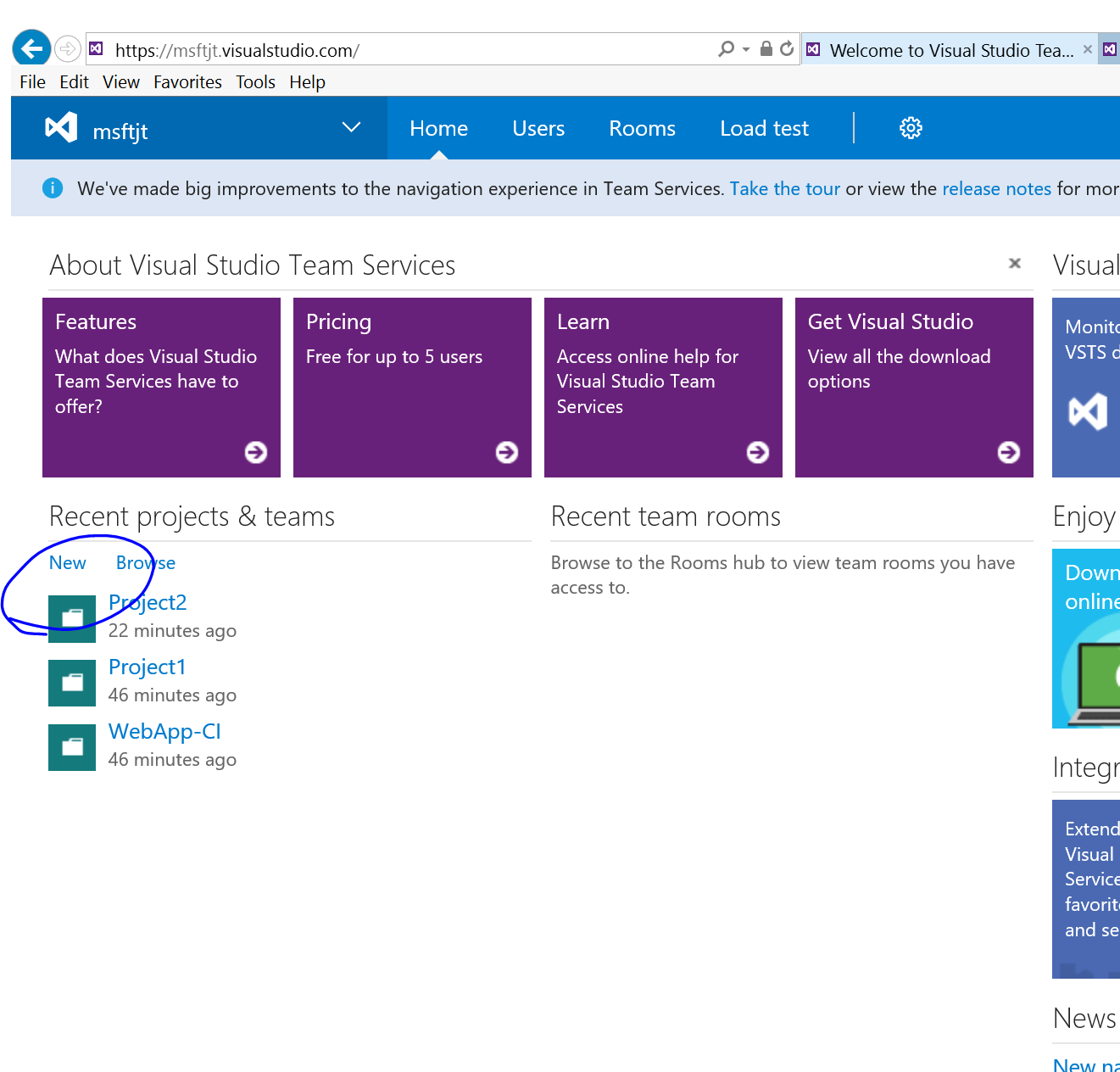
### Exercises

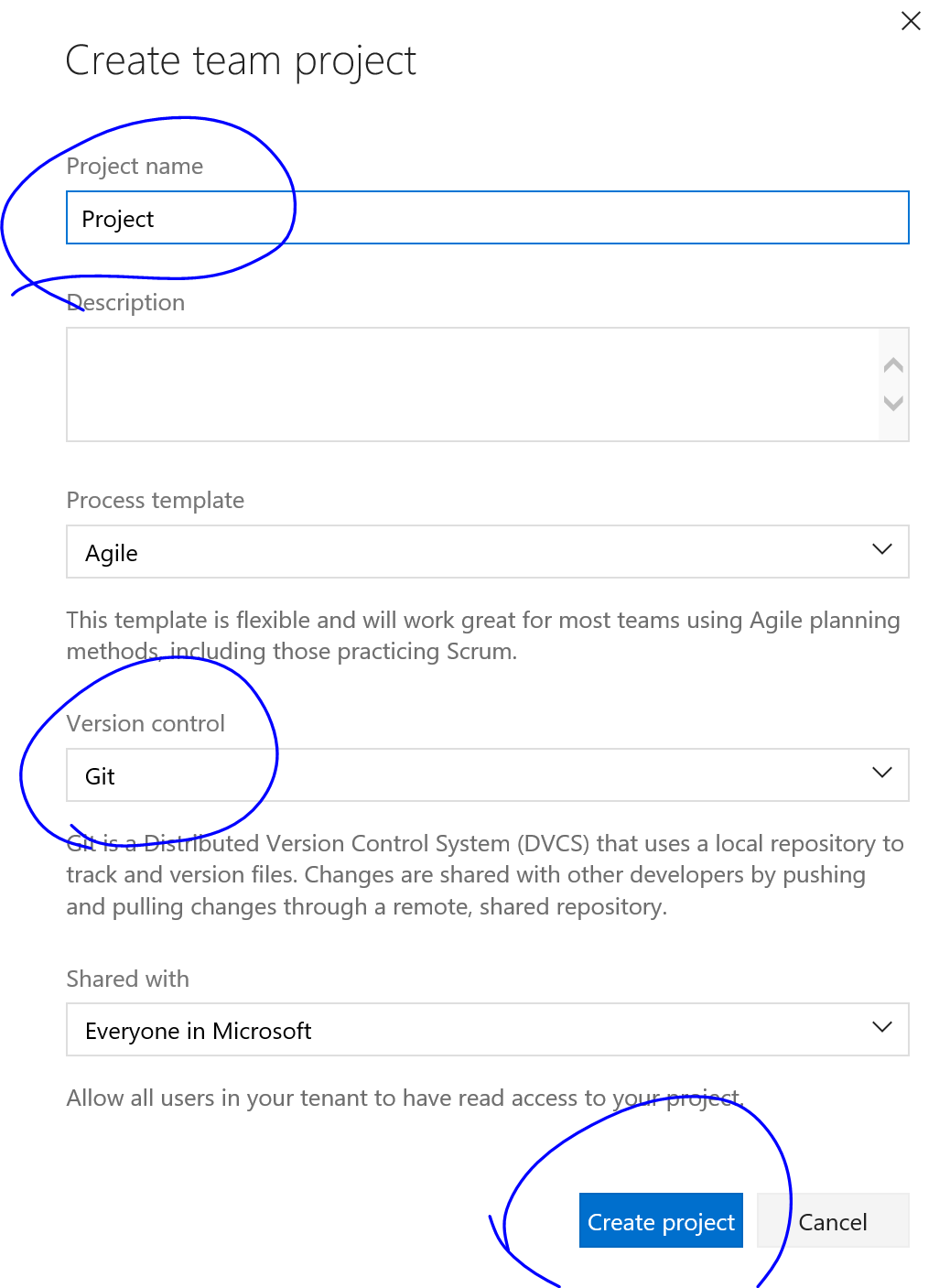
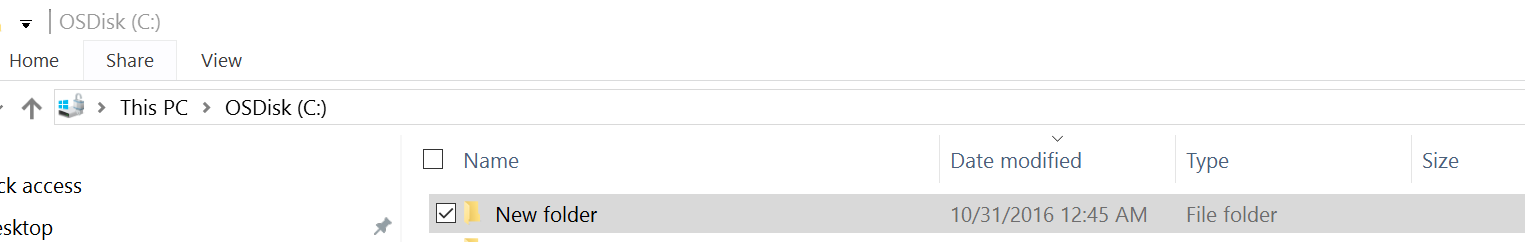
* 1. This hands-on lab includes the following exercises:
  2. Create an ARM template
  3. Customize an ARM template
  4. Deploy an ARM template
  5. Add Code and redeploy
  6. Estimated time to complete this exersize: **15 minutes**.

