

Deploy resources with Resource Manager templates and Azure PowerShell

📅 10/23/2018 ⌚ 7 minutes to read Contributors 👤👤👤👤👤 all

In this article

- Deploy a template from your local machine
- Deploy a template from an external source
- Deploy template from Cloud Shell
- Deploy to more than one resource group or subscription
- Parameters
- Test a template deployment
- Next steps

This article explains how to use Azure PowerShell with Resource Manager templates to deploy your resources to Azure. If you aren't familiar with the concepts of deploying and managing your Azure solutions, see [Azure Resource Manager overview](#).

The Resource Manager template you deploy can either be a local file on your machine or an external file that is located in a repository like GitHub. The template you deploy in this article is available as [storage account template in GitHub](#).

If needed, install the Azure PowerShell module using the instructions found in the [Azure PowerShell guide](#), and then run `Connect-AzureRmAccount` to create a connection with Azure.

Deploy a template from your local machine

When deploying resources to Azure, you:

1. Sign in to your Azure account
2. Create a resource group that serves as the container for the deployed resources. The name of the resource group can only include alphanumeric characters, periods, underscores, hyphens, and parenthesis. It can be up to 90 characters. It can't end in a period.
3. Deploy to the resource group the template that defines the resources to create

A template can include parameters that enable you to customize the deployment. For example, you can provide values that are tailored for a particular environment (such as dev, test, and production). The sample template defines a parameter for the storage account SKU.

The following example creates a resource group, and deploys a template from your local machine:

PowerShell	📄 Copy
<pre>Connect-AzureRmAccount Select-AzureRmSubscription -SubscriptionName <yourSubscriptionName> New-AzureRmResourceGroup -Name ExampleResourceGroup -Location "South Central US"</pre>	

```
New-AzureRmResourceGroupDeployment -Name ExampleDeployment -ResourceGroupName ExampleResourceGroup -TemplateFile c:\MyTemplates\storage.json -storageAccountType Standard_GRS
```

The deployment can take a few minutes to complete. When it finishes, you see a message that includes the result:

PowerShell	Copy
ProvisioningState	: Succeeded

Deploy a template from an external source

Instead of storing Resource Manager templates on your local machine, you may prefer to store them in an external location. You can store templates in a source control repository (such as GitHub). Or, you can store them in an Azure storage account for shared access in your organization.

To deploy an external template, use the **TemplateUri** parameter. Use the URI in the example to deploy the sample template from GitHub.

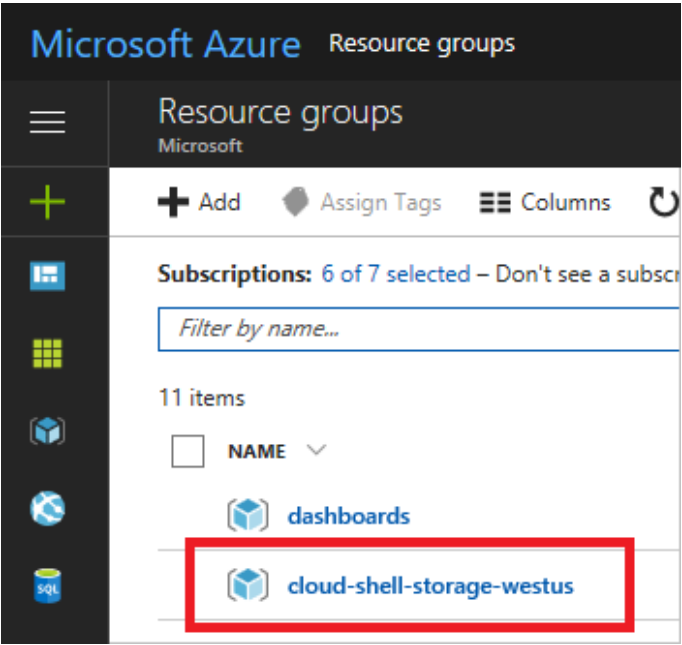
PowerShell	Copy
<pre>New-AzureRmResourceGroupDeployment -Name ExampleDeployment -ResourceGroupName ExampleResourceGroup -TemplateUri https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101-storage-account-create/azuredeploy.json -storageAccountType Standard_GRS</pre>	

