Tell us about your PDF experience.

ARM template documentation

Azure Resource Manager templates are JavaScript Object Notation (JSON) files that define the infrastructure and configuration for your project.

About ARM templates

OVERVIEW

What are templates?

CONCEPT

Best practices

Frequently asked questions

Template specs

Deployment modes

Linked templates

VIDEO

Build 2020 presentation

Get started

QUICKSTART

Create JSON templates - VS Code

Create JSON templates - portal

Create & deploy template specs

TUTORIAL

Beginner template tutorials

TRAINING

Deploy with ARM templates

Develop templates

TUTORIAL

Advanced templates

Template specs

HOW-TO GUIDE

Use functions

Define parameters

Define variables

Define resources

Define outputs

REFERENCE

Template file syntax

Azure Quickstart templates

Deploy templates

TUTORIAL

Deployment

HOW-TO GUIDE

PowerShell

Azure CLI

Portal

REST API

Deploy to Azure button

Cloud Shell

What-if deployment

Scoped deployments

HOW-TO GUIDE

Resource group

Subscription

Management group

Tenant

Scoping extension resource

Template functions across scopes

Explore reference content

REFERENCE

Template reference

REST API

Azure PowerShell

Azure CLI

.NET

Java

Python

Manage templates

HOW-TO GUIDE

Test toolkit

Export templates

CI/CD

View deployment history

What are ARM templates?

Article • 04/09/2023

With the move to the cloud, many teams have adopted agile development methods. These teams iterate quickly. They need to repeatedly deploy their solutions to the cloud, and know their infrastructure is in a reliable state. As infrastructure has become part of the iterative process, the division between operations and development has disappeared. Teams need to manage infrastructure and application code through a unified process.

To meet these challenges, you can automate deployments and use the practice of infrastructure as code. In code, you define the infrastructure that needs to be deployed. The infrastructure code becomes part of your project. Just like application code, you store the infrastructure code in a source repository and version it. Anyone on your team can run the code and deploy similar environments.

To implement infrastructure as code for your Azure solutions, use Azure Resource Manager templates (ARM templates). The template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax, which lets you state what you intend to deploy without having to write the sequence of programming commands to create it. In the template, you specify the resources to deploy and the properties for those resources.

> **Tip**
>
> We've introduced a new language named **Bicep** that offers the same capabilities as ARM templates but with a syntax that's easier to use. Each Bicep file is automatically converted to an ARM template during deployment. If you're considering infrastructure as code options, we recommend looking at Bicep. For more information, see **What is Bicep?**.

To learn about how you can get started with ARM templates, see the following video. https://learn.microsoft.com/Shows/Azure-Enablement/How-and-why-to-learn-about-ARM-templates/player

Why choose ARM templates?

If you're trying to decide between using ARM templates and one of the other infrastructure as code services, consider the following advantages of using templates:

- **Declarative syntax**: ARM templates allow you to create and deploy an entire Azure infrastructure declaratively. For example, you can deploy not only virtual machines, but also the network infrastructure, storage systems, and any other resources you may need.

- **Repeatable results**: Repeatedly deploy your infrastructure throughout the development lifecycle and have confidence your resources are deployed in a consistent manner. Templates are idempotent, which means you can deploy the same template many times and get the same resource types in the same state. You can develop one template that represents the desired state, rather than developing lots of separate templates to represent updates.

- **Orchestration**: You don't have to worry about the complexities of ordering operations. Resource Manager orchestrates the deployment of interdependent resources so they're created in the correct order. When possible, Resource Manager deploys resources in parallel so your deployments finish faster than serial deployments. You deploy the template through one command, rather than through multiple imperative commands.

- **Modular files**: You can break your templates into smaller, reusable components and link them together at deployment time. You can also nest one template inside another template.

- **Create any Azure resource**: You can immediately use new Azure services and features in templates. As soon as a resource provider introduces new resources, you can deploy those resources through templates. You don't have to wait for tools or modules to be updated before using the new services.

- **Extensibility**: With deployment scripts, you can add PowerShell or Bash scripts to your templates. The deployment scripts extend your ability to set up resources during deployment. A script can be included in the template, or stored in an external source and referenced in the template. Deployment scripts give you the ability to complete your end-to-end environment setup in a single ARM template.

- **Testing**: You can make sure your template follows recommended guidelines by testing it with the ARM template tool kit (arm-ttk). This test kit is a PowerShell script that you can download from GitHub. The tool kit makes it easier for you to develop expertise using the template language.

- **Preview changes**: You can use the what-if operation to get a preview of changes before deploying the template. With what-if, you see which resources will be created, updated, or deleted, and any resource properties that will be changed. The what-if operation checks the current state of your environment and eliminates the need to manage state.

- **Built-in validation**: Your template is deployed only after passing validation. Resource Manager checks the template before starting the deployment to make sure the deployment will succeed. Your deployment is less likely to stop in a half-finished state.

- **Tracked deployments**: In the Azure portal, you can review the deployment history and get information about the template deployment. You can see the template that was deployed, the parameter values passed in, and any output values. Other infrastructure as code services aren't tracked through the portal.

- **Policy as code**: Azure Policy is a policy as code framework to automate governance. If you're using Azure policies, policy remediation is done on non-compliant resources when deployed through templates.

- **Deployment Blueprints**: You can take advantage of Blueprints provided by Microsoft to meet regulatory and compliance standards. These blueprints include pre-built templates for various architectures.

- **CI/CD integration**: You can integrate templates into your continuous integration and continuous deployment (CI/CD) tools, which can automate your release pipelines for fast and reliable application and infrastructure updates. By using Azure DevOps and Resource Manager template task, you can use Azure Pipelines

to continuously build and deploy ARM template projects. To learn more, see VS project with pipelines and Tutorial: Continuous integration of Azure Resource Manager templates with Azure Pipelines.

- **Exportable code**: You can get a template for an existing resource group by either exporting the current state of the resource group, or viewing the template used for a particular deployment. Viewing the exported template is a helpful way to learn about the template syntax.
- **Authoring tools**: You can author templates with Visual Studio Code and the template tool extension. You get intellisense, syntax highlighting, in-line help, and many other language functions. In addition to Visual Studio Code, you can also use Visual Studio.

Template file

Within your template, you can write template expressions that extend the capabilities of JSON. These expressions make use of the functions provided by Resource Manager.

The template has the following sections:

- Parameters - Provide values during deployment that allow the same template to be used with different environments.
- Variables - Define values that are reused in your templates. They can be constructed from parameter values.
- User-defined functions - Create customized functions that simplify your template.
- Resources - Specify the resources to deploy.
- Outputs - Return values from the deployed resources.

Template deployment process

When you deploy a template, Resource Manager converts the template into REST API operations. For example, when Resource Manager receives a template with the following resource definition:

JSON

```
"resources": [
  {
    "type": "Microsoft.Storage/storageAccounts",
```

```
    "apiVersion": "2019-04-01",
    "name": "mystorageaccount",
    "location": "westus",
    "sku": {
      "name": "Standard_LRS"
    },
    "kind": "StorageV2",
    "properties": {}
  }
]
```

It converts the definition to the following REST API operation, which is sent to the Microsoft.Storage resource provider:

```HTTP
PUT
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Storage/storageAccounts/mystorageaccount?api-version=2019-04-01
REQUEST BODY
{
  "location": "westus",
  "sku": {
    "name": "Standard_LRS"
  },
  "kind": "StorageV2",
  "properties": {}
}
```

Notice that the **apiVersion** you set in the template for the resource is used as the API version for the REST operation. You can repeatedly deploy the template and have confidence it will continue to work. By using the same API version, you don't have to worry about breaking changes that might be introduced in later versions.

To deploy a template, use any of the following options:

- Azure portal
- Azure CLI
- PowerShell
- REST API
- Button in GitHub repository
- Azure Cloud Shell

Template design

How you define templates and resource groups is entirely up to you and how you want to manage your solution. For example, you can deploy your three tier application through a single template to a single resource group.

But, you don't have to define your entire infrastructure in a single template. Often, it makes sense to divide your deployment requirements into a set of targeted, purpose-specific templates. You can easily reuse these templates for different solutions. To deploy a particular solution, you create a main template that links all the required templates. The following image shows how to deploy a three tier solution through a parent template that includes three nested templates.

If you envision your tiers having separate lifecycles, you can deploy your three tiers to separate resource groups. Notice the resources can still be linked to resources in other resource groups.

For information about nested templates, see Using linked templates with Azure Resource Manager.

Share templates

After creating your template, you may wish to share it with other users in your organization. Template specs enable you to store a template as a resource type. You use role-based access control to manage access to the template spec. Users with read access to the template spec can deploy it, but not change the template.

This approach means you can safely share templates that meet your organization's standards.

Next steps

- For a step-by-step tutorial that guides you through the process of creating a template, see Tutorial: Create and deploy your first ARM template.
- To learn about ARM templates through a guided set of Learn modules, see Deploy and manage resources in Azure by using ARM templates.
- For information about the properties in template files, see Understand the structure and syntax of ARM templates.
- To learn about exporting templates, see Quickstart: Create and deploy ARM templates by using the Azure portal.
- For answers to common questions, see Frequently asked questions about ARM templates.

Quickstart: Create ARM templates with Visual Studio Code

Article • 03/09/2023

The Azure Resource Manager Tools for Visual Studio Code provide language support, resource snippets, and resource autocompletion. These tools help create and validate Azure Resource Manager templates (ARM templates), and are therefore the recommended method of ARM template creation and configuration. In this quickstart, you use the extension to create an ARM template from scratch. While doing so you experience the extensions capabilities such as ARM template snippets, validation, completions, and parameter file support.

To complete this quickstart, you need Visual Studio Code, with the Azure Resource Manager tools extension installed. You also need either the Azure CLI or the Azure PowerShell module installed and authenticated.

If you don't have an Azure subscription, create a free account before you begin.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **Quickstart: Create Bicep files with Visual Studio Code**.

Create an ARM template

Create and open with Visual Studio Code a new file named *azuredeploy.json*. Enter `arm` into the code editor, which initiates Azure Resource Manager snippets for scaffolding out an ARM template.

Select `arm!` to create a template scoped for an Azure resource group deployment.

This snippet creates the basic building blocks for an ARM template.

Notice that the Visual Studio Code language mode has changed from *JSON* to *Azure Resource Manager Template*. The extension includes a language server specific to ARM templates that provides ARM template-specific validation, completion, and other language services.

Add an Azure resource

The extension includes snippets for many Azure resources. These snippets can be used to easily add resources to your template deployment.

Place the cursor in the template **resources** block, type in `storage`, and select the *arm-storage* snippet.

This action adds a storage resource to the template.

The **tab** key can be used to tab through configurable properties on the storage account.

Completion and validation

One of the most powerful capabilities of the extension is its integration with Azure schemas. Azure schemas provide the extension with validation and resource-aware

completion capabilities. Let's modify the storage account to see validation and completion in action.

First, update the storage account kind to an invalid value such as `megaStorage`. Notice that this action produces a warning indicating that `megaStorage` isn't a valid value.

To use the completion capabilities, remove `megaStorage`, place the cursor inside of the double quotes, and press `ctrl` + `space`. This action presents a completion list of valid values.

Add template parameters

Now create and use a parameter to specify the storage account name.

Place your cursor in the parameters block, add a carriage return, type `"`, and then select the `new-parameter` snippet. This action adds a generic parameter to the template.

Update the name of the parameter to `storageAccountName` and the description to `Storage Account Name`.

Azure storage account names have a minimum length of 3 characters and a maximum of 24. Add both `minLength` and `maxLength` to the parameter and provide appropriate values.

Now, on the storage resource, update the name property to use the parameter. To do so, remove the current name. Enter a double quote and an opening square bracket `[`, which produces a list of ARM template functions. Select *parameters* from the list.

Entering a single quote `'` inside of the round brackets produces a list of all parameters defined in the template, in this case, *storageAccountName*. Select the parameter.

Create a parameter file

An ARM template parameter file allows you to store environment-specific parameter values and pass these values in as a group at deployment time. For example, you may have a parameter file with values specific to a test environment and another for a production environment.

The extension makes it easy to create a parameter file from your existing templates. To do so, right-click on the template in the code editor and select `Select/Create Parameter File`.

Select `New` > `All Parameters` > Select a name and location for the parameter file.

This action creates a new parameter file and maps it with the template from which it was created. You can see and modify the current template/parameter file mapping in the Visual Studio Code status bar while the template is selected.

Now that the parameter file has been mapped to the template, the extension validates both the template and parameter file together. To see this validation in practice, add a two-character value to the `storageAccountName` parameter in the parameter file and save the file.

Navigate back to the ARM template and notice that an error has been raised indicating that the value doesn't meet the parameter criteria.

Update the value to something appropriate, save the file, and navigate back to the template. Notice that the error on the parameter has been resolved.

Deploy the template

Open the integrated Visual Studio Code terminal using the `ctrl` + `` ` `` key combination and use either the Azure CLI or Azure PowerShell module to deploy the template.

CLI

Azure CLI

```
az group create --name arm-vscode --location eastus

az deployment group create --resource-group arm-vscode --template-file
azuredeploy.json --parameters azuredeploy.parameters.json
```

Clean up resources

When the Azure resources are no longer needed, use the Azure CLI or Azure PowerShell module to delete the quickstart resource group.

CLI

Azure CLI

```
az group delete --name arm-vscode
```

Next steps

Beginner tutorials

Quickstart: Create and deploy ARM templates by using the Azure portal

Article • 04/09/2023

In this quickstart, you learn how to create an Azure Resource Manager template (ARM template) in the Azure portal. You edit and deploy the template from the portal.

ARM templates are JSON or Bicep files that define the resources you need to deploy for your solution. To understand the concepts associated with deploying and managing your Azure solutions, see template deployment overview.

After completing the tutorial, you deploy an Azure Storage account. The same process can be used to deploy other Azure resources.

If you don't have an Azure subscription, create a free account before you begin.

Retrieve a custom template

Rather than manually building an entire ARM template, let's start by retrieving a pre-built template that accomplishes our goal. The Azure Quickstart Templates repo repo contains a large collection of templates that deploy common scenarios. The portal makes it easy for you find and use templates from this repo. You can save the template and reuse it later.

1. In a web browser, go to the Azure portal and sign in.

2. From the Azure portal search bar, search for **deploy a custom template** and then select it from the available options.

3. For **Template** source, notice that **Quickstart template** is selected by default. You can keep this selection. In the drop-down, search for

quickstarts/microsoft.storage/storage-account-create and select it. After finding the quickstart template, select **Select template**.

4. In the next blade, you provide custom values to use for the deployment.

   For **Resource group**, select **Create new** and provide *myResourceGroup* for the name. You can use the default values for the other fields. When you've finished providing values, select **Review + create**.

5. The portal validates your template and the values you provided. After validation succeeds, select **Create** to start the deployment.

6. Once your validation has passed, you'll see the status of the deployment. When it completes successfully, select **Go to resource** to see the storage account.

7. From this screen, you can view the new storage account and its properties.

Edit and deploy the template

You can use the portal for quickly developing and deploying ARM templates. In general, we recommend using Visual Studio Code for developing your ARM templates, and Azure CLI or Azure PowerShell for deploying the template, but you can use the portal for quick deployments without installing those tools.

In this section, let's suppose you have an ARM template that you want to deploy one time without setting up the other tools.

1. Again, select **Deploy a custom template** in the portal.

2. This time, select **Build your own template in the editor**.

3. You see a blank template.

4. Replace the blank template with the following template. It deploys a virtual network with a subnet.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "vnetName": {
      "type": "string",
```

```
        "defaultValue": "VNet1",
        "metadata": {
          "description": "VNet name"
        }
      },
      "vnetAddressPrefix": {
        "type": "string",
        "defaultValue": "10.0.0.0/16",
        "metadata": {
          "description": "Address prefix"
        }
      },
      "subnetPrefix": {
        "type": "string",
        "defaultValue": "10.0.0.0/24",
        "metadata": {
          "description": "Subnet Prefix"
        }
      },
      "subnetName": {
        "type": "string",
        "defaultValue": "Subnet1",
        "metadata": {
          "description": "Subnet Name"
        }
      },
      "location": {
        "type": "string",
        "defaultValue": "[resourceGroup().location]",
        "metadata": {
          "description": "Location for all resources."
        }
      }
    },
    "resources": [
      {
        "type": "Microsoft.Network/virtualNetworks",
        "apiVersion": "2021-08-01",
        "name": "[parameters('vnetName')]",
        "location": "[parameters('location')]",
        "properties": {
          "addressSpace": {
            "addressPrefixes": [
              "[parameters('vnetAddressPrefix')]"
            ]
          },
          "subnets": [
            {
              "name": "[parameters('subnetName')]",
              "properties": {
                "addressPrefix": "[parameters('subnetPrefix')]"
              }
            }
          ]
        }
```

```
      }
    ]
  }
```

5. Select **Save**.

6. You see the blade for providing deployment values. Again, select **myResourceGroup** for the resource group. You can use the other default values. When you're done providing values, select **Review + create**

7. After the portal validates the template, select **Create**.

8. When the deployment completes, you see the status of the deployment. This time select the name of the resource group.

9. Notice that your resource group now contains a storage account and a virtual network.

Export a custom template

Sometimes the easiest way to work with an ARM template is to have the portal generate it for you. The portal can create an ARM template based on the current state of your resource group.

1. In your resource group, select **Export template**.

2. The portal generates a template for you based on the current state of the resource group. Notice that this template isn't the same as either template you deployed earlier. It contains definitions for both the storage account and virtual network, along with other resources like a blob service that was automatically created for your storage account.

3. To save this template for later use, select **Download**.

You now have an ARM template that represents the current state of the resource group. This template is auto-generated. Before using the template for production deployments, you may want to revise it, such as adding parameters for template reuse.

Clean up resources

When the Azure resources are no longer needed, clean up the resources you deployed by deleting the resource group.

1. In the Azure portal, select **Resource groups** on the left menu.
2. Enter the resource group name in the **Filter for any field** search box.
3. Select the resource group name. You shall see the storage account in the resource group.
4. Select **Delete resource group** in the top menu.

Next steps

In this tutorial, you learned how to generate a template from the Azure portal, and how to deploy the template using the portal. The template used in this Quickstart is a simple template with one Azure resource. When the template is complex, it's easier to use

Visual Studio Code, or Visual Studio to develop the template. To learn more about template development, see our new beginner tutorial series:

Beginner tutorials

Quickstart: Create and deploy template spec

Article • 04/09/2023

This quickstart shows you how to package an Azure Resource Manager template (ARM template) into a template spec. Then, you deploy that template spec. Your template spec contains an ARM template that deploys a storage account.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **Quickstart: Create and deploy a template spec with Bicep**.

Prerequisites

An Azure account with an active subscription. Create an account for free.

> **Note**
>
> To use template spec with Azure PowerShell, you must install **version 5.0.0 or later**. To use it with Azure CLI, use **version 2.14.2 or later**.

Create template

You create a template spec from a local template. Copy the following template and save it locally to a file named **azuredeploy.json**. This quickstart assumes you've saved to a path **c:\Templates\azuredeploy.json** but you can use any path.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.13.1.58284",
      "templateHash": "13120038605368246703"
```

```
    }
  },
  "parameters": {
    "storageAccountType": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Premium_LRS",
        "Premium_ZRS",
        "Standard_GRS",
        "Standard_GZRS",
        "Standard_LRS",
        "Standard_RAGRS",
        "Standard_RAGZRS",
        "Standard_ZRS"
      ],
      "metadata": {
        "description": "Storage Account type"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "The storage account location."
      }
    },
    "storageAccountName": {
      "type": "string",
      "defaultValue": "[format('store{0}',
uniqueString(resourceGroup().id))]",
      "metadata": {
        "description": "The name of the storage account"
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2022-09-01",
      "name": "[parameters('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('storageAccountType')]"
      },
      "kind": "StorageV2",
      "properties": {}
    }
  ],
  "outputs": {
    "storageAccountName": {
      "type": "string",
      "value": "[parameters('storageAccountName')]"
    },
    "storageAccountId": {
```

```
        "type": "string",
        "value": "[resourceId('Microsoft.Storage/storageAccounts',
parameters('storageAccountName'))]"
      }
    }
  }
```

Create template spec

The template spec is a resource type named `Microsoft.Resources/templateSpecs`. To create a template spec, use PowerShell, Azure CLI, the portal, or an ARM template.

PowerShell

1. Create a new resource group to contain the template spec.

   Azure PowerShell

   ```
   New-AzResourceGroup `
     -Name templateSpecRG `
     -Location westus2
   ```

2. Create the template spec in that resource group. Give the new template spec the name **storageSpec**.

   Azure PowerShell

   ```
   New-AzTemplateSpec `
     -Name storageSpec `
     -Version "1.0" `
     -ResourceGroupName templateSpecRG `
     -Location westus2 `
     -TemplateFile "c:\Templates\azuredeploy.json"
   ```

Deploy template spec

To deploy a template spec, use the same deployment commands as you would use to deploy a template. Pass in the resource ID of the template spec to deploy.

PowerShell

1. Create a resource group to contain the new storage account.

   Azure PowerShell

   ```
   New-AzResourceGroup `
     -Name storageRG `
     -Location westus2
   ```

2. Get the resource ID of the template spec.

   Azure PowerShell

   ```
   $id = (Get-AzTemplateSpec -ResourceGroupName templateSpecRG -Name
   storageSpec -Version "1.0").Versions.Id
   ```

3. Deploy the template spec.

   Azure PowerShell

   ```
   New-AzResourceGroupDeployment `
     -TemplateSpecId $id `
     -ResourceGroupName storageRG
   ```

4. You provide parameters exactly as you would for an ARM template. Redeploy the template spec with a parameter for the storage account type.

   Azure PowerShell

   ```
   New-AzResourceGroupDeployment `
     -TemplateSpecId $id `
     -ResourceGroupName storageRG `
     -storageAccountType Standard_GRS
   ```

Grant access

If you want to let other users in your organization deploy your template spec, you need to grant them read access. You can assign the Reader role to an Azure AD group for the resource group that contains template specs you want to share. For more information, see Tutorial: Grant a group access to Azure resources using Azure PowerShell.

Update template

Let's suppose you've identified a change you want to make to the template in your template spec. The following template is similar to your earlier template except it adds a prefix for the storage account name. Copy the following template and update your azuredeploy.json file.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageAccountType": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_ZRS",
        "Premium_LRS"
      ],
      "metadata": {
        "description": "Storage Account type"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "namePrefix": {
      "type": "string",
      "maxLength": 11,
      "defaultValue": "store",
      "metadata": {
        "description": "Prefix for storage account name"
      }
    }
  },
  "variables": {
    "storageAccountName": "[concat(parameters('namePrefix'), uniquestring(resourceGroup().id))]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-04-01",
      "name": "[variables('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('storageAccountType')]"
```

```
      },
      "kind": "StorageV2",
      "properties": {}
    }
  ],
  "outputs": {
    "storageAccountName": {
      "type": "string",
      "value": "[variables('storageAccountName')]"
    }
  }
}
```

Update template spec version

Rather than creating a new template spec for the revised template, add a new version named `2.0` to the existing template spec. Users can choose either version to deploy.

PowerShell

1. Create a new version for the template spec.

   PowerShell

   ```
   New-AzTemplateSpec `
     -Name storageSpec `
     -Version "2.0" `
     -ResourceGroupName templateSpecRG `
     -Location westus2 `
     -TemplateFile "c:\Templates\azuredeploy.json"
   ```

2. To deploy the new version, get the resource ID for the `2.0` version.

   PowerShell

   ```
   $id = (Get-AzTemplateSpec -ResourceGroupName templateSpecRG -Name
   storageSpec -Version "2.0").Versions.Id
   ```

3. Deploy that version. Provide a prefix for the storage account name.

   PowerShell

   ```
   New-AzResourceGroupDeployment `
     -TemplateSpecId $id `
     -ResourceGroupName storageRG `
     -namePrefix "demoaccount"
   ```

Clean up resources

To clean up the resource you deployed in this quickstart, delete both resource groups that you created.

1. From the Azure portal, select Resource group from the left menu.
2. Enter the resource group name (templateSpecRG and storageRG) in the Filter by name field.
3. Select the resource group name.
4. Select Delete resource group from the top menu.

Next steps

To learn about creating a template spec that includes linked templates, see Create a template spec of a linked template.

Tutorial: Create and deploy your first ARM template

Article • 03/09/2023

This tutorial introduces you to Azure Resource Manager templates (ARM templates). It shows you how to create a starter template and deploy it to Azure. It teaches you about the template structure and the tools you need to work with templates. This instruction takes **12 minutes** to complete, but the actual finish time varies based on how many tools you need to install.

This tutorial is the first of a series. As you progress through the series, you modify the starting template, step by step, until you explore all of the core parts of an ARM template. These elements are the building blocks for more complex templates. We hope by the end of the series you're confident in creating your own templates and ready to automate your deployments with templates.

If you want to learn about the benefits of using templates and why you should automate deployments with templates, see ARM template overview. To learn about ARM templates through a guided set of Learn modules, see Deploy and manage resources in Azure by using JSON ARM templates.

If you don't have a Microsoft Azure subscription, create a free account before you begin.

> **Tip**
>
> If you're new to ARM templates, you might consider learning Bicep instead. Bicep is a new language that offers the same capabilities as ARM templates but with an easier-to-use syntax. To start learning Bicep, see **Quickstart: Create Bicep files with Visual Studio Code**.

Get tools

Let's start by making sure you have the tools you need to create and deploy templates. Install these tools on your local machine.

Editor

Templates are JavaScript Object Notation (JSON) files. To create templates, you need a good JSON editor. We recommend Visual Studio Code with the Azure Resource Manager Tools extension. If you need to install these tools, see Quickstart: Create ARM templates with Visual Studio Code.

Command-line deployment

You also need either Azure PowerShell or Azure Command-Line Interface (CLI) to deploy the template. If you use Azure CLI, you need to have version 2.37.0 or later. For the installation instructions, see:

- Install Azure PowerShell
- Install Azure CLI on Windows
- Install Azure CLI on Linux
- Install Azure CLI on macOS

After installing either Azure PowerShell or Azure CLI, make sure you sign in for the first time. For help, see Sign in - PowerShell or Sign in - Azure CLI.

> **ⓘ Important**
>
> If you're using Azure CLI, make sure you have version 2.37.0 or later. If you're using Azure PowerShell, make sure you have version 7.2.4 or later. The commands shown in this tutorial don't work if you're using earlier versions. To check your installed version, use: `az --version`.

Okay, you're ready to start learning about templates.

Create your first template

1. Open Visual Studio Code with the installed ARM processor Tools extension.

2. From the **File** menu, select **New File** to create a new file.

3. From the **File** menu, select **Save As**.

4. Name the file *azuredeploy* and select the *json* file extension. The complete name of the file is *azuredeploy.json*.

5. Save the file to your workstation. Select a path that's easy to remember because you need to provide that path later when deploying the template.

6. Copy and paste the following JSON into the file:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": []
}
```

Here's what your Visual Studio Code environment looks like:

This template doesn't deploy any resources. We're starting with a blank template so you can get familiar with the steps to deploy a template while minimizing the chance of something going wrong.

The JSON file has these elements:

- `$schema`: Specifies the location of the JSON schema file. The schema file describes the properties that are available within a template. The schema, for example, defines `resources` as one of the valid properties for a template. Don't worry that the date for the schema is 2019-04-01. This schema version is up to date and includes all of the latest features. The schema date hasn't been changed because there have been no breaking changes since its introduction.
- `contentVersion`: Specifies the version of the template, such as 1.0.0.0. You can provide any value for this element. Use this value to document significant changes in your template. When you deploy resources using the template, you can use this value to make sure you're using the right template.
- `resources`: Contains the resources you want to deploy or update. Currently, it's empty, but you can add resources later.

7. Save the file.

Congratulations, you've created your first template.

Sign in to Azure

To start working with Azure PowerShell or Azure CLI, sign in with your Azure credentials.

Select the tabs in the following code sections to pick between Azure PowerShell and Azure CLI. The CLI examples in this article are written for the Bash shell.

PowerShell

Azure PowerShell

```
Connect-AzAccount
```

If you have multiple Azure subscriptions, choose the subscription you want to use. Replace `SubscriptionName` with your subscription name. You can also use your subscription ID instead of your subscription name.

PowerShell

Azure PowerShell

```
Set-AzContext SubscriptionName
```

Create resource group

When you deploy a template, you can specify a resource group to contain the resources. Before running the deployment command, create the resource group with either the Bash Azure CLI or Azure PowerShell.

> **Note**
>
> Samples for the Azure CLI are written for the bash shell. To run this sample in Windows PowerShell or the Command Prompt, you may need to remove the back slashes and write the command as one line such as:

```
az group create --name myResourceGroup --location "Central US"
```

PowerShell

Azure PowerShell

```
New-AzResourceGroup `
  -Name myResourceGroup `
  -Location "Central US"
```

Deploy template

To deploy the template, use either Azure CLI or Azure PowerShell. Use the resource group you created. Give a name to the deployment so you can easily identify it in the deployment history. For convenience, also create a variable that stores the path to the template file. This variable makes it easier for you to run the deployment commands because you don't have to retype the path every time you deploy. Replace `{provide-the-path-to-the-template-file}` and the curly braces `{}` with the path to your template file.

PowerShell

Azure PowerShell

```
$templateFile = "{provide-the-path-to-the-template-file}"
New-AzResourceGroupDeployment `
  -Name blanktemplate `
  -ResourceGroupName myResourceGroup `
  -TemplateFile $templateFile
```

The deployment command returns results. Look for `ProvisioningState` to see whether the deployment succeeded.

PowerShell

```
Name                                   Account                                SubscriptionName                       Environment
----                                   -------                                ----------------                       -----------
Documentation Testing 1                                                       Documentation Testing 1                AzureCloud

PS /home/      > New-AzResourceGroup `
>>   -Name myResourceGroup `
>>   -Location "Central US"

Confirm
Provided resource group already exists. Are you sure you want to update it?
[Y] Yes  [N] No  [S] Suspend  [?] Help (default is "Y"): Y

ResourceGroupName : myResourceGroup
Location          : centralus
ProvisioningState : Succeeded
Tags              :
ResourceId        : /subscriptions/                                  /resourceGroups/myResourceGroup
```

> **Note**
>
> If the deployment fails, use the `verbose` switch to get information about the resources being created. Use the `debug` switch to get more information for debugging.

Verify deployment

You can verify the deployment by exploring the resource group from the Azure portal.

1. Sign in to the Azure portal.
2. From the left menu, select **Resource groups**.
3. Check the box to the left of **myResourceGroup** and select **myResourceGroup**.
4. Select the resource group you created in the earlier procedure. The default name is **myResourceGroup**. The resource group doesn't have any resources yet because you deployed a blank template.
5. Notice in the middle of the overview, in the **Essentials** section, the page displays the deployment status next to **Deployments**. Select **1 Succeeded**.

6. You see a history of deployment for the resource group. Check the box to the left of **blanktemplate** and select **blanktemplate**.

7. You see a summary of the deployment. In this case, there's not a lot to see because no resources are deployed. Later in this series you might find it helpful to review the summary in the deployment history. Notice on the left you can see inputs, outputs, and the template that the deployment used.

Clean up resources

If you're moving on to the next tutorial, you don't need to delete the resource group.

If you're stopping now, you might want to delete the resource group.

1. From the Azure portal, select **Resource groups** from the left menu.
2. Type the resource group name in the **Filter for any field...** text field.
3. Check the box next to **myResourceGroup** and select **myResourceGroup** or your resource group name.
4. Select **Delete resource group** from the top menu.

Next steps

You created a simple template to deploy to Azure. In the next tutorial, you can learn how to add a storage account to the template and deploy it to your resource group.

Add resource

Tutorial: Add a resource to your ARM template

Article • 04/09/2023

In the previous tutorial, you learned how to create and deploy your first blank Azure Resource Manager template (ARM template). Now, you're ready to deploy an actual resource to that template. In this case, an Azure storage account. This instruction takes **9 minutes** to complete.

Prerequisites

We recommend that you complete the introductory tutorial about templates, but it's not required.

You need to have Visual Studio Code installed and working with the Azure Resource Manager Tools extension, and either Azure PowerShell or Azure Command-Line Interface (CLI). For more information, see template tools.

Add resource

To add an Azure storage account definition to the existing template, look at the highlighted JSON file in the following example. Instead of trying to copy sections of the template, copy the whole file and replace your template with its contents.

Replace `{provide-unique-name}` and the curly braces `{}` with a unique storage account name.

> **Important**
>
> The storage account name needs to be unique across Azure. It's only lowercase letters or numbers and has a limit of 24 characters. You can use a name like **store1** as a prefix and then add your initials and today's date. The name, for example, can be **store1abc06132022**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
```

```
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-09-01",
      "name": "{provide-unique-name}",
      "location": "eastus",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "StorageV2",
      "properties": {
        "supportsHttpsTrafficOnly": true
      }
    }
  ]
}
```

Guessing a unique name for a storage account isn't easy and doesn't work well for automating large deployments. Later in this tutorial series, you use template features that make it easier to create a unique name.

Resource properties

You may be wondering how to find the properties to use for each resource type. You can use the ARM template reference to find the resource types you want to deploy.

Every resource you deploy has at least the following three properties:

- `type`: Type of the resource. This value is a combination of the namespace of the resource provider and the resource type such as `Microsoft.Storage/storageAccounts`.
- `apiVersion`: Version of the REST API to use for creating the resource. Each resource provider publishes its own API versions, so this value is specific to the type.
- `name`: Name of the resource.

Most resources also have a `location` property, which sets the region where you deploy the resource.

The other properties vary by resource type and API version. It's important to understand the connection between the API version and the available properties, so let's jump into more detail.

In this tutorial, you add a storage account to the template. You can see the storage account's API version at storageAccounts 2021-09-01. Notice that you don't add all the properties to your template. Many of the properties are optional. The

`Microsoft.Storage` resource provider could release a new API version, but the version you're deploying doesn't have to change. You can continue using that version and know that the results of your deployment are consistent.

If you view an older API version you might see that a smaller set of properties is available.

If you decide to change the API version for a resource, make sure you evaluate the properties for that version and adjust your template appropriately.

Deploy template

You can deploy the template to create the storage account. Give your deployment a different name so you can easily find it in the history.

If you haven't created the resource group, see Create resource group. The example assumes you've set the `templateFile` variable to the path to the template file, as shown in the first tutorial.

PowerShell

Azure PowerShell

```
New-AzResourceGroupDeployment `
  -Name addstorage `
  -ResourceGroupName myResourceGroup `
  -TemplateFile $templateFile
```

> **Note**
>
> If the deployment fails, use the `verbose` switch to get information about the resources you're creating. Use the `debug` switch to get more information for debugging.

These errors are two possible deployment failures that you might encounter:

- `Error: Code=AccountNameInvalid; Message={provide-unique-name}` isn't a valid storage account name. The storage account name needs to be between 3 and 24 characters in length and use numbers and lower-case letters only.

In the template, replace `{provide-unique-name}` with a unique storage account name. See Add resource.

- `Error: Code=StorageAccountAlreadyTaken; Message=The storage account named store1abc09092019` is already taken.

  In the template, try a different storage account name.

This deployment takes longer than your blank template deployment because you're creating a storage account. It can take about a minute.

Verify deployment

You can verify the deployment by exploring the resource group from the Azure portal.

1. Sign in to the Azure portal.
2. From the left menu, select **Resource groups**.
3. Check the box to the left of **myResourceGroup** and select **myResourceGroup**
4. Select the resource group you deployed to.
5. You see that a storage account has been deployed.
6. Notice that the deployment label now says: **Deployments: 2 Succeeded**.

Clean up resources

If you're moving on to the next tutorial, you don't need to delete the resource group.

If you're stopping now, you might want to clean up the resources you deployed by deleting the resource group.

1. From the Azure portal, select **Resource group** from the left menu.
2. Type the resource group name in the **Filter for any field ...** box.
3. Check the box next to myResourceGroup and select **myResourceGroup** or the resource group name you chose.
4. Select **Delete resource group** from the top menu.

Next steps

You created a simple template to deploy an Azure storage account. In the later tutorials, you learn how to add parameters, variables, resources, and outputs to a template. These features are the building blocks for much more complex templates.

Add parameters

Tutorial: Add parameters to your ARM template

Article • 03/09/2023

In the previous tutorial, you learned how to add an Azure storage account to the template and deploy it. In this tutorial, you learn how to improve the Azure Resource Manager template (ARM template) by adding parameters. This instruction takes **14 minutes** to complete.

Prerequisites

We recommend that you complete the tutorial about resources, but it's not required.

You need to have Visual Studio Code installed and working with the Azure Resource Manager Tools extension, and either Azure PowerShell or Azure Command-Line Interface (CLI). For more information, see template tools.

Review template

At the end of the previous tutorial, your template has the following JSON file:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-09-01",
      "name": "{provide-unique-name}",
      "location": "eastus",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "StorageV2",
      "properties": {
        "supportsHttpsTrafficOnly": true
      }
    }
  ]
}
```

You may notice that there's a problem with this template. The storage account name is hard-coded. You can only use this template to deploy the same storage account every time. To deploy a storage account with a different name, you need to create a new template, which obviously isn't a practical way to automate your deployments.

Make template reusable

To make your template reusable, let's add a parameter that you can use to pass in a storage account name. The highlighted JSON file in the following example shows the changes in your template. The `storageName` parameter is identified as a string. The storage account name is all lowercase letters or numbers and has a limit of 24 characters.

Copy the whole file and replace your template with its contents.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageName": {
      "type": "string",
      "minLength": 3,
      "maxLength": 24
    }
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-09-01",
      "name": "[parameters('storageName')]",
      "location": "eastus",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "StorageV2",
      "properties": {
        "supportsHttpsTrafficOnly": true
      }
    }
  ]
}
```

Deploy template

Let's deploy the template. The following example deploys the template with Azure CLI or PowerShell. Notice that you provide the storage account name as one of the values in the deployment command. For the storage account name, provide the same name you used in the previous tutorial.

If you haven't created the resource group, see Create resource group. The example assumes you set the `templateFile` variable to the path of the template file, as shown in the first tutorial.

PowerShell

Azure PowerShell

```
New-AzResourceGroupDeployment `
  -Name addnameparameter `
  -ResourceGroupName myResourceGroup `
  -TemplateFile $templateFile `
  -storageName "{your-unique-name}"
```

Understand resource updates

After you deploy a storage account with the same name you used earlier, you may wonder how the redeployment affects the resource.

If the resource already exists, and there's no change in the properties, there's no need for further action. If the resource exists and a property changes, it updates. If the resource doesn't exist, it's created.

This way of handling updates means your template can include all of the resources you need for an Azure solution. You can safely redeploy the template and know that resources change or are created only when needed. If you add files to your storage account, for example, you can redeploy the storage account without losing the files.

Customize by environment

Parameters let you customize the deployment by providing values that are tailored for a particular environment. You can pass different values, for example, based on whether you're deploying to a development, testing, or production environment.

The previous template always deploys a standard locally redundant storage (LRS) **Standard_LRS** account. You might want the flexibility to deploy different stock keeping

units (SKUs) depending on the environment. The following example shows the changes to add a parameter for SKU. Copy the whole file and paste it over your template.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageName": {
      "type": "string",
      "minLength": 3,
      "maxLength": 24
    },
    "storageSKU": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_RAGRS",
        "Standard_ZRS",
        "Premium_LRS",
        "Premium_ZRS",
        "Standard_GZRS",
        "Standard_RAGZRS"
      ]
    }
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-09-01",
      "name": "[parameters('storageName')]",
      "location": "eastus",
      "sku": {
        "name": "[parameters('storageSKU')]"
      },
      "kind": "StorageV2",
      "properties": {
        "supportsHttpsTrafficOnly": true
      }
    }
  ]
}
```

The `storageSKU` parameter has a default value. Use this value when the deployment doesn't specify it. It also has a list of allowed values. These values match the values that are needed to create a storage account. You want your template users to pass SKUs that work.

Redeploy template

You're ready to deploy again. Because the default SKU is set to **Standard_LRS**, you've already provided a parameter value.

PowerShell

Azure PowerShell

```
New-AzResourceGroupDeployment `
  -Name addskuparameter `
  -ResourceGroupName myResourceGroup `
  -TemplateFile $templateFile `
  -storageName "{your-unique-name}"
```

> **Note**
>
> If the deployment fails, use the `verbose` switch to get information about the resources being created. Use the `debug` switch to get more information for debugging.

To see the flexibility of your template, let's deploy it again. This time set the SKU parameter to standard geo-redundant storage (GRS) **Standard_GRS**. You can either pass in a new name to create a different storage account or use the same name to update your existing storage account. Both options work.

PowerShell

Azure PowerShell

```
New-AzResourceGroupDeployment `
  -Name usenondefaultsku `
  -ResourceGroupName myResourceGroup `
  -TemplateFile $templateFile `
  -storageName "{your-unique-name}" `
  -storageSKU Standard_GRS
```

Finally, let's run one more test and see what happens when you pass in an SKU that isn't one of the allowed values. In this case, we test the scenario where your template user thinks **basic** is one of the SKUs.

PowerShell

Azure PowerShell

```
New-AzResourceGroupDeployment `
  -Name testskuparameter `
  -ResourceGroupName myResourceGroup `
  -TemplateFile $templateFile `
  -storageName "{your-unique-name}" `
  -storageSKU basic
```

The command fails at once with an error message that gives the allowed values. The ARM processor finds the error before the deployment starts.

Clean up resources

If you're moving on to the next tutorial, you don't need to delete the resource group.

If you're stopping now, you might want to clean up your deployed resources by deleting the resource group.

1. From the Azure portal, select **Resource group** from the left menu.
2. Type the resource group name in the **Filter for any field ...** text field.
3. Check the box next to myResourceGroup and select **myResourceGroup** or your resource group name.
4. Select **Delete resource group** from the top menu.

Next steps

You improved the template you created in the first tutorial by adding parameters. In the next tutorial, you learn about template functions.

Add template functions

Tutorial: Add template functions to your ARM template

Article • 04/09/2023

In this tutorial, you learn how to add template functions to your Azure Resource Manager template (ARM template). You use functions to dynamically construct values. In addition to these system-provided template functions, you can also create user-defined functions. This tutorial takes **7 minutes** to complete.

Prerequisites

We recommend that you complete the tutorial about parameters, but it's not required.

You need to have Visual Studio Code installed and working with the Azure Resource Manager Tools extension, and either Azure PowerShell or Azure CLI. For more information, see template tools.

Review template

At the end of the previous tutorial, your template had the following JSON file:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageName": {
      "type": "string",
      "minLength": 3,
      "maxLength": 24
    },
    "storageSKU": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_RAGRS",
        "Standard_ZRS",
        "Premium_LRS",
        "Premium_ZRS",
        "Standard_GZRS",
        "Standard_RAGZRS"
```

```
      ]
    }
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-09-01",
      "name": "[parameters('storageName')]",
      "location": "eastus",
      "sku": {
        "name": "[parameters('storageSKU')]"
      },
      "kind": "StorageV2",
      "properties": {
        "supportsHttpsTrafficOnly": true
      }
    }
  ]
}
```

Suppose you hard-coded the location of the Azure storage account to **eastus**, but you need to deploy it to another region. You need to add a parameter to add flexibility to your template and allow it to have a different location.

Use function

If you completed the parameters tutorial, you used a function. When you added `"[parameters('storageName')]"`, you used the parameters function. The brackets indicate that the syntax inside the brackets is a template expression. Resource Manager resolves the syntax instead of treating it as a literal value.

Functions add flexibility to your template by dynamically getting values during deployment. In this tutorial, you use a function to get the resource group deployment location.

The following example highlights the changes to add a parameter called `location`. The parameter default value calls the resourceGroup function. This function returns an object with information about the deployed resource group. One of the object properties is a location property. When you use the default value, the storage account and the resource group have the same location. The resources inside a group have different locations.

Copy the whole file and replace your template with its contents.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
```

```
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageName": {
      "type": "string",
      "minLength": 3,
      "maxLength": 24
    },
    "storageSKU": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_RAGRS",
        "Standard_ZRS",
        "Premium_LRS",
        "Premium_ZRS",
        "Standard_GZRS",
        "Standard_RAGZRS"
      ]
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    }
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-09-01",
      "name": "[parameters('storageName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('storageSKU')]"
      },
      "kind": "StorageV2",
      "properties": {
        "supportsHttpsTrafficOnly": true
      }
    }
  ]
}
```

Deploy template

In the previous tutorials, you created a storage account in the East US, but your resource group is created in the Central US. For this tutorial, you create a storage account in the same region as the resource group. Use the default value for location, so you don't need to provide that parameter value. You need to provide a new name for the storage

account because you're creating a storage account in a different location. Use **store2**, for example, as the prefix instead of **store1**.

If you haven't created the resource group, see Create resource group. The example assumes you've set the `templateFile` variable to the path to the template file, as shown in the first tutorial.

PowerShell

Azure PowerShell

```
New-AzResourceGroupDeployment `
  -Name addlocationparameter `
  -ResourceGroupName myResourceGroup `
  -TemplateFile $templateFile `
  -storageName "{new-unique-name}"
```

> **Note**
>
> If the deployment fails, use the `verbose` switch to get information about the resources being created. Use the `debug` switch to get more information for debugging.

Verify deployment

You can verify the deployment by exploring the resource group from the Azure portal.

1. Sign in to the Azure portal.
2. From the left menu, select **Resource groups**.
3. Check the box to the left of **myResourceGroup** and select **myResourceGroup**.
4. Select the resource group you created. The default name is **myResourceGroup**.
5. Notice your deployed storage account and your resource group have the same location.

Clean up resources

If you're moving on to the next tutorial, you don't need to delete the resource group.

If you're stopping now, you might want to delete the resource group.

1. From the Azure portal, select **Resource groups** from the left menu.

2. Type the resource group name in the **Filter for any field...** text field.
3. Check the box next to **myResourceGroup** and select **myResourceGroup** or your resource group name.
4. Select **Delete resource group** from the top menu.

Next steps

In this tutorial, you use a function to define the default value for a parameter. In this tutorial series, you continue to use functions. By the end of the series, you add functions to every template section.

Add variables

Tutorial: Add variables to your ARM template

Article • 04/09/2023

In this tutorial, you learn how to add a variable to your Azure Resource Manager template (ARM template). Variables simplify your templates. They let you write an expression once and reuse it throughout the template. This tutorial takes **7 minutes** to complete.

Prerequisites

We recommend that you complete the tutorial about functions, but it's not required.

You need to have Visual Studio Code installed and working with the Azure Resource Manager Tools extension, and either Azure PowerShell or Azure CLI. For more information, see template tools.

Review template

At the end of the previous tutorial, your template had the following JSON file:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageName": {
      "type": "string",
      "minLength": 3,
      "maxLength": 24
    },
    "storageSKU": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_RAGRS",
        "Standard_ZRS",
        "Premium_LRS",
        "Premium_ZRS",
        "Standard_GZRS",
        "Standard_RAGZRS"
```

```
      ]
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    }
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-09-01",
      "name": "[parameters('storageName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('storageSKU')]"
      },
      "kind": "StorageV2",
      "properties": {
        "supportsHttpsTrafficOnly": true
      }
    }
  ]
}
```

Your Azure storage account name needs to be unique to easily continue to build your ARM template. If you've completed the earlier tutorials in this series, you're tired of coming up with a unique name. You solve this problem by adding a variable that creates a unique name for your storage account.

Use variable

The following example highlights the changes to add a variable to your template that creates a unique storage account name. Copy the whole file and replace your template with its contents.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storagePrefix": {
      "type": "string",
      "minLength": 3,
      "maxLength": 11
    },
    "storageSKU": {
      "type": "string",
      "defaultValue": "Standard_LRS",
```

```
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_RAGRS",
        "Standard_ZRS",
        "Premium_LRS",
        "Premium_ZRS",
        "Standard_GZRS",
        "Standard_RAGZRS"
      ]
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    }
  },
  "variables": {
    "uniqueStorageName": "[concat(parameters('storagePrefix'),
uniqueString(resourceGroup().id))]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-09-01",
      "name": "[variables('uniqueStorageName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('storageSKU')]"
      },
      "kind": "StorageV2",
      "properties": {
        "supportsHttpsTrafficOnly": true
      }
    }
  ]
}
```

Notice that it includes a variable named `uniqueStorageName`. This variable uses four functions to create a string value.

You're already familiar with the parameters function, so we won't examine it.

You're also familiar with the resourceGroup function. In this case, you get the `id` property instead of the `location` property, as shown in the previous tutorial. The `id` property returns the full identifier of the resource group, including the subscription ID and the resource group name.

The uniqueString function creates a 13-character hash value. The parameters you pass determine the returned value. For this tutorial, you use the resource group ID as the input for the hash value. That means you could deploy this template to different

resource groups and get a different unique string value. You get the same value, however, if you deploy to the same resource group.

The concat function takes values and combines them. For this variable, it takes the string from the parameter and the string from the `uniqueString` function and combines them into one string.

The `storagePrefix` parameter lets you pass in a prefix that helps you identify storage accounts. You can create your own naming convention that makes it easier to identify storage accounts after deployment from an extensive list of resources.

Finally, notice that the storage account name is now set to the variable instead of a parameter.

Deploy template

Let's deploy the template. Deploying this template is easier than the previous templates because you provide just the prefix for the storage account name.

If you haven't created the resource group, see Create resource group. The example assumes you've set the `templateFile` variable to the path to the template file, as shown in the first tutorial.

PowerShell

Azure PowerShell

```
New-AzResourceGroupDeployment `
  -Name addnamevariable `
  -ResourceGroupName myResourceGroup `
  -TemplateFile $templateFile `
  -storagePrefix "store" `
  -storageSKU Standard_LRS
```

Note

If the deployment fails, use the `verbose` switch to get information about the resources being created. Use the `debug` switch to get more information for debugging.

Verify deployment

You can verify the deployment by exploring the resource group from the Azure portal.

1. Sign in to the Azure portal.
2. From the left menu, select **Resource groups**.
3. Select your resource group.
4. Notice your deployed storage account name is **store**, plus a string of random characters.

Clean up resources

If you're moving on to the next tutorial, you don't need to delete the resource group.

If you're stopping now, you might want to delete the resource group.

1. From the Azure portal, select **Resource groups** from the left menu.
2. Type the resource group name in the **Filter for any field...** text field.
3. Check the box next to **myResourceGroup** and select **myResourceGroup** or your resource group name.
4. Select **Delete resource group** from the top menu.

Next steps

In this tutorial, you add a variable that creates a unique storage account name. In the next tutorial, you return a value from the deployed storage account.

Add outputs

Tutorial: Add outputs to your ARM template

Article • 04/09/2023

In this tutorial, you learn how to return a value from your Azure Resource Manager template (ARM template). You use outputs when you need a value for a resource you deploy. This tutorial takes **7 minutes** to complete.

Prerequisites

We recommend that you complete the tutorial about variables, but it's not required.

You need to have Visual Studio Code with the Resource Manager Tools extension, and either Azure PowerShell or Azure Command-Line Interface (CLI). For more information, see template tools.

Review template

At the end of the previous tutorial, your template had the following JSON:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storagePrefix": {
      "type": "string",
      "minLength": 3,
      "maxLength": 11
    },
    "storageSKU": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_RAGRS",
        "Standard_ZRS",
        "Premium_LRS",
        "Premium_ZRS",
        "Standard_GZRS",
        "Standard_RAGZRS"
      ]
```

```
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    }
  },
  "variables": {
    "uniqueStorageName": "[concat(parameters('storagePrefix'),
uniqueString(resourceGroup().id))]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-09-01",
      "name": "[variables('uniqueStorageName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('storageSKU')]"
      },
      "kind": "StorageV2",
      "properties": {
        "supportsHttpsTrafficOnly": true
      }
    }
  ]
}
```

It deploys a storage account, but it doesn't return any information about it. You might need to capture properties from your new resource so they're available later for reference.

Add outputs

You can use outputs to return values from the template. It might be helpful, for example, to get the endpoints for your new storage account.

The following example highlights the change to your template to add an output value. Copy the whole file and replace your template with its contents.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storagePrefix": {
      "type": "string",
      "minLength": 3,
      "maxLength": 11
```

```
    },
    "storageSKU": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_RAGRS",
        "Standard_ZRS",
        "Premium_LRS",
        "Premium_ZRS",
        "Standard_GZRS",
        "Standard_RAGZRS"
      ]
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    }
  },
  "variables": {
    "uniqueStorageName": "[concat(parameters('storagePrefix'),
uniqueString(resourceGroup().id))]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-09-01",
      "name": "[variables('uniqueStorageName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('storageSKU')]"
      },
      "kind": "StorageV2",
      "properties": {
        "supportsHttpsTrafficOnly": true
      }
    }
  ],
  "outputs": {
    "storageEndpoint": {
      "type": "object",
      "value": "
[reference(variables('uniqueStorageName')).primaryEndpoints]"
    }
  }
}
```

There are some important items to note about the output value you added.

The type of returned value is set to `object`, which means it returns a JSON object.

It uses the reference function to get the runtime state of the storage account. To get the runtime state of a resource, pass the name or ID of a resource. In this case, you use the same variable you used to create the name of the storage account.

Finally, it returns the `primaryEndpoints` property from the storage account.

Deploy template

You're ready to deploy the template and look at the returned value.

If you haven't created the resource group, see Create resource group. The example assumes you've set the `templateFile` variable to the path to the template file, as shown in the first tutorial.

PowerShell

Azure PowerShell

```
New-AzResourceGroupDeployment `
  -Name addoutputs `
  -ResourceGroupName myResourceGroup `
  -TemplateFile $templateFile `
  -storagePrefix "store" `
  -storageSKU Standard_LRS
```

In the output for the deployment command, you see an object similar to the following example only if the output is in JSON format:

JSON

```
{
    "dfs": "https://storeluktbfkpjjrkm.dfs.core.windows.net/",
    "web": "https://storeluktbfkpjjrkm.z19.web.core.windows.net/",
    "blob": "https://storeluktbfkpjjrkm.blob.core.windows.net/",
    "queue": "https://storeluktbfkpjjrkm.queue.core.windows.net/",
    "table": "https://storeluktbfkpjjrkm.table.core.windows.net/",
    "file": "https://storeluktbfkpjjrkm.file.core.windows.net/"
}
```

Note

If the deployment fails, use the `verbose` switch to get information about the resources being created. Use the `debug` switch to get more information for

debugging.

Review your work

You've done a lot in the last six tutorials. Let's take a moment to review what you've done. You created a template with parameters that are easy to provide. The template is reusable in different environments because it allows for customization and dynamically creates needed values. It also returns information about the storage account that you could use in your script.

Now, let's look at the resource group and deployment history.

1. Sign in to the Azure portal.
2. From the left menu, select **Resource groups**.
3. Select the resource group you deployed to.
4. Depending on the steps you did, you should have at least one and perhaps several storage accounts in the resource group.
5. You should also have several successful deployments listed in the history. Select that link.

6. You see all of your deployments in the history. Select the deployment called **addoutputs**.

7. You can review the inputs.

8. You can review the outputs.

9. You can review the template.

Clean up resources

If you're moving on to the next tutorial, you don't need to delete the resource group.

If you're stopping now, you might want to delete the resource group.

1. From the Azure portal, select **Resource groups** from the left menu.
2. Type the resource group name in the **Filter for any field...** text field.
3. Check the box next to **myResourceGroup** and select **myResourceGroup** or your resource group name.
4. Select **Delete resource group** from the top menu.

Next steps

In this tutorial, you added a return value to the template. In the next tutorial, you learn how to export a template and use parts of that exported template in your template.

Use exported template

Tutorial: Use exported template from the Azure portal

Article • 08/18/2022

In this tutorial series, you create a template to deploy an Azure storage account. In the next two tutorials, you add an **App Service plan** and a **website**. Instead of creating templates from scratch, you learn how to export templates from the Azure portal and how to use sample templates from the Azure Quickstart Templates. You customize those templates for your use. This tutorial focuses on exporting templates and customizing the result for your template. This instruction takes **14 minutes** to complete.

Prerequisites

We recommend that you complete the tutorial about outputs, but it's not required.

You need to have Visual Studio Code with the Resource Manager Tools extension and either Azure PowerShell or Azure Command-Line Interface (CLI). For more information, see template tools.

Review template

At the end of the previous tutorial, your template had the following JSON file:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storagePrefix": {
      "type": "string",
      "minLength": 3,
      "maxLength": 11
    },
    "storageSKU": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_RAGRS",
        "Standard_ZRS",
        "Premium_LRS",
```

```
          "Premium_ZRS",
          "Standard_GZRS",
          "Standard_RAGZRS"
        ]
      },
      "location": {
        "type": "string",
        "defaultValue": "[resourceGroup().location]"
      }
    },
    "variables": {
      "uniqueStorageName": "[concat(parameters('storagePrefix'),
uniqueString(resourceGroup().id))]"
    },
    "resources": [
      {
        "type": "Microsoft.Storage/storageAccounts",
        "apiVersion": "2021-09-01",
        "name": "[variables('uniqueStorageName')]",
        "location": "[parameters('location')]",
        "sku": {
          "name": "[parameters('storageSKU')]"
        },
        "kind": "StorageV2",
        "properties": {
          "supportsHttpsTrafficOnly": true
        }
      }
    ],
    "outputs": {
      "storageEndpoint": {
        "type": "object",
        "value": "
[reference(variables('uniqueStorageName')).primaryEndpoints]"
      }
    }
  }
```

This template works well for deploying storage accounts, but you might want to add more resources to it. You can export a template from an existing resource to quickly get the JSON for that resource.

Create App Service plan

1. Sign in to the Azure portal.

2. Select **Create a resource**.

3. In **Search services and Marketplace**, enter **App Service Plan**, and then select **App Service Plan**.

4. Select **Create**.

5. On the **Create App Service Plan** page, enter the following:

- **Subscription**: Select your Azure subscription from the drop-down menu.
- **Resource Group**: Select **Create new** and then specify a name. Provide a different resource group name than the one you've been using in this tutorial series.
- **Name**: enter a name for the App Service Plan.
- **Operating System**: Select **Linux**.
- **Region**: Select an Azure location from the drop-down menu, such as **Central US**.
- **Pricing Tier**: To save costs, select **Change size** to change the **SKU and size** to **first Basic (B1)**, under **Dev / Test** for less demanding workloads.

6. Select **Review and create**.

7. Select **Create**. It takes a few moments to create the resource.

Export template

1. Select **Go to resource**.

2. From the left menu, under **Automation**, select **Export template**.

The export template feature takes the current state of a resource and generates a template to deploy it. Exporting a template can be a helpful way of quickly getting the JSON you need to deploy a resource.

3. Look at the `Microsoft.Web/serverfarms` definition and the parameter definition in the exported template. You don't need to copy these sections. You can just use this exported template as an example of how you want to add this resource to your template.

```
1  {
2      "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
3      "contentVersion": "1.0.0.0",
4      "parameters": {
5          "serverfarms_myAppServicePlan_name": {
6              "defaultValue": "myAppServicePlan",
7              "type": "String"
8          }
9      },
10     "variables": {},
11     "resources": [
12         {
13             "type": "Microsoft.Web/serverfarms",
14             "apiVersion": "2016-09-01",
15             "name": "[parameters('serverfarms_myAppServicePlan_name')]",
16             "location": "Central US",
17             "sku": {
18                 "name": "B1",
19                 "tier": "Basic",
20                 "size": "B1",
21                 "family": "B",
22                 "capacity": 1
23             },
24             "kind": "linux",
25             "properties": {
26                 "name": "[parameters('serverfarms_myAppServicePlan_name')]",
27                 "perSiteScaling": false,
28                 "reserved": true,
29                 "targetWorkerCount": 0,
30                 "targetWorkerSizeId": 0
31             }
32         }
33     ]
34 }
```

> **Important**
>
> Typically, the exported template is more verbose than you might want when creating a template. The SKU object, for example, in the exported template has five properties. This template works, but you could just use the `name` property. You can start with the exported template and then modify it as you like to fit your requirements.

Revise existing template

The exported template gives you most of the JSON you need, but you have to customize it for your template. Pay particular attention to differences in parameters and variables between your template and the exported template. Obviously, the export process doesn't know the parameters and variables that you've already defined in your template.

The following example highlights the additions to your template. It contains the exported code plus some changes. First, it changes the name of the parameter to match your naming convention. Second, it uses your location parameter for the location of the app service plan. Third, it removes some of the properties where the default value is fine.

Copy the whole file and replace your template with its contents.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storagePrefix": {
      "type": "string",
      "minLength": 3,
      "maxLength": 11
    },
    "storageSKU": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_RAGRS",
        "Standard_ZRS",
        "Premium_LRS",
        "Premium_ZRS",
        "Standard_GZRS",
        "Standard_RAGZRS"
      ]
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    },
    "appServicePlanName": {
      "type": "string",
      "defaultValue": "exampleplan"
    }
  },
  "variables": {
    "uniqueStorageName": "[concat(parameters('storagePrefix'), uniqueString(resourceGroup().id))]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-09-01",
      "name": "[variables('uniqueStorageName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('storageSKU')]"
      },
      "kind": "StorageV2",
      "properties": {
        "supportsHttpsTrafficOnly": true
      }
```

```
    },
    {
      "type": "Microsoft.Web/serverfarms",
      "apiVersion": "2021-03-01",
      "name": "[parameters('appServicePlanName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "B1",
        "tier": "Basic",
        "size": "B1",
        "family": "B",
        "capacity": 1
      },
      "kind": "linux",
      "properties": {
        "perSiteScaling": false,
        "reserved": true,
        "targetWorkerCount": 0,
        "targetWorkerSizeId": 0
      }
    }
  ],
  "outputs": {
    "storageEndpoint": {
      "type": "object",
      "value": "
[reference(variables('uniqueStorageName')).primaryEndpoints]"
    }
  }
}
```

Deploy template

Use either Azure CLI or Azure PowerShell to deploy a template.

If you haven't created the resource group, see Create resource group. The example assumes you've set the `templateFile` variable to the path to the template file, as shown in the first tutorial.

PowerShell

Azure PowerShell

```
New-AzResourceGroupDeployment `
  -Name addappserviceplan `
  -ResourceGroupName myResourceGroup `
  -TemplateFile $templateFile `
  -storagePrefix "store" `
  -storageSKU Standard_LRS
```

> **Note**
>
> If the deployment fails, use the `verbose` switch to get information about the resources you're creating. Use the `debug` switch to get more information for debugging.

Verify deployment

You can verify the deployment by exploring the resource group from the Azure portal.

1. Sign in to the Azure portal.
2. From the left menu, select **Resource groups**.
3. Select the resource group you deployed to.
4. The resource group contains a storage account and an App Service Plan.

Clean up resources

If you're moving on to the next tutorial, you don't need to delete the resource group.

If you're stopping now, you might want to delete the resource group.

1. From the Azure portal, select **Resource groups** from the left menu.
2. Type the resource group name in the **Filter for any field...** text field.
3. Check the box next to **myResourceGroup** and select **myResourceGroup** or your resource group name.
4. Select **Delete resource group** from the top menu.

Next steps

You learned how to export a template from the Azure portal and how to use the exported template for your template development. You can also use the Azure Quickstart Templates to simplify template development.

Use Azure Quickstart Templates

Tutorial: Use Azure Quickstart Templates

Article • 08/19/2022

Azure Quickstart Templates is a repository of community contributed templates. You can use the sample templates in your template development. In this tutorial, you find a website resource definition and add it to your own template. This instruction takes **12 minutes** to complete.

Prerequisites

We recommend that you complete the tutorial about exported templates, but it's not required.

You need to have Visual Studio Code with the Resource Manager Tools extension, and either Azure PowerShell or Azure Command-Line Interface (CLI). For more information, see template tools.

Review template

At the end of the previous tutorial, your template had the following JSON file:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storagePrefix": {
      "type": "string",
      "minLength": 3,
      "maxLength": 11
    },
    "storageSKU": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_RAGRS",
        "Standard_ZRS",
        "Premium_LRS",
        "Premium_ZRS",
        "Standard_GZRS",
        "Standard_RAGZRS"
      ]
```

```
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    },
    "appServicePlanName": {
      "type": "string",
      "defaultValue": "exampleplan"
    }
  },
  "variables": {
    "uniqueStorageName": "[concat(parameters('storagePrefix'),
uniqueString(resourceGroup().id))]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-09-01",
      "name": "[variables('uniqueStorageName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('storageSKU')]"
      },
      "kind": "StorageV2",
      "properties": {
        "supportsHttpsTrafficOnly": true
      }
    },
    {
      "type": "Microsoft.Web/serverfarms",
      "apiVersion": "2021-03-01",
      "name": "[parameters('appServicePlanName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "B1",
        "tier": "Basic",
        "size": "B1",
        "family": "B",
        "capacity": 1
      },
      "kind": "linux",
      "properties": {
        "perSiteScaling": false,
        "reserved": true,
        "targetWorkerCount": 0,
        "targetWorkerSizeId": 0
      }
    }
  ],
  "outputs": {
    "storageEndpoint": {
      "type": "object",
      "value": "
[reference(variables('uniqueStorageName')).primaryEndpoints]"
    }
```

```
  }
}
```

This template works for deploying storage accounts and app service plans, but you might want to add a website to it. You can use pre-built templates to quickly discover the JSON required for deploying a resource.

Find template

1. Open Azure Quickstart Templates
2. Select the tile with the title **Deploy a basic Linux web app**. If you have trouble finding it, here's the direct link .
3. Select **Browse on GitHub**.
4. Select *azuredeploy.json*.
5. Review the template. Look for the `Microsoft.Web/sites` resource.

```
54      {
55        "type": "Microsoft.Web/sites",
56        "apiVersion": "2016-08-01",
57        "kind": "app",
58        "name": "[variables('webAppPortalName')]",
59        "location": "[parameters('location')]",
60        "properties": {
61          "serverFarmId": "[resourceId('Microsoft.Web/serverfarms', variables('appServicePlanName'))]",
62          "siteConfig": {
63              "linuxFxVersion": "[parameters('linuxFxVersion')]"
64            }
65        },
66        "dependsOn": [
67          "[resourceId('Microsoft.Web/serverfarms', variables('appServicePlanName'))]"
68        ]
69      }
70    ]
71  }
```

Revise existing template

Merge the quickstart template with the existing template:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
```

```
    "storagePrefix": {
      "type": "string",
      "minLength": 3,
      "maxLength": 11
    },
    "storageSKU": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_RAGRS",
        "Standard_ZRS",
        "Premium_LRS",
        "Premium_ZRS",
        "Standard_GZRS",
        "Standard_RAGZRS"
      ]
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    },
    "appServicePlanName": {
      "type": "string",
      "defaultValue": "exampleplan"
    },
    "webAppName": {
      "type": "string",
      "metadata": {
        "description": "Base name of the resource such as web app name and
app service plan "
      },
      "minLength": 2
    },
    "linuxFxVersion": {
      "type": "string",
      "defaultValue": "php|7.0",
      "metadata": {
        "description": "The Runtime stack of current web app"
      }
    }
  },
  "variables": {
    "uniqueStorageName": "[concat(parameters('storagePrefix'),
uniqueString(resourceGroup().id))]",
    "webAppPortalName": "[concat(parameters('webAppName'),
uniqueString(resourceGroup().id))]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-09-01",
      "name": "[variables('uniqueStorageName')]",
      "location": "[parameters('location')]",
```

```
      "sku": {
        "name": "[parameters('storageSKU')]"
      },
      "kind": "StorageV2",
      "properties": {
        "supportsHttpsTrafficOnly": true
      }
    },
    {
      "type": "Microsoft.Web/serverfarms",
      "apiVersion": "2021-03-01",
      "name": "[parameters('appServicePlanName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "B1",
        "tier": "Basic",
        "size": "B1",
        "family": "B",
        "capacity": 1
      },
      "kind": "linux",
      "properties": {
        "perSiteScaling": false,
        "reserved": true,
        "targetWorkerCount": 0,
        "targetWorkerSizeId": 0
      }
    },
    {
      "type": "Microsoft.Web/sites",
      "apiVersion": "2021-03-01",
      "name": "[variables('webAppPortalName')]",
      "location": "[parameters('location')]",
      "dependsOn": [
        "[resourceId('Microsoft.Web/serverfarms',
parameters('appServicePlanName'))]"
      ],
      "kind": "app",
      "properties": {
        "serverFarmId": "[resourceId('Microsoft.Web/serverfarms',
parameters('appServicePlanName'))]",
        "siteConfig": {
          "linuxFxVersion": "[parameters('linuxFxVersion')]"
        }
      }
    }
  ],
  "outputs": {
    "storageEndpoint": {
      "type": "object",
      "value": "
[reference(variables('uniqueStorageName')).primaryEndpoints]"
    }
  }
}
```

The web app name needs to be unique across Azure. To prevent having duplicate names, the `webAppPortalName` variable is updated from `"webAppPortalName": "[concat(parameters('webAppName'), '-webapp')]"` to `"webAppPortalName": "[concat(parameters('webAppName'), uniqueString(resourceGroup().id))]"`.

Add a comma at the end of the `Microsoft.Web/serverfarms` definition to separate the resource definition from the `Microsoft.Web/sites` definition.

There are a couple of important features to note in this new resource.

It has an element named `dependsOn` that's set to the app service plan. This setting is required because the app service plan needs to exist before the web app is created. The `dependsOn` element tells Resource Manager how to order the resources for deployment.

The `serverFarmId` property uses the resourceId function. This function gets the unique identifier for a resource. In this case, it gets the unique identifier for the app service plan. The web app is associated with one specific app service plan.

Deploy template

Use either Azure CLI or Azure PowerShell to deploy a template.

If you haven't created the resource group, see Create resource group. The example assumes you've set the **templateFile** variable to the path to the template file, as shown in the first tutorial.

PowerShell

Azure PowerShell

```
New-AzResourceGroupDeployment `
  -Name addwebapp `
  -ResourceGroupName myResourceGroup `
  -TemplateFile $templateFile `
  -storagePrefix "store" `
  -storageSKU Standard_LRS `
  -webAppName demoapp
```

Note

If the deployment fails, use the `verbose` switch to get information about the resources you're creating. Use the `debug` switch to get more information for debugging.

Clean up resources

If you're moving on to the next tutorial, you don't need to delete the resource group.

If you're stopping now, you might want to delete the resource group.

1. From the Azure portal, select **Resource groups** from the left menu.
2. Type the resource group name in the **Filter for any field...** text field.
3. Check the box next to **myResourceGroup** and select **myResourceGroup** or your resource group name.
4. Select **Delete resource group** from the top menu.

Next steps

You learned how to use a quickstart template for your template development. In the next tutorial, you add tags to the resources.

Add tags

Tutorial: Add tags in your ARM template

Article • 04/09/2023

In this tutorial, you learn how to add tags to resources in your Azure Resource Manager template (ARM template). Tags are metadata elements made up of key-value pairs that help you identify resources and show up in cost reports. This instruction takes **8 minutes** to complete.

Prerequisites

We recommend that you complete the tutorial about Quickstart Templates, but it's not required.

You need to have Visual Studio Code with the Resource Manager Tools extension and either Azure PowerShell or Azure Command-Line Interface (CLI). For more information, see template tools.

Review template

Your previous template deployed a storage account, an App Service plan, and a web app.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storagePrefix": {
      "type": "string",
      "minLength": 3,
      "maxLength": 11
    },
    "storageSKU": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_RAGRS",
        "Standard_ZRS",
        "Premium_LRS",
        "Premium_ZRS",
        "Standard_GZRS",
```

```
        "Standard_RAGZRS"
      ]
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    },
    "appServicePlanName": {
      "type": "string",
      "defaultValue": "exampleplan"
    },
    "webAppName": {
      "type": "string",
      "metadata": {
        "description": "Base name of the resource such as web app name and
app service plan "
      },
      "minLength": 2
    },
    "linuxFxVersion": {
      "type": "string",
      "defaultValue": "php|7.0",
      "metadata": {
        "description": "The Runtime stack of current web app"
      }
    }
  },
  "variables": {
    "uniqueStorageName": "[concat(parameters('storagePrefix'),
uniqueString(resourceGroup().id))]",
    "webAppPortalName": "[concat(parameters('webAppName'),
uniqueString(resourceGroup().id))]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-09-01",
      "name": "[variables('uniqueStorageName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('storageSKU')]"
      },
      "kind": "StorageV2",
      "properties": {
        "supportsHttpsTrafficOnly": true
      }
    },
    {
      "type": "Microsoft.Web/serverfarms",
      "apiVersion": "2021-03-01",
      "name": "[parameters('appServicePlanName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "B1",
        "tier": "Basic",
```

```
        "size": "B1",
        "family": "B",
        "capacity": 1
      },
      "kind": "linux",
      "properties": {
        "perSiteScaling": false,
        "reserved": true,
        "targetWorkerCount": 0,
        "targetWorkerSizeId": 0
      }
    },
    {
      "type": "Microsoft.Web/sites",
      "apiVersion": "2021-03-01",
      "name": "[variables('webAppPortalName')]",
      "location": "[parameters('location')]",
      "dependsOn": [
        "[resourceId('Microsoft.Web/serverfarms',
parameters('appServicePlanName'))]"
      ],
      "kind": "app",
      "properties": {
        "serverFarmId": "[resourceId('Microsoft.Web/serverfarms',
parameters('appServicePlanName'))]",
        "siteConfig": {
          "linuxFxVersion": "[parameters('linuxFxVersion')]"
        }
      }
    }
  ],
  "outputs": {
    "storageEndpoint": {
      "type": "object",
      "value": "
[reference(variables('uniqueStorageName')).primaryEndpoints]"
    }
  }
}
```

After you deploy these resources, you might need to track costs and find resources that belong to a category. You can add tags to help solve these issues.

Add tags

You tag resources to add values that help you identify their use. You can add tags that list the environment and the project. You can also add them to identify a cost center or the team that owns the resource. Add any values that make sense for your organization.

The following example highlights the changes to the template. Copy the whole file and replace your template with its contents.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storagePrefix": {
      "type": "string",
      "minLength": 3,
      "maxLength": 11
    },
    "storageSKU": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_RAGRS",
        "Standard_ZRS",
        "Premium_LRS",
        "Premium_ZRS",
        "Standard_GZRS",
        "Standard_RAGZRS"
      ]
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    },
    "appServicePlanName": {
      "type": "string",
      "defaultValue": "exampleplan"
    },
    "webAppName": {
      "type": "string",
      "metadata": {
        "description": "Base name of the resource such as web app name and
app service plan "
      },
      "minLength": 2
    },
    "linuxFxVersion": {
      "type": "string",
      "defaultValue": "php|7.0",
      "metadata": {
        "description": "The Runtime stack of current web app"
      }
    },
```

```
    "resourceTags": {
      "type": "object",
      "defaultValue": {
        "Environment": "Dev",
        "Project": "Tutorial"
      }
    }
  },
  "variables": {
    "uniqueStorageName": "[concat(parameters('storagePrefix'),
uniqueString(resourceGroup().id))]",
    "webAppPortalName": "[concat(parameters('webAppName'),
uniqueString(resourceGroup().id))]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-09-01",
      "name": "[variables('uniqueStorageName')]",
      "location": "[parameters('location')]",
      "tags": "[parameters('resourceTags')]",
      "sku": {
        "name": "[parameters('storageSKU')]"
      },
      "kind": "StorageV2",
      "properties": {
        "supportsHttpsTrafficOnly": true
      }
    },
    {
      "type": "Microsoft.Web/serverfarms",
      "apiVersion": "2021-03-01",
      "name": "[parameters('appServicePlanName')]",
      "location": "[parameters('location')]",
      "tags": "[parameters('resourceTags')]",
      "sku": {
        "name": "B1",
        "tier": "Basic",
        "size": "B1",
        "family": "B",
        "capacity": 1
      },
      "kind": "linux",
      "properties": {
        "perSiteScaling": false,
        "reserved": true,
        "targetWorkerCount": 0,
        "targetWorkerSizeId": 0
      }
    },
    {
      "type": "Microsoft.Web/sites",
      "apiVersion": "2021-03-01",
      "name": "[variables('webAppPortalName')]",
      "location": "[parameters('location')]",
```

```
      "dependsOn": [
        "[parameters('appServicePlanName')]"
      ],
      "tags": "[parameters('resourceTags')]",
      "kind": "app",
      "properties": {
        "serverFarmId": "[resourceId('Microsoft.Web/serverfarms',
parameters('appServicePlanName'))]",
        "siteConfig": {
          "linuxFxVersion": "[parameters('linuxFxVersion')]"
        }
      }
    }
  ],
  "outputs": {
    "storageEndpoint": {
      "type": "object",
      "value": "
[reference(variables('uniqueStorageName')).primaryEndpoints]"
    }
  }
}
```

Deploy template

It's time to deploy the template and look at the results.

If you haven't created the resource group, see Create resource group. The example assumes you've set the `templateFile` variable to the path to the template file, as shown in the first tutorial.