Exercise 1: Create ARM Project

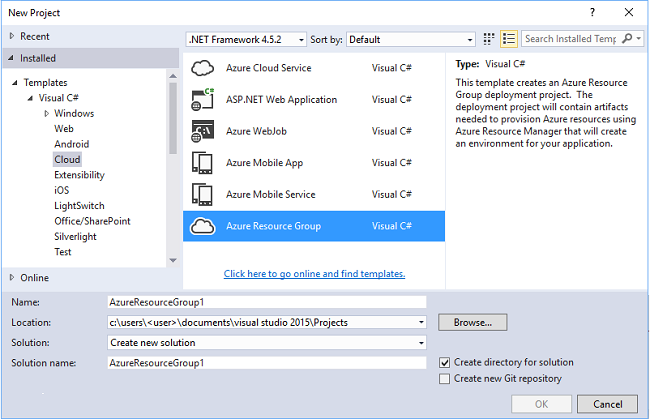
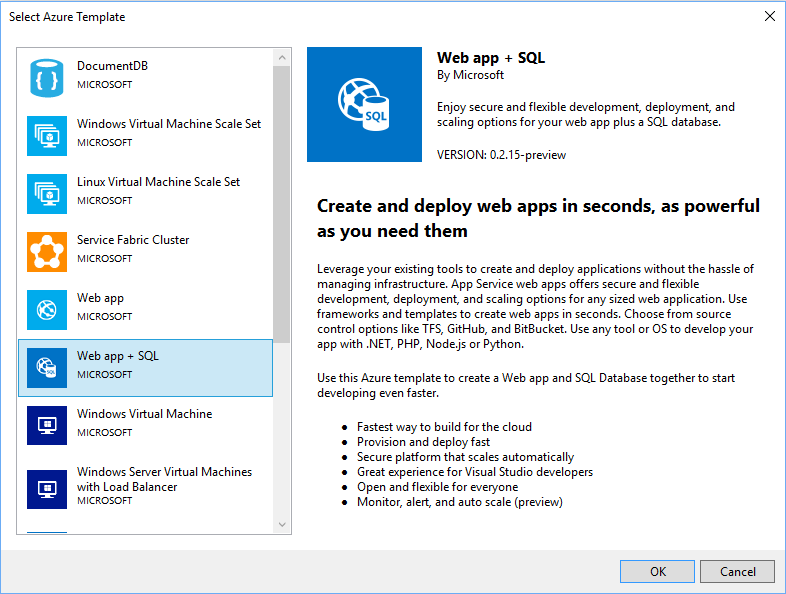
#### Task 1: Create a team project

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

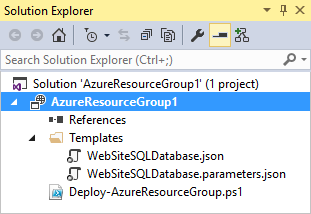
#### Task 2: Create an ARM Project

1. In Visual Studio, choose **File**, **New Project**, choose **C#** or **Visual Basic**. Then choose **Cloud**, and then choose **Azure Resource Group** project.
2. Choose the template that you want to deploy to Azure Resource Manager. Notice there are many different options based on the type of project you wish to deploy. For this topic, choose the **Web app + SQL** template.

The template you pick is just a starting point; you can add and remove resources to fulfill your scenario.

Visual Studio creates a resource group deployment project for the web app and SQL database.