The preceding example requires a publicly accessible URI for the template, which works for most scenarios because your template shouldn't include sensitive data. If you need to specify sensitive data (like an admin password), pass that value as a secure parameter. However, if you don't want your template to be publicly accessible, you can protect it by storing it in a private storage container. For information about deploying a template that requires a shared access signature (SAS) token, see [Deploy private template with SAS token](#).

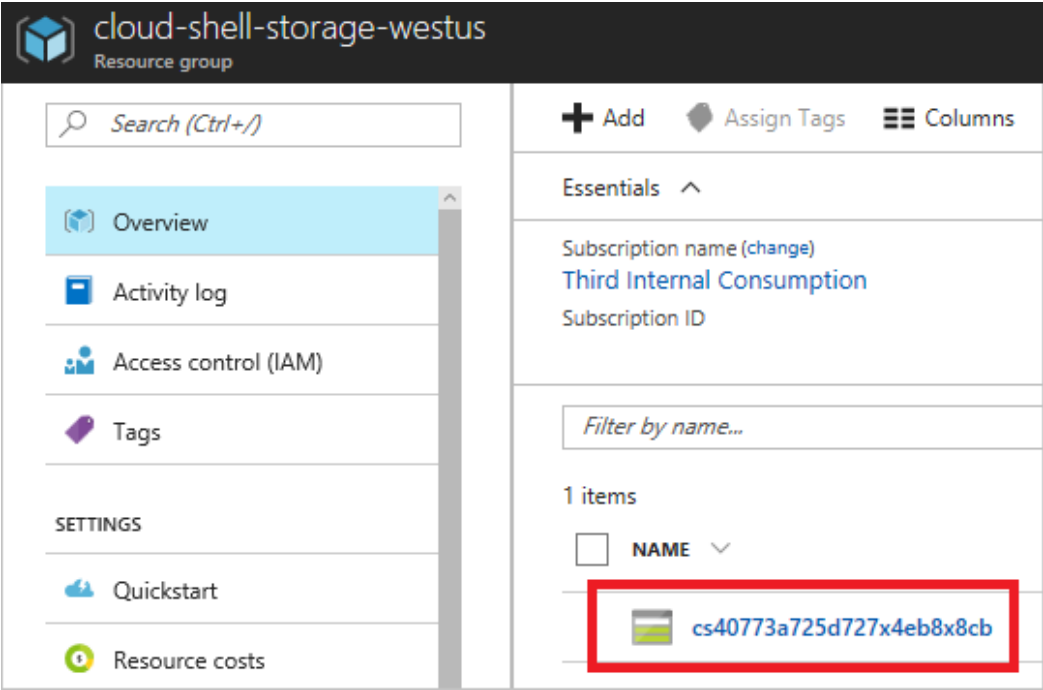
Deploy template from Cloud Shell

You can use [Cloud Shell](#) to deploy your template. However, you must first load your template into the storage account for your Cloud Shell. If you have not used Cloud Shell, see [Overview of Azure Cloud Shell](#) for information about setting it up.