PowerShell

Azure PowerShell

```
New-AzResourceGroupDeployment `
  -Name addtags `
  -ResourceGroupName myResourceGroup `
  -TemplateFile $templateFile `
  -storagePrefix "store" `
  -storageSKU Standard_LRS `
  -webAppName demoapp
```

ⓘ Note

If the deployment fails, use the `verbose` switch to get information about the resources you're creating. Use the `debug` switch to get more information for debugging.

Verify deployment

You can verify the deployment by exploring the resource group from the Azure portal.

1. Sign in to the Azure portal.
2. From the left menu, select **Resource groups**.
3. Select the resource group you deployed to.
4. Select one of the resources, such as the storage account resource. You see that it now has tags.

Clean up resources

If you're moving on to the next tutorial, you don't need to delete the resource group.

If you're stopping now, you might want to delete the resource group.

1. From the Azure portal, select **Resource groups** from the left menu.
2. Type the resource group name in the **Filter for any field...** text field.
3. Check the box next to **myResourceGroup** and select **myResourceGroup** or your resource group name.
4. Select **Delete resource group** from the top menu.

Next steps

In this tutorial, you add tags to the resources. In the next tutorial, you learn how to use parameter files to simplify passing in values to the template.

Use parameter file

Tutorial: Use parameter files to deploy your ARM template

Article • 03/09/2023

In this tutorial, you learn how to use parameter files to store the values you pass in during deployment. In the previous tutorials, you used inline parameters with your deployment command. This approach worked for testing your Azure Resource Manager template (ARM template), but when automating deployments it can be easier to pass a set of values for your environment. Parameter files make it easier to package parameter values for a specific environment. In this tutorial, you create parameter files for development and production environments. This instruction takes **12 minutes** to complete.

Prerequisites

We recommend that you complete the tutorial about tags, but it's not required.

You need to have Visual Studio Code with the Resource Manager Tools extension, and either Azure PowerShell or Azure Command-Line Interface (CLI). For more information, see template tools.

Review template

Your template has many parameters you can provide during deployment. At the end of the previous tutorial, your template had the following JSON file:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storagePrefix": {
      "type": "string",
      "minLength": 3,
      "maxLength": 11
    },
    "storageSKU": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
```

```
        "Standard_GRS",
        "Standard_RAGRS",
        "Standard_ZRS",
        "Premium_LRS",
        "Premium_ZRS",
        "Standard_GZRS",
        "Standard_RAGZRS"
      ]
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    },
    "appServicePlanName": {
      "type": "string",
      "defaultValue": "exampleplan"
    },
    "webAppName": {
      "type": "string",
      "metadata": {
        "description": "Base name of the resource such as web app name and
app service plan "
      },
      "minLength": 2
    },
    "linuxFxVersion": {
      "type": "string",
      "defaultValue": "php|7.0",
      "metadata": {
        "description": "The Runtime stack of current web app"
      }
    },
    "resourceTags": {
      "type": "object",
      "defaultValue": {
        "Environment": "Dev",
        "Project": "Tutorial"
      }
    }
  },
  "variables": {
    "uniqueStorageName": "[concat(parameters('storagePrefix'),
uniqueString(resourceGroup().id))]",
    "webAppPortalName": "[concat(parameters('webAppName'),
uniqueString(resourceGroup().id))]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-09-01",
      "name": "[variables('uniqueStorageName')]",
      "location": "[parameters('location')]",
      "tags": "[parameters('resourceTags')]",
      "sku": {
        "name": "[parameters('storageSKU')]"
```

```
      },
      "kind": "StorageV2",
      "properties": {
        "supportsHttpsTrafficOnly": true
      }
    },
    {
      "type": "Microsoft.Web/serverfarms",
      "apiVersion": "2021-03-01",
      "name": "[parameters('appServicePlanName')]",
      "location": "[parameters('location')]",
      "tags": "[parameters('resourceTags')]",
      "sku": {
        "name": "B1",
        "tier": "Basic",
        "size": "B1",
        "family": "B",
        "capacity": 1
      },
      "kind": "linux",
      "properties": {
        "perSiteScaling": false,
        "reserved": true,
        "targetWorkerCount": 0,
        "targetWorkerSizeId": 0
      }
    },
    {
      "type": "Microsoft.Web/sites",
      "apiVersion": "2021-03-01",
      "name": "[variables('webAppPortalName')]",
      "location": "[parameters('location')]",
      "dependsOn": [
        "[parameters('appServicePlanName')]"
      ],
      "tags": "[parameters('resourceTags')]",
      "kind": "app",
      "properties": {
        "serverFarmId": "[resourceId('Microsoft.Web/serverfarms',
parameters('appServicePlanName'))]",
        "siteConfig": {
          "linuxFxVersion": "[parameters('linuxFxVersion')]"
        }
      }
    }
  ],
  "outputs": {
    "storageEndpoint": {
      "type": "object",
      "value": "
[reference(variables('uniqueStorageName')).primaryEndpoints]"
    }
  }
}
```

This template works well, but now you want to easily manage the parameters that you pass in for the template.

Add parameter files

Parameter files are JSON files with a structure that's similar to your template. In the file, you provide the parameter values you want to pass in during deployment.

Within the parameter file, you provide values for the parameters in your template. The name of each parameter in your parameter file needs to match the name of a parameter in your template. The name is case-insensitive but to easily see the matching values we recommend that you match the casing from the template.

You don't have to provide a value for every parameter. If an unspecified parameter has a default value, that value is used during deployment. If a parameter doesn't have a default value and isn't specified in the parameter file, you're prompted to provide a value during deployment.

You can't specify a parameter name in your parameter file that doesn't match a parameter name in the template. You get an error when you provide unknown parameters.

In Visual Studio Code, create a new file with the following content. Save the file with the name *azuredeploy.parameters.dev.json*.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storagePrefix": {
      "value": "devstore"
    },
    "storageSKU": {
      "value": "Standard_LRS"
    },
    "appServicePlanName": {
      "value": "devplan"
    },
    "webAppName": {
      "value": "devapp"
    },
    "resourceTags": {
      "value": {
        "Environment": "Dev",
        "Project": "Tutorial"
```

```
        }
      }
    }
  }
```

This file is your parameter file for the development environment. Notice that it uses **Standard_LRS** for the storage account, names resources with a **dev** prefix, and sets the `Environment` tag to **Dev**.

Again, create a new file with the following content. Save the file with the name *azuredeploy.parameters.prod.json*.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storagePrefix": {
      "value": "contosodata"
    },
    "storageSKU": {
      "value": "Standard_GRS"
    },
    "appServicePlanName": {
      "value": "contosoplan"
    },
    "webAppName": {
      "value": "contosowebapp"
    },
    "resourceTags": {
      "value": {
        "Environment": "Production",
        "Project": "Tutorial"
      }
    }
  }
}
```

This file is your parameter file for the production environment. Notice that it uses **Standard_GRS** for the storage account, names resources with a **contoso** prefix, and sets the `Environment` tag to **Production**. In a real production environment, you would also want to use an app service with a SKU other than free, but we use that SKU for this tutorial.

Deploy template

Use either Azure CLI or Azure PowerShell to deploy the template.

As a final test of your template, let's create two new resource groups. One for the dev environment and one for the production environment.

For the template and parameter variables, replace `{path-to-the-template-file}`, `{path-to-azuredeploy.parameters.dev.json}`, `{path-to-azuredeploy.parameters.prod.json}`, and the curly braces `{}` with your template and parameter file paths.

First, let's deploy to the dev environment.

PowerShell

Azure PowerShell

```
$templateFile = "{path-to-the-template-file}"
$parameterFile="{path-to-azuredeploy.parameters.dev.json}"
New-AzResourceGroup `
  -Name myResourceGroupDev `
  -Location "East US"
New-AzResourceGroupDeployment `
  -Name devenvironment `
  -ResourceGroupName myResourceGroupDev `
  -TemplateFile $templateFile `
  -TemplateParameterFile $parameterFile
```

Now, we deploy to the production environment.

PowerShell

Azure PowerShell

```
$parameterFile="{path-to-azuredeploy.parameters.prod.json}"
New-AzResourceGroup `
  -Name myResourceGroupProd `
  -Location "West US"
New-AzResourceGroupDeployment `
  -Name prodenvironment `
  -ResourceGroupName myResourceGroupProd `
  -TemplateFile $templateFile `
  -TemplateParameterFile $parameterFile
```

Note

If the deployment fails, use the `verbose` switch to get information about the resources you're creating. Use the `debug` switch to get more information for debugging.

Verify deployment

You can verify the deployment by exploring the resource groups from the Azure portal.

1. Sign in to the Azure portal.
2. From the left menu, select **Resource groups**.
3. You see the two new resource groups you deploy in this tutorial.
4. Select either resource group and view the deployed resources. Notice that they match the values you specified in your parameter file for that environment.

Clean up resources

1. From the Azure portal, select **Resource groups** from the left menu.

2. Select the hyperlinked resource group name next to the check box. If you complete this series, you have three resource groups to delete - **myResourceGroup**, **myResourceGroupDev**, and **myResourceGroupProd**.

3. Select the **Delete resource group** icon from the top menu.

   > **Caution**
   >
   > Deleting a resource group is irreversible.

4. Type the resource group name in the pop-up window that displays and select **Delete**.

Next steps

Congratulations. You've finished this introduction to deploying templates to Azure. Let us know if you have any comments and suggestions in the feedback section.

The next tutorial series goes into more detail about deploying templates.

Deploy a local template

Tutorial: Deploy a local ARM template

Article • 03/09/2023

Learn how to deploy an Azure Resource Manager template (ARM template) from your local machine. It takes about **8 minutes** to complete.

This tutorial is the first of a series. As you progress through the series, you modularize the template by creating a linked template, you store the linked template in a storage account, and secure the linked template by using SAS token, and you learn how to create a DevOps pipeline to deploy templates. This series focuses on template deployment. If you want to learn template development, see the beginner tutorials.

Get tools

Let's start by making sure you have the tools you need to deploy templates.

Command-line deployment

You need either Azure PowerShell or Azure CLI to deploy the template. For the installation instructions, see:

- Install Azure PowerShell
- Install Azure CLI on Windows
- Install Azure CLI on Linux
- Install Azure CLI on macOS

After installing either Azure PowerShell or Azure CLI, make sure you sign in for the first time. For help, see Sign in - PowerShell or Sign in - Azure CLI.

Editor (Optional)

Templates are JSON files. To review/edit templates, you need a good JSON editor. We recommend Visual Studio Code with the Resource Manager Tools extension. If you need to install these tools, see Quickstart: Create ARM templates with Visual Studio Code.

Review template

The template deploys a storage account, app service plan, and web app. If you're interested in creating the template, you can go through tutorial about Quickstart templates. However it's not required for completing this tutorial.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "projectName": {
      "type": "string",
      "minLength": 3,
      "maxLength": 11,
      "metadata": {
        "description": "Specify a project name that is used to generate resource names."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Specify a location for the resources."
      }
    },
    "storageSKU": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_RAGRS",
        "Standard_ZRS",
        "Premium_LRS",
        "Premium_ZRS",
        "Standard_GZRS",
        "Standard_RAGZRS"
      ],
      "metadata": {
        "description": "Specify the storage account type."
      }
    },
    "linuxFxVersion": {
      "type": "string",
      "defaultValue": "php|7.0",
      "metadata": {
        "description": "Specify the Runtime stack of current web app"
      }
    }
  },
  "variables": {
    "storageAccountName": "[concat(parameters('projectName'), uniqueString(resourceGroup().id))]",
    "webAppName": "[concat(parameters('projectName'), 'WebApp')]",
    "appServicePlanName": "[concat(parameters('projectName'), 'Plan')]"
  },
```

```
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-04-01",
      "name": "[variables('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('storageSKU')]"
      },
      "kind": "StorageV2",
      "properties": {
        "supportsHttpsTrafficOnly": true
      }
    },
    {
      "type": "Microsoft.Web/serverfarms",
      "apiVersion": "2020-12-01",
      "name": "[variables('appServicePlanName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "B1",
        "tier": "Basic",
        "size": "B1",
        "family": "B",
        "capacity": 1
      },
      "kind": "linux",
      "properties": {
        "perSiteScaling": false,
        "reserved": true,
        "targetWorkerCount": 0,
        "targetWorkerSizeId": 0
      }
    },
    {
      "type": "Microsoft.Web/sites",
      "apiVersion": "2020-12-01",
      "name": "[variables('webAppName')]",
      "location": "[parameters('location')]",
      "dependsOn": [
        "[resourceId('Microsoft.Web/serverfarms',
variables('appServicePlanName'))]"
      ],
      "kind": "app",
      "properties": {
        "serverFarmId": "[resourceId('Microsoft.Web/serverfarms',
variables('appServicePlanName'))]",
        "siteConfig": {
          "linuxFxVersion": "[parameters('linuxFxVersion')]"
        }
      }
    }
  ],
  "outputs": {
    "storageEndpoint": {
```

```
      "type": "object",
      "value": "
[reference(variables('storageAccountName')).primaryEndpoints]"
    }
  }
}
```

> **Important**
>
> Storage account names must be unique, between 3 and 24 characters in length, and use **numbers** and **lowercase** letters only. The sample template's `storageAccountName` variable combines the `projectName` parameter's maximum of 11 characters with a **uniqueString** value of 13 characters.

Save a copy of the template to your local computer with the *.json* extension, for example, *azuredeploy.json*. You deploy this template later in the tutorial.

Sign in to Azure

To start working with Azure PowerShell/Azure CLI to deploy a template, sign in with your Azure credentials.

PowerShell

Azure PowerShell

```
Connect-AzAccount
```

If you have multiple Azure subscriptions, select the subscription you want to use. Replace `[SubscriptionID/SubscriptionName]` and the square brackets `[]` with your subscription information:

PowerShell

Azure PowerShell

```
Set-AzContext [SubscriptionID/SubscriptionName]
```

Create resource group

When you deploy a template, you specify a resource group that will contain the resources. Before running the deployment command, create the resource group with either Azure CLI or Azure PowerShell. Select the tabs in the following code section to choose between Azure PowerShell and Azure CLI. The CLI examples in this article are written for the Bash shell.

PowerShell

Azure PowerShell

```
$projectName = Read-Host -Prompt "Enter a project name that is used to generate resource and resource group names"
$resourceGroupName = "${projectName}rg"

New-AzResourceGroup `
  -Name $resourceGroupName `
  -Location "Central US"
```

Deploy template

Use one or both deployment options to deploy the template.

PowerShell

Azure PowerShell

```
$projectName = Read-Host -Prompt "Enter the same project name"
$templateFile = Read-Host -Prompt "Enter the template file path and file name"
$resourceGroupName = "${projectName}rg"

New-AzResourceGroupDeployment `
  -Name DeployLocalTemplate `
  -ResourceGroupName $resourceGroupName `
  -TemplateFile $templateFile `
  -projectName $projectName `
  -verbose
```

To learn more about deploying template by using Azure PowerShell, see Deploy resources with ARM templates and Azure PowerShell.

Clean up resources

Clean up the resources you deployed by deleting the resource group.

1. From the Azure portal, select **Resource group** from the left menu.
2. Enter the resource group name in the **Filter by name** field.
3. Select the resource group name.
4. Select **Delete resource group** from the top menu.

Next steps

You learned how to deploy a local template. In the next tutorial, you separate the template into a main template and a linked template, and learn how to store and secure the linked template.

Deploy a linked template

Tutorial: Deploy a linked template

Article • 09/17/2021

In the previous tutorials, you learned how to deploy a template that is stored in your local computer. To deploy complex solutions, you can break a template into many templates, and deploy these templates through a main template. In this tutorial, you learn how to deploy a main template that contains the reference to a linked template. When the main template gets deployed, it triggers the deployment of the linked template. You also learn how to store and secure the templates by using SAS token. It takes about **12 minutes** to complete.

Prerequisites

We recommend that you complete the previous tutorial, but it's not required.

Review template

In the previous tutorials, you deploy a template that creates a storage account, App Service plan, and web app. The template used was:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "projectName": {
      "type": "string",
      "minLength": 3,
      "maxLength": 11,
      "metadata": {
        "description": "Specify a project name that is used to generate resource names."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Specify a location for the resources."
      }
    },
    "storageSKU": {
      "type": "string",
```

```
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_RAGRS",
        "Standard_ZRS",
        "Premium_LRS",
        "Premium_ZRS",
        "Standard_GZRS",
        "Standard_RAGZRS"
      ],
      "metadata": {
        "description": "Specify the storage account type."
      }
    },
    "linuxFxVersion": {
      "type": "string",
      "defaultValue": "php|7.0",
      "metadata": {
        "description": "Specify the Runtime stack of current web app"
      }
    }
  },
  "variables": {
    "storageAccountName": "[concat(parameters('projectName'),
uniqueString(resourceGroup().id))]",
    "webAppName": "[concat(parameters('projectName'), 'WebApp')]",
    "appServicePlanName": "[concat(parameters('projectName'), 'Plan')]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-04-01",
      "name": "[variables('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('storageSKU')]"
      },
      "kind": "StorageV2",
      "properties": {
        "supportsHttpsTrafficOnly": true
      }
    },
    {
      "type": "Microsoft.Web/serverfarms",
      "apiVersion": "2020-12-01",
      "name": "[variables('appServicePlanName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "B1",
        "tier": "Basic",
        "size": "B1",
        "family": "B",
        "capacity": 1
      },
```

```
      "kind": "linux",
      "properties": {
        "perSiteScaling": false,
        "reserved": true,
        "targetWorkerCount": 0,
        "targetWorkerSizeId": 0
      }
    },
    {
      "type": "Microsoft.Web/sites",
      "apiVersion": "2020-12-01",
      "name": "[variables('webAppName')]",
      "location": "[parameters('location')]",
      "dependsOn": [
        "[resourceId('Microsoft.Web/serverfarms',
variables('appServicePlanName'))]"
      ],
      "kind": "app",
      "properties": {
        "serverFarmId": "[resourceId('Microsoft.Web/serverfarms',
variables('appServicePlanName'))]",
        "siteConfig": {
          "linuxFxVersion": "[parameters('linuxFxVersion')]"
        }
      }
    }
  ],
  "outputs": {
    "storageEndpoint": {
      "type": "object",
      "value": "
[reference(variables('storageAccountName')).primaryEndpoints]"
    }
  }
}
```

Create a linked template

You can separate the storage account resource into a linked template:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageAccountName": {
      "type": "string",
      "metadata": {
        "description": "Specify the storage account name."
      }
```

```
    },
    "location": {
      "type": "string",
      "metadata": {
        "description": "Specify a location for the resources."
      }
    },
    "storageSKU": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_RAGRS",
        "Standard_ZRS",
        "Premium_LRS",
        "Premium_ZRS",
        "Standard_GZRS",
        "Standard_RAGZRS"
      ],
      "metadata": {
        "description": "Specify the storage account type."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-04-01",
      "name": "[parameters('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('storageSKU')]"
      },
      "kind": "StorageV2",
      "properties": {
        "supportsHttpsTrafficOnly": true
      }
    }
  ],
  "outputs": {
    "storageEndpoint": {
      "type": "object",
      "value": "
[reference(parameters('storageAccountName')).primaryEndpoints]"
    }
  }
}
```

The following template is the main template. The highlighted `Microsoft.Resources/deployments` object shows how to call a linked template. The linked template cannot be stored as a local file or a file that is only available on your local

network. You can either provide a URI value of the linked template that includes either HTTP or HTTPS, or use the *relativePath* property to deploy a remote linked template at a location relative to the parent template. One option is to place both the main template and the linked template in a storage account.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "projectName": {
      "type": "string",
      "minLength": 3,
      "maxLength": 11,
      "metadata": {
        "description": "Specify a project name that is used to generate
resource names."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Specify a location for the resources."
      }
    },
    "linuxFxVersion": {
      "type": "string",
      "defaultValue": "php|7.0",
      "metadata": {
        "description": "The Runtime stack of current web app"
      }
    }
  },
  "variables": {
    "storageAccountName": "[concat(parameters('projectName'),
uniqueString(resourceGroup().id))]",
    "webAppName": "[concat(parameters('projectName'), 'WebApp')]",
    "appServicePlanName": "[concat(parameters('projectName'), 'Plan')]"
  },
  "resources": [
```

```
    {
      "name": "linkedTemplate",
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "properties": {
        "mode": "Incremental",
        "templateLink": {
          "relativePath": "linkedStorageAccount.json"
        },
        "parameters": {
          "storageAccountName": {
            "value": "[variables('storageAccountName')]"
          },
          "location": {
            "value": "[parameters('location')]"
          }
        }
      }
    },
    {
      "type": "Microsoft.Web/serverfarms",
      "apiVersion": "2020-12-01",
      "name": "[variables('appServicePlanName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "B1",
        "tier": "Basic",
        "size": "B1",
        "family": "B",
        "capacity": 1
      },
      "kind": "linux",
      "properties": {
        "perSiteScaling": false,
        "reserved": true,
        "targetWorkerCount": 0,
        "targetWorkerSizeId": 0
      }
    },
    {
      "type": "Microsoft.Web/sites",
      "apiVersion": "2020-12-01",
      "name": "[variables('webAppName')]",
      "location": "[parameters('location')]",
      "dependsOn": [
        "[resourceId('Microsoft.Web/serverfarms',
variables('appServicePlanName'))]"
      ],
      "kind": "app",
      "properties": {
        "serverFarmId": "[resourceId('Microsoft.Web/serverfarms',
variables('appServicePlanName'))]",
        "siteConfig": {
          "linuxFxVersion": "[parameters('linuxFxVersion')]"
        }
```

```
      }
    }
  ],
  "outputs": {
    "storageEndpoint": {
      "type": "object",
      "value": "[reference('linkedTemplate').outputs.storageEndpoint.value]"
    }
  }
}
```

Store the linked template

Both of the main template and the linked template are stored in GitHub:

The following PowerShell script creates a storage account, creates a container, and copies the two templates from a GitHub repository to the container. These two templates are:

- The main template: https://raw.githubusercontent.com/Azure/azure-docs-json-samples/master/get-started-deployment/linked-template/azuredeploy.json
- The linked template: https://raw.githubusercontent.com/Azure/azure-docs-json-samples/master/get-started-deployment/linked-template/linkedStorageAccount.json

Select **Try-it** to open the Cloud Shell, select **Copy** to copy the PowerShell script, and right-click the shell pane to paste the script:

> **Important**
>
> Storage account names must be unique, between 3 and 24 characters in length, and use **numbers** and **lowercase** letters only. The sample template's `storageAccountName` variable combines the `projectName` parameter's maximum of 11 characters with a **uniqueString** value of 13 characters.

Azure PowerShell

```
$projectName = Read-Host -Prompt "Enter a project name:"   # This name is
used to generate names for Azure resources, such as storage account name.
$location = Read-Host -Prompt "Enter a location (i.e. centralus)"

$resourceGroupName = $projectName + "rg"
$storageAccountName = $projectName + "store"
$containerName = "templates" # The name of the Blob container to be created.
```

```
$mainTemplateURL = "https://raw.githubusercontent.com/Azure/azure-docs-json-samples/master/get-started-deployment/linked-template/azuredeploy.json"
$linkedTemplateURL = "https://raw.githubusercontent.com/Azure/azure-docs-json-samples/master/get-started-deployment/linked-template/linkedStorageAccount.json"

$mainFileName = "azuredeploy.json" # A file name used for downloading and uploading the main template.Add-PSSnapin
$linkedFileName = "linkedStorageAccount.json" # A file name used for downloading and uploading the linked template.

# Download the templates
Invoke-WebRequest -Uri $mainTemplateURL -OutFile "$home/$mainFileName"
Invoke-WebRequest -Uri $linkedTemplateURL -OutFile "$home/$linkedFileName"

# Create a resource group
New-AzResourceGroup -Name $resourceGroupName -Location $location

# Create a storage account
$storageAccount = New-AzStorageAccount `
    -ResourceGroupName $resourceGroupName `
    -Name $storageAccountName `
    -Location $location `
    -SkuName "Standard_LRS"

$context = $storageAccount.Context

# Create a container
New-AzStorageContainer -Name $containerName -Context $context -Permission Container

# Upload the templates
Set-AzStorageBlobContent `
    -Container $containerName `
    -File "$home/$mainFileName" `
    -Blob $mainFileName `
    -Context $context

Set-AzStorageBlobContent `
    -Container $containerName `
    -File "$home/$linkedFileName" `
    -Blob $linkedFileName `
    -Context $context

Write-Host "Press [ENTER] to continue ..."
```

Deploy template

To deploy templates in a storage account, generate a SAS token and supply it to the *-QueryString* parameter. Set the expiry time to allow enough time to complete the deployment. The blobs containing the templates are accessible to only the account

owner. However, when you create a SAS token for a blob, the blob is accessible to anyone with that SAS token. If another user intercepts the URI and the SAS token, that user is able to access the template. A SAS token is a good way of limiting access to your templates, but you should not include sensitive data like passwords directly in the template.

If you haven't created the resource group, see Create resource group.

> **Note**
>
> In the below Azure CLI code, `date` parameter `-d` is an invalid argument in macOS. So macOS users, to add 2 hours to current time in terminal on macOS you should use `-v+2H`.

PowerShell

Azure PowerShell

```
$projectName = Read-Host -Prompt "Enter the same project name:"   # This name is used to generate names for Azure resources, such as storage account name.

$resourceGroupName="${projectName}rg"
$storageAccountName="${projectName}store"
$containerName = "templates"

$key = (Get-AzStorageAccountKey -ResourceGroupName $resourceGroupName -Name $storageAccountName).Value[0]
$context = New-AzStorageContext -StorageAccountName $storageAccountName -StorageAccountKey $key

$mainTemplateUri = $context.BlobEndPoint + "$containerName/azuredeploy.json"
$sasToken = New-AzStorageContainerSASToken `
    -Context $context `
    -Container $containerName `
    -Permission r `
    -ExpiryTime (Get-Date).AddHours(2.0)
$newSas = $sasToken.substring(1)


New-AzResourceGroupDeployment `
  -Name DeployLinkedTemplate `
  -ResourceGroupName $resourceGroupName `
  -TemplateUri $mainTemplateUri `
  -QueryString $newSas `
  -projectName $projectName `
```

```
  -verbose

Write-Host "Press [ENTER] to continue ..."
```

Clean up resources

Clean up the resources you deployed by deleting the resource group.

1. From the Azure portal, select **Resource group** from the left menu.
2. Enter the resource group name in the **Filter by name** field.
3. Select the resource group name.
4. Select **Delete resource group** from the top menu.

Next steps

You learned how to deploy a linked template. In the next tutorial, you learn how to create a DevOps pipeline to deploy a template.

Create a pipeline

Tutorial: Continuous integration of ARM templates with Azure Pipelines

Article • 04/09/2023

In the previous tutorial, you deploy a linked template. In this tutorial, you learn how to use Azure Pipelines to continuously build and deploy Azure Resource Manager template (ARM template) projects.

Azure DevOps provides developer services to support teams to plan work, collaborate on code development, and build and deploy applications. Developers can work in the cloud using Azure DevOps Services. Azure DevOps provides an integrated set of features that you can access through your web browser or IDE client. Azure Pipelines is one of these features. Azure Pipelines is a fully featured continuous integration (CI) and continuous delivery (CD) service. It works with your preferred Git provider and can deploy to most major cloud services. Then you can automate the build, testing, and deployment of your code to Microsoft Azure, Google Cloud Platform, or Amazon Web Services.

> **Note**
>
> Pick a project name. When you go through the tutorial, replace any of the **AzureRmPipeline** with your project name. This project name is used to generate resource names. One of the resources is a storage account. Storage account names must be between 3 and 24 characters in length and use numbers and lower-case letters only. The name must be unique. In the template, the storage account name is the project name with **store** appended, and the project name must be between 3 and 11 characters. So the project name must meet the storage account name requirements and has less than 11 characters.

This tutorial covers the following tasks:

- ✓ Prepare a GitHub repository
- ✓ Create an Azure DevOps project
- ✓ Create an Azure pipeline
- ✓ Verify the pipeline deployment
- ✓ Update the template and redeploy
- ✓ Clean up resources

If you don't have an Azure subscription, create a free account before you begin.

Prerequisites

To complete this article, you need:

- **A GitHub account**, where you use it to create a repository for your templates. If you don't have one, you can create one for free. For more information about using GitHub repositories, see Build GitHub repositories.
- **Install Git**. This tutorial instruction uses *Git Bash* or *Git Shell*. For instructions, see Install Git.
- **An Azure DevOps organization**. If you don't have one, you can create one for free. See Create an organization or project collection.
- (optional) **Visual Studio Code with Resource Manager Tools extension**. See Quickstart: Create ARM templates with Visual Studio Code.

Prepare a GitHub repository

GitHub is used to store your project source code including Resource Manager templates. For other supported repositories, see repositories supported by Azure DevOps.

Create a GitHub repository

If you don't have a GitHub account, see Prerequisites.

1. Sign in to GitHub.

2. Select your account image on the upper right corner, and then select **Your repositories**.

3. Select **New**, a green button.

4. In **Repository name**, enter a repository name. For example, **AzureRmPipeline-repo**. Remember to replace any of **AzureRmPipeline** with your project name. You can select either **Public** or **private** for going through this tutorial. And then select **Create repository**.

5. Write down the URL. The repository URL is the following format - `https://github.com/[YourAccountName]/[YourRepositoryName]`.

This repository is referred to as a *remote repository*. Each of the developers of the same project can clone their own *local repository*, and merge the changes to the remote repository.

Clone the remote repository

1. Open Git Shell or Git Bash. See Prerequisites.

2. Verify your current folder is **GitHub**.

3. Run the following command:

   ```bash
   git clone
   https://github.com/[YourAccountName]/[YourGitHubRepositoryName]
   cd [YourGitHubRepositoryName]
   mkdir create_web_app
   cd create_web_app
   pwd
   ```

   Replace `[YourAccountName]` with your GitHub account name, and replace `[YourGitHubRepositoryName]` with your repository name you created in the previous procedure.

The *create_web_app* folder is the folder where the template is stored. The `pwd` command shows the folder path. The path is where you save the template to in the following procedure.

Download a Quickstart template

Instead of creating the templates, you can download the templates and save them to the *create_web_app* folder.

- The main template: https://raw.githubusercontent.com/Azure/azure-docs-json-samples/master/get-started-deployment/linked-template/azuredeploy.json
- The linked template: https://raw.githubusercontent.com/Azure/azure-docs-json-samples/master/get-started-deployment/linked-template/linkedStorageAccount.json

Both the folder name and the file names are used as they are in the pipeline. If you change these names, you must update the names used in the pipeline.

Push the template to the remote repository

The *azuredeploy.json* has been added to the local repository. Next, you upload the template to the remote repository.

1. Open *Git Shell* or *Git Bash*, if it is not opened.

2. Change directory to the *create_web_app* folder in your local repository.

3. Verify the *azuredeploy.json* file is in the folder.

4. Run the following command:

   ```bash
   git add .
   git commit -m "Add web app templates."
   git push origin main
   ```

   You might get a warning about LF. You can ignore the warning. **main** is the main branch. You typically create a branch for each update. To simplify the tutorial, you use the main branch directly.

5. Browse to your GitHub repository from a browser. The URL is `https://github.com/[YourAccountName]/[YourGitHubRepository]`. You shall see the *create_web_app* folder and the two files inside the folder.

6. Select *azuredeploy.json* to open the template.

7. Select the **Raw** button. The URL begins with `https://raw.githubusercontent.com`.

8. Make a copy of the URL. You need to provide this value when you configure the pipeline later in the tutorial.

So far, you have created a GitHub repository, and uploaded the templates to the repository.

Create a DevOps project

A DevOps organization is needed before you can proceed to the next procedure. If you don't have one, see Prerequisites.

1. Sign in to Azure DevOps.

2. Select a DevOps organization from the left.

3. Select **New project**. If you don't have any projects, the create project page is opened automatically.

4. Enter the following values:

   - **Project name**: enter a project name. You can use the project name you picked at the very beginning of the tutorial.
   - **Version control**: Select **Git**. You might need to expand **Advanced** to see **Version control**.

   Use the default value for the other properties.

5. Select **Create**.

Create a service connection that is used to deploy projects to Azure.

1. Select **Project settings** from the bottom of the left menu.

2. Select **Service connections** under **Pipelines**.

3. Select **Create Service connection**, select **Azure Resource Manager**, and then select **Next**.

4. Select **Service principal**, and then select **Next**.

5. Enter the following values:

    - **Scope level**: select **Subscription**.
    - **Subscription**: select your subscription.
    - **Resource Group**: Leave it blank.
    - **Connection name**: enter a connection name. For example, **AzureRmPipeline-conn**. Write down this name, you need the name when you create your pipeline.
    - **Grant access permission to all pipelines**. (selected)

6. Select **Save**.

Create a pipeline

Until now, you have completed the following tasks. If you skip the previous sections because you are familiar with GitHub and DevOps, you must complete the tasks before you continue.

- Create a GitHub repository, and save the templates to the *create_web_app* folder in the repository.
- Create a DevOps project, and create an Azure Resource Manager service connection.

To create a pipeline with a step to deploy a template:

1. Select **Pipelines** from the left menu.

2. Select **Create pipeline**.

3. From the **Connect** tab, select **GitHub**. If asked, enter your GitHub credentials, and then follow the instructions. If you see the following screen, select **Only select repositories**, and verify your repository is in the list before you select **Approve & Install**.

4. From the **Select** tab, select your repository. The default name is `[YourAccountName]/[YourGitHubRepositoryName]`.

5. From the **Configure** tab, select **Starter pipeline**. It shows the *azure-pipelines.yml* pipeline file with two script steps.

6. Delete the two script steps from the *.yml* file.

7. Move the cursor to the line after **steps:**.

8. Select **Show assistant** on the right of the screen to open **Tasks** pane.

9. Select **ARM template deployment**.

10. Enter the following values:

- **deploymentScope**: Select **Resource Group**. To learn more about the scopes, see Deployment scopes.
- **Azure Resource Manager connection**: Select the service connection name that you created earlier.
- **Subscription**: Specify the target subscription ID.
- **Action**: Select the **Create Or Update Resource Group** action does 2 actions - 1. create a resource group if a new resource group name is provided; 2. deploy the template specified.
- **Resource group**: Enter a new resource group name. For example, **AzureRmPipeline-rg**.
- **Location**: Select a location for the resource group, for example, **Central US**.
- **Template location**: Select **URL of the file**, which means the task looks for the template file by using the URL. Because *relativePath* is used in the main template and *relativePath* is only supported on URI-based deployments, you must use URL here.
- **Template link**: Enter the URL that you got at the end of the Prepare a GitHub repository section. It starts with `https://raw.githubusercontent.com`.
- **Template parameters link**: Leave this field blank. You will specify the parameter values in the **Override template parameters**.
- **Override template parameters**: Enter `-projectName [EnterAProjectName]`.
- **Deployment mode**: Select **Incremental**.
- **Deployment name**: Enter **DeployPipelineTemplate**. Select **Advanced** before you can see **Deployment name**.

11. Select **Add**.

    For more information about the task, see Azure Resource Group Deployment task, and Azure Resource Manager template deployment task

    The *.yml* file shall be similar to:

```
master    mumian/AzureRMPipeline-repo / azure-pipelines.yml *

# Starter pipeline
# Start with a minimal pipeline that you can customize to build and deploy your code.
# Add steps that build, run tests, deploy, and more:
# https://aka.ms/yaml

trigger:
- master

pool:
  vmImage: 'ubuntu-latest'

steps:
Settings
- task: AzureResourceManagerTemplateDeployment@3
  inputs:
    deploymentScope: 'Resource Group'
    azureResourceManagerConnection: 'AzureRmPipeline-conn'
    subscriptionId:
    action: 'Create Or Update Resource Group'
    resourceGroupName: 'AzureRmPipeline-rg'
    location: 'Central US'
    templateLocation: 'URL of the file'
    csmFileLink: 'https://raw.githubusercontent.com/mumian/AzureRMPipeline-repo/master/CreateWebApp/azuredeploy.json'
    overrideParameters: '-projectName jgao0214'
    deploymentMode: 'Incremental'
    deploymentName: 'DeployPipelineTemplate'
```

12. Select **Save and run**.

13. From the **Save and run** pane, select **Save and run** again. A copy of the YAML file is saved into the connected repository. You can see the YAML file by browse to your repository.

14. Verify that the pipeline is executed successfully.

Verify the deployment

1. Sign in to the Azure portal.
2. Open the resource group. The name is what you specified in the pipeline YAML file. You shall see one storage account created. The storage account name starts with **store**.
3. Select the storage account name to open it.
4. Select **Properties**. Notice the **Replication** is **Locally-redundant storage (LRS)**.

Update and redeploy

When you update the template and push the changes to the remote repository, the pipeline automatically updates the resources, the storage account in this case.

1. Open *linkedStorageAccount.json* from your local repository in Visual Studio Code or any text editor.

2. Update the **defaultValue** of **storageAccountType** to **Standard_GRS**. See the following screenshot:

```
1  {
2    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
3    "contentVersion": "1.0.0.0",
4    "parameters": {
5      "storageAccountName": {
6        "type": "string",
7        "metadata": {
8          "description": "Specify the storage account name."
9        }
10     },
11     "location": {
12       "type": "string",
13       "metadata": {
14         "description": "Specify a location for the resources."
15       }
16     },
17     "storageSKU": {
18       "type": "string",
19       "defaultValue": "Standard_GRS",
20       "allowedValues": [
21         "Standard_LRS",
22         "Standard_GRS",
23         "Standard_RAGRS",
24         "Standard_ZRS",
25         "Premium_LRS",
26         "Premium_ZRS",
27         "Standard_GZRS",
28         "Standard_RAGZRS"
29       ],
30       "metadata": {
31         "description": "Specify the storage account type."
32       }
33     }
34   },
35   "resources": [
```

3. Save the changes.

4. Push the changes to the remote repository by running the following commands from Git Bash/Shell.

Bash

```bash
git pull origin main
git add .
git commit -m "Update the storage account type."
git push origin main
```

The first command (`pull`) syncs the local repository with the remote repository. The pipeline YAML file was only added to the remote repository. Running the `pull` command downloads a copy of the YAML file to the local branch.

The fourth command (`push`) uploads the revised *linkedStorageAccount.json* file to the remote repository. With the main branch of the remote repository updated, the pipeline is fired again.

To verify the changes, you can check the Replication property of the storage account. See Verify the deployment.

Clean up resources

When the Azure resources are no longer needed, clean up the resources you deployed by deleting the resource group.

1. From the Azure portal, select **Resource group** from the left menu.
2. Enter the resource group name in the **Filter by name** field.
3. Select the resource group name.
4. Select **Delete resource group** from the top menu.

You might also want to delete the GitHub repository and the Azure DevOps project.

Next steps

Congratulations, you've finished this Resource Manager template deployment tutorial. Let us know if you have any comments and suggestions in the feedback section. Thanks! You're ready to jump into more advanced concepts about templates. The next tutorial goes into more detail about using template reference documentation to help with defining resources to deploy.

Utilize template reference

Tutorial: Create a template spec with linked templates

Article • 05/11/2023

Learn how to create a template spec with a main template and a linked template. You use template specs to share ARM templates with other users in your organization. This article shows you how to create a template spec to package a main template and its linked templates using the `relativePath` property of the deployment resource.

Prerequisites

An Azure account with an active subscription. Create an account for free.

> **Note**
>
> To use template specs with Azure PowerShell, you must install **version 5.0.0 or later**. To use it with Azure CLI, use **version 2.14.2 or later**.

Create linked templates

Create the main template and the linked template.

To link a template, add a deployments resource to your main template. In the `templateLink` property, specify the relative path of the linked template in accordance with the path of the parent template.

The linked template is called **linkedTemplate.json**, and is stored in a subfolder called **artifacts** in the path where the main template is stored. You can use one of the following values for the relativePath:

- `./artifacts/linkedTemplate.json`
- `/artifacts/linkedTemplate.json`
- `artifacts/linkedTemplate.json`

The `relativePath` property is always relative to the template file where `relativePath` is declared, so if there is another linkedTemplate2.json that is called from linkedTemplate.json and linkedTemplate2.json is stored in the same artifacts subfolder, the relativePath specified in linkedTemplate.json is just `linkedTemplate2.json`.

1. Create the main template with the following JSON. Save the main template as **azuredeploy.json** to your local computer. This tutorial assumes you've saved to a path **c:\Templates\linkedTS\azuredeploy.json** but you can use any path.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "location": {
      "type": "string",
      "defaultValue": "westus2",
      "metadata":{
        "description": "Specify the location for the resources."
      }
    },
    "storageAccountType": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "metadata":{
        "description": "Specify the storage account type."
      }
    }
  },
  "variables": {
    "appServicePlanName": "[concat('plan', uniquestring(resourceGroup().id))]"
  },
  "resources": [
    {
      "type": "Microsoft.Web/serverfarms",
      "apiVersion": "2016-09-01",
      "name": "[variables('appServicePlanName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "B1",
        "tier": "Basic",
        "size": "B1",
        "family": "B",
        "capacity": 1
      },
      "kind": "linux",
      "properties": {
        "perSiteScaling": false,
        "reserved": true,
        "targetWorkerCount": 0,
        "targetWorkerSizeId": 0
      }
    },
    {
      "type": "Microsoft.Resources/deployments",
```

```
      "apiVersion": "2020-10-01",
      "name": "createStorage",
      "properties": {
        "mode": "Incremental",
        "templateLink": {
          "relativePath": "artifacts/linkedTemplate.json"
        },
        "parameters": {
          "storageAccountType": {
            "value": "[parameters('storageAccountType')]"
          }
        }
      }
    }
  ]
}
```

> **Note**
>
> The apiVersion of `Microsoft.Resources/deployments` must be 2020-06-01 or later.

2. Create a directory called **artifacts** in the folder where the main template is saved.

3. Create the linked template with the following JSON:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageAccountType": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_ZRS",
        "Premium_LRS"
      ],
      "metadata": {
        "description": "Storage Account type"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
```

```
          "description": "Location for all resources."
        }
      }
    },
    "variables": {
      "storageAccountName": "[concat('store',
  uniquestring(resourceGroup().id))]"
    },
    "resources": [
      {
        "type": "Microsoft.Storage/storageAccounts",
        "apiVersion": "2019-04-01",
        "name": "[variables('storageAccountName')]",
        "location": "[parameters('location')]",
        "sku": {
          "name": "[parameters('storageAccountType')]"
        },
        "kind": "StorageV2",
        "properties": {}
      }
    ],
    "outputs": {
      "storageAccountName": {
        "type": "string",
        "value": "[variables('storageAccountName')]"
      }
    }
  }
```

4. Save the template as **linkedTemplate.json** in the **artifacts** folder.

Create template spec

Templates specs are stored in resource Groups. Create a resource group, and then create a template spec with the following script. The template spec name is **webSpec**.

PowerShell

Azure PowerShell

```
New-AzResourceGroup `
  -Name templateSpecRG `
  -Location westus2

New-AzTemplateSpec `
  -Name webSpec `
  -Version "1.0.0.0" `
  -ResourceGroupName templateSpecRG `
  -Location westus2 `
  -TemplateFile "c:\Templates\linkedTS\azuredeploy.json"
```

When you are done, you can view the template spec from the Azure portal or by using the following cmdlet:

PowerShell

Azure PowerShell

```
Get-AzTemplateSpec -ResourceGroupName templatespecRG -Name webSpec
```

Deploy template spec

You can now deploy the template spec. Deploying the template spec is just like deploying the template it contains, except that you pass in the resource ID of the template spec. You use the same deployment commands, and if needed, pass in parameter values for the template spec.

PowerShell

Azure PowerShell

```
New-AzResourceGroup `
  -Name webRG `
  -Location westus2

$id = (Get-AzTemplateSpec -ResourceGroupName templateSpecRG -Name
webSpec -Version "1.0.0.0").Versions.Id

New-AzResourceGroupDeployment `
  -TemplateSpecId $id `
  -ResourceGroupName webRG
```

Next steps

To learn about deploying a template spec as a linked template, see Tutorial: Deploy a template spec as a linked template.

Tutorial: Deploy a template spec as a linked template

Article • 03/09/2023

Learn how to deploy an existing template spec by using a linked deployment. You use template specs to share ARM templates with other users in your organization. After you have created a template spec, you can deploy the template spec by using Azure PowerShell or Azure CLI. You can also deploy the template spec as a part of your solution by using a linked template.

Prerequisites

An Azure account with an active subscription. Create an account for free.

> **Note**
>
> To use template spec with Azure PowerShell, you must install **version 5.0.0 or later**. To use it with Azure CLI, use **version 2.14.2 or later**.

Create a template spec

Follow Quickstart: Create and deploy template spec to create a template spec for deploying a storage account. You need the resource group name of the template spec, template spec name, and template spec version in the next section.

Create the main template

To deploy a template spec in an ARM template, add a deployments resource to your main template. In the `templateLink` property, specify the resource ID of a template spec. Create a template with the following JSON called **azuredeploy.json**. This tutorial assumes you've saved to a path **c:\Templates\deployTS\azuredeploy.json** but you can use any path.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
```

```
  "parameters": {
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    },
    "tsResourceGroup":{
      "type": "string",
      "metadata": {
        "Description": "Specifies the resource group name of the template
spec."
      }
    },
    "tsName": {
      "type": "string",
      "metadata": {
        "Description": "Specifies the name of the template spec."
      }
    },
    "tsVersion": {
      "type": "string",
      "defaultValue": "1.0.0.0",
      "metadata": {
        "Description": "Specifies the version the template spec."
      }
    },
    "storageAccountType": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "metadata": {
        "Description": "Specifies the storage account type required by the
template spec."
      }
    }
  },
  "variables": {
    "appServicePlanName": "[concat('plan',
uniquestring(resourceGroup().id))]"
  },
  "resources": [
    {
      "type": "Microsoft.Web/serverfarms",
      "apiVersion": "2016-09-01",
      "name": "[variables('appServicePlanName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "B1",
        "tier": "Basic",
        "size": "B1",
        "family": "B",
        "capacity": 1
      },
      "kind": "linux",
      "properties": {
        "perSiteScaling": false,
        "reserved": true,
```

```
        "targetWorkerCount": 0,
        "targetWorkerSizeId": 0
      }
    },
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2020-10-01",
      "name": "createStorage",
      "properties": {
        "mode": "Incremental",
        "templateLink": {
          "id": "[resourceId(parameters('tsResourceGroup'),
'Microsoft.Resources/templateSpecs/versions', parameters('tsName'),
parameters('tsVersion'))]"
        },
        "parameters": {
          "storageAccountType": {
            "value": "[parameters('storageAccountType')]"
          }
        }
      }
    }
  ],
  "outputs": {
    "templateSpecId": {
      "type": "string",
      "value": "[resourceId(parameters('tsResourceGroup'),
'Microsoft.Resources/templateSpecs/versions', parameters('tsName'),
parameters('tsVersion'))]"
    }
  }
}
```

The template spec ID is generated by using the resourceID() function. The resource group argument in the resourceID() function is optional if the templateSpec is in the same resource group of the current deployment. You can also directly pass in resource ID as a parameter. To get the ID, use:

PowerShell

Azure PowerShell

```
$id = (Get-AzTemplateSpec -ResourceGroupName $resourceGroupName -Name
$templateSpecName -Version $templateSpecVersion).Versions.Id
```

The syntax for passing parameters to the template spec is:

JSON

```
"parameters": {
  "storageAccountType": {
    "value": "[parameters('storageAccountType')]"
  }
}
```

> **Note**
>
> The apiVersion of `Microsoft.Resources/deployments` must be 2020-06-01 or later.

Deploy the template

When you deploy the linked template, it deploys both the web application and the storage account. The deployment is same as deploying other ARM templates.

PowerShell

Azure PowerShell

```
New-AzResourceGroup `
  -Name webRG `
  -Location westus2

New-AzResourceGroupDeployment `
  -ResourceGroupName webRG `
  -TemplateFile "c:\Templates\deployTS\azuredeploy.json" `
  -tsResourceGroup templateSpecRg `
  -tsName storageSpec `
  -tsVersion 1.0
```

Next steps

To learn about creating a template spec that includes linked templates, see Create a template spec of a linked template.

Tutorial: Create Azure portal forms for a template spec

Article • 03/09/2023

You can create a form that appears in the Azure portal to assist users in deploying a template spec. The form allows users to enter values that are passed as parameters to the template spec.

When you create the template spec, you package the form and Azure Resource Manager template (ARM template) together. Deploying the template spec through the portal automatically launches the form.

The following screenshot shows a form opened in the Azure portal.

Prerequisites

An Azure account with an active subscription. Create an account for free.

For Azure PowerShell, use version 6.0.0 or later. For Azure CLI, use version 2.24.0 or later.

Create template

To show the different portal elements that are available in a form, you'll use an ARM template with several parameters. The following template creates a key vault, configures permissions to key vault for a user, and adds a secret.

Copy this file and save it locally. This tutorial assumes you've named it **keyvault.json** but you can give it any name.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "keyVaultName": {
      "type": "string",
      "metadata": {
        "description": "Specifies the name of the key vault."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Specifies the Azure location where the key vault should be created."
      }
    },
    "enabledForDeployment": {
      "type": "bool",
      "defaultValue": false,
      "allowedValues": [
        true,
        false
      ],
      "metadata": {
        "description": "Specifies whether Azure Virtual Machines are permitted to retrieve certificates stored as secrets from the key vault."
      }
    },
    "enabledForDiskEncryption": {
      "type": "bool",
      "defaultValue": false,
      "allowedValues": [
        true,
        false
      ],
      "metadata": {
        "description": "Specifies whether Azure Disk Encryption is permitted to retrieve secrets from the vault and unwrap keys."
```

```
      }
    },
    "enabledForTemplateDeployment": {
      "type": "bool",
      "defaultValue": false,
      "allowedValues": [
        true,
        false
      ],
      "metadata": {
        "description": "Specifies whether Azure Resource Manager is
permitted to retrieve secrets from the key vault."
      }
    },
    "tenantId": {
      "type": "string",
      "defaultValue": "[subscription().tenantId]",
      "metadata": {
        "description": "Specifies the Azure Active Directory tenant ID that
should be used for authenticating requests to the key vault. Get it by using
Get-AzSubscription cmdlet."
      }
    },
    "objectId": {
      "type": "string",
      "metadata": {
        "description": "Specifies the object ID of a user, service principal
or security group in the Azure Active Directory tenant for the vault. The
object ID must be unique for the list of access policies. Get it by using
Get-AzADUser or Get-AzADServicePrincipal cmdlets."
      }
    },
    "keysPermissions": {
      "type": "array",
      "defaultValue": [
        "list"
      ],
      "metadata": {
        "description": "Specifies the permissions to keys in the vault.
Valid values are: all, encrypt, decrypt, wrapKey, unwrapKey, sign, verify,
get, list, create, update, import, delete, backup, restore, recover, and
purge."
      }
    },
    "secretsPermissions": {
      "type": "array",
      "defaultValue": [
        "list"
      ],
      "metadata": {
        "description": "Specifies the permissions to secrets in the vault.
Valid values are: all, get, list, set, delete, backup, restore, recover, and
purge."
      }
    },
```

```
    "skuName": {
      "type": "string",
      "defaultValue": "standard",
      "allowedValues": [
        "standard",
        "premium"
      ],
      "metadata": {
        "description": "Specifies whether the key vault is a standard vault
or a premium vault."
      }
    },
    "secretName": {
      "type": "string",
      "metadata": {
        "description": "Specifies the name of the secret that you want to
create."
      }
    },
    "secretValue": {
      "type": "secureString",
      "metadata": {
        "description": "Specifies the value of the secret that you want to
create."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.KeyVault/vaults",
      "apiVersion": "2022-07-01",
      "name": "[parameters('keyVaultName')]",
      "location": "[parameters('location')]",
      "properties": {
        "enabledForDeployment": "[parameters('enabledForDeployment')]",
        "enabledForDiskEncryption": "
[parameters('enabledForDiskEncryption')]",
        "enabledForTemplateDeployment": "
[parameters('enabledForTemplateDeployment')]",
        "tenantId": "[parameters('tenantId')]",
        "accessPolicies": [
          {
            "objectId": "[parameters('objectId')]",
            "tenantId": "[parameters('tenantId')]",
            "permissions": {
              "keys": "[parameters('keysPermissions')]",
              "secrets": "[parameters('secretsPermissions')]"
            }
          }
        ],
        "sku": {
          "name": "[parameters('skuName')]",
          "family": "A"
        },
        "networkAcls": {
```

```
            "defaultAction": "Allow",
            "bypass": "AzureServices"
          }
        }
      },
      {
        "type": "Microsoft.KeyVault/vaults/secrets",
        "apiVersion": "2022-07-01",
        "name": "[format('{0}/{1}', parameters('keyVaultName'),
parameters('secretName'))]",
        "dependsOn": [
          "[resourceId('Microsoft.KeyVault/vaults',
parameters('keyVaultName'))]"
        ],
        "properties": {
          "value": "[parameters('secretValue')]"
        }
      }
    ]
}
```

Create default form

The Azure portal provides a sandbox for creating and previewing forms. This sandbox can render a form from an existing ARM template. You'll use this default form to get started with creating a form for your template spec. For more information about the form structure, see FormViewType.

1. Open the Form view sandbox.

2. In **Package Type**, select **CustomTemplate**. Make sure you select the package type before specify deployment template.

3. In **Deployment template (optional)**, select the key vault template you saved locally. When prompted if you want to overwrite current changes, select **Yes**. The autogenerated form is displayed in the code window. The form is editable from the portal. To customize the form, see customize form. If you look closely into the autogenerated form, the default title is called **Test Form View**, and there's only one step called **basics** defined.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2021-09-09/uiFormDefinition.schema.json",
  "view": {
    "kind": "Form",
    "properties": {
      "title": "Test Form View",
      "steps": [
        {
          "name": "basics",
          "label": "Basics",
          "elements": [
            ...
          ]
        }
      ]
    },
```

```
    "outputs": {
      ...
    }
  }
}
```

4. To see that works it without any modifications, select **Preview**.

The sandbox displays the form. It has fields for selecting a subscription, resource group, and region. It also fields for all of the parameters from the template.

Most of the fields are text boxes, but some fields are specific for the type of parameter. When your template includes allowed values for a parameter, the autogenerated form uses a drop-down element. The drop-down element is pre-populated with the allowed values.

In between the title and **Project details**, there are no tabs because the default form only has one step defined. In the **Customize form** section, you'll break the parameters into multiple tabs.

> ⚠ **Warning**
>
> Don't select **Create** as it will launch a real deployment. You'll have a chance to deploy the template spec later in this tutorial.

5. To exit from the preview, select **Cancel**.

Customize form

The default form is a good starting point for understanding forms but usually you'll want to customize it. You can edit it in the sandbox or in Visual Studio Code. The preview option is only available in the sandbox.

1. Give the form a **title** that describes its use.

   JSON

   ```
   {
     "$schema": "https://schema.management.azure.com/schemas/2021-09-09/uiFormDefinition.schema.json#",
     "view": {
       "kind": "Form",
       "properties": {
         "title": "Key Vault and secret",
   ```

2. Your default form had all of the fields for your template combined into one step called **Basics**. To help users to understand the values they're providing, divide the form into steps. Each step contains fields related to a logical part of the solution to deploy.

   Find the step labeled **Basics**. You'll keep this step but add steps below it. The new steps will focus on configuring the key vault, setting user permissions, and specifying the secret. Make sure you add a comma after the basics step.

   JSON

   ```
   {
     "$schema": "https://schema.management.azure.com/schemas/2021-09-09/uiFormDefinition.schema.json#",
     "view": {
       "kind": "Form",
       "properties": {
         "title": "Key Vault and secret",
         "steps": [
           {
             "name": "basics",
             "label": "Basics",
             "elements": [
               ...
             ]
           },
   ```

```
        {
          "name": "keyvault",
          "label": "Key Vault",
          "elements": [
          ]
        },
        {
          "name": "permissions",
          "label": "Permissions",
          "elements": [
          ]
        },
        {
          "name": "secret",
          "label": "Secret",
          "elements": [
          ]
        }
      ]
    },
    "outputs": {
      ...
    }
  }
}
```

> **Important**
>
> Properties in the form are case-sensitive. Make sure you use the casing shown in the examples.

3. Select **Preview**. You'll see the steps, but most of them don't have any elements.

4. Now, move elements to the appropriate steps. Start with the elements labeled **Secret Name** and **Secret Value**. Remove these elements from the **Basics** step and add them to the **Secret** step.

```
JSON
```

```
{
  "name": "secret",
  "label": "Secret",
  "elements": [
  {
      "name": "secretName",
      "type": "Microsoft.Common.TextBox",
      "label": "Secret Name",
      "defaultValue": "",
      "toolTip": "Specifies the name of the secret that you want to
create.",
      "constraints": {
        "required": true,
        "regex": "",
        "validationMessage": ""
      },
      "visible": true
    },
    {
      "name": "secretValue",
      "type": "Microsoft.Common.PasswordBox",
      "label": {
        "password": "Secret Value",
        "confirmPassword": "Confirm password"
      },
      "toolTip": "Specifies the value of the secret that you want to
create.",
      "constraints": {
        "required": true,
        "regex": "",
        "validationMessage": ""
      },
      "options": {
        "hideConfirmation": true
      },
      "visible": true
    }
  ]
}
```

5. When you move elements, you need to fix the `outputs` section. Currently, the outputs section references those elements as if they were still in the basics step. Fix the syntax so it references the elements in the `secret` step.

JSON

```
"outputs": {
  "parameters": {
    ...
    "secretName": "[steps('secret').secretName]",
```

```
    "secretValue": "[steps('secret').secretValue]"
  }
```

6. Continue moving elements to the appropriate steps. Rather than go through each one, take a look at the updated form.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2021-09-09/uiFormDefinition.schema.json#",
  "view": {
    "kind": "Form",
    "properties": {
      "title": "Key Vault and secret",
      "steps": [
        {
          "name": "basics",
          "label": "Basics",
          "elements": [
            {
              "name": "resourceScope",
              "type": "Microsoft.Common.ResourceScope",
              "location": {
                "resourceTypes": [
                  "microsoft.resources/resourcegroups"
                ]
              }
            }
          ]
        },
        {
          "name": "keyvault",
          "label": "Key Vault",
          "elements": [
            {
              "name": "keyVaultName",
              "type": "Microsoft.Common.TextBox",
              "label": "Key Vault Name",
              "defaultValue": "",
              "toolTip": "Specifies the name of the key vault.",
              "constraints": {
                "required": true,
                "regex": "",
                "validationMessage": ""
              },
              "visible": true
            },
            {
              "name": "skuName",
              "type": "Microsoft.Common.DropDown",
              "label": "Sku Name",
              "defaultValue": "Standard",
```

```
              "toolTip": "Specifies whether the key vault is a standard
vault or a premium vault.",
              "constraints": {
                "required": false,
                "allowedValues": [
                  {
                    "label": "Standard",
                    "value": "Standard"
                  },
                  {
                    "label": "Premium",
                    "value": "Premium"
                  }
                ]
              },
              "visible": true
            },
            {
              "name": "location",
              "type": "Microsoft.Common.TextBox",
              "label": "Location",
              "defaultValue": "[[resourceGroup().location]",
              "toolTip": "Specifies the Azure location where the key
vault should be created.",
              "constraints": {
                "required": false,
                "regex": "",
                "validationMessage": ""
              },
              "visible": true
            },
            {
              "name": "enabledForDeployment",
              "type": "Microsoft.Common.DropDown",
              "label": "Enabled For Deployment",
              "defaultValue": "false",
              "toolTip": "Specifies whether Azure Virtual Machines are
permitted to retrieve certificates stored as secrets from the key
vault.",
              "constraints": {
                "required": false,
                "allowedValues": [
                  {
                    "label": "true",
                    "value": true
                  },
                  {
                    "label": "false",
                    "value": false
                  }
                ]
              },
              "visible": true
            },
            {
```

```
              "name": "enabledForDiskEncryption",
              "type": "Microsoft.Common.DropDown",
              "label": "Enabled For Disk Encryption",
              "defaultValue": "false",
              "toolTip": "Specifies whether Azure Disk Encryption is
permitted to retrieve secrets from the vault and unwrap keys.",
              "constraints": {
                "required": false,
                "allowedValues": [
                  {
                    "label": "true",
                    "value": true
                  },
                  {
                    "label": "false",
                    "value": false
                  }
                ]
              },
              "visible": true
            },
            {
              "name": "enabledForTemplateDeployment",
              "type": "Microsoft.Common.DropDown",
              "label": "Enabled For Template Deployment",
              "defaultValue": "false",
              "toolTip": "Specifies whether Azure Resource Manager is
permitted to retrieve secrets from the key vault.",
              "constraints": {
                "required": false,
                "allowedValues": [
                  {
                    "label": "true",
                    "value": true
                  },
                  {
                    "label": "false",
                    "value": false
                  }
                ]
              },
              "visible": true
            }
          ]
        },
        {
          "name": "permissions",
          "label": "Permissions",
          "elements": [
            {
              "name": "tenantId",
              "type": "Microsoft.Common.TextBox",
              "label": "Tenant Id",
              "defaultValue": "[[subscription().tenantId]",
              "toolTip": "Specifies the Azure Active Directory tenant
```

```
ID that should be used for authenticating requests to the key vault.
Get it by using Get-AzSubscription cmdlet.",
              "constraints": {
                "required": false,
                "regex": "",
                "validationMessage": ""
              },
              "visible": true
            },
            {
              "name": "objectId",
              "type": "Microsoft.Common.TextBox",
              "label": "Object Id",
              "defaultValue": "",
              "toolTip": "Specifies the object ID of a user, service
principal or security group in the Azure Active Directory tenant for
the vault. The object ID must be unique for the list of access
policies. Get it by using Get-AzADUser or Get-AzADServicePrincipal
cmdlets.",
              "constraints": {
                "required": true,
                "regex": "",
                "validationMessage": ""
              },
              "visible": true
            },
            {
              "name": "keysPermissions",
              "type": "Microsoft.Common.TextBox",
              "label": "Keys Permissions",
              "defaultValue": "[[\"list\"]",
              "toolTip": "Specifies the permissions to keys in the
vault. Valid values are: all, encrypt, decrypt, wrapKey, unwrapKey,
sign, verify, get, list, create, update, import, delete, backup,
restore, recover, and purge.",
              "constraints": {
                "required": false,
                "regex": "",
                "validationMessage": ""
              },
              "visible": true
            },
            {
              "name": "secretsPermissions",
              "type": "Microsoft.Common.TextBox",
              "label": "Secrets Permissions",
              "defaultValue": "[[\"list\"]",
              "toolTip": "Specifies the permissions to secrets in the
vault. Valid values are: all, get, list, set, delete, backup, restore,
recover, and purge.",
              "constraints": {
                "required": false,
                "regex": "",
                "validationMessage": ""
              },
```

```
              "visible": true
            }
          ]
        },
        {
          "name": "secret",
          "label": "Secret",
          "elements": [
            {
              "name": "secretName",
              "type": "Microsoft.Common.TextBox",
              "label": "Secret Name",
              "defaultValue": "",
              "toolTip": "Specifies the name of the secret that you
want to create.",
              "constraints": {
                "required": true,
                "regex": "",
                "validationMessage": ""
              },
              "visible": true
            },
            {
              "name": "secretValue",
              "type": "Microsoft.Common.PasswordBox",
              "label": {
                "password": "Secret Value",
                "confirmPassword": "Confirm password"
              },
              "toolTip": "Specifies the value of the secret that you
want to create.",
              "constraints": {
                "required": true,
                "regex": "",
                "validationMessage": ""
              },
              "options": {
                "hideConfirmation": true
              },
              "visible": true
            }
          ]
        }
      ]
    },
    "outputs": {
      "parameters": {
        "keyVaultName": "[steps('keyvault').keyVaultName]",
        "location": "[steps('keyvault').location]",
        "enabledForDeployment": "
[steps('keyvault').enabledForDeployment]",
        "enabledForDiskEncryption": "
[steps('keyvault').enabledForDiskEncryption]",
        "enabledForTemplateDeployment": "
[steps('keyvault').enabledForTemplateDeployment]",
```

```
          "tenantId": "[steps('permissions').tenantId]",
          "objectId": "[steps('permissions').objectId]",
          "keysPermissions": "[steps('permissions').keysPermissions]",
          "secretsPermissions": "
[steps('permissions').secretsPermissions]",
          "skuName": "[steps('keyvault').skuName]",
          "secretName": "[steps('secret').secretName]",
          "secretValue": "[steps('secret').secretValue]"
        },
        "kind": "ResourceGroup",
        "location": "[steps('basics').resourceScope.location.name]",
        "resourceGroupId": "
[steps('basics').resourceScope.resourceGroup.id]"
      }
    }
}
```

7. Save this file locally with the name **keyvaultform.json**.

Create template spec

When you create the template spec, provide both files.

For PowerShell, use New-AzTemplateSpec and provide the form in the `-UIFormDefinitionFile` parameter.

Azure PowerShell

```
New-AzTemplateSpec `
  -name keyvaultspec `
  -version 1 `
  -ResourceGroupName templateSpecRG `
  -location westus2 `
  -templatefile keyvault.json `
  -UIFormDefinitionFile keyvaultform.json
```

For Azure CLI, use az ts create and provide the form in the `--ui-form-definition` parameter.

Azure CLI

```
az ts create \
  --name keyvaultspec \
  --version 1 \
  --resource-group templatespecRG \
  --location westus2 \
  --template-file keyvault.json \
  --ui-form-definition keyvaultform.json
```

Deploy through portal

To test the form, go to the portal and navigate to your template spec. Select **Deploy**.

You'll see the form you created. Go through the steps and provide values for the fields.

On the **Basics** step, you'll see a field for **Region**. This field is used for the location of the resource group. On the **Key Vault** step, you'll see a field for **Location**. This field is used for the location of the key vault.

On the **Permissions** step, you can provide your own user ID for the object ID. Use the default value (`["list"]`) for key and secret permissions. You'll improve that option in the next section.

When you have finished providing values, select **Create** to deploy the template spec.

Improve the form

In the previous section, you added steps and moved elements, but you didn't change any of the default behaviors. In this section, you'll make changes that improve the experience for users of your template spec.

Previously, the two permissions fields were text boxes. Now, you'll use a drop-down. Set the type to `Microsoft.Common.DropDown`.

Update `keysPermissions`:

JSON

```
{
  "name": "keysPermissions",
  "type": "Microsoft.Common.DropDown",
```

And `secretsPermissions`:

JSON

```
{
  "name": "secretsPermissions",
  "type": "Microsoft.Common.DropDown",
```

These fields need to pass an array to the template. A regular drop-down won't work because it only lets you select one value. To select more than one value and pass them as an array, add the `multiselect` field and set to `true`.

JSON

```
{
  "name": "keysPermissions",
  "type": "Microsoft.Common.DropDown",
  "label": "Keys Permissions",
  "multiselect": true,
```

JSON

```
{
  "name": "secretsPermissions",
  "type": "Microsoft.Common.DropDown",
  "label": "Secrets Permissions",
  "multiselect": true,
```

Finally, you must specify the allowed values for the drop-down and a default value.

JSON

```
{
  "name": "keysPermissions",
  "type": "Microsoft.Common.DropDown",
  "label": "Keys Permissions",
  "multiselect": true,
  "defaultValue":{
    "value": "list"
  },
  "toolTip": "Specifies the permissions to keys in the vault. Valid values
```

```
are: all, encrypt, decrypt, wrapKey, unwrapKey, sign, verify, get, list,
create, update, import, delete, backup, restore, recover, and purge.",
  "constraints": {
    "required": false,
    "allowedValues":[
      {
        "label": "all",
        "value": "all"
      },
      {
        "label": "encrypt",
        "value": "encrypt"
      },
      {
        "label": "decrypt",
        "value": "decrypt"
      },
      {
        "label": "list",
        "value": "list"
      },
      {
        "label": "delete",
        "value": "delete"
      },
      {
        "label": "backup",
        "value": "backup"
      },
      {
        "label": "restore",
        "value": "restore"
      },
      {
        "label": "recover",
        "value": "recover"
      },
      {
        "label": "purge",
        "value": "purge"
      },
      {
        "label": "wrapKey",
        "value": "wrapKey"
      },
      {
        "label": "unwrapKey",
        "value": "unwrapKey"
      },
      {
        "label": "sign",
        "value": "sign"
      },
      {
        "label": "verify",
```

```
          "value": "verify"
        },
        {
          "label": "get",
          "value": "get"
        },
        {
          "label": "create",
          "value": "create"
        },
        {
          "label": "update",
          "value": "update"
        },
        {
          "label": "import",
          "value": "import"
        }
      ]
    },
    "visible": true
  },
  {
    "name": "secretsPermissions",
    "type": "Microsoft.Common.DropDown",
    "label": "Secrets Permissions",
    "multiselect": true,
    "defaultValue":{
      "value": "list"
    },
    "toolTip": "Specifies the permissions to secrets in the vault. Valid
values are: all, get, list, set, delete, backup, restore, recover, and
purge.",
    "constraints": {
      "required": false,
```

```
      "allowedValues":[
        {
          "label": "all",
          "value": "all"
        },
        {
          "label": "get",
          "value": "get"
        },
        {
          "label": "list",
          "value": "list"
        },
        {
          "label": "set",
          "value": "set"
        },
        {
          "label": "delete",
          "value": "delete"
        },
        {
          "label": "backup",
          "value": "backup"
        },
        {
          "label": "restore",
          "value": "restore"
        },
        {
          "label": "recover",
          "value": "recover"
        },
        {
          "label": "purge",
          "value": "purge"
        }
      ]
    },
    "visible": true
  }
```

Create a new version of template spec.

With PowerShell:

Azure PowerShell

```
New-AzTemplateSpec `
  -name keyvaultspec `
  -version 2 `
  -ResourceGroupName templateSpecRG `
  -location westus2 `
```

```
  -templatefile keyvault.json `
  -UIFormDefinitionFile keyvaultform.json
```

Or Azure CLI:

Azure CLI

```
az ts create \
  --name keyvaultspec \
  --version 2 \
  --resource-group templatespecRG \
  --location westus2 \
  --template-file keyvault.json \
  --ui-form-definition keyvaultform.json
```

Redeploy your template spec with the improved portal form.

Notice that your permission fields are now drop-down that allow multiple values.

Next steps

To learn about deploying a template spec as a linked template, see Tutorial: Deploy a template spec as a linked template.

Tutorial: Utilize the ARM template reference

Article • 04/09/2023

Learn how to find the template schema information, and use the information to create Azure Resource Manager templates (ARM templates).

In this tutorial, you use a base template from Azure Quickstart Templates. Using template reference documentation, you validate the template.

This tutorial covers the following tasks:

- ✓ Open a Quickstart template
- ✓ Understand the template
- ✓ Find the template reference
- ✓ Edit the template
- ✓ Deploy the template

If you don't have an Azure subscription, create a free account before you begin.

Prerequisites

To complete this article, you need:

- Visual Studio Code with Resource Manager Tools extension. See Quickstart: Create ARM templates with Visual Studio Code.

Open a Quickstart template

Azure Quickstart Templates is a repository for ARM templates. Instead of creating a template from scratch, you can find a sample template and customize it. The template used in this quickstart is called Create a standard storage account. The template defines an Azure Storage account resource.

1. From Visual Studio Code, select **File** > **Open File**.

2. In **File name**, paste the following URL:

   ```
   https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.storage/storage-account-create/azuredeploy.json
   ```

   (header: url)

3. Select **Open** to open the file.

4. Select **File** > **Save As** to save the file as *azuredeploy.json* to your local computer.

Understand the schema

1. From Visual Studio Code, collapse the template to the root level. You have the simplest structure with the following elements:

```
1   {
2     "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
3     "contentVersion": "1.0.0.0",
4   > "metadata": { ···
10    },
      Select or create a parameter file...
11  > "parameters": { ···
43    },
44  > "resources": [ ···
56    ],
57  > "outputs": { ···
66    }
67  }
```

   - `$schema`: specify the location of the JSON schema file that describes the version of the template language.
   - `contentVersion`: specify any value for this element to document significant changes in your template.
   - `metadata`: specify the metadata of the template.
   - `parameters`: specify the values that are provided when deployment is executed to customize resource deployment.
   - `variables`: specify the values that are used as JSON fragments in the template to simplify template language expressions. This template doesn't contain this element.
   - `resources`: specify the resource types that are deployed or updated in a resource group.
   - `outputs`: specify the values that are returned after deployment.

2. Expand `resources`. There's a `Microsoft.Storage/storageAccounts` resource defined. The API version shown on the screenshot is **2021-06-01**. The SKU name uses a parameter value. The parameter is called `storageAccountType`.

```
"resources": [
  {
    "type": "Microsoft.Storage/storageAccounts",
    "apiVersion": "2021-06-01",
    "name": "[parameters('storageAccountName')]",
    "location": "[parameters('location')]",
    "sku": {
      "name": "[parameters('storageAccountType')]"
    },
    "kind": "StorageV2",
    "properties": {}
  }
],
```

3. Expand `parameters` to see how `storageAccountType` is defined. The parameter has eight allowed values.

```
"parameters": {
  "storageAccountType": {
    "type": "string",
    "defaultValue": "Standard_LRS",
    "allowedValues": [
      "Premium_LRS",
      "Premium_ZRS",
      "Standard_GRS",
      "Standard_GZRS",
      "Standard_LRS",
      "Standard_RAGRS",
      "Standard_RAGZRS",
      "Standard_ZRS"
    ],
    "metadata": {
      "description": "Storage Account type"
    }
  },
```

Using the template reference, you can find out whether you are using the latest API version, and whether the parameter definition includes all the allowed values.

Find the template reference

1. Browse to Azure Template reference.

2. In the **Filter by title** box, enter **storage accounts**, and select the first **Storage Accounts** under **Reference** > **Storage**.

3. A resource type usually has several API versions. This page shows the latest template schema version by default. Select the **Latest** dropdown box to see the versions. The latest version shown on the screenshot is **2021-06-01**. Select either **Latest** or the version right beneath **Latest** to see the latest version. Make sure this version matches the version used for the storage account resource in your template. If you update the API version, verify the resource definition matches the template reference.

4. This page lists the details of the storageAccount resource type. For example, it lists the allowed values for the **Sku** object. There are eight SKUs, and they match the ones listed in the quickstart template that you opened earlier.

Sku

Name	Description	Value
name	The SKU name. Required for account creation; optional for update. Note that in older versions, SKU name was called accountType.	'Premium_LRS' 'Premium_ZRS' 'Standard_GRS' 'Standard_GZRS' 'Standard_LRS' 'Standard_RAGRS' 'Standard_RAGZRS' 'Standard_ZRS'

Deploy the template

1. Sign in to the Azure Cloud Shell

2. Choose your preferred environment by selecting either **PowerShell** or **Bash** (for CLI) on the upper left corner. Restarting the shell is required when you switch.

3. Select **Upload/download files**, and then select **Upload**. See the previous screenshot. Select the file you saved in the previous section. After uploading the file, you can use the `ls` command and the `cat` command to verify the file was uploaded successfully.

4. From the Cloud Shell, run the following commands. Select the tab to show the PowerShell code or the CLI code.

   When you deploy the template, specify the `storageAccountType` parameter with a newly added value, for example, **Standard_RAGRS**. The deploy would fail if you used the original quickstart template because **Standard_RAGRS** wasn't an allowed value.

CLI

Azure CLI

```
echo "Enter a project name that is used to generate resource group
name:" &&
read projectName &&
echo "Enter the location (i.e. centralus):" &&
read location &&
resourceGroupName="${projectName}rg" &&
az group create --name $resourceGroupName --location "$location" &&
az deployment group create --resource-group $resourceGroupName --
template-file "$HOME/azuredeploy.json" --parameters
storageAccountType='Standard_RAGRS'
```

Clean up resources

When the Azure resources are no longer needed, clean up the resources you deployed by deleting the resource group.

1. From the Azure portal, select **Resource group** from the left menu.
2. Enter the resource group name in the **Filter by name** field.
3. Select the resource group name. You shall see a total of six resources in the resource group.
4. Select **Delete resource group** from the top menu.

Next steps

In this tutorial, you learned how to use template reference to customize an existing template. To learn how to create multiple storage account instances, see:

Create multiple instances

Tutorial: Create multiple resource instances with ARM templates

Article • 03/09/2023

Learn how to iterate in your Azure Resource Manager template (ARM template) to create multiple instances of an Azure resource. In this tutorial, you modify a template to create three storage account instances.

This tutorial covers the following tasks:

- ✓ Open a Quickstart template
- ✓ Edit the template
- ✓ Deploy the template

If you don't have an Azure subscription, create a free account before you begin.

For a Learn module that covers resource copy, see Manage complex cloud deployments by using advanced ARM template features.

Prerequisites

To complete this article, you need:

- Visual Studio Code with Resource Manager Tools extension. See Quickstart: Create ARM templates with Visual Studio Code.

Open a Quickstart template

Azure Quickstart Templates is a repository for ARM templates. Instead of creating a template from scratch, you can find a sample template and customize it. The template used in this quickstart is called Create a standard storage account. The template defines an Azure Storage account resource.

1. From Visual Studio Code, select **File** > **Open File**.

2. In **File name**, paste the following URL:

```url
https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.storage/storage-account-create/azuredeploy.json
```

3. Select **Open** to open the file.

4. There's a `Microsoft.Storage/storageAccounts` resource defined in the template. Compare the template to the template reference. It's helpful to get some basic understanding of the template before customizing it.

5. Select **File** > **Save As** to save the file as *azuredeploy.json* to your local computer.

Edit the template

The existing template creates one storage account. You customize the template to create three storage accounts.

From Visual Studio Code, make the following four changes:

```
1   {
2     "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
3     "contentVersion": "1.0.0.0",
4     "metadata": { ···
10    },
      Select or create a parameter file...
11    "parameters": {
12      "storageAccountType": { ···
28      },
29      "location": { ···
35      },
36      "storageAccountName": {
37        "type": "string",
38        "defaultValue": "[format('store{0}', uniqueString(resourceGroup().id))]",
39        "metadata": {
40          "description": "The name of the Storage Account"
41        }
42      }
43    },
44    "resources": [
45      {
46        "type": "Microsoft.Storage/storageAccounts",
47        "apiVersion": "2021-06-01",
48        "name": "[format('{0}storage{1}', copyIndex(), uniqueString(resourceGroup().id))]",
49        "location": "[parameters('location')]",
50        "sku": {
51          "name": "[parameters('storageAccountType')]"
52        },
53        "kind": "StorageV2",
54        "copy": {
55          "name": "storageCopy",
56          "count": 3
57        },
58        "properties": {}
59      }
60    ],
61    "outputs": { ···
70    }
71  }
```

(Remove this part)

(Remove this part)

(Remove this part)

1. Add a `copy` element to the storage account resource definition. In the `copy` element, you specify the number of iterations and a variable for this loop. The count value must be a positive integer and can't exceed 800.

JSON

```
"copy": {
  "name": "storageCopy",
  "count": 3
},
```

2. The `copyIndex()` function returns the current iteration in the loop. You use the index as the name prefix. `copyIndex()` is zero-based. To offset the index value, you can pass a value in the `copyIndex()` function. For example, `copyIndex(1)`.

JSON

```
"name": "[format('{0}storage{1}', copyIndex(),
uniqueString(resourceGroup().id))]",
```

3. Delete the `storageAccountName` parameter definition, because it's not used anymore.

4. Delete the `outputs` element. It's no longer needed.

5. Delete the `metadata` element.

The completed template looks like:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageAccountType": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Premium_LRS",
        "Premium_ZRS",
        "Standard_GRS",
        "Standard_GZRS",
        "Standard_LRS",
        "Standard_RAGRS",
        "Standard_RAGZRS",
        "Standard_ZRS"
      ],
      "metadata": {
        "description": "Storage Account type"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for the storage account."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-06-01",
      "name": "[format('{0}storage{1}', copyIndex(),
```

```
uniqueString(resourceGroup().id))]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('storageAccountType')]"
      },
      "kind": "StorageV2",
      "copy": {
        "name": "storageCopy",
        "count": 3
      },
      "properties": {}
    }
  ]
}
```

Save the changes.