1. To see what you created, expand the nodes in the deployment project.

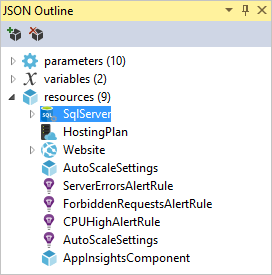


#### Task 2: Customize the template

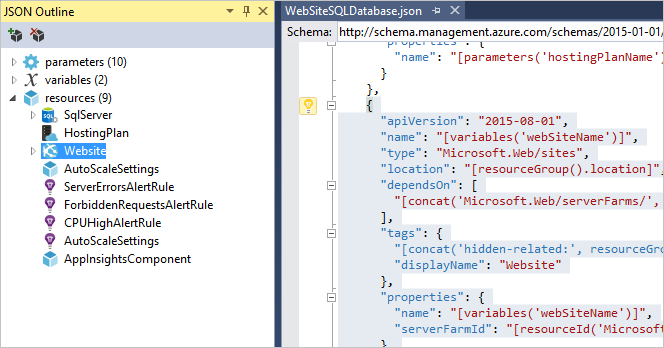
You can customize a deployment project by modifying the JSON templates that describe the resources you want to deploy. JSON stands for JavaScript Object Notation, and is a serialized data format that is easy to work with. The JSON files use a schema that you reference at the top of each file. If you want to understand the schema, you can download and analyze it. The schema defines what elements are valid, the types and formats of fields, the possible values of enumerated values, and so on. To learn about the elements of the Resource Manager template, see [Authoring Azure Resource Manager templates](https://azure.microsoft.com/en-us/documentation/articles/resource-group-authoring-templates/).

To work on your template, open **WebSiteSQLDatabase.json**.

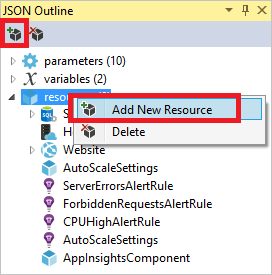
The Visual Studio editor provides tools to assist you with editing the Resource Manager template. The **JSON Outline** window makes it easy to see the elements defined in your template.



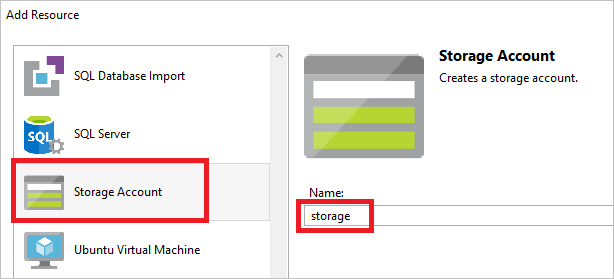
Selecting any of the elements in the outline takes you to that part of the template and highlights the corresponding JSON.



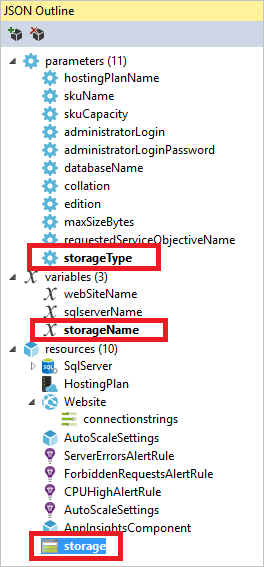
You can add a resource by either selecting the **Add Resource** button at the top of the JSON Outline window, or by right-clicking **resources** and selecting **Add New Resource**.



For this tutorial, select **Storage Account** and give it a name. Provide a name that is no more than 11 characters, and only contains numbers and lower-case letters.



Notice that not only was the resource added, but also a parameter for the type storage account, and a variable for the name of the storage account.



The **storageType** parameter is pre-defined with allowed types and a default type. You can leave these values or edit them for your scenario. If you do not want anyone to deploy a **Premium\_LRS** storage account through this template, remove it from the allowed types.

Copy to clipboardCopy

"storageType": {

"type": "string",

"defaultValue": "Standard\_LRS",

"allowedValues": [

"Standard\_LRS",

"Standard\_ZRS",

"Standard\_GRS",

"Standard\_RAGRS"

]

}

Visual Studio also provides intellisense to help you understand what properties are available when editing the template. For example, to edit the properties for your App Service plan, navigate to the **HostingPlan** resource, and add a value for the **properties**. Notice that intellisense shows the available values and provides a description of that value.



You can set **numberOfWorkers** to 1.