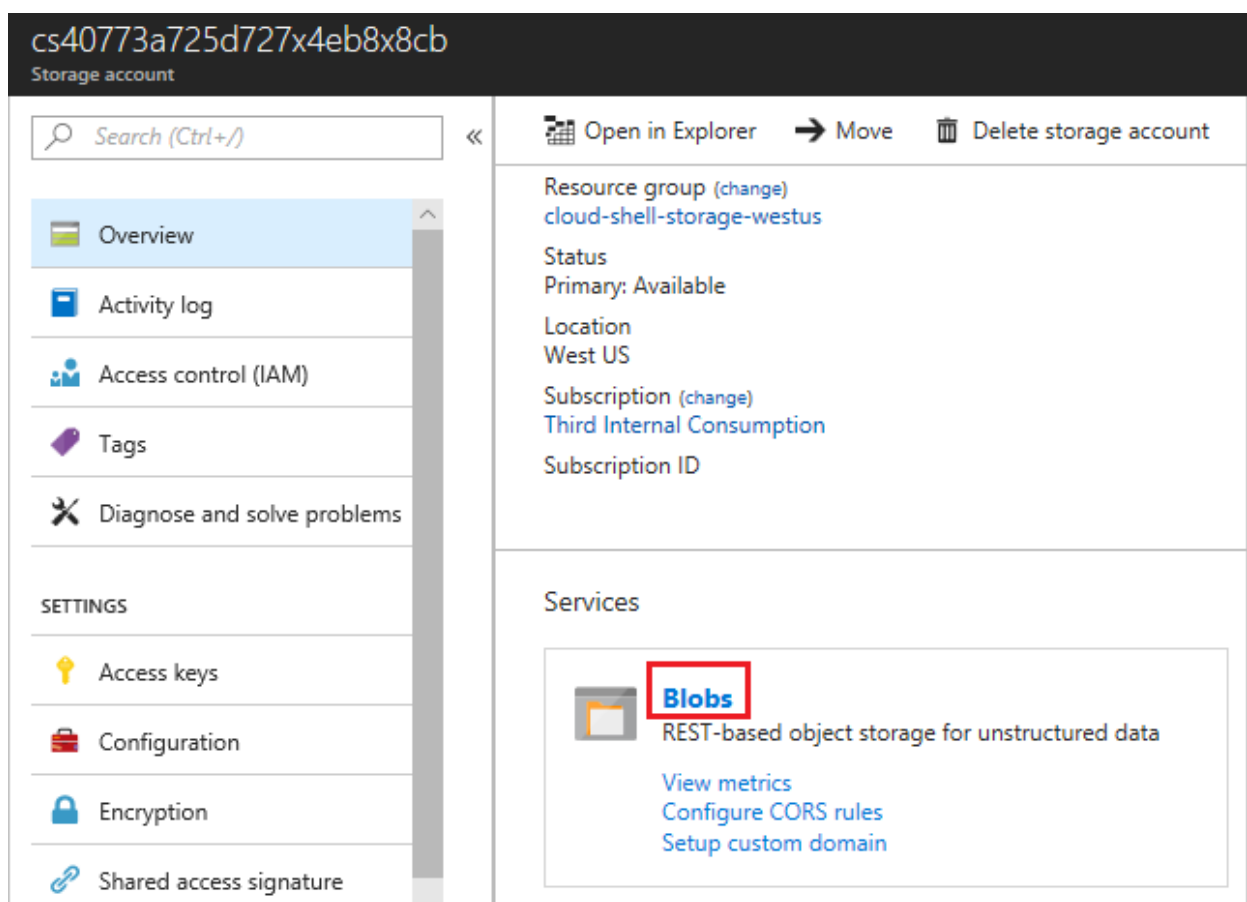
1. Sign in to the [Azure portal](#).
2. Select your Cloud Shell resource group. The name pattern is `cloud-shell-storage-<region>`.



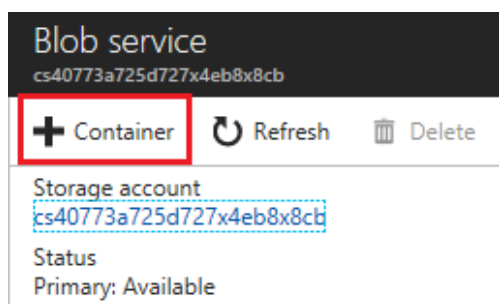
3. Select the storage account for your Cloud Shell.