For more information about creating multiple instances, see Resource iteration in ARM templates

Deploy the template

1. Sign in to the Azure Cloud Shell

2. Choose your preferred environment by selecting either **PowerShell** or **Bash** (for CLI) on the upper left corner. Restarting the shell is required when you switch.

3. Select **Upload/download files**, and then select **Upload**. See the previous screenshot. Select the file you saved in the previous section. After uploading the file, you can use the `ls` command and the `cat` command to verify the file was uploaded successfully.

4. From the Cloud Shell, run the following commands. Select the tab to show the PowerShell code or the CLI code.

CLI

Azure CLI

```
echo "Enter a project name that is used to generate resource group
name:" &&
read projectName &&
echo "Enter the location (i.e. centralus):" &&
read location &&
resourceGroupName="${projectName}rg" &&
az group create --name $resourceGroupName --location "$location" &&
az deployment group create --resource-group $resourceGroupName --
template-file "$HOME/azuredeploy.json"
```

After a successful template deployment, you can display the three storage accounts created in the specified resource group. Compare the storage account names with the name definition in the template.

CLI

Azure CLI

```
echo "Enter a project name that is used to generate resource group
name:" &&
read projectName &&
resourceGroupName="${projectName}rg" &&
az storage account list --resource-group $resourceGroupName &&
echo "Press [ENTER] to continue ..."
```

Clean up resources

When the Azure resources are no longer needed, clean up the resources you deployed by deleting the resource group.

1. From the Azure portal, select **Resource group** from the left menu.
2. Enter the resource group name in the **Filter by name** field.
3. Select the resource group name. You shall see a total of three resources in the resource group.
4. Select **Delete resource group** from the top menu.

Next steps

In this tutorial, you learned how to create multiple storage account instances. In the next tutorial, you develop a template with multiple resources and multiple resource types. Some of the resources have dependent resources.

Create dependent resources

Tutorial: Create ARM templates with dependent resources

Article • 04/09/2023

Learn how to create an Azure Resource Manager template (ARM template) to deploy multiple resources and configure the deployment order. After you create the template, you deploy the template using Azure Cloud Shell from the Azure portal.

In this tutorial, you create a storage account, a virtual machine, a virtual network, and some other dependent resources. Some of the resources cannot be deployed until another resource exists. For example, you can't create the virtual machine until its storage account and network interface exist. You define this relationship by making one resource as dependent on the other resources. Resource Manager evaluates the dependencies between resources, and deploys them in their dependent order. When resources aren't dependent on each other, Resource Manager deploys them in parallel. For more information, see Define the order for deploying resources in ARM templates.

This tutorial covers the following tasks:

- ✓ Open a Quickstart template
- ✓ Explore the template
- ✓ Deploy the template

If you don't have an Azure subscription, create a free account before you begin.

For a Learn module that covers resource dependencies, see Manage complex cloud deployments by using advanced ARM template features.

Prerequisites

To complete this article, you need:

- Visual Studio Code with Resource Manager Tools extension. See Quickstart: Create ARM templates with Visual Studio Code.
- To increase security, use a generated password for the virtual machine administrator account. You can use Azure Cloud Shell to run the following command in PowerShell or Bash:

  ```shell
  openssl rand -base64 32
  ```

  To learn more, run `man openssl rand` to open the manual page.

  Azure Key Vault is designed to safeguard cryptographic keys and other secrets. For more information, see Tutorial: Integrate Azure Key Vault in ARM template deployment. We also recommend you to update your password every three months.

Open a Quickstart template

Azure Quickstart Templates is a repository for ARM templates. Instead of creating a template from scratch, you can find a sample template and customize it. The template used in this tutorial is called Deploy a simple Windows VM.

1. From Visual Studio Code, select **File** > **Open File**.
2. In **File name**, paste the following URL:

   ```url
   https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.compute/vm-simple-windows/azuredeploy.json
   ```

3. Select **Open** to open the file.
4. Select **File** > **Save As** to save a copy of the file to your local computer with the name *azuredeploy.json*.

Explore the template

When you explore the template in this section, try to answer these questions:

- How many Azure resources defined in this template?
- One of the resources is an Azure storage account. Does the definition look like the one used in the last tutorial?
- Can you find the template references for the resources defined in this template?
- Can you find the dependencies of the resources?

1. From Visual Studio Code, collapse the elements until you only see the first-level elements and the second-level elements inside `resources`:

```
1   {
2     "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
3     "contentVersion": "1.0.0.0",
4 >   "parameters": {···
53    },
54 >  "variables": {···
65    },
66    "resources": [
67 >    1 {···
77        },
78 >    2 {···
89        },
90 >    3 {···
113       },
114 >   4 {···
140       },
141 >   5 {···
166       },
167 >   6 {···
217       }
218   ],
219 >  "outputs": {···
224   }
225 }
```

   There are six resources defined by the template:

   - Microsoft.Storage/storageAccounts.
   - Microsoft.Network/publicIPAddresses.
   - Microsoft.Network/networkSecurityGroups.
   - Microsoft.Network/virtualNetworks.
   - Microsoft.Network/networkInterfaces.
   - Microsoft.Compute/virtualMachines.

     It's helpful to review the template reference before customizing a template.

2. Expand the first resource. It's a storage account. Compare the resource definition to the template reference.

```
"resources": [
  {
    "type": "Microsoft.Storage/storageAccounts",
    "apiVersion": "2018-11-01",
    "name": "[variables('storageAccountName')]",
    "location": "[parameters('location')]",
    "sku": {
      "name": "Standard_LRS"
    },
    "kind": "Storage",
    "properties": {}
  },
```

3. Expand the second resource. The resource type is `Microsoft.Network/publicIPAddresses`. Compare the resource definition to the template reference.

```
"resources": [
  { ...
  },
  {
    "type": "Microsoft.Network/publicIPAddresses",
    "apiVersion": "2018-11-01",
    "name": "[variables('publicIPAddressName')]",
    "location": "[parameters('location')]",
    "properties": {
      "publicIPAllocationMethod": "Dynamic",
      "dnsSettings": {
        "domainNameLabel": "[parameters('dnsLabelPrefix')]"
      }
    }
  },
```

4. Expand the third resource. The resource type is `Microsoft.Network/networkSecurityGroups`. Compare the resource definition to the template reference.

```
"resources": [
  { ...
  },
  { ...
  },
  {
    "comments": "Default Network Security Group for template",
    "type": "Microsoft.Network/networkSecurityGroups",
    "apiVersion": "2019-08-01",
    "name": "[variables('networkSecurityGroupName')]",
    "location": "[parameters('location')]",
    "properties": {
      "securityRules": [
        {
          "name": "default-allow-3389",
          "properties": {
            "priority": 1000,
            "access": "Allow",
            "direction": "Inbound",
            "destinationPortRange": "3389",
            "protocol": "Tcp",
            "sourcePortRange": "*",
            "sourceAddressPrefix": "*",
            "destinationAddressPrefix": "*"
          }
        }
      ]
    }
  },
```

5. Expand the fourth resource. The resource type is `Microsoft.Network/virtualNetworks`:

```
"resources": [
  { ...
  },
  { ...
  },
  { ...
  },
  {
    "type": "Microsoft.Network/virtualNetworks",
    "apiVersion": "2018-11-01",
    "name": "[variables('virtualNetworkName')]",
    "location": "[parameters('location')]",
    "dependsOn": [
      "[resourceId('Microsoft.Network/networkSecurityGroups', variables('networkSecurityGroupName'))]"
    ],
    "properties": {
      "addressSpace": {
        "addressPrefixes": [
          "[variables('addressPrefix')]"
        ]
      },
      "subnets": [
        {
          "name": "[variables('subnetName')]",
          "properties": {
            "addressPrefix": "[variables('subnetPrefix')]",
            "networkSecurityGroup": {
              "id": "[resourceId('Microsoft.Network/networkSecurityGroups', variables('networkSecurityGroupName'))]"
            }
          }
        }
      ]
    }
  },
```

The `dependsOn` element enables you to define one resource as a dependent on one or more resources. This resource depends on one other resource:

- `Microsoft.Network/networkSecurityGroups`

6. Expand the fifth resource. The resource type is `Microsoft.Network/networkInterfaces`. The resource depends on two other resources:

   - `Microsoft.Network/publicIPAddresses`
   - `Microsoft.Network/virtualNetworks`

7. Expand the sixth resource. This resource is a virtual machine. It depends on two other resources:

   - `Microsoft.Storage/storageAccounts`
   - `Microsoft.Network/networkInterfaces`

The following diagram illustrates the resources and the dependency information for this template:

By specifying the dependencies, Resource Manager efficiently deploys the solution. It deploys the storage account, public IP address, and virtual network in parallel because they have no dependencies. After the public IP address and virtual network are deployed, the network interface is created. When all other resources are deployed, Resource Manager deploys the virtual machine.

Deploy the template

1. Sign in to Cloud Shell.

2. Choose your preferred environment by selecting either **PowerShell** or **Bash** (for CLI) on the upper left corner. Restarting the shell is required when you switch.

3. Select **Upload/download files**, and then select **Upload**. See the previous screenshot. Select the file you saved earlier. After uploading the file, you can use the `ls` command and the `cat` command to verify the file was uploaded successfully.

4. Run the following PowerShell script to deploy the template.

CLI

Azure CLI

```
echo "Enter a project name that is used to generate resource group name:" &&
read projectName &&
echo "Enter the location (i.e. centralus):" &&
read location &&
echo "Enter the virtual machine admin username:" &&
read adminUsername &&
echo "Enter the DNS label prefix:" &&
read dnsLabelPrefix &&
resourceGroupName="${projectName}rg" &&
az group create --name $resourceGroupName --location $location &&
az deployment group create --resource-group $resourceGroupName --template-file "$HOME/azuredeploy.json" --parameters adminUsername=$adminUsername dnsLabelPrefix=$dnsLabelPrefix
```

5. RDP to the virtual machine to verify the virtual machine has been created successfully.

Clean up resources

When the Azure resources are no longer needed, clean up the resources you deployed by deleting the resource group.

1. From the Azure portal, select **Resource group** from the left menu.
2. Enter the resource group name in the **Filter by name** field.
3. Select the resource group name. You'll see a total of six resources in the resource group.
4. Select **Delete resource group** from the top menu.

Next steps

In this tutorial, you developed and deployed a template to create a virtual machine, a virtual network, and the dependent resources. To learn how to use deployment scripts to perform pre/post deployment operations, see:

Use deployment script

Tutorial: Use deployment scripts to create a self-signed certificate

Article • 03/09/2023

Learn how to use deployment scripts in Azure Resource Manager templates (ARM templates). Deployment scripts can be used to perform custom steps that can't be done by ARM templates. For example, creating a self-signed certificate. In this tutorial, you create a template to deploy an Azure key vault, and then use a `Microsoft.Resources/deploymentScripts` resource in the same template to create a certificate and then add the certificate to the key vault. To learn more about deployment script, see Use deployment scripts in ARM templates.

> **Important**
>
> Two deployment script resources, a storage account and a container instance, are created in the same resource group for script execution and troubleshooting. These resources are usually deleted by the script service when the script execution gets in a terminal state. You're billed for the resources until the resources are deleted. To learn more, see **Clean up deployment script resources**.

This tutorial covers the following tasks:

- ✓ Open a quickstart template
- ✓ Edit the template
- ✓ Deploy the template
- ✓ Debug the failed script
- ✓ Clean up resources

For a Learn module that covers deployment scripts, see Extend ARM templates by using deployment scripts.

Prerequisites

To complete this article, you need:

- **Visual Studio Code with the Resource Manager Tools extension**. See Quickstart: Create ARM templates with Visual Studio Code.
- **A user-assigned managed identity**. This identity is used to perform Azure-specific actions in the script. To create one, see User-assigned managed identity. You need

the identity ID when you deploy the template. The format of the identity is:

JSON

```
/subscriptions/<SubscriptionID>/resourcegroups/<ResourceGroupName>/prov
iders/Microsoft.ManagedIdentity/userAssignedIdentities/<IdentityID>
```

Use the following CLI script to get the ID by providing the resource group name and the identity name.

CLI

Azure CLI

```
echo "Enter the Resource Group name:" &&
read resourceGroupName &&
az identity list -g $resourceGroupName
```

Open a Quickstart template

Instead of creating a template from scratch, you open a template from Azure Quickstart Templates. Azure Quickstart Templates is a repository for ARM templates.

The template used in this quickstart is called Create an Azure Key Vault and a secret. The template creates a key vault, and then adds a secret to the key vault.

1. From Visual Studio Code, select **File** > **Open File**.

2. In **File name**, paste the following URL:

   url

   ```
   https://raw.githubusercontent.com/Azure/azure-quickstart-
   templates/master/quickstarts/microsoft.keyvault/key-vault-
   create/azuredeploy.json
   ```

3. Select **Open** to open the file.

4. Select **File** > **Save As** to save the file as *azuredeploy.json* to your local computer.

Edit the template

Make the following changes to the template:

Clean up the template (optional)

The original template adds a secret to the key vault. To simplify the tutorial, remove the following resource:

- `Microsoft.KeyVault/vaults/secrets`

Remove the following two parameter definitions:

- `secretName`
- `secretValue`

If you choose not to remove these definitions, you need to specify the parameter values during the deployment.

Configure the key vault access policies

The deployment script adds a certificate to the key vault. Configure the key vault access policies to give the permission to the managed identity:

1. Add a parameter to get the managed identity ID:

   JSON

   ```
   "identityId": {
     "type": "string",
     "metadata": {
       "description": "Specifies the ID of the user-assigned managed
   identity."
     }
   },
   ```

   > **Note**
   >
   > The Resource Manager template extension of Visual Studio Code isn't capable to format deployment scripts yet. Don't use Shift+Alt+F to format the `deploymentScripts` resources, like the following one.

2. Add a parameter for configuring the key vault access policies so that the managed identity can add certificates to the key vault:

   JSON

```
"certificatesPermissions": {
  "type": "array",
  "defaultValue": [
    "get",
    "list",
    "update",
    "create"
  ],
  "metadata": {
  "description": "Specifies the permissions to certificates in the
vault. Valid values are: all, get, list, update, create, import,
delete, recover, backup, restore, manage contacts, manage certificate
authorities, get certificate authorities, list certificate authorities,
set certificate authorities, delete certificate authorities."
  }
}
```

3. Update the existing key vault access policies to:

JSON

```
"accessPolicies": [
  {
    "objectId": "[parameters('objectId')]",
    "tenantId": "[parameters('tenantId')]",
    "permissions": {
      "keys": "[parameters('keysPermissions')]",
      "secrets": "[parameters('secretsPermissions')]",
      "certificates": "[parameters('certificatesPermissions')]"
    }
  },
  {
    "objectId": "[reference(parameters('identityId'), '2018-11-
30').principalId]",
    "tenantId": "[parameters('tenantId')]",
    "permissions": {
      "keys": "[parameters('keysPermissions')]",
      "secrets": "[parameters('secretsPermissions')]",
      "certificates": "[parameters('certificatesPermissions')]"
    }
  }
],
```

There are two policies defined, one for the signed-in user, and the other is for the managed identity. The signed-in user only needs the *list* permission to verify the deployment. To simplify the tutorial, the same certificate is assigned to both the managed identity and the signed-in users.

Add the deployment script

1. Add three parameters that are used by the deployment script:

JSON

```
"certificateName": {
  "type": "string",
  "defaultValue": "DeploymentScripts2019"
},
"subjectName": {
  "type": "string",
  "defaultValue": "CN=contoso.com"
},
"utcValue": {
  "type": "string",
  "defaultValue": "[utcNow()]"
}
```

2. Add a `deploymentScripts` resource:

> **Note**
>
> Because the inline deployment scripts are enclosed in double quotes, the strings inside the deployment scripts need to be enclosed in single quotes instead. The **PowerShell escape character** is the backtick (`).

JSON

```
{
  "type": "Microsoft.Resources/deploymentScripts",
  "apiVersion": "2020-10-01",
  "name": "createAddCertificate",
  "location": "[resourceGroup().location]",
  "dependsOn": [
    "[resourceId('Microsoft.KeyVault/vaults',
parameters('keyVaultName'))]"
  ],
  "identity": {
    "type": "UserAssigned",
    "userAssignedIdentities": {
      "[parameters('identityId')]": {
      }
    }
  },
  "kind": "AzurePowerShell",
  "properties": {
    "forceUpdateTag": "[parameters('utcValue')]",
    "azPowerShellVersion": "3.0",
    "timeout": "PT30M",
    "arguments": "[format(' -vaultName {0} -certificateName {1} -
```

```
subjectName {2}', parameters('keyVaultName'),
parameters('certificateName'), parameters('subjectName'))]", // can
pass an argument string, double quotes must be escaped
    "scriptContent": "
      param(
        [string] [Parameter(Mandatory=$true)] $vaultName,
        [string] [Parameter(Mandatory=$true)] $certificateName,
        [string] [Parameter(Mandatory=$true)] $subjectName
      )

      $ErrorActionPreference = 'Stop'
      $DeploymentScriptOutputs = @{}

      $existingCert = Get-AzKeyVaultCertificate -VaultName $vaultName -
Name $certificateName

      if ($existingCert -and $existingCert.Certificate.Subject -eq
$subjectName) {

        Write-Host 'Certificate $certificateName in vault $vaultName is
already present.'

        $DeploymentScriptOutputs['certThumbprint'] =
$existingCert.Thumbprint
        $existingCert | Out-String
      }
      else {
        $policy = New-AzKeyVaultCertificatePolicy -SubjectName
$subjectName -IssuerName Self -ValidityInMonths 12 -Verbose

        # private key is added as a secret that can be retrieved in the
Resource Manager template
        Add-AzKeyVaultCertificate -VaultName $vaultName -Name
$certificateName -CertificatePolicy $policy -Verbose

        # it takes a few seconds for KeyVault to finish
        $tries = 0
        do {
          Write-Host 'Waiting for certificate creation completion...'
          Start-Sleep -Seconds 10
          $operation = Get-AzKeyVaultCertificateOperation -VaultName
$vaultName -Name $certificateName
          $tries++

          if ($operation.Status -eq 'failed')
          {
            throw 'Creating certificate $certificateName in vault
$vaultName failed with error $($operation.ErrorMessage)'
          }

          if ($tries -gt 120)
          {
            throw 'Timed out waiting for creation of certificate
$certificateName in vault $vaultName'
          }
```

```
        } while ($operation.Status -ne 'completed')

        $newCert = Get-AzKeyVaultCertificate -VaultName $vaultName -
Name $certificateName
        $DeploymentScriptOutputs['certThumbprint'] =
$newCert.Thumbprint
        $newCert | Out-String
      }
    ",
    "cleanupPreference": "OnSuccess",
    "retentionInterval": "P1D"
  }
}
```

The `deploymentScripts` resource depends on the key vault resource and the role assignment resource. It has these properties:

- `identity`: Deployment script uses a user-assigned managed identity to perform the operations in the script.
- `kind`: Specify the type of script. Currently, only PowerShell scripts are supported.
- `forceUpdateTag`: Determine whether the deployment script should be executed even if the script source hasn't changed. Can be current time stamp or a GUID. To learn more, see Run script more than once.
- `azPowerShellVersion`: Specifies the Azure PowerShell module version to be used. Currently, deployment script supports version 2.7.0, 2.8.0, and 3.0.0.
- `timeout`: Specify the maximum allowed script execution time specified in the ISO 8601 format. Default value is **P1D**.
- `arguments`: Specify the parameter values. The values are separated by spaces.
- `scriptContent`: Specify the script content. To run an external script, use `primaryScriptURI` instead. For more information, see Use external script. Declaring `$DeploymentScriptOutputs` is only required when testing the script on a local machine. Declaring the variable allows the script to be run on a local machine and in a `deploymentScript` resource without having to make changes. The value assigned to `$DeploymentScriptOutputs` is available as outputs in the deployments. For more information, see Work with outputs from PowerShell deployment scripts or Work with outputs from CLI deployment scripts.
- `cleanupPreference`: Specify the preference on when to delete the deployment script resources. The default value is **Always**, which means the deployment script resources are deleted despite the terminal state (Succeeded, Failed, Canceled). In this tutorial, **OnSuccess** is used so that you get a chance to view the script execution results.

- `retentionInterval`: Specify the interval for which the service retains the script resources after it reaches a terminal state. Resources will be deleted when this duration expires. Duration is based on ISO 8601 pattern. This tutorial uses **P1D**, which means one day. This property is used when `cleanupPreference` is set to **OnExpiration**. This property isn't enabled currently.

The deployment script takes three parameters: `keyVaultName`, `certificateName`, and `subjectName`. It creates a certificate, and then adds the certificate to the key vault.

`$DeploymentScriptOutputs` is used to store output value. To learn more, see Work with outputs from PowerShell deployment scripts or Work with outputs from CLI deployment scripts.

The completed template can be found here.

3. To see the debugging process, place an error in the code by adding the following line to the deployment script:

```
PowerShell
Write-Output1 $keyVaultName
```

The correct command is `Write-Output` instead of `Write-Output1`.

4. Select **File** > **Save** to save the file.

Deploy the template

1. Sign in to the Azure Cloud Shell

2. Choose your preferred environment by selecting either **PowerShell** or **Bash** (for CLI) on the upper-left corner. Restarting the shell is required when you switch.

3. Select **Upload/download files**, and then select **Upload**. See the previous screenshot. Select the file you saved in the previous section. After uploading the file, you can use the `ls` command and the `cat` command to verify the file was uploaded successfully.

4. Run the following Azure CLI or Azure PowerShell script to deploy the template.

CLI

Azure CLI

```
echo "Enter a project name that is used to generate resource
names:" &&
read projectName &&
echo "Enter the location (i.e. centralus):" &&
read location &&
echo "Enter your email address used to sign in to Azure:" &&
read upn &&
echo "Enter the user-assigned managed identity ID:" &&
read identityId &&
adUserId=$((az ad user show --id ${upn}) | jq -r '.id') &&
resourceGroupName="${projectName}rg" &&
keyVaultName="${projectName}kv" &&
az group create --name $resourceGroupName --location $location &&
az deployment group create --resource-group $resourceGroupName --
template-file "$HOME/azuredeploy.json" --parameters
identityId=$identityId keyVaultName=$keyVaultName
objectId=$adUserId
```

The deployment script service needs to create additional deployment script resources for script execution. The preparation and the cleanup process can take up to one minute to complete in addition to the actual script execution time.

The deployment failed because the invalid command, `Write-Output1` is used in the script. You will get an error saying:

error

```
The term 'Write-Output1' is not recognized as the name of a cmdlet,
function, script file, or operable
program. Check the spelling of the name, or if a path was included,
verify that the path is correct and try again.
```

The deployment script execution result is stored in the deployment script resources for the troubleshooting purpose.

Debug the failed script

1. Sign in to the Azure portal.

2. Open the resource group. It's the project name with **rg** appended. You will see two additional resources in the resource group. These resources are referred as *deployment script resources*.

   Both files have the *azscripts* suffix. One is a storage account and the other is a container instance.

   Select **Show hidden types** to list the `deploymentScripts` resource.

3. Select the storage account with the *azscripts* suffix.

4. Select the **File shares** tile. You will see an *azscripts* folder that contains the deployment script execution files.

5. Select *azscripts*. You will see two folders *azscriptinput* and *azscriptoutput*. The input folder contains a system PowerShell script file and the user deployment script files. The output folder contains a *executionresult.json* and the script output file. You can see the error message in *executionresult.json*. The output file isn't there because the execution failed.

Remove the `Write-Output1` line and redeploy the template.

When the second deployment runs successfully, the deployment script resources will be removed by the script service, because the `cleanupPreference` property is set to **OnSuccess**.

Clean up resources

When the Azure resources are no longer needed, clean up the resources you deployed by deleting the resource group.

1. From the Azure portal, select **Resource group** from the left menu.
2. Enter the resource group name in the **Filter by name** field.
3. Select the resource group name.
4. Select **Delete resource group** from the top menu.

Next steps

In this tutorial, you learned how to use a deployment script in ARM templates. To learn how to deploy Azure resources based on conditions, see:

Use conditions

Tutorial: Use condition in ARM templates

Article • 03/09/2023

Learn how to deploy Azure resources based on conditions in an Azure Resource Manager template (ARM template).

In the Set resource deployment order tutorial, you create a virtual machine, a virtual network, and some other dependent resources including a storage account. Instead of creating a new storage account every time, you let people choose between creating a new storage account and using an existing storage account. To accomplish this goal, you define an additional parameter. If the value of the parameter is **new**, a new storage account is created. Otherwise, an existing storage account with the name provided is used.

This tutorial covers the following tasks:

- ✓ Open a Quickstart template
- ✓ Modify the template
- ✓ Deploy the template
- ✓ Clean up resources

This tutorial only covers a basic scenario of using conditions. For more information, see:

- Template file structure: Condition.
- Conditionally deploy a resource in an ARM template.
- Template function: If.
- Comparison functions for ARM templates

For a Learn module that covers conditions, see Manage complex cloud deployments by using advanced ARM template features.

If you don't have an Azure subscription, create a free account before you begin.

Prerequisites

To complete this article, you need:

- Visual Studio Code with Resource Manager Tools extension. See Quickstart: Create ARM templates with Visual Studio Code.
- To increase security, use a generated password for the virtual machine administrator account. You can use Azure Cloud Shell to run the following command in PowerShell or Bash:

  ```shell
  openssl rand -base64 32
  ```

  To learn more, run `man openssl rand` to open the manual page.

  Azure Key Vault is designed to safeguard cryptographic keys and other secrets. For more information, see Tutorial: Integrate Azure Key Vault in ARM template deployment. We also recommend you to update your password every three months.

Open a Quickstart template

Azure Quickstart Templates is a repository for ARM templates. Instead of creating a template from scratch, you can find a sample template and customize it. The template used in this tutorial is called Deploy a simple Windows VM.

1. From Visual Studio Code, select **File** > **Open File**.
2. In **File name**, paste the following URL:

   ```url
   https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.compute/vm-simple-windows/azuredeploy.json
   ```

3. Select **Open** to open the file.

4. There are six resources defined by the template:

   - Microsoft.Storage/storageAccounts.
   - Microsoft.Network/publicIPAddresses.
   - Microsoft.Network/networkSecurityGroups.
   - Microsoft.Network/virtualNetworks.
   - Microsoft.Network/networkInterfaces.
   - Microsoft.Compute/virtualMachines.

   It is helpful to review the template reference before customizing a template.

5. Select **File** > **Save As** to save a copy of the file to your local computer with the name *azuredeploy.json*.

Modify the template

Make two changes to the existing template:

- Add a storage account name parameter. Users can specify either a new storage account name or an existing storage account name.
- Add a new parameter called `newOrExisting`. The deployment uses this parameter to determine whether to create a new storage account or use an existing storage account.

Here is the procedure to make the changes:

1. Open *azuredeploy.json* in Visual Studio Code.

2. Replace the three `variables('storageAccountName')` with `parameters('storageAccountName')` in the whole template.

3. Remove the following variable definition:

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": { ...
  },
  "variables": {
    "storageAccountName": "[concat(uniquestring(resourceGroup().id), 'sawinvm')]",
    "nicName": "myVMNic",
    "addressPrefix": "10.0.0.0/16",
    "subnetName": "Subnet",
    "subnetPrefix": "10.0.0.0/24",
    "publicIPAddressName": "myPublicIP",
    "vmName": "SimpleWinVM",
    "virtualNetworkName": "MyVNET",
    "subnetRef": "[resourceId('Microsoft.Network/virtualNetworks/subnets', variables('virtualNetworkName'), variables('subnetName'))]",
    "networkSecurityGroupName": "default-NSG"
  },
```

4. Add the following two parameters to the beginning of the parameters section:

JSON

```
"storageAccountName": {
  "type": "string"
},
"newOrExisting": {
  "type": "string",
  "allowedValues": [
    "new",
    "existing"
  ]
},
```

Press Alt+Shift+F to format the template in Visual Studio Code.

The updated parameters definition looks like:

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageAccountName": {
      "type": "string"
    },
    "newOrExisting": {
      "type": "string",
      "allowedValues": [
        "new",
        "existing"
      ]
    },
    "adminUsername": {···
    },
    "adminPassword": {···
    },
    "dnsLabelPrefix": {···
    },
    "windowsOSVersion": {···
    },
    "vmSize": {···
    },
    "location": {···
    }
  },
```

5. Add the following line to the beginning of the storage account definition.

JSON

```
"condition": "[equals(parameters('newOrExisting'),'new')]",
```

The condition checks the value of the parameter `newOrExisting`. If the parameter value is **new**, the deployment creates the storage account.

The updated storage account definition looks like:

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {…
  },
  "variables": {…
  },
  "resources": [
    {
      "condition": "[equals(parameters('newOrExisting'),'new')]",
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2018-11-01",
      "name": "[parameters('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "Storage",
      "properties": {
      }
    },
```

6. Update the `storageUri` property of the virtual machine resource definition with the following value:

   JSON

   ```
   "storageUri": "[concat('https://', parameters('storageAccountName'),
   '.blob.core.windows.net')]"
   ```

   This change is necessary when you use an existing storage account under a different resource group.

7. Save the changes.

Deploy the template

1. Sign in to Cloud Shell.

2. Choose your preferred environment by selecting either **PowerShell** or **Bash** (for CLI) on the upper left corner. Restarting the shell is required when you switch.

3. Select **Upload/download files**, and then select **Upload**. See the previous screenshot. Select the file you saved in the previous section. After uploading the file, you can use the `ls` command and the `cat` command to verify the file was uploaded successfully.

4. Run the following PowerShell script to deploy the template.

> **Important**
>
> The storage account name must be unique across Azure. The name must have only lowercase letters or numbers. It can be no longer than 24 characters. The storage account name is the project name with **store** appended. Make sure the project name and the generated storage account name meet the storage account name requirements.

Azure PowerShell

```
$projectName = Read-Host -Prompt "Enter a project name that is used to
generate resource group name and resource names"
$newOrExisting = Read-Host -Prompt "Create new or use existing (Enter
new or existing)"
$location = Read-Host -Prompt "Enter the Azure location (i.e.
centralus)"
$vmAdmin = Read-Host -Prompt "Enter the admin username"
$vmPassword = Read-Host -Prompt "Enter the admin password" -
AsSecureString
$dnsLabelPrefix = Read-Host -Prompt "Enter the DNS Label prefix"

$resourceGroupName = "${projectName}rg"
$storageAccountName = "${projectName}store"

New-AzResourceGroup -Name $resourceGroupName -Location $location
New-AzResourceGroupDeployment `
    -ResourceGroupName $resourceGroupName `
    -adminUsername $vmAdmin `
    -adminPassword $vmPassword `
    -dnsLabelPrefix $dnsLabelPrefix `
    -storageAccountName $storageAccountName `
    -newOrExisting $newOrExisting `
    -TemplateFile "$HOME/azuredeploy.json"

Write-Host "Press [ENTER] to continue ..."
```

> **Note**

> The deployment fails if `newOrExisting` is **new**, but the storage account with the storage account name specified already exists.

Try making another deployment with `newOrExisting` set to **existing** and specify an existing storage account. To create a storage account beforehand, see Create a storage account.

Clean up resources

When the Azure resources are no longer needed, clean up the resources you deployed by deleting the resource group. To delete the resource group, select **Try it** to open the Cloud Shell. To paste the PowerShell script, right-click the shell pane, and then select **Paste**.

Azure PowerShell

```
$projectName = Read-Host -Prompt "Enter the same project name you used in
the last procedure"
$resourceGroupName = "${projectName}rg"

Remove-AzResourceGroup -Name $resourceGroupName

Write-Host "Press [ENTER] to continue ..."
```

Next steps

In this tutorial, you developed a template that allows users to choose between creating a new storage account and using an existing storage account. To learn how to retrieve secrets from Azure Key Vault, and use the secrets as passwords in the template deployment, see:

Integrate Key Vault in template deployment

Tutorial: Integrate Azure Key Vault in your ARM template deployment

Article • 03/09/2023

Learn how to retrieve secrets from an Azure key vault and pass the secrets as parameters when you deploy an Azure Resource Manager template (ARM template). The parameter value is never exposed, because you reference only its key vault ID. You can reference the key vault secret by using a static ID or a dynamic ID. This tutorial uses a static ID. With the static ID approach, you reference the key vault in the template parameter file, not the template file. For more information about both approaches, see Use Azure Key Vault to pass secure parameter value during deployment.

In the Set resource deployment order tutorial, you create a virtual machine (VM). You need to provide the VM administrator username and password. Instead of providing the password, you can pre-store the password in an Azure key vault and then customize the template to retrieve the password from the key vault during the deployment.

This tutorial covers the following tasks:

- ✓ Prepare a key vault
- ✓ Open a quickstart template
- ✓ Edit the parameters file
- ✓ Deploy the template
- ✓ Validate the deployment
- ✓ Clean up resources

If you don't have an Azure subscription, create a free account before you begin.

For a Learn module that uses a secure value from a key vault, see Manage complex cloud deployments by using advanced ARM template features.

Prerequisites

To complete this article, you need:

- Visual Studio Code with Resource Manager Tools extension. See Quickstart: Create ARM templates with Visual Studio Code.
- To increase security, use a generated password for the VM administrator account. You can use Azure Cloud Shell to run the following command in PowerShell or Bash:

  ```shell
  openssl rand -base64 32
  ```

  To learn more, run `man openssl rand` to open the manual page.

  Verify that the generated password meets the VM password requirements. Each Azure service has specific password requirements. For the VM password requirements, see What are the password requirements when you create a VM?.

Prepare a key vault

In this section, you create a key vault and add a secret to it, so that you can retrieve the secret when you deploy your template. There are many ways to create a key vault. In this tutorial, you use Azure PowerShell to deploy an ARM template. This template does two things:

- Creates a key vault with the `enabledForTemplateDeployment` property enabled. This property must be *true* before the template deployment process can access the secrets that are defined in the key vault.
- Adds a secret to the key vault. The secret stores the VM administrator password.

> **Note**
>
> As the user who's deploying the virtual machine template, if you're not the Owner of or a Contributor to the key vault, the Owner or a Contributor must grant you

access to the `Microsoft.KeyVault/vaults/deploy/action` permission for the key vault. For more information, see **Use Azure Key Vault to pass a secure parameter value during deployment**.

To run the following Azure PowerShell script, select **Try it** to open Cloud Shell. To paste the script, right-click the shell pane, and then select **Paste**.

Azure PowerShell

```
$projectName = Read-Host -Prompt "Enter a project name that is used for generating resource names"
$location = Read-Host -Prompt "Enter the location (i.e. centralus)"
$upn = Read-Host -Prompt "Enter your user principal name (email address) used to sign in to Azure"
$secretValue = Read-Host -Prompt "Enter the virtual machine administrator password" -AsSecureString

$resourceGroupName = "${projectName}rg"
$keyVaultName = $projectName
$adUserId = (Get-AzADUser -UserPrincipalName $upn).Id
$templateUri = "https://raw.githubusercontent.com/Azure/azure-docs-json-samples/master/tutorials-use-key-vault/CreateKeyVault.json"

New-AzResourceGroup -Name $resourceGroupName -Location $location
New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName -TemplateUri $templateUri -keyVaultName $keyVaultName -adUserId $adUserId -secretValue $secretValue

Write-Host "Press [ENTER] to continue ..."
```

Important

- The resource group name is the project name, but with **rg** appended to it. To make it easier to **clean up the resources that you created in this tutorial**, use the same project name and resource group name when you **deploy the next template**.
- The default name for the secret is **vmAdminPassword**. It's hardcoded in the template.
- To enable the template to retrieve the secret, you must enable an access policy called **Enable access to Azure Resource Manager for template deployment** for the key vault. This policy is enabled in the template. For more information about the access policy, see **Deploy key vaults and secrets**.

The template has one output value, called `keyVaultId`. You will use this ID along with the secret name to retrieve the secret value later in the tutorial. The resource ID format is:

JSON

```
/subscriptions/<SubscriptionID>/resourceGroups/mykeyvaultdeploymentrg/provid
ers/Microsoft.KeyVault/vaults/<KeyVaultName>
```

When you copy and paste the ID, it might be broken into multiple lines. Merge the lines and trim the extra spaces.

To validate the deployment, run the following PowerShell command in the same shell pane to retrieve the secret in clear text. The command works only in the same shell session, because it uses the variable `$keyVaultName`, which is defined in the preceding PowerShell script.

Azure PowerShell

```
$secret = Get-AzKeyVaultSecret -VaultName $keyVaultName -Name
"vmAdminPassword"
$ssPtr =
[System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($secret.SecretV
alue)
try {
   $secretValueText =
[System.Runtime.InteropServices.Marshal]::PtrToStringBSTR($ssPtr)
} finally {
   [System.Runtime.InteropServices.Marshal]::ZeroFreeBSTR($ssPtr)
}
Write-Output $secretValueText
```

Now you've prepared a key vault and a secret. The following sections show you how to customize an existing template to retrieve the secret during the deployment.

Open a quickstart template

Azure Quickstart Templates is a repository for ARM templates. Instead of creating a template from scratch, you can find a sample template and customize it. The template that's used in this tutorial is called Deploy a simple Windows VM.

1. In Visual Studio Code, select **File** > **Open File**.
2. In the **File name** box, paste the following URL:

```
url
https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.compute/vm-simple-windows/azuredeploy.json
```

3. Select **Open** to open the file. The scenario is the same as the one that's used in Tutorial: Create ARM templates with dependent resources. The template defines six resources:

   - **Microsoft.Storage/storageAccounts**.
   - **Microsoft.Network/publicIPAddresses**.
   - **Microsoft.Network/networkSecurityGroups**.
   - **Microsoft.Network/virtualNetworks**.
   - **Microsoft.Network/networkInterfaces**.
   - **Microsoft.Compute/virtualMachines**.

   It's helpful to have some basic understanding of the template before you customize it.

4. Select **File** > **Save As**, and then save a copy of the file to your local computer with the name *azuredeploy.json*.

5. Repeat steps 1-3 to open the following URL, and then save the file as *azuredeploy.parameters.json*.

```
url
https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.compute/vm-simple-windows/azuredeploy.parameters.json
```

Edit the parameters file

By using the static ID method, you don't need to make any changes to the template file. Retrieving the secret value is done by configuring the template parameter file.

1. In Visual Studio Code, open *azuredeploy.parameters.json* if it's not already open.

2. Update the `adminPassword` parameter to:

```
JSON
```

```
"adminPassword": {
    "reference": {
        "keyVault": {
            "id":
"/subscriptions/<SubscriptionID>/resourceGroups/mykeyvaultdeploymentrg/
providers/Microsoft.KeyVault/vaults/<KeyVaultName>"
        },
        "secretName": "vmAdminPassword"
    }
},
```

> **Important**
>
> Replace the value for `id` with the resource ID of the key vault that you created in the previous procedure. The `secretName` is hardcoded as **vmAdminPassword**. See **Prepare a key vault**.

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "adminUsername": {
      "value": "johndole"
    },
    "adminPassword": {
      "reference": {
        "keyVault": {
          "id": "/subscriptions/<SubscriptionID>/resourceGroups/mykeyvaulttutorialrg/providers/Microsoft.KeyVault/vaults/<KeyVaultName>"
        },
        "secretName": "vmAdminPassword"
      }
    },
    "dnsLabelPrefix": {
      "value": "johndolevm"
    }
  }
}
```

3. Update the following values:

   - `adminUsername`: The name of the virtual machine administrator account.
   - `dnsLabelPrefix`: Name the `dnsLabelPrefix` value.

   For examples of names, see the preceding image.

4. Save the changes.

Deploy the template

1. Sign in to Cloud Shell.

2. Choose your preferred environment by selecting either **PowerShell** or **Bash** (for CLI) on the upper left corner. Restarting the shell is required when you switch.

3. Select **Upload/download files**, and then select **Upload**. Upload both *azuredeploy.json* and *azuredeploy.parameters.json* to Cloud Shell. After uploading the file, you can use the `ls` command and the `cat` command to verify the file is uploaded successfully.

4. Run the following PowerShell script to deploy the template.

Azure PowerShell

```
$projectName = Read-Host -Prompt "Enter the same project name that is
used for creating the key vault"
$location = Read-Host -Prompt "Enter the same location that is used for
creating the key vault (i.e. centralus)"
$resourceGroupName = "${projectName}rg"

New-AzResourceGroupDeployment `
    -ResourceGroupName $resourceGroupName `
    -TemplateFile "$HOME/azuredeploy.json" `
    -TemplateParameterFile "$HOME/azuredeploy.parameters.json"

Write-Host "Press [ENTER] to continue ..."
```

When you deploy the template, use the same resource group that you used in the key vault. This approach makes it easier for you to clean up the resources, because you need to delete only one resource group instead of two.

Validate the deployment

After you've successfully deployed the virtual machine, test the sign-in credentials by using the password that's stored in the key vault.

1. Open the Azure portal.

2. Select **Resource groups** > <*YourResourceGroupName*> > **simpleWinVM**.

3. Select **Connect** at the top.

4. Select **Download RDP File**, and then follow the instructions to sign in to the virtual machine by using the password that's stored in the key vault.

Clean up resources

When you no longer need your Azure resources, clean up the resources that you deployed by deleting the resource group.

Azure PowerShell

```
$projectName = Read-Host -Prompt "Enter the same project name that is used
for creating the key vault"
$resourceGroupName = "${projectName}rg"

Remove-AzResourceGroup -Name $resourceGroupName

Write-Host "Press [ENTER] to continue ..."
```

Next steps

In this tutorial, you retrieved a secret from your Azure key vault. You then used the secret in your template deployment. To learn how to use virtual machine extensions to perform post deployment tasks, see:

Deploy virtual machine extensions

Tutorial: Deploy virtual machine extensions with ARM templates

Article • 03/09/2023

Learn how to use Azure virtual machine extensions to perform post-deployment configuration and automation tasks on Azure VMs. Many different VM extensions are available for use with Azure VMs. In this tutorial, you deploy a Custom Script extension from an Azure Resource Manager template (ARM template) to run a PowerShell script on a Windows VM. The script installs Web Server on the VM.

This tutorial covers the following tasks:

- ✓ Prepare a PowerShell script
- ✓ Open a quickstart template
- ✓ Edit the template
- ✓ Deploy the template

If you don't have an Azure subscription, create a free account before you begin.

Prerequisites

To complete this article, you need:

- Visual Studio Code with Resource Manager Tools extension. See Quickstart: Create ARM templates with Visual Studio Code.
- To increase security, use a generated password for the virtual machine administrator account. You can use Azure Cloud Shell to run the following command in PowerShell or Bash:

  ```shell
  openssl rand -base64 32
  ```

  To learn more, run `man openssl rand` to open the manual page.

  Azure Key Vault is designed to safeguard cryptographic keys and other secrets. For more information, see Tutorial: Integrate Azure Key Vault in your ARM template deployment. We also recommend that you update your password every three months.

Prepare a PowerShell script

You can use an inline PowerShell script or a script file. This tutorial shows how to use a script file. A PowerShell script with the following content is shared from GitHub:

Azure PowerShell

```
Install-WindowsFeature -Name Web-Server -IncludeManagementTools
```

If you choose to publish the file to your own location, update the `fileUri` element in the template later in the tutorial.

Open a quickstart template

Azure Quickstart Templates is a repository for ARM templates. Instead of creating a template from scratch, you can find a sample template and customize it. The template used in this tutorial is called Deploy a simple Windows VM.

1. In Visual Studio Code, select **File** > **Open File**.
2. In the **File name** box, paste the following URL:

   url

   ```
   https://raw.githubusercontent.com/Azure/azure-quickstart-
   templates/master/quickstarts/microsoft.compute/vm-simple-
   windows/azuredeploy.json
   ```

3. To open the file, select **Open**. The template defines five resources:

   - **Microsoft.Storage/storageAccounts**.
   - **Microsoft.Network/publicIPAddresses**.
   - **Microsoft.Network/networkSecurityGroups**.
   - **Microsoft.Network/virtualNetworks**.
   - **Microsoft.Network/networkInterfaces**.
   - **Microsoft.Compute/virtualMachines**.

   It's helpful to get some basic understanding of the template before you customize it.

4. Save a copy of the file to your local computer with the name *azuredeploy.json* by selecting **File** > **Save As**.

Edit the template

Add a virtual machine extension resource to the existing template with the following content:

JSON

```
{
  "type": "Microsoft.Compute/virtualMachines/extensions",
  "apiVersion": "2021-04-01",
  "name": "[concat(variables('vmName'),'/', 'InstallWebServer')]",
  "location": "[parameters('location')]",
  "dependsOn": [
    "[concat('Microsoft.Compute/virtualMachines/',variables('vmName'))]"
  ],
  "properties": {
    "publisher": "Microsoft.Compute",
    "type": "CustomScriptExtension",
    "typeHandlerVersion": "1.7",
    "autoUpgradeMinorVersion": true,
    "settings": {
      "fileUris": [
        "https://raw.githubusercontent.com/Azure/azure-docs-json-
samples/master/tutorial-vm-extension/installWebServer.ps1"
      ],
      "commandToExecute": "powershell.exe -ExecutionPolicy Unrestricted -
File installWebServer.ps1"
    }
  }
}
```

For more information about this resource definition, see the extension reference. The following are some important elements:

- `name`: Because the extension resource is a child resource of the virtual machine object, the name must have the virtual machine name prefix. See Set name and type for child resources.
- `dependsOn`: Create the extension resource after you've created the virtual machine.
- `fileUris`: The locations where the script files are stored. If you choose not to use the provided location, you need to update the values.
- `commandToExecute`: This command invokes the script.

To use an inline script, remove `fileUris`, and update `commandToExecute` to:

PowerShell

```
powershell.exe Install-WindowsFeature -name Web-Server -
IncludeManagementTools && powershell.exe remove-item
'C:\\inetpub\\wwwroot\\iisstart.htm' && powershell.exe Add-Content -Path
'C:\\inetpub\\wwwroot\\iisstart.htm' -Value $('Hello World from ' +
$env:computername)
```

This inline script also updates the *iisstart.html* content.

You must also open the HTTP port so that you can access the web server.

1. Find `securityRules` in the template.

2. Add the following rule next to **default-allow-3389**.

JSON

```
{
  "name": "AllowHTTPInBound",
  "properties": {
    "priority": 1010,
    "access": "Allow",
    "direction": "Inbound",
    "destinationPortRange": "80",
    "protocol": "Tcp",
    "sourcePortRange": "*",
    "sourceAddressPrefix": "*",
    "destinationAddressPrefix": "*"
  }
}
```

Deploy the template

For the deployment procedure, see the **Deploy the template** section of Tutorial: Create ARM templates with dependent resources. We recommended that you use a generated password for the virtual machine administrator account. See this article's Prerequisites section.

From the Cloud Shell, run the following command to retrieve the public IP address of the VM:

Azure PowerShell

```
(Get-AzPublicIpAddress -ResourceGroupName $resourceGroupName).IpAddress
```

Paste the IP address into a Web browser. The default Internet Information Services (IIS) welcome page opens:

Clean up resources

When you no longer need the Azure resources you deployed, clean them up by deleting the resource group.

1. In the Azure portal, in the left pane, select **Resource group**.
2. In the **Filter by name** box, enter the resource group name.
3. Select the resource group name. Six resources are displayed in the resource group.
4. In the top menu, select **Delete resource group**.

Next steps

In this tutorial, you deployed a virtual machine and a virtual machine extension. The extension installed the IIS web server on the virtual machine. To learn how to use the Azure SQL Database extension to import a BACPAC file, see:

Deploy SQL extensions

Tutorial: Import SQL BACPAC files with ARM templates

Article • 04/09/2023

Learn how to use Azure SQL Database extensions to import a BACPAC file with Azure Resource Manager templates (ARM templates). Deployment artifacts are any files, in addition to the main template files, that are needed to complete a deployment. The BACPAC file is an artifact.

In this tutorial, you create a template to deploy a logical SQL server and a single database and import a BACPAC file. For information about how to deploy Azure virtual machine extensions by using ARM templates, see Tutorial: Deploy virtual machine extensions with ARM templates.

This tutorial covers the following tasks:

- ✓ Prepare a BACPAC file.
- ✓ Open a quickstart template.
- ✓ Edit the template.
- ✓ Deploy the template.
- ✓ Verify the deployment.

If you don't have an Azure subscription, create a free account before you begin.

Prerequisites

To complete this article, you need:

- Visual Studio Code with the Resource Manager Tools extension. See Quickstart: Create ARM templates with Visual Studio Code.
- To increase security, use a generated password for the server administrator account. You can use Azure Cloud Shell to run the following command in PowerShell or Bash:

  ```shell
  openssl rand -base64 32
  ```

  To learn more, run `man openssl rand` to open the manual page.

Azure Key Vault is designed to safeguard cryptographic keys and other secrets. For more information, see Tutorial: Integrate Azure Key Vault in ARM template deployment. We also recommend you to update your password every three months.

Prepare a BACPAC file

A BACPAC file is shared in GitHub. To create your own, see Export a database from Azure SQL Database to a BACPAC file. If you choose to publish the file to your own location, you must update the template later in the tutorial.

The BACPAC file must be stored in an Azure Storage account before it can be imported with an ARM template. The following PowerShell script prepares the BACPAC file with these steps:

- Download the BACPAC file.
- Create an Azure Storage account.
- Create a storage account blob container.
- Upload the BACPAC file to the container.
- Display the storage account key, blob URL, resource group name, and location.

1. Select **Try It** to open Cloud Shell. Then copy and paste the following PowerShell script into the shell window.

Azure PowerShell

```
$projectName = Read-Host -Prompt "Enter a project name that is used to
generate Azure resource names"
$location = Read-Host -Prompt "Enter the location (i.e. centralus)"

$resourceGroupName = "${projectName}rg"
$storageAccountName = "${projectName}store"
$containerName = "bacpacfiles"
$bacpacFileName = "SQLDatabaseExtension.bacpac"
$bacpacUrl = "https://github.com/Azure/azure-docs-json-
samples/raw/master/tutorial-sql-extension/SQLDatabaseExtension.bacpac"

# Download the bacpac file
Invoke-WebRequest -Uri $bacpacUrl -OutFile "$HOME/$bacpacFileName"

# Create a resource group
New-AzResourceGroup -Name $resourceGroupName -Location $location

# Create a storage account
$storageAccount = New-AzStorageAccount -ResourceGroupName
$resourceGroupName `
                                        -Name $storageAccountName `
```

```
                                        -SkuName Standard_LRS `
                                        -Location $location
$storageAccountKey = (Get-AzStorageAccountKey -ResourceGroupName
$resourceGroupName `
                                                  -Name
$storageAccountName).Value[0]

# Create a container
New-AzStorageContainer -Name $containerName -Context
$storageAccount.Context

# Upload the BACPAC file to the container
Set-AzStorageBlobContent -File $HOME/$bacpacFileName `
                         -Container $containerName `
                         -Blob $bacpacFileName `
                         -Context $storageAccount.Context

Write-Host "The project name:        $projectName`
    The location:            $location`
    The storage account key: $storageAccountKey`
    The BACPAC file URL:
https://$storageAccountName.blob.core.windows.net/$containerName/$bacpa
cFileName`
    "
Write-Host "Press [ENTER] to continue ..."
```

2. Save the storage account key, BACPAC file URL, project name, and location. You'll use those values when you deploy the template later in this tutorial.

Open a quickstart template

The template used in this tutorial is stored in GitHub.

1. From Visual Studio Code, select **File** > **Open File**.

2. In **File name**, paste the following URL:

```
url

https://raw.githubusercontent.com/Azure/azure-docs-json-
samples/master/tutorial-sql-extension/azuredeploy.json
```

3. Select **Open** to open the file.

   There are two resources defined in the template:

   - `Microsoft.Sql/servers`. See the template reference.

   - `Microsoft.SQL.servers/databases`. See the template reference.

It's helpful to get some basic understanding of the template before you customize it.

4. Select **File** > **Save As** to save a copy of the file to your local computer with the name *azuredeploy.json*.

Edit the template

1. Add two parameters at the end of the `parameters` section to set the storage account key and the BACPAC URL.

   JSON

   ```
   "storageAccountKey": {
     "type":"string",
     "metadata":{
       "description": "Specifies the key of the storage account where
   the BACPAC file is stored."
     }
   },
   "bacpacUrl": {
     "type":"string",
     "metadata":{
       "description": "Specifies the URL of the BACPAC file."
     }
   }
   ```

   Add a comma after the `adminPassword` property's closing curly brace (`}`). To format the JSON file from Visual Studio Code, select **Shift+Alt+F**.

2. Add two resources to the template.

   - To allow the SQL Database extension to import BACPAC files, you must allow traffic from Azure services. When the SQL server is deployed, the firewall rule turns on the setting for **Allow Azure services and resources to access this server**.

     Add the following firewall rule under the server definition:

     JSON

     ```
     "resources": [
       {
         "type": "firewallrules",
         "apiVersion": "2021-02-01-preview",
         "name": "AllowAllAzureIps",
         "location": "[parameters('location')]",
     ```

```
      "dependsOn": [
        "[parameters('databaseServerName')]"
      ],
      "properties": {
        "startIpAddress": "0.0.0.0",
        "endIpAddress": "0.0.0.0"
      }
    }
  ]
```

The following example shows the updated template:

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {…
  },
  "resources": [
    {
      "type": "Microsoft.Sql/servers",
      "apiVersion": "2021-02-01-preview",
      "name": "[parameters('databaseServerName')]",
      "location": "[parameters('location')]",
      "properties": {…
      },
      "resources": [
        {
          "type": "firewallrules",
          "apiVersion": "2021-02-01-preview",
          "name": "AllowAllAzureIps",
          "location": "[parameters('location')]",
          "dependsOn": [
            "[parameters('databaseServerName')]"
          ],
          "properties": {
            "startIpAddress": "0.0.0.0",
            "endIpAddress": "0.0.0.0"
          }
        }
      ]
    },
```

- Add a SQL Database extension resource to the database definition with the following JSON:

JSON

```
"resources": [
  {
    "type": "extensions",
    "apiVersion": "2014-04-01",
    "name": "Import",
    "dependsOn": [
      "[resourceId('Microsoft.Sql/servers/databases',
parameters('databaseServerName'), parameters('databaseName'))]"
    ],
    "properties": {
      "storageKeyType": "StorageAccessKey",
      "storageKey": "[parameters('storageAccountKey')]",
      "storageUri": "[parameters('bacpacUrl')]",
      "administratorLogin": "[parameters('adminUser')]",
```

```
        "administratorLoginPassword": "
[parameters('adminPassword')]",
        "operationMode": "Import"
      }
    }
  ]
```

The following example shows the updated template:

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {…
  },
  "resources": [
    {…
    },
    {
      "type": "Microsoft.Sql/servers/databases",
      "apiVersion": "2021-02-01-preview",
      "name": "[concat(string(parameters('databaseServerName')), '/', string(parameters('databaseName')))]",
      "location": "[parameters('location')]",
      "dependsOn": [
        "[concat('Microsoft.Sql/servers/', parameters('databaseServerName'))]"
      ],
      "resources": [
        {
          "type": "extensions",
          "apiVersion": "2014-04-01",
          "name": "Import",
          "dependsOn": [
            "[resourceId('Microsoft.Sql/servers/databases', parameters('databaseServerName'), parameters('databaseName'))]"
          ],
          "properties": {
            "storageKeyType": "StorageAccessKey",
            "storageKey": "[parameters('storageAccountKey')]",
            "storageUri": "[parameters('bacpacUrl')]",
            "administratorLogin": "[parameters('adminUser')]",
            "administratorLoginPassword": "[parameters('adminPassword')]",
            "operationMode": "Import"
          }
        }
      ]
    }
```

To understand the resource definition, see the API version's SQL Database extension reference. The following are some important elements:

- `dependsOn`: The extension resource must be created after the database has been created.
- `storageKeyType`: Specify the type of the storage key to use. The value can be either `StorageAccessKey` or `SharedAccessKey`. Use `StorageAccessKey` in this tutorial.
- `storageKey`: Specify the key for the storage account where the BACPAC file is stored. If the storage key type is `SharedAccessKey`, it must be preceded with a "?".
- `storageUri`: Specify the URL of the BACPAC file stored in a storage account.
- `administratorLogin`: The SQL administrator's account name.
- `administratorLoginPassword`: The SQL administrator's password. To use a generated password, see Prerequisites.

The following example shows the completed template:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "databaseServerName": {
      "type": "string",
      "defaultValue": "[concat('server-', uniqueString(resourceGroup().id,
deployment().name))]",
      "metadata": {
        "description": "Specifies the name for the SQL server"
      }
    },
    "databaseName": {
      "type": "string",
      "defaultValue": "[concat('db-', uniqueString(resourceGroup().id,
deployment().name), '-1')]",
      "metadata": {
        "description": "Specifies the name for the SQL database under the
SQL server"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Specifies the location for server and database"
      }
    },
    "adminUser": {
      "type": "string",
      "metadata": {
        "description": "Specifies the username for admin"
      }
    },
    "adminPassword": {
      "type": "securestring",
      "metadata": {
        "description": "Specifies the password for admin"
      }
    },
```

```
    "storageAccountKey": {
      "type": "string",
      "metadata": {
        "description": "Specifies the key of the storage account where the
BACPAC file is stored."
      }
    },
    "bacpacUrl": {
      "type": "string",
      "metadata": {
        "description": "Specifies the URL of the BACPAC file."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Sql/servers",
      "apiVersion": "2021-02-01-preview",
      "name": "[parameters('databaseServerName')]",
      "location": "[parameters('location')]",
      "properties": {
        "administratorLogin": "[parameters('adminUser')]",
        "administratorLoginPassword": "[parameters('adminPassword')]",
        "version": "12.0"
      },
      "resources": [
        {
          "type": "firewallrules",
          "apiVersion": "2021-02-01-preview",
          "name": "AllowAllAzureIps",
          "location": "[parameters('location')]",
          "dependsOn": [
            "[parameters('databaseServerName')]"
          ],
          "properties": {
            "startIpAddress": "0.0.0.0",
            "endIpAddress": "0.0.0.0"
          }
        }
      ]
    },
    {
      "type": "Microsoft.Sql/servers/databases",
      "apiVersion": "2021-02-01-preview",
      "name": "[concat(string(parameters('databaseServerName')), '/',
string(parameters('databaseName')))]",
      "location": "[parameters('location')]",
      "dependsOn": [
        "[concat('Microsoft.Sql/servers/',
parameters('databaseServerName'))]"
      ],
```

```
      "resources": [
        {
          "type": "extensions",
          "apiVersion": "2014-04-01",
          "name": "Import",
          "dependsOn": [
            "[resourceId('Microsoft.Sql/servers/databases',
parameters('databaseServerName'), parameters('databaseName'))]"
          ],
          "properties": {
            "storageKeyType": "StorageAccessKey",
            "storageKey": "[parameters('storageAccountKey')]",
            "storageUri": "[parameters('bacpacUrl')]",
            "administratorLogin": "[parameters('adminUser')]",
            "administratorLoginPassword": "[parameters('adminPassword')]",
            "operationMode": "Import"
          }
        }
      ]
    }
  ]
}
```

Deploy the template

Use the project name and location that were used when you prepared the BACPAC file. That puts all resources in the same resource group, which is helpful when you delete resources.

1. Sign in to Cloud Shell.

2. Select **PowerShell** from the upper left corner.

3. Select **Upload/Download files** and upload your *azuredeploy.json* file.

4. To deploy the template, copy and paste the following script into the shell window.

Azure PowerShell

```
$projectName = Read-Host -Prompt "Enter the same project name that is
used earlier"
$adminUsername = Read-Host -Prompt "Enter the SQL admin username"
$adminPassword = Read-Host -Prompt "Enter the admin password" -
```

```
AsSecureString
$storageAccountKey = Read-Host -Prompt "Enter the storage account key"
$bacpacUrl = Read-Host -Prompt "Enter the URL of the BACPAC file"
$resourceGroupName = "${projectName}rg"

New-AzResourceGroupDeployment `
    -ResourceGroupName $resourceGroupName `
    -adminUser $adminUsername `
    -adminPassword $adminPassword `
    -TemplateFile "$HOME/azuredeploy.json" `
    -storageAccountKey $storageAccountKey `
    -bacpacUrl $bacpacUrl

Write-Host "Press [ENTER] to continue ..."
```

Verify the deployment

To access the server from your client computer, you need to add a firewall rule. Your client's IP address and the IP address that's used to connect to the server might be different because of network address translation (NAT). For more information, see Create and manage IP firewall rules.

For example, when you sign in to **Query editor** a message is displayed that the IP address isn't allowed. The address is different from your client's IP address because of NAT. Select the message's link to add a firewall rule for the IP address. When you're finished, you can delete the IP address from the server's **Firewalls and virtual networks** settings.

In the Azure portal, from the resource group select the database. Select **Query editor (preview)**, and enter the administrator credentials. You'll see two tables were imported into the database.

Clean up resources

When the Azure resources you deployed are no longer needed, delete the resource group. The resource group, storage account, SQL server, and SQL databases are deleted.

1. In the Azure portal, enter **Resource groups** in the search box.
2. In the **Filter by name** field, enter the resource group name.
3. Select the resource group name.
4. Select **Delete resource group**.
5. To confirm the deletion, enter the resource group name and select **Delete**.

Next steps

In this tutorial, you deployed a server and a database and imported a BACPAC file. To learn how to troubleshoot template deployment, see:

Troubleshoot ARM template deployments

Quickstart: Create Azure Advisor alerts on new recommendations using an ARM template

Article • 03/09/2023

This article shows you how to set up an alert for new recommendations from Azure Advisor using an Azure Resource Manager template (ARM template).

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

Whenever Azure Advisor detects a new recommendation for one of your resources, an event is stored in Azure Activity log. You can set up alerts for these events from Azure Advisor using a recommendation-specific alerts creation experience. You can select a subscription and optionally a resource group to specify the resources that you want to receive alerts on.

You can also determine the types of recommendations by using these properties:

- Category
- Impact level
- Recommendation type

You can also configure the action that will take place when an alert is triggered by:

- Selecting an existing action group
- Creating a new action group

To learn more about action groups, see Create and manage action groups.

> **Note**
>
> Advisor alerts are currently only available for High Availability, Performance, and Cost recommendations. Security recommendations are not supported.

Prerequisites

- If you don't have an Azure subscription, create a free account before you begin.

- To run the commands from your local computer, install Azure CLI or the Azure PowerShell modules. For more information, see Install the Azure CLI and Install Azure PowerShell.

Review the template

The following template creates an action group with an email target and enables all service health notifications for the target subscription. Save this template as *CreateAdvisorAlert.json*.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "actionGroups_name": {
      "defaultValue": "advisorAlert",
      "type": "string"
    },
    "activityLogAlerts_name": {
      "defaultValue": "AdvisorAlertsTest",
      "type": "string"
    },
    "emailAddress": {
      "defaultValue": "<email address>",
      "type": "string"
    }
  },
  "variables": {
    "alertScope": "[concat('/','subscriptions','/',subscription().subscriptionId)]"
  },
  "resources": [
    {
      "comments": "Action Group",
      "type": "microsoft.insights/actionGroups",
      "apiVersion": "2019-06-01",
      "name": "[parameters('actionGroups_name')]",
      "location": "Global",
      "scale": null,
      "dependsOn": [],
      "tags": {},
      "properties": {
        "groupShortName": "[parameters('actionGroups_name')]",
        "enabled": true,
        "emailReceivers": [
          {
            "name": "[parameters('actionGroups_name')]",
            "emailAddress": "[parameters('emailAddress')]"
```

```
          }
        ],
        "smsReceivers": [],
        "webhookReceivers": []
      }
    },
    {
      "comments": "Azure Advisor Activity Log Alert",
      "type": "microsoft.insights/activityLogAlerts",
      "apiVersion": "2017-04-01",
      "name": "[parameters('activityLogAlerts_name')]",
      "location": "Global",
      "scale": null,
      "tags": {},
      "properties": {
        "scopes": [
          "[variables('alertScope')]"
        ],
        "condition": {
          "allOf": [
            {
              "field": "category",
              "equals": "Recommendation"
            },
            {
              "field": "properties.recommendationCategory",
              "equals": "Cost"
            },
            {
              "field": "properties.recommendationImpact",
              "equals": "Medium"
            },
            {
              "field": "operationName",
              "equals": "Microsoft.Advisor/recommendations/available/action"
            }
          ]
        },
        "actions": {
          "actionGroups": [
            {
              "actionGroupId": "
[resourceId('microsoft.insights/actionGroups',
parameters('actionGroups_name'))]",
              "webhookProperties": {}
            }
          ]
        },
        "enabled": true,
        "description": ""
      },
      "dependsOn": [
        "[resourceId('microsoft.insights/actionGroups',
parameters('actionGroups_name'))]"
      ]
```

```
    }
  ]
}
```

The template defines two resources:

- Microsoft.Insights/actionGroups
- Microsoft.Insights/activityLogAlerts

Deploy the template

Deploy the template using any standard method for deploying an ARM template such as the following examples using CLI and PowerShell. Replace the sample values for **Resource Group**, and **emailAddress** with appropriate values for your environment. The workspace name must be unique among all Azure subscriptions.

CLI

Azure CLI

```
az login
az deployment group create --name CreateAdvisorAlert --resource-group
my-resource-group --template-file CreateAdvisorAlert.json --parameters
emailAddress='user@contoso.com'
```

Validate the deployment

Verify that the workspace has been created using one of the following commands. Replace the sample values for **Resource Group** with the value you used above.

CLI

Azure CLI

```
az monitor activity-log alert show --resource-group my-resource-group --
name AdvisorAlertsTest
```

Clean up resources

If you plan to continue working with subsequent quickstarts and tutorials, you might want to leave these resources in place. When no longer needed, delete the resource group, which deletes the alert rule and the related resources. To delete the resource group by using Azure CLI or Azure PowerShell

CLI

Azure CLI

```
az group delete --name my-resource-group
```

Next steps

- Get an overview of activity log alerts, and learn how to receive alerts.
- Learn more about action groups.

Quickstart: Deploy Cognitive Search using an Azure Resource Manager template

Article • 04/13/2023

This article walks you through the process for using an Azure Resource Manager (ARM) template to deploy an Azure Cognitive Search resource in the Azure portal.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

Only those properties included in the template are used in the deployment. If more customization is required, such as setting up network security, you can update the service as a post-deployment task. To customize an existing service with the fewest steps, use Azure CLI or Azure PowerShell. If you're evaluating preview features, use the Management REST API.

Assuming your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
```

```
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "11257266040777038564"
    }
  },
  "parameters": {
    "name": {
      "type": "string",
      "maxLength": 60,
      "minLength": 2,
      "metadata": {
        "description": "Service name must only contain lowercase letters,
digits or dashes, cannot use dash as the first two or last one characters,
cannot contain consecutive dashes, and is limited between 2 and 60
characters in length."
      }
    },
    "sku": {
      "type": "string",
      "defaultValue": "standard",
      "metadata": {
        "description": "The pricing tier of the search service you want to
create (for example, basic or standard)."
      },
      "allowedValues": [
        "free",
        "basic",
        "standard",
        "standard2",
        "standard3",
        "storage_optimized_l1",
        "storage_optimized_l2"
      ]
    },
    "replicaCount": {
      "type": "int",
      "defaultValue": 1,
      "maxValue": 12,
      "minValue": 1,
      "metadata": {
        "description": "Replicas distribute search workloads across the
service. You need at least two replicas to support high availability of
query workloads (not applicable to the free tier)."
      }
    },
    "partitionCount": {
      "type": "int",
      "defaultValue": 1,
      "allowedValues": [
        1,
        2,
        3,
        4,
        6,
```

```
        12
      ],
      "metadata": {
        "description": "Partitions allow for scaling of document count as
well as faster indexing by sharding your index over multiple search units."
      }
    },
    "hostingMode": {
      "type": "string",
      "defaultValue": "default",
      "allowedValues": [
        "default",
        "highDensity"
      ],
      "metadata": {
        "description": "Applicable only for SKUs set to standard3. You can
set this property to enable a single, high density partition that allows up
to 1000 indexes, which is much higher than the maximum indexes allowed for
any other SKU."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Search/searchServices",
      "apiVersion": "2020-08-01",
      "name": "[parameters('name')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('sku')]"
      },
      "properties": {
        "replicaCount": "[parameters('replicaCount')]",
        "partitionCount": "[parameters('partitionCount')]",
        "hostingMode": "[parameters('hostingMode')]"
      }
    }
  ]
}
```

The Azure resource defined in this template:

- Microsoft.Search/searchServices: create an Azure Cognitive Search service

Deploy the template

Select the following image to sign in to Azure and open a template. The template creates an Azure Cognitive Search resource.

Deploy to Azure

The portal displays a form that allows you to easily provide parameter values. Some parameters are pre-filled with the default values from the template. You will need to provide your subscription, resource group, location, and service name. If you want to use Cognitive Services in an AI enrichment pipeline, for example to analyze binary image files for text, choose a location that offers both Cognitive Search and Cognitive Services. Both services are required to be in the same region for AI enrichment workloads. Once you have completed the form, you will need to agree to the terms and conditions and then select the purchase button to complete your deployment.

Review deployed resources

When your deployment is complete you can access your new resource group and new search service in the portal.

Clean up resources

Other Cognitive Search quickstarts and tutorials build upon this quickstart. If you plan to continue on to work with subsequent quickstarts and tutorials, you may wish to leave this resource in place. When no longer needed, you can delete the resource group, which deletes the Cognitive Search service and related resources.

Next steps

In this quickstart, you created a Cognitive Search service using an ARM template, and validated the deployment. To learn more about Cognitive Search and Azure Resource Manager, continue on to the articles below.

- Read an overview of Azure Cognitive Search.
- Create an index for your search service.
- Create a demo app using the portal wizard.
- Create a skillset to extract information from your data.

Quickstart: Create a Cognitive Services resource using an ARM template

Article • 09/11/2022

This quickstart describes how to use an Azure Resource Manager template (ARM template) to create Cognitive Services.

Azure Cognitive Services are cloud-based artificial intelligence (AI) services that help developers build cognitive intelligence into applications without having direct AI or data science skills or knowledge. They are available through REST APIs and client library SDKs in popular development languages. Azure Cognitive Services enables developers to easily add cognitive features into their applications with cognitive solutions that can see, hear, speak, and analyze.

Create a resource using an Azure Resource Manager template (ARM template). This multi-service resource lets you:

- Access multiple Azure Cognitive Services with a single key and endpoint.
- Consolidate billing from the services you use.
- You must create your first Face, Language service, or Computer Vision resources from the Azure portal to review and acknowledge the terms and conditions. You can do so here: Face, Language service, Computer Vision. After that, you can create subsequent resources using any deployment tool (SDK, CLI, or ARM template, etc) under the same Azure subscription.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

Prerequisites

- If you don't have an Azure subscription, create one for free.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.15.31.15270",
      "templateHash": "17000396405303018145"
    }
  },
  "parameters": {
    "cognitiveServiceName": {
      "type": "string",
      "defaultValue": "[format('CognitiveService-{0}', uniqueString(resourceGroup().id))]",
      "metadata": {
        "description": "That name is the name of our application. It has to be unique.Type a name followed by your resource group name. (<name>-<resourceGroupName>)"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "sku": {
      "type": "string",
      "defaultValue": "S0",
      "allowedValues": [
        "S0"
      ]
    }
  },
  "resources": [
    {
      "type": "Microsoft.CognitiveServices/accounts",
      "apiVersion": "2021-10-01",
      "name": "[parameters('cognitiveServiceName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('sku')]"
      },
      "kind": "CognitiveServices",
      "properties": {
        "apiProperties": {
          "statisticsEnabled": false
        }
      }
    }
```

```
  ]
}
```

One Azure resource is defined in the Bicep file: Microsoft.CognitiveServices/accounts specifies that it is a Cognitive Services resource. The `kind` field in the Bicep file defines the type of resource.

See the list of SKUs and pricing information below.

Multi-service

Service	Kind
Multiple services. For more information, see the pricing page.	`CognitiveServices`

Vision

Service	Kind
Computer Vision	`ComputerVision`
Custom Vision - Prediction	`CustomVision.Prediction`
Custom Vision - Training	`CustomVision.Training`
Face	`Face`
Form Recognizer	`FormRecognizer`

Speech

Service	Kind
Speech Services	`SpeechServices`

Language

Service	Kind
LUIS	`LUIS`
QnA Maker	`QnAMaker`

Service	Kind
Language service	`TextAnalytics`
Text Translation	`TextTranslation`

Decision

Service	Kind
Anomaly Detector	`AnomalyDetector`
Content Moderator	`ContentModerator`
Personalizer	`Personalizer`

Azure OpenAI

Service	Kind
Azure OpenAI	`OpenAI`

Pricing tiers and billing

Pricing tiers (and the amount you get billed) are based on the number of transactions you send using your authentication information. Each pricing tier specifies the:

- maximum number of allowed transactions per second (TPS).
- service features enabled within the pricing tier.
- cost for a predefined number of transactions. Going above this number will cause an extra charge as specified in the pricing details for your service.

> **Note**
>
> Many of the Cognitive Services have a free tier you can use to try the service. To use the free tier, use `F0` as the SKU for your resource.

Deploy the template

Azure portal

1. Click the **Deploy to Azure** button.

   Deploy to Azure

2. Enter the following values.

Value	Description
Subscription	Select an Azure subscription.
Resource group	Select **Create new**, enter a unique name for the resource group, and then click **OK**.
Region	Select a region. For example, **East US**
Cognitive Service Name	Replace with a unique name for your resource. You will need the name in the next section when you validate the deployment.
Location	Replace with the region used above.
Sku	The pricing tier for your resource.