Copy to clipboardCopy

"properties": {

"name": "[parameters('hostingPlanName')]",

"numberOfWorkers": 1

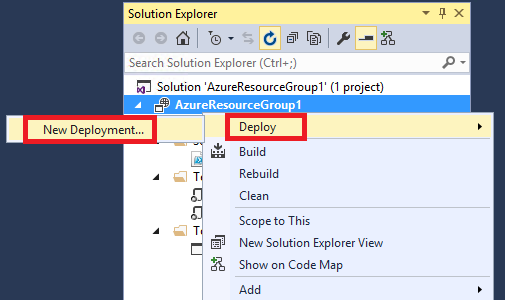
}

#### 

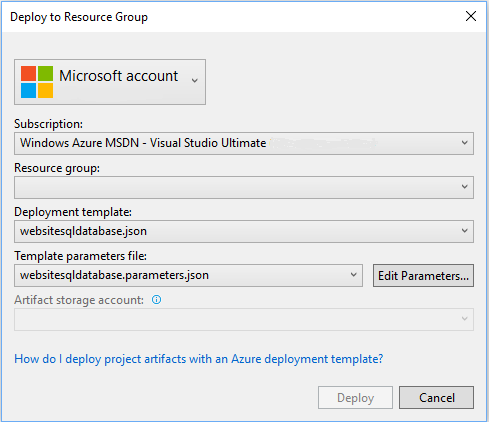
#### Task 3: Deploy to Azure

You are now ready to deploy your project. When you deploy an Azure Resource Group project, you deploy it to an Azure resource group. The resource group is a logical grouping of resources that share a common lifecycle.

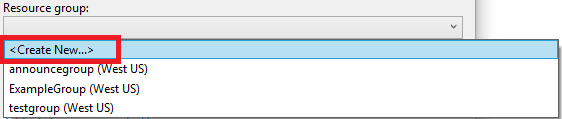
1. On the shortcut menu of the deployment project node, choose **Deploy** > **New Deployment**.



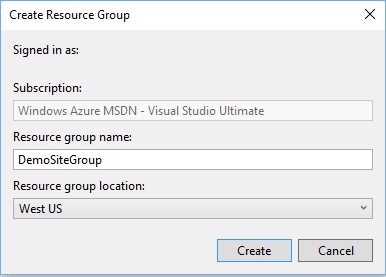
The **Deploy to Resource Group** dialog box appears.



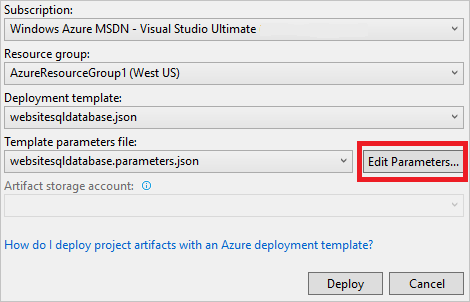
1. In the **Resource group** dropdown box, choose an existing resource group or create a new one. To create a resource group, open the **Resource Group** dropdown box and choose **Create New**.



The **Create Resource Group** dialog box appears. Give your group a name and location, and select the **Create** button.



1. Edit the parameters for the deployment by selecting the **Edit Parameters** button.



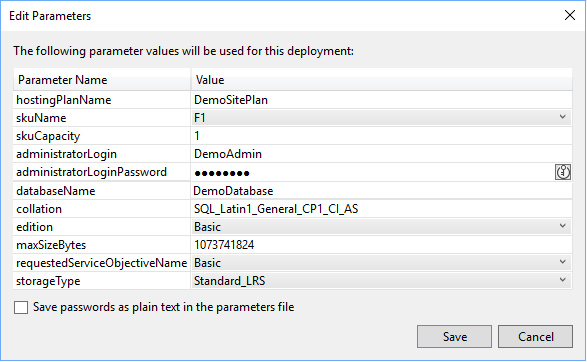
1. Provide values for the empty parameters and select the **Save** button. The empty parameters are **hostingPlanName**, **administratorLogin**, **administratorLoginPassword**, and **databaseName**.

**hostingPlanName** specifies a name for the [App Service plan](https://azure.microsoft.com/en-us/documentation/articles/azure-web-sites-web-hosting-plans-in-depth-overview/) to create.

**administratorLogin** specifies the user name for the SQL Server administrator. Do not use common admin names like **sa** or **admin**.

The **administratorLoginPassword** specifies a password for SQL Server administrator. The **Save passwords as plain text in the parameters file** option is not secure; therefore, do not select this option. Since the password is not saved as plain text, you will need to provide this password again during deployment.

**databaseName** specifies a name for the database to create.



1. Choose the **Deploy** button to deploy the project to Azure. A PowerShell console opens outside of the Visual Studio instance. Enter the SQL Server administrator password in the PowerShell console when prompted. **Your PowerShell console may be hidden behind other items or minimized in the task bar.** Look for this console and select it to provide the password.

##### **Note:**

Visual Studio may ask you to install the Azure PowerShell cmdlets. You need the Azure PowerShell cmdlets to successfully deploy resource groups. If prompted, install them.