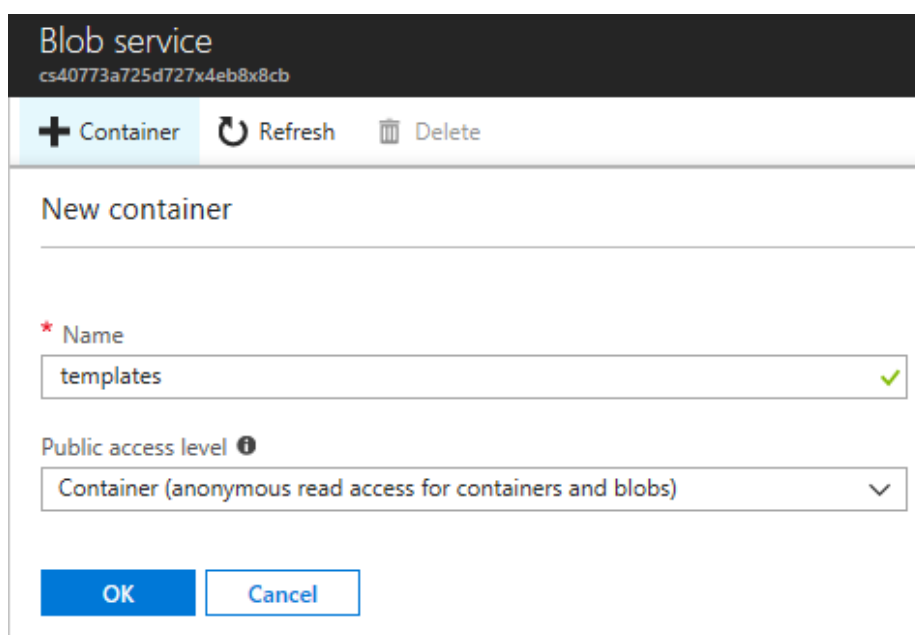
4. Select **Blobs**.



5. Select + **Container**.



6. Give your container a name and an access level. The sample template in this article contains no sensitive information, so allow anonymous read access. Select **OK**.



7. Select the container you created.

Blob service

cs40773a725d727x4eb8x8cb

+

Container

↺

Refresh

🗑

Delete

Storage account

cs40773a725d727x4eb8x8cb

Status

Primary: Available

Location

West US

Subscription (change)

Third Internal Consumption

Subscription ID

🔍

Search containers by prefix

NAME

templates

8. Select Upload.

templates

Container

⬆

Upload

↺

Refresh

Location: templates

🔍

Search blobs by prefix (case

9. Find and upload your template.

Upload blob

templates/

Files ⓘ

"azuredeploy.json"