3. Select **Review + Create**, then **Create**. After the resource has successfully finished deploying, the **Go to resource** button will be highlighted.

> **Tip**
>
> If your subscription doesn't allow you to create a Cognitive Service resource, you may need to enable the privilege of that **Azure resource provider** using the **Azure portal**, **PowerShell command** or an **Azure CLI command**. If you are not the subscription owner, ask the *Subscription Owner* or someone with a role of *admin* to complete the registration for you or ask for the **/register/action** privileges to be granted to your account.

Review deployed resources

Portal

When your deployment finishes, you will be able to click the **Go to resource** button to see your new resource. You can also find the resource group by:

1. Selecting **Resource groups** from the left navigation menu.
2. Selecting the resource group name.

Clean up resources

If you want to clean up and remove a Cognitive Services subscription, you can delete the resource or resource group. Deleting the resource group also deletes any other resources contained in the group.

Azure portal

1. In the Azure portal, expand the menu on the left side to open the menu of services, and choose **Resource Groups** to display the list of your resource groups.
2. Locate the resource group containing the resource to be deleted
3. Right-click on the resource group listing. Select **Delete resource group**, and confirm.

If you need to recover a deleted resource, see Recover deleted Cognitive Services resources.

See also

- See **Authenticate requests to Azure Cognitive Services** on how to securely work with Cognitive Services.
- See **What are Azure Cognitive Services?** to get a list of different categories within Cognitive Services.
- See **Natural language support** to see the list of natural languages that Cognitive Services supports.
- See **Use Cognitive Services as containers** to understand how to use Cognitive Services on-prem.
- See **Plan and manage costs for Cognitive Services** to estimate cost of using Cognitive Services.

Quickstart: Create an Ubuntu Data Science Virtual Machine using an ARM template

Article • 10/05/2022 • 5 minutes to read

This quickstart will show you how to create an Ubuntu Data Science Virtual Machine using an Azure Resource Manager template (ARM template). Data Science Virtual Machines are cloud-based virtual machines preloaded with a suite of data science and machine learning frameworks and tools. When deployed on GPU-powered compute resources, all tools and libraries are configured to use the GPU.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

- An Azure subscription. If you don't have an Azure subscription, create a free account before you begin.
- To use the CLI commands in this document from your **local environment**, you need the Azure CLI.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
```

```
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.8.9.13224",
      "templateHash": "4895680407304578048"
    }
  },
  "parameters": {
    "adminUsername": {
      "type": "string",
      "metadata": {
        "description": "Username for Administrator Account"
      }
    },
    "vmName": {
      "type": "string",
      "defaultValue": "vmName",
      "metadata": {
        "description": "The name of you Virtual Machine."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "cpu_gpu": {
      "type": "string",
      "defaultValue": "CPU-4GB",
      "allowedValues": [
        "CPU-4GB",
        "CPU-7GB",
        "CPU-8GB",
        "CPU-14GB",
        "CPU-16GB",
        "GPU-56GB"
      ],
      "metadata": {
        "description": "Choose between CPU or GPU processing"
      }
    },
    "virtualNetworkName": {
      "type": "string",
      "defaultValue": "vNet",
      "metadata": {
        "description": "Name of the VNET"
      }
    },
    "subnetName": {
      "type": "string",
      "defaultValue": "subnet",
      "metadata": {
        "description": "Name of the subnet in the virtual network"
```

```
      }
    },
    "networkSecurityGroupName": {
      "type": "string",
      "defaultValue": "SecGroupNet",
      "metadata": {
        "description": "Name of the Network Security Group"
      }
    },
    "authenticationType": {
      "type": "string",
      "defaultValue": "sshPublicKey",
      "allowedValues": [
        "sshPublicKey",
        "password"
      ],
      "metadata": {
        "description": "Type of authentication to use on the Virtual
Machine. SSH key is recommended."
      }
    },
    "adminPasswordOrKey": {
      "type": "secureString",
      "metadata": {
        "description": "SSH Key or password for the Virtual Machine. SSH key
is recommended."
      }
    }
  },
  "variables": {
    "networkInterfaceName": "[format('{0}NetInt', parameters('vmName'))]",
    "virtualMachineName": "[parameters('vmName')]",
    "publicIpAddressName": "[format('{0}PublicIP', parameters('vmName'))]",
    "subnetRef": "[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('virtualNetworkName'), parameters('subnetName'))]",
    "nsgId": "[resourceId('Microsoft.Network/networkSecurityGroups',
parameters('networkSecurityGroupName'))]",
    "osDiskType": "StandardSSD_LRS",
    "storageAccountName": "[format('storage{0}',
uniqueString(resourceGroup().id))]",
    "storageAccountType": "Standard_LRS",
    "storageAccountKind": "Storage",
    "vmSize": {
      "CPU-4GB": "Standard_B2s",
      "CPU-7GB": "Standard_D2s_v3",
      "CPU-8GB": "Standard_D2s_v3",
      "CPU-14GB": "Standard_D4s_v3",
      "CPU-16GB": "Standard_D4s_v3",
      "GPU-56GB": "Standard_NC6_Promo"
    },
    "linuxConfiguration": {
      "disablePasswordAuthentication": true,
      "ssh": {
        "publicKeys": [
          {
```

```
            "path": "[format('/home/{0}/.ssh/authorized_keys',
parameters('adminUsername'))]",
            "keyData": "[parameters('adminPasswordOrKey')]"
          }
        ]
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Network/networkInterfaces",
      "apiVersion": "2021-05-01",
      "name": "[variables('networkInterfaceName')]",
      "location": "[parameters('location')]",
      "properties": {
        "ipConfigurations": [
          {
            "name": "ipconfig1",
            "properties": {
              "subnet": {
                "id": "[variables('subnetRef')]"
              },
              "privateIPAllocationMethod": "Dynamic",
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses',
variables('publicIpAddressName'))]"
              }
            }
          }
        ],
        "networkSecurityGroup": {
          "id": "[variables('nsgId')]"
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkSecurityGroups',
parameters('networkSecurityGroupName'))]",
        "[resourceId('Microsoft.Network/publicIPAddresses',
variables('publicIpAddressName'))]",
        "[resourceId('Microsoft.Network/virtualNetworks',
parameters('virtualNetworkName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/networkSecurityGroups",
      "apiVersion": "2021-05-01",
      "name": "[parameters('networkSecurityGroupName')]",
      "location": "[parameters('location')]",
      "properties": {
        "securityRules": [
          {
            "name": "JupyterHub",
            "properties": {
              "priority": 1010,
              "protocol": "Tcp",
```

```
            "access": "Allow",
            "direction": "Inbound",
            "sourceAddressPrefix": "*",
            "sourcePortRange": "*",
            "destinationAddressPrefix": "*",
            "destinationPortRange": "8000"
          }
        },
        {
          "name": "RStudioServer",
          "properties": {
            "priority": 1020,
            "protocol": "Tcp",
            "access": "Allow",
            "direction": "Inbound",
            "sourceAddressPrefix": "*",
            "sourcePortRange": "*",
            "destinationAddressPrefix": "*",
            "destinationPortRange": "8787"
          }
        },
        {
          "name": "SSH",
          "properties": {
            "priority": 1030,
            "protocol": "Tcp",
            "access": "Allow",
            "direction": "Inbound",
            "sourceAddressPrefix": "*",
            "sourcePortRange": "*",
            "destinationAddressPrefix": "*",
            "destinationPortRange": "22"
          }
        }
      ]
    }
  },
  {
    "type": "Microsoft.Network/virtualNetworks",
    "apiVersion": "2021-05-01",
    "name": "[parameters('virtualNetworkName')]",
    "location": "[parameters('location')]",
    "properties": {
      "addressSpace": {
        "addressPrefixes": [
          "10.0.0.0/24"
        ]
      },
      "subnets": [
        {
          "name": "[parameters('subnetName')]",
          "properties": {
            "addressPrefix": "10.0.0.0/24",
            "privateEndpointNetworkPolicies": "Enabled",
            "privateLinkServiceNetworkPolicies": "Enabled"
```

```
            }
          }
        ]
      }
    },
    {
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2021-05-01",
      "name": "[variables('publicIpAddressName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Basic",
        "tier": "Regional"
      },
      "properties": {
        "publicIPAllocationMethod": "Dynamic"
      }
    },
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-08-01",
      "name": "[variables('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[variables('storageAccountType')]"
      },
      "kind": "[variables('storageAccountKind')]"
    },
    {
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2021-11-01",
      "name": "[format('{0}-{1}', variables('virtualMachineName'),
parameters('cpu_gpu'))]",
      "location": "[parameters('location')]",
      "properties": {
        "hardwareProfile": {
          "vmSize": "[variables('vmSize')[parameters('cpu_gpu')]]"
        },
        "storageProfile": {
          "osDisk": {
            "createOption": "FromImage",
            "managedDisk": {
              "storageAccountType": "[variables('osDiskType')]"
            }
          },
          "imageReference": {
            "publisher": "microsoft-dsvm",
            "offer": "ubuntu-1804",
            "sku": "1804-gen2",
            "version": "latest"
          }
        },
        "networkProfile": {
          "networkInterfaces": [
            {
```

```
              "id": "[resourceId('Microsoft.Network/networkInterfaces',
variables('networkInterfaceName'))]"
            }
          ]
        },
        "osProfile": {
          "computerName": "[variables('virtualMachineName')]",
          "adminUsername": "[parameters('adminUsername')]",
          "adminPassword": "[parameters('adminPasswordOrKey')]",
          "linuxConfiguration": "
[if(equals(parameters('authenticationType'), 'password'), json('null'),
variables('linuxConfiguration'))]"
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkInterfaces',
variables('networkInterfaceName'))]"
      ]
    }
  ],
  "outputs": {
    "adminUsername": {
      "type": "string",
      "value": "[parameters('adminUsername')]"
    }
  }
}
```

The following resources are defined in the template:

- Microsoft.Network/networkInterfaces
- Microsoft.Network/networkSecurityGroups
- Microsoft.Network/virtualNetworks
- Microsoft.Network/publicIPAddresses
- Microsoft.Storage/storageAccounts
- Microsoft.Compute/virtualMachines: Create a cloud-based virtual machine. In this template, the virtual machine is configured as a Data Science Virtual Machine running Ubuntu.

Deploy the template

To use the template from the Azure CLI, login and choose your subscription (See Sign in with Azure CLI). Then run:

Azure CLI

```
read -p "Enter the name of the resource group to create:" resourceGroupName
&&
```

```
read -p "Enter the Azure location (e.g., centralus):" location &&
read -p "Enter the authentication type (must be 'password' or
'sshPublicKey') :" authenticationType &&
read -p "Enter the login name for the administrator account (may not be
'admin'):" adminUsername &&
read -p "Enter administrator account secure string (value of password or ssh
public key):" adminPasswordOrKey &&
templateUri="https://raw.githubusercontent.com/Azure/azure-quickstart-
templates/master/application-workloads/datascience/vm-ubuntu-DSVM-GPU-or-
CPU/azuredeploy.json" &&
az group create --name $resourceGroupName --location "$location" &&
az deployment group create --resource-group $resourceGroupName --template-
uri $templateUri --parameters adminUsername=$adminUsername
authenticationType=$authenticationType
adminPasswordOrKey=$adminPasswordOrKey &&
echo "Press [ENTER] to continue ..." &&
read
```

When you run the above command, enter:

1. The name of the resource group you'd like to create to contain the DSVM and associated resources.
2. The Azure location in which you wish to make the deployment.
3. The authentication type you'd like to use (enter the string `password` or `sshPublicKey`).
4. The login name of the administrator account (this value may not be `admin`).
5. The value of the password or ssh public key for the account.

Review deployed resources

To see your Data Science Virtual Machine:

1. Go to the Azure portal
2. Sign in.
3. Choose the resource group you just created.

You'll see the Resource Group's information:

Click on the Virtual Machine resource to go to its information page. Here you can find information on the VM, including connection details.

Clean up resources

If you don't want to use this virtual machine, delete it. Since the DSVM is associated with other resources such as a storage account, you'll probably want to delete the entire resource group you created. You can delete the resource group using the portal by clicking on the **Delete** button and confirming. Or, you can delete the resource group from the CLI with:

Azure CLI

```
echo "Enter the Resource Group name:" &&
read resourceGroupName &&
az group delete --name $resourceGroupName &&
echo "Press [ENTER] to continue ..."
```

Next steps

In this quickstart, you created a Data Science Virtual Machine from an ARM template.

Sample programs & ML walkthroughs

Additional resources

Documentation

Quickstart: Create an Ubuntu Data Science Virtual Machine - Azure Data Science Virtual Machine

Configure and create a Data Science Virtual Machine for Linux (Ubuntu) to do analytics and machine learning.

Create and manage a compute instance - Azure Machine Learning

Learn how to create and manage an Azure Machine Learning compute instance. Use as your development environment, or as compute target for dev/test purposes.

Tools included on the Azure Data Science Virtual Machine - Azure Data Science Virtual Machine

A list of tools included on the Windows and Ubuntu DSVM images

Search for assets - Azure Machine Learning

Find your Azure Machine Learning assets with search

Sample programs & ML walkthroughs - Azure Data Science Virtual Machine

Through these samples and walkthroughs, learn how to handle common tasks and scenarios with the Data Science Virtual Machine.

Explore Linux - Azure Data Science Virtual Machine

Learn how to complete several common data science tasks by using the Linux Data Science Virtual Machine.

Customize compute instance with a script (preview) - Azure Machine Learning

Create a customized compute instance, using a startup script. Use the compute instance as your development environment, or as compute target for dev/test purposes.

What is an Azure Machine Learning compute instance? - Azure Machine Learning

Learn about the Azure Machine Learning compute instance, a fully managed cloud-based workstation.

Show 5 more

Training

Learning path

Deploy and manage resources in Azure by using JSON ARM templates - Training

Learn how JSON ARM templates enable you to manage both basic and complex infrastructure deployments on Azure.

Certification

Microsoft Certified: Azure Virtual Desktop Specialty - Certifications

Candidates for this certification are server or desktop administrators with subject matter expertise in designing, implementing, managing, and maintaining Microsoft Azure Virtual Desktop experiences and remote apps for any device.

Use an Azure Resource Manager template to create a workspace for Azure Machine Learning

Article • 03/10/2023 • 15 minutes to read

In this article, you learn several ways to create an Azure Machine Learning workspace using Azure Resource Manager templates. A Resource Manager template makes it easy to create resources as a single, coordinated operation. A template is a JSON document that defines the resources that are needed for a deployment. It may also specify deployment parameters. Parameters are used to provide input values when using the template.

For more information, see Deploy an application with Azure Resource Manager template.

Prerequisites

- An **Azure subscription**. If you do not have one, try the free or paid version of Azure Machine Learning.
- To use a template from a CLI, you need either Azure PowerShell or the Azure CLI.

Limitations

- When creating a new workspace, you can either automatically create services needed by the workspace or use existing services. If you want to use **existing services from a different Azure subscription** than the workspace, you must register the Azure Machine Learning namespace in the subscription that contains those services. For example, creating a workspace in subscription A that uses a storage account from subscription B, the Azure Machine Learning namespace must be registered in subscription B before you can use the storage account with the workspace.

  The resource provider for Azure Machine Learning is **Microsoft.MachineLearningServices**. For information on how to see if it is registered and how to register it, see the Azure resource providers and types article.

> **Important**
>
> This only applies to resources provided during workspace creation; Azure Storage Accounts, Azure Container Register, Azure Key Vault, and Application Insights.

- The example template may not always use the latest API version for Azure Machine Learning. Before using the template, we recommend modifying it to use the latest API versions. For information on the latest API versions for Azure Machine Learning, see the Azure Machine Learning REST API.

  > **Tip**
  >
  > Each Azure service has its own set of API versions. For information on the API for a specific service, check the service information in the **Azure REST API reference**.

  To update the API version, find the `"apiVersion": "YYYY-MM-DD"` entry for the resource type and update it to the latest version. The following example is an entry for Azure Machine Learning:

  JSON

  ```
  "type": "Microsoft.MachineLearningServices/workspaces",
  "apiVersion": "2020-03-01",
  ```

Multiple workspaces in the same VNet

The template doesn't support multiple Azure Machine Learning workspaces deployed in the same VNet. This is because the template creates new DNS zones during deployment.

If you want to create a template that deploys multiple workspaces in the same VNet, set this up manually (using the Azure Portal or CLI) and then use the Azure portal to generate a template.

Workspace Resource Manager template

The Azure Resource Manager template used throughout this document can be found in the microsoft.machineleaerningservices/machine-learning-workspace-vnet directory

of the Azure quickstart templates GitHub repository.

This template creates the following Azure services:

- Azure Storage Account
- Azure Key Vault
- Azure Application Insights
- Azure Container Registry
- Azure Machine Learning workspace

The resource group is the container that holds the services. The various services are required by the Azure Machine Learning workspace.

The example template has two **required** parameters:

- The **location** where the resources will be created.

  The template will use the location you select for most resources. The exception is the Application Insights service, which is not available in all of the locations that the other services are. If you select a location where it is not available, the service will be created in the South Central US location.

- The **workspaceName**, which is the friendly name of the Azure Machine Learning workspace.

  > **Note**
  >
  > The workspace name is case-insensitive.

  The names of the other services are generated randomly.

> **Tip**
>
> While the template associated with this document creates a new Azure Container Registry, you can also create a new workspace without creating a container registry. One will be created when you perform an operation that requires a container registry. For example, training or deploying a model.
>
> You can also reference an existing container registry or storage account in the Azure Resource Manager template, instead of creating a new one. When doing so, you must either **use a managed identity** (preview), or **enable the admin account** for the container registry.

> ⚠ **Warning**
>
> Once an Azure Container Registry has been created for a workspace, do not delete it. Doing so will break your Azure Machine Learning workspace.

For more information on templates, see the following articles:

- Author Azure Resource Manager templates
- Deploy an application with Azure Resource Manager templates
- Microsoft.MachineLearningServices resource types

Deploy template

To deploy your template you have to create a resource group.

See the Azure portal section if you prefer using the graphical user interface.

Azure CLI

Azure CLI

```
az group create --name "examplegroup" --location "eastus"
```

Once your resource group is successfully created, deploy the template with the following command:

Azure CLI

Azure CLI

```
az deployment group create \
    --name "exampledeployment" \
    --resource-group "examplegroup" \
    --template-uri "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.machinelearningservices/machine-learning-workspace-vnet/azuredeploy.json" \
    --parameters workspaceName="exampleworkspace" location="eastus"
```

By default, all of the resources created as part of the template are new. However, you also have the option of using existing resources. By providing additional parameters to the template, you can use existing resources. For example, if you want to use an existing

storage account set the **storageAccountOption** value to **existing** and provide the name of your storage account in the **storageAccountName** parameter.

> **Important**
>
> If you want to use an existing Azure Storage account, it cannot be a premium account (Premium_LRS and Premium_GRS). It also cannot have a hierarchical namespace (used with Azure Data Lake Storage Gen2). Neither premium storage or hierarchical namespace are supported with the default storage account of the workspace. Neither premium storage or hierarchical namespaces are supported with the *default* storage account of the workspace. You can use premium storage or hierarchical namespace with *non-default* storage accounts.

Azure CLI

Azure CLI

```
az deployment group create \
    --name "exampledeployment" \
    --resource-group "examplegroup" \
    --template-uri "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.machinelearningservices/machine-learning-workspace-vnet/azuredeploy.json" \
    --parameters workspaceName="exampleworkspace" \
      location="eastus" \
      storageAccountOption="existing" \
      storageAccountName="existingstorageaccountname"
```

Deploy an encrypted workspace

The following example template demonstrates how to create a workspace with three settings:

- Enable high confidentiality settings for the workspace. This creates a new Azure Cosmos DB instance.
- Enable encryption for the workspace.
- Uses an existing Azure Key Vault to retrieve customer-managed keys. Customer-managed keys are used to create a new Azure Cosmos DB instance for the workspace.

> **Important**
>
> Once a workspace has been created, you cannot change the settings for confidential data, encryption, key vault ID, or key identifiers. To change these values, you must create a new workspace using the new values.

For more information, see Customer-managed keys.

> **Important**
>
> There are some specific requirements your subscription must meet before using this template:
>
> - You must have an existing Azure Key Vault that contains an encryption key.
> - The Azure Key Vault must be in the same region where you plan to create the Azure Machine Learning workspace.
> - You must specify the ID of the Azure Key Vault and the URI of the encryption key.
>
> For steps on creating the vault and key, see **Configure customer-managed keys**.

To get the values for the `cmk_keyvault` (ID of the Key Vault) and the `resource_cmk_uri` (key URI) parameters needed by this template, use the following steps:

1. To get the Key Vault ID, use the following command:

   Azure CLI

   Azure CLI

   ```
   az keyvault show --name <keyvault-name> --query 'id' --output tsv
   ```

   This command returns a value similar to `/subscriptions/{subscription-guid}/resourceGroups/<resource-group-name>/providers/Microsoft.KeyVault/vaults/<keyvault-name>`.

2. To get the value for the URI for the customer managed key, use the following command:

   Azure CLI

Azure CLI

```
az keyvault key show --vault-name <keyvault-name> --name <key-name>
--query 'key.kid' --output tsv
```

This command returns a value similar to `https://mykeyvault.vault.azure.net/keys/mykey/{guid}`.

> **Important**
>
> Once a workspace has been created, you cannot change the settings for confidential data, encryption, key vault ID, or key identifiers. To change these values, you must create a new workspace using the new values.

To enable use of Customer Managed Keys, set the following parameters when deploying the template:

- **encryption_status** to **Enabled**.
- **cmk_keyvault** to the `cmk_keyvault` value obtained in previous steps.
- **resource_cmk_uri** to the `resource_cmk_uri` value obtained in previous steps.

Azure CLI

Azure CLI

```
az deployment group create \
    --name "exampledeployment" \
    --resource-group "examplegroup" \
    --template-uri "https://raw.githubusercontent.com/Azure/azure-
quickstart-
templates/master/quickstarts/microsoft.machinelearningservices/machine-
learning-workspace-vnet/azuredeploy.json" \
    --parameters workspaceName="exampleworkspace" \
      location="eastus" \
      encryption_status="Enabled" \
      cmk_keyvault="/subscriptions/{subscription-
guid}/resourceGroups/<resource-group-
name>/providers/Microsoft.KeyVault/vaults/<keyvault-name>" \

resource_cmk_uri="https://mykeyvault.vault.azure.net/keys/mykey/{guid}"
\
```

When using a customer-managed key, Azure Machine Learning creates a secondary resource group which contains the Azure Cosmos DB instance. For more information, see Encryption at rest in Azure Cosmos DB.

An additional configuration you can provide for your data is to set the **confidential_data** parameter to **true**. Doing so, does the following:

- Starts encrypting the local scratch disk for Azure Machine Learning compute clusters, providing you have not created any previous clusters in your subscription. If you have previously created a cluster in the subscription, open a support ticket to have encryption of the scratch disk enabled for your compute clusters.
- Cleans up the local scratch disk between jobs.
- Securely passes credentials for the storage account, container registry, and SSH account from the execution layer to your compute clusters by using key vault.
- Enables IP filtering to ensure the underlying batch pools cannot be called by any external services other than AzureMachineLearningService.

  > **Important**
  >
  > Once a workspace has been created, you cannot change the settings for confidential data, encryption, key vault ID, or key identifiers. To change these values, you must create a new workspace using the new values.

  For more information, see encryption at rest.

Deploy workspace behind a virtual network

By setting the `vnetOption` parameter value to either `new` or `existing`, you are able to create the resources used by a workspace behind a virtual network.

> **Important**
>
> For container registry, only the 'Premium' sku is supported.

> **Important**
>
> Application Insights does not support deployment behind a virtual network.

Only deploy workspace behind private endpoint

If your associated resources are not behind a virtual network, you can set the **privateEndpointType** parameter to `AutoAproval` or `ManualApproval` to deploy the workspace behind a private endpoint. This can be done for both new and existing workspaces. When updating an existing workspace, fill in the template parameters with the information from the existing workspace.

Azure CLI

Azure CLI

```
az deployment group create \
    --name "exampledeployment" \
    --resource-group "examplegroup" \
    --template-uri "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.machinelearningservices/machine-learning-workspace-vnet/azuredeploy.json" \
    --parameters workspaceName="exampleworkspace" \
      location="eastus" \
      privateEndpointType="AutoApproval"
```

Use a new virtual network

To deploy a resource behind a new virtual network, set the **vnetOption** to **new** along with the virtual network settings for the respective resource. The deployment below shows how to deploy a workspace with the storage account resource behind a new virtual network.

Azure CLI

Azure CLI

```
az deployment group create \
    --name "exampledeployment" \
    --resource-group "examplegroup" \
    --template-uri "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.machinelearningservices/machine-learning-workspace-vnet/azuredeploy.json" \
    --parameters workspaceName="exampleworkspace" \
      location="eastus" \
      vnetOption="new" \
      vnetName="examplevnet" \
```

```
      storageAccountBehindVNet="true"
      privateEndpointType="AutoApproval"
```

Alternatively, you can deploy multiple or all dependent resources behind a virtual network.

Azure CLI

Azure CLI

```
az deployment group create \
    --name "exampledeployment" \
    --resource-group "examplegroup" \
    --template-uri "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.machinelearningservices/machine-learning-workspace-vnet/azuredeploy.json" \
    --parameters workspaceName="exampleworkspace" \
      location="eastus" \
      vnetOption="new" \
      vnetName="examplevnet" \
      storageAccountBehindVNet="true" \
      keyVaultBehindVNet="true" \
      containerRegistryBehindVNet="true" \
      containerRegistryOption="new" \
      containerRegistrySku="Premium"
      privateEndpointType="AutoApproval"
```

Use an existing virtual network & resources

To deploy a workspace with existing associated resources you have to set the **vnetOption** parameter to **existing** along with subnet parameters. However, you need to create service endpoints in the virtual network for each of the resources **before** deployment. Like with new virtual network deployments, you can have one or all of your resources behind a virtual network.

> **Important**
>
> Subnet should have `Microsoft.Storage` service endpoint

> **Important**

Subnets do not allow creation of private endpoints. Disable private endpoint to enable subnet.

1. Enable service endpoints for the resources.

Azure CLI

Azure CLI

```
az network vnet subnet update --resource-group "examplegroup" --
vnet-name "examplevnet" --name "examplesubnet" --service-endpoints
"Microsoft.Storage"
az network vnet subnet update --resource-group "examplegroup" --
vnet-name "examplevnet" --name "examplesubnet" --service-endpoints
"Microsoft.KeyVault"
az network vnet subnet update --resource-group "examplegroup" --
vnet-name "examplevnet" --name "examplesubnet" --service-endpoints
"Microsoft.ContainerRegistry"
```

2. Deploy the workspace

Azure CLI

Azure CLI

```
az deployment group create \
--name "exampledeployment" \
--resource-group "examplegroup" \
--template-uri "https://raw.githubusercontent.com/Azure/azure-
quickstart-
templates/master/quickstarts/microsoft.machinelearningservices/mach
ine-learning-workspace-vnet/azuredeploy.json" \
--parameters workspaceName="exampleworkspace" \
  location="eastus" \
  vnetOption="existing" \
  vnetName="examplevnet" \
  vnetResourceGroupName="examplegroup" \
  storageAccountBehindVNet="true" \
  keyVaultBehindVNet="true" \
  containerRegistryBehindVNet="true" \
  containerRegistryOption="new" \
  containerRegistrySku="Premium" \
  subnetName="examplesubnet" \
  subnetOption="existing"
  privateEndpointType="AutoApproval"
```

Use the Azure portal

1. Follow the steps in Deploy resources from custom template. When you arrive at the **Select a template** screen, choose the **quickstarts** entry. When it appears, select the link labeled "Click here to open template repository". This link takes you to the `quickstarts` directory in the Azure quickstart templates repository.

2. In the list of quickstart templates, select `microsoft.machinelearningservices`. Finally, select `Deploy to Azure`.

3. When the template appears, provide the following required information and any other parameters depending on your deployment scenario.

   - Subscription: Select the Azure subscription to use for these resources.
   - Resource group: Select or create a resource group to contain the services.
   - Region: Select the Azure region where the resources will be created.
   - Workspace name: The name to use for the Azure Machine Learning workspace that will be created. The workspace name must be between 3 and 33 characters. It may only contain alphanumeric characters and '-'.
   - Location: Select the location where the resources will be created.

4. Select **Review + create**.

5. In the **Review + create** screen, agree to the listed terms and conditions and select **Create**.

For more information, see Deploy resources from custom template.

Troubleshooting

Resource provider errors

When creating an Azure Machine Learning workspace, or a resource used by the workspace, you may receive an error similar to the following messages:

- `No registered resource provider found for location {location}`
- `The subscription is not registered to use namespace {resource-provider-namespace}`

Most resource providers are automatically registered, but not all. If you receive this message, you need to register the provider mentioned.

The following table contains a list of the resource providers required by Azure Machine Learning:

Resource provider	Why it's needed
Microsoft.MachineLearningServices	Creating the Azure Machine Learning workspace.
Microsoft.Storage	Azure Storage Account is used as the default storage for the workspace.
Microsoft.ContainerRegistry	Azure Container Registry is used by the workspace to build Docker images.
Microsoft.KeyVault	Azure Key Vault is used by the workspace to store secrets.
Microsoft.Notebooks/NotebookProxies	Integrated notebooks on Azure Machine Learning compute instance.
Microsoft.ContainerService	If you plan on deploying trained models to Azure Kubernetes Services.

If you plan on using a customer-managed key with Azure Machine Learning, then the following service providers must be registered:

Resource provider	Why it's needed
Microsoft.DocumentDB/databaseAccounts	Azure CosmosDB instance that logs metadata for the workspace.
Microsoft.Search/searchServices	Azure Search provides indexing capabilities for the workspace.

For information on registering resource providers, see Resolve errors for resource provider registration.

Azure Key Vault access policy and Azure Resource Manager templates

When you use an Azure Resource Manager template to create the workspace and associated resources (including Azure Key Vault), multiple times. For example, using the template multiple times with the same parameters as part of a continuous integration and deployment pipeline.

Most resource creation operations through templates are idempotent, but Key Vault clears the access policies each time the template is used. Clearing the access policies

breaks access to the Key Vault for any existing workspace that is using it. For example, Stop/Create functionalities of Azure Notebooks VM may fail.

To avoid this problem, we recommend one of the following approaches:

- Do not deploy the template more than once for the same parameters. Or delete the existing resources before using the template to recreate them.
- Examine the Key Vault access policies and then use these policies to set the `accessPolicies` property of the template. To view the access policies, use the following Azure CLI command:

  Azure CLI

  ```
  az keyvault show --name mykeyvault --resource-group myresourcegroup --
  query properties.accessPolicies
  ```

  For more information on using the `accessPolicies` section of the template, see the AccessPolicyEntry object reference.

- Check if the Key Vault resource already exists. If it does, do not recreate it through the template. For example, to use the existing Key Vault instead of creating a new one, make the following changes to the template:
  - **Add** a parameter that accepts the ID of an existing Key Vault resource:

    JSON

    ```
    "keyVaultId":{
      "type": "string",
      "metadata": {
        "description": "Specify the existing Key Vault ID."
      }
    }
    ```

  - **Remove** the section that creates a Key Vault resource:

    JSON

    ```
    {
      "type": "Microsoft.KeyVault/vaults",
      "apiVersion": "2018-02-14",
      "name": "[variables('keyVaultName')]",
      "location": "[parameters('location')]",
      "properties": {
        "tenantId": "[variables('tenantId')]",
        "sku": {
    ```

```
      "name": "standard",
      "family": "A"
    },
    "accessPolicies": [
    ]
  }
},
```

- **Remove** the `"[resourceId('Microsoft.KeyVault/vaults', variables('keyVaultName'))]",` line from the `dependsOn` section of the workspace. Also **Change** the `keyVault` entry in the `properties` section of the workspace to reference the `keyVaultId` parameter:

JSON

```
{
  "type": "Microsoft.MachineLearningServices/workspaces",
  "apiVersion": "2019-11-01",
  "name": "[parameters('workspaceName')]",
  "location": "[parameters('location')]",
  "dependsOn": [
    "[resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))]",
    "[resourceId('Microsoft.Insights/components',
variables('applicationInsightsName'))]"
  ],
  "identity": {
    "type": "systemAssigned"
  },
  "sku": {
    "tier": "[parameters('sku')]",
    "name": "[parameters('sku')]"
  },
  "properties": {
    "friendlyName": "[parameters('workspaceName')]",
    "keyVault": "[parameters('keyVaultId')]",
    "applicationInsights": "
[resourceId('Microsoft.Insights/components',variables('applicationIn
sightsName'))]",
    "storageAccount": "
[resourceId('Microsoft.Storage/storageAccounts/',variables('storageA
ccountName'))]"
  }
}
```

After these changes, you can specify the ID of the existing Key Vault resource when running the template. The template will then reuse the Key Vault by setting the `keyVault` property of the workspace to its ID.

To get the ID of the Key Vault, you can reference the output of the original template job or use the Azure CLI. The following command is an example of using the Azure CLI to get the Key Vault resource ID:

Azure CLI

```
az keyvault show --name mykeyvault --resource-group myresourcegroup --
query id
```

This command returns a value similar to the following text:

text

```
/subscriptions/{subscription-
guid}/resourceGroups/myresourcegroup/providers/Microsoft.KeyVault/vault
s/mykeyvault
```

Next steps

- Deploy resources with Resource Manager templates and Resource Manager REST API.
- Creating and deploying Azure resource groups through Visual Studio.
- For other templates related to Azure Machine Learning, see the Azure Quickstart Templates repository.
- How to use workspace diagnostics.
- Move an Azure Machine Learning workspace to another subscription.

Additional resources

Documentation

azureml.core.compute.amlcompute.AmlCompute class - Azure Machine Learning Python

Manages an Azure Machine Learning compute in Azure Machine Learning. An Azure Machine Learning Compute (AmlCompute) is a managed-compute infrastructure that allows you to easily create a single or multi-node compute. The compute is created within your workspace region as a…

Collect data on your production models - Azure Machine Learning

Learn how to collect data from an Azure Machine Learning model deployed on an Azure Kubernetes Service (AKS) cluster.

azureml.core.Webservice class - Azure Machine Learning Python

Defines base functionality for deploying models as web service endpoints in Azure Machine Learning.

Webservice constructor is used to retrieve a cloud representation of a Webservice object associated with the provided Workspace. Returns an instance of a child class corresponding to the specific typ…

Create workspaces with Azure CLI - Azure Machine Learning

Learn how to use the Azure CLI extension for machine learning to create a new Azure Machine Learning workspace.

GitHub Actions for CI/CD - Azure Machine Learning

Learn about how to create a GitHub Actions workflow to train a model on Azure Machine Learning

Install and set up the CLI (v1)

Learn how to use the Azure CLI extension (v1) for ML to create & manage resources such as your workspace, datastores, datasets, pipelines, models, and deployments.

What are compute targets - Azure Machine Learning

Learn how to designate a compute resource or environment to train or deploy your model with Azure Machine Learning.

azureml.pipeline.steps package - Azure Machine Learning Python

Contains pre-built steps that can be executed in an Azure Machine Learning Pipeline. Azure ML Pipeline steps can be configured together to construct a Pipeline, which represents a shareable and reusable Azure Machine Learning workflow. Each step of a pipeline can be configured to allow reus…

Show 5 more

Training

Module

Create Azure Machine Learning resources with the CLI (v2) - Training

Create Azure Machine Learning resources with the CLI (v2)

Certification

Microsoft Certified: Azure Administrator Associate - Certifications

Azure administrators implement, manage, and monitor an organization's Microsoft Azure environment, including virtual networks, storage, compute, identity, security, and governance.

Quickstart: Create a server - ARM template

Article • 03/09/2023

This quickstart describes how to create an Analysis Services server resource in your Azure subscription by using an Azure Resource Manager template (ARM template).

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

- **Azure subscription**: Visit Azure Free Trial to create an account.
- **Azure Active Directory**: Your subscription must be associated with an Azure Active Directory tenant. And, you need to be signed in to Azure with an account in that Azure Active Directory. To learn more, see Authentication and user permissions.

Review the template

The template used in this quickstart is from Azure templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "17179487501489240263"
    }
  },
```

```
  "parameters": {
    "serverName": {
      "type": "string",
      "metadata": {
        "description": "The name of the Azure Analysis Services server to
create. Server name must begin with a letter, be lowercase alphanumeric, and
between 3 and 63 characters in length. Server name must be unique per
region."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location of the Azure Analysis Services server. For
supported regions, see https://docs.microsoft.com/en-us/azure/analysis-
services/analysis-services-overview#availability-by-region"
      }
    },
    "skuName": {
      "type": "string",
      "defaultValue": "S0",
      "metadata": {
        "description": "The sku name of the Azure Analysis Services server
to create. Choose from: B1, B2, D1, S0, S1, S2, S3, S4, S8, S9. Some skus
are region specific. See https://docs.microsoft.com/en-us/azure/analysis-
services/analysis-services-overview#availability-by-region"
      }
    },
    "capacity": {
      "type": "int",
      "defaultValue": 1,
      "metadata": {
        "description": "The total number of query replica scale-out
instances. Scale-out of more than one instance is supported on selected
regions only. See https://docs.microsoft.com/en-us/azure/analysis-
services/analysis-services-overview#availability-by-region"
      }
    },
    "firewallSettings": {
      "type": "object",
      "defaultValue": {
        "firewallRules": [
          {
            "firewallRuleName": "AllowFromAll",
            "rangeStart": "0.0.0.0",
            "rangeEnd": "255.255.255.255"
          }
        ],
        "enablePowerBIService": true
      },
      "metadata": {
        "description": "The inbound firewall rules to define on the server.
If not specified, firewall is disabled."
      }
```

```
    },
    "backupBlobContainerUri": {
      "type": "string",
      "defaultValue": "",
      "metadata": {
        "description": "The SAS URI to a private Azure Blob Storage
container with read, write and list permissions. Required only if you intend
to use the backup/restore functionality. See https://docs.microsoft.com/en-
us/azure/analysis-services/analysis-services-backup"
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.AnalysisServices/servers",
      "apiVersion": "2017-08-01",
      "name": "[parameters('serverName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('skuName')]",
        "capacity": "[parameters('capacity')]"
      },
      "properties": {
        "ipV4FirewallSettings": "[parameters('firewallSettings')]",
        "backupBlobContainerUri": "[parameters('backupBlobContainerUri')]"
      }
    }
  ]
}
```

A single Microsoft.AnalysisServices/servers resource with a firewall rule is defined in the template.

Deploy the template

1. Select the following Deploy to Azure link to sign in to Azure and open a template. The template is used to create an Analysis Services server resource and specify required and optional properties.

   Deploy to Azure

2. Select or enter the following values.

   Unless specified otherwise, use default values.

   - **Subscription**: Select an Azure subscription.
   - **Resource group**: Click **Create new**, and then enter a unique name for the new resource group.

- **Location**: Select a default location for resources created in the resource group.
- **Server Name**: Enter a name for the server resource.
- **Location**: Ignore for Analysis Services. Location is specified in Server Location.
- **Server location**: Enter the location of the Analysis Services server. This is often the same region as the default Location specified for the Resource Group, but not required. For example **North Central US**. For supported regions, see Analysis Services availability by region.
- **Sku Name**: Enter the sku name for the Analysis Services server to create. Choose from: B1, B2, D1, S0, S1, S2, S3, S4, S8v2, S9v2. Sku availability depends on region. S0 or D1 is recommended for evaluation and testing.
- **Capacity**: Enter the total number of query replica scale-out instances. Scale-out of more than one instance is supported in select regions only.
- **Firewall Settings**: Enter inbound firewall rules to define for the server. If not specified, firewall is disabled.
- **Backup Blob Container Uri**: Enter the SAS URI to a private Azure Blob Storage container with read, write and list permissions. Required only if you intend to use Backup/restore.
- **I agree to the terms and conditions state above**: Select.

3. Select **Purchase**. After the server has been deployed successfully, you get a notification:

Validate the deployment

Use the Azure portal or Azure PowerShell to verify the resource group and server resource was created.

PowerShell

Azure PowerShell

```
$resourceGroupName = Read-Host -Prompt "Enter the Resource Group name"
(Get-AzResource -ResourceType "Microsoft.AnalysisServices/servers" -
ResourceGroupName $resourceGroupName).Name
 Write-Host "Press [ENTER] to continue..."
```

Clean up resources

When no longer needed, use the Azure portal, Azure CLI, or Azure PowerShell to delete the resource group and the server resource.

CLI

Azure CLI

```
echo "Enter the Resource Group name:" &&
read resourceGroupName &&
az group delete --name $resourceGroupName &&
echo "Press [ENTER] to continue ..."
```

Next steps

In this quickstart, you used an ARM template to create a new resource group and an Azure Analysis Services server resource. After you've created a server resource by using the template, consider the following:

Quickstart: Configure server firewall - Portal

Quickstart: Create an Azure Data Factory using ARM template

Article • 03/08/2023

APPLIES TO: Azure Data Factory Azure Synapse Analytics

This quickstart describes how to use an Azure Resource Manager template (ARM template) to create an Azure data factory. The pipeline you create in this data factory **copies** data from one folder to another folder in an Azure blob storage. For a tutorial on how to **transform** data using Azure Data Factory, see Tutorial: Transform data using Spark.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

> **Note**
>
> This article does not provide a detailed introduction of the Data Factory service. For an introduction to the Azure Data Factory service, see **Introduction to Azure Data Factory**.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

Azure subscription

If you don't have an Azure subscription, create a free account before you begin.

Create a file

Open a text editor such as **Notepad**, and create a file named **emp.txt** with the following content:

emp.txt

```
John, Doe
Jane, Doe
```

Save the file in the **C:\ADFv2QuickStartPSH** folder. (If the folder doesn't already exist, create it.)

Review template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.9.1.41621",
      "templateHash": "5567387787764483116"
    }
  },
  "parameters": {
    "dataFactoryName": {
      "type": "string",
      "defaultValue": "[format('datafactory{0}', uniqueString(resourceGroup().id))]",
      "metadata": {
        "description": "Data Factory Name"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location of the data factory."
      }
    },
    "storageAccountName": {
      "type": "string",
      "defaultValue": "[format('storage{0}', uniqueString(resourceGroup().id))]",
      "metadata": {
        "description": "Name of the Azure storage account that contains the
```

```
input/output data."
      }
    },
    "blobContainerName": {
      "type": "string",
      "defaultValue": "[format('blob{0}',
uniqueString(resourceGroup().id))]",
      "metadata": {
        "description": "Name of the blob container in the Azure Storage
account."
      }
    }
  },
  "variables": {
    "dataFactoryLinkedServiceName": "ArmtemplateStorageLinkedService",
    "dataFactoryDataSetInName": "ArmtemplateTestDatasetIn",
    "dataFactoryDataSetOutName": "ArmtemplateTestDatasetOut",
    "pipelineName": "ArmtemplateSampleCopyPipeline"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-08-01",
      "name": "[parameters('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "StorageV2"
    },
    {
      "type": "Microsoft.Storage/storageAccounts/blobServices/containers",
      "apiVersion": "2021-08-01",
      "name": "[format('{0}/default/{1}', parameters('storageAccountName'),
parameters('blobContainerName'))]",
      "dependsOn": [
        "[resourceId('Microsoft.Storage/storageAccounts',
parameters('storageAccountName'))]"
      ]
    },
    {
      "type": "Microsoft.DataFactory/factories",
      "apiVersion": "2018-06-01",
      "name": "[parameters('dataFactoryName')]",
      "location": "[parameters('location')]",
      "identity": {
        "type": "SystemAssigned"
      }
    },
    {
      "type": "Microsoft.DataFactory/factories/linkedservices",
      "apiVersion": "2018-06-01",
      "name": "[format('{0}/{1}', parameters('dataFactoryName'),
variables('dataFactoryLinkedServiceName'))]",
      "properties": {
```

```
        "type": "AzureBlobStorage",
        "typeProperties": {
          "connectionString": "
[format('DefaultEndpointsProtocol=https;AccountName={0};AccountKey={1}',
parameters('storageAccountName'),
listKeys(resourceId('Microsoft.Storage/storageAccounts',
parameters('storageAccountName')), '2021-08-01').keys[0].value)]"
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.DataFactory/factories',
parameters('dataFactoryName'))]",
        "[resourceId('Microsoft.Storage/storageAccounts',
parameters('storageAccountName'))]"
      ]
    },
    {
      "type": "Microsoft.DataFactory/factories/datasets",
      "apiVersion": "2018-06-01",
      "name": "[format('{0}/{1}', parameters('dataFactoryName'),
variables('dataFactoryDataSetInName'))]",
      "properties": {
        "linkedServiceName": {
          "referenceName": "[variables('dataFactoryLinkedServiceName')]",
          "type": "LinkedServiceReference"
        },
        "type": "Binary",
        "typeProperties": {
          "location": {
            "type": "AzureBlobStorageLocation",
            "container": "[parameters('blobContainerName')]",
            "folderPath": "input",
            "fileName": "emp.txt"
          }
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.DataFactory/factories',
parameters('dataFactoryName'))]",
        "[resourceId('Microsoft.DataFactory/factories/linkedservices',
parameters('dataFactoryName'), variables('dataFactoryLinkedServiceName'))]"
      ]
    },
    {
      "type": "Microsoft.DataFactory/factories/datasets",
      "apiVersion": "2018-06-01",
      "name": "[format('{0}/{1}', parameters('dataFactoryName'),
variables('dataFactoryDataSetOutName'))]",
      "properties": {
        "linkedServiceName": {
          "referenceName": "[variables('dataFactoryLinkedServiceName')]",
          "type": "LinkedServiceReference"
        },
        "type": "Binary",
        "typeProperties": {
```

```
          "location": {
            "type": "AzureBlobStorageLocation",
            "container": "[parameters('blobContainerName')]",
            "folderPath": "output"
          }
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.DataFactory/factories',
parameters('dataFactoryName'))]",
        "[resourceId('Microsoft.DataFactory/factories/linkedservices',
parameters('dataFactoryName'), variables('dataFactoryLinkedServiceName'))]"
      ]
    },
    {
      "type": "Microsoft.DataFactory/factories/pipelines",
      "apiVersion": "2018-06-01",
      "name": "[format('{0}/{1}', parameters('dataFactoryName'),
variables('pipelineName'))]",
      "properties": {
        "activities": [
          {
            "name": "MyCopyActivity",
            "type": "Copy",
            "typeProperties": {
              "source": {
                "type": "BinarySource",
                "storeSettings": {
                  "type": "AzureBlobStorageReadSettings",
                  "recursive": true
                }
              },
              "sink": {
                "type": "BinarySink",
                "storeSettings": {
                  "type": "AzureBlobStorageWriteSettings"
                }
              },
              "enableStaging": false
            },
            "inputs": [
              {
                "referenceName": "[variables('dataFactoryDataSetInName')]",
                "type": "DatasetReference"
              }
            ],
            "outputs": [
              {
                "referenceName": "[variables('dataFactoryDataSetOutName')]",
                "type": "DatasetReference"
              }
            ]
          }
        ]
      },
```

```
      "dependsOn": [
        "[resourceId('Microsoft.DataFactory/factories',
parameters('dataFactoryName'))]",
        "[resourceId('Microsoft.DataFactory/factories/datasets',
parameters('dataFactoryName'), variables('dataFactoryDataSetInName'))]",
        "[resourceId('Microsoft.DataFactory/factories/datasets',
parameters('dataFactoryName'), variables('dataFactoryDataSetOutName'))]"
      ]
    }
  ]
}
```

There are Azure resources defined in the template:

- Microsoft.Storage/storageAccounts: Defines a storage account.
- Microsoft.DataFactory/factories: Create an Azure Data Factory.
- Microsoft.DataFactory/factories/linkedServices: Create an Azure Data Factory linked service.
- Microsoft.DataFactory/factories/datasets: Create an Azure Data Factory dataset.
- Microsoft.DataFactory/factories/pipelines: Create an Azure Data Factory pipeline.

More Azure Data Factory template samples can be found in the quickstart template gallery.

Deploy the template

1. Select the following image to sign in to Azure and open a template. The template creates an Azure Data Factory account, a storage account, and a blob container.

   Deploy to Azure

2. Select or enter the following values.

Unless it's specified, use the default values to create the Azure Data Factory resources:

- **Subscription**: Select an Azure subscription.
- **Resource group**: Select **Create new**, enter a unique name for the resource group, and then select **OK**.
- **Region**: Select a location. For example, *East US*.
- **Data Factory Name**: Use default value.
- **Location**: Use default value.
- **Storage Account Name**: Use default value.
- **Blob Container**: Use default value.

Review deployed resources

1. Select **Go to resource group**.

2. Verify your Azure Data Factory is created.
   a. Your Azure Data Factory name is in the format - datafactory<uniqueid>.

3. Verify your storage account is created.
   a. The storage account name is in the format - storage<uniqueid>.

4. Select the storage account created and then select **Containers**.
   a. On the **Containers** page, select the blob container you created.
      i. The blob container name is in the format - blob<uniqueid>.

Upload a file

1. On the **Containers** page, select **Upload**.

2. In te right pane, select the **Files** box, and then browse to and select the **emp.txt** file that you created earlier.

3. Expand the **Advanced** heading.

4. In the **Upload to folder** box, enter *input*.

5. Select the **Upload** button. You should see the **emp.txt** file and the status of the upload in the list.

6. Select the **Close** icon (an **X**) to close the **Upload blob** page.

Keep the container page open, because you can use it to verify the output at the end of this quickstart.

Start Trigger

1. Navigate to the **Data factories** page, and select the data factory you created.

2. Select **Open** on the **Open Azure Data Factory Studio** tile.

3. Select the **Author** tab .

4. Select the pipeline created - ArmtemplateSampleCopyPipeline.

5. Select **Add Trigger** > **Trigger Now**.

6. In the right pane under **Pipeline run**, select **OK**.

Monitor the pipeline

1. Select the **Monitor** tab .

2. You see the activity runs associated with the pipeline run. In this quickstart, the pipeline has only one activity of type: Copy. As such, you see a run for that activity.

PIPELINE NAME	RUN START	DURATION	TRIGGERED BY	STATUS	PARAMETERS	ANNOTATIONS	ERROR
ArmtemplateSampleCopyPipeline	7/16/20, 9:48:29 PM	00:00:07	Manual trigger	Succeeded			

Verify the output file

The pipeline automatically creates an output folder in the blob container. Then, it copies the emp.txt file from the input folder to the output folder.

1. In the Azure portal, on the **Containers** page, select **Refresh** to see the output folder.

2. Select **output** in the folder list.

3. Confirm that the **emp.txt** is copied to the output folder.

Clean up resources

You can clean up the resources that you created in the Quickstart in two ways. You can delete the Azure resource group, which includes all the resources in the resource group. If you want to keep the other resources intact, delete only the data factory you created in this tutorial.

Deleting a resource group deletes all resources including data factories in it. Run the following command to delete the entire resource group:

Azure PowerShell

```
Remove-AzResourceGroup -ResourceGroupName $resourcegroupname
```

If you want to delete just the data factory, and not the entire resource group, run the following command:

Azure PowerShell

```
Remove-AzDataFactoryV2 -Name $dataFactoryName -ResourceGroupName
$resourceGroupName
```

Next steps

In this quickstart, you created an Azure Data Factory using an ARM template and validated the deployment. To learn more about Azure Data Factory and Azure Resource Manager, continue on to the articles below.

- Azure Data Factory documentation
- Learn more about Azure Resource Manager
- Get other Azure Data Factory ARM templates

Quickstart: Create an event hub by using an ARM template

Article • 03/14/2023

In this quickstart, you create an event hub by using an Azure Resource Manager template (ARM template). You deploy an ARM template to create a namespace of type Event Hubs, with one event hub.

Prerequisites

- If you don't have an Azure subscription, create a free account before you begin.
- If you're new to Azure Event Hubs, see Event Hubs overview and Event Hubs features.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "16940368634879422816"
    }
  },
  "parameters": {
    "projectName": {
      "type": "string",
      "metadata": {
        "description": "Specifies a project name that is used to generate the Event Hub name and the Namespace name."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Specifies the Azure location for all resources."
      }
```

```
    },
    "eventHubSku": {
      "type": "string",
      "defaultValue": "Standard",
      "allowedValues": [
        "Basic",
        "Standard"
      ],
      "metadata": {
        "description": "Specifies the messaging tier for Event Hub
Namespace."
      }
    }
  },
  "variables": {
    "eventHubNamespaceName": "[format('{0}ns', parameters('projectName'))]",
    "eventHubName": "[parameters('projectName')]"
  },
  "resources": [
    {
      "type": "Microsoft.EventHub/namespaces",
      "apiVersion": "2021-11-01",
      "name": "[variables('eventHubNamespaceName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('eventHubSku')]",
        "tier": "[parameters('eventHubSku')]",
        "capacity": 1
      },
      "properties": {
        "isAutoInflateEnabled": false,
        "maximumThroughputUnits": 0
      }
    },
    {
      "type": "Microsoft.EventHub/namespaces/eventhubs",
      "apiVersion": "2021-11-01",
      "name": "[format('{0}/{1}', variables('eventHubNamespaceName'),
variables('eventHubName'))]",
      "properties": {
        "messageRetentionInDays": 7,
        "partitionCount": 1
      },
      "dependsOn": [
        "[resourceId('Microsoft.EventHub/namespaces',
variables('eventHubNamespaceName'))]"
      ]
    }
  ]
}
```

The resources defined in the template include:

- Microsoft.EventHub/namespaces
- Microsoft.EventHub/namespaces/eventhubs

To find more template samples, see Azure Quickstart Templates.

Deploy the template

Using Azure portal user interface

1. If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template opens in the Azure portal.

   Deploy to Azure

2. Select an existing **resource group** or create a resource group and select it.

3. Select the **region**.

4. Enter a unique **name** for the **project**. This name is used to generate names for an Event Hubs namespace and an event hub in the namespace.

5. Select **Review + create**.

6. On the **Review + create** page, select **Create**.

Using Azure Cloud Shell

To deploy the template using Azure Cloud Shell:

1. Select **Open Cloud Shell** from the following code block, and then follow the instructions to sign in to the Azure Cloud Shell.

   Azure PowerShell

```
$projectName = Read-Host -Prompt "Enter a project name that is used for
generating resource names"
$location = Read-Host -Prompt "Enter the location (i.e. centralus)"
$resourceGroupName = "${projectName}rg"
$templateUri = "https://raw.githubusercontent.com/Azure/azure-
quickstart-templates/master/quickstarts/microsoft.eventhub/eventhubs-
create-namespace-and-eventhub/azuredeploy.json"

New-AzResourceGroup -Name $resourceGroupName -Location $location
New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName -
```

```
TemplateUri $templateUri -projectName $projectName

Write-Host "Press [ENTER] to continue ..."
```

It takes a few moments to create an event hub.

2. Select **Copy** to copy the PowerShell script.

3. Right-click the shell console, and then select **Paste**.

4. Press **ENTER** to run the commands.

Validate the deployment

To verify the deployment, you can either open the resource group from the Azure portal, or use the following Azure PowerShell script. If the Cloud Shell is still open, you don't need to copy/run the first line (Read-Host).

Azure PowerShell

```
$projectName = Read-Host -Prompt "Enter the same project name that you used
in the last procedure"
$resourceGroupName = "${projectName}rg"
$namespaceName = "${projectName}ns"

Get-AzEventHub -ResourceGroupName $resourceGroupName -Namespace
$namespaceName

Write-Host "Press [ENTER] to continue ..."
```

Clean up resources

When the Azure resources are no longer needed, clean up the resources you deployed by deleting the resource group. If the Cloud Shell is still open, you don't need to copy/run the first line (Read-Host).

Azure PowerShell

```
$projectName = Read-Host -Prompt "Enter the same project name that you used
in the last procedure"
$resourceGroupName = "${projectName}rg"

Remove-AzResourceGroup -ResourceGroupName $resourceGroupName

Write-Host "Press [ENTER] to continue ..."
```

Next steps

In this article, you created an Event Hubs namespace, and an event hub in the namespace. For step-by-step instructions to send events to (or) receive events from an event hub, see the **Send and receive events** tutorials:

- .NET Core
- Java
- Python
- JavaScript
- Go
- C (send only)
- Apache Storm (receive only)

Quickstart: Create Apache Hadoop cluster in Azure HDInsight using ARM template

Article • 04/13/2023

In this quickstart, you use an Azure Resource Manager template (ARM template) to create an Apache Hadoop cluster in Azure HDInsight. Hadoop was the original open-source framework for distributed processing and analysis of big data sets on clusters. The Hadoop ecosystem includes related software and utilities, including Apache Hive, Apache HBase, Spark, Kafka, and many others.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

Currently HDInsight comes with seven different cluster types. Each cluster type supports a different set of components. All cluster types support Hive. For a list of supported components in HDInsight, see What's new in the Hadoop cluster versions provided by HDInsight?

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
```

```
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "12109539600369727322"
    }
  },
  "parameters": {
    "clusterName": {
      "type": "string",
      "metadata": {
        "description": "The name of the HDInsight cluster to create."
      }
    },
    "clusterType": {
      "type": "string",
      "allowedValues": [
        "hadoop",
        "intractivehive",
        "hbase",
        "storm",
        "spark"
      ],
      "metadata": {
        "description": "The type of the HDInsight cluster to create."
      }
    },
    "clusterLoginUserName": {
      "type": "string",
      "metadata": {
        "description": "These credentials can be used to submit jobs to the
cluster and to log into cluster dashboards."
      }
    },
    "clusterLoginPassword": {
      "type": "secureString",
      "minLength": 10,
      "metadata": {
        "description": "The password must be at least 10 characters in
length and must contain at least one digit, one upper case letter, one lower
case letter, and one non-alphanumeric character except (single-quote,
double-quote, backslash, right-bracket, full-stop). Also, the password must
not contain 3 consecutive characters from the cluster username or SSH
username."
      }
    },
    "sshUserName": {
      "type": "string",
      "metadata": {
        "description": "These credentials can be used to remotely access the
cluster. The username cannot be admin."
      }
    },
    "sshPassword": {
```

```
      "type": "secureString",
      "maxLength": 72,
      "minLength": 6,
      "metadata": {
        "description": "SSH password must be 6-72 characters long and must
contain at least one digit, one upper case letter, and one lower case
letter.  It must not contain any 3 consecutive characters from the cluster
login name"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "HeadNodeVirtualMachineSize": {
      "type": "string",
      "defaultValue": "Standard_E4_v3",
      "allowedValues": [
        "Standard_A4_v2",
        "Standard_A8_v2",
        "Standard_E2_v3",
        "Standard_E4_v3",
        "Standard_E8_v3",
        "Standard_E16_v3",
        "Standard_E20_v3",
        "Standard_E32_v3",
        "Standard_E48_v3"
      ],
      "metadata": {
        "description": "This is the headnode Azure Virtual Machine size, and
will affect the cost. If you don't know, just leave the default value."
      }
    },
    "WorkerNodeVirtualMachineSize": {
      "type": "string",
      "defaultValue": "Standard_E4_v3",
      "allowedValues": [
        "Standard_A4_v2",
        "Standard_A8_v2",
        "Standard_E2_v3",
        "Standard_E4_v3",
        "Standard_E8_v3",
        "Standard_E16_v3",
        "Standard_E20_v3",
        "Standard_E32_v3",
        "Standard_E48_v3"
      ],
      "metadata": {
        "description": "This is the workdernode Azure Virtual Machine size,
and will affect the cost. If you don't know, just leave the default value."
      }
    }
```

```
  },
  "variables": {
    "defaultStorageAccount": {
      "name": "[uniqueString(resourceGroup().id)]",
      "type": "Standard_LRS"
    }
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-08-01",
      "name": "[variables('defaultStorageAccount').name]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[variables('defaultStorageAccount').type]"
      },
      "kind": "StorageV2",
      "properties": {}
    },
    {
      "type": "Microsoft.HDInsight/clusters",
      "apiVersion": "2021-06-01",
      "name": "[parameters('clusterName')]",
      "location": "[parameters('location')]",
      "properties": {
        "clusterVersion": "4.0",
        "osType": "Linux",
        "clusterDefinition": {
          "kind": "[parameters('clusterType')]",
          "configurations": {
            "gateway": {
              "restAuthCredential.isEnabled": true,
              "restAuthCredential.username": "
[parameters('clusterLoginUserName')]",
              "restAuthCredential.password": "
[parameters('clusterLoginPassword')]"
            }
          }
        },
        "storageProfile": {
          "storageaccounts": [
            {
              "name": "
[replace(replace(concat(reference(resourceId('Microsoft.Storage/storageAccou
nts', variables('defaultStorageAccount').name)).primaryEndpoints.blob),
'https:', ''), '/', '')]",
              "isDefault": true,
              "container": "[parameters('clusterName')]",
              "key": "
[listKeys(resourceId('Microsoft.Storage/storageAccounts',
variables('defaultStorageAccount').name), '2021-08-01').keys[0].value]"
            }
          ]
        },
        "computeProfile": {
```

```
          "roles": [
            {
              "name": "headnode",
              "targetInstanceCount": 2,
              "hardwareProfile": {
                "vmSize": "[parameters('HeadNodeVirtualMachineSize')]"
              },
              "osProfile": {
                "linuxOperatingSystemProfile": {
                  "username": "[parameters('sshUserName')]",
                  "password": "[parameters('sshPassword')]"
                }
              }
            },
            {
              "name": "workernode",
              "targetInstanceCount": 2,
              "hardwareProfile": {
                "vmSize": "[parameters('WorkerNodeVirtualMachineSize')]"
              },
              "osProfile": {
                "linuxOperatingSystemProfile": {
                  "username": "[parameters('sshUserName')]",
                  "password": "[parameters('sshPassword')]"
                }
              }
            }
          ]
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Storage/storageAccounts',
variables('defaultStorageAccount').name)]"
      ]
    }
  ],
  "outputs": {
    "storage": {
      "type": "object",
      "value": "[reference(resourceId('Microsoft.Storage/storageAccounts',
variables('defaultStorageAccount').name))]"
    },
    "cluster": {
      "type": "object",
      "value": "[reference(resourceId('Microsoft.HDInsight/clusters',
parameters('clusterName')))]"
    }
  }
}
```

Two Azure resources are defined in the template:

- Microsoft.Storage/storageAccounts: create an Azure Storage Account.

- Microsoft.HDInsight/cluster: create an HDInsight cluster.

Deploy the template

1. Select the **Deploy to Azure** button below to sign in to Azure and open the ARM template.

   Deploy to Azure

2. Enter or select the following values:

Property	Description
Subscription	From the drop-down list, select the Azure subscription that's used for the cluster.
Resource group	From the drop-down list, select your existing resource group, or select **Create new**.
Location	The value will autopopulate with the location used for the resource group.
Cluster Name	Enter a globally unique name. For this template, use only lowercase letters, and numbers.
Cluster Type	Select **hadoop**.
Cluster Login User Name	Provide the username, default is `admin`.
Cluster Login Password	Provide a password. The password must be at least 10 characters in length and must contain at least one digit, one uppercase, and one lower case letter, one non-alphanumeric character (except characters `' ` "`).
Ssh User Name	Provide the username, default is `sshuser`.
Ssh Password	Provide the password.

Some properties have been hardcoded in the template. You can configure these values from the template. For more explanation of these properties, see Create Apache Hadoop clusters in HDInsight.

> **Note**

The values you provide must be unique and should follow the naming guidelines. The template does not perform validation checks. If the values you provide are already in use, or do not follow the guidelines, you get an error after you have submitted the template.

3. Review the **TERMS AND CONDITIONS**. Then select **I agree to the terms and conditions stated above**, then **Purchase**. You'll receive a notification that your deployment is in progress. It takes about 20 minutes to create a cluster.

Review deployed resources

Once the cluster is created, you'll receive a **Deployment succeeded** notification with a **Go to resource** link. Your Resource group page will list your new HDInsight cluster and the default storage associated with the cluster. Each cluster has an Azure Blob Storage account, an Azure Data Lake Storage Gen1, or an Azure Data Lake Storage Gen2 dependency. It's referred as the default storage account. The HDInsight cluster and its

default storage account must be colocated in the same Azure region. Deleting clusters doesn't delete the storage account.

> **Note**
>
> For other cluster creation methods and understanding the properties used in this quickstart, see **Create HDInsight clusters**.

Clean up resources

After you complete the quickstart, you may want to delete the cluster. With HDInsight, your data is stored in Azure Storage, so you can safely delete a cluster when it isn't in use. You're also charged for an HDInsight cluster, even when it isn't in use. Since the charges for the cluster are many times more than the charges for storage, it makes economic sense to delete clusters when they aren't in use.

> **Note**
>
> If you are *immediately* proceeding to the next tutorial to learn how to run ETL operations using Hadoop on HDInsight, you may want to keep the cluster running. This is because in the tutorial you have to create a Hadoop cluster again. However, if you are not going through the next tutorial right away, you must delete the cluster now.

From the Azure portal, navigate to your cluster, and select **Delete**.

You can also select the resource group name to open the resource group page, and then select **Delete resource group**. By deleting the resource group, you delete both the HDInsight cluster, and the default storage account.

Next steps

In this quickstart, you learned how to create an Apache Hadoop cluster in HDInsight using an ARM template. In the next article, you learn how to perform an extract, transform, and load (ETL) operation using Hadoop on HDInsight.

Extract, transform, and load data using Interactive Query on HDInsight

Quickstart: Create Apache HBase cluster in Azure HDInsight using ARM template

Article • 04/13/2023

In this quickstart, you use an Azure Resource Manager template (ARM template) to create an Apache HBase cluster in Azure HDInsight. HBase is an open-source, NoSQL database that is built on Apache Hadoop and modeled after Google BigTable.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "8546919507151042 44"
    }
  },
  "parameters": {
    "clusterName": {
      "type": "string",
```

```
      "metadata": {
        "description": "The name of the HDInsight cluster to create."
      }
    },
    "clusterLoginUserName": {
      "type": "string",
      "metadata": {
        "description": "These credentials can be used to submit jobs to the
cluster and to log into cluster dashboards."
      }
    },
    "clusterLoginPassword": {
      "type": "secureString",
      "minLength": 10,
      "metadata": {
        "description": "The password must be at least 10 characters in
length and must contain at least one digit, one upper case letter, one lower
case letter, and one non-alphanumeric character except (single-quote,
double-quote, backslash, right-bracket, full-stop). Also, the password must
not contain 3 consecutive characters from the cluster username or SSH
username."
      }
    },
    "sshUserName": {
      "type": "string",
      "metadata": {
        "description": "These credentials can be used to remotely access the
cluster."
      }
    },
    "sshPassword": {
      "type": "secureString",
      "maxLength": 72,
      "minLength": 6,
      "metadata": {
        "description": "SSH password must be 6-72 characters long and must
contain at least one digit, one upper case letter, and one lower case
letter.  It must not contain any 3 consecutive characters from the cluster
login name"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "HeadNodeVirtualMachineSize": {
      "type": "string",
      "defaultValue": "Standard_E4_v3",
      "allowedValues": [
        "Standard_A4_v2",
        "Standard_A8_v2",
        "Standard_E2_v3",
```

```
        "Standard_E4_v3",
        "Standard_E8_v3",
        "Standard_E16_v3",
        "Standard_E20_v3",
        "Standard_E32_v3",
        "Standard_E48_v3"
      ],
      "metadata": {
        "description": "This is the headnode Azure Virtual Machine size, and
will affect the cost. If you don't know, just leave the default value."
      }
    },
    "WorkerNodeVirtualMachineSize": {
      "type": "string",
      "defaultValue": "Standard_E4_v3",
      "allowedValues": [
        "Standard_A4_v2",
        "Standard_A8_v2",
        "Standard_E2_v3",
        "Standard_E4_v3",
        "Standard_E8_v3",
        "Standard_E16_v3",
        "Standard_E20_v3",
        "Standard_E32_v3",
        "Standard_E48_v3"
      ],
      "metadata": {
        "description": "This is the worerdnode Azure Virtual Machine size,
and will affect the cost. If you don't know, just leave the default value."
      }
    },
    "ZookeeperNodeVirtualMachineSize": {
      "type": "string",
      "defaultValue": "Standard_E4_v3",
      "allowedValues": [
        "Standard_A4_v2",
        "Standard_A8_v2",
        "Standard_E2_v3",
        "Standard_E4_v3",
        "Standard_E8_v3",
        "Standard_E16_v3",
        "Standard_E20_v3",
        "Standard_E32_v3",
        "Standard_E48_v3"
      ],
      "metadata": {
        "description": "This is the Zookeepernode Azure Virtual Machine
size, and will affect the cost. If you don't know, just leave the default
value."
      }
    }
  },
  "variables": {
    "defaultStorageAccount": {
      "name": "[uniqueString(resourceGroup().id)]",
```

```
      "type": "Standard_LRS"
    }
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-08-01",
      "name": "[variables('defaultStorageAccount').name]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[variables('defaultStorageAccount').type]"
      },
      "kind": "Storage",
      "properties": {}
    },
    {
      "type": "Microsoft.HDInsight/clusters",
      "apiVersion": "2021-06-01",
      "name": "[parameters('clusterName')]",
      "location": "[parameters('location')]",
      "properties": {
        "clusterVersion": "4.0",
        "osType": "Linux",
        "clusterDefinition": {
          "kind": "hbase",
          "configurations": {
            "gateway": {
              "restAuthCredential.isEnabled": true,
              "restAuthCredential.username": "
[parameters('clusterLoginUserName')]",
              "restAuthCredential.password": "
[parameters('clusterLoginPassword')]"
            }
          }
        },
        "storageProfile": {
          "storageaccounts": [
            {
              "name": "
[replace(replace(reference(resourceId('Microsoft.Storage/storageAccounts',
variables('defaultStorageAccount').name), '2021-08-
01').primaryEndpoints.blob, 'https://', ''), '/', '')]",
              "isDefault": true,
              "container": "[parameters('clusterName')]",
              "key": "
[listKeys(resourceId('Microsoft.Storage/storageAccounts',
variables('defaultStorageAccount').name), '2021-08-01').keys[0].value]"
            }
          ]
        },
        "computeProfile": {
          "roles": [
            {
              "name": "headnode",
              "targetInstanceCount": 2,
```

```
              "hardwareProfile": {
                "vmSize": "[parameters('HeadNodeVirtualMachineSize')]"
              },
              "osProfile": {
                "linuxOperatingSystemProfile": {
                  "username": "[parameters('sshUserName')]",
                  "password": "[parameters('sshPassword')]"
                }
              }
            },
            {
              "name": "workernode",
              "targetInstanceCount": 2,
              "hardwareProfile": {
                "vmSize": "[parameters('WorkerNodeVirtualMachineSize')]"
              },
              "osProfile": {
                "linuxOperatingSystemProfile": {
                  "username": "[parameters('sshUserName')]",
                  "password": "[parameters('sshPassword')]"
                }
              }
            },
            {
              "name": "zookeepernode",
              "targetInstanceCount": 3,
              "hardwareProfile": {
                "vmSize": "[parameters('ZookeeperNodeVirtualMachineSize')]"
              },
              "osProfile": {
                "linuxOperatingSystemProfile": {
                  "username": "[parameters('sshUserName')]",
                  "password": "[parameters('sshPassword')]"
                }
              }
            }
          ]
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Storage/storageAccounts',
variables('defaultStorageAccount').name)]"
      ]
    }
  ],
  "outputs": {
    "cluster": {
      "type": "object",
      "value": "[reference(resourceId('Microsoft.HDInsight/clusters',
parameters('clusterName')))]"
    }
  }
}
```

Two Azure resources are defined in the template:

- Microsoft.Storage/storageAccounts: create an Azure Storage Account.
- Microsoft.HDInsight/cluster: create an HDInsight cluster.

Deploy the template

1. Select the **Deploy to Azure** button below to sign in to Azure and open the ARM template.

   Deploy to Azure

2. Enter or select the following values:

Property	Description
Subscription	From the drop-down list, select the Azure subscription that's used for the cluster.
Resource group	From the drop-down list, select your existing resource group, or select **Create new**.
Location	The value will autopopulate with the location used for the resource group.
Cluster Name	Enter a globally unique name. For this template, use only lowercase letters, and numbers.
Cluster Login User Name	Provide the username, default is `admin`.
Cluster Login Password	Provide a password. The password must be at least 10 characters in length and must contain at least one digit, one uppercase, and one lower case letter, one non-alphanumeric character (except characters `' ` "`).
Ssh User Name	Provide the username, default is `sshuser`.
Ssh Password	Provide the password.

3. Review the **TERMS AND CONDITIONS**. Then select **I agree to the terms and conditions stated above**, then **Purchase**. You'll receive a notification that your deployment is in progress. It takes about 20 minutes to create a cluster.

Review deployed resources

Once the cluster is created, you'll receive a **Deployment succeeded** notification with a **Go to resource** link. Your Resource group page will list your new HDInsight cluster and

the default storage associated with the cluster. Each cluster has an Azure Blob Storage account, an Azure Data Lake Storage Gen1, or an Azure Data Lake Storage Gen2 dependency. It's referred as the default storage account. The HDInsight cluster and its default storage account must be colocated in the same Azure region. Deleting clusters doesn't delete the storage account.

Clean up resources

After you complete the quickstart, you may want to delete the cluster. With HDInsight, your data is stored in Azure Storage, so you can safely delete a cluster when it isn't in use. You're also charged for an HDInsight cluster, even when it isn't in use. Since the charges for the cluster are many times more than the charges for storage, it makes economic sense to delete clusters when they aren't in use.

From the Azure portal, navigate to your cluster, and select **Delete**.

Delete Resource Manager template HBase

You can also select the resource group name to open the resource group page, and then select **Delete resource group**. By deleting the resource group, you delete both the HDInsight cluster, and the default storage account.

Next steps

In this quickstart, you learned how to create an Apache HBase cluster in HDInsight using an ARM template. In the next article, you learn how to query HBase in HDInsight with HBase Shell.