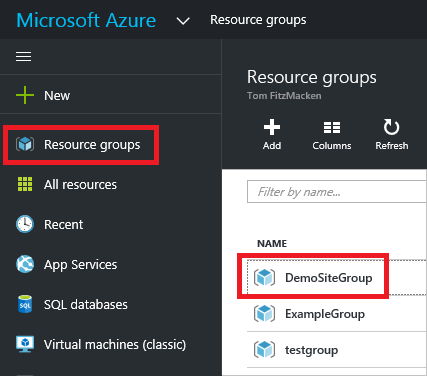
1. The deployment may take a few minutes. In the **Output** windows, you see the status of the deployment. When the deployment has finished, the last message indicates a successful deployment with something similar to:

Copy to clipboardCopy

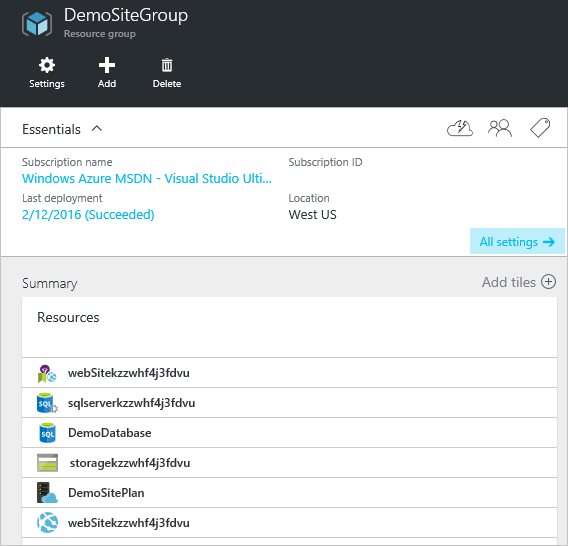
...

18:00:58 - Successfully deployed template 'c:\users\user\documents\visual studio 2015\projects\azureresourcegroup1\azureresourcegroup1\templates\websitesqldatabase.json' to resource group 'DemoSiteGroup'.

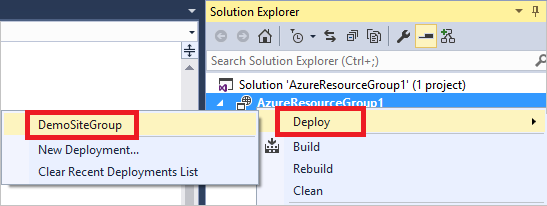
1. In a browser, open the [Azure portal](https://portal.azure.com/) and sign in to your account. To see the resource group, select **Resource groups** and the resource group you deployed to.



1. You see all the deployed resources. Notice that the name of the storage account is not exactly what you specified when adding that resource. The storage account must be unique. The template automatically adds a string of characters to the name you provided to provide a unique name.



1. If you make changes and want to redeploy your project, choose the existing resource group from the shortcut menu of Azure resource group project. On the shortcut menu, choose **Deploy**, and then choose the resource group you deployed.

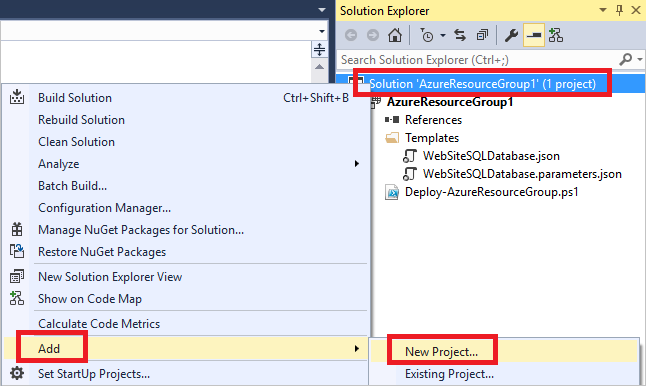


Exercise 2: Deploy code with infrastructure

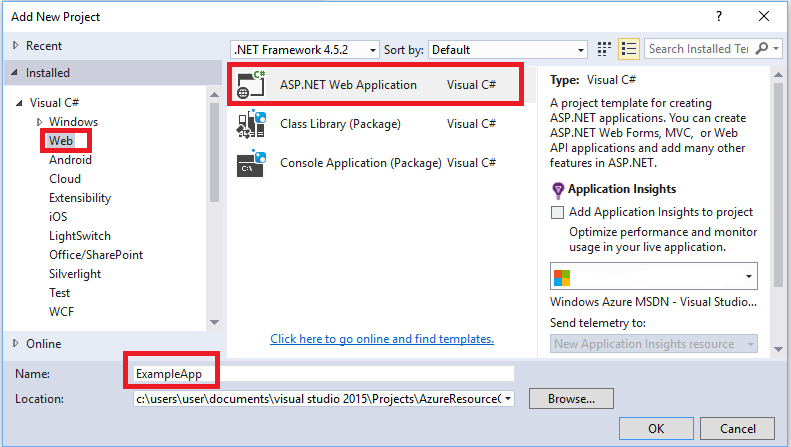
#### Task 1: Add code project and redeploy

At this point, you have deployed the infrastructure for your app, but there is no actual code deployed with the project. This topic shows how to deploy a web app and SQL Database tables during deployment. If you are deploying a Virtual Machine instead of a web app, you want to run some code on the machine as part of deployment. The process for deploying code for a web app or for setting up a Virtual Machine is almost the same.