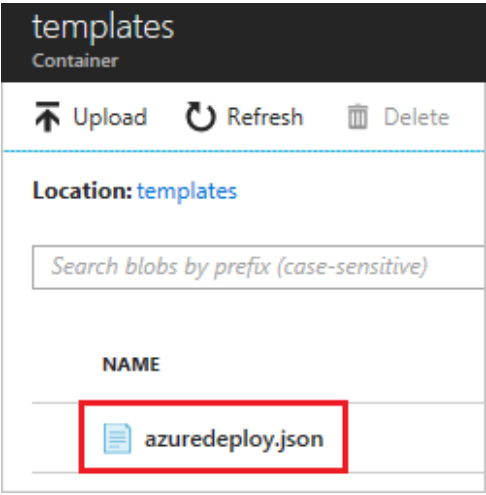
📁

☐ Overwrite if files already exist

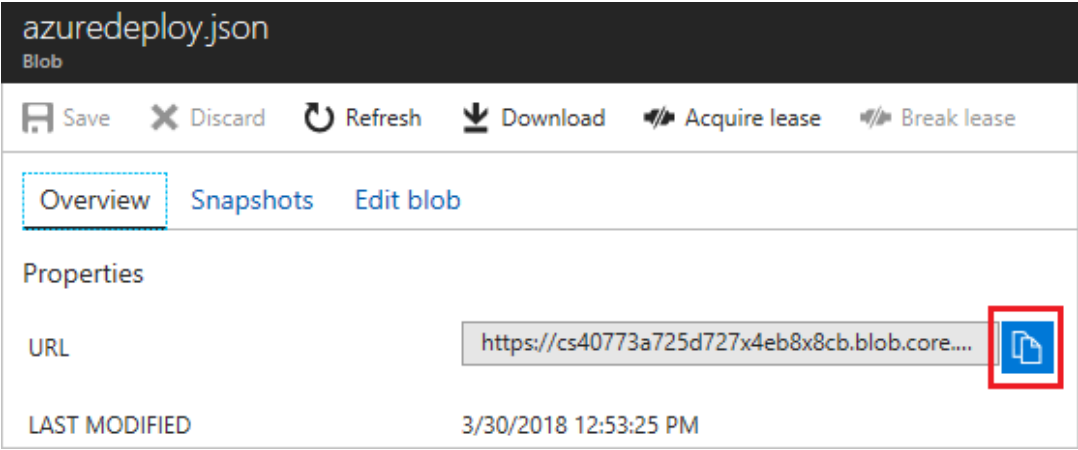
⌵ Advanced

Upload

10. After it has uploaded, select the template.



11. Copy the URL.



12. Open the prompt.



In the Cloud Shell, use the following commands:

PowerShell Copy

```
New-AzureRmResourceGroup -Name ExampleResourceGroup -Location "South Central US"
New-AzureRmResourceGroupDeployment -ResourceGroupName ExampleResourceGroup `
  -TemplateUri <copied URL> `
  -storageAccountType Standard_GRS
```

Deploy to more than one resource group or subscription

Typically, you deploy all the resources in your template to a single resource group. However, there are scenarios where you want to deploy a set of resources together but place them in different resource groups or subscriptions. You can deploy to only five resource groups in a single deployment. For more information, see [Deploy Azure resources to more than one subscription or resource group](#).

Parameters

To pass parameter values, you can use either inline parameters or a parameter file. The preceding examples in this article show inline parameters.

Inline parameters

To pass inline parameters, provide the names of the parameter with the `New-AzureRmResourceGroupDeployment` command. For example, to pass a string and array to a template, use:

PowerShell	
<pre>\$arrayParam = "value1", "value2" New-AzureRmResourceGroupDeployment -ResourceGroupName testgroup ` -TemplateFile c:\MyTemplates\demotemplate.json ` -exampleString "inline string" ` -exampleArray \$arrayParam</pre>	

You can also get the contents of file and provide that content as an inline parameter.


PowerShell	
<pre>\$arrayParam = "value1", "value2" New-AzureRmResourceGroupDeployment -ResourceGroupName testgroup ` -TemplateFile c:\MyTemplates\demotemplate.json ` -exampleString \$(Get-Content -Path c:\MyTemplates\stringcontent.txt -Raw) ` -exampleArray \$arrayParam</pre>	

Getting a parameter value from a file is helpful when you need to provide configuration values. For example, you can provide [cloud-init values for a Linux virtual machine](#).

Parameter files

Rather than passing parameters as inline values in your script, you may find it easier to use a JSON file that contains the parameter values. The parameter file can be a local file or an external file with an accessible URI.

The parameter file must be in the following format:


JSON	
<pre>{ "\$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#", "contentVersion": "1.0.0.0", "parameters": { "storageAccountType": { "value": "Standard_GRS" } } }</pre>	

Notice that the parameters section includes a parameter name that matches the parameter defined in your template (storageAccountType). The parameter file contains a value for the parameter. This


value is automatically passed to the template during deployment. You can create more than one parameter file, and then pass in the appropriate parameter file for the scenario.