Query Apache HBase in Azure HDInsight with HBase Shell

Quickstart: Create Interactive Query cluster in Azure HDInsight using ARM template

Article • 04/13/2023

In this quickstart, you use an Azure Resource Manager template (ARM template) to create an Interactive Query cluster in Azure HDInsight. Interactive Query (also called Apache Hive LLAP, or Low Latency Analytical Processing) is an Azure HDInsight cluster type.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "16722292906964971415"
    }
```

```
  },
  "parameters": {
    "clusterName": {
      "type": "string",
      "metadata": {
        "description": "The name of the HDInsight cluster to create."
      }
    },
    "clusterLoginUserName": {
      "type": "string",
      "metadata": {
        "description": "These credentials can be used to submit jobs to the
cluster and to log into cluster dashboards."
      }
    },
    "clusterLoginPassword": {
      "type": "secureString",
      "minLength": 10,
      "metadata": {
        "description": "The password must be at least 10 characters in
length and must contain at least one digit, one upper case letter, one lower
case letter, and one non-alphanumeric character except (single-quote,
double-quote, backslash, right-bracket, full-stop). Also, the password must
not contain 3 consecutive characters from the cluster username or SSH
username."
      }
    },
    "sshUserName": {
      "type": "string",
      "metadata": {
        "description": "These credentials can be used to remotely access the
cluster."
      }
    },
    "sshPassword": {
      "type": "secureString",
      "maxLength": 72,
      "minLength": 6,
      "metadata": {
        "description": "SSH password must be 6-72 characters long and must
contain at least one digit, one upper case letter, and one lower case
letter.  It must not contain any 3 consecutive characters from the cluster
login name"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "HeadNodeVirtualMachineSize": {
      "type": "string",
      "defaultValue": "Standard_E8_v3",
```

```
      "allowedValues": [
        "Standard_A4_v2",
        "Standard_A8_v2",
        "Standard_E2_v3",
        "Standard_E4_v3",
        "Standard_E8_v3",
        "Standard_E16_v3",
        "Standard_E20_v3",
        "Standard_E32_v3",
        "Standard_E48_v3"
      ],
      "metadata": {
        "description": "This is the headnode Azure Virtual Machine size, and
will affect the cost. If you don't know, just leave the default value."
      }
    },
    "WorkerNodeVirtualMachineSize": {
      "type": "string",
      "defaultValue": "Standard_E16_v3",
      "allowedValues": [
        "Standard_A4_v2",
        "Standard_A8_v2",
        "Standard_E2_v3",
        "Standard_E4_v3",
        "Standard_E8_v3",
        "Standard_E16_v3",
        "Standard_E20_v3",
        "Standard_E32_v3",
        "Standard_E48_v3"
      ],
      "metadata": {
        "description": "This is the worerdnode Azure Virtual Machine size,
and will affect the cost. If you don't know, just leave the default value."
      }
    },
    "ZookeeperNodeVirtualMachineSize": {
      "type": "string",
      "defaultValue": "Standard_E4_v3",
      "allowedValues": [
        "Standard_A4_v2",
        "Standard_A8_v2",
        "Standard_E2_v3",
        "Standard_E4_v3",
        "Standard_E8_v3",
        "Standard_E16_v3",
        "Standard_E20_v3",
        "Standard_E32_v3",
        "Standard_E48_v3"
      ],
      "metadata": {
        "description": "This is the worerdnode Azure Virtual Machine size,
and will affect the cost. If you don't know, just leave the default value."
      }
    }
  },
```

```
  "variables": {
    "defaultStorageAccount": {
      "name": "[uniqueString(resourceGroup().id)]",
      "type": "Standard_LRS"
    }
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-08-01",
      "name": "[variables('defaultStorageAccount').name]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[variables('defaultStorageAccount').type]"
      },
      "kind": "StorageV2",
      "properties": {}
    },
    {
      "type": "Microsoft.HDInsight/clusters",
      "apiVersion": "2021-06-01",
      "name": "[parameters('clusterName')]",
      "location": "[parameters('location')]",
      "properties": {
        "clusterVersion": "4.0",
        "osType": "Linux",
        "tier": "Standard",
        "clusterDefinition": {
          "kind": "interactivehive",
          "configurations": {
            "gateway": {
              "restAuthCredential.isEnabled": true,
              "restAuthCredential.username": "
[parameters('clusterLoginUserName')]",
              "restAuthCredential.password": "
[parameters('clusterLoginPassword')]"
            }
          }
        },
        "storageProfile": {
          "storageaccounts": [
            {
              "name": "
[replace(replace(concat(reference(resourceId('Microsoft.Storage/storageAccou
nts', variables('defaultStorageAccount').name), '2021-08-
01').primaryEndpoints.blob), 'https:', ''), '/', '')]",
              "isDefault": true,
              "container": "[parameters('clusterName')]",
              "key": "
[listKeys(resourceId('Microsoft.Storage/storageAccounts',
variables('defaultStorageAccount').name), '2021-08-01').keys[0].value]"
            }
          ]
        },
        "computeProfile": {
```

```
          "roles": [
            {
              "name": "headnode",
              "minInstanceCount": 1,
              "targetInstanceCount": 2,
              "hardwareProfile": {
                "vmSize": "[parameters('HeadNodeVirtualMachineSize')]"
              },
              "osProfile": {
                "linuxOperatingSystemProfile": {
                  "username": "[parameters('sshUserName')]",
                  "password": "[parameters('sshPassword')]"
                }
              }
            },
            {
              "name": "workernode",
              "minInstanceCount": 1,
              "targetInstanceCount": 2,
              "hardwareProfile": {
                "vmSize": "[parameters('WorkerNodeVirtualMachineSize')]"
              },
              "osProfile": {
                "linuxOperatingSystemProfile": {
                  "username": "[parameters('sshUserName')]",
                  "password": "[parameters('sshPassword')]"
                }
              }
            },
            {
              "name": "zookeepernode",
              "minInstanceCount": 1,
              "targetInstanceCount": 3,
              "hardwareProfile": {
                "vmSize": "[parameters('ZookeeperNodeVirtualMachineSize')]"
              },
              "osProfile": {
                "linuxOperatingSystemProfile": {
                  "username": "[parameters('sshUserName')]",
                  "password": "[parameters('sshPassword')]"
                }
              }
            }
          ]
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Storage/storageAccounts',
variables('defaultStorageAccount').name)]"
      ]
    }
  ],
  "outputs": {
    "storage": {
      "type": "object",
```

```
      "value": "[reference(resourceId('Microsoft.Storage/storageAccounts',
variables('defaultStorageAccount').name))]"
    },
    "cluster": {
      "type": "object",
      "value": "[reference(resourceId('Microsoft.HDInsight/clusters',
parameters('clusterName')))]"
    }
  }
}
```

Two Azure resources are defined in the template:

- Microsoft.Storage/storageAccounts: create an Azure Storage Account.
- Microsoft.HDInsight/cluster: create an HDInsight cluster.

Deploy the template

1. Select the **Deploy to Azure** button below to sign in to Azure and open the ARM template.

   Deploy to Azure

2. Enter or select the following values:

Property	Description
Subscription	From the drop-down list, select the Azure subscription that's used for the cluster.
Resource group	From the drop-down list, select your existing resource group, or select **Create new**.
Location	The value will autopopulate with the location used for the resource group.
Cluster Name	Enter a globally unique name. For this template, use only lowercase letters, and numbers.
Cluster Login User Name	Provide the username, default is `admin`.
Cluster Login Password	Provide a password. The password must be at least 10 characters in length and must contain at least one digit, one uppercase, and one lower case letter, one non-alphanumeric character (except characters `' ` "`).
Ssh User Name	Provide the username, default is sshuser

Property	Description
Ssh Password	Provide the password.

3. Review the **TERMS AND CONDITIONS**. Then select **I agree to the terms and conditions stated above**, then **Purchase**. You'll receive a notification that your

deployment is in progress. It takes about 20 minutes to create a cluster.

Review deployed resources

Once the cluster is created, you'll receive a **Deployment succeeded** notification with a **Go to resource** link. Your Resource group page will list your new HDInsight cluster and the default storage associated with the cluster. Each cluster has an Azure Blob Storage account, an Azure Data Lake Storage Gen1, or an Azure Data Lake Storage Gen2 dependency. It's referred as the default storage account. The HDInsight cluster and its default storage account must be colocated in the same Azure region. Deleting clusters doesn't delete the storage account.

Clean up resources

After you complete the quickstart, you may want to delete the cluster. With HDInsight, your data is stored in Azure Storage, so you can safely delete a cluster when it isn't in use. You're also charged for an HDInsight cluster, even when it isn't in use. Since the charges for the cluster are many times more than the charges for storage, it makes economic sense to delete clusters when they aren't in use.

From the Azure portal, navigate to your cluster, and select **Delete**.

Delete Resource Manager template HBase

You can also select the resource group name to open the resource group page, and then select **Delete resource group**. By deleting the resource group, you delete both the HDInsight cluster, and the default storage account.

Next steps

In this quickstart, you learned how to create an Interactive Query cluster in HDInsight using an ARM template. In the next article, you learn how to use Apache Zeppelin to run Apache Hive queries.

Execute Apache Hive queries in Azure HDInsight with Apache Zeppelin

Quickstart: Create Apache Kafka cluster in Azure HDInsight using ARM template

Article • 10/21/2022

In this quickstart, you use an Azure Resource Manager template (ARM template) to create an Apache Kafka cluster in Azure HDInsight. Kafka is an open-source, distributed streaming platform. It's often used as a message broker, as it provides functionality similar to a publish-subscribe message queue.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

The Kafka API can only be accessed by resources inside the same virtual network. In this quickstart, you access the cluster directly using SSH. To connect other services, networks, or virtual machines to Kafka, you must first create a virtual network and then create the resources within the network. For more information, see the Connect to Apache Kafka using a virtual network document.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
```

```
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "15765736955222338858"
    }
  },
  "parameters": {
    "clusterName": {
      "type": "string",
      "metadata": {
        "description": "The name of the Kafka cluster to create. This must
be a unique name."
      }
    },
    "clusterLoginUserName": {
      "type": "string",
      "metadata": {
        "description": "These credentials can be used to submit jobs to the
cluster and to log into cluster dashboards."
      }
    },
    "clusterLoginPassword": {
      "type": "secureString",
      "minLength": 10,
      "metadata": {
        "description": "The password must be at least 10 characters in
length and must contain at least one digit, one upper case letter, one lower
case letter, and one non-alphanumeric character except (single-quote,
double-quote, backslash, right-bracket, full-stop). Also, the password must
not contain 3 consecutive characters from the cluster username or SSH
username."
      }
    },
    "sshUserName": {
      "type": "string",
      "metadata": {
        "description": "These credentials can be used to remotely access the
cluster."
      }
    },
    "sshPassword": {
      "type": "secureString",
      "maxLength": 72,
      "minLength": 6,
      "metadata": {
        "description": "SSH password must be 6-72 characters long and must
contain at least one digit, one upper case letter, and one lower case
letter.  It must not contain any 3 consecutive characters from the cluster
login name"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
```

```
        "description": "Location for all resources."
      }
    },
    "HeadNodeVirtualMachineSize": {
      "type": "string",
      "defaultValue": "Standard_E4_v3",
      "allowedValues": [
        "Standard_A4_v2",
        "Standard_A8_v2",
        "Standard_E2_v3",
        "Standard_E4_v3",
        "Standard_E8_v3",
        "Standard_E16_v3",
        "Standard_E20_v3",
        "Standard_E32_v3",
        "Standard_E48_v3"
      ],
      "metadata": {
        "description": "This is the headnode Azure Virtual Machine size, and
will affect the cost. If you don't know, just leave the default value."
      }
    },
    "WorkerNodeVirtualMachineSize": {
      "type": "string",
      "defaultValue": "Standard_E4_v3",
      "allowedValues": [
        "Standard_A4_v2",
        "Standard_A8_v2",
        "Standard_E2_v3",
        "Standard_E4_v3",
        "Standard_E8_v3",
        "Standard_E16_v3",
        "Standard_E20_v3",
        "Standard_E32_v3",
        "Standard_E48_v3"
      ],
      "metadata": {
        "description": "This is the worerdnode Azure Virtual Machine size,
and will affect the cost. If you don't know, just leave the default value."
      }
    },
    "ZookeeperNodeVirtualMachineSize": {
      "type": "string",
      "defaultValue": "Standard_E4_v3",
      "allowedValues": [
        "Standard_A4_v2",
        "Standard_A8_v2",
        "Standard_E2_v3",
        "Standard_E4_v3",
        "Standard_E8_v3",
        "Standard_E16_v3",
        "Standard_E20_v3",
        "Standard_E32_v3",
        "Standard_E48_v3"
      ],
```

```
      "metadata": {
        "description": "This is the Zookeepernode Azure Virtual Machine
size, and will affect the cost. If you don't know, just leave the default
value."
      }
    }
  },
  "variables": {
    "defaultStorageAccount": {
      "name": "[uniqueString(resourceGroup().id)]",
      "type": "Standard_LRS"
    }
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-08-01",
      "name": "[variables('defaultStorageAccount').name]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[variables('defaultStorageAccount').type]"
      },
      "kind": "StorageV2",
      "properties": {}
    },
    {
      "type": "Microsoft.HDInsight/clusters",
      "apiVersion": "2021-06-01",
      "name": "[parameters('clusterName')]",
      "location": "[parameters('location')]",
      "properties": {
        "clusterVersion": "4.0",
        "osType": "Linux",
        "clusterDefinition": {
          "kind": "kafka",
          "configurations": {
            "gateway": {
              "restAuthCredential.isEnabled": true,
              "restAuthCredential.username": "
[parameters('clusterLoginUserName')]",
              "restAuthCredential.password": "
[parameters('clusterLoginPassword')]"
            }
          }
        },
        "storageProfile": {
          "storageaccounts": [
            {
              "name": "
[replace(replace(concat(reference(resourceId('Microsoft.Storage/storageAccou
nts', variables('defaultStorageAccount').name), '2021-08-
01').primaryEndpoints.blob), 'https:', ''), '/', '')]",
              "isDefault": true,
              "container": "[parameters('clusterName')]",
              "key": "
```

```
[listKeys(resourceId('Microsoft.Storage/storageAccounts',
variables('defaultStorageAccount').name), '2021-08-01').keys[0].value]"
            }
          ]
        },
        "computeProfile": {
          "roles": [
            {
              "name": "headnode",
              "targetInstanceCount": 2,
              "hardwareProfile": {
                "vmSize": "[parameters('HeadNodeVirtualMachineSize')]"
              },
              "osProfile": {
                "linuxOperatingSystemProfile": {
                  "username": "[parameters('sshUserName')]",
                  "password": "[parameters('sshPassword')]"
                }
              }
            },
            {
              "name": "workernode",
              "targetInstanceCount": 4,
              "hardwareProfile": {
                "vmSize": "[parameters('WorkerNodeVirtualMachineSize')]"
              },
              "dataDisksGroups": [
                {
                  "disksPerNode": 2
                }
              ],
              "osProfile": {
                "linuxOperatingSystemProfile": {
                  "username": "[parameters('sshUserName')]",
                  "password": "[parameters('sshPassword')]"
                }
              }
            },
            {
              "name": "zookeepernode",
              "targetInstanceCount": 3,
              "hardwareProfile": {
                "vmSize": "[parameters('ZookeeperNodeVirtualMachineSize')]"
              },
              "osProfile": {
                "linuxOperatingSystemProfile": {
                  "username": "[parameters('sshUserName')]",
                  "password": "[parameters('sshPassword')]"
                }
              }
            }
          ]
        }
      },
      "dependsOn": [
```

```
        "[resourceId('Microsoft.Storage/storageAccounts',
variables('defaultStorageAccount').name)]"
      ]
    }
  ],
  "outputs": {
    "cluster": {
      "type": "object",
      "value": "[reference(resourceId('Microsoft.HDInsight/clusters',
parameters('clusterName')))]"
    }
  }
}
```

Two Azure resources are defined in the template:

- Microsoft.Storage/storageAccounts: create an Azure Storage Account.
- Microsoft.HDInsight/cluster: create an HDInsight cluster.

Deploy the template

1. Select the **Deploy to Azure** button below to sign in to Azure and open the ARM template.

   Deploy to Azure

2. Enter or select the following values:

Property	Description
Subscription	From the drop-down list, select the Azure subscription that's used for the cluster.
Resource group	From the drop-down list, select your existing resource group, or select **Create new**.
Location	The value will autopopulate with the location used for the resource group.
Cluster Name	Enter a globally unique name. For this template, use only lowercase letters, and numbers.
Cluster Login User Name	Provide the username, default is `admin`.
Cluster Login Password	Provide a password. The password must be at least 10 characters in length and must contain at least one digit, one uppercase, and one lower case letter, one non-alphanumeric character (except characters `' ` "`).

Property	Description
Ssh User Name	Provide the username, default is `sshuser`.
Ssh Password	Provide the password.

3. Review the **TERMS AND CONDITIONS**. Then select **I agree to the terms and conditions stated above**, then **Purchase**. You'll receive a notification that your deployment is in progress. It takes about 20 minutes to create a cluster.

Review deployed resources

Once the cluster is created, you'll receive a **Deployment succeeded** notification with a **Go to resource** link. Your Resource group page will list your new HDInsight cluster and the default storage associated with the cluster. Each cluster has an Azure Blob Storage account, an Azure Data Lake Storage Gen1, or an Azure Data Lake Storage Gen2 dependency. It's referred as the default storage account. The HDInsight cluster and its default storage account must be colocated in the same Azure region. Deleting clusters doesn't delete the storage account.

Get the Apache Zookeeper and Broker host information

When working with Kafka, you must know the *Apache Zookeeper* and *Broker* hosts. These hosts are used with the Kafka API and many of the utilities that ship with Kafka.

In this section, you get the host information from the Ambari REST API on the cluster.

1. Use ssh command to connect to your cluster. Edit the command below by replacing CLUSTERNAME with the name of your cluster, and then enter the command:

   Windows Command Prompt

   ```
   ssh sshuser@CLUSTERNAME-ssh.azurehdinsight.net
   ```

2. From the SSH connection, use the following command to install the `jq` utility. This utility is used to parse JSON documents, and is useful in retrieving the host information:

   Bash

   ```
   sudo apt -y install jq
   ```

3. To set an environment variable to the cluster name, use the following command:

   Bash

```
read -p "Enter the Kafka on HDInsight cluster name: " CLUSTERNAME
```

When prompted, enter the name of the Kafka cluster.

4. To set an environment variable with Zookeeper host information, use the command below. The command retrieves all Zookeeper hosts, then returns only the first two entries. This is because you want some redundancy in case one host is unreachable.

```bash
export KAFKAZKHOSTS=`curl -sS -u admin -G
https://$CLUSTERNAME.azurehdinsight.net/api/v1/clusters/$CLUSTERNAME/se
rvices/ZOOKEEPER/components/ZOOKEEPER_SERVER | jq -r '["\
(.host_components[].HostRoles.host_name):2181"] | join(",")' | cut -
d',' -f1,2`
```

When prompted, enter the password for the cluster login account (not the SSH account).

5. To verify that the environment variable is set correctly, use the following command:

```bash
echo '$KAFKAZKHOSTS='$KAFKAZKHOSTS
```

This command returns information similar to the following text:

`<zookeepername1>.eahjefxxp1netdbyklgqj5y1ud.ex.internal.cloudapp.net:2181,`
`<zookeepername2>.eahjefxxp1netdbyklgqj5y1ud.ex.internal.cloudapp.net:2181`

6. To set an environment variable with Kafka broker host information, use the following command:

```bash
export KAFKABROKERS=`curl -sS -u admin -G
https://$CLUSTERNAME.azurehdinsight.net/api/v1/clusters/$CLUSTERNAME/se
rvices/KAFKA/components/KAFKA_BROKER | jq -r '["\
(.host_components[].HostRoles.host_name):9092"] | join(",")' | cut -
d',' -f1,2`
```

When prompted, enter the password for the cluster login account (not the SSH account).

7. To verify that the environment variable is set correctly, use the following command:

```bash
echo '$KAFKABROKERS='$KAFKABROKERS
```

This command returns information similar to the following text:

`<brokername1>.eahjefxxp1netdbyklgqj5y1ud.cx.internal.cloudapp.net:9092,`
`<brokername2>.eahjefxxp1netdbyklgqj5y1ud.cx.internal.cloudapp.net:9092`

Manage Apache Kafka topics

Kafka stores streams of data in *topics*. You can use the `kafka-topics.sh` utility to manage topics.

- **To create a topic**, use the following command in the SSH connection:

```bash
/usr/hdp/current/kafka-broker/bin/kafka-topics.sh --create --
replication-factor 3 --partitions 8 --topic test --zookeeper
$KAFKAZKHOSTS
```

This command connects to Zookeeper using the host information stored in `$KAFKAZKHOSTS`. It then creates a Kafka topic named **test**.

  - Data stored in this topic is partitioned across eight partitions.
  - Each partition is replicated across three worker nodes in the cluster.

    If you created the cluster in an Azure region that provides three fault domains, use a replication factor of 3. Otherwise, use a replication factor of 4.

    In regions with three fault domains, a replication factor of 3 allows replicas to be spread across the fault domains. In regions with two fault domains, a replication factor of four spreads the replicas evenly across the domains.

    For information on the number of fault domains in a region, see the Availability of Linux virtual machines document.

    Kafka isn't aware of Azure fault domains. When creating partition replicas for topics, it may not distribute replicas properly for high availability.

To ensure high availability, use the Apache Kafka partition rebalance tool. This tool must be ran from an SSH connection to the head node of your Kafka cluster.

For the highest availability of your Kafka data, you should rebalance the partition replicas for your topic when:

- You create a new topic or partition
- You scale up a cluster

- **To list topics**, use the following command:

  ```bash
  /usr/hdp/current/kafka-broker/bin/kafka-topics.sh --list --zookeeper $KAFKAZKHOSTS
  ```

  This command lists the topics available on the Kafka cluster.

- **To delete a topic**, use the following command:

  ```bash
  /usr/hdp/current/kafka-broker/bin/kafka-topics.sh --delete --topic topicname --zookeeper $KAFKAZKHOSTS
  ```

  This command deletes the topic named `topicname`.

  > ⚠ **Warning**
  >
  > If you delete the `test` topic created earlier, then you must recreate it. It is used by steps later in this document.

For more information on the commands available with the `kafka-topics.sh` utility, use the following command:

```bash
/usr/hdp/current/kafka-broker/bin/kafka-topics.sh
```

Produce and consume records

Kafka stores *records* in topics. Records are produced by *producers*, and consumed by *consumers*. Producers and consumers communicate with the *Kafka broker* service. Each worker node in your HDInsight cluster is a Kafka broker host.

To store records into the test topic you created earlier, and then read them using a consumer, use the following steps:

1. To write records to the topic, use the `kafka-console-producer.sh` utility from the SSH connection:

   ```bash
   /usr/hdp/current/kafka-broker/bin/kafka-console-producer.sh --broker-list $KAFKABROKERS --topic test
   ```

   After this command, you arrive at an empty line.

2. Type a text message on the empty line and hit enter. Enter a few messages this way, and then use **Ctrl + C** to return to the normal prompt. Each line is sent as a separate record to the Kafka topic.

3. To read records from the topic, use the `kafka-console-consumer.sh` utility from the SSH connection:

   ```bash
   /usr/hdp/current/kafka-broker/bin/kafka-console-consumer.sh --bootstrap-server $KAFKABROKERS --topic test --from-beginning
   ```

   This command retrieves the records from the topic and displays them. Using `--from-beginning` tells the consumer to start from the beginning of the stream, so all records are retrieved.

   If you're using an older version of Kafka, replace `--bootstrap-server $KAFKABROKERS` with `--zookeeper $KAFKAZKHOSTS`.

4. Use **Ctrl + C** to stop the consumer.

You can also programmatically create producers and consumers. For an example of using this API, see the Apache Kafka Producer and Consumer API with HDInsight document.

Clean up resources

After you complete the quickstart, you may want to delete the cluster. With HDInsight, your data is stored in Azure Storage, so you can safely delete a cluster when it isn't in use. You're also charged for an HDInsight cluster, even when it isn't in use. Since the charges for the cluster are many times more than the charges for storage, it makes economic sense to delete clusters when they aren't in use.

From the Azure portal, navigate to your cluster, and select **Delete**.

You can also select the resource group name to open the resource group page, and then select **Delete resource group**. By deleting the resource group, you delete both the HDInsight cluster, and the default storage account.

Next steps

In this quickstart, you learned how to create an Apache Kafka cluster in HDInsight using an ARM template. In the next article, you learn how to create an application that uses the Apache Kafka Streams API and run it with Kafka on HDInsight.

Use Apache Kafka streams API in Azure HDInsight

Quickstart: Create Apache Spark cluster in Azure HDInsight using ARM template

Article • 04/13/2023

In this quickstart, you use an Azure Resource Manager template (ARM template) to create an Apache Spark cluster in Azure HDInsight. You then create a Jupyter Notebook file, and use it to run Spark SQL queries against Apache Hive tables. Azure HDInsight is a managed, full-spectrum, open-source analytics service for enterprises. The Apache Spark framework for HDInsight enables fast data analytics and cluster computing using in-memory processing. Jupyter Notebook lets you interact with your data, combine code with markdown text, and do simple visualizations.

If you're using multiple clusters together, you'll want to create a virtual network, and if you're using a Spark cluster you'll also want to use the Hive Warehouse Connector. For more information, see Plan a virtual network for Azure HDInsight and Integrate Apache Spark and Apache Hive with the Hive Warehouse Connector.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
```

```
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "4742950082151195489"
    }
  },
  "parameters": {
    "clusterName": {
      "type": "string",
      "metadata": {
        "description": "The name of the HDInsight cluster to create."
      }
    },
    "clusterLoginUserName": {
      "type": "string",
      "maxLength": 20,
      "minLength": 2,
      "metadata": {
        "description": "These credentials can be used to submit jobs to the
cluster and to log into cluster dashboards. The username must consist of
digits, upper or lowercase letters, and/or the following special characters:
(!#$%&'()-^_`{}~)."
      }
    },
    "clusterLoginPassword": {
      "type": "secureString",
      "minLength": 10,
      "metadata": {
        "description": "The password must be at least 10 characters in
length and must contain at least one digit, one upper case letter, one lower
case letter, and one non-alphanumeric character except (single-quote,
double-quote, backslash, right-bracket, full-stop). Also, the password must
not contain 3 consecutive characters from the cluster username or SSH
username."
      }
    },
    "sshUserName": {
      "type": "string",
      "minLength": 2,
      "metadata": {
        "description": "These credentials can be used to remotely access the
cluster. The sshUserName can only consit of digits, upper or lowercase
letters, and/or the following special characters (%&'^_`{}~). Also, it
cannot be the same as the cluster login username or a reserved word"
      }
    },
    "sshPassword": {
      "type": "secureString",
      "maxLength": 72,
      "minLength": 6,
      "metadata": {
        "description": "SSH password must be 6-72 characters long and must
```

```
contain at least one digit, one upper case letter, and one lower case
letter.  It must not contain any 3 consecutive characters from the cluster
login name"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "headNodeVirtualMachineSize": {
      "type": "string",
      "defaultValue": "Standard_E8_v3",
      "allowedValues": [
        "Standard_A4_v2",
        "Standard_A8_v2",
        "Standard_E2_v3",
        "Standard_E4_v3",
        "Standard_E8_v3",
        "Standard_E16_v3",
        "Standard_E20_v3",
        "Standard_E32_v3",
        "Standard_E48_v3"
      ],
      "metadata": {
        "description": "This is the headnode Azure Virtual Machine size, and
will affect the cost. If you don't know, just leave the default value."
      }
    },
    "workerNodeVirtualMachineSize": {
      "type": "string",
      "defaultValue": "Standard_E8_v3",
      "allowedValues": [
        "Standard_A4_v2",
        "Standard_A8_v2",
        "Standard_E2_v3",
        "Standard_E4_v3",
        "Standard_E8_v3",
        "Standard_E16_v3",
        "Standard_E20_v3",
        "Standard_E32_v3",
        "Standard_E48_v3"
      ],
      "metadata": {
        "description": "This is the workernode Azure Virtual Machine size,
and will affect the cost. If you don't know, just leave the default value."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-08-01",
```

```
      "name": "[format('storage{0}', uniqueString(resourceGroup().id))]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "StorageV2"
    },
    {
      "type": "Microsoft.HDInsight/clusters",
      "apiVersion": "2021-06-01",
      "name": "[parameters('clusterName')]",
      "location": "[parameters('location')]",
      "properties": {
        "clusterVersion": "4.0",
        "osType": "Linux",
        "tier": "Standard",
        "clusterDefinition": {
          "kind": "spark",
          "configurations": {
            "gateway": {
              "restAuthCredential.isEnabled": true,
              "restAuthCredential.username": "
[parameters('clusterLoginUserName')]",
              "restAuthCredential.password": "
[parameters('clusterLoginPassword')]"
            }
          }
        },
        "storageProfile": {
          "storageaccounts": [
            {
              "name": "
[replace(replace(reference(resourceId('Microsoft.Storage/storageAccounts',
format('storage{0}',
uniqueString(resourceGroup().id)))).primaryEndpoints.blob, 'https://', ''),
'/', '')]",
              "isDefault": true,
              "container": "[parameters('clusterName')]",
              "key": "
[listKeys(resourceId('Microsoft.Storage/storageAccounts',
format('storage{0}', uniqueString(resourceGroup().id))), '2021-08-
01').keys[0].value]"
            }
          ]
        },
        "computeProfile": {
          "roles": [
            {
              "name": "headnode",
              "targetInstanceCount": 2,
              "hardwareProfile": {
                "vmSize": "[parameters('headNodeVirtualMachineSize')]"
              },
              "osProfile": {
                "linuxOperatingSystemProfile": {
```

```
                "username": "[parameters('sshUserName')]",
                "password": "[parameters('sshPassword')]"
              }
            }
          },
          {
            "name": "workernode",
            "targetInstanceCount": 2,
            "hardwareProfile": {
              "vmSize": "[parameters('workerNodeVirtualMachineSize')]"
            },
            "osProfile": {
              "linuxOperatingSystemProfile": {
                "username": "[parameters('sshUserName')]",
                "password": "[parameters('sshPassword')]"
              }
            }
          }
        ]
      }
    },
    "dependsOn": [
      "[resourceId('Microsoft.Storage/storageAccounts',
format('storage{0}', uniqueString(resourceGroup().id)))]"
    ]
  }
],
"outputs": {
  "storage": {
    "type": "object",
    "value": "[reference(resourceId('Microsoft.Storage/storageAccounts',
format('storage{0}', uniqueString(resourceGroup().id))))]"
  },
  "cluster": {
    "type": "object",
    "value": "[reference(resourceId('Microsoft.HDInsight/clusters',
parameters('clusterName')))]"
  }
}
}
```

Two Azure resources are defined in the template:

- Microsoft.Storage/storageAccounts: create an Azure Storage Account.
- Microsoft.HDInsight/cluster: create an HDInsight cluster.

Deploy the template

1. Select the **Deploy to Azure** button below to sign in to Azure and open the ARM template.

Deploy to Azure

2. Enter or select the following values:

Property	Description
Subscription	From the drop-down list, select the Azure subscription that's used for the cluster.
Resource group	From the drop-down list, select your existing resource group, or select **Create new**.
Location	The value will autopopulate with the location used for the resource group.
Cluster Name	Enter a globally unique name. For this template, use only lowercase letters, and numbers.
Cluster Login User Name	Provide the username, default is `admin`.
Cluster Login Password	Provide a password. The password must be at least 10 characters in length and must contain at least one digit, one uppercase, and one lower case letter, one non-alphanumeric character (except characters `' ` "`).
Ssh User Name	Provide the username, default is `sshuser`.
Ssh Password	Provide the password.

3. Review the **TERMS AND CONDITIONS**. Then select **I agree to the terms and conditions stated above**, then **Purchase**. You'll receive a notification that your deployment is in progress. It takes about 20 minutes to create a cluster.

If you run into an issue with creating HDInsight clusters, it could be that you don't have the right permissions to do so. For more information, see Access control requirements.

Review deployed resources

Once the cluster is created, you'll receive a **Deployment succeeded** notification with a **Go to resource** link. Your Resource group page will list your new HDInsight cluster and the default storage associated with the cluster. Each cluster has an Azure Storage, an Azure Data Lake Storage Gen1, or an Azure Data Lake Storage Gen2 dependency. It's referred as the default storage account. HDInsight cluster and its default storage account must be colocated in the same Azure region. Deleting clusters doesn't delete the storage account dependency. It's referred as the default storage account. The HDInsight cluster and its default storage account must be colocated in the same Azure region. Deleting clusters doesn't delete the storage account.

Create a Jupyter Notebook file

Jupyter Notebook is an interactive notebook environment that supports various programming languages. You can use a Jupyter Notebook file to interact with your data, combine code with markdown text, and perform simple visualizations.

1. Open the Azure portal.

2. Select **HDInsight clusters**, and then select the cluster you created.

3. From the portal, in **Cluster dashboards** section, select **Jupyter Notebook**. If prompted, enter the cluster login credentials for the cluster.

4. Select **New** > **PySpark** to create a notebook.

A new notebook is created and opened with the name Untitled(Untitled.pynb).

Run Apache Spark SQL statements

SQL (Structured Query Language) is the most common and widely used language for querying and transforming data. Spark SQL functions as an extension to Apache Spark for processing structured data, using the familiar SQL syntax.

1. Verify the kernel is ready. The kernel is ready when you see a hollow circle next to the kernel name in the notebook. Solid circle denotes that the kernel is busy.

alt-text="Kernel status" border="true":::

When you start the notebook for the first time, the kernel performs some tasks in the background. Wait for the kernel to be ready.

2. Paste the following code in an empty cell, and then press **SHIFT + ENTER** to run the code. The command lists the Hive tables on the cluster:

```
SQL

%%sql
SHOW TABLES
```

When you use a Jupyter Notebook file with your HDInsight cluster, you get a preset `spark` session that you can use to run Hive queries using Spark SQL. `%%sql` tells Jupyter Notebook to use the preset `spark` session to run the Hive query. The query retrieves the top 10 rows from a Hive table (**hivesampletable**) that comes with all HDInsight clusters by default. The first time you submit the query, Jupyter will create a Spark application for the notebook. It takes about 30 seconds to complete. Once the Spark application is ready, the query is executed in about a second and produces the results. The output looks like:

y in HDInsight" border="true":::

Every time you run a query in Jupyter, your web browser window title shows a **(Busy)** status along with the notebook title. You also see a solid circle next to the

PySpark text in the top-right corner.

3. Run another query to see the data in `hivesampletable`.

```
SQL

%%sql
SELECT * FROM hivesampletable LIMIT 10
```

The screen shall refresh to show the query output.

	clientid	querytime	market	deviceplatform	devicemake	devicemodel	state	country	querydwelltime	sessionid	sessionpagevieworder
0	71448	2018-04-05 05:51:45	en-US	Android	Samsung	SCH-i500	California	United States	31.423273	1	41
1	71448	2018-04-05 05:51:33	en-US	Android	Samsung	SCH-i500	California	United States	11.878175	1	40
2	71448	2018-04-05 05:51:03	en-US	Android	Samsung	SCH-i500	California	United States	30.195784	1	39
3	71448	2018-04-05 05:51:03	en-US	Android	Samsung	SCH-i500	California	United States	0.129492	1	38
4	71448	2018-04-05 05:50:44	en-US	Android	Samsung	SCH-i500	California	United States	19.026435	1	37
5	71448	2018-04-05 05:50:27	en-US	Android	Samsung	SCH-i500	California	United States	16.411516	1	36
6	71448	2018-04-05 05:50:13	en-US	Android	Samsung	SCH-i500	California	United States	13.761518	1	35
7	71448	2018-04-05 05:50:04	en-US	Android	Samsung	SCH-i500	California	United States	9.091954	1	34
8	71448	2018-04-05 05:49:57	en-US	Android	Samsung	SCH-i500	California	United States	7.709461	1	33
9	71448	2018-04-05 05:54:21	en-US	Android	Samsung	SCH-i500	California	United States	0.899126	1	48

Insight" border="true":::

4. From the **File** menu on the notebook, select **Close and Halt**. Shutting down the notebook releases the cluster resources, including Spark application.

Clean up resources

After you complete the quickstart, you may want to delete the cluster. With HDInsight, your data is stored in Azure Storage, so you can safely delete a cluster when it isn't in use. You're also charged for an HDInsight cluster, even when it isn't in use. Since the charges for the cluster are many times more than the charges for storage, it makes economic sense to delete clusters when they aren't in use.

From the Azure portal, navigate to your cluster, and select **Delete**.

sight cluster" border="true":::

You can also select the resource group name to open the resource group page, and then select **Delete resource group**. By deleting the resource group, you delete both the HDInsight cluster, and the default storage account.

Next steps

In this quickstart, you learned how to create an Apache Spark cluster in HDInsight and run a basic Spark SQL query. Advance to the next tutorial to learn how to use an HDInsight cluster to run interactive queries on sample data.

Run interactive queries on Apache Spark

Quickstart: Create an Azure Stream Analytics job by using an ARM template

Article • 03/10/2023

In this quickstart, you use an Azure Resource Manager template (ARM template) to create an Azure Stream Analytics job. Once the job is created, you validate the deployment.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

To complete this article, you need to:

- Have an Azure subscription - create one for free.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.8.9.13224",
      "templateHash": "10972314977021709659"
    }
  },
  "parameters": {
```

```
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for the resources."
      }
    },
    "streamAnalyticsJobName": {
      "type": "string",
      "maxLength": 63,
      "minLength": 3,
      "metadata": {
        "description": "Stream Analytics Job Name, can contain alphanumeric
characters and hypen and must be 3-63 characters long"
      }
    },
    "numberOfStreamingUnits": {
      "type": "int",
      "allowedValues": [
        1,
        3,
        6,
        12,
        18,
        24,
        30,
        36,
        42,
        48
      ],
      "maxValue": 48,
      "minValue": 1,
      "metadata": {
        "description": "Number of Streaming Units"
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.StreamAnalytics/streamingjobs",
      "apiVersion": "2021-10-01-preview",
      "name": "[parameters('streamAnalyticsJobName')]",
      "location": "[parameters('location')]",
      "properties": {
        "sku": {
          "name": "Standard"
        },
        "outputErrorPolicy": "Stop",
        "eventsOutOfOrderPolicy": "Adjust",
        "eventsOutOfOrderMaxDelayInSeconds": 0,
        "eventsLateArrivalMaxDelayInSeconds": 5,
        "dataLocale": "en-US",
        "transformation": {
          "name": "Transformation",
          "properties": {
```

```
            "streamingUnits": "[parameters('numberOfStreamingUnits')]",
            "query": "SELECT\r\n    *\r\nINTO\r\n
[YourOutputAlias]\r\nFROM\r\n    [YourInputAlias]"
          }
        }
      }
    }
  ]
}
```

The Azure resource defined in the template is Microsoft.StreamAnalytics/StreamingJobs: create an Azure Stream Analytics job.

Deploy the template

In this section, you create an Azure Stream Analytics job using the ARM template.

1. Select the following image to sign in to Azure and open a template. The template creates an Azure Stream Analytics job.

   Deploy to Azure

2. Provide the required values to create your Azure Stream Analytics job.

Provide the following values:

Property	Description
Subscription	From the drop-down, select your Azure subscription.
Resource group	Specify whether you want to create a new resource group or use an existing one. A resource group is a container that holds related resources for an Azure solution. For more information, see Azure Resource Group overview.
Region	Select **East US**. For other available regions, see Azure services available by region.
Stream Analytics Job Name	Provide a name for your Stream Analytics job.
Number of Streaming Units	Choose the number of streaming units you need. For more information, see Understand and adjust Streaming Units.

3. Select **Review + Create**, then **Create**.

Review deployed resources

You can either use the Azure portal to check the Azure Stream Analytics job or use the following Azure CLI or Azure PowerShell script to list the resource.

Azure CLI

Azure CLI

```
echo "Enter your Azure Stream Analytics job name:" &&
read streamAnalyticsJobName &&
echo "Enter the resource group where the Azure Stream Analytics job exists:"
&&
read resourcegroupName &&
az stream-analytics job show -g $resourcegroupName -n
$streamAnalyticsJobName
```

Azure PowerShell

Azure PowerShell

```
$resourceGroupName = Read-Host -Prompt "Enter the resource group name where
your Azure Stream Analytics job exists"
(Get-AzResource -ResourceType "Microsoft.StreamAnalytics/StreamingJobs" -
ResourceGroupName $resourceGroupName).Name
 Write-Host "Press [ENTER] to continue..."
```

Clean up resources

If you plan to continue on to subsequent tutorials, you may wish to leave these resources in place. When no longer needed, delete the resource group, which deletes the Azure Stream Analytics job. To delete the resource group by using Azure CLI or Azure PowerShell:

Azure CLI

Azure CLI

```
echo "Enter the Resource Group name:" &&
read resourceGroupName &&
az group delete --name $resourceGroupName &&
echo "Press [ENTER] to continue ..."
```

Azure PowerShell

Azure PowerShell

```
$resourceGroupName = Read-Host -Prompt "Enter the Resource Group name"
Remove-AzResourceGroup -Name $resourceGroupName
Write-Host "Press [ENTER] to continue..."
```

Next steps

In this quickstart, you created an Azure Stream Analytics job by using an ARM template and validated the deployment. Advance to the next article to learn how to export an ARM template for an existing job using VS Code.

Export an Azure Stream Analytics job ARM template

Quickstart: Create an Azure Synapse Analytics dedicated SQL pool (formerly SQL DW) by using an ARM template

Article • 03/10/2023

This Azure Resource Manager template (ARM template) will create an dedicated SQL pool (formerly SQL DW) with Transparent Data Encryption enabled. Dedicated SQL pool (formerly SQL DW) refers to the enterprise data warehousing features that are generally available in Azure Synapse.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.6.18.56646",
      "templateHash": "8855658719083429210"
    }
```

```
  },
  "parameters": {
    "sqlServerName": {
      "type": "string",
      "defaultValue": "[format('sql{0}',
uniqueString(resourceGroup().id))]",
      "metadata": {
        "description": "The SQL Logical Server name."
      }
    },
    "sqlAdministratorLogin": {
      "type": "string",
      "metadata": {
        "description": "The administrator username of the SQL Server."
      }
    },
    "sqlAdministratorPassword": {
      "type": "secureString",
      "metadata": {
        "description": "The administrator password of the SQL Server."
      }
    },
    "databasesName": {
      "type": "string",
      "metadata": {
        "description": "The name of the Database."
      }
    },
    "transparentDataEncryption": {
      "type": "string",
      "defaultValue": "Enabled",
      "allowedValues": [
        "Enabled",
        "Disabled"
      ],
      "metadata": {
        "description": "Enable/Disable Transparent Data Encryption"
      }
    },
    "capacity": {
      "type": "int",
      "maxValue": 54000,
      "minValue": 900,
      "metadata": {
        "description": "DW Performance Level expressed in DTU (i.e. 900 DTU
= DW100c)"
      }
    },
    "databaseCollation": {
      "type": "string",
      "defaultValue": "SQL_Latin1_General_CP1_CI_AS",
      "metadata": {
        "description": "The SQL Database collation."
      }
    },
```

```
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Resource location"
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Sql/servers",
      "apiVersion": "2021-11-01-preview",
      "name": "[parameters('sqlServerName')]",
      "location": "[parameters('location')]",
      "properties": {
        "administratorLogin": "[parameters('sqlAdministratorLogin')]",
        "administratorLoginPassword": "
[parameters('sqlAdministratorPassword')]",
        "version": "12.0",
        "publicNetworkAccess": "Enabled",
        "restrictOutboundNetworkAccess": "Disabled"
      }
    },
    {
      "type": "Microsoft.Sql/servers/databases",
      "apiVersion": "2021-11-01-preview",
      "name": "[format('{0}/{1}', parameters('sqlServerName'),
parameters('databasesName'))]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "DataWarehouse",
        "tier": "DataWarehouse",
        "capacity": "[parameters('capacity')]"
      },
      "properties": {
        "collation": "[parameters('databaseCollation')]",
        "catalogCollation": "[parameters('databaseCollation')]",
        "readScale": "Disabled",
        "requestedBackupStorageRedundancy": "Geo",
        "isLedgerOn": false
      },
      "dependsOn": [
        "[resourceId('Microsoft.Sql/servers', parameters('sqlServerName'))]"
      ]
    },
    {
      "type": "Microsoft.Sql/servers/databases/transparentDataEncryption",
      "apiVersion": "2021-11-01-preview",
      "name": "[format('{0}/{1}/{2}', parameters('sqlServerName'),
parameters('databasesName'), 'current')]",
      "properties": {
        "state": "[parameters('transparentDataEncryption')]"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Sql/servers/databases',
```

```
parameters('sqlServerName'), parameters('databasesName'))]"
      ]
    }
  ]
}
```

The template defines one resource:

- Microsoft.Sql/servers

Deploy the template

1. Select the following image to sign in to Azure and open the template. This template creates a dedicated SQL pool (formerly SQL DW).

   Deploy to Azure

2. Enter or update the following values:

   - **Subscription**: Select an Azure subscription.
   - **Resource group**: Select **Create new** and enter a unique name for the resource group and select **OK**. A new resource group will facilitate resource clean up.
   - **Region**: Select a region. For example, **Central US**.
   - **SQL Server name**: Accept the default name or enter a name for the SQL Server name.
   - **SQL Administrator login**: Enter the administrator username for the SQL Server.
   - **SQL Administrator password**: Enter the administrator password for the SQL Server.
   - **Data Warehouse Name**: Enter a dedicated SQL pool name.
   - **Transparent Data Encryption**: Accept the default, enabled.
   - **Service Level Objective**: Accept the default, DW400c.
   - **Location**: Accept the default location of the resource group.
   - **Review and Create**: Select.
   - **Create**: Select.

Review deployed resources

You can either use the Azure portal to check the deployed resources, or use Azure CLI or Azure PowerShell script to list the deployed resources.

CLI

Azure CLI

```
echo "Enter the resource group where your dedicated SQL pool (formerly
SQL DW) exists:" &&
read resourcegroupName &&
az resource list --resource-group $resourcegroupName
```

Clean up resources

When no longer needed, delete the resource group by using Azure CLI or Azure PowerShell:

CLI

Azure CLI

```
echo "Enter the Resource Group name:" &&
read resourceGroupName &&
az group delete --name $resourceGroupName &&
echo "Press [ENTER] to continue ..."
```

Next steps

In this quickstart, you created a dedicated SQL pool (formerly SQL DW) using an ARM template and validated the deployment. To learn more about Azure Synapse Analytics and Azure Resource Manager, see the articles below.

- Read an Overview of Azure Synapse Analytics
- Learn more about Azure Resource Manager
- Create and deploy your first ARM template

Quickstart: Create a Batch account by using ARM template

Article • 10/24/2021

Get started with Azure Batch by using an Azure Resource Manager template (ARM template) to create a Batch account, including storage. You need a Batch account to create compute resources (pools of compute nodes) and Batch jobs. You can link an Azure Storage account with your Batch account, which is useful to deploy applications and store input and output data for most real-world workloads.

After completing this quickstart, you'll understand the key concepts of the Batch service and be ready to try Batch with more realistic workloads at larger scale.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

You must have an active Azure subscription.

- If you don't have an Azure subscription, create an Azure free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
```

```
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "3759031252992325041"
    }
  },
  "parameters": {
    "batchAccountName": {
      "type": "string",
      "defaultValue": "[format('{0}batch',
toLower(uniqueString(resourceGroup().id)))]",
      "metadata": {
        "description": "Batch Account Name"
      }
    },
    "storageAccountsku": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_ZRS",
        "Premium_LRS"
      ],
      "metadata": {
        "description": "Storage Account type"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    }
  },
  "variables": {
    "storageAccountName": "[format('{0}storage',
uniqueString(resourceGroup().id))]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-08-01",
      "name": "[variables('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('storageAccountsku')]"
      },
      "kind": "StorageV2",
      "tags": {
        "ObjectName": "[variables('storageAccountName')]"
      },
      "properties": {}
```

```
    },
    {
      "type": "Microsoft.Batch/batchAccounts",
      "apiVersion": "2021-06-01",
      "name": "[parameters('batchAccountName')]",
      "location": "[parameters('location')]",
      "tags": {
        "ObjectName": "[parameters('batchAccountName')]"
      },
      "properties": {
        "autoStorage": {
          "storageAccountId": "
[resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))]"
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))]"
      ]
    }
  ],
  "outputs": {
    "storageAccountName": {
      "type": "string",
      "value": "[variables('storageAccountName')]"
    },
    "batchAccountName": {
      "type": "string",
      "value": "[parameters('batchAccountName')]"
    }
  }
}
```

Two Azure resources are defined in the template:

- Microsoft.Storage/storageAccounts: Creates a storage account.
- Microsoft.Batch/batchAccounts: Creates a Batch account.

Deploy the template

1. Select the following image to sign in to Azure and open a template. The template creates an Azure Batch account and a storage account.

   Deploy to Azure

2. Select or enter the following values.

- **Subscription**: select an Azure subscription.
- **Resource group**: select **Create new**, enter a unique name for the resource group, and then click **OK**.
- **Location**: select a location. For example, **Central US**.
- **Batch Account Name**: Leave the default value.
- **Storage Accountsku**: select a storage account type. For example, **Standard_LRS**.
- **Location**: Leave the default so that the resources will be in the same location as your resource group.

3. Select **Review + create**, then select **Create**.

After a few minutes, you should see a notification that the Batch account was successfully created.

In this example, the Azure portal is used to deploy the template. In addition to the Azure portal, you can also use the Azure PowerShell, Azure CLI, and REST API. To learn other deployment methods, see Deploy templates.

Validate the deployment

You can validate the deployment in the Azure portal by navigating to the resource group you created. In the **Overview** screen, confirm that the Batch account and the storage account are present.

Clean up resources

If you plan to continue on with more of our tutorials, you may wish to leave these resources in place. Or, if you no longer need them, you can delete the resource group, which will also delete the Batch account and the storage account that you created.

Next steps

In this quickstart, you created a Batch account and a storage account. To learn more about Azure Batch, continue to the Azure Batch tutorials.

Azure Batch tutorials

Quickstart: Create and deploy Azure Functions resources from an ARM template

Article • 04/05/2023

In this article, you use Azure Functions with an Azure Resource Manager template (ARM template) to create a function app and related resources in Azure. The function app provides an execution context for your function code executions.

Completing this quickstart incurs a small cost of a few USD cents or less in your Azure account.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

After you create the function app, you can deploy Azure Functions project code to that app.

Prerequisites

Azure account

Before you begin, you must have an Azure account with an active subscription. Create an account for free.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.15.31.15270",
      "templateHash": "11861629922040246994"
    }
  },
  "parameters": {
    "appName": {
      "type": "string",
      "defaultValue": "[format('fnapp{0}',
uniqueString(resourceGroup().id))]",
      "metadata": {
        "description": "The name of the function app that you wish to
create."
      }
    },
    "storageAccountType": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_RAGRS"
      ],
      "metadata": {
        "description": "Storage Account type"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "appInsightsLocation": {
      "type": "string",
      "metadata": {
        "description": "Location for Application Insights"
      }
    },
    "runtime": {
      "type": "string",
      "defaultValue": "node",
      "allowedValues": [
        "node",
        "dotnet",
        "java"
      ],
```

```
      "metadata": {
        "description": "The language worker runtime to load in the function
app."
      }
    }
  },
  "variables": {
    "functionAppName": "[parameters('appName')]",
    "hostingPlanName": "[parameters('appName')]",
    "applicationInsightsName": "[parameters('appName')]",
    "storageAccountName": "[format('{0}azfunctions',
uniqueString(resourceGroup().id))]",
    "functionWorkerRuntime": "[parameters('runtime')]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2022-05-01",
      "name": "[variables('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('storageAccountType')]"
      },
      "kind": "Storage",
      "properties": {
        "supportsHttpsTrafficOnly": true,
        "defaultToOAuthAuthentication": true
      }
    },
    {
      "type": "Microsoft.Web/serverfarms",
      "apiVersion": "2021-03-01",
      "name": "[variables('hostingPlanName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Y1",
        "tier": "Dynamic"
      },
      "properties": {}
    },
    {
      "type": "Microsoft.Web/sites",
      "apiVersion": "2021-03-01",
      "name": "[variables('functionAppName')]",
      "location": "[parameters('location')]",
      "kind": "functionapp",
      "identity": {
        "type": "SystemAssigned"
      },
      "properties": {
        "serverFarmId": "[resourceId('Microsoft.Web/serverfarms',
variables('hostingPlanName'))]",
        "siteConfig": {
          "appSettings": [
            {
```

```
              "name": "AzureWebJobsStorage",
              "value": "[format('DefaultEndpointsProtocol=https;AccountName=
{0};EndpointSuffix={1};AccountKey={2}', variables('storageAccountName'),
environment().suffixes.storage,
listKeys(resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName')), '2022-05-01').keys[0].value)]"
            },
            {
              "name": "WEBSITE_CONTENTAZUREFILECONNECTIONSTRING",
              "value": "[format('DefaultEndpointsProtocol=https;AccountName=
{0};EndpointSuffix={1};AccountKey={2}', variables('storageAccountName'),
environment().suffixes.storage,
listKeys(resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName')), '2022-05-01').keys[0].value)]"
            },
            {
              "name": "WEBSITE_CONTENTSHARE",
              "value": "[toLower(variables('functionAppName'))]"
            },
            {
              "name": "FUNCTIONS_EXTENSION_VERSION",
              "value": "~4"
            },
            {
              "name": "WEBSITE_NODE_DEFAULT_VERSION",
              "value": "~14"
            },
            {
              "name": "APPINSIGHTS_INSTRUMENTATIONKEY",
              "value": "
[reference(resourceId('Microsoft.Insights/components',
variables('applicationInsightsName')), '2020-02-02').InstrumentationKey]"
            },
            {
              "name": "FUNCTIONS_WORKER_RUNTIME",
              "value": "[variables('functionWorkerRuntime')]"
            }
          ],
          "ftpsState": "FtpsOnly",
          "minTlsVersion": "1.2"
        },
        "httpsOnly": true
      },
      "dependsOn": [
        "[resourceId('Microsoft.Insights/components',
variables('applicationInsightsName'))]",
        "[resourceId('Microsoft.Web/serverfarms',
variables('hostingPlanName'))]",
        "[resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))]"
      ]
    },
    {
      "type": "Microsoft.Insights/components",
      "apiVersion": "2020-02-02",
```

```
      "name": "[variables('applicationInsightsName')]",
      "location": "[parameters('appInsightsLocation')]",
      "kind": "web",
      "properties": {
        "Application_Type": "web",
        "Request_Source": "rest"
      }
    }
  ]
}
```

The following four Azure resources are created by this template:

- **Microsoft.Storage/storageAccounts**: create an Azure Storage account, which is required by Functions.
- **Microsoft.Web/serverfarms**: create a serverless Consumption hosting plan for the function app.
- **Microsoft.Web/sites**: create a function app.
- **microsoft.insights/components**: create an Application Insights instance for monitoring.

> **Important**
>
> The storage account is used to store important app data, sometimes including the application code itself. You should limit access from other apps and users to the storage account.

Deploy the template

The following scripts are designed for and tested in Azure Cloud Shell. Choose **Try It** to open a Cloud Shell instance right in your browser.

Azure CLI

Azure CLI

```
read -p "Enter a resource group name that is used for generating
resource names:" resourceGroupName &&
read -p "Enter the location (like 'eastus' or 'northeurope'):" location
&&
templateUri="https://raw.githubusercontent.com/Azure/azure-quickstart-
templates/master/quickstarts/microsoft.web/function-app-create-
dynamic/azuredeploy.json" &&
az group create --name $resourceGroupName --location "$location" &&
az deployment group create --resource-group $resourceGroupName --
```

```
template-uri  $templateUri &&
echo "Press [ENTER] to continue ..." &&
read
```

Visit function app welcome page

1. Use the output from the previous validation step to retrieve the unique name created for your function app.

2. Open a browser and enter the following URL: **<https://<appName.azurewebsites.net>**. Make sure to replace **<\appName>** with the unique name created for your function app.

   When you visit the URL, you should see a page like this:

Clean up resources

If you continue to the next step and add an Azure Storage queue output binding, keep all your resources in place as you'll build on what you've already done.

Otherwise, use the following command to delete the resource group and all its contained resources to avoid incurring further costs.

Azure CLI

Azure CLI

```
az group delete --name <RESOURCE_GROUP_NAME>
```

Replace `<RESOURCE_GROUP_NAME>` with the name of your resource group.

Next steps

Now that you've created your function app resources in Azure, you can deploy your code to the existing app by using one of the following tools:

- Visual Studio Code
- Visual Studio
- Azure Functions Core Tools

Quickstart: Create an Ubuntu Linux virtual machine by using an ARM template

Article • 04/17/2023

Applies to: ✔ Linux VMs

This quickstart shows you how to use an Azure Resource Manager template (ARM template) to deploy an Ubuntu Linux virtual machine (VM) in Azure.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with ARM templates, select the **Deploy to Azure** button. The template opens in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

For more information on this template, see Deploy a simple Ubuntu Linux VM 18.04-LTS.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.14.46.61228",
      "templateHash": "9969522307620169 09"
    }
  },
```

```
  "parameters": {
    "vmName": {
      "type": "string",
      "defaultValue": "simpleLinuxVM",
      "metadata": {
        "description": "The name of you Virtual Machine."
      }
    },
    "adminUsername": {
      "type": "string",
      "metadata": {
        "description": "Username for the Virtual Machine."
      }
    },
    "authenticationType": {
      "type": "string",
      "defaultValue": "password",
      "allowedValues": [
        "sshPublicKey",
        "password"
      ],
      "metadata": {
        "description": "Type of authentication to use on the Virtual
Machine. SSH key is recommended."
      }
    },
    "adminPasswordOrKey": {
      "type": "securestring",
      "metadata": {
        "description": "SSH Key or password for the Virtual Machine. SSH key
is recommended."
      }
    },
    "dnsLabelPrefix": {
      "type": "string",
      "defaultValue": "[toLower(format('{0}-{1}', parameters('vmName'),
uniqueString(resourceGroup().id)))]",
      "metadata": {
        "description": "Unique DNS Name for the Public IP used to access the
Virtual Machine."
      }
    },
    "ubuntuOSVersion": {
      "type": "string",
      "defaultValue": "Ubuntu-2004",
      "allowedValues": [
        "Ubuntu-1804",
        "Ubuntu-2004",
        "Ubuntu-2204"
      ],
      "metadata": {
        "description": "The Ubuntu version for the VM. This will pick a
fully patched image of this given Ubuntu version."
      }
    },
```

```
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "vmSize": {
      "type": "string",
      "defaultValue": "Standard_D2s_v3",
      "metadata": {
        "description": "The size of the VM"
      }
    },
    "virtualNetworkName": {
      "type": "string",
      "defaultValue": "vNet",
      "metadata": {
        "description": "Name of the VNET"
      }
    },
    "subnetName": {
      "type": "string",
      "defaultValue": "Subnet",
      "metadata": {
        "description": "Name of the subnet in the virtual network"
      }
    },
    "networkSecurityGroupName": {
      "type": "string",
      "defaultValue": "SecGroupNet",
      "metadata": {
        "description": "Name of the Network Security Group"
      }
    },
    "securityType": {
      "type": "string",
      "defaultValue": "TrustedLaunch",
      "allowedValues": [
        "Standard",
        "TrustedLaunch"
      ],
      "metadata": {
        "description": "Security Type of the Virtual Machine."
      }
    }
  },
  "variables": {
    "imageReference": {
      "Ubuntu-1804": {
        "publisher": "Canonical",
        "offer": "UbuntuServer",
        "sku": "18_04-lts-gen2",
        "version": "latest"
      },
```

```
      "Ubuntu-2004": {
        "publisher": "Canonical",
        "offer": "0001-com-ubuntu-server-focal",
        "sku": "20_04-lts-gen2",
        "version": "latest"
      },
      "Ubuntu-2204": {
        "publisher": "Canonical",
        "offer": "0001-com-ubuntu-server-jammy",
        "sku": "22_04-lts-gen2",
        "version": "latest"
      }
    },
    "publicIPAddressName": "[format('{0}PublicIP', parameters('vmName'))]",
    "networkInterfaceName": "[format('{0}NetInt', parameters('vmName'))]",
    "osDiskType": "Standard_LRS",
    "subnetAddressPrefix": "10.1.0.0/24",
    "addressPrefix": "10.1.0.0/16",
    "linuxConfiguration": {
      "disablePasswordAuthentication": true,
      "ssh": {
        "publicKeys": [
          {
            "path": "[format('/home/{0}/.ssh/authorized_keys',
parameters('adminUsername'))]",
            "keyData": "[parameters('adminPasswordOrKey')]"
          }
        ]
      }
    },
    "securityProfileJson": {
      "uefiSettings": {
        "secureBootEnabled": true,
        "vTpmEnabled": true
      },
      "securityType": "[parameters('securityType')]"
    },
    "extensionName": "GuestAttestation",
    "extensionPublisher": "Microsoft.Azure.Security.LinuxAttestation",
    "extensionVersion": "1.0",
    "maaTenantName": "GuestAttestation",
    "maaEndpoint": "[substring('emptystring', 0, 0)]"
  },
  "resources": [
    {
      "type": "Microsoft.Network/networkInterfaces",
      "apiVersion": "2021-05-01",
      "name": "[variables('networkInterfaceName')]",
      "location": "[parameters('location')]",
      "properties": {
        "ipConfigurations": [
          {
            "name": "ipconfig1",
            "properties": {
              "subnet": {
```

```
                "id": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('virtualNetworkName'), parameters('subnetName'))]"
              },
              "privateIPAllocationMethod": "Dynamic",
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses',
variables('publicIPAddressName'))]"
              }
            }
          }
        ],
        "networkSecurityGroup": {
          "id": "[resourceId('Microsoft.Network/networkSecurityGroups',
parameters('networkSecurityGroupName'))]"
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkSecurityGroups',
parameters('networkSecurityGroupName'))]",
        "[resourceId('Microsoft.Network/publicIPAddresses',
variables('publicIPAddressName'))]",
        "[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('virtualNetworkName'), parameters('subnetName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/networkSecurityGroups",
      "apiVersion": "2021-05-01",
      "name": "[parameters('networkSecurityGroupName')]",
      "location": "[parameters('location')]",
      "properties": {
        "securityRules": [
          {
            "name": "SSH",
            "properties": {
              "priority": 1000,
              "protocol": "Tcp",
              "access": "Allow",
              "direction": "Inbound",
              "sourceAddressPrefix": "*",
              "sourcePortRange": "*",
              "destinationAddressPrefix": "*",
              "destinationPortRange": "22"
            }
          }
        ]
      }
    },
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "2021-05-01",
      "name": "[parameters('virtualNetworkName')]",
      "location": "[parameters('location')]",
      "properties": {
```

```
        "addressSpace": {
          "addressPrefixes": [
            "[variables('addressPrefix')]"
          ]
        }
      }
    },
    {
      "type": "Microsoft.Network/virtualNetworks/subnets",
      "apiVersion": "2021-05-01",
      "name": "[format('{0}/{1}', parameters('virtualNetworkName'),
parameters('subnetName'))]",
      "properties": {
        "addressPrefix": "[variables('subnetAddressPrefix')]",
        "privateEndpointNetworkPolicies": "Enabled",
        "privateLinkServiceNetworkPolicies": "Enabled"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/virtualNetworks',
parameters('virtualNetworkName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2021-05-01",
      "name": "[variables('publicIPAddressName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Basic"
      },
      "properties": {
        "publicIPAllocationMethod": "Dynamic",
        "publicIPAddressVersion": "IPv4",
        "dnsSettings": {
          "domainNameLabel": "[parameters('dnsLabelPrefix')]"
        },
        "idleTimeoutInMinutes": 4
      }
    },
    {
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2021-11-01",
      "name": "[parameters('vmName')]",
      "location": "[parameters('location')]",
      "properties": {
        "hardwareProfile": {
          "vmSize": "[parameters('vmSize')]"
        },
        "storageProfile": {
          "osDisk": {
            "createOption": "FromImage",
            "managedDisk": {
              "storageAccountType": "[variables('osDiskType')]"
            }
          },
```

```
          "imageReference": "[variables('imageReference')
[parameters('ubuntuOSVersion')]]"
        },
        "networkProfile": {
          "networkInterfaces": [
            {
              "id": "[resourceId('Microsoft.Network/networkInterfaces',
variables('networkInterfaceName'))]"
            }
          ]
        },
        "osProfile": {
          "computerName": "[parameters('vmName')]",
          "adminUsername": "[parameters('adminUsername')]",
          "adminPassword": "[parameters('adminPasswordOrKey')]",
          "linuxConfiguration": "
[if(equals(parameters('authenticationType'), 'password'), null(),
variables('linuxConfiguration'))]"
        },
        "securityProfile": "[if(equals(parameters('securityType'),
'TrustedLaunch'), variables('securityProfileJson'), json('null'))]"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkInterfaces',
variables('networkInterfaceName'))]"
      ]
    },
    {
      "condition": "[and(equals(parameters('securityType'),
'TrustedLaunch'),
and(equals(variables('securityProfileJson').uefiSettings.secureBootEnabled,
true()), equals(variables('securityProfileJson').uefiSettings.vTpmEnabled,
true())))]",
      "type": "Microsoft.Compute/virtualMachines/extensions",
      "apiVersion": "2022-03-01",
      "name": "[format('{0}/{1}', parameters('vmName'),
variables('extensionName'))]",
      "location": "[parameters('location')]",
      "properties": {
        "publisher": "[variables('extensionPublisher')]",
        "type": "[variables('extensionName')]",
        "typeHandlerVersion": "[variables('extensionVersion')]",
        "autoUpgradeMinorVersion": true,
        "settings": {
          "AttestationConfig": {
            "MaaSettings": {
              "maaEndpoint": "[variables('maaEndpoint')]",
              "maaTenantName": "[variables('maaTenantName')]"
            }
          }
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Compute/virtualMachines',
parameters('vmName'))]"
```

```
      ]
    }
  ],
  "outputs": {
    "adminUsername": {
      "type": "string",
      "value": "[parameters('adminUsername')]"
    },
    "hostname": {
      "type": "string",
      "value": "[reference(resourceId('Microsoft.Network/publicIPAddresses',
variables('publicIPAddressName')), '2021-05-01').dnsSettings.fqdn]"
    },
    "sshCommand": {
      "type": "string",
      "value": "[format('ssh {0}@{1}', parameters('adminUsername'),
reference(resourceId('Microsoft.Network/publicIPAddresses',
variables('publicIPAddressName')), '2021-05-01').dnsSettings.fqdn)]"
    }
  }
}
```

Several resources are defined in the template:

- Microsoft.Network/virtualNetworks/subnets: create a subnet.
- Microsoft.Storage/storageAccounts: create a storage account.
- Microsoft.Network/networkInterfaces: create a NIC.
- Microsoft.Network/networkSecurityGroups: create a network security group.
- Microsoft.Network/virtualNetworks: create a virtual network.
- Microsoft.Network/publicIPAddresses: create a public IP address.
- Microsoft.Compute/virtualMachines: create a virtual machine.

Deploy the template

1. Select the following image to sign in to Azure and open a template. The template creates a key vault and a secret.

   Deploy to Azure

2. Select or enter the following values. Use the default values, when available.

   - **Subscription**: select an Azure subscription.
   - **Resource group**: select an existing resource group from the drop-down, or select **Create new**, enter a unique name for the resource group, and select **OK**.
   - **Region**: select a region. For example, **Central US**.

- **Admin username**: provide a username, such as *azureuser*.
- **Authentication type**: You can choose between an SSH key or a password.
- **Admin Password Or Key** depending on what you choose for authentication type:
  - If you choose **password**, the password must be at least 12 characters long and meet the defined complexity requirements.
  - If you choose **sshPublicKey**, paste in the contents of your public key.
- **DNS label prefix**: enter a unique identifier to use as part of the DNS label.
- **Ubuntu OS version**: select which version of Ubuntu you want to run on the VM.
- **Location**: the default is the same location as the resource group, if it already exists.
- **VM size**: select the size to use for the VM.
- **Virtual Network Name**: name to be used for the vNet.
- **Subnet Name**: name for the subnet the VM should use.
- **Network Security Group Name**: name for the NSG.

3. Select **Review + create**. After validation completes, select **Create** to create and deploy the VM.

The Azure portal is used to deploy the template. In addition to the Azure portal, you can also use the Azure CLI, Azure PowerShell, and REST API. To learn other deployment methods, see Deploy templates.

Review deployed resources

You can use the Azure portal to check on the VM and other resource that were created. After the deployment is finished, select **Resource groups** to see the VM and other resources.

Clean up resources

When no longer needed, delete the resource group, which deletes the VM and all of the resources in the resource group.

1. Select the **Resource group**.
2. On the page for the resource group, select **Delete resource group**.
3. When prompted, type the name of the resource group and then select **Delete**.

Next steps

In this quickstart, you deployed a virtual machine by using an ARM template. To learn more about Azure virtual machines, continue to the tutorial for Linux VMs.

Create and Manage Linux VMs with the Azure CLI

Quickstart: Create a Linux Virtual Machine Scale Set with an ARM template

Article • 03/31/2023

Applies to: ✔ Linux VMs ✔ Uniform scale sets

> **Note**
>
> The following article is for Uniform Virtual Machine Scale Sets. We recommend using Flexible Virtual Machine Scale Sets for new workloads. Learn more about this new orchestration mode in our **Flexible Virtual Machine Scale Sets overview**.

A Virtual Machine Scale Set allows you to deploy and manage a set of auto-scaling virtual machines. You can scale the number of VMs in the scale set manually, or define rules to autoscale based on resource usage like CPU, memory demand, or network traffic. An Azure load balancer then distributes traffic to the VM instances in the scale set. In this quickstart, you create a Virtual Machine Scale Set and deploy a sample application with an Azure Resource Manager template (ARM template).

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

ARM templates let you deploy groups of related resources. In a single template, you can create the Virtual Machine Scale Set, install applications, and configure autoscale rules. With the use of variables and parameters, this template can be reused to update existing, or create additional, scale sets. You can deploy templates through the Azure portal, Azure CLI, or Azure PowerShell, or from continuous integration / continuous delivery (CI/CD) pipelines.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

These resources are defined in the template:

- **Microsoft.Network/virtualNetworks**
- **Microsoft.Network/publicIPAddresses**
- **Microsoft.Network/loadBalancers**
- **Microsoft.Compute/virtualMachineScaleSets**
- **Microsoft.Insights/autoscaleSettings**

Define a scale set

To create a scale with a template, you define the appropriate resources. The core parts of the Virtual Machine Scale Set resource type are:

Property	Description of property	Example template value
type	Azure resource type to create	Microsoft.Compute/virtualMachineScaleSets
name	The scale set name	myScaleSet
location	The location to create the scale set	East US
sku.name	The VM size for each scale set instance	Standard_A1
sku.capacity	The number of VM instances to initially create	2
upgradePolicy.mode	VM instance upgrade mode when changes occur	Automatic

Property	Description of property	Example template value
imageReference	The platform or custom image to use for the VM instances	Canonical Ubuntu Server 16.04-LTS
osProfile.computerNamePrefix	The name prefix for each VM instance	myvmss
osProfile.adminUsername	The username for each VM instance	azureuser
osProfile.adminPassword	The password for each VM instance	P@ssw0rd!

To customize a scale set template, you can change the VM size or initial capacity. Another option is to use a different platform or a custom image.

Add a sample application

To test your scale set, install a basic web application. When you deploy a scale set, VM extensions can provide post-deployment configuration and automation tasks, such as installing an app. Scripts can be downloaded from Azure storage or GitHub, or provided to the Azure portal at extension run-time. To apply an extension to your scale set, you add the *extensionProfile* section to the preceding resource example. The extension profile typically defines the following properties:

- Extension type
- Extension publisher
- Extension version
- Location of configuration or install scripts
- Commands to execute on the VM instances

The template uses the Custom Script Extension to install Bottle, a Python web framework, and a simple HTTP server.

Two scripts are defined in **fileUris** - *installserver.sh*, and *workserver.py*. These files are downloaded from GitHub, then *commandToExecute* runs `bash installserver.sh` to install and configure the app.

Deploy the template

You can deploy the template by selecting the following **Deploy to Azure** button. This button opens the Azure portal, loads the complete template, and prompts for a few parameters such as a scale set name, instance count, and admin credentials.

Deploy to Azure

You can also deploy a Resource Manager template by using Azure CLI:

Azure CLI

```
# Create a resource group
az group create --name myResourceGroup --location EastUS

# Deploy template into resource group
az deployment group create \
    --resource-group myResourceGroup \
    --template-uri https://raw.githubusercontent.com/Azure/azure-quickstart-
templates/master/application-workloads/python/vmss-bottle-
autoscale/azuredeploy.json
```

Answer the prompts to provide a scale set name, instance count, and admin credentials for the VM instances. It takes a few minutes for the scale set and supporting resources to be created.

Validate the deployment

To see your scale set in action, access the sample web application in a web browser. Obtain the public IP address of the load balancer with az network public-ip list as follows:

Azure CLI

```
az network public-ip list \
    --resource-group myResourceGroup \
    --query [*].ipAddress -o tsv
```

Enter the public IP address of the load balancer in to a web browser in the format *http://publicIpAddress:9000/do_work*. The load balancer distributes traffic to one of your VM instances, as shown in the following example:

Clean up resources

When no longer needed, you can use az group delete to remove the resource group, scale set, and all related resources as follows. The `--no-wait` parameter returns control to the prompt without waiting for the operation to complete. The `--yes` parameter confirms that you wish to delete the resources without an additional prompt to do so.

Azure CLI

```
az group delete --name myResourceGroup --yes --no-wait
```

Next steps

In this quickstart, you created a Linux scale set with an ARM template and used the Custom Script Extension to install a basic Python web server on the VM instances. To learn more, continue to the tutorial for how to create and manage Azure Virtual Machine Scale Sets.

Create and manage Azure Virtual Machine Scale Sets

Quickstart: Create a Windows Virtual Machine Scale Set with an ARM template

Article • 03/10/2023

Applies to: ✔ Windows VMs ✔ Uniform scale sets

> **Note**
>
> The following article is for Uniform Virtual Machine Scale Sets. We recommend using Flexible Virtual Machine Scale Sets for new workloads. Learn more about this new orchestration mode in our **Flexible Virtual Machine Scale Sets overview**.

A Virtual Machine Scale Set allows you to deploy and manage a set of auto-scaling virtual machines. You can scale the number of VMs in the scale set manually, or define rules to autoscale based on resource usage like CPU, memory demand, or network traffic. An Azure load balancer then distributes traffic to the VM instances in the scale set. In this quickstart, you create a Virtual Machine Scale Set and deploy a sample application with an Azure Resource Manager template (ARM template).