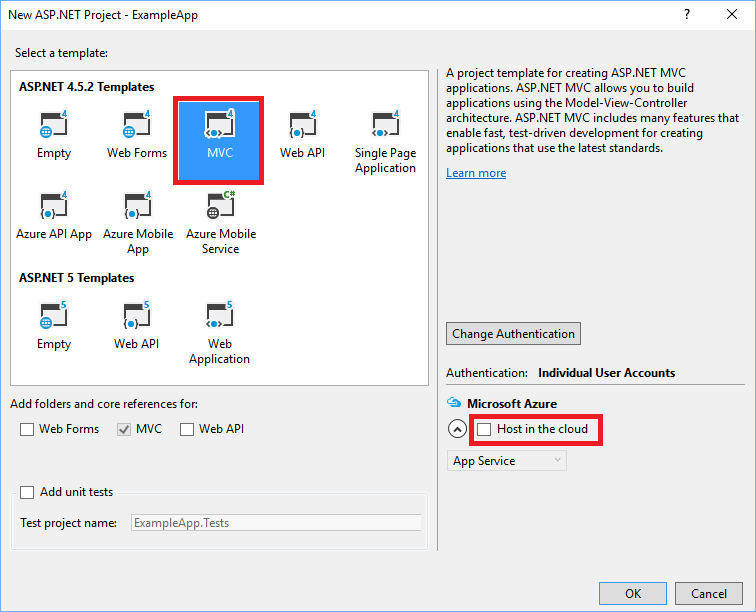
1. Add a project to your Visual Studio solution. Right-click the solution, and select **Add** > **New Project**.



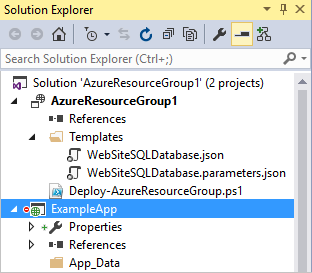
1. Add an **ASP.NET Web Application**.



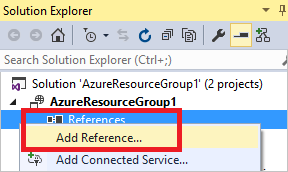
1. Select **MVC** and clear the field for **Host in the cloud** because the resource group project performs that task.



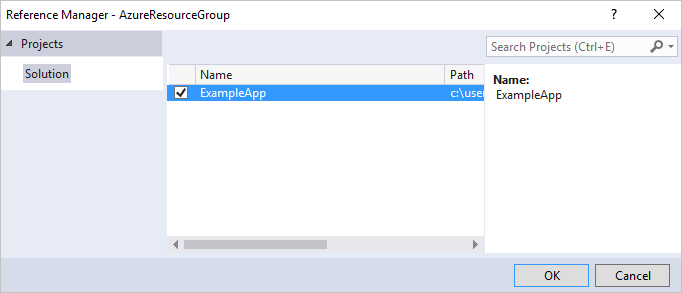
1. After Visual Studio creates your web app, you see both projects in the solution.



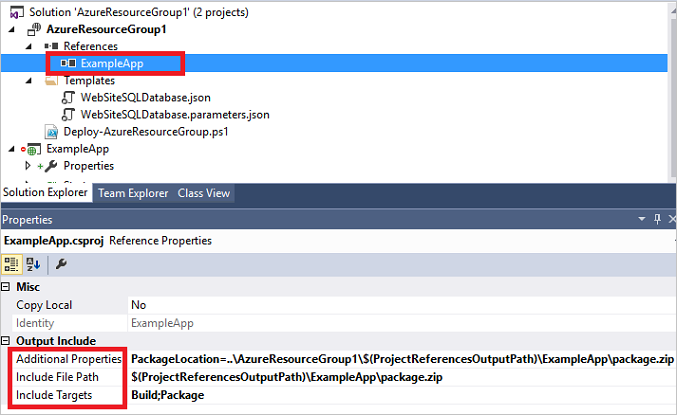
1. Now, you need to make sure your resource group project is aware of the new project. Go back to your resource group project (AzureResourceGroup1). Right-click **References** and select **Add Reference**.