Copy the preceding example and save it as a file named `storage.parameters.json`.

To pass a local parameter file, use the **TemplateParameterFile** parameter:

PowerShell	
<pre>New-AzureRmResourceGroupDeployment -Name ExampleDeployment -ResourceGroupName ExampleResourceGroup ` -TemplateFile c:\MyTemplates\storage.json ` -TemplateParameterFile c:\MyTemplates\storage.parameters.json</pre>	

To pass an external parameter file, use the **TemplateParameterUri** parameter:

PowerShell	
<pre>New-AzureRmResourceGroupDeployment -Name ExampleDeployment -ResourceGroupName ExampleResourceGroup ` -TemplateUri https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101-storage-account-create/azuredeploy.json ` -TemplateParameterUri https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101-storage-account-create/azuredeploy.parameters.json</pre>	

Parameter precedence

You can use inline parameters and a local parameter file in the same deployment operation. For example, you can specify some values in the local parameter file and add other values inline during deployment. If you provide values for a parameter in both the local parameter file and inline, the inline value takes precedence.


However, when you use an external parameter file, you can't pass other values either inline or from a local file. When you specify a parameter file in the **TemplateParameterUri** parameter, all inline parameters are ignored. Provide all parameter values in the external file. If your template includes a sensitive value that you can't include in the parameter file, either add that value to a key vault, or dynamically provide all parameter values inline.

Parameter name conflicts


If your template includes a parameter with the same name as one of the parameters in the PowerShell command, PowerShell presents the parameter from your template with the postfix **FromTemplate**. For example, a parameter named **ResourceGroupName** in your template conflicts with the **ResourceGroupName** parameter in the [New-AzureRmResourceGroupDeployment](#) cmdlet. You're prompted to provide a value for **ResourceGroupNameFromTemplate**. In general, you should avoid this confusion by not naming parameters with the same name as parameters used for deployment operations.

Test a template deployment


To test your template and parameter values without actually deploying any resources, use [Test-Azure RmResourceGroupDeployment](#).

PowerShell	
<pre>Test-AzureRmResourceGroupDeployment -ResourceGroupName ExampleResourceGroup ` -TemplateFile c:\MyTemplates\storage.json -storageAccountType Standard_GRS</pre>	

If no errors are detected, the command finishes without a response. If an error is detected, the command returns an error message. For example, passing an incorrect value for the storage account SKU, returns the following error:

PowerShell	
<pre>Test-AzureRmResourceGroupDeployment -ResourceGroupName testgroup ` -TemplateFile c:\MyTemplates\storage.json -storageAccountType badSku</pre> <p>Code : InvalidTemplate Message : Deployment template validation failed: 'The provided value 'badSku' for the template parameter 'storageAccountType' at line '15' and column '24' is not valid. The parameter value is not part of the allowed value(s): 'Standard_LRS,Standard_ZRS,Standard_GRS,Standard_RAGRS,Premium_LRS'.'. Details :</p>	

If your template has a syntax error, the command returns an error indicating it couldn't parse the template. The message indicates the line number and position of the parsing error.

PowerShell	
<pre>Test-AzureRmResourceGroupDeployment : After parsing a value an unexpected character was encountered: ". Path 'variables', line 31, position 3.</pre>	

Next steps

- The examples in this article deploy resources to a resource group in your default subscription. To use a different subscription, see [Manage multiple Azure subscriptions](#).
- To specify how to handle resources that exist in the resource group but aren't defined in the template, see [Azure Resource Manager deployment modes](#).
- To understand how to define parameters in your template, see [Understand the structure and syntax of Azure Resource Manager templates](#).
- For tips on resolving common deployment errors, see [Troubleshoot common Azure deployment errors with Azure Resource Manager](#).
- For information about deploying a template that requires a SAS token, see [Deploy private template with SAS token](#).
- To safely roll out your service to more than one region, see [Azure Deployment Manager](#).