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

ARM templates let you deploy groups of related resources. In a single template, you can create the Virtual Machine Scale Set, install applications, and configure autoscale rules. With the use of variables and parameters, this template can be reused to update existing, or create additional, scale sets. You can deploy templates through the Azure portal, Azure CLI, Azure PowerShell, or from continuous integration / continuous delivery (CI/CD) pipelines.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.7.4.23292",
      "templateHash": "15689099884164541808"
    }
  },
  "parameters": {
    "vmssName": {
      "type": "string",
      "maxLength": 61,
      "minLength": 3,
      "metadata": {
        "description": "String used as a base for naming resources. Must be 3-61 characters in length and globally unique across Azure. A hash is prepended to this string for some resources, and resource-specific information is appended."
      }
    },
    "vmSku": {
      "type": "string",
      "defaultValue": "Standard_A1_v2",
      "metadata": {
        "description": "Size of VMs in the VM Scale Set."
      }
    },
    "windowsOSVersion": {
      "type": "string",
      "defaultValue": "2019-Datacenter",
      "allowedValues": [
        "2008-R2-SP1",
        "2012-Datacenter",
        "2012-R2-Datacenter",
        "2016-Datacenter",
        "2019-Datacenter"
      ],
      "metadata": {
```

```
        "description": "The Windows version for the VM. This will pick a
fully patched image of this given Windows version. Allowed values: 2008-R2-
SP1, 2012-Datacenter, 2012-R2-Datacenter & 2016-Datacenter, 2019-
Datacenter."
      }
    },
    "instanceCount": {
      "type": "int",
      "defaultValue": 3,
      "maxValue": 100,
      "minValue": 1,
      "metadata": {
        "description": "Number of VM instances (100 or less)."
      }
    },
    "singlePlacementGroup": {
      "type": "bool",
      "defaultValue": true,
      "metadata": {
        "description": "When true this limits the scale set to a single
placement group, of max size 100 virtual machines. NOTE: If
singlePlacementGroup is true, it may be modified to false. However, if
singlePlacementGroup is false, it may not be modified to true."
      }
    },
    "adminUsername": {
      "type": "string",
      "defaultValue": "vmssadmin",
      "metadata": {
        "description": "Admin username on all VMs."
      }
    },
    "adminPassword": {
      "type": "secureString",
      "metadata": {
        "description": "Admin password on all VMs."
      }
    },
    "_artifactsLocation": {
      "type": "string",
      "defaultValue": "[deployment().properties.templateLink.uri]",
      "metadata": {
        "description": "The base URI where artifacts required by this
template are located. For example, if stored on a public GitHub repo, you'd
use the following URI: https://raw.githubusercontent.com/Azure/azure-
quickstart-templates/master/201-vmss-windows-webapp-dsc-autoscale/."
      }
    },
    "_artifactsLocationSasToken": {
      "type": "secureString",
      "defaultValue": "",
      "metadata": {
        "description": "The sasToken required to access _artifactsLocation.
If your artifacts are stored on a public repo or public storage account you
can leave this blank."
```

```
      }
    },
    "powershelldscZip": {
      "type": "string",
      "defaultValue": "DSC/InstallIIS.zip",
      "metadata": {
        "description": "Location of the PowerShell DSC zip file relative to
the URI specified in the _artifactsLocation, i.e. DSC/IISInstall.ps1.zip"
      }
    },
    "webDeployPackage": {
      "type": "string",
      "defaultValue": "WebDeploy/DefaultASPWebApp.v1.0.zip",
      "metadata": {
        "description": "Location of the  of the WebDeploy package zip file
relative to the URI specified in _artifactsLocation, i.e.
WebDeploy/DefaultASPWebApp.v1.0.zip"
      }
    },
    "powershelldscUpdateTagVersion": {
      "type": "string",
      "defaultValue": "1.0",
      "metadata": {
        "description": "Version number of the DSC deployment. Changing this
value on subsequent deployments will trigger the extension to run."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "platformFaultDomainCount": {
      "type": "int",
      "defaultValue": 1,
      "metadata": {
        "description": "Fault Domain count for each placement group."
      }
    }
  },
  "variables": {
    "vmScaleSetName": "[toLower(substring(format('vmssName{0}',
uniqueString(resourceGroup().id)), 0, 9))]",
    "longvmScaleSet": "[toLower(parameters('vmssName'))]",
    "addressPrefix": "10.0.0.0/16",
    "subnetPrefix": "10.0.0.0/24",
    "vNetName": "[format('{0}vnet', variables('vmScaleSetName'))]",
    "publicIPAddressName": "[format('{0}pip',
variables('vmScaleSetName'))]",
    "subnetName": "[format('{0}subnet', variables('vmScaleSetName'))]",
    "loadBalancerName": "[format('{0}lb', variables('vmScaleSetName'))]",
    "publicIPAddressID": "[resourceId('Microsoft.Network/publicIPAddresses',
variables('publicIPAddressName'))]",
```

```
    "lbProbeID": "[resourceId('Microsoft.Network/loadBalancers/probes',
variables('loadBalancerName'), 'tcpProbe')]",
    "natPoolName": "[format('{0}natpool', variables('vmScaleSetName'))]",
    "bePoolName": "[format('{0}bepool', variables('vmScaleSetName'))]",
    "lbPoolID": "
[resourceId('Microsoft.Network/loadBalancers/backendAddressPools',
variables('loadBalancerName'), variables('bePoolName'))]",
    "natStartPort": 50000,
    "natEndPort": 50119,
    "natBackendPort": 3389,
    "nicName": "[format('{0}nic', variables('vmScaleSetName'))]",
    "ipConfigName": "[format('{0}ipconfig', variables('vmScaleSetName'))]",
    "frontEndIPConfigID": "
[resourceId('Microsoft.Network/loadBalancers/frontendIPConfigurations',
variables('loadBalancerName'), 'loadBalancerFrontEnd')]",
    "osType": {
      "publisher": "MicrosoftWindowsServer",
      "offer": "WindowsServer",
      "sku": "[parameters('windowsOSVersion')]",
      "version": "latest"
    },
    "imageReference": "[variables('osType')]",
    "webDeployPackageFullPath": "[uri(parameters('_artifactsLocation'),
format('{0}{1}', parameters('webDeployPackage'),
parameters('_artifactsLocationSasToken')))]",
    "powershelldscZipFullPath": "[uri(parameters('_artifactsLocation'),
format('{0}{1}', parameters('powershelldscZip'),
parameters('_artifactsLocationSasToken')))]"
  },
  "resources": [
    {
      "type": "Microsoft.Network/loadBalancers",
      "apiVersion": "2021-05-01",
      "name": "[variables('loadBalancerName')]",
      "location": "[parameters('location')]",
      "properties": {
        "frontendIPConfigurations": [
          {
            "name": "LoadBalancerFrontEnd",
            "properties": {
              "publicIPAddress": {
                "id": "[variables('publicIPAddressID')]"
              }
            }
          }
        ],
        "backendAddressPools": [
          {
            "name": "[variables('bePoolName')]"
          }
        ],
        "inboundNatPools": [
          {
            "name": "[variables('natPoolName')]",
            "properties": {
```

```
              "frontendIPConfiguration": {
                "id": "[variables('frontEndIPConfigID')]"
              },
              "protocol": "Tcp",
              "frontendPortRangeStart": "[variables('natStartPort')]",
              "frontendPortRangeEnd": "[variables('natEndPort')]",
              "backendPort": "[variables('natBackendPort')]"
            }
          }
        ],
        "loadBalancingRules": [
          {
            "name": "LBRule",
            "properties": {
              "frontendIPConfiguration": {
                "id": "[variables('frontEndIPConfigID')]"
              },
              "backendAddressPool": {
                "id": "[variables('lbPoolID')]"
              },
              "protocol": "Tcp",
              "frontendPort": 80,
              "backendPort": 80,
              "enableFloatingIP": false,
              "idleTimeoutInMinutes": 5,
              "probe": {
                "id": "[variables('lbProbeID')]"
              }
            }
          }
        ],
        "probes": [
          {
            "name": "tcpProbe",
            "properties": {
              "protocol": "Tcp",
              "port": 80,
              "intervalInSeconds": 5,
              "numberOfProbes": 2
            }
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/publicIPAddresses',
variables('publicIPAddressName'))]"
      ]
    },
    {
      "type": "Microsoft.Compute/virtualMachineScaleSets",
      "apiVersion": "2021-11-01",
      "name": "[variables('vmScaleSetName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('vmSku')]",
```

```
        "tier": "Standard",
        "capacity": "[parameters('instanceCount')]"
      },
      "properties": {
        "overprovision": true,
        "upgradePolicy": {
          "mode": "Automatic"
        },
        "singlePlacementGroup": "[parameters('singlePlacementGroup')]",
        "platformFaultDomainCount": "
[parameters('platformFaultDomainCount')]",
        "virtualMachineProfile": {
          "storageProfile": {
            "osDisk": {
              "caching": "ReadWrite",
              "createOption": "FromImage"
            },
            "imageReference": "[variables('imageReference')]"
          },
          "osProfile": {
            "computerNamePrefix": "[variables('vmScaleSetName')]",
            "adminUsername": "[parameters('adminUsername')]",
            "adminPassword": "[parameters('adminPassword')]"
          },
          "networkProfile": {
            "networkInterfaceConfigurations": [
              {
                "name": "[variables('nicName')]",
                "properties": {
                  "primary": true,
                  "ipConfigurations": [
                    {
                      "name": "[variables('ipConfigName')]",
                      "properties": {
                        "subnet": {
                          "id": "
[reference(resourceId('Microsoft.Network/virtualNetworks',
variables('vNetName'))).subnets[0].id]"
                        },
                        "loadBalancerBackendAddressPools": [
                          {
                            "id": "[variables('lbPoolID')]"
                          }
                        ]
                      }
                    }
                  ]
                }
              }
            ]
          },
          "extensionProfile": {
            "extensions": [
              {
                "name": "Microsoft.Powershell.DSC",
```

```
                "properties": {
                  "publisher": "Microsoft.Powershell",
                  "type": "DSC",
                  "typeHandlerVersion": "2.9",
                  "autoUpgradeMinorVersion": true,
                  "forceUpdateTag": "
[parameters('powershelldscUpdateTagVersion')]",
                  "settings": {
                    "configuration": {
                      "url": "[variables('powershelldscZipFullPath')]",
                      "script": "InstallIIS.ps1",
                      "function": "InstallIIS"
                    },
                    "configurationArguments": {
                      "nodeName": "localhost",
                      "WebDeployPackagePath": "
[variables('webDeployPackageFullPath')]"
                    }
                  }
                }
              }
            ]
          }
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/virtualNetworks',
variables('vNetName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2021-05-01",
      "name": "[variables('publicIPAddressName')]",
      "location": "[parameters('location')]",
      "properties": {
        "publicIPAllocationMethod": "Static",
        "dnsSettings": {
          "domainNameLabel": "[variables('longvmScaleSet')]"
        }
      }
    },
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "2021-05-01",
      "name": "[variables('vNetName')]",
      "location": "[parameters('location')]",
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[variables('addressPrefix')]"
          ]
        },
        "subnets": [
          {
```

```
            "name": "[variables('subnetName')]",
            "properties": {
              "addressPrefix": "[variables('subnetPrefix')]"
            }
          }
        ]
      }
    },
    {
      "type": "Microsoft.Insights/autoscalesettings",
      "apiVersion": "2021-05-01-preview",
      "name": "autoscalehost",
      "location": "[parameters('location')]",
      "properties": {
        "name": "autoscalehost",
        "targetResourceUri": "
[resourceId('Microsoft.Compute/virtualMachineScaleSets',
variables('vmScaleSetName'))]",
        "enabled": true,
        "profiles": [
          {
            "name": "Profile1",
            "capacity": {
              "minimum": "1",
              "maximum": "10",
              "default": "1"
            },
            "rules": [
              {
                "metricTrigger": {
                  "metricName": "Percentage CPU",
                  "metricResourceUri": "
[resourceId('Microsoft.Compute/virtualMachineScaleSets',
variables('vmScaleSetName'))]",
                  "timeGrain": "PT1M",
                  "statistic": "Average",
                  "timeWindow": "PT5M",
                  "timeAggregation": "Average",
                  "operator": "GreaterThan",
                  "threshold": 50
                },
                "scaleAction": {
                  "direction": "Increase",
                  "type": "ChangeCount",
                  "value": "1",
                  "cooldown": "PT5M"
                }
              },
              {
                "metricTrigger": {
                  "metricName": "Percentage CPU",
                  "metricResourceUri": "
[resourceId('Microsoft.Compute/virtualMachineScaleSets',
variables('vmScaleSetName'))]",
                  "timeGrain": "PT1M",
```

```
                    "statistic": "Average",
                    "timeWindow": "PT5M",
                    "timeAggregation": "Average",
                    "operator": "LessThan",
                    "threshold": 30
                  },
                  "scaleAction": {
                    "direction": "Decrease",
                    "type": "ChangeCount",
                    "value": "1",
                    "cooldown": "PT5M"
                  }
                }
              ]
            }
          ]
        },
        "dependsOn": [
          "[resourceId('Microsoft.Compute/virtualMachineScaleSets',
variables('vmScaleSetName'))]"
        ]
      }
    ],
    "outputs": {
      "applicationUrl": {
        "type": "string",
        "value": "[uri(format('http://{0}',
reference(resourceId('Microsoft.Network/publicIPAddresses',
variables('publicIPAddressName'))).dnsSettings.fqdn), '/MyApp')]"
      }
    }
}
```

These resources are defined in these templates:

- Microsoft.Network/virtualNetworks
- Microsoft.Network/publicIPAddresses
- Microsoft.Network/loadBalancers
- Microsoft.Compute/virtualMachineScaleSets
- Microsoft.Insights/autoscaleSettings

Define a scale set

To create a scale with a template, you define the appropriate resources. The core parts of the Virtual Machine Scale Set resource type are:

Property	Description of property	Example template value

Property	Description of property	Example template value
type	Azure resource type to create	Microsoft.Compute/virtualMachineScaleSets
name	The scale set name	myScaleSet
location	The location to create the scale set	East US
sku.name	The VM size for each scale set instance	Standard_A1
sku.capacity	The number of VM instances to initially create	2
upgradePolicy.mode	VM instance upgrade mode when changes occur	Automatic
imageReference	The platform or custom image to use for the VM instances	Microsoft Windows Server 2016 Datacenter
osProfile.computerNamePrefix	The name prefix for each VM instance	myvmss
osProfile.adminUsername	The username for each VM instance	azureuser
osProfile.adminPassword	The password for each VM instance	P@ssw0rd!

To customize a scale set template, you can change the VM size or initial capacity. Another option is to use a different platform or a custom image.

Add a sample application

To test your scale set, install a basic web application. When you deploy a scale set, VM extensions can provide post-deployment configuration and automation tasks, such as installing an app. Scripts can be downloaded from Azure storage or GitHub, or provided

to the Azure portal at extension run-time. To apply an extension to your scale set, you add the *extensionProfile* section to the preceding resource example. The extension profile typically defines the following properties:

- Extension type
- Extension publisher
- Extension version
- Location of configuration or install scripts
- Commands to execute on the VM instances

The template uses the PowerShell DSC extension to install an ASP.NET MVC app that runs in IIS.

An install script is downloaded from GitHub, as defined in *url*. The extension then runs *InstallIIS* from the *IISInstall.ps1* script, as defined in *function* and *Script*. The ASP.NET app itself is provided as a Web Deploy package, which is also downloaded from GitHub, as defined in *WebDeployPackagePath*:

Deploy the template

You can deploy the template by selecting the **Deploy to Azure** button. This button opens the Azure portal, loads the complete template, and prompts for a few parameters such as a scale set name, instance count, and admin credentials.

Deploy to Azure

You can also deploy a Resource Manager template by using Azure PowerShell:

Azure PowerShell

```
# Create a resource group
New-AzResourceGroup -Name myResourceGroup -Location EastUS

# Deploy template into resource group
New-AzResourceGroupDeployment `
    -ResourceGroupName myResourceGroup `
    -TemplateURI https://raw.githubusercontent.com/Azure/azure-quickstart-
templates/master/demos/vmss-windows-webapp-dsc-autoscale/azuredeploy.json

# Update the scale set and apply the extension
Update-AzVmss `
    -ResourceGroupName myResourceGroup `
    -VmScaleSetName myVMSS `
    -VirtualMachineScaleSet $vmssConfig
```

Answer the prompts to provide a scale set name and admin credentials for the VM instances. It can take 10-15 minutes for the scale set to be created and apply the extension to configure the app.

Validate the deployment

To see your scale set in action, access the sample web application in a web browser. Obtain the public IP address of your load balancer with Get-AzPublicIpAddress as follows:

Azure PowerShell

```
Get-AzPublicIpAddress -ResourceGroupName myResourceGroup | Select IpAddress
```

Enter the public IP address of the load balancer in to a web browser in the format *http://publicIpAddress/MyApp*. The load balancer distributes traffic to one of your VM instances, as shown in the following example:

Clean up resources

When no longer needed, you can use the Remove-AzResourceGroup to remove the resource group, scale set. The `-Force` parameter confirms that you wish to delete the resources without an additional prompt to do so. The `-AsJob` parameter returns control to the prompt without waiting for the operation to complete.

Azure PowerShell

```
Remove-AzResourceGroup -Name "myResourceGroup" -Force -AsJob
```

Next steps

In this quickstart, you created a Windows scale set with an ARM template and used the PowerShell DSC extension to install a basic ASP.NET app on the VM instances. To learn more, continue to the tutorial for how to create and manage Azure Virtual Machine Scale Sets.

Create and manage Azure Virtual Machine Scale Sets

Quickstart: Create a Windows virtual machine using a template

Article • 04/05/2023

Applies to: ✔ Windows VMs

This quickstart shows you how to use an Azure Resource Manager template to deploy a Windows virtual machine (VM) in Azure.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using templates, select the **Deploy to Azure** button. The template opens in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.15.31.15270",
      "templateHash": "3941101056511835882"
    }
  },
  "parameters": {
    "adminUsername": {
      "type": "string",
```

```
      "metadata": {
        "description": "Username for the Virtual Machine."
      }
    },
    "adminPassword": {
      "type": "securestring",
      "minLength": 12,
      "metadata": {
        "description": "Password for the Virtual Machine."
      }
    },
    "dnsLabelPrefix": {
      "type": "string",
      "defaultValue": "[toLower(format('{0}-{1}', parameters('vmName'),
uniqueString(resourceGroup().id, parameters('vmName'))))]",
      "metadata": {
        "description": "Unique DNS Name for the Public IP used to access the
Virtual Machine."
      }
    },
    "publicIpName": {
      "type": "string",
      "defaultValue": "myPublicIP",
      "metadata": {
        "description": "Name for the Public IP used to access the Virtual
Machine."
      }
    },
    "publicIPAllocationMethod": {
      "type": "string",
      "defaultValue": "Dynamic",
      "allowedValues": [
        "Dynamic",
        "Static"
      ],
      "metadata": {
        "description": "Allocation method for the Public IP used to access
the Virtual Machine."
      }
    },
    "publicIpSku": {
      "type": "string",
      "defaultValue": "Basic",
      "allowedValues": [
        "Basic",
        "Standard"
      ],
      "metadata": {
        "description": "SKU for the Public IP used to access the Virtual
Machine."
      }
    },
    "OSVersion": {
      "type": "string",
      "defaultValue": "2022-datacenter-azure-edition",
```

```
      "allowedValues": [
        "2016-datacenter-gensecond",
        "2016-datacenter-server-core-g2",
        "2016-datacenter-server-core-smalldisk-g2",
        "2016-datacenter-smalldisk-g2",
        "2016-datacenter-with-containers-g2",
        "2016-datacenter-zhcn-g2",
        "2019-datacenter-core-g2",
        "2019-datacenter-core-smalldisk-g2",
        "2019-datacenter-core-with-containers-g2",
        "2019-datacenter-core-with-containers-smalldisk-g2",
        "2019-datacenter-gensecond",
        "2019-datacenter-smalldisk-g2",
        "2019-datacenter-with-containers-g2",
        "2019-datacenter-with-containers-smalldisk-g2",
        "2019-datacenter-zhcn-g2",
        "2022-datacenter-azure-edition",
        "2022-datacenter-azure-edition-core",
        "2022-datacenter-azure-edition-core-smalldisk",
        "2022-datacenter-azure-edition-smalldisk",
        "2022-datacenter-core-g2",
        "2022-datacenter-core-smalldisk-g2",
        "2022-datacenter-g2",
        "2022-datacenter-smalldisk-g2"
      ],
      "metadata": {
        "description": "The Windows version for the VM. This will pick a
fully patched image of this given Windows version."
      }
    },
    "vmSize": {
      "type": "string",
      "defaultValue": "Standard_D2s_v5",
      "metadata": {
        "description": "Size of the virtual machine."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "vmName": {
      "type": "string",
      "defaultValue": "simple-vm",
      "metadata": {
        "description": "Name of the virtual machine."
      }
    },
    "securityType": {
      "type": "string",
      "defaultValue": "TrustedLaunch",
      "allowedValues": [
```

```
        "Standard",
        "TrustedLaunch"
      ],
      "metadata": {
        "description": "Security Type of the Virtual Machine."
      }
    }
  },
  "variables": {
    "storageAccountName": "[format('bootdiags{0}',
uniqueString(resourceGroup().id))]",
    "nicName": "myVMNic",
    "addressPrefix": "10.0.0.0/16",
    "subnetName": "Subnet",
    "subnetPrefix": "10.0.0.0/24",
    "virtualNetworkName": "MyVNET",
    "networkSecurityGroupName": "default-NSG",
    "securityProfileJson": {
      "uefiSettings": {
        "secureBootEnabled": true,
        "vTpmEnabled": true
      },
      "securityType": "[parameters('securityType')]"
    },
    "extensionName": "GuestAttestation",
    "extensionPublisher": "Microsoft.Azure.Security.WindowsAttestation",
    "extensionVersion": "1.0",
    "maaTenantName": "GuestAttestation",
    "maaEndpoint": "[substring('emptyString', 0, 0)]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2022-05-01",
      "name": "[variables('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "Storage"
    },
    {
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2022-05-01",
      "name": "[parameters('publicIpName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('publicIpSku')]"
      },
      "properties": {
        "publicIPAllocationMethod": "
[parameters('publicIPAllocationMethod')]",
        "dnsSettings": {
          "domainNameLabel": "[parameters('dnsLabelPrefix')]"
        }
```

```
      }
    },
    {
      "type": "Microsoft.Network/networkSecurityGroups",
      "apiVersion": "2022-05-01",
      "name": "[variables('networkSecurityGroupName')]",
      "location": "[parameters('location')]",
      "properties": {
        "securityRules": [
          {
            "name": "default-allow-3389",
            "properties": {
              "priority": 1000,
              "access": "Allow",
              "direction": "Inbound",
              "destinationPortRange": "3389",
              "protocol": "Tcp",
              "sourcePortRange": "*",
              "sourceAddressPrefix": "*",
              "destinationAddressPrefix": "*"
            }
          }
        ]
      }
    },
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "2022-05-01",
      "name": "[variables('virtualNetworkName')]",
      "location": "[parameters('location')]",
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[variables('addressPrefix')]"
          ]
        },
        "subnets": [
          {
            "name": "[variables('subnetName')]",
            "properties": {
              "addressPrefix": "[variables('subnetPrefix')]",
              "networkSecurityGroup": {
                "id": "
[resourceId('Microsoft.Network/networkSecurityGroups',
variables('networkSecurityGroupName'))]"
              }
            }
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('networkSecurityGroupName'))]"
      ]
    },
```

```
    {
      "type": "Microsoft.Network/networkInterfaces",
      "apiVersion": "2022-05-01",
      "name": "[variables('nicName')]",
      "location": "[parameters('location')]",
      "properties": {
        "ipConfigurations": [
          {
            "name": "ipconfig1",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses',
parameters('publicIpName'))]"
              },
              "subnet": {
                "id": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
variables('virtualNetworkName'), variables('subnetName'))]"
              }
            }
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/publicIPAddresses',
parameters('publicIpName'))]",
        "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]"
      ]
    },
    {
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2022-03-01",
      "name": "[parameters('vmName')]",
      "location": "[parameters('location')]",
      "properties": {
        "hardwareProfile": {
          "vmSize": "[parameters('vmSize')]"
        },
        "osProfile": {
          "computerName": "[parameters('vmName')]",
          "adminUsername": "[parameters('adminUsername')]",
          "adminPassword": "[parameters('adminPassword')]"
        },
        "storageProfile": {
          "imageReference": {
            "publisher": "MicrosoftWindowsServer",
            "offer": "WindowsServer",
            "sku": "[parameters('OSVersion')]",
            "version": "latest"
          },
          "osDisk": {
            "createOption": "FromImage",
            "managedDisk": {
```

```
              "storageAccountType": "StandardSSD_LRS"
            }
          },
          "dataDisks": [
            {
              "diskSizeGB": 1023,
              "lun": 0,
              "createOption": "Empty"
            }
          ]
        },
        "networkProfile": {
          "networkInterfaces": [
            {
              "id": "[resourceId('Microsoft.Network/networkInterfaces',
variables('nicName'))]"
            }
          ]
        },
        "diagnosticsProfile": {
          "bootDiagnostics": {
            "enabled": true,
            "storageUri": "
[reference(resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName')), '2022-05-01').primaryEndpoints.blob]"
          }
        },
        "securityProfile": "[if(equals(parameters('securityType'),
'TrustedLaunch'), variables('securityProfileJson'), json('null'))]"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkInterfaces',
variables('nicName'))]",
        "[resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))]"
      ]
    },
    {
      "condition": "[and(equals(parameters('securityType'),
'TrustedLaunch'),
and(equals(variables('securityProfileJson').uefiSettings.secureBootEnabled,
true()), equals(variables('securityProfileJson').uefiSettings.vTpmEnabled,
true())))]",
      "type": "Microsoft.Compute/virtualMachines/extensions",
      "apiVersion": "2022-03-01",
      "name": "[format('{0}/{1}', parameters('vmName'),
variables('extensionName'))]",
      "location": "[parameters('location')]",
      "properties": {
        "publisher": "[variables('extensionPublisher')]",
        "type": "[variables('extensionName')]",
        "typeHandlerVersion": "[variables('extensionVersion')]",
        "autoUpgradeMinorVersion": true,
        "settings": {
          "AttestationConfig": {
```

```
              "MaaSettings": {
                "maaEndpoint": "[variables('maaEndpoint')]",
                "maaTenantName": "[variables('maaTenantName')]"
              }
            }
          }
        },
        "dependsOn": [
          "[resourceId('Microsoft.Compute/virtualMachines',
parameters('vmName'))]"
        ]
      }
    ],
    "outputs": {
      "hostname": {
        "type": "string",
        "value": "[reference(resourceId('Microsoft.Network/publicIPAddresses',
parameters('publicIpName')), '2022-05-01').dnsSettings.fqdn]"
      }
    }
  }
```

Several resources are defined in the template:

- Microsoft.Network/virtualNetworks/subnets: create a subnet.
- Microsoft.Storage/storageAccounts: create a storage account.
- Microsoft.Network/publicIPAddresses: create a public IP address.
- Microsoft.Network/networkSecurityGroups: create a network security group.
- Microsoft.Network/virtualNetworks: create a virtual network.
- Microsoft.Network/networkInterfaces: create a NIC.
- Microsoft.Compute/virtualMachines: create a virtual machine.

Deploy the template

1. Select the following image to sign in to Azure and open a template. The template creates a key vault and a secret.

   Deploy to Azure

2. Select or enter the following values. Use the default values, when available.

   - **Subscription**: select an Azure subscription.
   - **Resource group**: select an existing resource group from the drop-down, or select **Create new**, enter a unique name for the resource group, and then select **OK**.
   - **Region**: select a region. For example, **Central US**.

- **Admin username**: provide a username, such as *azureuser*.
- **Admin password**: provide a password to use for the admin account. The password must be at least 12 characters long and meet the defined complexity requirements.
- **DNS label prefix**: enter a unique identifier to use as part of the DNS label.
- **OS version**: select which OS version you want to run on the VM.
- **VM size**: select the size to use for the VM.
- **Location**: the default is the same location as the resource group, if it already exists.

3. Select **Review + create**. After validation completes, select **Create** to create and deploy the VM.

The Azure portal is used to deploy the template. In addition to the Azure portal, you can also use the Azure PowerShell, Azure CLI, and REST API. To learn other deployment methods, see Deploy templates.

Review deployed resources

You can use the Azure portal to check on the VM and other resource that were created. After the deployment is finished, select **Resource groups** to see the VM and other resources.

Clean up resources

When no longer needed, delete the resource group, which deletes the VM and all of the resources in the resource group.

1. Select the **Resource group**.
2. On the page for the resource group, select **Delete resource group**.
3. When prompted, type the name of the resource group and then select **Delete**.

Next steps

In this quickstart, you deployed a simple virtual machine using a Resource Manager template. To learn more about Azure virtual machines, continue to the tutorial for managing VMs.

Azure Windows virtual machine tutorials

Quickstart: Deploy an Azure Kubernetes Service (AKS) cluster using an ARM template

Article • 03/09/2023

Azure Kubernetes Service (AKS) is a managed Kubernetes service that lets you quickly deploy and manage clusters. In this quickstart, you will:

- Deploy an AKS cluster using an Azure Resource Manager template.
- Run a sample multi-container application with a web front-end and a Redis instance in the cluster.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

This quickstart assumes a basic understanding of Kubernetes concepts. For more information, see Kubernetes core concepts for Azure Kubernetes Service (AKS).

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure

portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create an Azure free account before you begin.

Azure CLI

- Use the Bash environment in Azure Cloud Shell. For more information, see Quickstart for Bash in Azure Cloud Shell.

  Launch Cloud Shell

- If you prefer to run CLI reference commands locally, install the Azure CLI. If you're running on Windows or macOS, consider running Azure CLI in a Docker container. For more information, see How to run the Azure CLI in a Docker container.
  - If you're using a local installation, sign in to the Azure CLI by using the az login command. To finish the authentication process, follow the steps displayed in your terminal. For other sign-in options, see Sign in with the Azure CLI.
  - When you're prompted, install the Azure CLI extension on first use. For more information about extensions, see Use extensions with the Azure CLI.
  - Run az version to find the version and dependent libraries that are installed. To upgrade to the latest version, run az upgrade.
- This article requires version 2.0.64 or later of the Azure CLI. If using Azure Cloud Shell, the latest version is already installed.

- To create an AKS cluster using a Resource Manager template, you provide an SSH public key. If you need this resource, see the following section; otherwise skip to the Review the template section.
- The identity you are using to create your cluster has the appropriate minimum permissions. For more details on access and identity for AKS, see Access and

identity options for Azure Kubernetes Service (AKS).

- To deploy a Bicep file or ARM template, you need write access on the resources you're deploying and access to all operations on the Microsoft.Resources/deployments resource type. For example, to deploy a virtual machine, you need Microsoft.Compute/virtualMachines/write and Microsoft.Resources/deployments/* permissions. For a list of roles and permissions, see Azure built-in roles.

Create an SSH key pair

To access AKS nodes, you connect using an SSH key pair (public and private), which you generate using the `ssh-keygen` command. By default, these files are created in the *~/.ssh* directory. Running the `ssh-keygen` command will overwrite any SSH key pair with the same name already existing in the given location.

1. Go to https://shell.azure.com to open Cloud Shell in your browser.
2. Run the `ssh-keygen` command. The following example creates an SSH key pair using RSA encryption and a bit length of 4096:

Console

```
ssh-keygen -t rsa -b 4096
```

For more information about creating SSH keys, see Create and manage SSH keys for authentication in Azure.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.9.1.41621",
      "templateHash": "2637152180661081755"
    }
  },
```

```
  "parameters": {
    "clusterName": {
      "type": "string",
      "defaultValue": "aks101cluster",
      "metadata": {
        "description": "The name of the Managed Cluster resource."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "The location of the Managed Cluster resource."
      }
    },
    "dnsPrefix": {
      "type": "string",
      "metadata": {
        "description": "Optional DNS prefix to use with hosted Kubernetes
API server FQDN."
      }
    },
    "osDiskSizeGB": {
      "type": "int",
      "defaultValue": 0,
      "maxValue": 1023,
      "minValue": 0,
      "metadata": {
        "description": "Disk size (in GB) to provision for each of the agent
pool nodes. This value ranges from 0 to 1023. Specifying 0 will apply the
default disk size for that agentVMSize."
      }
    },
    "agentCount": {
      "type": "int",
      "defaultValue": 3,
      "maxValue": 50,
      "minValue": 1,
      "metadata": {
        "description": "The number of nodes for the cluster."
      }
    },
    "agentVMSize": {
      "type": "string",
      "defaultValue": "standard_d2s_v3",
      "metadata": {
        "description": "The size of the Virtual Machine."
      }
    },
    "linuxAdminUsername": {
      "type": "string",
      "metadata": {
        "description": "User name for the Linux Virtual Machines."
      }
    },
```

```
    "sshRSAPublicKey": {
      "type": "string",
      "metadata": {
        "description": "Configure all linux machines with the SSH RSA public
key string. Your key should include three parts, for example 'ssh-rsa
AAAAB...snip...UcyupgH azureuser@linuxvm'"
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.ContainerService/managedClusters",
      "apiVersion": "2022-05-02-preview",
      "name": "[parameters('clusterName')]",
      "location": "[parameters('location')]",
      "identity": {
        "type": "SystemAssigned"
      },
      "properties": {
        "dnsPrefix": "[parameters('dnsPrefix')]",
        "agentPoolProfiles": [
          {
            "name": "agentpool",
            "osDiskSizeGB": "[parameters('osDiskSizeGB')]",
            "count": "[parameters('agentCount')]",
            "vmSize": "[parameters('agentVMSize')]",
            "osType": "Linux",
            "mode": "System"
          }
        ],
        "linuxProfile": {
          "adminUsername": "[parameters('linuxAdminUsername')]",
          "ssh": {
            "publicKeys": [
              {
                "keyData": "[parameters('sshRSAPublicKey')]"
              }
            ]
          }
        }
      }
    }
  ],
  "outputs": {
    "controlPlaneFQDN": {
      "type": "string",
      "value": "
[reference(resourceId('Microsoft.ContainerService/managedClusters',
parameters('clusterName'))).fqdn]"
    }
  }
}
```

The resource defined in the ARM template includes:

- **Microsoft.ContainerService/managedClusters**

For more AKS samples, see the AKS quickstart templates site.

Deploy the template

1. Select the following button to sign in to Azure and open a template.

   Deploy to Azure

2. Select or enter the following values.

   For this quickstart, leave the default values for the *OS Disk Size GB*, *Agent Count*, *Agent VM Size*, *OS Type*, and *Kubernetes Version*. Provide your own values for the following template parameters:

   - **Subscription**: Select an Azure subscription.
   - **Resource group**: Select **Create new**. Enter a unique name for the resource group, such as *myResourceGroup*, then choose **OK**.
   - **Location**: Select a location, such as **East US**.
   - **Cluster name**: Enter a unique name for the AKS cluster, such as *myAKSCluster*.
   - **DNS prefix**: Enter a unique DNS prefix for your cluster, such as *myakscluster*.
   - **Linux Admin Username**: Enter a username to connect using SSH, such as *azureuser*.
   - **SSH RSA Public Key**: Copy and paste the *public* part of your SSH key pair (by default, the contents of *~/.ssh/id_rsa.pub*).

3. Select **Review + Create**.

It takes a few minutes to create the AKS cluster. Wait for the cluster to be successfully deployed before you move on to the next step.

Validate the deployment

Connect to the cluster

To manage a Kubernetes cluster, use the Kubernetes command-line client, kubectl. `kubectl` is already installed if you use Azure Cloud Shell.

Azure CLI

1. Install `kubectl` locally using the az aks install-cli command:

   Azure CLI

   ```
   az aks install-cli
   ```

2. Configure `kubectl` to connect to your Kubernetes cluster using the az aks get-credentials command. This command downloads credentials and configures the Kubernetes CLI to use them.

   Azure CLI

   ```
   az aks get-credentials --resource-group myResourceGroup --name
   myAKSCluster
   ```

3. Verify the connection to your cluster using the kubectl get command. This command returns a list of the cluster nodes.

   Console

   ```
   kubectl get nodes
   ```

   The following output example shows the three nodes created in the previous steps. Make sure the node status is *Ready*:

   Output

   ```
   NAME                       STATUS   ROLES   AGE     VERSION
   aks-agentpool-41324942-0   Ready    agent   6m44s   v1.12.6
   aks-agentpool-41324942-1   Ready    agent   6m46s   v1.12.6
   aks-agentpool-41324942-2   Ready    agent   6m45s   v1.12.6
   ```

Deploy the application

A Kubernetes manifest file defines a cluster's desired state, such as which container images to run.

In this quickstart, you will use a manifest to create all objects needed to run the Azure Vote application. This manifest includes two Kubernetes deployments:

- The sample Azure Vote Python applications.
- A Redis instance.

Two Kubernetes Services are also created:

- An internal service for the Redis instance.
- An external service to access the Azure Vote application from the internet.

1. Create a file named `azure-vote.yaml`.

2. Copy in the following YAML definition:

YAML

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: azure-vote-back
spec:
  replicas: 1
  selector:
    matchLabels:
      app: azure-vote-back
  template:
    metadata:
      labels:
        app: azure-vote-back
    spec:
      nodeSelector:
        "kubernetes.io/os": linux
      containers:
      - name: azure-vote-back
        image: mcr.microsoft.com/oss/bitnami/redis:6.0.8
        env:
        - name: ALLOW_EMPTY_PASSWORD
          value: "yes"
        resources:
          requests:
            cpu: 100m
            memory: 128Mi
          limits:
            cpu: 250m
            memory: 256Mi
        ports:
        - containerPort: 6379
          name: redis
---
apiVersion: v1
kind: Service
metadata:
  name: azure-vote-back
spec:
  ports:
  - port: 6379
  selector:
    app: azure-vote-back
```

```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: azure-vote-front
spec:
  replicas: 1
  selector:
    matchLabels:
      app: azure-vote-front
  template:
    metadata:
      labels:
        app: azure-vote-front
    spec:
      nodeSelector:
        "kubernetes.io/os": linux
      containers:
      - name: azure-vote-front
        image: mcr.microsoft.com/azuredocs/azure-vote-front:v1
        resources:
          requests:
            cpu: 100m
            memory: 128Mi
          limits:
            cpu: 250m
            memory: 256Mi
        ports:
        - containerPort: 80
        env:
        - name: REDIS
          value: "azure-vote-back"
---
apiVersion: v1
kind: Service
metadata:
  name: azure-vote-front
spec:
  type: LoadBalancer
  ports:
  - port: 80
  selector:
    app: azure-vote-front
```

For a breakdown of YAML manifest files, see Deployments and YAML manifests.

3. Deploy the application using the kubectl apply command and specify the name of your YAML manifest:

Console

```
kubectl apply -f azure-vote.yaml
```

The following example resembles output showing the successfully created deployments and services:

Output

```
deployment "azure-vote-back" created
service "azure-vote-back" created
deployment "azure-vote-front" created
service "azure-vote-front" created
```

Test the application

When the application runs, a Kubernetes service exposes the application front end to the internet. This process can take a few minutes to complete.

Monitor progress using the kubectl get service command with the `--watch` argument.

Console

```
kubectl get service azure-vote-front --watch
```

The **EXTERNAL-IP** output for the `azure-vote-front` service will initially show as *pending*.

Output

```
NAME                TYPE           CLUSTER-IP   EXTERNAL-IP   PORT(S)
AGE
azure-vote-front    LoadBalancer   10.0.37.27   <pending>     80:30572/TCP
6s
```

Once the **EXTERNAL-IP** address changes from *pending* to an actual public IP address, use `CTRL-C` to stop the `kubectl` watch process. The following example output shows a valid public IP address assigned to the service:

Output

```
azure-vote-front    LoadBalancer   10.0.37.27   52.179.23.131   80:30572/TCP
2m
```

To see the Azure Vote app in action, open a web browser to the external IP address of your service.

Clean up resources

Azure CLI

To avoid Azure charges, if you don't plan on going through the tutorials that follow, clean up your unnecessary resources. Use the az group delete command to remove the resource group, container service, and all related resources.

Azure CLI

```
az group delete --name myResourceGroup --yes --no-wait
```

Note

The AKS cluster was created with system-assigned managed identity (default identity option used in this quickstart), the identity is managed by the platform and does not require removal.

Next steps

In this quickstart, you deployed a Kubernetes cluster and then deployed a sample multi-container application to it.

To learn more about AKS, and walk through a complete code to deployment example, continue to the Kubernetes cluster tutorial.

AKS tutorial

Quickstart: Deploy a container instance in Azure using an ARM template

Article • 04/09/2023

Use Azure Container Instances to run serverless Docker containers in Azure with simplicity and speed. Deploy an application to a container instance on-demand when you don't need a full container orchestration platform like Azure Kubernetes Service. In this quickstart, you use an Azure Resource Manager template (ARM template) to deploy an isolated Docker container and make its web application available with a public IP address.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "17016281914347876853"
    }
```

```
  },
  "parameters": {
    "name": {
      "type": "string",
      "defaultValue": "acilinuxpublicipcontainergroup",
      "metadata": {
        "description": "Name for the container group"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "image": {
      "type": "string",
      "defaultValue": "mcr.microsoft.com/azuredocs/aci-helloworld",
      "metadata": {
        "description": "Container image to deploy. Should be of the form
repoName/imagename:tag for images stored in public Docker Hub, or a fully
qualified URI for other registries. Images from private registries require
additional registry credentials."
      }
    },
    "port": {
      "type": "int",
      "defaultValue": 80,
      "metadata": {
        "description": "Port to open on the container and the public IP
address."
      }
    },
    "cpuCores": {
      "type": "int",
      "defaultValue": 1,
      "metadata": {
        "description": "The number of CPU cores to allocate to the
container."
      }
    },
    "memoryInGb": {
      "type": "int",
      "defaultValue": 2,
      "metadata": {
        "description": "The amount of memory to allocate to the container in
gigabytes."
      }
    },
    "restartPolicy": {
      "type": "string",
      "defaultValue": "Always",
      "allowedValues": [
        "Always",
```

```
        "Never",
        "OnFailure"
      ],
      "metadata": {
        "description": "The behavior of Azure runtime if container has
stopped."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.ContainerInstance/containerGroups",
      "apiVersion": "2021-09-01",
      "name": "[parameters('name')]",
      "location": "[parameters('location')]",
      "properties": {
        "containers": [
          {
            "name": "[parameters('name')]",
            "properties": {
              "image": "[parameters('image')]",
              "ports": [
                {
                  "port": "[parameters('port')]",
                  "protocol": "TCP"
                }
              ],
              "resources": {
                "requests": {
                  "cpu": "[parameters('cpuCores')]",
                  "memoryInGB": "[parameters('memoryInGb')]"
                }
              }
            }
          }
        ],
        "osType": "Linux",
        "restartPolicy": "[parameters('restartPolicy')]",
        "ipAddress": {
          "type": "Public",
          "ports": [
            {
              "port": "[parameters('port')]",
              "protocol": "TCP"
            }
          ]
        }
      }
    }
  ],
  "outputs": {
    "containerIPv4Address": {
      "type": "string",
      "value": "
[reference(resourceId('Microsoft.ContainerInstance/containerGroups',
```

```
parameters('name'))).ipAddress.ip]"
    }
  }
}
```

The following resource is defined in the template:

- **Microsoft.ContainerInstance/containerGroups**: create an Azure container group. This template defines a group consisting of a single container instance.

More Azure Container Instances template samples can be found in the quickstart template gallery.

Deploy the template

1. Select the following image to sign in to Azure and open a template. The template creates a registry and a replica in another location.

   Deploy to Azure

2. Select or enter the following values.

   - **Subscription**: select an Azure subscription.
   - **Resource group**: select **Create new**, enter a unique name for the resource group, and then select **OK**.
   - **Location**: select a location for the resource group. Example: **Central US**.
   - **Name**: accept the generated name for the instance, or enter a name.
   - **Image**: accept the default image name. This sample Linux image packages a small web app written in Node.js that serves a static HTML page.

   Accept default values for the remaining properties.

   Review the terms and conditions. If you agree, select **I agree to the terms and conditions stated above**.

3. After the instance has been created successfully, you get a notification:

The Azure portal is used to deploy the template. In addition to the Azure portal, you can use the Azure PowerShell, Azure CLI, and REST API. To learn other deployment methods, see Deploy templates.

Review deployed resources

Use the Azure portal or a tool such as the Azure CLI to review the properties of the container instance.

1. In the portal, search for Container Instances, and select the container instance you created.

2. On the **Overview** page, note the **Status** of the instance and its **IP address**.

3. Once its status is *Running*, navigate to the IP address in your browser.

View container logs

Viewing the logs for a container instance is helpful when troubleshooting issues with your container or the application it runs.

To view the container's logs, under **Settings**, select **Containers** > **Logs**. You should see the HTTP GET request generated when you viewed the application in your browser.

Clean up resources

When you're done with the container, on the **Overview** page for the container instance, select **Delete**. When prompted, confirm the deletion.

Next steps

In this quickstart, you created an Azure container instance from a public Microsoft image. If you'd like to build a container image and deploy it from a private Azure container registry, continue to the Azure Container Instances tutorial.

Tutorial: Create a container image for deployment to Azure Container Instances

For a step-by-step tutorial that guides you through the process of creating a template, see:

Tutorial: Create and deploy your first ARM template

Quickstart: Create a geo-replicated container registry by using an ARM template

Article • 04/09/2023

This quickstart shows how to create an Azure Container Registry instance by using an Azure Resource Manager template (ARM template). The template sets up a geo-replicated registry, which automatically synchronizes registry content across more than one Azure region. Geo-replication enables network-close access to images from regional deployments, while providing a single management experience. It's a feature of the Premium registry service tier.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

The registry with replications does not support the ARM/Bicep template Complete mode deployments.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates. The template sets up a registry and an additional regional replica.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
```

```
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "12610175857982700190"
    }
  },
  "parameters": {
    "acrName": {
      "type": "string",
      "defaultValue": "[format('acr{0}',
uniqueString(resourceGroup().id))]",
      "maxLength": 50,
      "minLength": 5,
      "metadata": {
        "description": "Globally unique name of your Azure Container
Registry"
      }
    },
    "acrAdminUserEnabled": {
      "type": "bool",
      "defaultValue": false,
      "metadata": {
        "description": "Enable admin user that has push / pull permission to
the registry."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for registry home replica."
      }
    },
    "acrSku": {
      "type": "string",
      "defaultValue": "Premium",
      "allowedValues": [
        "Premium"
      ],
      "metadata": {
        "description": "Tier of your Azure Container Registry. Geo-
replication requires Premium SKU."
      }
    },
    "acrReplicaLocation": {
      "type": "string",
      "metadata": {
        "description": "Short name for registry replica location."
      }
    }
  },
  "resources": [
    {
```

```
      "type": "Microsoft.ContainerRegistry/registries",
      "apiVersion": "2019-12-01-preview",
      "name": "[parameters('acrName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('acrSku')]"
      },
      "tags": {
        "displayName": "Container Registry",
        "container.registry": "[parameters('acrName')]"
      },
      "properties": {
        "adminUserEnabled": "[parameters('acrAdminUserEnabled')]"
      }
    },
    {
      "type": "Microsoft.ContainerRegistry/registries/replications",
      "apiVersion": "2019-12-01-preview",
      "name": "[format('{0}/{1}', parameters('acrName'),
parameters('acrReplicaLocation'))]",
      "location": "[parameters('acrReplicaLocation')]",
      "properties": {},
      "dependsOn": [
        "[resourceId('Microsoft.ContainerRegistry/registries',
parameters('acrName'))]"
      ]
    }
  ],
  "outputs": {
    "acrLoginServer": {
      "type": "string",
      "value": "
[reference(resourceId('Microsoft.ContainerRegistry/registries',
parameters('acrName'))).loginServer]"
    }
  }
}
```

The following resources are defined in the template:

- **Microsoft.ContainerRegistry/registries**: create an Azure container registry
- **Microsoft.ContainerRegistry/registries/replications**: create a container registry replica

More Azure Container Registry template samples can be found in the quickstart template gallery.

Deploy the template

1. Select the following image to sign in to Azure and open a template.

Deploy to Azure

2. Select or enter the following values.

- **Subscription**: select an Azure subscription.
- **Resource group**: select **Create new**, enter a unique name for the resource group, and then select **OK**.
- **Region**: select a location for the resource group. Example: **Central US**.
- **Acr Name**: accept the generated name for the registry, or enter a name. It must be globally unique.
- **Acr Admin User Enabled**: accept the default value.
- **Location**: accept the generated location for the registry's home replica, or enter a location such as **Central US**.
- **Acr Sku**: accept the default value.
- **Acr Replica Location**: enter a location for the registry replica, using the region's short name. It must be different from the home registry location. Example: **westeurope**.

3. Select **Review + Create**, then review the terms and conditions. If you agree, select **Create**.

4. After the registry has been created successfully, you get a notification:

The Azure portal is used to deploy the template. In addition to the Azure portal, you can use the Azure PowerShell, Azure CLI, and REST API. To learn other deployment methods, see Deploy templates.

Review deployed resources

Use the Azure portal or a tool such as the Azure CLI to review the properties of the container registry.

1. In the portal, search for Container Registries, and select the container registry you created.

2. On the **Overview** page, note the **Login server** of the registry. Use this URI when you use Docker to tag and push images to your registry. For information, see Push your first image using the Docker CLI.

3. On the **Replications** page, confirm the locations of the home replica and the replica added through the template. If desired, add more replicas on this page.

Clean up resources

When you no longer need them, delete the resource group, the registry, and the registry replica. To do so, go to the Azure portal, select the resource group that contains the registry, and then select **Delete resource group**.

Next steps

In this quickstart, you created an Azure Container Registry with an ARM template, and configured a registry replica in another location. Continue to the Azure Container Registry tutorials for a deeper look at ACR.

Azure Container Registry tutorials

For a step-by-step tutorial that guides you through the process of creating a template, see:

Tutorial: Create and deploy your first ARM template

Quickstart: Create a Service Fabric cluster using ARM template

Article • 04/13/2023

Azure Service Fabric is a distributed systems platform that makes it easy to package, deploy, and manage scalable and reliable microservices and containers. A Service Fabric *cluster* is a network-connected set of virtual machines into which your microservices are deployed and managed. This article describes how to deploy a Service Fabric test cluster in Azure using an Azure Resource Manager template (ARM template).

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

This five-node Windows cluster is secured with a self-signed certificate and thus only intended for instructional purposes (rather than production workloads). We'll use Azure PowerShell to deploy the template. In addition to Azure PowerShell, you can also use the Azure portal, Azure CLI, and REST API. To learn other deployment methods, see Deploy templates.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Install Service Fabric SDK and PowerShell modules

To complete this quickstart, you'll need to:

- Install the Service Fabric SDK and PowerShell module.
- Install Azure PowerShell.

Download the sample template and certificate helper script

Clone or download the Azure Resource Manager quickstart Templates repo. Alternatively, copy down locally the following files we'll be using from the *service-fabric-secure-cluster-5-node-1-nodetype* folder:

- New-ServiceFabricClusterCertificate.ps1
- azuredeploy.json
- azuredeploy.parameters.json

Sign in to Azure

Sign in to Azure and designate the subscription to use for creating your Service Fabric cluster.

PowerShell

```
# Sign in to your Azure account
Login-AzAccount -SubscriptionId "<subscription ID>"
```

Create a self-signed certificate stored in Key Vault

Service Fabric uses X.509 certificates to secure a cluster and provide application security features, and Key Vault to manage those certificates. Successful cluster creation requires a cluster certificate to enable node-to-node communication. For the purpose of creating this quickstart test cluster, we'll create a self-signed certificate for cluster authentication. Production workloads require certificates created using a correctly configured Windows Server certificate service or one from an approved certificate authority (CA).

PowerShell

```
# Designate unique (within cloudapp.azure.com) names for your resources
$resourceGroupName = "SFQuickstartRG"
$keyVaultName = "SFQuickstartKV"

# Create a new resource group for your Key Vault and Service Fabric cluster
New-AzResourceGroup -Name $resourceGroupName -Location SouthCentralUS

# Create a Key Vault enabled for deployment
New-AzKeyVault -VaultName $keyVaultName -ResourceGroupName
$resourceGroupName -Location SouthCentralUS -EnabledForDeployment

# Generate a certificate and upload it to Key Vault
.\scripts\New-ServiceFabricClusterCertificate.ps1
```

The script will prompt you for the following (be sure to modify *CertDNSName* and *KeyVaultName* from the example values below):

- **Password:** Password!1
- **CertDNSName:** *sfquickstart*.southcentralus.cloudapp.azure.com
- **KeyVaultName:** *SFQuickstartKV*
- **KeyVaultSecretName:** clustercert

Upon completion, the script will provide the parameter values needed for template deployment. Be sure to store these in the following variables, as they will be needed to deploy your cluster template:

PowerShell

```
$sourceVaultId = "<Source Vault Resource Id>"
$certUrlValue = "<Certificate URL>"
$certThumbprint = "<Certificate Thumbprint>"
```

Review the template

The template used in this quickstart is from Azure Quickstart Templates. The template for this article is too long to show here. To view the template, see the azuredeploy.json file.

Multiple Azure resources have been defined in the template:

- Microsoft.Storage/storageAccounts
- Microsoft.Network/virtualNetworks
- Microsoft.Network/publicIPAddresses
- Microsoft.Network/loadBalancers
- Microsoft.Compute/virtualMachineScaleSets
- Microsoft.ServiceFabric/clusters

To find more templates that are related to Azure Service Fabric, see Azure quickstart Templates.

Customize the parameters file

Open *azuredeploy.parameters.json* and edit the parameter values so that:

- **clusterName** matches the value you supplied for *CertDNSName* when creating your cluster certificate

- **adminUserName** is some value other than the default *GEN-UNIQUE* token
- **adminPassword** is some value other than the default *GEN-PASSWORD* token
- **certificateThumbprint**, **sourceVaultResourceId**, and **certificateUrlValue** are all empty string (`""`)

For example:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "clusterName": {
      "value": "sfquickstart"
    },
    "adminUsername": {
      "value": "testadm"
    },
    "adminPassword": {
      "value": "Password#1234"
    },
    "certificateThumbprint": {
      "value": ""
    },
    "sourceVaultResourceId": {
      "value": ""
    },
    "certificateUrlValue": {
      "value": ""
    }
  }
}
```

Deploy the template

Store the paths of your ARM template and parameter files in variables, then deploy the template.

PowerShell

```
$templateFilePath = "<full path to azuredeploy.json>"
$parameterFilePath = "<full path to azuredeploy.parameters.json>"

New-AzResourceGroupDeployment `
    -ResourceGroupName $resourceGroupName `
    -TemplateFile $templateFilePath `
    -TemplateParameterFile $parameterFilePath `
```

```
    -CertificateThumbprint $certThumbprint `
    -CertificateUrlValue $certUrlValue `
    -SourceVaultResourceId $sourceVaultId `
    -Verbose
```

Review deployed resources

Once the deployment completes, find the `managementEndpoint` value in the output and open the address in a web browser to view your cluster in Service Fabric Explorer.

You can also find the Service Fabric Explorer endpoint from your Service Explorer resource blade in Azure portal.

Clean up resources

When no longer needed, delete the resource group, which deletes the resources in the resource group.

PowerShell

```
$resourceGroupName = Read-Host -Prompt "Enter the Resource Group name"
Remove-AzResourceGroup -Name $resourceGroupName
Write-Host "Press [ENTER] to continue..."
```

Next, remove the cluster certificate from your local store. List installed certificates to find the thumbprint for your cluster:

PowerShell

```
Get-ChildItem Cert:\CurrentUser\My\
```

Then remove the certificate:

PowerShell

```
Get-ChildItem Cert:\CurrentUser\My\{THUMBPRINT} | Remove-Item
```

Next steps

To learn about creating a custom Azure Service Fabric cluster template, see:

Create a Service Fabric cluster Resource Manager template

Quickstart: Create an Azure Cache for Redis using an ARM template

Article • 03/09/2023

Learn how to create an Azure Resource Manager template (ARM template) that deploys an Azure Cache for Redis. The cache can be used with an existing storage account to keep diagnostic data. You also learn how to define which resources are deployed and how to define parameters that are specified when the deployment is executed. You can use this template for your own deployments, or customize it to meet your requirements. Currently, diagnostic settings are shared for all caches in the same region for a subscription. Updating one cache in the region affects all other caches in the region.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

- **Azure subscription**: If you don't have an Azure subscription, create a free account before you begin.
- **A storage account**: To create one, see Create an Azure Storage account. The storage account is used for diagnostic data.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
```

```
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.6.1.6515",
      "templateHash": "6977754622100752630"
    }
  },
  "parameters": {
    "redisCacheName": {
      "type": "string",
      "defaultValue": "[format('redisCache-{0}',
uniqueString(resourceGroup().id))]",
      "metadata": {
        "description": "Specify the name of the Azure Redis Cache to
create."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location of all resources"
      }
    },
    "redisCacheSKU": {
      "type": "string",
      "defaultValue": "Standard",
      "allowedValues": [
        "Basic",
        "Standard",
        "Premium"
      ],
      "metadata": {
        "description": "Specify the pricing tier of the new Azure Redis
Cache."
      }
    },
    "redisCacheFamily": {
      "type": "string",
      "defaultValue": "C",
      "allowedValues": [
        "C",
        "P"
      ],
      "metadata": {
        "description": "Specify the family for the sku. C = Basic/Standard,
P = Premium."
      }
    },
    "redisCacheCapacity": {
      "type": "int",
      "defaultValue": 1,
      "allowedValues": [
        0,
        1,
```

```
        2,
        3,
        4,
        5,
        6
      ],
      "metadata": {
        "description": "Specify the size of the new Azure Redis Cache
instance. Valid values: for C (Basic/Standard) family (0, 1, 2, 3, 4, 5, 6),
for P (Premium) family (1, 2, 3, 4)"
      }
    },
    "enableNonSslPort": {
      "type": "bool",
      "defaultValue": false,
      "metadata": {
        "description": "Specify a boolean value that indicates whether to
allow access via non-SSL ports."
      }
    },
    "diagnosticsEnabled": {
      "type": "bool",
      "defaultValue": false,
      "metadata": {
        "description": "Specify a boolean value that indicates whether
diagnostics should be saved to the specified storage account."
      }
    },
    "existingDiagnosticsStorageAccountName": {
      "type": "string",
      "metadata": {
        "description": "Specify the name of an existing storage account for
diagnostics."
      }
    },
    "existingDiagnosticsStorageAccountResourceGroup": {
      "type": "string",
      "metadata": {
        "description": "Specify the resource group name of an existing
storage account for diagnostics."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Cache/redis",
      "apiVersion": "2020-06-01",
      "name": "[parameters('redisCacheName')]",
      "location": "[parameters('location')]",
      "properties": {
        "enableNonSslPort": "[parameters('enableNonSslPort')]",
        "minimumTlsVersion": "1.2",
        "sku": {
          "capacity": "[parameters('redisCacheCapacity')]",
          "family": "[parameters('redisCacheFamily')]",
```

```
          "name": "[parameters('redisCacheSKU')]"
        }
      }
    },
    {
      "type": "Microsoft.Insights/diagnosticSettings",
      "apiVersion": "2021-05-01-preview",
      "scope": "[format('Microsoft.Cache/redis/{0}',
parameters('redisCacheName'))]",
      "name": "[parameters('redisCacheName')]",
      "properties": {
        "storageAccountId": "
[extensionResourceId(format('/subscriptions/{0}/resourceGroups/{1}',
subscription().subscriptionId,
parameters('existingDiagnosticsStorageAccountResourceGroup')),
'Microsoft.Storage/storageAccounts',
parameters('existingDiagnosticsStorageAccountName'))]",
        "metrics": [
          {
            "timeGrain": "AllMetrics",
            "enabled": "[parameters('diagnosticsEnabled')]",
            "retentionPolicy": {
              "days": 90,
              "enabled": "[parameters('diagnosticsEnabled')]"
            }
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Cache/redis',
parameters('redisCacheName'))]"
      ]
    }
  ]
}
```

The following resources are defined in the template:

- Microsoft.Cache/Redis
- Microsoft.Insights/diagnosticsettings

Resource Manager templates for the new Premium tier are also available.

- Create a Premium Azure Cache for Redis with clustering
- Create Premium Azure Cache for Redis with data persistence
- Create Premium Redis Cache deployed into a Virtual Network

To check for the latest templates, see Azure Quickstart Templates and search for *Azure Cache for Redis*.

Deploy the template

1. Select the following image to sign in to Azure and open the template.

   Deploy to Azure

2. Select or enter the following values:

   - **Subscription**: select an Azure subscription used to create the data share and the other resources.
   - **Resource group**: select **Create new** to create a new resource group or select an existing resource group.
   - **Location**: select a location for the resource group. The storage account and the Redis cache must be in the same region. By default the Redis cache uses the same location as the resource group. So, specify the same location as the storage account.
   - **Redis Cache Name**: enter a name for the Redis cache.
   - **Existing Diagnostics Storage Account**: enter the resource ID of a storage account. The syntax is `/subscriptions/<SUBSCRIPTION ID>/resourceGroups/<RESOURCE GROUP NAME>/providers/Microsoft.Storage/storageAccounts/<STORAGE ACCOUNT NAME>`.

   Use the default value for the rest of the settings.

3. select **I agree to the terms and conditions stated above**, and the select **Purchase**.

Review deployed resources

1. Sign in to the Azure portal.
2. Open the Redis cache that you created.

Clean up resources

When no longer needed, delete the resource group, which deletes the resources in the resource group.

Azure PowerShell

```
$resourceGroupName = Read-Host -Prompt "Enter the resource group name"
Remove-AzResourceGroup -Name $resourceGroupName
Write-Host "Press [ENTER] to continue..."
```

Next steps

In this tutorial, you learnt how to create an Azure Resource Manager template that deploys an Azure Cache for Redis. To learn how to create an Azure Resource Manager template that deploys an Azure Web App with Azure Cache for Redis, see Create a Web App plus Azure Cache for Redis using a template.

Quickstart: Create an Azure Cosmos DB and a container by using an ARM template

Article • 03/08/2023

APPLIES TO: NoSQL

Azure Cosmos DB is Microsoft's fast NoSQL database with open APIs for any scale. You can use Azure Cosmos DB to quickly create and query key/value databases, document databases, and graph databases. Without a credit card or an Azure subscription, you can set up a free Try Azure Cosmos DB account. This quickstart focuses on the process of deploying an Azure Resource Manager template (ARM template) to create an Azure Cosmos DB database and a container within that database. You can later store data in this container.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

An Azure subscription or free Azure Cosmos DB trial account

- If you don't have an Azure subscription, create an Azure free account before you begin.
- You can try Azure Cosmos DB for free, without an Azure subscription, and with no commitment required. Alternatively, you can create an Azure Cosmos DB free tier account, with the first 1000 RU/s and 25 GB of storage for free. You can also use the Azure Cosmos DB emulator with a URI of `https://localhost:8081`. For the key to use with the emulator, see Authenticating requests.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.9.1.41621",
      "templateHash": "13701948159885019128"
    }
  },
  "parameters": {
    "accountName": {
      "type": "string",
      "defaultValue": "[format('sql-{0}',
uniqueString(resourceGroup().id))]",
      "metadata": {
        "description": "Azure Cosmos DB account name, max length 44
characters"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for the Azure Cosmos DB account."
      }
    },
    "primaryRegion": {
      "type": "string",
      "metadata": {
        "description": "The primary region for the Azure Cosmos DB account."
      }
    },
    "secondaryRegion": {
      "type": "string",
      "metadata": {
        "description": "The secondary region for the Azure Cosmos DB
account."
      }
    },
    "defaultConsistencyLevel": {
      "type": "string",
      "defaultValue": "Session",
      "metadata": {
        "description": "The default consistency level of the Cosmos DB
account."
```

```
      },
      "allowedValues": [
        "Eventual",
        "ConsistentPrefix",
        "Session",
        "BoundedStaleness",
        "Strong"
      ]
    },
    "maxStalenessPrefix": {
      "type": "int",
      "defaultValue": 100000,
      "metadata": {
        "description": "Max stale requests. Required for BoundedStaleness.
Valid ranges, Single Region: 10 to 2147483647. Multi Region: 100000 to
2147483647."
      },
      "maxValue": 2147483647,
      "minValue": 10
    },
    "maxIntervalInSeconds": {
      "type": "int",
      "defaultValue": 300,
      "metadata": {
        "description": "Max lag time (minutes). Required for
BoundedStaleness. Valid ranges, Single Region: 5 to 84600. Multi Region: 300
to 86400."
      },
      "maxValue": 86400,
      "minValue": 5
    },
    "systemManagedFailover": {
      "type": "bool",
      "defaultValue": true,
      "metadata": {
        "description": "Enable system managed failover for regions"
      },
      "allowedValues": [
        true,
        false
      ]
    },
    "databaseName": {
      "type": "string",
      "defaultValue": "myDatabase",
      "metadata": {
        "description": "The name for the database"
      }
    },
    "containerName": {
      "type": "string",
      "defaultValue": "myContainer",
      "metadata": {
        "description": "The name for the container"
      }
```

```
    },
    "throughput": {
      "type": "int",
      "defaultValue": 400,
      "metadata": {
        "description": "The throughput for the container"
      },
      "maxValue": 1000000,
      "minValue": 400
    }
  },
  "variables": {
    "consistencyPolicy": {
      "Eventual": {
        "defaultConsistencyLevel": "Eventual"
      },
      "ConsistentPrefix": {
        "defaultConsistencyLevel": "ConsistentPrefix"
      },
      "Session": {
        "defaultConsistencyLevel": "Session"
      },
      "BoundedStaleness": {
        "defaultConsistencyLevel": "BoundedStaleness",
        "maxStalenessPrefix": "[parameters('maxStalenessPrefix')]",
        "maxIntervalInSeconds": "[parameters('maxIntervalInSeconds')]"
      },
      "Strong": {
        "defaultConsistencyLevel": "Strong"
      }
    },
    "locations": [
      {
        "locationName": "[parameters('primaryRegion')]",
        "failoverPriority": 0,
        "isZoneRedundant": false
      },
      {
        "locationName": "[parameters('secondaryRegion')]",
        "failoverPriority": 1,
        "isZoneRedundant": false
      }
    ]
  },
  "resources": [
    {
      "type": "Microsoft.DocumentDB/databaseAccounts",
      "apiVersion": "2022-05-15",
      "name": "[toLower(parameters('accountName'))]",
      "location": "[parameters('location')]",
      "kind": "GlobalDocumentDB",
      "properties": {
        "consistencyPolicy": "[variables('consistencyPolicy')
[parameters('defaultConsistencyLevel')]]",
        "locations": "[variables('locations')]",
```

```
        "databaseAccountOfferType": "Standard",
        "enableAutomaticFailover": "[parameters('systemManagedFailover')]"
      }
    },
    {
      "type": "Microsoft.DocumentDB/databaseAccounts/sqlDatabases",
      "apiVersion": "2022-05-15",
      "name": "[format('{0}/{1}', toLower(parameters('accountName')),
parameters('databaseName'))]",
      "properties": {
        "resource": {
          "id": "[parameters('databaseName')]"
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.DocumentDB/databaseAccounts',
toLower(parameters('accountName')))]"
      ]
    },
    {
      "type":
"Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers",
      "apiVersion": "2022-05-15",
      "name": "[format('{0}/{1}', format('{0}/{1}',
toLower(parameters('accountName')), parameters('databaseName')),
parameters('containerName'))]",
      "properties": {
        "resource": {
          "id": "[parameters('containerName')]",
          "partitionKey": {
            "paths": [
              "/myPartitionKey"
            ],
            "kind": "Hash"
          },
          "indexingPolicy": {
            "indexingMode": "consistent",
            "includedPaths": [
              {
                "path": "/*"
              }
            ],
            "excludedPaths": [
              {
                "path": "/myPathToNotIndex/*"
              },
              {
                "path": "/_etag/?"
              }
            ],
            "compositeIndexes": [
              [
                {
                  "path": "/name",
                  "order": "ascending"
```

```
                },
                {
                  "path": "/age",
                  "order": "descending"
                }
              ]
            ],
            "spatialIndexes": [
              {
                "path": "/location/*",
                "types": [
                  "Point",
                  "Polygon",
                  "MultiPolygon",
                  "LineString"
                ]
              }
            ]
          },
          "defaultTtl": 86400,
          "uniqueKeyPolicy": {
            "uniqueKeys": [
              {
                "paths": [
                  "/phoneNumber"
                ]
              }
            ]
          }
        },
        "options": {
          "throughput": "[parameters('throughput')]"
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.DocumentDB/databaseAccounts/sqlDatabases',
split(format('{0}/{1}', toLower(parameters('accountName')),
parameters('databaseName')), '/')[0], split(format('{0}/{1}',
toLower(parameters('accountName')), parameters('databaseName')), '/')[1])]"
      ]
    }
  ]
}
```

Three Azure resources are defined in the template:

- Microsoft.DocumentDB/databaseAccounts: Create an Azure Cosmos DB account.
- Microsoft.DocumentDB/databaseAccounts/sqlDatabases: Create an Azure Cosmos DB database.

- Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers: Create an Azure Cosmos DB container.

More Azure Cosmos DB template samples can be found in the quickstart template gallery.

Deploy the template

1. Select the following image to sign in to Azure and open a template. The template creates an Azure Cosmos DB account, a database, and a container.

   Deploy to Azure

2. Select or enter the following values.

Unless it is specified, use the default values to create the Azure Cosmos DB resources.

- **Subscription**: select an Azure subscription.
- **Resource group**: select **Create new**, enter a unique name for the resource group, and then click **OK**.
- **Location**: select a location. For example, **Central US**.
- **Account Name**: enter a name for the Azure Cosmos DB account. It must be globally unique.
- **Location**: enter a location where you want to create your Azure Cosmos DB account. The Azure Cosmos DB account can be in the same location as the

resource group.

- **Primary Region**: The primary replica region for the Azure Cosmos DB account.
- **Secondary region**: The secondary replica region for the Azure Cosmos DB account.
- **Default Consistency Level**: The default consistency level for the Azure Cosmos DB account.
- **Max Staleness Prefix**: Max stale requests. Required for BoundedStaleness.
- **Max Interval in Seconds**: Max lag time. Required for BoundedStaleness.
- **Database Name**: The name of the Azure Cosmos DB database.
- **Container Name**: The name of the Azure Cosmos DB container.
- **Throughput**: The throughput for the container, minimum throughput value is 400 RU/s.
- **I agree to the terms and conditions state above**: Select.

3. Select **Purchase**. After the Azure Cosmos DB account has been deployed successfully, you get a notification:

The Azure portal is used to deploy the template. In addition to the Azure portal, you can also use the Azure PowerShell, Azure CLI, and REST API. To learn other deployment methods, see Deploy templates.

Validate the deployment

You can either use the Azure portal to check the Azure Cosmos DB account, the database, and the container or use the following Azure CLI or Azure PowerShell script to list the secret created.

CLI

Azure CLI

```
echo "Enter your Azure Cosmos DB account name:" &&
read cosmosAccountName &&
echo "Enter the resource group where the Azure Cosmos DB account
exists:" &&
read resourcegroupName &&
az cosmosdb show -g $resourcegroupName -n $cosmosAccountName
```

Clean up resources

If you plan to continue working with subsequent quickstarts and tutorials, you might want to leave these resources in place. When no longer needed, delete the resource group, which deletes the Azure Cosmos DB account and the related resources. To delete the resource group by using Azure CLI or Azure PowerShell:

CLI

Azure CLI

```
echo "Enter the Resource Group name:" &&
read resourceGroupName &&
az group delete --name $resourceGroupName &&
echo "Press [ENTER] to continue ..."
```

Next steps

In this quickstart, you created an Azure Cosmos DB account, a database and a container by using an ARM template and validated the deployment. To learn more about Azure Cosmos DB and Azure Resource Manager, continue on to the articles below.

- Read an Overview of Azure Cosmos DB
- Learn more about Azure Resource Manager
- Get other Azure Cosmos DB Resource Manager templates
- Trying to do capacity planning for a migration to Azure Cosmos DB? You can use information about your existing database cluster for capacity planning.
  - If all you know is the number of vcores and servers in your existing database cluster, read about estimating request units using vCores or vCPUs
  - If you know typical request rates for your current database workload, read about estimating request units using Azure Cosmos DB capacity planner

Quickstart: Use an ARM template to create an Azure Database for MariaDB server

Article • 03/10/2023

Azure Database for MariaDB is a managed service that you use to run, manage, and scale highly available MariaDB databases in the cloud. In this quickstart, you use an Azure Resource Manager template (ARM template) to create an Azure Database for MariaDB server in the Azure portal, PowerShell, or Azure CLI.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

Portal

An Azure account with an active subscription. Create one for free.

Review the template

You create an Azure Database for MariaDB server with a defined set of compute and storage resources. To learn more, see Azure Database for MariaDB pricing tiers. You create the server within an Azure resource group.

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.15.31.15270",
      "templateHash": "13920051172609281830"
    }
  },
  "parameters": {
    "serverName": {
      "type": "string",
      "metadata": {
        "description": "Server Name for Azure database for MariaDB"
      }
    },
    "administratorLogin": {
      "type": "string",
      "minLength": 1,
      "metadata": {
        "description": "Database administrator login name"
      }
    },
    "administratorLoginPassword": {
      "type": "securestring",
      "minLength": 8,
      "metadata": {
        "description": "Database administrator password"
      }
    },
    "skuCapacity": {
      "type": "int",
      "defaultValue": 2,
      "metadata": {
        "description": "Azure database for MariaDB compute capacity in
vCores (2,4,8,16,32)"
      }
    },
    "skuName": {
      "type": "string",
      "defaultValue": "GP_Gen5_2",
      "metadata": {
        "description": "Azure database for MariaDB sku name "
      }
    },
    "skuSizeMB": {
      "type": "int",
      "defaultValue": 51200,
      "metadata": {
        "description": "Azure database for MariaDB Sku Size "
      }
    },
```

```
    "skuTier": {
      "type": "string",
      "defaultValue": "GeneralPurpose",
      "metadata": {
        "description": "Azure database for MariaDB pricing tier"
      }
    },
    "skuFamily": {
      "type": "string",
      "defaultValue": "Gen5",
      "metadata": {
        "description": "Azure database for MariaDB sku family"
      }
    },
    "mariadbVersion": {
      "type": "string",
      "defaultValue": "10.3",
      "allowedValues": [
        "10.2",
        "10.3"
      ],
      "metadata": {
        "description": "MariaDB version"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "backupRetentionDays": {
      "type": "int",
      "defaultValue": 7,
      "metadata": {
        "description": "MariaDB Server backup retention days"
      }
    },
    "geoRedundantBackup": {
      "type": "string",
      "defaultValue": "Disabled",
      "metadata": {
        "description": "Geo-Redundant Backup setting"
      }
    },
    "virtualNetworkName": {
      "type": "string",
      "defaultValue": "azure_mariadb_vnet",
      "metadata": {
        "description": "Virtual Network Name"
      }
    },
    "subnetName": {
      "type": "string",
```

```
      "defaultValue": "azure_mariadb_subnet",
      "metadata": {
        "description": "Subnet Name"
      }
    },
    "virtualNetworkRuleName": {
      "type": "string",
      "defaultValue": "AllowSubnet",
      "metadata": {
        "description": "Virtual Network RuleName"
      }
    },
    "vnetAddressPrefix": {
      "type": "string",
      "defaultValue": "10.0.0.0/16",
      "metadata": {
        "description": "Virtual Network Address Prefix"
      }
    },
    "subnetPrefix": {
      "type": "string",
      "defaultValue": "10.0.0.0/16",
      "metadata": {
        "description": "Subnet Address Prefix"
      }
    }
  },
  "variables": {
    "firewallrules": [
      {
        "Name": "rule1",
        "StartIpAddress": "0.0.0.0",
        "EndIpAddress": "255.255.255.255"
      },
      {
        "Name": "rule2",
        "StartIpAddress": "0.0.0.0",
        "EndIpAddress": "255.255.255.255"
      }
    ]
  },
  "resources": [
    {
      "type": "Microsoft.DBforMariaDB/servers/virtualNetworkRules",
      "apiVersion": "2018-06-01",
      "name": "[format('{0}/{1}', parameters('serverName'),
parameters('virtualNetworkRuleName'))]",
      "properties": {
        "virtualNetworkSubnetId": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('virtualNetworkName'), parameters('subnetName'))]",
        "ignoreMissingVnetServiceEndpoint": true
      },
      "dependsOn": [
        "[resourceId('Microsoft.DBforMariaDB/servers',
```

```
parameters('serverName'))]",
        "[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('virtualNetworkName'), parameters('subnetName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "2021-05-01",
      "name": "[parameters('virtualNetworkName')]",
      "location": "[parameters('location')]",
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[parameters('vnetAddressPrefix')]"
          ]
        }
      }
    },
    {
      "type": "Microsoft.Network/virtualNetworks/subnets",
      "apiVersion": "2021-05-01",
      "name": "[format('{0}/{1}', parameters('virtualNetworkName'),
parameters('subnetName'))]",
      "properties": {
        "addressPrefix": "[parameters('subnetPrefix')]"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/virtualNetworks',
parameters('virtualNetworkName'))]"
      ]
    },
    {
      "type": "Microsoft.DBforMariaDB/servers",
      "apiVersion": "2018-06-01",
      "name": "[parameters('serverName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('skuName')]",
        "tier": "[parameters('skuTier')]",
        "capacity": "[parameters('skuCapacity')]",
        "size": "[format('{0}', parameters('skuSizeMB'))]",
        "family": "[parameters('skuFamily')]"
      },
      "properties": {
        "createMode": "Default",
        "version": "[parameters('mariadbVersion')]",
        "administratorLogin": "[parameters('administratorLogin')]",
        "administratorLoginPassword": "
[parameters('administratorLoginPassword')]",
        "storageProfile": {
          "storageMB": "[parameters('skuSizeMB')]",
          "backupRetentionDays": "[parameters('backupRetentionDays')]",
          "geoRedundantBackup": "[parameters('geoRedundantBackup')]"
        }
      }
```

```
    },
    {
      "copy": {
        "name": "firewallRules",
        "count": "[length(variables('firewallrules'))]",
        "mode": "serial",
        "batchSize": 1
      },
      "type": "Microsoft.DBforMariaDB/servers/firewallRules",
      "apiVersion": "2018-06-01",
      "name": "[format('{0}/{1}', parameters('serverName'),
variables('firewallrules')[copyIndex()].Name)]",
      "properties": {
        "startIpAddress": "[variables('firewallrules')
[copyIndex()].StartIpAddress]",
        "endIpAddress": "[variables('firewallrules')
[copyIndex()].EndIpAddress]"
      },
      "dependsOn": [
        "[resourceId('Microsoft.DBforMariaDB/servers',
parameters('serverName'))]"
      ]
    }
  ]
}
```

The template defines five Azure resources:

- Microsoft.Network/virtualNetworks
- Microsoft.Network/virtualNetworks/subnets
- Microsoft.DBforMariaDB/servers
- Microsoft.DBforMariaDB/servers/virtualNetworkRules
- Microsoft.DBforMariaDB/servers/firewallRules

More Azure Database for MariaDB template samples can be found in Azure Quickstart Templates.

Deploy the template

Portal

Select the following link to deploy the Azure Database for MariaDB server template in the Azure portal:

Deploy to Azure

On the **Deploy Azure Database for MariaDB with VNet** page:

1. For **Resource group**, select **Create new**, enter a name for the new resource group, and select **OK**.

2. If you created a new resource group, select a **Location** for the resource group and the new server.

3. Enter a **Server Name**, **Administrator Login**, and **Administrator Login Password**.

4. Change the other default settings if you want:

   - **Subscription**: the Azure subscription you want to use for the server.
   - **Sku Capacity**: the vCore capacity, which can be *2* (the default), *4*, *8*, *16*, *32*, or *64*.

- **Sku Name**: the SKU tier prefix, SKU family, and SKU capacity, joined by underscores, such as *B_Gen5_1*, *GP_Gen5_2* (the default), or *MO_Gen5_32*.
- **Sku Size MB**: the storage size, in megabytes, of the Azure Database for MariaDB server (default *51200*).
- **Sku Tier**: the deployment tier, such as *Basic*, *GeneralPurpose* (the default), or *MemoryOptimized*.
- **Sku Family**: *Gen4* or *Gen5* (the default), which indicates hardware generation for server deployment.
- **Mariadb Version**: the version of MariaDB server to deploy, such as *10.2*, or *10.3* (the default).
- **Backup Retention Days**: the desired period for geo-redundant backup retention, in days (default *7*).
- **Geo Redundant Backup**: *Enabled* or *Disabled* (the default), depending on geo-disaster recovery (Geo-DR) requirements.
- **Virtual Network Name**: the name of the virtual network (default *azure_mariadb_vnet*).
- **Subnet Name**: the name of the subnet (default *azure_mariadb_subnet*).
- **Virtual Network Rule Name**: the name of the virtual network rule allowing the subnet (default *AllowSubnet*).
- **Vnet Address Prefix**: the address prefix for the virtual network (default *10.0.0.0/16*).
- **Subnet Prefix**: the address prefix for the subnet (default *10.0.0.0/16*).

5. Read the terms and conditions, and then select **I agree to the terms and conditions stated above**.

6. Select **Purchase**.

Review deployed resources

Portal

Follow these steps to see an overview of your new Azure Database for MariaDB server:

1. In the Azure portal, search for and select **Azure Database for MariaDB servers**.

2. In the database list, select your new server. The **Overview** page for your new Azure Database for MariaDB server appears.