1. Select the web app project that you created.



By adding a reference, you link the web app project to the resource group project, and automatically set three key properties. You see these properties in the **Properties** window for the reference.

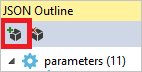


The properties are:

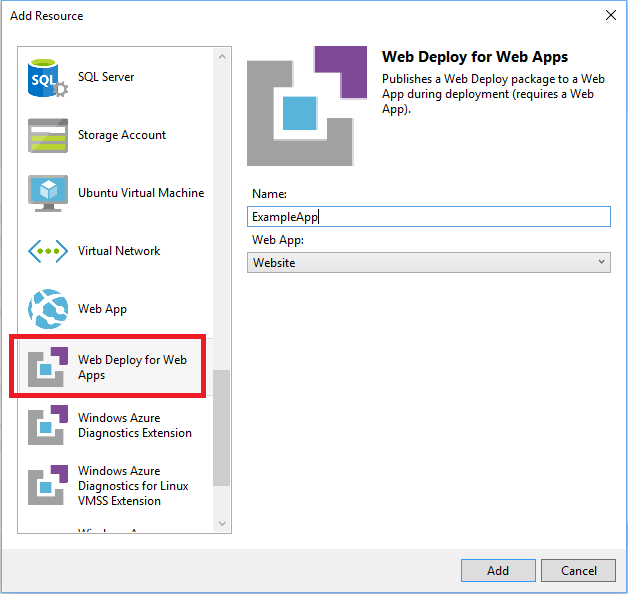
* + The **Additional Properties** contains the web deployment package staging location that is pushed to the Azure Storage. Note the folder (ExampleApp) and file (package.zip). You will provide these values as parameters when deploying the app.
  + The **Include File Path** contains the path where the package is created. The **Include Targets** contains the command that deployment executes.
  + The default value of **Build;Package** enables the deployment to build and create a web deployment package (package.zip).

You do not need a publish profile as the deployment gets the necessary information from the properties to create the package.

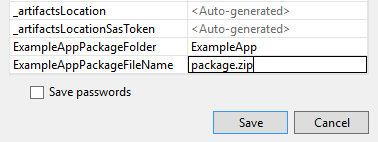
1. Add a resource to the template.



1. This time select **Web Deploy for Web Apps**.

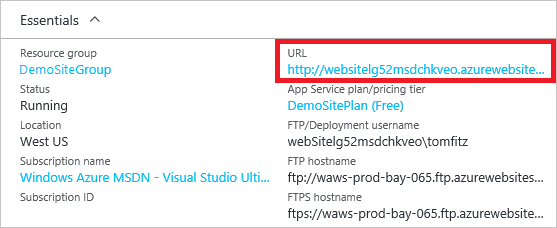


1. Redeploy your resource group project to the resource group. This time there are some new parameters. You do not need to provide values for **\_artifactsLocation** or **\_artifactsLocationSasToken** because Visual Studio automatically generates those values. However, you have to set the folder and file name to the path that contains the deployment package (shown as **ExampleAppPackageFolder** and **ExampleAppPackageFileName** in the following image). Provide the values you saw earlier in the reference properties (**ExampleApp** and **package.zip**).

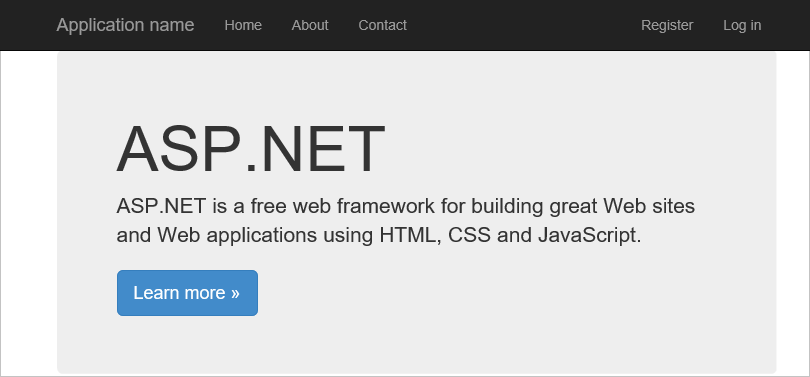


For the **Artifact storage account**, select the one deployed with this resource group.

1. After the deployment has finished, select your web app in the portal. Select the URL to browse to the site.



1. Notice that you have successfully deployed the default ASP.NET app.



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