Clean up resources

When it's no longer needed, delete the resource group, which deletes the resources in the resource group.

Portal

1. In the Azure portal, search for and select **Resource groups**.
2. In the resource group list, choose the name of your resource group.
3. In the **Overview** page of your resource group, select **Delete resource group**.
4. In the confirmation dialog box, type the name of your resource group, and then select **Delete**.

Next steps

For a step-by-step tutorial that guides you through the process of creating an ARM template, see:

Tutorial: Create and deploy your first ARM template

Quickstart: Use an ARM template to create an Azure Database for MySQL server

Article • 03/10/2023

APPLIES TO: Azure Database for MySQL - Single Server

> **Important**
>
> Azure Database for MySQL - Single Server is on the retirement path. We strongly recommend for you to upgrade to Azure Database for MySQL - Flexible Server. For more information about migrating to Azure Database for MySQL - Flexible Server, see **What's happening to Azure Database for MySQL Single Server**?

Azure Database for MySQL is a managed service that you use to run, manage, and scale highly available MySQL databases in the cloud. In this quickstart, you use an Azure Resource Manager template (ARM template) to create an Azure Database for MySQL server with virtual network integration. You can create the server in the Azure portal, Azure CLI, or Azure PowerShell.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

Portal

An Azure account with an active subscription. Create one for free.

Review the template

You create an Azure Database for MySQL server with a defined set of compute and storage resources. To learn more, see Azure Database for MySQL pricing tiers. You create the server within an Azure resource group.

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "7949891959208359208"
    }
  },
  "parameters": {
    "serverName": {
      "type": "string",
      "metadata": {
        "description": "Server Name for Azure database for MySQL"
      }
    },
    "administratorLogin": {
      "type": "string",
      "minLength": 1,
      "metadata": {
        "description": "Database administrator login name"
      }
    },
    "administratorLoginPassword": {
      "type": "secureString",
      "minLength": 8,
      "metadata": {
        "description": "Database administrator password"
      }
    },
    "skuCapacity": {
      "type": "int",
      "defaultValue": 2,
      "metadata": {
        "description": "Azure database for MySQL compute capacity in vCores (2,4,8,16,32)"
      }
    },
    "skuName": {
      "type": "string",
```

```
        "defaultValue": "GP_Gen5_2",
        "metadata": {
          "description": "Azure database for MySQL sku name "
        }
      },
      "SkuSizeMB": {
        "type": "int",
        "defaultValue": 5120,
        "metadata": {
          "description": "Azure database for MySQL Sku Size "
        }
      },
      "SkuTier": {
        "type": "string",
        "defaultValue": "GeneralPurpose",
        "allowedValues": [
          "Basic",
          "GeneralPurpose",
          "MemoryOptimized"
        ],
        "metadata": {
          "description": "Azure database for MySQL pricing tier"
        }
      },
      "skuFamily": {
        "type": "string",
        "defaultValue": "Gen5",
        "metadata": {
          "description": "Azure database for MySQL sku family"
        }
      },
      "mysqlVersion": {
        "type": "string",
        "defaultValue": "8.0",
        "allowedValues": [
          "5.6",
          "5.7",
          "8.0"
        ],
        "metadata": {
          "description": "MySQL version"
        }
      },
      "location": {
        "type": "string",
        "defaultValue": "[resourceGroup().location]",
        "metadata": {
          "description": "Location for all resources."
        }
      },
      "backupRetentionDays": {
        "type": "int",
        "defaultValue": 7,
        "metadata": {
          "description": "MySQL Server backup retention days"
```

```
      }
    },
    "geoRedundantBackup": {
      "type": "string",
      "defaultValue": "Disabled",
      "metadata": {
        "description": "Geo-Redundant Backup setting"
      }
    },
    "virtualNetworkName": {
      "type": "string",
      "defaultValue": "azure_mysql_vnet",
      "metadata": {
        "description": "Virtual Network Name"
      }
    },
    "subnetName": {
      "type": "string",
      "defaultValue": "azure_mysql_subnet",
      "metadata": {
        "description": "Subnet Name"
      }
    },
    "virtualNetworkRuleName": {
      "type": "string",
      "defaultValue": "AllowSubnet",
      "metadata": {
        "description": "Virtual Network RuleName"
      }
    },
    "vnetAddressPrefix": {
      "type": "string",
      "defaultValue": "10.0.0.0/16",
      "metadata": {
        "description": "Virtual Network Address Prefix"
      }
    },
    "subnetPrefix": {
      "type": "string",
      "defaultValue": "10.0.0.0/16",
      "metadata": {
        "description": "Subnet Address Prefix"
      }
    }
  },
  "variables": {
    "firewallrules": [
      {
        "Name": "rule1",
        "StartIpAddress": "0.0.0.0",
        "EndIpAddress": "255.255.255.255"
      },
      {
        "Name": "rule2",
        "StartIpAddress": "0.0.0.0",
```

```
        "EndIpAddress": "255.255.255.255"
      }
    ]
  },
  "resources": [
    {
      "type": "Microsoft.DBforMySQL/servers/virtualNetworkRules",
      "apiVersion": "2017-12-01",
      "name": "[format('{0}/{1}', parameters('serverName'),
parameters('virtualNetworkRuleName'))]",
      "properties": {
        "virtualNetworkSubnetId": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('virtualNetworkName'), parameters('subnetName'))]",
        "ignoreMissingVnetServiceEndpoint": true
      },
      "dependsOn": [
        "[resourceId('Microsoft.DBforMySQL/servers',
parameters('serverName'))]",
        "[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('virtualNetworkName'), parameters('subnetName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "2021-05-01",
      "name": "[parameters('virtualNetworkName')]",
      "location": "[parameters('location')]",
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[parameters('vnetAddressPrefix')]"
          ]
        }
      }
    },
    {
      "type": "Microsoft.Network/virtualNetworks/subnets",
      "apiVersion": "2021-05-01",
      "name": "[format('{0}/{1}', parameters('virtualNetworkName'),
parameters('subnetName'))]",
      "properties": {
        "addressPrefix": "[parameters('subnetPrefix')]"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/virtualNetworks',
parameters('virtualNetworkName'))]"
      ]
    },
    {
      "type": "Microsoft.DBforMySQL/servers",
      "apiVersion": "2017-12-01",
      "name": "[parameters('serverName')]",
      "location": "[parameters('location')]",
      "sku": {
```

```
        "name": "[parameters('skuName')]",
        "tier": "[parameters('SkuTier')]",
        "capacity": "[parameters('skuCapacity')]",
        "size": "[format('{0}', parameters('SkuSizeMB'))]",
        "family": "[parameters('skuFamily')]"
      },
      "properties": {
        "createMode": "Default",
        "version": "[parameters('mysqlVersion')]",
        "administratorLogin": "[parameters('administratorLogin')]",
        "administratorLoginPassword": "
[parameters('administratorLoginPassword')]",
        "storageProfile": {
          "storageMB": "[parameters('SkuSizeMB')]",
          "backupRetentionDays": "[parameters('backupRetentionDays')]",
          "geoRedundantBackup": "[parameters('geoRedundantBackup')]"
        }
      }
    },
    {
      "copy": {
        "name": "firewallRules",
        "count": "[length(variables('firewallrules'))]",
        "mode": "serial",
        "batchSize": 1
      },
      "type": "Microsoft.DBforMySQL/servers/firewallRules",
      "apiVersion": "2017-12-01",
      "name": "[format('{0}/{1}', parameters('serverName'),
variables('firewallrules')[copyIndex()].Name)]",
      "properties": {
        "startIpAddress": "[variables('firewallrules')
[copyIndex()].StartIpAddress]",
        "endIpAddress": "[variables('firewallrules')
[copyIndex()].EndIpAddress]"
      },
      "dependsOn": [
        "[resourceId('Microsoft.DBforMySQL/servers',
parameters('serverName'))]"
      ]
    }
  ]
}
```

The template defines five Azure resources:

- Microsoft.Network/virtualNetworks
- Microsoft.Network/virtualNetworks/subnets
- Microsoft.DBforMySQL/servers
- Microsoft.DBforMySQL/servers/virtualNetworkRules
- Microsoft.DBforMySQL/servers/firewallRules

More Azure Database for MySQL template samples can be found in the quickstart template gallery.

Deploy the template

Portal

Select the following link to deploy the Azure Database for MySQL server template in the Azure portal:

Deploy to Azure

On the **Deploy Azure Database for MySQL with VNet** page:

1. For **Resource group**, select **Create new**, enter a name for the new resource group, and select **OK**.
2. If you created a new resource group, select a **Location** for the resource group and the new server.
3. Enter a **Server Name**, **Administrator Login**, and **Administrator Login Password**.

4. Change the other default settings if you want:

- **Subscription**: the Azure subscription you want to use for the server.
- **Sku Capacity**: the vCore capacity, which can be *2* (the default), *4*, *8*, *16*, *32*, or *64*.
- **Sku Name**: the SKU tier prefix, SKU family, and SKU capacity, joined by underscores, such as *B_Gen5_1*, *GP_Gen5_2* (the default), or *MO_Gen5_32*.
- **Sku Size MB**: the storage size, in megabytes, of the Azure Database for MySQL server (default *5120*).
- **Sku Tier**: the deployment tier, such as *Basic*, *GeneralPurpose* (the default), or *MemoryOptimized*.
- **Sku Family**: *Gen4* or *Gen5* (the default), which indicates hardware generation for server deployment.

- **Mysql Version**: the version of MySQL server to deploy, such as *5.6* or *5.7* (the default).
- **Backup Retention Days**: the desired period for geo-redundant backup retention, in days (default *7*).
- **Geo Redundant Backup**: *Enabled* or *Disabled* (the default), depending on geo-disaster recovery (Geo-DR) requirements.
- **Virtual Network Name**: the name of the virtual network (default *azure_mysql_vnet*).
- **Subnet Name**: the name of the subnet (default *azure_mysql_subnet*).
- **Virtual Network Rule Name**: the name of the virtual network rule allowing the subnet (default *AllowSubnet*).
- **Vnet Address Prefix**: the address prefix for the virtual network (default *10.0.0.0/16*).
- **Subnet Prefix**: the address prefix for the subnet (default *10.0.0.0/16*).

5. Read the terms and conditions, and then select **I agree to the terms and conditions stated above**.

6. Select **Purchase**.

Review deployed resources

Portal

Follow these steps to see an overview of your new Azure Database for MySQL server:

1. In the Azure portal, search for and select **Azure Database for MySQL servers**.

2. In the database list, select your new server. The **Overview** page for your new Azure Database for MySQL server appears.

Exporting ARM template from the portal

You can export an ARM template from the Azure portal. There are two ways to export a template:

- Export from resource group or resource. This option generates a new template from existing resources. The exported template is a "snapshot" of the current state of the resource group. You can export an entire resource group or specific resources within that resource group.
- Export before deployment or from history. This option retrieves an exact copy of a template used for deployment.

When exporting the template, in the `"properties":{ }` section of the MySQL server resource you will notice that `administratorLogin` and `administratorLoginPassword` will not be included for security reasons. You **MUST** add these parameters to your template before deploying the template or the template will fail.

```
"resources": [
    {
      "type": "Microsoft.DBforMySQL/servers",
      "apiVersion": "2017-12-01",
      "name": "[parameters('servers_name')]",
      "location": "southcentralus",
      "sku": {
                "name": "B_Gen5_1",
                "tier": "Basic",
                "family": "Gen5",
                "capacity": 1
            },
      "properties": {
        "administratorLogin": "[parameters('administratorLogin')]",
        "administratorLoginPassword": "
[parameters('administratorLoginPassword')]",
```

Clean up resources

When it's no longer needed, delete the resource group, which deletes the resources in the resource group.

Portal

1. In the Azure portal, search for and select **Resource groups**.
2. In the resource group list, choose the name of your resource group.
3. In the **Overview** page of your resource group, select **Delete resource group**.

4. In the confirmation dialog box, type the name of your resource group, and then select **Delete**.

Next steps

For a step-by-step tutorial that guides you through the process of creating an ARM template, see:

Tutorial: Create and deploy your first ARM template

Quickstart: Use an ARM template to create an Azure Database for MySQL - Flexible Server

Article • 03/10/2023

APPLIES TO: Azure Database for MySQL - Flexible Server

Azure Database for MySQL - Flexible Server is a managed service that you use to run, manage, and scale highly available MySQL databases in the cloud. You can use an Azure Resource Manager template (ARM template) to provision a flexible server to deploy multiple servers or multiple databases on a server.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

Prerequisites

- An Azure account with an active subscription.

If you don't have an Azure subscription, create an Azure free account before you begin. With an Azure free account, you can now try Azure Database for MySQL - Flexible Server for free for 12 months. For more information, see Try Flexible Server for free.

Create server with public access

Create an **azuredeploy.json** file with the following content to create a server using public access connectivity method and also create a database on the server. Update the **firewallRules** default value if needed.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "resourceNamePrefix": {
      "type": "string",
      "metadata": {
```

```
        "description": "Provide a prefix for creating resource names."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    },
    "administratorLogin": {
      "type": "string"
    },
    "administratorLoginPassword": {
      "type": "securestring"
    },
    "firewallRules": {
      "type": "array",
      "defaultValue": [
        {
          "name": "rule1",
          "startIPAddress": "192.168.0.1",
          "endIPAddress": "192.168.0.255"
        },
        {
          "name": "rule2",
          "startIPAddress": "192.168.1.1",
          "endIPAddress": "192.168.1.255"
        }
      ]
    },
    "serverEdition": {
      "type": "string",
      "defaultValue": "Burstable",
      "allowedValues": [
        "Burstable",
        "Generalpurpose",
        "MemoryOptimized"
      ],
      "metadata": {
        "description": "The tier of the particular SKU. High Availability is
available only for GeneralPurpose and MemoryOptimized sku."
      }
    },
    "version": {
      "type": "string",
      "defaultValue": "8.0.21",
      "allowedValues": [
        "5.7",
        "8.0.21"
      ],
      "metadata": {
        "description": "Server version"
      }
    },
    "availabilityZone": {
      "type": "string",
      "defaultValue": "1",
```

```
      "metadata": {
        "description": "Availability Zone information of the server. (Leave
blank for No Preference)."
      }
    },
    "haEnabled": {
      "type": "string",
      "defaultValue": "Disabled",
      "allowedValues": [
        "Disabled",
        "SameZone",
        "ZoneRedundant"
      ],
      "metadata": {
        "description": "High availability mode for a server : Disabled,
SameZone, or ZoneRedundant"
      }
    },
    "standbyAvailabilityZone": {
      "type": "string",
      "defaultValue": "2",
      "metadata": {
        "description": "Availability zone of the standby server."
      }
    },
    "storageSizeGB": {
      "type": "int",
      "defaultValue": 20
    },
    "storageIops": {
      "type": "int",
      "defaultValue": 360
    },
    "storageAutogrow": {
      "type": "string",
      "defaultValue": "Enabled",
      "allowedValues": [
        "Enabled",
        "Disabled"
      ]
    },
    "skuName": {
      "type": "string",
      "defaultValue": "Standard_B1ms",
      "metadata": {
        "description": "The name of the sku, e.g. Standard_D32ds_v4."
      }
    },
    "backupRetentionDays": {
      "type": "int",
      "defaultValue": 7
    },
    "geoRedundantBackup": {
      "type": "string",
      "defaultValue": "Disabled",
```

```
      "allowedValues": [
        "Disabled",
        "Enabled"
      ]
    },
    "serverName": {
      "type": "string",
      "defaultValue": "[format('{0}mysqlserver',
parameters('resourceNamePrefix'))]"
    },
    "databaseName": {
      "type": "string",
      "defaultValue": "[format('{0}mysqldb',
parameters('resourceNamePrefix'))]"
    }
  },
  "resources": [
    {
      "type": "Microsoft.DBforMySQL/flexibleServers",
      "apiVersion": "2021-12-01-preview",
      "name": "[parameters('serverName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('skuName')]",
        "tier": "[parameters('serverEdition')]"
      },
      "properties": {
        "version": "[parameters('version')]",
        "administratorLogin": "[parameters('administratorLogin')]",
        "administratorLoginPassword": "
[parameters('administratorLoginPassword')]",
        "availabilityZone": "[parameters('availabilityZone')]",
        "highAvailability": {
          "mode": "[parameters('haEnabled')]",
          "standbyAvailabilityZone": "
[parameters('standbyAvailabilityZone')]"
        },
        "storage": {
          "storageSizeGB": "[parameters('storageSizeGB')]",
          "iops": "[parameters('storageIops')]",
          "autoGrow": "[parameters('storageAutogrow')]"
        },
        "backup": {
          "backupRetentionDays": "[parameters('backupRetentionDays')]",
          "geoRedundantBackup": "[parameters('geoRedundantBackup')]"
        }
      }
    },
    {
      "type": "Microsoft.DBforMySQL/flexibleServers/databases",
      "apiVersion": "2021-12-01-preview",
      "name": "[format('{0}/{1}', parameters('serverName'),
parameters('databaseName'))]",
      "properties": {
        "charset": "utf8",
```

```
        "collation": "utf8_general_ci"
      },
      "dependsOn": [
        "[resourceId('Microsoft.DBforMySQL/flexibleServers',
parameters('serverName'))]"
      ]
    },
    {
      "copy": {
        "name": "createFirewallRules",
        "count": "[length(range(0,
if(greater(length(parameters('firewallRules')), 0),
length(parameters('firewallRules')), 1)))]",
        "mode": "serial",
        "batchSize": 1
      },
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2020-10-01",
      "name": "[format('firewallRules-{0}', range(0,
if(greater(length(parameters('firewallRules')), 0),
length(parameters('firewallRules')), 1))[copyIndex()])]",
      "properties": {
        "expressionEvaluationOptions": {
          "scope": "inner"
        },
        "mode": "Incremental",
        "parameters": {
          "ip": {
            "value": "[parameters('firewallRules')[range(0,
if(greater(length(parameters('firewallRules')), 0),
length(parameters('firewallRules')), 1))[copyIndex()]]]"
          },
          "serverName": {
            "value": "[parameters('serverName')]"
          }
        },
        "template": {
          "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
          "contentVersion": "1.0.0.0",
          "parameters": {
            "serverName": {
              "type": "string"
            },
            "ip": {
              "type": "object"
            }
          },
          "resources": [
            {
              "type": "Microsoft.DBforMySQL/flexibleServers/firewallRules",
              "apiVersion": "2021-12-01-preview",
              "name": "[format('{0}/{1}', parameters('serverName'),
parameters('ip').name)]",
              "properties": {
```

```
                "startIpAddress": "[parameters('ip').startIPAddress]",
                "endIpAddress": "[parameters('ip').endIPAddress]"
              }
            }
          ]
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.DBforMySQL/flexibleServers',
parameters('serverName'))]"
      ]
    }
  ]
}
```

Create a server with private access

Create an **azuredeploy.json** file with the following content to create a server using private access connectivity method inside a virtual network.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "resourceNamePrefix": {
      "type": "string",
      "metadata": {
        "description": "Provide a prefix for creating resource names."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    },
    "administratorLogin": {
      "type": "string"
    },
    "administratorLoginPassword": {
      "type": "securestring"
    },
    "firewallRules": {
      "type": "array",
      "defaultValue": [
        {
          "name": "rule1",
          "startIPAddress": "192.168.0.1",
          "endIPAddress": "192.168.0.255"
        },
```

```
        {
          "name": "rule2",
          "startIPAddress": "192.168.1.1",
          "endIPAddress": "192.168.1.255"
        }
      ]
    },
    "serverEdition": {
      "type": "string",
      "defaultValue": "Burstable",
      "allowedValues": [
        "Burstable",
        "Generalpurpose",
        "MemoryOptimized"
      ],
      "metadata": {
        "description": "The tier of the particular SKU. High Availability is
available only for GeneralPurpose and MemoryOptimized sku."
      }
    },
    "version": {
      "type": "string",
      "defaultValue": "8.0.21",
      "allowedValues": [
        "5.7",
        "8.0.21"
      ],
      "metadata": {
        "description": "Server version"
      }
    },
    "availabilityZone": {
      "type": "string",
      "defaultValue": "1",
      "metadata": {
        "description": "Availability Zone information of the server. (Leave
blank for No Preference)."
      }
    },
    "haEnabled": {
      "type": "string",
      "defaultValue": "Disabled",
      "allowedValues": [
        "Disabled",
        "SameZone",
        "ZoneRedundant"
      ],
      "metadata": {
        "description": "High availability mode for a server : Disabled,
SameZone, or ZoneRedundant"
      }
    },
    "standbyAvailabilityZone": {
      "type": "string",
      "defaultValue": "2",
```

```
      "metadata": {
        "description": "Availability zone of the standby server."
      }
    },
    "storageSizeGB": {
      "type": "int",
      "defaultValue": 20
    },
    "storageIops": {
      "type": "int",
      "defaultValue": 360
    },
    "storageAutogrow": {
      "type": "string",
      "defaultValue": "Enabled",
      "allowedValues": [
        "Enabled",
        "Disabled"
      ]
    },
    "skuName": {
      "type": "string",
      "defaultValue": "Standard_B1ms",
      "metadata": {
        "description": "The name of the sku, e.g. Standard_D32ds_v4."
      }
    },
    "backupRetentionDays": {
      "type": "int",
      "defaultValue": 7
    },
    "geoRedundantBackup": {
      "type": "string",
      "defaultValue": "Disabled",
      "allowedValues": [
        "Disabled",
        "Enabled"
      ]
    },
    "serverName": {
      "type": "string",
      "defaultValue": "[format('{0}mysqlserver',
parameters('resourceNamePrefix'))]"
    },
    "databaseName": {
      "type": "string",
      "defaultValue": "[format('{0}mysqldb',
parameters('resourceNamePrefix'))]"
    }
  },
  "resources": [
    {
      "type": "Microsoft.DBforMySQL/flexibleServers",
      "apiVersion": "2021-12-01-preview",
      "name": "[parameters('serverName')]",
```

```
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('skuName')]",
        "tier": "[parameters('serverEdition')]"
      },
      "properties": {
        "version": "[parameters('version')]",
        "administratorLogin": "[parameters('administratorLogin')]",
        "administratorLoginPassword": "
[parameters('administratorLoginPassword')]",
        "availabilityZone": "[parameters('availabilityZone')]",
        "highAvailability": {
          "mode": "[parameters('haEnabled')]",
          "standbyAvailabilityZone": "
[parameters('standbyAvailabilityZone')]"
        },
        "storage": {
          "storageSizeGB": "[parameters('storageSizeGB')]",
          "iops": "[parameters('storageIops')]",
          "autoGrow": "[parameters('storageAutogrow')]"
        },
        "backup": {
          "backupRetentionDays": "[parameters('backupRetentionDays')]",
          "geoRedundantBackup": "[parameters('geoRedundantBackup')]"
        }
      }
    },
    {
      "type": "Microsoft.DBforMySQL/flexibleServers/databases",
      "apiVersion": "2021-12-01-preview",
      "name": "[format('{0}/{1}', parameters('serverName'),
parameters('databaseName'))]",
      "properties": {
        "charset": "utf8",
        "collation": "utf8_general_ci"
      },
      "dependsOn": [
        "[resourceId('Microsoft.DBforMySQL/flexibleServers',
parameters('serverName'))]"
      ]
    },
    {
      "copy": {
        "name": "createFirewallRules",
        "count": "[length(range(0,
if(greater(length(parameters('firewallRules')), 0),
length(parameters('firewallRules')), 1)))]",
        "mode": "serial",
        "batchSize": 1
      },
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2020-10-01",
      "name": "[format('firewallRules-{0}', range(0,
if(greater(length(parameters('firewallRules')), 0),
length(parameters('firewallRules')), 1))[copyIndex()])]",
```

```
      "properties": {
        "expressionEvaluationOptions": {
          "scope": "inner"
        },
        "mode": "Incremental",
        "parameters": {
          "ip": {
            "value": "[parameters('firewallRules')[range(0,
if(greater(length(parameters('firewallRules')), 0),
length(parameters('firewallRules')), 1))[copyIndex()]]]"
          },
          "serverName": {
            "value": "[parameters('serverName')]"
          }
        },
        "template": {
          "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
          "contentVersion": "1.0.0.0",
          "parameters": {
            "serverName": {
              "type": "string"
            },
            "ip": {
              "type": "object"
            }
          },
          "resources": [
            {
              "type": "Microsoft.DBforMySQL/flexibleServers/firewallRules",
              "apiVersion": "2021-12-01-preview",
              "name": "[format('{0}/{1}', parameters('serverName'),
parameters('ip').name)]",
              "properties": {
                "startIpAddress": "[parameters('ip').startIPAddress]",
                "endIpAddress": "[parameters('ip').endIPAddress]"
              }
            }
          ]
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.DBforMySQL/flexibleServers',
parameters('serverName'))]"
      ]
    }
  ]
}
```

Deploy the template

Deploy the Bicep file using either Azure CLI or Azure PowerShell.

CLI

Azure CLI

```
az group create --name exampleRG --location eastus
az deployment group create --resource-group exampleRG --template-file
azuredeploy.json
```

Follow the instructions to enter the parameter values. When the deployment finishes, you should see a message indicating the deployment succeeded.

Review deployed resources

Follow these steps to verify if your server was created in the resource group.

CLI

Azure CLI

```
az resource list --resource-group exampleRG
```

Clean up resources

To delete the resource group and the resources contained in the resource group:

CLI

Azure CLI

```
az group delete --name exampleRG
```

Next steps

For a step-by-step tutorial that guides you through the process of creating an ARM template, see:

Tutorial: Create and deploy your first ARM template

For a step-by-step tutorial to build an app with App Service using MySQL, see:

Build a PHP (Laravel) web app with MySQL

Quickstart: Use an ARM template to create an Azure Database for PostgreSQL - single server

Article • 03/29/2023

APPLIES TO: Azure Database for PostgreSQL - Single Server

> **Important**
>
> Azure Database for PostgreSQL - Single Server is on the retirement path. We strongly recommend for you to upgrade to Azure Database for PostgreSQL - Flexible Server. For more information about migrating to Azure Database for PostgreSQL - Flexible Server, see **What's happening to Azure Database for PostgreSQL Single Server**?

Azure Database for PostgreSQL is a managed service that you use to run, manage, and scale highly available PostgreSQL databases in the cloud. In this quickstart, you use an Azure Resource Manager template (ARM template) to create an Azure Database for PostgreSQL - single server in the Azure portal, PowerShell, or Azure CLI.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

Portal

APPLIES TO: Azure Database for PostgreSQL - Single Server

> **Important**
>
> Azure Database for PostgreSQL - Single Server is on the retirement path. We strongly recommend for you to upgrade to Azure Database for PostgreSQL - Flexible Server. For more information about migrating to Azure Database for PostgreSQL - Flexible Server, see **What's happening to Azure Database for PostgreSQL Single Server**?

An Azure account with an active subscription. Create one for free.

Review the template

You create an Azure Database for PostgreSQL server with a configured set of compute and storage resources. To learn more, see Pricing tiers in Azure Database for PostgreSQL - Single Server. You create the server within an Azure resource group.

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.15.31.15270",
      "templateHash": "9297745813759574 35"
    }
  },
  "parameters": {
    "serverName": {
      "type": "string",
      "metadata": {
        "description": "Server Name for Azure Database for PostgreSQL"
      }
    },
    "administratorLogin": {
      "type": "string",
      "minLength": 1,
      "metadata": {
        "description": "Database administrator login name"
      }
    },
    "administratorLoginPassword": {
      "type": "securestring",
```

```
      "minLength": 8,
      "metadata": {
        "description": "Database administrator password"
      }
    },
    "skuCapacity": {
      "type": "int",
      "defaultValue": 2,
      "metadata": {
        "description": "Azure Database for PostgreSQL compute capacity in
vCores (2,4,8,16,32)"
      }
    },
    "skuName": {
      "type": "string",
      "defaultValue": "GP_Gen5_2",
      "metadata": {
        "description": "Azure Database for PostgreSQL sku name "
      }
    },
    "skuSizeMB": {
      "type": "int",
      "defaultValue": 51200,
      "metadata": {
        "description": "Azure Database for PostgreSQL Sku Size "
      }
    },
    "skuTier": {
      "type": "string",
      "defaultValue": "GeneralPurpose",
      "allowedValues": [
        "Basic",
        "GeneralPurpose",
        "MemoryOptimized"
      ],
      "metadata": {
        "description": "Azure Database for PostgreSQL pricing tier"
      }
    },
    "skuFamily": {
      "type": "string",
      "defaultValue": "Gen5",
      "metadata": {
        "description": "Azure Database for PostgreSQL sku family"
      }
    },
    "postgresqlVersion": {
      "type": "string",
      "defaultValue": "11",
      "allowedValues": [
        "9.5",
        "9.6",
        "10",
        "10.0",
        "10.2",
```

```
        "11"
      ],
      "metadata": {
        "description": "PostgreSQL version"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "backupRetentionDays": {
      "type": "int",
      "defaultValue": 7,
      "metadata": {
        "description": "PostgreSQL Server backup retention days"
      }
    },
    "geoRedundantBackup": {
      "type": "string",
      "defaultValue": "Disabled",
      "metadata": {
        "description": "Geo-Redundant Backup setting"
      }
    },
    "virtualNetworkName": {
      "type": "string",
      "defaultValue": "azure_postgresql_vnet",
      "metadata": {
        "description": "Virtual Network Name"
      }
    },
    "subnetName": {
      "type": "string",
      "defaultValue": "azure_postgresql_subnet",
      "metadata": {
        "description": "Subnet Name"
      }
    },
    "virtualNetworkRuleName": {
      "type": "string",
      "defaultValue": "AllowSubnet",
      "metadata": {
        "description": "Virtual Network RuleName"
      }
    },
    "vnetAddressPrefix": {
      "type": "string",
      "defaultValue": "10.0.0.0/16",
      "metadata": {
        "description": "Virtual Network Address Prefix"
      }
    },
```

```
    "subnetPrefix": {
      "type": "string",
      "defaultValue": "10.0.0.0/16",
      "metadata": {
        "description": "Subnet Address Prefix"
      }
    }
  },
  "variables": {
    "firewallrules": [
      {
        "Name": "rule1",
        "StartIpAddress": "0.0.0.0",
        "EndIpAddress": "255.255.255.255"
      },
      {
        "Name": "rule2",
        "StartIpAddress": "0.0.0.0",
        "EndIpAddress": "255.255.255.255"
      }
    ]
  },
  "resources": [
    {
      "type": "Microsoft.DBforPostgreSQL/servers/virtualNetworkRules",
      "apiVersion": "2017-12-01",
      "name": "[format('{0}/{1}', parameters('serverName'),
parameters('virtualNetworkRuleName'))]",
      "properties": {
        "virtualNetworkSubnetId": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('virtualNetworkName'), parameters('subnetName'))]",
        "ignoreMissingVnetServiceEndpoint": true
      },
      "dependsOn": [
        "[resourceId('Microsoft.DBforPostgreSQL/servers',
parameters('serverName'))]",
        "[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('virtualNetworkName'), parameters('subnetName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "2021-05-01",
      "name": "[parameters('virtualNetworkName')]",
      "location": "[parameters('location')]",
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[parameters('vnetAddressPrefix')]"
          ]
        }
      }
    },
    {
```

```
      "type": "Microsoft.Network/virtualNetworks/subnets",
      "apiVersion": "2021-05-01",
      "name": "[format('{0}/{1}', parameters('virtualNetworkName'),
parameters('subnetName'))]",
      "properties": {
        "addressPrefix": "[parameters('subnetPrefix')]"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/virtualNetworks',
parameters('virtualNetworkName'))]"
      ]
    },
    {
      "type": "Microsoft.DBforPostgreSQL/servers",
      "apiVersion": "2017-12-01",
      "name": "[parameters('serverName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('skuName')]",
        "tier": "[parameters('skuTier')]",
        "capacity": "[parameters('skuCapacity')]",
        "size": "[format('{0}', parameters('skuSizeMB'))]",
        "family": "[parameters('skuFamily')]"
      },
      "properties": {
        "createMode": "Default",
        "version": "[parameters('postgresqlVersion')]",
        "administratorLogin": "[parameters('administratorLogin')]",
        "administratorLoginPassword": "
[parameters('administratorLoginPassword')]",
        "storageProfile": {
          "storageMB": "[parameters('skuSizeMB')]",
          "backupRetentionDays": "[parameters('backupRetentionDays')]",
          "geoRedundantBackup": "[parameters('geoRedundantBackup')]"
        }
      }
    },
    {
      "copy": {
        "name": "firewallRules",
        "count": "[length(variables('firewallrules'))]",
        "mode": "serial",
        "batchSize": 1
      },
      "type": "Microsoft.DBforPostgreSQL/servers/firewallRules",
      "apiVersion": "2017-12-01",
      "name": "[format('{0}/{1}', parameters('serverName'),
variables('firewallrules')[copyIndex()].Name)]",
      "properties": {
        "startIpAddress": "[variables('firewallrules')
[copyIndex()].StartIpAddress]",
        "endIpAddress": "[variables('firewallrules')
[copyIndex()].EndIpAddress]"
      },
      "dependsOn": [
```

```
        "[resourceId('Microsoft.DBforPostgreSQL/servers',
parameters('serverName'))]"
      ]
    }
  ]
}
```

The template defines five Azure resources:

- Microsoft.Network/virtualNetworks
- Microsoft.Network/virtualNetworks/subnets
- Microsoft.DBforPostgreSQL/servers
- Microsoft.DBforPostgreSQL/servers/virtualNetworkRules
- Microsoft.DBforPostgreSQL/servers/firewallRules

More Azure Database for PostgreSQL template samples can be found in Azure Quickstart Templates.

Deploy the template

Portal

APPLIES TO: Azure Database for PostgreSQL - Single Server

> **Important**
>
> Azure Database for PostgreSQL - Single Server is on the retirement path. We strongly recommend for you to upgrade to Azure Database for PostgreSQL - Flexible Server. For more information about migrating to Azure Database for PostgreSQL - Flexible Server, see **What's happening to Azure Database for PostgreSQL Single Server**?

Select the following link to deploy the Azure Database for PostgreSQL server template in the Azure portal:

Deploy to Azure

On the **Deploy Azure Database for PostgreSQL with VNet** page:

1. For **Resource group**, select **Create new**, enter a name for the new resource group, and select **OK**.

2. If you created a new resource group, select a **Location** for the resource group and the new server.

3. Enter a **Server Name**, **Administrator Login**, and **Administrator Login Password**.

4. Change the other default settings if you want:

   - **Subscription**: the Azure subscription you want to use for the server.
   - **Sku Capacity**: the vCore capacity, which can be *2* (the default), *4*, *8*, *16*, *32*, or *64*.
   - **Sku Name**: the SKU tier prefix, SKU family, and SKU capacity, joined by underscores, such as *B_Gen5_1*, *GP_Gen5_2* (the default), or *MO_Gen5_32*.

- **Sku Size MB**: the storage size, in megabytes, of the Azure Database for PostgreSQL server (default *51200*).
- **Sku Tier**: the deployment tier, such as *Basic*, *GeneralPurpose* (the default), or *MemoryOptimized*.
- **Sku Family**: *Gen4* or *Gen5* (the default), which indicates hardware generation for server deployment.
- **PostgreSQL Version**: the version of PostgreSQL server to deploy, such as *9.5*, *9.6*, *10*, or *11* (the default).
- **Backup Retention Days**: the desired period for geo-redundant backup retention, in days (default *7*).
- **Geo Redundant Backup**: *Enabled* or *Disabled* (the default), depending on geo-disaster recovery (Geo-DR) requirements.
- **Virtual Network Name**: the name of the virtual network (default *azure_postgresql_vnet*).
- **Subnet Name**: the name of the subnet (default *azure_postgresql_subnet*).
- **Virtual Network Rule Name**: the name of the virtual network rule allowing the subnet (default *AllowSubnet*).
- **Vnet Address Prefix**: the address prefix for the virtual network (default *10.0.0.0/16*).
- **Subnet Prefix**: the address prefix for the subnet (default *10.0.0.0/16*).

5. Read the terms and conditions, and then select **I agree to the terms and conditions stated above**.

6. Select **Purchase**.

Review deployed resources

Portal

APPLIES TO: Azure Database for PostgreSQL - Single Server

> **Important**
>
> Azure Database for PostgreSQL - Single Server is on the retirement path. We strongly recommend for you to upgrade to Azure Database for PostgreSQL - Flexible Server. For more information about migrating to Azure Database for PostgreSQL - Flexible Server, see **What's happening to Azure Database for PostgreSQL Single Server**?

Follow these steps to see an overview of your new Azure Database for PostgreSQL server:

1. In the Azure portal, search for and select **Azure Database for PostgreSQL servers**.
2. In the database list, select your new server. The **Overview** page for your new Azure Database for PostgreSQL server appears.

Exporting ARM template from the portal

You can export an ARM template from the Azure portal. There are two ways to export a template:

- Export from resource group or resource. This option generates a new template from existing resources. The exported template is a "snapshot" of the current state of the resource group. You can export an entire resource group or specific resources within that resource group.
- Export before deployment or from history. This option retrieves an exact copy of a template used for deployment.

When exporting the template, in the `"properties":{ }` section of the PostgreSQL server resource you will notice that `administratorLogin` and `administratorLoginPassword` will not be included for security reasons. You **MUST** add these parameters to your template before deploying the template or the template will fail.

```
"resources": [
    {
      "type": "Microsoft.DBforPostgreSQL/servers",
      "apiVersion": "2017-12-01",
      "name": "[parameters('servers_name')]",
      "location": "southcentralus",
      "sku": {
                "name": "B_Gen5_1",
                "tier": "Basic",
                "family": "Gen5",
                "capacity": 1
            },
      "properties": {
        "administratorLogin": "[parameters('administratorLogin')]",
        "administratorLoginPassword": "
[parameters('administratorLoginPassword')]",
```

Clean up resources

When it's no longer needed, delete the resource group, which deletes the resources in the resource group.

Portal

APPLIES TO: Azure Database for PostgreSQL - Single Server

> **Important**
>
> Azure Database for PostgreSQL - Single Server is on the retirement path. We strongly recommend for you to upgrade to Azure Database for PostgreSQL - Flexible Server. For more information about migrating to Azure Database for PostgreSQL - Flexible Server, see **What's happening to Azure Database for PostgreSQL Single Server**?

1. In the Azure portal, search for and select **Resource groups**.
2. In the resource group list, choose the name of your resource group.
3. In the **Overview** page of your resource group, select **Delete resource group**.
4. In the confirmation dialog box, type the name of your resource group, and then select **Delete**.

Next steps

For a step-by-step tutorial that guides you through the process of creating a template, see:

Tutorial: Create and deploy your first ARM template

Quickstart: Use an ARM template to create an Azure Database for PostgreSQL - Flexible Server

Article • 03/10/2023

APPLIES TO: Azure Database for PostgreSQL - Flexible Server

Flexible server is a managed service that you use to run, manage, and scale highly available PostgreSQL databases in the cloud. You can use an Azure Resource Manager template (ARM template) to provision a PostgreSQL Flexible Server to deploy multiple servers or multiple databases on a server.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

Azure Resource Manager is the deployment and management service for Azure. It provides a management layer that enables you to create, update, and delete resources in your Azure account. You use management features, like access control, locks, and tags, to secure and organize your resources after deployment. To learn about Azure Resource Manager templates, see Template deployment overview.

Prerequisites

An Azure account with an active subscription. Create one for free.

Review the template

An Azure Database for PostgreSQL Server is the parent resource for one or more databases within a region. It provides the scope for management policies that apply to its databases: login, firewall, users, roles, and configurations.

Create a *postgres-flexible-server-template.json* file and copy the following JSON script into it.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
```

```
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "administratorLogin": {
      "type": "string"
    },
    "administratorLoginPassword": {
      "type": "secureString"
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    },
    "serverName": {
      "type": "string"
    },
    "serverEdition": {
      "type": "string",
      "defaultValue": "GeneralPurpose"
    },
    "skuSizeGB": {
      "type": "int",
      "defaultValue": 128
    },
    "dbInstanceType": {
      "type": "string",
      "defaultValue": "Standard_D4ds_v4"
    },
    "haMode": {
      "type": "string",
      "defaultValue": "ZoneRedundant"
    },
    "availabilityZone": {
      "type": "string",
      "defaultValue": "1"
    },
    "version": {
      "type": "string",
      "defaultValue": "12"
    },
    "virtualNetworkExternalId": {
      "type": "string",
      "defaultValue": ""
    },
    "subnetName": {
      "type": "string",
      "defaultValue": ""
    },
    "privateDnsZoneArmResourceId": {
      "type": "string",
      "defaultValue": ""
    }
  },
  "resources": [
    {
```

```
      "type": "Microsoft.DBforPostgreSQL/flexibleServers",
      "apiVersion": "2021-06-01",
      "name": "[parameters('serverName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('dbInstanceType')]",
        "tier": "[parameters('serverEdition')]"
      },
      "properties": {
        "version": "[parameters('version')]",
        "administratorLogin": "[parameters('administratorLogin')]",
        "administratorLoginPassword": "
[parameters('administratorLoginPassword')]",
        "network": {
          "delegatedSubnetResourceId": "
[if(empty(parameters('virtualNetworkExternalId')), json('null'),
json(format('{0}/subnets/{1}', parameters('virtualNetworkExternalId'),
parameters('subnetName'))))]",
          "privateDnsZoneArmResourceId": "
[if(empty(parameters('virtualNetworkExternalId')), json('null'),
parameters('privateDnsZoneArmResourceId'))]"
        },
        "highAvailability": {
          "mode": "[parameters('haMode')]"
        },
        "storage": {
          "storageSizeGB": "[parameters('skuSizeGB')]"
        },
        "backup": {
          "backupRetentionDays": 7,
          "geoRedundantBackup": "Disabled"
        },
        "availabilityZone": "[parameters('availabilityZone')]"
      }
    }
  ]
}
```

These resources are defined in the template:

- Microsoft.DBforPostgreSQL/flexibleServers

Deploy the template

Select **Try it** from the following PowerShell code block to open Azure Cloud Shell.

Azure PowerShell

```
$serverName = Read-Host -Prompt "Enter a name for the new Azure Database for
PostgreSQL server"
$resourceGroupName = Read-Host -Prompt "Enter a name for the new resource
```

```
group where the server will exist"
$location = Read-Host -Prompt "Enter an Azure region (for example,
centralus) for the resource group"
$adminUser = Read-Host -Prompt "Enter the Azure Database for PostgreSQL
server's administrator account name"
$adminPassword = Read-Host -Prompt "Enter the administrator password" -
AsSecureString

New-AzResourceGroup -Name $resourceGroupName -Location $location # Use this
command when you need to create a new resource group for your deployment
New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName `
    -TemplateFile "postgres-flexible-server-template.json" `
    -serverName $serverName `
    -administratorLogin $adminUser `
    -administratorLoginPassword $adminPassword

Read-Host -Prompt "Press [ENTER] to continue ..."
```

Review deployed resources

Follow these steps to verify if your server was created in Azure.

Azure portal

APPLIES TO: Azure Database for PostgreSQL - Flexible Server

1. In the Azure portal, search for and select **Azure Database for PostgreSQL Flexible Servers**.
2. In the database list, select your new server to view the **Overview** page to manage the server.

Clean up resources

Keep this resource group, server, and single database if you want to go to the Next steps. The next steps show you how to connect and query your database using different methods.

To delete the resource group:

Portal

APPLIES TO: Azure Database for PostgreSQL - Flexible Server

In the portal, select the resource group you want to delete.

1. Select **Delete resource group**.
2. To confirm the deletion, type the name of the resource group

Next steps

Migrate your database using dump and restore

Quickstart: Create instance of Azure Database Migration Service using ARM template

Article • 03/08/2023

Use this Azure Resource Manager template (ARM template) to deploy an instance of the Azure Database Migration Service.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

The Azure Database Migration Service ARM template requires the following:

- The latest version of the Azure CLI and/or PowerShell.
- An Azure subscription. If you don't have one, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
```

```
      "templateHash": "8168829270523391611"
    }
  },
  "parameters": {
    "serviceName": {
      "type": "string",
      "metadata": {
        "description": "Name of the new migration service."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location where the resources will be deployed."
      }
    },
    "vnetName": {
      "type": "string",
      "metadata": {
        "description": "Name of the new virtual network."
      }
    },
    "subnetName": {
      "type": "string",
      "metadata": {
        "description": "Name of the new subnet associated with the virtual
network."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "2021-05-01",
      "name": "[parameters('vnetName')]",
      "location": "[parameters('location')]",
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "10.0.0.0/16"
          ]
        }
      }
    },
    {
      "type": "Microsoft.Network/virtualNetworks/subnets",
      "apiVersion": "2021-05-01",
      "name": "[format('{0}/{1}', parameters('vnetName'),
parameters('subnetName'))]",
      "properties": {
        "addressPrefix": "10.0.0.0/24"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/virtualNetworks',
```

```
parameters('vnetName'))]"
      ]
    },
    {
      "type": "Microsoft.DataMigration/services",
      "apiVersion": "2021-10-30-preview",
      "name": "[parameters('serviceName')]",
      "location": "[parameters('location')]",
      "sku": {
        "tier": "Standard",
        "size": "1 vCores",
        "name": "Standard_1vCores"
      },
      "properties": {
        "virtualSubnetId": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('vnetName'), parameters('subnetName'))]"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('vnetName'), parameters('subnetName'))]"
      ]
    }
  ]
}
```

Three Azure resources are defined in the template:

- Microsoft.Network/virtualNetworks: Creates the virtual network.
- Microsoft.Network/virtualNetworks/subnets: Creates the subnet.
- Microsoft.DataMigration/services: Deploys an instance of the Azure Database Migration Service.

More Azure Database Migration Services templates can be found in the quickstart template gallery.

Deploy the template

1. Select the following image to sign in to Azure and open a template. The template creates an instance of the Azure Database Migration Service.

   Deploy to Azure

2. Select or enter the following values.

   - **Subscription**: Select an Azure subscription.
   - **Resource group**: Select an existing resource group from the drop down, or select **Create new** to create a new resource group.

- **Region**: Location where the resources will be deployed.
- **Service Name**: Name of the new migration service.
- **Location**: The location of the resource group, leave as the default of `[resourceGroup().location]`.
- **Vnet Name**: Name of the new virtual network.
- **Subnet Name**: Name of the new subnet associated with the virtual network.

3. Select **Review + create**. After the instance of Azure Database Migration Service has been deployed successfully, you get a notification.

The Azure portal is used to deploy the template. In addition to the Azure portal, you can also use the Azure PowerShell, Azure CLI, and REST API. To learn other deployment methods, see Deploy templates.

Review deployed resources

You can use the Azure CLI to check deployed resources.

Azure CLI

```
echo "Enter the resource group where your SQL Server VM exists:" &&
read resourcegroupName &&
az resource list --resource-group $resourcegroupName
```

Clean up resources

When no longer needed, delete the resource group by using Azure CLI or Azure PowerShell:

CLI

Azure CLI

```
echo "Enter the Resource Group name:" &&
read resourceGroupName &&
az group delete --name $resourceGroupName &&
echo "Press [ENTER] to continue ..."
```

Next steps

For a step-by-step tutorial that guides you through the process of creating a template, see:

Tutorial: Create and deploy your first ARM template

For other ways to deploy Azure Database Migration Service, see:

- Azure portal

To learn more, see an overview of Azure Database Migration Service

Quickstart: Create a single database in Azure SQL Database using an ARM template

Article • 03/30/2023

Creating a single database is the quickest and simplest option for creating a database in Azure SQL Database. This quickstart shows you how to create a single database using an Azure Resource Manager template (ARM template).

An ARM template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account.

Review the template

A single database has a defined set of compute, memory, IO, and storage resources using one of two purchasing models. When you create a single database, you also define a server to manage it and place it within Azure resource group in a specified region.

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
```

```
    "_generator": {
      "name": "bicep",
      "version": "0.12.40.16777",
      "templateHash": "16856611863128783179"
    }
  },
  "parameters": {
    "serverName": {
      "type": "string",
      "defaultValue": "[uniqueString('sql', resourceGroup().id)]",
      "metadata": {
        "description": "The name of the SQL logical server."
      }
    },
    "sqlDBName": {
      "type": "string",
      "defaultValue": "SampleDB",
      "metadata": {
        "description": "The name of the SQL Database."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "administratorLogin": {
      "type": "string",
      "metadata": {
        "description": "The administrator username of the SQL logical
server."
      }
    },
    "administratorLoginPassword": {
      "type": "secureString",
      "metadata": {
        "description": "The administrator password of the SQL logical
server."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Sql/servers",
      "apiVersion": "2022-05-01-preview",
      "name": "[parameters('serverName')]",
      "location": "[parameters('location')]",
      "properties": {
        "administratorLogin": "[parameters('administratorLogin')]",
        "administratorLoginPassword": "
[parameters('administratorLoginPassword')]"
      }
    },
```

```
    {
      "type": "Microsoft.Sql/servers/databases",
      "apiVersion": "2022-05-01-preview",
      "name": "[format('{0}/{1}', parameters('serverName'),
parameters('sqlDBName'))]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard",
        "tier": "Standard"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Sql/servers', parameters('serverName'))]"
      ]
    }
  ]
}
```

These resources are defined in the template:

- **Microsoft.Sql/servers**
- **Microsoft.Sql/servers/databases**

More Azure SQL Database template samples can be found in Azure Quickstart Templates.

Deploy the template

Select **Try it** from the following PowerShell code block to open Azure Cloud Shell.

Azure PowerShell

```
$projectName = Read-Host -Prompt "Enter a project name that is used for
generating resource names"
$location = Read-Host -Prompt "Enter an Azure location (i.e. centralus)"
$adminUser = Read-Host -Prompt "Enter the SQL server administrator username"
$adminPassword = Read-Host -Prompt "Enter the SQl server administrator
password" -AsSecureString

$resourceGroupName = "${projectName}rg"

New-AzResourceGroup -Name $resourceGroupName -Location $location
New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName -
TemplateUri "https://raw.githubusercontent.com/Azure/azure-quickstart-
templates/master/quickstarts/microsoft.sql/sql-database/azuredeploy.json" -
administratorLogin $adminUser -administratorLoginPassword $adminPassword

Read-Host -Prompt "Press [ENTER] to continue ..."
```

Validate the deployment

To query the database, see Query the database.

Clean up resources

Keep this resource group, server, and single database if you want to go to the Next steps. The next steps show you how to connect and query your database using different methods.

To delete the resource group:

Azure PowerShell

```
$resourceGroupName = Read-Host -Prompt "Enter the Resource Group name"
Remove-AzResourceGroup -Name $resourceGroupName
```

Next steps

- Create a server-level firewall rule to connect to the single database from on-premises or remote tools. For more information, see Create a server-level firewall rule.
- After you create a server-level firewall rule, connect and query your database using several different tools and languages.
  - Connect and query using SQL Server Management Studio
  - Connect and query using Azure Data Studio
- To create a single database using the Azure CLI, see Azure CLI samples.
- To create a single database using Azure PowerShell, see Azure PowerShell samples.
- To learn how to create ARM templates, see Create your first template.

Quickstart: Create an Azure SQL Managed Instance using an ARM template

Article • 03/30/2023

This quickstart focuses on the process of deploying an Azure Resource Manager template (ARM template) to create an Azure SQL Managed Instance and vNet. Azure SQL Managed Instance is an intelligent, fully managed, scalable cloud database, with almost 100% feature parity with the SQL Server database engine.

An ARM template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.6.1.6515",
      "templateHash": "13317687096436273875"
    }
```

```
  },
  "parameters": {
    "managedInstanceName": {
      "type": "string",
      "metadata": {
        "description": "Enter managed instance name."
      }
    },
    "administratorLogin": {
      "type": "string",
      "metadata": {
        "description": "Enter user name."
      }
    },
    "administratorLoginPassword": {
      "type": "secureString",
      "metadata": {
        "description": "Enter password."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Enter location. If you leave this field blank
resource group location would be used."
      }
    },
    "virtualNetworkName": {
      "type": "string",
      "defaultValue": "SQLMI-VNET",
      "metadata": {
        "description": "Enter virtual network name. If you leave this field
blank name will be created by the template."
      }
    },
    "addressPrefix": {
      "type": "string",
      "defaultValue": "10.0.0.0/16",
      "metadata": {
        "description": "Enter virtual network address prefix."
      }
    },
    "subnetName": {
      "type": "string",
      "defaultValue": "ManagedInstance",
      "metadata": {
        "description": "Enter subnet name."
      }
    },
    "subnetPrefix": {
      "type": "string",
      "defaultValue": "10.0.0.0/24",
      "metadata": {
        "description": "Enter subnet address prefix."
```

```
      }
    },
    "skuName": {
      "type": "string",
      "defaultValue": "GP_Gen5",
      "allowedValues": [
        "GP_Gen5",
        "BC_Gen5"
      ],
      "metadata": {
        "description": "Enter sku name."
      }
    },
    "vCores": {
      "type": "int",
      "defaultValue": 16,
      "allowedValues": [
        8,
        16,
        24,
        32,
        40,
        64,
        80
      ],
      "metadata": {
        "description": "Enter number of vCores."
      }
    },
    "storageSizeInGB": {
      "type": "int",
      "defaultValue": 256,
      "maxValue": 8192,
      "minValue": 32,
      "metadata": {
        "description": "Enter storage size."
      }
    },
    "licenseType": {
      "type": "string",
      "defaultValue": "LicenseIncluded",
      "allowedValues": [
        "BasePrice",
        "LicenseIncluded"
      ],
      "metadata": {
        "description": "Enter license type."
      }
    }
  },
  "variables": {
    "networkSecurityGroupName": "[format('SQLMI-{0}-NSG',
parameters('managedInstanceName'))]",
    "routeTableName": "[format('SQLMI-{0}-Route-Table',
parameters('managedInstanceName'))]"
```

```
  },
  "resources": [
    {
      "type": "Microsoft.Network/networkSecurityGroups",
      "apiVersion": "2021-08-01",
      "name": "[variables('networkSecurityGroupName')]",
      "location": "[parameters('location')]",
      "properties": {
        "securityRules": [
          {
            "name": "allow_tds_inbound",
            "properties": {
              "description": "Allow access to data",
              "protocol": "Tcp",
              "sourcePortRange": "*",
              "destinationPortRange": "1433",
              "sourceAddressPrefix": "VirtualNetwork",
              "destinationAddressPrefix": "*",
              "access": "Allow",
              "priority": 1000,
              "direction": "Inbound"
            }
          },
          {
            "name": "allow_redirect_inbound",
            "properties": {
              "description": "Allow inbound redirect traffic to Managed
Instance inside the virtual network",
              "protocol": "Tcp",
              "sourcePortRange": "*",
              "destinationPortRange": "11000-11999",
              "sourceAddressPrefix": "VirtualNetwork",
              "destinationAddressPrefix": "*",
              "access": "Allow",
              "priority": 1100,
              "direction": "Inbound"
            }
          },
          {
            "name": "deny_all_inbound",
            "properties": {
              "description": "Deny all other inbound traffic",
              "protocol": "*",
              "sourcePortRange": "*",
              "destinationPortRange": "*",
              "sourceAddressPrefix": "*",
              "destinationAddressPrefix": "*",
              "access": "Deny",
              "priority": 4096,
              "direction": "Inbound"
            }
          },
          {
            "name": "deny_all_outbound",
            "properties": {
```

```
              "description": "Deny all other outbound traffic",
              "protocol": "*",
              "sourcePortRange": "*",
              "destinationPortRange": "*",
              "sourceAddressPrefix": "*",
              "destinationAddressPrefix": "*",
              "access": "Deny",
              "priority": 4096,
              "direction": "Outbound"
            }
          }
        ]
      }
    },
    {
      "type": "Microsoft.Network/routeTables",
      "apiVersion": "2021-08-01",
      "name": "[variables('routeTableName')]",
      "location": "[parameters('location')]",
      "properties": {
        "disableBgpRoutePropagation": false
      }
    },
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "2021-08-01",
      "name": "[parameters('virtualNetworkName')]",
      "location": "[parameters('location')]",
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[parameters('addressPrefix')]"
          ]
        },
        "subnets": [
          {
            "name": "[parameters('subnetName')]",
            "properties": {
              "addressPrefix": "[parameters('subnetPrefix')]",
              "routeTable": {
                "id": "[resourceId('Microsoft.Network/routeTables',
variables('routeTableName'))]"
              },
              "networkSecurityGroup": {
                "id": "
[resourceId('Microsoft.Network/networkSecurityGroups',
variables('networkSecurityGroupName'))]"
              },
              "delegations": [
                {
                  "name": "managedInstanceDelegation",
                  "properties": {
                    "serviceName": "Microsoft.Sql/managedInstances"
                  }
                }
```

```
                ]
              }
            }
          ]
        },
        "dependsOn": [
          "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('networkSecurityGroupName'))]",
          "[resourceId('Microsoft.Network/routeTables',
variables('routeTableName'))]"
        ]
      },
      {
        "type": "Microsoft.Sql/managedInstances",
        "apiVersion": "2021-11-01-preview",
        "name": "[parameters('managedInstanceName')]",
        "location": "[parameters('location')]",
        "sku": {
          "name": "[parameters('skuName')]"
        },
        "identity": {
          "type": "SystemAssigned"
        },
        "properties": {
          "administratorLogin": "[parameters('administratorLogin')]",
          "administratorLoginPassword": "
[parameters('administratorLoginPassword')]",
          "subnetId": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('virtualNetworkName'), parameters('subnetName'))]",
          "storageSizeInGB": "[parameters('storageSizeInGB')]",
          "vCores": "[parameters('vCores')]",
          "licenseType": "[parameters('licenseType')]"
        },
        "dependsOn": [
          "[resourceId('Microsoft.Network/virtualNetworks',
parameters('virtualNetworkName'))]"
        ]
      }
    ]
}
```

These resources are defined in the template:

- Microsoft.Network/networkSecurityGroups
- Microsoft.Network/routeTables
- Microsoft.Network/virtualNetworks
- Microsoft.Sql/managedinstances

More template samples can be found in Azure Quickstart Templates.

Deploy the template

Select **Try it** from the following PowerShell code block to open Azure Cloud Shell.

> **Important**
>
> Deploying a managed instance is a long-running operation. Deployment of the first instance in the subnet typically takes much longer than deploying into a subnet with existing managed instances. For average provisioning times, see **SQL Managed Instance management operations**.

PowerShell

Azure PowerShell

```
$projectName = Read-Host -Prompt "Enter a project name that is used for generating resource names"
$location = Read-Host -Prompt "Enter the location (i.e. centralus)"
$templateUri = "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.sql/sqlmi-new-vnet/azuredeploy.json"

$resourceGroupName = "${projectName}rg"

New-AzResourceGroup -Name $resourceGroupName -Location $location
New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName -TemplateUri $templateUri

Read-Host -Prompt "Press [ENTER] to continue ..."
```

Review deployed resources

Visit the Azure portal and verify the managed instance is in your selected resource group. Because creating a managed instance can take some time, you might need to check the **Deployments** link on your resource group's **Overview** page.

- For a quickstart that shows how to connect to SQL Managed Instance from an Azure virtual machine, see Configure an Azure virtual machine connection.
- For a quickstart that shows how to connect to SQL Managed Instance from an on-premises client computer by using a point-to-site connection, see Configure a point-to-site connection.

Clean up resources

Keep the managed instance if you want to go to the Next steps, but delete the managed instance and related resources after completing any additional tutorials. After deleting a managed instance, see Delete a subnet after deleting a managed instance.

To delete the resource group:

PowerShell

Azure PowerShell

```
$resourceGroupName = Read-Host -Prompt "Enter the Resource Group name"
Remove-AzResourceGroup -Name $resourceGroupName
```

Next steps

Configure an Azure VM to connect to Azure SQL Managed Instance

Quickstart: Create SQL Server VM using an ARM template

Article • 03/30/2023

Use this Azure Resource Manager template (ARM template) to deploy a SQL Server on Azure Virtual Machine (VM).

An ARM template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

The SQL Server VM ARM template requires the following:

- The latest version of the Azure CLI and/or PowerShell.
- A preconfigured resource group with a prepared virtual network and subnet.
- An Azure subscription. If you don't have one, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.10.61.36676",
      "templateHash": "11696421255365761999"
```

```
      }
    },
    "parameters": {
      "virtualMachineName": {
        "type": "string",
        "defaultValue": "myVM",
        "metadata": {
          "description": "The name of the VM"
        }
      },
      "virtualMachineSize": {
        "type": "string",
        "defaultValue": "Standard_D8s_v3",
        "metadata": {
          "description": "The virtual machine size."
        }
      },
      "existingVirtualNetworkName": {
        "type": "string",
        "metadata": {
          "description": "Specify the name of an existing VNet in the same
resource group"
        }
      },
      "existingVnetResourceGroup": {
        "type": "string",
        "defaultValue": "[resourceGroup().name]",
        "metadata": {
          "description": "Specify the resrouce group of the existing VNet"
        }
      },
      "existingSubnetName": {
        "type": "string",
        "metadata": {
          "description": "Specify the name of the Subnet Name"
        }
      },
      "imageOffer": {
        "type": "string",
        "defaultValue": "sql2019-ws2019",
        "allowedValues": [
          "sql2019-ws2019",
          "sql2017-ws2019",
          "SQL2017-WS2016",
          "SQL2016SP1-WS2016",
          "SQL2016SP2-WS2016",
          "SQL2014SP3-WS2012R2",
          "SQL2014SP2-WS2012R2"
        ],
        "metadata": {
          "description": "Windows Server and SQL Offer"
        }
      },
      "sqlSku": {
        "type": "string",
```

```
      "defaultValue": "Standard",
      "allowedValues": [
        "Standard",
        "Enterprise",
        "SQLDEV",
        "Web",
        "Express"
      ],
      "metadata": {
        "description": "SQL Server Sku"
      }
    },
    "adminUsername": {
      "type": "string",
      "metadata": {
        "description": "The admin user name of the VM"
      }
    },
    "adminPassword": {
      "type": "secureString",
      "metadata": {
        "description": "The admin password of the VM"
      }
    },
    "storageWorkloadType": {
      "type": "string",
      "defaultValue": "General",
      "allowedValues": [
        "General",
        "OLTP",
        "DW"
      ],
      "metadata": {
        "description": "SQL Server Workload Type"
      }
    },
    "sqlDataDisksCount": {
      "type": "int",
      "defaultValue": 1,
      "maxValue": 8,
      "minValue": 1,
      "metadata": {
        "description": "Amount of data disks (1TB each) for SQL Data files"
      }
    },
    "dataPath": {
      "type": "string",
      "defaultValue": "F:\\SQLData",
      "metadata": {
        "description": "Path for SQL Data files. Please choose drive letter
from F to Z, and other drives from A to E are reserved for system"
      }
    },
    "sqlLogDisksCount": {
      "type": "int",
```

```
      "defaultValue": 1,
      "maxValue": 8,
      "minValue": 1,
      "metadata": {
        "description": "Amount of data disks (1TB each) for SQL Log files"
      }
    },
    "logPath": {
      "type": "string",
      "defaultValue": "G:\\SQLLog",
      "metadata": {
        "description": "Path for SQL Log files. Please choose drive letter
from F to Z and different than the one used for SQL data. Drive letter from
A to E are reserved for system"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    }
  },
  "variables": {
    "networkInterfaceName": "[format('{0}-nic',
parameters('virtualMachineName'))]",
    "networkSecurityGroupName": "[format('{0}-nsg',
parameters('virtualMachineName'))]",
    "networkSecurityGroupRules": [
      {
        "name": "RDP",
        "properties": {
          "priority": 300,
          "protocol": "Tcp",
          "access": "Allow",
          "direction": "Inbound",
          "sourceAddressPrefix": "*",
          "sourcePortRange": "*",
          "destinationAddressPrefix": "*",
          "destinationPortRange": "3389"
        }
      }
    ],
    "publicIpAddressName": "[format('{0}-publicip-{1}',
parameters('virtualMachineName'),
uniqueString(parameters('virtualMachineName')))]",
    "publicIpAddressType": "Dynamic",
    "publicIpAddressSku": "Basic",
    "diskConfigurationType": "NEW",
    "nsgId": "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('networkSecurityGroupName'))]",
    "subnetRef": "[resourceId(parameters('existingVnetResourceGroup'),
'Microsoft.Network/virtualNetWorks/subnets',
parameters('existingVirtualNetworkName'),
```

```
parameters('existingSubnetName'))]",
    "dataDisksLuns": "[range(0, parameters('sqlDataDisksCount'))]",
    "logDisksLuns": "[range(parameters('sqlDataDisksCount'),
parameters('sqlLogDisksCount'))]",
    "dataDisks": {
      "createOption": "Empty",
      "caching": "ReadOnly",
      "writeAcceleratorEnabled": false,
      "storageAccountType": "Premium_LRS",
      "diskSizeGB": 1023
    },
    "tempDbPath": "D:\\SQLTemp"
  },
  "resources": [
    {
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2022-01-01",
      "name": "[variables('publicIpAddressName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[variables('publicIpAddressSku')]"
      },
      "properties": {
        "publicIPAllocationMethod": "[variables('publicIpAddressType')]"
      }
    },
    {
      "type": "Microsoft.Network/networkSecurityGroups",
      "apiVersion": "2022-01-01",
      "name": "[variables('networkSecurityGroupName')]",
      "location": "[parameters('location')]",
      "properties": {
        "securityRules": "[variables('networkSecurityGroupRules')]"
      }
    },
    {
      "type": "Microsoft.Network/networkInterfaces",
      "apiVersion": "2022-01-01",
      "name": "[variables('networkInterfaceName')]",
      "location": "[parameters('location')]",
      "properties": {
        "ipConfigurations": [
          {
            "name": "ipconfig1",
            "properties": {
              "subnet": {
                "id": "[variables('subnetRef')]"
              },
              "privateIPAllocationMethod": "Dynamic",
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses',
variables('publicIpAddressName'))]"
              }
            }
          }
```

```
        ],
        "enableAcceleratedNetworking": true,
        "networkSecurityGroup": {
          "id": "[variables('nsgId')]"
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('networkSecurityGroupName'))]",
        "[resourceId('Microsoft.Network/publicIPAddresses',
variables('publicIpAddressName'))]"
      ]
    },
    {
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2022-03-01",
      "name": "[parameters('virtualMachineName')]",
      "location": "[parameters('location')]",
      "properties": {
        "hardwareProfile": {
          "vmSize": "[parameters('virtualMachineSize')]"
        },
        "storageProfile": {
          "copy": [
            {
              "name": "dataDisks",
              "count": "[length(range(0,
add(parameters('sqlDataDisksCount'), parameters('sqlLogDisksCount'))))]",
              "input": {
                "lun": "[range(0, add(parameters('sqlDataDisksCount'),
parameters('sqlLogDisksCount')))[copyIndex('dataDisks')]]",
                "createOption": "[variables('dataDisks').createOption]",
                "caching": "[if(greaterOrEquals(range(0,
add(parameters('sqlDataDisksCount'), parameters('sqlLogDisksCount')))
[copyIndex('dataDisks')], parameters('sqlDataDisksCount')), 'None',
variables('dataDisks').caching)]",
                "writeAcceleratorEnabled": "
[variables('dataDisks').writeAcceleratorEnabled]",
                "diskSizeGB": "[variables('dataDisks').diskSizeGB]",
                "managedDisk": {
                  "storageAccountType": "
[variables('dataDisks').storageAccountType]"
                }
              }
            }
          ],
          "osDisk": {
            "createOption": "FromImage",
            "managedDisk": {
              "storageAccountType": "Premium_LRS"
            }
          },
          "imageReference": {
            "publisher": "MicrosoftSQLServer",
            "offer": "[parameters('imageOffer')]",
```

```
            "sku": "[parameters('sqlSku')]",
            "version": "latest"
          }
        },
        "networkProfile": {
          "networkInterfaces": [
            {
              "id": "[resourceId('Microsoft.Network/networkInterfaces',
variables('networkInterfaceName'))]"
            }
          ]
        },
        "osProfile": {
          "computerName": "[parameters('virtualMachineName')]",
          "adminUsername": "[parameters('adminUsername')]",
          "adminPassword": "[parameters('adminPassword')]",
          "windowsConfiguration": {
            "enableAutomaticUpdates": true,
            "provisionVMAgent": true
          }
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkInterfaces',
variables('networkInterfaceName'))]"
      ]
    },
    {
      "type": "Microsoft.SqlVirtualMachine/sqlVirtualMachines",
      "apiVersion": "2022-07-01-preview",
      "name": "[parameters('virtualMachineName')]",
      "location": "[parameters('location')]",
      "properties": {
        "virtualMachineResourceId": "
[resourceId('Microsoft.Compute/virtualMachines',
parameters('virtualMachineName'))]",
        "sqlManagement": "Full",
        "sqlServerLicenseType": "PAYG",
        "storageConfigurationSettings": {
          "diskConfigurationType": "[variables('diskConfigurationType')]",
          "storageWorkloadType": "[parameters('storageWorkloadType')]",
          "sqlDataSettings": {
            "luns": "[variables('dataDisksLuns')]",
            "defaultFilePath": "[parameters('dataPath')]"
          },
          "sqlLogSettings": {
            "luns": "[variables('logDisksLuns')]",
            "defaultFilePath": "[parameters('logPath')]"
          },
          "sqlTempDbSettings": {
            "defaultFilePath": "[variables('tempDbPath')]"
          }
        }
      },
      "dependsOn": [
```

```
        "[resourceId('Microsoft.Compute/virtualMachines',
parameters('virtualMachineName'))]"
      ]
    }
  ],
  "outputs": {
    "adminUsername": {
      "type": "string",
      "value": "[parameters('adminUsername')]"
    }
  }
}
```

Five Azure resources are defined in the template:

- Microsoft.Network/publicIpAddresses: Creates a public IP address.
- Microsoft.Network/networkSecurityGroups: Creates a network security group.
- Microsoft.Network/networkInterfaces: Configures the network interface.
- Microsoft.Compute/virtualMachines: Creates a virtual machine in Azure.
- Microsoft.SqlVirtualMachine/SqlVirtualMachines: registers the virtual machine with the SQL IaaS Agent extension.

More SQL Server on Azure VM templates can be found in the quickstart template gallery.

Deploy the template

1. Select the following image to sign in to Azure and open a template. The template creates a virtual machine with the intended SQL Server version installed to it, and registered with the SQL IaaS Agent extension.

   Deploy to Azure

2. Select or enter the following values.

   - **Subscription**: Select an Azure subscription.
   - **Resource group**: The prepared resource group for your SQL Server VM.
   - **Region**: Select a region. For example, **Central US**.
   - **Virtual Machine Name**: Enter a name for SQL Server virtual machine.
   - **Virtual Machine Size**: Choose the appropriate size for your virtual machine from the drop-down.
   - **Existing Virtual Network Name**: Enter the name of the prepared virtual network for your SQL Server VM.

- **Existing Vnet Resource Group**: Enter the resource group where your virtual network was prepared.
- **Existing Subnet Name**: The name of your prepared subnet.
- **Image Offer**: Choose the SQL Server and Windows Server image that best suits your business needs.
- **SQL Sku**: Choose the edition of SQL Server SKU that best suits your business needs.
- **Admin Username**: The username for the administrator of the virtual machine.
- **Admin Password**: The password used by the VM administrator account.
- **Storage Workload Type**: The type of storage for the workload that best matches your business.
- **Sql Data Disks Count**: The number of disks SQL Server uses for data files.
- **Data Path**: The path for the SQL Server data files.
- **Sql Log Disks Count**: The number of disks SQL Server uses for log files.
- **Log Path**: The path for the SQL Server log files.
- **Location**: The location for all of the resources, this value should remain the default of `[resourceGroup().location]`.

3. Select **Review + create**. After the SQL Server VM has been deployed successfully, you get a notification.

The Azure portal is used to deploy the template. In addition to the Azure portal, you can also use Azure PowerShell, the Azure CLI, and REST API. To learn other deployment methods, see Deploy templates.

Review deployed resources

You can use the Azure CLI to check deployed resources.

Azure CLI

```
echo "Enter the resource group where your SQL Server VM exists:" &&
read resourcegroupName &&
az resource list --resource-group $resourcegroupName
```

Clean up resources

When no longer needed, delete the resource group by using Azure CLI or Azure PowerShell:

CLI

Azure CLI

```
echo "Enter the Resource Group name:" &&
read resourceGroupName &&
az group delete --name $resourceGroupName &&
echo "Press [ENTER] to continue ..."
```

Next steps

For a step-by-step tutorial that guides you through the process of creating a template, see:

Tutorial: Create and deploy your first ARM template

For other ways to deploy a SQL Server VM, see:

- Azure portal
- PowerShell

To learn more, see an overview of SQL Server on Azure VMs.

Quickstart: Create an Azure App Configuration store by using an ARM template

Article • 03/09/2023

This quickstart describes how to :

- Deploy an App Configuration store using an Azure Resource Manager template (ARM template).
- Create key-values in an App Configuration store using ARM template.
- Read key-values in an App Configuration store from ARM template.

> **Tip**
>
> Feature flags and Key Vault references are special types of key-values. Check out the **Next steps** for examples of creating them using the ARM template.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Authorization

Accessing key-value data inside an ARM template requires an Azure Resource Manager role, such as contributor or owner. Access via one of the Azure App Configuration data plane roles currently is not supported.

> **Note**
>
> Key-value data access inside an ARM template is disabled if access key authentication is disabled. For more information, see **disable access key authentication**.

Review the template

The template used in this quickstart is from Azure Quickstart Templates. It creates a new App Configuration store with two key-values inside. It then uses the `reference` function to output the values of the two key-value resources. Reading the key's value in this way allows it to be used in other places in the template.

The quickstart uses the `copy` element to create multiple instances of key-value resource. To learn more about the `copy` element, see Resource iteration in ARM templates.

> **Important**
>
> This template requires App Configuration resource provider version `2020-07-01-preview` or later. This version uses the `reference` function to read key-values. The `listKeyValue` function that was used to read key-values in the previous version is not available starting in version `2020-07-01-preview`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "2173262573838896712"
    }
  },
  "parameters": {
    "configStoreName": {
      "type": "string",
      "metadata": {
        "description": "Specifies the name of the App Configuration store."
      }
    },
    "location": {
      "type": "string",
```

```
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Specifies the Azure location where the app
configuration store should be created."
      }
    },
    "keyValueNames": {
      "type": "array",
      "defaultValue": [
        "myKey",
        "myKey$myLabel"
      ],
      "metadata": {
        "description": "Specifies the names of the key-value resources. The
name is a combination of key and label with $ as delimiter. The label is
optional."
      }
    },
    "keyValueValues": {
      "type": "array",
      "defaultValue": [
        "Key-value without label",
        "Key-value with label"
      ],
      "metadata": {
        "description": "Specifies the values of the key-value resources.
It's optional"
      }
    },
    "contentType": {
      "type": "string",
      "defaultValue": "the-content-type",
      "metadata": {
        "description": "Specifies the content type of the key-value
resources. For feature flag, the value should be
application/vnd.microsoft.appconfig.ff+json;charset=utf-8. For Key Value
reference, the value should be
application/vnd.microsoft.appconfig.keyvaultref+json;charset=utf-8.
Otherwise, it's optional."
      }
    },
    "tags": {
      "type": "object",
      "defaultValue": {
        "tag1": "tag-value-1",
        "tag2": "tag-value-2"
      },
      "metadata": {
        "description": "Adds tags for the key-value resources. It's
optional"
      }
    }
  },
  "resources": [
    {
```

```
      "type": "Microsoft.AppConfiguration/configurationStores",
      "apiVersion": "2021-10-01-preview",
      "name": "[parameters('configStoreName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "standard"
      }
    },
    {
      "copy": {
        "name": "configStoreKeyValue",
        "count": "[length(parameters('keyValueNames'))]"
      },
      "type": "Microsoft.AppConfiguration/configurationStores/keyValues",
      "apiVersion": "2021-10-01-preview",
      "name": "[format('{0}/{1}', parameters('configStoreName'),
parameters('keyValueNames')[copyIndex()])]",
      "properties": {
        "value": "[parameters('keyValueValues')[copyIndex()]]",
        "contentType": "[parameters('contentType')]",
        "tags": "[parameters('tags')]"
      },
      "dependsOn": [
        "[resourceId('Microsoft.AppConfiguration/configurationStores',
parameters('configStoreName'))]"
      ]
    }
  ],
  "outputs": {
    "reference_key_value_value": {
      "type": "string",
      "value": "
[reference(resourceId('Microsoft.AppConfiguration/configurationStores/keyVal
ues', parameters('configStoreName'), parameters('keyValueNames')
[0])).value]"
    },
    "reference_key_value_object": {
      "type": "object",
      "value": "
[reference(resourceId('Microsoft.AppConfiguration/configurationStores/keyVal
ues', parameters('configStoreName'), parameters('keyValueNames')[1]), '2021-
10-01-preview', 'full')]"
    }
  }
}
```

Two Azure resources are defined in the template:

- Microsoft.AppConfiguration/configurationStores: create an App Configuration store.
- Microsoft.AppConfiguration/configurationStores/keyValues: create a key-value inside the App Configuration store.

Tip

The `keyValues` resource's name is a combination of key and label. The key and label are joined by the `$` delimiter. The label is optional. In the above example, the `keyValues` resource with name `myKey` creates a key-value without a label.

Percent-encoding, also known as URL encoding, allows keys or labels to include characters that are not allowed in ARM template resource names. `%` is not an allowed character either, so `~` is used in its place. To correctly encode a name, follow these steps:

1. Apply URL encoding
2. Replace `~` with `~7E`
3. Replace `%` with `~`

For example, to create a key-value pair with key name `AppName:DbEndpoint` and label name `Test`, the resource name should be `AppName~3ADbEndpoint$Test`.

Note

App Configuration allows key-value data access over a **private link** from your virtual network. By default, when the feature is enabled, all requests for your App Configuration data over the public network are denied. Because the ARM template runs outside your virtual network, data access from an ARM template isn't allowed. To allow data access from an ARM template when a private link is used, you can enable public network access by using the following Azure CLI command. It's important to consider the security implications of enabling public network access in this scenario.

Azure CLI

```
az appconfig update -g MyResourceGroup -n MyAppConfiguration --enable-public-network true
```

Deploy the template

Select the following image to sign in to Azure and open a template. The template creates an App Configuration store with two key-values inside.

Deploy to Azure

You can also deploy the template by using the following PowerShell cmdlet. The key-values will be in the output of PowerShell console.

Azure PowerShell

```
$projectName = Read-Host -Prompt "Enter a project name that is used for generating resource names"
$location = Read-Host -Prompt "Enter the location (i.e. centralus)"
$templateUri = "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.appconfiguration/app-configuration-store-kv/azuredeploy.json"

$resourceGroupName = "${projectName}rg"

New-AzResourceGroup -Name $resourceGroupName -Location "$location"
New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName -TemplateUri $templateUri

Read-Host -Prompt "Press [ENTER] to continue ..."
```

Review deployed resources

1. Sign in to the Azure portal.
2. In the Azure portal search box, type **App Configuration**. Select **App Configuration** from the list.
3. Select the newly created App Configuration resource.
4. Under **Operations**, click **Configuration explorer**.
5. Verify that two key-values exist.

Clean up resources

When no longer needed, delete the resource group, the App Configuration store, and all related resources. If you're planning to use the App Configuration store in the future, you can skip deleting it. If you aren't going to continue to use this store, delete all resources created by this quickstart by running the following cmdlet:

Azure PowerShell

```
$resourceGroupName = Read-Host -Prompt "Enter the Resource Group name"
Remove-AzResourceGroup -Name $resourceGroupName
Write-Host "Press [ENTER] to continue..."
```

Next steps

To learn about adding feature flag and Key Vault reference to an App Configuration store, check below ARM template examples.

- ARM template for feature flag
- ARM template for Key Vault reference

Quickstart: Use an ARM template to create a lab in DevTest Labs

Article • 04/13/2023

This quickstart uses an Azure Resource Manager (ARM) template to create a lab in Azure DevTest Labs that has one Windows Server 2019 Datacenter virtual machine (VM) in it.

In this quickstart, you take the following actions:

- ✓ Review the ARM template.
- ✓ Deploy the ARM template to create a lab and VM.
- ✓ Verify the deployment.
- ✓ Clean up resources.

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

DevTest Labs can use ARM templates for many tasks, from creating and provisioning labs to adding users. This quickstart uses the Creates a lab with a claimed VM ARM template from the Azure Quickstart Templates gallery. The template defines the following resource types:

- Microsoft.DevTestLab/labs creates the lab.
- Microsoft.DevTestLab/labs/virtualnetworks creates a virtual network.
- Microsoft.DevTestLab/labs/virtualmachines creates the lab VM.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
```

```
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "12097184829658896385"
    }
  },
  "parameters": {
    "labName": {
      "type": "string",
      "metadata": {
        "description": "The name of the new lab instance to be created"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "vmName": {
      "type": "string",
      "metadata": {
        "description": "The name of the vm to be created."
      }
    },
    "vmSize": {
      "type": "string",
      "defaultValue": "Standard_D4_v3",
      "metadata": {
        "description": "The size of the vm to be created."
      }
    },
    "userName": {
      "type": "string",
      "metadata": {
        "description": "The username for the local account that will be
created on the new vm."
      }
    },
    "password": {
      "type": "secureString",
      "metadata": {
        "description": "The password for the local account that will be
created on the new vm."
      }
    }
  },
  "variables": {
    "labSubnetName": "[format('{0}Subnet',
variables('labVirtualNetworkName'))]",
    "labVirtualNetworkId": "
[resourceId('Microsoft.DevTestLab/labs/virtualnetworks',
parameters('labName'), variables('labVirtualNetworkName'))]",
    "labVirtualNetworkName": "[format('Dtl{0}', parameters('labName'))]"
  },
```

```
  "resources": [
    {
      "type": "Microsoft.DevTestLab/labs",
      "apiVersion": "2018-09-15",
      "name": "[parameters('labName')]",
      "location": "[parameters('location')]"
    },
    {
      "type": "Microsoft.DevTestLab/labs/virtualnetworks",
      "apiVersion": "2018-09-15",
      "name": "[format('{0}/{1}', parameters('labName'),
variables('labVirtualNetworkName'))]",
      "dependsOn": [
        "[resourceId('Microsoft.DevTestLab/labs', parameters('labName'))]"
      ]
    },
    {
      "type": "Microsoft.DevTestLab/labs/virtualmachines",
      "apiVersion": "2018-09-15",
      "name": "[format('{0}/{1}', parameters('labName'),
parameters('vmName'))]",
      "location": "[parameters('location')]",
      "properties": {
        "userName": "[parameters('userName')]",
        "password": "[parameters('password')]",
        "labVirtualNetworkId": "[variables('labVirtualNetworkId')]",
        "labSubnetName": "[variables('labSubnetName')]",
        "size": "[parameters('vmSize')]",
        "allowClaim": false,
        "galleryImageReference": {
          "offer": "WindowsServer",
          "publisher": "MicrosoftWindowsServer",
          "sku": "2019-Datacenter",
          "osType": "Windows",
          "version": "latest"
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.DevTestLab/labs', parameters('labName'))]",
        "[resourceId('Microsoft.DevTestLab/labs/virtualnetworks',
parameters('labName'), variables('labVirtualNetworkName'))]"
      ]
    }
  ],
  "outputs": {
    "labId": {
      "type": "string",
      "value": "[resourceId('Microsoft.DevTestLab/labs',
parameters('labName'))]"
    }
  }
}
```

The Azure Quickstart Templates gallery and Azure Quickstart Templates public GitHub repository have several other DevTest Labs ARM quickstart templates.

The Azure Lab Services Community public GitHub repository also has many DevTest Labs artifacts, environments, PowerShell scripts, and quickstart ARM templates you can use or customize for your needs.

Deploy the template

1. Select the following **Deploy to Azure** button to sign in to the Azure portal and open the quickstart ARM template:

   Deploy to Azure

2. On the **Creates a lab in Azure DevTest Labs with a claimed VM** screen, complete the following items:

   - **Resource group**: Select an existing resource group from the dropdown list, or create a new resource group so it's easy to clean up later.
   - **Region**: If you created a new resource group, select a location for the resource group and lab.
   - **Lab Name**: Enter a name for the new lab.
   - **Vm Name**: Enter a name for the new VM.
   - **User Name**: Enter a name for the user who can access the VM.
   - **Password**: Enter a password for the VM user.

3. Select **Review + create**, and when validation passes, select **Create**.

4. During the deployment, you can select the **Notifications** icon at the top of the screen to see deployment progress on the template **Overview** page. Deployment, especially creating a VM, takes a while.

Validate the deployment

1. When the deployment is complete, select **Go to resource group** from the template **Overview** page or from **Notifications**.

2. The **Resource group** page lists the resources in the resource group, including your lab and its dependent resources like virtual networks and VMs. Select the **DevTest Lab** resource to go to the lab's **Overview** page.

3. On the lab **Overview** page, you can see the VM under **My virtual machines**.

> **Note**
>
> The deployment also creates a resource group for the VM. The resource group contains VM resources like the IP address, network interface, and disk. The resource group appears in your subscription's **Resource groups** list with the name **<lab name>-<vm name>-<numerical string>**.

Clean up resources

When you're done using these lab resources, delete them to avoid further charges. You can't delete a resource group that has a lab in it, so delete the lab first:

1. On the lab overview page, select **Delete** from the top menu.

2. On the **Are you sure you want to delete it** page, enter the lab name, and then select **Delete**.

   During the deletion, you can select **Notifications** at the top of your screen to view progress. Deleting the lab takes a while.

You can now delete the resource group that contained the lab, which deletes all resources in the resource group.

1. Select the resource group that contained the lab from your subscription's **Resource groups** list.

2. At the top of the page, select **Delete resource group**.

3. On the **Are you sure you want to delete "<resource group name>"** page, enter the resource group name, and then select **Delete**.

Next steps

In this quickstart, you created a lab that has a Windows VM. To learn how to connect to and manage lab VMs, see the next tutorial:

Tutorial: Work with lab VMs

Quickstart: Use an ARM template to deploy a Linux web app to Azure

Article • 03/30/2023

Azure DevOps Services

Get started with Azure Resource Manager templates (ARM templates) by deploying a Linux web app with MySQL. ARM templates give you a way to save your configuration in code. Using an ARM template is an example of infrastructure as code and a good DevOps practice.

An ARM template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

You can use either JSON or Bicep syntax to deploy Azure resources. Learn more about the difference between JSON and Bicep for templates.

Prerequisites

Before you begin, you need:

- An Azure account with an active subscription. Create an account for free.
- An active Azure DevOps organization. Sign up for Azure Pipelines.
- (For Bicep deployments) An existing resource group. Create a resource group with Azure portal, Azure CLI, or Azure PowerShell.

Get the code

Fork this repository on GitHub:

```
https://github.com/Azure/azure-quickstart-
templates/tree/master/quickstarts/microsoft.web/webapp-linux-managed-mysql
```

Review the template

JSON

The template used in this quickstart is from Azure Quickstart Templates.

The template defines several resources:

- Microsoft.Web/serverfarms
- Microsoft.Web/sites
- Microsoft.DBforMySQL/servers
- Microsoft.DBforMySQL/servers/firewallrules
- Microsoft.DBforMySQL/servers/databases

Create your pipeline and deploy your template

1. Sign in to your Azure DevOps organization and navigate to your project. Create a project if you do not already have one.

2. Go to **Pipelines**, and then select **Create Pipeline**.

3. Select **GitHub** as the location of your source code.

   > **Note**
   >
   > You may be redirected to GitHub to sign in. If so, enter your GitHub credentials.

4. When the list of repositories appears, select `yourname/azure-quickstart-templates/`.

   > **Note**
   >
   > You may be redirected to GitHub to install the Azure Pipelines app. If so, select **Approve and install**.

5. When the Configure tab appears, select `Starter pipeline`.

6. Replace the content of your pipeline with this code:

   ```yml
   ```

```yaml
trigger:
- none

pool:
  vmImage: 'ubuntu-latest'
```

7. Create three variables: `siteName`, `administratorLogin`, and `adminPass`. `adminPass` needs to be a secret variable.

   - Select **Variables**.
   - Use the `+` sign to add three variables. When you create `adminPass`, select **Keep this value secret**.
   - Click **Save** when you're done.

Variable	Value	Secret?
siteName	`mytestsite`	No
adminUser	`fabrikam`	No
adminPass	`Fqdn:5362!`	Yes

8. Map the secret variable `$(adminPass)` so that it is available in your Azure Resource Group Deployment task. At the top of your YAML file, map `$(adminPass)` to `$(ARM_PASS)`.

```yml
variables:
  ARM_PASS: $(adminPass)

trigger:
- none

pool:
  vmImage: 'ubuntu-latest'
```

9. Add the Copy Files task to the YAML file. You will use the `101-webapp-linux-managed-mysql` project. For more information, see Build a Web app on Linux with Azure database for MySQL repo for more details.

```yml
variables:
  ARM_PASS: $(adminPass)
```

```yaml
trigger:
- none

pool:
  vmImage: 'ubuntu-latest'

steps:
- task: CopyFiles@2
  inputs:
    SourceFolder: 'quickstarts/microsoft.web/webapp-linux-managed-mysql/'
    Contents: '**'
    TargetFolder: '$(Build.ArtifactStagingDirectory)'
```

10. Add and configure the **Azure Resource Group Deployment** task.

    The task references both the artifact you built with the Copy Files task and your pipeline variables. Set these values when configuring your task.

    - **Deployment scope (deploymentScope)**: Set the deployment scope to `Resource Group`. You can target your deployment to a management group, an Azure subscription, or a resource group.
    - **Azure Resource Manager connection (azureResourceManagerConnection)**: Select your Azure Resource Manager service connection. To configure new service connection, select the Azure subscription from the list and click **Authorize**. See Connect to Microsoft Azure for more details
    - **Subscription (subscriptionId)**: Select the subscription where the deployment should go.
    - **Action (action)**: Set to `Create or update resource group` to create a new resource group or to update an existing one.
    - **Resource group**: Set to `ARMPipelinesLAMP-rg` to name your new resource group. If this is an existing resource group, it will be updated.
    - **Location(location)**: Location for deploying the resource group. Set to your closest location (for example, West US). If the resource group already exists in your subscription, this value will be ignored.
    - **Template location (templateLocation)**: Set to `Linked artifact`. This is location of your template and the parameters files.
    - **Template (csmFile)**: Set to `$(Build.ArtifactStagingDirectory)/azuredeploy.json`. This is the path to the ARM template.
    - **Template parameters (csmParametersFile)**: Set to `$(Build.ArtifactStagingDirectory)/azuredeploy.parameters.json`. This is the path to the parameters file for your ARM template.

- **Override template parameters (overrideParameters)**: Set to `-siteName $(siteName) -administratorLogin $(adminUser) -administratorLoginPassword $(ARM_PASS)` to use the variables you created earlier. These values will replace the parameters set in your template parameters file.
- **Deployment mode (deploymentMode)**: The way resources should be deployed. Set to `Incremental`. Incremental keeps resources that are not in the ARM template and is faster than `Complete`. `Validate` mode lets you find problems with the template before deploying.

yml

```yml
variables:
  ARM_PASS: $(adminPass)

trigger:
- none

pool:
  vmImage: 'ubuntu-latest'

steps:
- task: CopyFiles@2
  inputs:
    SourceFolder: 'quickstarts/microsoft.web/webapp-linux-managed-mysql/'
    Contents: '**'
    TargetFolder: '$(Build.ArtifactStagingDirectory)'

- task: AzureResourceManagerTemplateDeployment@3
  inputs:
    deploymentScope: 'Resource Group'
    azureResourceManagerConnection: '<your-resource-manager-connection>'
    subscriptionId: '<your-subscription-id>'
    action: 'Create Or Update Resource Group'
    resourceGroupName: 'ARMPipelinesLAMP-rg'
    location: '<your-closest-location>'
    templateLocation: 'Linked artifact'
    csmFile: '$(Build.ArtifactStagingDirectory)/azuredeploy.json'
    csmParametersFile: '$(Build.ArtifactStagingDirectory)/azuredeploy.parameters.json'
    overrideParameters: '-siteName $(siteName) -administratorLogin $(adminUser) -administratorLoginPassword $(ARM_PASS)'
    deploymentMode: 'Incremental'
```

11. Click **Save and run** to deploy your template. The pipeline job will be launched and after few minutes, depending on your agent, the job status should indicate `Success`.

Review deployed resources

JSON

1. Verify that the resources deployed. Go to the `ARMPipelinesLAMP-rg` resource group in the Azure portal and verify that you see App Service, App Service Plan, and Azure Database for MySQL server resources.

Name	Type
armpipelinetestsite	App Service
hpn-jgc4v36g4nflm	App Service plan
mysql-jgc4v36g4nflm	Azure Database for MySQL server

   You can also verify the resources using Azure CLI.

   Azure CLI

```
az resource list --resource-group ARMPipelinesLAMP-rg --output
table
```

2. Go to your new site. If you set `siteName` to `armpipelinetestsite`, the site is located at `https://armpipelinetestsite.azurewebsites.net/`.

Clean up resources

JSON

You can also use an ARM template to delete resources. Change the `action` value in your **Azure Resource Group Deployment** task to `DeleteRG`. You can also remove the inputs for `templateLocation`, `csmFile`, `csmParametersFile`, `overrideParameters`, and `deploymentMode`.

yml

```yml
variables:
  ARM_PASS: $(adminPass)
```

```yaml
trigger:
- none

pool:
  vmImage: 'ubuntu-latest'

steps:
- task: CopyFiles@2
  inputs:
    SourceFolder: 'quickstarts/microsoft.web/webapp-linux-managed-
mysql/'
    Contents: '**'
    TargetFolder: '$(Build.ArtifactStagingDirectory)'

- task: AzureResourceManagerTemplateDeployment@3
  inputs:
    deploymentScope: 'Resource Group'
    azureResourceManagerConnection: '<your-resource-manager-connection>'
    subscriptionId: '<your-subscription-id>'
    action: 'DeleteRG'
    resourceGroupName: 'ARMPipelinesLAMP-rg'
    location: ''<your-closest-location>'
```

Next steps

Create your first ARM template

Deploy a FHIR service within Azure Health Data Services - using ARM template

Article • 04/13/2023

In this article, you'll learn how to deploy FHIR service within the Azure Health Data Services (hereby called FHIR service) using the Azure Resource Manager template (ARM template). We provide you two options using PowerShell or using CLI.

An ARM template is a JSON file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

Prerequisites

PowerShell

- An Azure account with an active subscription. Create one for free.
- If you want to run the code locally:
  - Azure PowerShell.

Review the ARM template

The template used in this article is from Azure Quickstart Templates.

The template defines three Azure resources:

- Microsoft.HealthcareApis/workspaces
- Microsoft.HealthcareApis/workspaces/fhirservices
- Microsoft.Storage/storageAccounts

You can deploy the FHIR service resource by **removing** the workspaces resource, the storage resource, and the `dependsOn` property in the "Microsoft.HealthcareApis/workspaces/fhirservices" resource.

JSON

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "authorityurl": {
            "type": "string",
            "defaultValue": "https://login.microsoftonline.com"
        },
        "tagName": {
            "type": "string",
            "defaultValue": "My Deployment"
        },
        "region": {
            "type": "string",
                  "allowedValues": [
                "australiaeast",
                "canadacentral",
                "eastus",
                "eastus2",
                "germanywestcentral",
                "japaneast",
                "northcentralus",
                "northeurope",
                "southafricanorth",
                "southcentralus",
                "southeastasia",
                "switzerlandnorth",
                "uksouth",
                "ukwest",
                "westcentralus",
                "westeurope",
                "westus2"
            ]
        },
        "workspaceName": {
            "type": "string"
        },
        "fhirServiceName": {
            "type": "string"
        },
        "tenantid": {
            "type": "string"
        },
        "storageAccountName": {
            "type": "string"
        },
        "storageAccountConfirm": {
            "type": "bool",
            "defaultValue": true
        },
        "AccessPolicies": {
            "type": "array",
            "defaultValue": []
```

```
        },
        "smartProxyEnabled": {
            "type": "bool",
            "defaultValue": false
        }
    },
    "variables": {
        "authority": "[Concat(parameters('authorityurl'), '/',
parameters('tenantid'))]",
        "createManagedIdentity": true,
        "managedIdentityType": {
            "type": "SystemAssigned"
        },
        "storageBlobDataContributerRoleId": "[concat('/subscriptions/',
subscription().subscriptionId,
'/providers/Microsoft.Authorization/roleDefinitions/', 'ba92f5b4-2d11-453d-
a403-e96b0029c9fe')]"
    },
    "resources": [
        {
            "type": "Microsoft.HealthcareApis/workspaces",
            "name": "[parameters('workspaceName')]",
            "apiVersion": "2020-11-01-preview",
            "location": "[parameters('region')]",
            "properties": {}
        },
       {
            "type": "Microsoft.HealthcareApis/workspaces/fhirservices",
            "kind": "fhir-R4",
            "name": "[concat(parameters('workspaceName'), '/',
parameters('fhirServiceName'))]",
            "apiVersion": "2020-11-01-preview",
            "location": "[parameters('region')]",
            "dependsOn": [
                "[resourceId('Microsoft.HealthcareApis/workspaces',
parameters('workspaceName'))]"
            ],
            "tags": {
                "environmentName": "[parameters('tagName')]"
            },
            "properties": {
                "accessPolicies": "[parameters('AccessPolicies')]",
                "authenticationConfiguration": {
                    "authority": "[variables('Authority')]",
                    "audience": "[concat('https//',
parameters('workspaceName'), '-', parameters('fhirServiceName'),
'.fhir.azurehealthcareapis.com')]",
                    "smartProxyEnabled": "[parameters('smartProxyEnabled')]"
                },
                "corsConfiguration": {
                    "allowCredentials": false,
                    "headers": ["*"],
                    "maxAge": 1440,
                    "methods": ["DELETE", "GET", "OPTIONS", "PATCH", "POST",
"PUT"],
```

```
                    "origins": ["https://localhost:6001"]
                },
                "exportConfiguration": {
                    "storageAccountName": "
[parameters('storageAccountName')]"
                }
            },
            "identity": "[if(variables('createManagedIdentity'),
variables('managedIdentityType'), json('null'))]"
        },
        {
            "name": "[parameters('storageAccountName')]",
            "type": "Microsoft.Storage/storageAccounts",
            "apiVersion": "2019-06-01",
            "location": "[resourceGroup().location]",
            "properties": {
                "supportsHttpsTrafficOnly": "true"
            },
            "condition": "[parameters('storageAccountConfirm')]",
            "dependsOn": [
                "[parameters('fhirServiceName')]"
            ],
            "sku": {
                "name": "Standard_LRS"
            },
            "kind": "Storage",
            "tags": {
                "environmentName": "[parameters('tagName')]",
                "test-account-rg": "true"
            }
        }
    ],
    "outputs": {
    }
}
```

Deploy ARM template

You can deploy the ARM template using two options: PowerShell or CLI.

The sample code provided below uses the template in the “templates” subfolder of the subfolder “src”. You may want to change the location path to reference the template file properly.

The deployment process takes a few minutes to complete. Take a note of the names for the FHIR service and the resource group, which you'll use later.

PowerShell

Deploy the template: using PowerShell

Run the code in PowerShell locally, in Visual Studio Code, or in Azure Cloud Shell, to deploy the FHIR service.

If you haven't logged in to Azure, use "Connect-AzAccount" to log in. Once you've logged in, use "Get-AzContext" to verify the subscription and tenant you want to use. You can change the subscription and tenant if needed.

You can create a new resource group, or use an existing one by skipping the step or commenting out the line starting with “New-AzResourceGroup”.

PowerShell

```
### variables
$resourcegroupname="your resource group"
$location="South Central US"
$workspacename="your workspace name"
$servicename="your fhir service name"
$tenantid="xxx"
$subscriptionid="xxx"
$storageaccountname="storage account name"
$storageaccountconfirm=1

### login to azure
Connect-AzAccount
#Connect-AzAccount SubscriptionId $subscriptionid
Set-AzContext -Subscription $subscriptionid
Connect-AzAccount -Tenant $tenantid -SubscriptionId $subscriptionid
#Get-AzContext

### create resource group
New-AzResourceGroup -Name $resourcegroupname -Location $location

### deploy the resource
New-AzResourceGroupDeployment -ResourceGroupName $resourcegroupname -
TemplateFile "src/templates/fhirtemplate.json" -region $location -
workspaceName $workspacename -fhirServiceName $fhirservicename -tenantid
$tenantid -storageAccountName $storageaccountname -storageAccountConfirm
$storageaccountconfirm
```

Review the deployed resources

You can verify that the FHIR service is up and running by opening the browser and navigating to `https://<yourfhir servic>.azurehealthcareapis.com/metadata`. If the capability statement is displayed or downloaded automatically, your deployment is successful.

Clean up the resources

When the resource is no longer needed, run the code below to delete the resource group.

PowerShell

PowerShell

```
$resourceGroupName = “your resource group name”
Remove-AzResourceGroup -Name $resourceGroupName
```

Next steps

In this quickstart guide, you've deployed the FHIR service within Azure Health Data Services using an ARM template. For more information about FHIR service supported features, see.

Supported FHIR Features

FHIR® is a registered trademark of HL7 and is used with the permission of HL7.

Quickstart: Route Blob storage events to web endpoint by using an ARM template

Article • 04/13/2023

Azure Event Grid is an eventing service for the cloud. In this article, you use an Azure Resource Manager template (ARM template) to create a Blob storage account, subscribe to events for that blob storage, and trigger an event to view the result. Typically, you send events to an endpoint that processes the event data and takes actions. However, to simplify this article, you send the events to a web app that collects and displays the messages.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Create a message endpoint

Before subscribing to the events for the Blob storage, let's create the endpoint for the event message. Typically, the endpoint takes actions based on the event data. To simplify this quickstart, you deploy a pre-built web app that displays the event messages. The deployed solution includes an App Service plan, an App Service web app, and source code from GitHub.

1. Select **Deploy to Azure** to deploy the solution to your subscription. In the Azure portal, provide values for the parameters.

   Deploy to Azure

2. The deployment may take a few minutes to complete. After the deployment has succeeded, view your web app to make sure it's running. In a web browser, navigate to: `https://<your-site-name>.azurewebsites.net`

3. You see the site but no events have been posted to it yet.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "5934802899125195053"
    }
  },
  "parameters": {
    "storageAccountName": {
      "type": "string",
      "defaultValue": "[format('storage{0}', uniqueString(resourceGroup().id))]",
      "metadata": {
        "description": "Provide a unique name for the Blob Storage account."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Provide a location for the Blob Storage account that supports Event Grid."
      }
```

```
    },
    "eventSubName": {
      "type": "string",
      "defaultValue": "subToStorage",
      "metadata": {
        "description": "Provide a name for the Event Grid subscription."
      }
    },
    "endpoint": {
      "type": "string",
      "metadata": {
        "description": "Provide the URL for the WebHook to receive events.
Create your own endpoint for events."
      }
    },
    "systemTopicName": {
      "type": "string",
      "defaultValue": "mystoragesystemtopic",
      "metadata": {
        "description": "Provide a name for the system topic."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-08-01",
      "name": "[parameters('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "StorageV2",
      "properties": {
        "accessTier": "Hot"
      }
    },
    {
      "type": "Microsoft.EventGrid/systemTopics",
      "apiVersion": "2021-12-01",
      "name": "[parameters('systemTopicName')]",
      "location": "[parameters('location')]",
      "properties": {
        "source": "[resourceId('Microsoft.Storage/storageAccounts',
parameters('storageAccountName'))]",
        "topicType": "Microsoft.Storage.StorageAccounts"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Storage/storageAccounts',
parameters('storageAccountName'))]"
      ]
    },
    {
      "type": "Microsoft.EventGrid/systemTopics/eventSubscriptions",
      "apiVersion": "2021-12-01",
```

```
      "name": "[format('{0}/{1}', parameters('systemTopicName'),
parameters('eventSubName'))]",
      "properties": {
        "destination": {
          "properties": {
            "endpointUrl": "[parameters('endpoint')]"
          },
          "endpointType": "WebHook"
        },
        "filter": {
          "includedEventTypes": [
            "Microsoft.Storage.BlobCreated",
            "Microsoft.Storage.BlobDeleted"
          ]
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.EventGrid/systemTopics',
parameters('systemTopicName'))]"
      ]
    }
  ]
}
```

Two Azure resources are defined in the template:

- **Microsoft.Storage/storageAccounts**: create an Azure Storage account.
- **Microsoft.EventGrid/systemTopics**: create a system topic with the specified name for the storage account.
- **Microsoft.EventGrid/systemTopics/eventSubscriptions**: create an Azure Event Grid subscription for the system topic.

Deploy the template

1. Select the following link to sign in to Azure and open a template. The template creates a key vault and a secret.

   Deploy to Azure

2. Specify the **endpoint**: provide the URL of your web app and add `api/updates` to the home page URL.

3. Select **Purchase** to deploy the template.

The Azure portal is used here to deploy the template. You can also use the Azure PowerShell, Azure CLI, and REST API. To learn other deployment methods, see Deploy templates.

> **Note**
>
> You can find more Azure Event Grid template samples here.

Validate the deployment

View your web app again, and notice that a subscription validation event has been sent to it. Select the eye icon to expand the event data. Event Grid sends the validation event so the endpoint can verify that it wants to receive event data. The web app includes code to validate the subscription.

Now, let's trigger an event to see how Event Grid distributes the message to your endpoint.

You trigger an event for the Blob storage by uploading a file. The file doesn't need any specific content. The articles assumes you have a file named testfile.txt, but you can use any file.

When you upload the file to the Azure Blob storage, Event Grid sends a message to the endpoint you configured when subscribing. The message is in the JSON format and it

contains an array with one or more events. In the following example, the JSON message contains an array with one event. View your web app and notice that a blob created event was received.

```
{
  "topic": "/subscriptions/{subscription-id}/resourceGroups/gridGroup/providers/Microsoft.Storage/storageAccounts/tfstore0524demo",
  "subject": "/blobServices/default/containers/testcontainer/blobs/testfile.txt",
  "eventType": "Microsoft.Storage.BlobCreated",
  "eventTime": "2018-05-24T19:31:26.4482862Z",
  "id": "274133e6-801e-0010-0795-f352e606b022",
  "data": {
    "api": "PutBlob",
    "clientRequestId": "0c119d90-5f89-11e8-ba2a-0a580af49105",
    "requestId": "274133e6-801e-0010-0795-f352e6000000",
    "eTag": "0x8D5C1ACF0FDA22E",
    "contentType": "text/plain",
    "contentLength": 0,
    "blobType": "BlockBlob",
    "url": "https://tfstore0524demo.blob.core.windows.net/testcontainer/testfile.txt",
    "sequencer": "000000000000000000000000000026600000000000011d8",
    "storageDiagnostics": {
      "batchId": "95e759a5-e334-4584-b7fd-3ec32abaacc1"
    }
  },
  "dataVersion": "",
  "metadataVersion": "1"
}
```


Clean up resources

When no longer needed, delete the resource group.

Next steps

For more information about Azure Resource Manager templates, see the following articles:

- Azure Resource Manager documentation
- Define resources in Azure Resource Manager templates
- Azure Quickstart templates
- Azure Event Grid templates.

Quickstart: Create and deploy a Consumption logic app workflow in multi-tenant Azure Logic Apps with an ARM template

Article • 03/10/2023

Applies to: **Azure Logic Apps (Consumption)**

Azure Logic Apps is a cloud service that helps you create and run automated workflows that integrate data, apps, cloud-based services, and on-premises systems by choosing from hundreds of connectors. This quickstart focuses on the process for deploying an Azure Resource Manager template (ARM template) to create a basic Consumption logic app workflow that checks the status for Azure on an hourly schedule and runs in multi-tenant Azure Logic Apps.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites, and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free Azure account before you start.

Review the template

This quickstart uses the **Create a logic app** template, which you can find in the Azure Quickstart Templates Gallery but is too long to show here. Instead, you can review the quickstart template's "azuredeploy.json file" in the templates gallery.

The quickstart template creates a Consumption logic app workflow that uses the *built-in* Recurrence trigger, which is set to run every hour, and a built-in HTTP action, which calls

a URL that returns the status for Azure. Built-in operations run natively on Azure Logic Apps platform.

This template creates the following Azure resource:

- **Microsoft.Logic/workflows**, which creates the workflow for a Consumption logic app resource.

To find more quickstart templates for Azure Logic Apps, review the Microsoft.Logic templates in the gallery.

Deploy the template

Follow the option that you want to use for deploying the quickstart template:

Option	Description
Azure portal	If your Azure environment meets the prerequisites, and you're familiar with using ARM templates, these steps help you sign in directly to Azure and open the quickstart template in the Azure portal. For more information, see Deploy resources with ARM templates and Azure portal.
Azure CLI	The Azure CLI provides a command-line experience for creating and managing Azure resources. To run these commands, you need Azure CLI version 2.6 or later. To check your CLI version, enter **az --version**. For more information, see the following documentation: - What is Azure CLI - Get started with Azure CLI
Azure PowerShell	Azure PowerShell provides a set of cmdlets that use the Azure Resource Manager model for managing your Azure resources. For more information, see the following documentation: - Azure PowerShell Overview - Introducing the Azure PowerShell Az module - Get started with Azure PowerShell
Azure Resource Management REST API	Azure provides Representational State Transfer (REST) APIs, which are service endpoints that support HTTP operations (methods) that you use to create, retrieve, update, or delete access to service resources. For more information, see Get started with Azure REST API.

Portal

1. To sign in with your Azure account and open the quickstart template in the Azure portal, select the following image:

   Deploy to Azure

2. In the portal, on the **Create a logic app using a template** page, enter or select the following values:

Property	Value	Description
Subscription	*<Azure-subscription-name>*	The name for the Azure subscription to use
Resource group	*<Azure-resource-group-name>*	The name for a new or existing Azure resource group. This example uses **Check-Azure-Status-RG**.
Region	*<Azure-region>*	The Azure datacenter region to use your logic app. This example uses **West US**.
Logic App Name	*<logic-app-name>*	The name to use for your logic app. This example uses **Check-Azure-Status-LA**.
Test Uri	*<test-URI>*	The URI for the service to call based on a specific schedule. This example uses https://azure.status.microsoft/en-us/status/, which is the Azure status page.
Location	*<Azure-region-for-all-resources>*	The Azure region to use for all resources, if different from the default value. This example uses the default value, **[resourceGroup().location]**, which is the resource group location.

   The following example shows how the page looks with sample values:

3. When you're done, select **Review + create**.

4. Continue with the steps in Review deployed resources.

Review deployed resources

To view the logic app workflow, you can use the Azure portal, run a script that you create with Azure CLI or Azure PowerShell, or use the Logic App REST API.

Portal

1. In the Azure portal search box, enter your logic app's name, which is **Check-Azure-Status-LA** in this example. From the results list, select your logic app.

2. In the Azure portal, find and select your logic app, which is **Check-Azure-Status-RG** in this example.

3. When the workflow designer opens, review the logic app workflow created by the quickstart template.

4. To test the logic app, on the designer toolbar, select **Run**.

Clean up resources

If you plan to continue working with subsequent quickstarts and tutorials, you might want to keep these resources. When you no longer need the logic app, delete the resource group by using either the Azure portal, Azure CLI, Azure PowerShell, or Resource Management REST API.

Portal

1. In the Azure portal, find and select the resource group that you want to delete, which is **Check-Azure-Status-RG** in this example.
2. On the resource group menu, select **Overview** if not already selected. On the overview page, select **Delete resource group**.
3. To confirm, enter the name for the resource group.

For more information, see Delete resource group.

Next steps

Tutorial: Create and deploy your first ARM template

Quickstart: Create a Service Bus namespace and a queue using an ARM template

Article • 04/13/2023

This article shows how to use an Azure Resource Manager template (ARM template) that creates a Service Bus namespace and a queue within that namespace. The article explains how to specify which resources are deployed and how to define parameters that are specified when the deployment is executed. You can use this template for your own deployments, or customize it to meet your requirements.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.9.1.41621",
      "templateHash": "7245898084363028 89"
```

```
    }
  },
  "parameters": {
    "serviceBusNamespaceName": {
      "type": "string",
      "metadata": {
        "description": "Name of the Service Bus namespace"
      }
    },
    "serviceBusQueueName": {
      "type": "string",
      "metadata": {
        "description": "Name of the Queue"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.ServiceBus/namespaces",
      "apiVersion": "2022-01-01-preview",
      "name": "[parameters('serviceBusNamespaceName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard"
      },
      "properties": {}
    },
    {
      "type": "Microsoft.ServiceBus/namespaces/queues",
      "apiVersion": "2022-01-01-preview",
      "name": "[format('{0}/{1}', parameters('serviceBusNamespaceName'),
parameters('serviceBusQueueName'))]",
      "properties": {
        "lockDuration": "PT5M",
        "maxSizeInMegabytes": 1024,
        "requiresDuplicateDetection": false,
        "requiresSession": false,
        "defaultMessageTimeToLive": "P10675199DT2H48M5.4775807S",
        "deadLetteringOnMessageExpiration": false,
        "duplicateDetectionHistoryTimeWindow": "PT10M",
        "maxDeliveryCount": 10,
        "autoDeleteOnIdle": "P10675199DT2H48M5.4775807S",
        "enablePartitioning": false,
        "enableExpress": false
      },
      "dependsOn": [
        "[resourceId('Microsoft.ServiceBus/namespaces',
parameters('serviceBusNamespaceName'))]"
```

```
        ]
      }
    ]
  }
```

The resources defined in the template include:

- Microsoft.ServiceBus/namespaces
- Microsoft.ServiceBus/namespaces/queues

> **Note**
>
> The following ARM templates are available for download and deployment.
>
> - Create a Service Bus namespace with queue and authorization rule
> - Create a Service Bus namespace with topic and subscription
> - Create a Service Bus namespace
> - Create a Service Bus namespace with topic, subscription, and rule

You can find more template from Azure Quickstart Templates

Deploy the template

With this template, you deploy a Service Bus namespace with a queue.

Service Bus queues offer First In, First Out (FIFO) message delivery to one or more competing consumers.

To run the deployment automatically, click the following button: Create a new resource group for the deployment so that you can easily clean up later.

Deploy to Azure

Validate the deployment

1. Select **Notifications** at the top to see the status of the deployment. Wait until the deployment succeeds. Then, select **Go to resource group** in the notification message to navigate to the page for the resource group that contains the Service Bus namespace.

2. Confirm that you see your Service Bus namespace in the list of resources.

3. Select the namespace from the list to see the **Service Bus Namespace** page.

Clean up resources

1. In the Azure portal, navigate to the **Resource group** page for your resource group.

2. Select **Delete resource group** from the toolbar.

3. Type the name of the resource group, and select **Delete**.

Next steps

See the following topic that shows how to create an authorization rule for the namespace/queue:

Create a Service Bus authorization rule for namespace and queue using an ARM template

Learn how to manage these resources by viewing these articles:

- Manage Service Bus with PowerShell
- Manage Service Bus resources with the Service Bus Explorer

Run Azure IoT Edge on Ubuntu Virtual Machines

Article • 04/13/2023

Applies to: IoT Edge 1.4

The Azure IoT Edge runtime is what turns a device into an IoT Edge device. The runtime can be deployed on devices as small as a Raspberry Pi or as large as an industrial server. Once a device is configured with the IoT Edge runtime, you can start deploying business logic to it from the cloud.

To learn more about how the IoT Edge runtime works and what components are included, see Understand the Azure IoT Edge runtime and its architecture.

This article lists the steps to deploy an Ubuntu 20.04 LTS virtual machine with the Azure IoT Edge runtime installed and configured using a pre-supplied device connection string. The deployment is accomplished using a cloud-init based Azure Resource Manager template maintained in the iotedge-vm-deploy project repository.

On first boot, the virtual machine installs the latest version of the Azure IoT Edge runtime via cloud-init. It also sets a supplied connection string before the runtime starts, allowing you to easily configure and connect the IoT Edge device without the need to start an SSH or remote desktop session.

Deploy using Deploy to Azure Button

The Deploy to Azure Button allows for streamlined deployment of Azure Resource Manager templates maintained on GitHub. This section will demonstrate usage of the Deploy to Azure Button contained in the iotedge-vm-deploy project repository.

1. We will deploy an Azure IoT Edge enabled Linux VM using the iotedge-vm-deploy Azure Resource Manager template. To begin, click the button below:

   Deploy to Azure

2. On the newly launched window, fill in the available form fields:

Field	Description
Subscription	The active Azure subscription to deploy the virtual machine into.
Resource group	An existing or newly created Resource Group to contain the virtual machine and it's associated resources.
Region	The geographic region to deploy the virtual machine into, this value defaults to the location of the selected Resource Group.
DNS Label Prefix	A required value of your choosing that is used to prefix the hostname of the virtual machine.
Admin Username	A username, which will be provided root privileges on deployment.
Device Connection String	A device connection string for a device that was created within your intended IoT Hub.
VM Size	The size of the virtual machine to be deployed.

Field	Description
Ubuntu OS Version	The version of the Ubuntu OS to be installed on the base virtual machine.
Authentication Type	Choose **sshPublicKey** or **password** depending on your preference.
Admin Password or Key	The value of the SSH Public Key or the value of the password depending on the choice of Authentication Type.

When all fields have been filled in, click the button at the bottom to move to `Next : Review + create` where you can review the terms and click **Create** to begin the deployment.

3. Verify that the deployment has completed successfully. A virtual machine resource should have been deployed into the selected resource group. Take note of the machine name, this should be in the format `vm-0000000000000`. Also, take note of the associated **DNS Name**, which should be in the format `<dnsLabelPrefix>`.`<location>`.cloudapp.azure.com.

   The **DNS Name** can be obtained from the **Overview** section of the newly deployed virtual machine within the Azure portal.

4. If you want to SSH into this VM after setup, use the associated **DNS Name** with the command: `ssh <adminUsername>@<DNS_Name>`

Deploy from Azure CLI

1. Ensure that you have installed the Azure CLI iot extension with:

   Azure CLI

   ```
   az extension add --name azure-iot
   ```

2. Next, if you're using Azure CLI on your desktop, start by logging in:

Azure CLI

```
az login
```

3. If you have multiple subscriptions, select the subscription you'd like to use:

   a. List your subscriptions:

   Azure CLI

   ```
   az account list --output table
   ```

   b. Copy the SubscriptionID field for the subscription you'd like to use.

   c. Set your working subscription with the ID that you copied:

   Azure CLI

   ```
   az account set -s <SubscriptionId>
   ```

4. Create a new resource group (or specify an existing one in the next steps):

   Azure CLI

   ```
   az group create --name IoTEdgeResources --location westus2
   ```

5. Create a new virtual machine:

   To use an **authenticationType** of `password`, see the example below:

   Azure CLI

   ```
   az deployment group create \
   --resource-group IoTEdgeResources \
   --template-uri "https://raw.githubusercontent.com/Azure/iotedge-vm-deploy/1.4/edgeDeploy.json" \
   --parameters dnsLabelPrefix='my-edge-vm1' \
   --parameters adminUsername='<REPLACE_WITH_USERNAME>' \
   --parameters deviceConnectionString=$(az iot hub device-identity connection-string show --device-id <REPLACE_WITH_DEVICE-NAME> --hub-name <REPLACE-WITH-HUB-NAME> -o tsv) \
   --parameters authenticationType='password' \
   --parameters adminPasswordOrKey="<REPLACE_WITH_SECRET_PASSWORD>"
   ```

   To authenticate with an SSH key, you may do so by specifying an **authenticationType** of `sshPublicKey`, then provide the value of the SSH key in the

adminPasswordOrKey parameter. An example is shown below.

Azure CLI

```
#Generate the SSH Key
ssh-keygen -m PEM -t rsa -b 4096 -q -f ~/.ssh/iotedge-vm-key -N ""

#Create a VM using the iotedge-vm-deploy script
az deployment group create \
--resource-group IoTEdgeResources \
--template-uri "https://raw.githubusercontent.com/Azure/iotedge-vm-deploy/1.4/edgeDeploy.json" \
--parameters dnsLabelPrefix='my-edge-vm1' \
--parameters adminUsername='<REPLACE_WITH_USERNAME>' \
--parameters deviceConnectionString=$(az iot hub device-identity connection-string show --device-id <REPLACE_WITH_DEVICE-NAME> --hub-name <REPLACE-WITH-HUB-NAME> -o tsv) \
--parameters authenticationType='sshPublicKey' \
--parameters adminPasswordOrKey="$(< ~/.ssh/iotedge-vm-key.pub)"
```

6. Verify that the deployment has completed successfully. A virtual machine resource should have been deployed into the selected resource group. Take note of the machine name, this should be in the format `vm-0000000000000`. Also, take note of the associated **DNS Name**, which should be in the format `<dnsLabelPrefix>`.`<location>`.cloudapp.azure.com.

   The **DNS Name** can be obtained from the JSON-formatted output of the previous step, within the **outputs** section as part of the **public SSH** entry. The value of this entry can be used to SSH into to the newly deployed machine.

   Bash

   ```
   "outputs": {
     "public SSH": {
       "type": "String",
       "value": "ssh <adminUsername>@<DNS_Name>"
     }
   }
   ```

   The **DNS Name** can also be obtained from the **Overview** section of the newly deployed virtual machine within the Azure portal.

7. If you want to SSH into this VM after setup, use the associated **DNS Name** with the command: `ssh <adminUsername>@<DNS_Name>`

Next steps

Now that you have an IoT Edge device provisioned with the runtime installed, you can deploy IoT Edge modules.

If you are having problems with the IoT Edge runtime installing properly, check out the troubleshooting page.

To update an existing installation to the newest version of IoT Edge, see Update the IoT Edge security daemon and runtime.

If you'd like to open up ports to access the VM through SSH or other inbound connections, refer to the Azure Virtual Machines documentation on opening up ports and endpoints to a Linux VM

Quickstart: Deploy an Azure IoT Hub and a storage account using an ARM template

Article • 03/16/2023

In this quickstart, you use an Azure Resource Manager template (ARM template) to create an IoT Hub that will route messages to Azure Storage, and a storage account to hold the messages. After manually adding a virtual IoT device to the hub to submit the messages, you configure that connection information in an application called *arm-read-write* to submit messages from the device to the hub. The hub is configured so the messages sent to the hub are automatically routed to the storage account. At the end of this quickstart, you can open the storage account and see the messages sent.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

- If you don't have an Azure subscription, create a free Azure account before you begin.
- The sample application you run in this quickstart is written using C#. You need the .NET SDK 6.0 or greater on your development machine.

  You can download the .NET Core SDK for multiple platforms from .NET.

  You can verify the current version of C# on your development machine using the following command:

  cmd/sh

  ```
  dotnet --version
  ```

- Download and unzip the IoT C# SDK.

Review the template

The template used in this quickstart is called `101-iothub-auto-route-messages` from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.6.18.56646",
      "templateHash": "9942060226287533039"
    }
  },
  "parameters": {
    "projectName": {
      "type": "string",
      "defaultValue": "contoso",
      "maxLength": 11,
      "minLength": 1,
      "metadata": {
        "description": "Define the project name or prefix for all objects."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "The datacenter to use for the deployment."
      }
    },
    "skuName": {
      "type": "string",
      "defaultValue": "S1",
      "metadata": {
        "description": "The SKU to use for the IoT Hub."
      }
    },
    "skuUnits": {
      "type": "int",
      "defaultValue": 1,
      "metadata": {
        "description": "The number of IoT Hub units."
      }
    },
```

```
    "d2cPartitions": {
      "type": "int",
      "defaultValue": 4,
      "metadata": {
        "description": "Partitions used for the event stream."
      }
    }
  },
  "variables": {
    "iotHubName": "[format('{0}Hub{1}', parameters('projectName'),
uniqueString(resourceGroup().id))]",
    "storageAccountName": "[format('{0}{1}',
toLower(parameters('projectName')), uniqueString(resourceGroup().id))]",
    "storageEndpoint": "[format('{0}StorageEndpont',
parameters('projectName'))]",
    "storageContainerName": "[format('{0}results',
toLower(parameters('projectName')))]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-08-01",
      "name": "[variables('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "Storage"
    },
    {
      "type": "Microsoft.Storage/storageAccounts/blobServices/containers",
      "apiVersion": "2021-08-01",
      "name": "[format('{0}/default/{1}', variables('storageAccountName'),
variables('storageContainerName'))]",
      "properties": {
        "publicAccess": "None"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))]"
      ]
    },
    {
      "type": "Microsoft.Devices/IotHubs",
      "apiVersion": "2021-07-02",
      "name": "[variables('iotHubName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('skuName')]",
        "capacity": "[parameters('skuUnits')]"
      },
      "properties": {
        "eventHubEndpoints": {
          "events": {
            "retentionTimeInDays": 1,
```

```
            "partitionCount": "[parameters('d2cPartitions')]"
          }
        },
        "routing": {
          "endpoints": {
            "storageContainers": [
              {
                "connectionString": "
[format('DefaultEndpointsProtocol=https;AccountName={0};EndpointSuffix=
{1};AccountKey={2}', variables('storageAccountName'),
environment().suffixes.storage,
listKeys(resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName')), '2021-08-01').keys[0].value)]",
                "containerName": "[variables('storageContainerName')]",
                "fileNameFormat": "
{iothub}/{partition}/{YYYY}/{MM}/{DD}/{HH}/{mm}",
                "batchFrequencyInSeconds": 100,
                "maxChunkSizeInBytes": 104857600,
                "encoding": "JSON",
                "name": "[variables('storageEndpoint')]"
              }
            ]
          },
          "routes": [
            {
              "name": "ContosoStorageRoute",
              "source": "DeviceMessages",
              "condition": "level=\"storage\"",
              "endpointNames": [
                "[variables('storageEndpoint')]"
              ],
              "isEnabled": true
            }
          ],
          "fallbackRoute": {
            "name": "$fallback",
            "source": "DeviceMessages",
            "condition": "true",
            "endpointNames": [
              "events"
            ],
            "isEnabled": true
          }
        },
        "messagingEndpoints": {
          "fileNotifications": {
            "lockDurationAsIso8601": "PT1M",
            "ttlAsIso8601": "PT1H",
            "maxDeliveryCount": 10
          }
        },
        "enableFileUploadNotifications": false,
        "cloudToDevice": {
          "maxDeliveryCount": 10,
          "defaultTtlAsIso8601": "PT1H",
```

```
          "feedback": {
            "lockDurationAsIso8601": "PT1M",
            "ttlAsIso8601": "PT1H",
            "maxDeliveryCount": 10
          }
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))]"
      ]
    }
  ]
}
```

Two Azure resources are defined in the template:

- Microsoft.Storage/storageAccounts
- Microsoft.Devices/IotHubs

Deploy the template and run the sample app

This section provides the steps to deploy the template, create a virtual device, and run the arm-read-write application to send the messages.

1. Create the resources by deploying the ARM template.

   > **Tip**
   >
   > Select the button below to start the deployment of the template. While it's running, set up the arm-read-write application to run.

   Deploy to Azure

2. Open a command window and go to the folder where you unzipped the IoT C# SDK. Find the folder with the arm-read-write.csproj file. You create the environment variables in this command window. Log into the Azure portal to get the keys. Select **Resource Groups** then select the resource group used for this quickstart.

3. You see the IoT Hub and storage account that were created when you deployed the ARM template. Wait until the template is fully deployed before continuing. Then select your resource group to see your resources.

4. You need the **hub name**. Select the hub in the list of resources. Copy the name of the hub from the top of the IoT Hub section to the Windows clipboard.

   Substitute the hub name in this command where noted, and execute this command in the command window:

```
Windows Command Prompt
```

```
SET IOT_HUB_URI=<hub name goes here>.azure-devices-net;
```

which will look this example:

Windows Command Prompt

```
SET IOT_HUB_URI=ContosoTestHubdlxlud5h.azure-devices-net;
```

5. The next environment variable is the IoT Device Key. Add a new device to the hub by selecting **Devices** from the IoT Hub menu for the hub.

6. On the right side of the screen, select + **Add Device** to add a new device.

   Fill in the new device name. This quickstart uses a name starting with **Contoso-Test-Device**. Save the device and then open that screen again to retrieve the device key. (The key is generated for you when you close the pane.) Select either the primary or secondary key and copy it to the Windows clipboard. In the command window, set the command to execute and then press **Enter**. The command should look like this one but with the device key pasted in:

Windows Command Prompt

```
SET IOT_DEVICE_KEY=<device-key-goes-here>
```

7. The last environment variable is the **Device ID**. In the command window, set up the command and execute it.

Windows Command Prompt

```
SET IOT_DEVICE_ID=<device-id-goes-here>
```

which will look like this example:

Windows Command Prompt

```
SET IOT_DEVICE_ID=Contoso-Test-Device
```

8. To see the environment variables you've defined, type SET on the command line and press **Enter**, then look for the ones starting with **IoT**.

```
IOT_DEVICE_ID=Contoso-Test-Device
IOT_DEVICE_KEY=RClD0LGxZCYav
IOT_HUB_URI=ContosoTestHubdlxlud5h.azure-devices-net
```

   Now the environment variables are set, run the application from the same command window. Because you're using the same window, the variables will be accessible in memory when you run the application.

9. To run the application, type the following command in the command window and press **Enter**.

   `dotnet run arm-read-write`

   The application generates and displays messages on the console as it sends each message to the IoT hub. The hub was configured in the ARM template to have automated routing. Messages containing the text `level = storage` are automatically routed to the storage account. Let the app run for 10 to 15 minutes, then press **Enter** once or twice until it stops running.

Review deployed resources

1. Log in to the Azure portal and select the Resource Group, then select the storage account.

2. Drill down into the storage account until you find files.

3. Select one of the files and select **Download** and download the file to a location you can find later. It will have a name that's numeric, like 47. Add *.txt* to the end and then double-click on the file to open it.

4. When you open the file, each row is for a different message; the body of each message is also encrypted. It must be in order for you to perform queries against the body of the message.

> **Note**
>
> These messages are encoded in UTF-32 and base64. If you read the message back, you have to decode it from base64 and utf-32 in order to read it as ASCII. If you're interested, you can use the method ReadOneRowFromFile in the Routing Tutorial to read one for from one of these message files and decode it into ASCII. ReadOneRowFromFile is in the IoT C# SDK repository that you unzipped for this quickstart. Here is the path from the top of that folder: *./iothub/device/samples/getting started/RoutingTutorial/SimulatedDevice/Program.cs*. Set the boolean `readTheFile` to true, and hardcode the path to the file on disk, and it will open and translate the first row in the file.

You have deployed an ARM template to create an IoT Hub and a storage account, and run a program to send messages to the hub. The messages are then automatically stored in the storage account where they can be viewed.

Clean up resources

To remove the resources added during this quickstart, log into the Azure portal. Select **Resource Groups**, then find the resource group you used for this quickstart. Select the resource group and then select *Delete*. It will delete all of the resources in the group.

Next steps

Tutorial: Create and deploy your first ARM template

Quickstart: Set up the IoT Hub Device Provisioning Service (DPS) with an ARM template

Article • 04/25/2023

You can use an Azure Resource Manager template (ARM template) to programmatically set up the Azure cloud resources necessary for provisioning your devices. These steps show how to create an IoT hub and a new IoT Hub Device Provisioning Service with an ARM template. The Iot Hub is also linked to the DPS resource using the template. This linking allows the DPS resource to assign devices to the hub based on allocation policies you configure.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

This quickstart uses Azure portal and the Azure CLI to perform the programmatic steps necessary to create a resource group and deploy the template. However, you can also use PowerShell, .NET, Ruby, or other programming languages to perform these steps and deploy your template.

If your environment meets the prerequisites, and you're already familiar with using ARM templates, selecting the **Deploy to Azure** button opens the template for deployment in the Azure portal.

Deploy to Azure

If you don't have an Azure subscription, create an Azure free account before you begin.

Prerequisites

- Use the Bash environment in Azure Cloud Shell. For more information, see Quickstart for Bash in Azure Cloud Shell.

  Launch Cloud Shell

- If you prefer to run CLI reference commands locally, install the Azure CLI. If you're running on Windows or macOS, consider running Azure CLI in a Docker container. For more information, see How to run the Azure CLI in a Docker container.
    - If you're using a local installation, sign in to the Azure CLI by using the az login command. To finish the authentication process, follow the steps displayed in your terminal. For other sign-in options, see Sign in with the Azure CLI.
    - When you're prompted, install the Azure CLI extension on first use. For more information about extensions, see Use extensions with the Azure CLI.
    - Run az version to find the version and dependent libraries that are installed. To upgrade to the latest version, run az upgrade.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

> **Note**
>
> Currently there is no ARM template support for creating enrollments with new DPS resources. This is a common and understood request that is being considered for implementation.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.12.40.16777",
      "templateHash": "13184692430416822033"
    }
  },
  "parameters": {
    "iotHubName": {
      "type": "string",
      "metadata": {
        "description": "Specify the name of the Iot hub."
      }
    },
    "provisioningServiceName": {
      "type": "string",
      "metadata": {
```

```
        "description": "Specify the name of the provisioning service."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Specify the location of the resources."
      }
    },
    "skuName": {
      "type": "string",
      "defaultValue": "S1",
      "metadata": {
        "description": "The SKU to use for the IoT Hub."
      }
    },
    "skuUnits": {
      "type": "int",
      "defaultValue": 1,
      "metadata": {
        "description": "The number of IoT Hub units."
      }
    }
  },
  "variables": {
    "iotHubKey": "iothubowner"
  },
  "resources": [
    {
      "type": "Microsoft.Devices/IotHubs",
      "apiVersion": "2021-07-02",
      "name": "[parameters('iotHubName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('skuName')]",
        "capacity": "[parameters('skuUnits')]"
      },
      "properties": {}
    },
    {
      "type": "Microsoft.Devices/provisioningServices",
      "apiVersion": "2022-02-05",
      "name": "[parameters('provisioningServiceName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('skuName')]",
        "capacity": "[parameters('skuUnits')]"
      },
      "properties": {
        "iotHubs": [
          {
            "connectionString": "[format('HostName={0};SharedAccessKeyName=
{1};SharedAccessKey={2}', reference(resourceId('Microsoft.Devices/IotHubs',
parameters('iotHubName')), '2021-07-02').hostName, variables('iotHubKey'),
```

```
listkeys(resourceId('Microsoft.Devices/IotHubs', parameters('iotHubName')),
'2021-07-02').value[0].primaryKey)]",
            "location": "[parameters('location')]"
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Devices/IotHubs',
parameters('iotHubName'))]"
      ]
    }
  ]
}
```

Two Azure resources are defined in the previous template:

- **Microsoft.Devices/IotHubs**: Creates a new Azure IoT hub.
- **Microsoft.Devices/provisioningServices**: Creates a new Azure IoT Hub Device Provisioning Service with the new IoT hub already linked to it.

Deploy the template

Deploy with the Portal

1. Select the following image to sign in to Azure and open the template for deployment. The template creates a new Iot hub and DPS resource. The new IoT hub is linked to the DPS resource.

   Deploy to Azure

2. Select or enter the following values and select **Review + Create**.

Unless otherwise specified for the following fields, use the default value to create the Iot Hub and DPS resource.

Field	Description
Subscription	Select your Azure subscription.
Resource group	Select **Create new**, and enter a unique name for the resource group, and then select **OK**.
Region	Select a region for your resources. For example, **East US**. For resiliency and reliability, we recommend deploying to one of the regions that support Availability Zones.

Field	Description
Iot Hub Name	Enter a name for the IoT Hub that must be globally unique within the *.azure-devices.net* namespace. You need the hub name in the next section when you validate the deployment.
Provisioning Service Name	Enter a name for the new Device Provisioning Service (DPS) resource. The name must be globally unique within the *.azure-devices-provisioning.net* namespace. You need the DPS name in the next section when you validate the deployment.

3. On the next screen, read the terms. If you agree to all terms, select **Create**.

   The deployment takes a few moments to complete.

   In addition to the Azure portal, you can also use the Azure PowerShell, Azure CLI, and REST API. To learn other deployment methods, see Deploy templates.

Deploy with the Azure CLI

Using the Azure CLI requires version 2.6 or later. If you're running the Azure CLI locally, verify your version by running: `az --version`

Sign in to your Azure account and select your subscription.

1. If you're running the Azure CLI locally instead of running it in the portal, you need to sign in. To sign in at the command prompt, run the login command:

   Azure CLI

   ```
   az login
   ```

   Follow the instructions to authenticate using the code and sign in to your Azure account through a web browser.

2. If you have multiple Azure subscriptions, signing in to Azure grants you access to all the Azure accounts associated with your credentials. Use the following command to list the Azure accounts available for you to use:

   Azure CLI

   ```
   az account list -o table
   ```

   Use the following command to select subscription that you want to use to run the commands to create your IoT hub and DPS resources. You can use either the

subscription name or ID from the output of the previous command:

Azure CLI

```
az account set --subscription {your subscription name or id}
```

3. Copy and paste the following commands into your CLI prompt. Then execute the commands by selecting the Enter key.

> **Tip**
>
> The commands prompt for a resource group location. You can view a list of available locations by first running the command:
>
> `az account list-locations -o table`

Azure CLI

```
read -p "Enter a project name that is used for generating resource
names:" projectName &&
read -p "Enter the location (i.e. centralus):" location &&
templateUri="https://raw.githubusercontent.com/Azure/azure-quickstart-
templates/master/quickstarts/microsoft.devices/iothub-device-
provisioning/azuredeploy.json" &&
resourceGroupName="${projectName}rg" &&
az group create --name $resourceGroupName --location "$location" &&
az deployment group create --resource-group $resourceGroupName --
template-uri  $templateUri &&
echo "Press [ENTER] to continue ..." &&
read
```

4. The commands prompt you for the following information. Provide each value and select the Enter key.

Parameter	Description
Project name	The value of this parameter is used to create a resource group to hold all resources. The string `rg` is added to the end of the value for your resource group name.
location	This value is the region where all resources are created.
iotHubName	Enter a name for the IoT Hub that must be globally unique within the *.azure-devices.net* namespace. You need the hub name in the next section when you validate the deployment.

Parameter	Description
provisioningServiceName	Enter a name for the new Device Provisioning Service (DPS) resource. The name must be globally unique within the *.azure-devices-provisioning.net* namespace. You need the DPS name in the next section when you validate the deployment.

The Azure CLI is used to deploy the template. In addition to the Azure CLI, you can also use the Azure PowerShell, Azure portal, and REST API. To learn other deployment methods, see Deploy templates.

Review deployed resources

1. To verify the deployment, run the following command to list resources and look for the new provisioning service and IoT hub in the output:

   Azure CLI

   ```
   az resource list -g "${projectName}rg"
   ```

2. To verify that the hub is already linked to the DPS resource, run the following DPS extension show command.

   Azure CLI

   ```
   az iot dps show --name <Your provisioningServiceName>
   ```

   Notice the hubs that are linked on the `iotHubs` member.

Clean up resources

Other quickstarts in this collection build upon this quickstart. If you plan to continue on to work with subsequent quickstarts or with the tutorials, don't clean up the resources created in this quickstart. If you don't plan to continue, you can use the Azure portal or Azure CLI to delete the resource group and all of its resources.

To delete a resource group and all its resources from the Azure portal, just open the resource group and select **Delete resource group** and the top.

To delete the resource group deployed using the Azure CLI:

Azure CLI

```
az group delete --name "${projectName}rg"
```

You can also delete resource groups and individual resources using any of the following options:

- Azure portal
- PowerShell
- REST APIs
- Supported platform SDKs published for Azure Resource Manager or IoT Hub Device Provisioning Service

Next steps

In this quickstart, you deployed an IoT hub and a Device Provisioning Service instance, and linked the two resources. To learn how to use this setup to provision a device, continue to the quickstart for creating a device.

Quickstart: Provision a simulated symmetric key device

Quickstart: Create Azure Advisor alerts on new recommendations using an ARM template

Article • 03/09/2023

This article shows you how to set up an alert for new recommendations from Azure Advisor using an Azure Resource Manager template (ARM template).

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

Whenever Azure Advisor detects a new recommendation for one of your resources, an event is stored in Azure Activity log. You can set up alerts for these events from Azure Advisor using a recommendation-specific alerts creation experience. You can select a subscription and optionally a resource group to specify the resources that you want to receive alerts on.

You can also determine the types of recommendations by using these properties:

- Category
- Impact level
- Recommendation type

You can also configure the action that will take place when an alert is triggered by:

- Selecting an existing action group
- Creating a new action group

To learn more about action groups, see Create and manage action groups.

> **Note**
>
> Advisor alerts are currently only available for High Availability, Performance, and Cost recommendations. Security recommendations are not supported.

Prerequisites

- If you don't have an Azure subscription, create a free account before you begin.

- To run the commands from your local computer, install Azure CLI or the Azure PowerShell modules. For more information, see Install the Azure CLI and Install Azure PowerShell.

Review the template

The following template creates an action group with an email target and enables all service health notifications for the target subscription. Save this template as *CreateAdvisorAlert.json*.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "actionGroups_name": {
      "defaultValue": "advisorAlert",
      "type": "string"
    },
    "activityLogAlerts_name": {
      "defaultValue": "AdvisorAlertsTest",
      "type": "string"
    },
    "emailAddress": {
      "defaultValue": "<email address>",
      "type": "string"
    }
  },
  "variables": {
    "alertScope": "[concat('/','subscriptions','/',subscription().subscriptionId)]"
  },
  "resources": [
    {
      "comments": "Action Group",
      "type": "microsoft.insights/actionGroups",
      "apiVersion": "2019-06-01",
      "name": "[parameters('actionGroups_name')]",
      "location": "Global",
      "scale": null,
      "dependsOn": [],
      "tags": {},
      "properties": {
        "groupShortName": "[parameters('actionGroups_name')]",
        "enabled": true,
        "emailReceivers": [
          {
            "name": "[parameters('actionGroups_name')]",
            "emailAddress": "[parameters('emailAddress')]"
```

```
          }
        ],
        "smsReceivers": [],
        "webhookReceivers": []
      }
    },
    {
      "comments": "Azure Advisor Activity Log Alert",
      "type": "microsoft.insights/activityLogAlerts",
      "apiVersion": "2017-04-01",
      "name": "[parameters('activityLogAlerts_name')]",
      "location": "Global",
      "scale": null,
      "tags": {},
      "properties": {
        "scopes": [
          "[variables('alertScope')]"
        ],
        "condition": {
          "allOf": [
            {
              "field": "category",
              "equals": "Recommendation"
            },
            {
              "field": "properties.recommendationCategory",
              "equals": "Cost"
            },
            {
              "field": "properties.recommendationImpact",
              "equals": "Medium"
            },
            {
              "field": "operationName",
              "equals": "Microsoft.Advisor/recommendations/available/action"
            }
          ]
        },
        "actions": {
          "actionGroups": [
            {
              "actionGroupId": "
[resourceId('microsoft.insights/actionGroups',
parameters('actionGroups_name'))]",
              "webhookProperties": {}
            }
          ]
        },
        "enabled": true,
        "description": ""
      },
      "dependsOn": [
        "[resourceId('microsoft.insights/actionGroups',
parameters('actionGroups_name'))]"
      ]
```

```
    }
  ]
}
```

The template defines two resources:

- Microsoft.Insights/actionGroups
- Microsoft.Insights/activityLogAlerts

Deploy the template

Deploy the template using any standard method for deploying an ARM template such as the following examples using CLI and PowerShell. Replace the sample values for **Resource Group**, and **emailAddress** with appropriate values for your environment. The workspace name must be unique among all Azure subscriptions.

CLI

Azure CLI

```
az login
az deployment group create --name CreateAdvisorAlert --resource-group
my-resource-group --template-file CreateAdvisorAlert.json --parameters
emailAddress='user@contoso.com'
```

Validate the deployment

Verify that the workspace has been created using one of the following commands. Replace the sample values for **Resource Group** with the value you used above.

CLI

Azure CLI

```
az monitor activity-log alert show --resource-group my-resource-group --
name AdvisorAlertsTest
```

Clean up resources

If you plan to continue working with subsequent quickstarts and tutorials, you might want to leave these resources in place. When no longer needed, delete the resource group, which deletes the alert rule and the related resources. To delete the resource group by using Azure CLI or Azure PowerShell

CLI

Azure CLI

```
az group delete --name my-resource-group
```

Next steps

- Get an overview of activity log alerts, and learn how to receive alerts.
- Learn more about action groups.

Create an Azure Automation account using a Resource Manager template

Article • 04/12/2023

Azure Automation delivers a cloud-based automation and configuration service that supports consistent management across your Azure and non-Azure environments. This article shows you how to deploy an Azure Resource Manager template (ARM template) that creates an Automation account. Using an ARM template takes fewer steps compared to other deployment methods. The JSON template specifies default values for parameters that would likely be used as a standard configuration in your environment. You can store the template in an Azure storage account for shared access in your organization. For more information about working with templates, see Deploy resources with ARM templates and the Azure CLI.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

The sample template does the following steps:

- Automates the creation of an Azure Monitor Log Analytics workspace.
- Automates the creation of an Azure Automation account.
- Links the Automation account to the Log Analytics workspace.
- Adds sample Automation runbooks to the account.

If you don't have an Azure subscription, create a free account before you begin.

Prerequisites

If you're new to Azure Automation and Azure Monitor, it's important that you understand the configuration details. The understanding can help you avoid errors when you try to create, configure, and use a Log Analytics workspace linked to your new Automation account.

- Review additional details to fully understand workspace configuration options, such as access control mode, pricing tier, retention, and capacity reservation level.

- Review workspace mappings to specify the supported regions inline or in a parameter file. Only certain regions are supported for linking a Log Analytics workspace and an Automation account in your subscription.

- If you're new to Azure Monitor Logs and haven't deployed a workspace already, review the workspace design guidance. This document will help you learn about access control, and help you understand the recommended design implementation strategies for your organization.

Review the template

The template used in this article is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "workspaceName": {
      "type": "string",
      "metadata": {
        "description": "Workspace name"
      }
    },
    "sku": {
      "type": "string",
      "defaultValue": "pergb2018",
      "allowedValues": [
        "pergb2018",
        "Free",
        "Standalone",
        "PerNode",
        "Standard",
        "Premium"
      ],
      "metadata": {
        "description": "Pricing tier: perGB2018 or legacy tiers (Free, Standalone, PerNode, Standard or Premium), which are not available to all customers."
      }
    },
    "dataRetention": {
      "type": "int",
      "defaultValue": 30,
      "minValue": 7,
      "maxValue": 730,
      "metadata": {
        "description": "Number of days to retain data."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
```

```
      "metadata": {
        "description": "Specifies the location in which to create the
workspace."
      }
    },
    "automationAccountName": {
      "type": "string",
      "metadata": {
        "description": "Automation account name"
      }
    },
    "sampleGraphicalRunbookName": {
      "type": "String",
      "defaultValue": "AzureAutomationTutorial"
    },
    "sampleGraphicalRunbookDescription": {
      "type": "String",
      "defaultValue": "An example runbook that gets all the Resource Manager
resources by using the Run As account (service principal)."
    },
    "samplePowerShellRunbookName": {
      "type": "String",
      "defaultValue": "AzureAutomationTutorialScript"
    },
    "samplePowerShellRunbookDescription": {
      "type": "String",
      "defaultValue": "An example runbook that gets all the Resource Manager
resources by using the Run As account (service principal)."
    },
    "samplePython2RunbookName": {
      "type": "String",
      "defaultValue": "AzureAutomationTutorialPython2"
    },
    "samplePython2RunbookDescription": {
      "type": "String",
      "defaultValue": "An example runbook that gets all the Resource Manager
resources by using the Run As account (service principal)."
    },
    "_artifactsLocation": {
      "type": "string",
      "defaultValue": "[deployment().properties.templateLink.uri]",
      "metadata": {
        "description": "URI to artifacts location"
      }
    },
    "_artifactsLocationSasToken": {
      "type": "securestring",
      "defaultValue": "",
      "metadata": {
        "description": "The sasToken required to access _artifactsLocation.
When the template is deployed using the accompanying scripts, a sasToken
will be automatically generated"
      }
    }
  },
```

```
  "resources": [
    {
      "type": "Microsoft.OperationalInsights/workspaces",
      "apiVersion": "2020-08-01",
      "name": "[parameters('workspaceName')]",
      "location": "[parameters('location')]",
      "properties": {
        "sku": {
          "name": "[parameters('sku')]"
        },
        "retentionInDays": "[parameters('dataRetention')]",
        "features": {
          "searchVersion": 1,
          "legacy": 0
        }
      }
    },
    {
      "type": "Microsoft.Automation/automationAccounts",
      "apiVersion": "2020-01-13-preview",
      "name": "[parameters('automationAccountName')]",
      "location": "[parameters('location')]",
      "dependsOn": [
        "[parameters('workspaceName')]"
      ],
      "identity": {
        "type": "SystemAssigned"
      },
      "properties": {
        "sku": {
          "name": "Basic"
        }
      },
      "resources": [
        {
          "type": "runbooks",
          "apiVersion": "2020-01-13-preview",
          "name": "[parameters('sampleGraphicalRunbookName')]",
          "location": "[parameters('location')]",
          "dependsOn": [
            "[parameters('automationAccountName')]"
          ],
          "properties": {
            "runbookType": "GraphPowerShell",
            "logProgress": "false",
            "logVerbose": "false",
            "description": "
[parameters('sampleGraphicalRunbookDescription')]",
            "publishContentLink": {
              "uri": "[uri(parameters('_artifactsLocation'),
concat('scripts/AzureAutomationTutorial.graphrunbook',
parameters('_artifactsLocationSasToken')))]",
              "version": "1.0.0.0"
            }
          }
```

```
        },
        {
          "type": "runbooks",
          "apiVersion": "2020-01-13-preview",
          "name": "[parameters('samplePowerShellRunbookName')]",
          "location": "[parameters('location')]",
          "dependsOn": [
            "[parameters('automationAccountName')]"
          ],
          "properties": {
            "runbookType": "PowerShell",
            "logProgress": "false",
            "logVerbose": "false",
            "description": "
[parameters('samplePowerShellRunbookDescription')]",
            "publishContentLink": {
              "uri": "[uri(parameters('_artifactsLocation'),
concat('scripts/AzureAutomationTutorial.ps1',
parameters('_artifactsLocationSasToken')))]",
              "version": "1.0.0.0"
            }
          }
        },
        {
          "type": "runbooks",
          "apiVersion": "2020-01-13-preview",
          "name": "[parameters('samplePython2RunbookName')]",
          "location": "[parameters('location')]",
          "dependsOn": [
            "[parameters('automationAccountName')]"
          ],
          "properties": {
            "runbookType": "Python2",
            "logProgress": "false",
            "logVerbose": "false",
            "description": "
[parameters('samplePython2RunbookDescription')]",
            "publishContentLink": {
              "uri": "[uri(parameters('_artifactsLocation'),
concat('scripts/AzureAutomationTutorialPython2.py',
parameters('_artifactsLocationSasToken')))]",
              "version": "1.0.0.0"
            }
          }
        }
      ]
    },
    {
      "type": "Microsoft.OperationalInsights/workspaces/linkedServices",
      "apiVersion": "2020-08-01",
      "name": "[concat(parameters('workspaceName'), '/' , 'Automation')]",
      "location": "[parameters('location')]",
      "dependsOn": [
        "[parameters('workspaceName')]",
        "[parameters('automationAccountName')]"
```

```
      ],
      "properties": {
        "resourceId": "
[resourceId('Microsoft.Automation/automationAccounts',
parameters('automationAccountName'))]"
      }
    }
  ]
}
```

The Azure resources defined in the template:

- **Microsoft.OperationalInsights/workspaces**: creates an Azure Log Analytics workspace.
- **Microsoft.Automation/automationAccounts**: creates an Azure Automation account.
- **Microsoft.Automation/automationAccounts/runbooks**: creates an Azure Automation account runbook.

Deploy the template

1. Select the **Deploy to Azure** button below to sign in to Azure and open the ARM template.

   Deploy to Azure

2. Enter or select the following values:

Property	Description
Subscription	From the drop-down list, select your Azure subscription.
Resource group	From the drop-down list, select your existing resource group, or select **Create new**.
Region	This value will autopopulate.
Workspace name	Enter a name for your new Log Analytics Workspace.
Sku	Defaults to the per GB pricing tier released in the April 2018 pricing model. If you want to create or configure a Log Analytics workspace in a subscription that has opted into the April 2018 pricing model, the only valid Log Analytics pricing tier is `PerGB2018`.
Data retention	Defaults to 30 days.

Property	Description
Location	The value will autopopulate with the location used for the resource group.
Automation Account name	Enter a name for your new Automation account.
Sample graphical runbook name	Leave as is.
Sample graphical runbook description	Leave as is.
Sample PowerShell runbook name	Leave as is.
Sample PowerShell runbook description	Leave as is.
Sample Python2Runbook name	Leave as is.
Sample Python2Runbook description	Leave as is.
_artifacts Location	Leave as is.* URI to artifacts location.
_artifacts Location Sas Token	Leave blank. The sasToken required to access `_artifactsLocation`. When the template is deployed using the accompanying scripts, a `sasToken` will be automatically generated.

* When you attempt to run the ARM template from PowerShell, CLI, or the Templates feature in the portal, if the `_artifactsLocation` parameter isn't properly set, you'll receive an error message similar to the following:

```
"message": "Deployment template validation failed: 'The template resource '_artifactsLocation' at line '96' and column '31' is not valid: The language expression property 'templateLink' doesn't exist, available properties are 'template, templateHash, parameters, mode, debugSetting, provisioningState'.. Please see https://aka.ms/arm-template-expressions for usage details.'."
```

To prevent this error, when running from the Templates feature in the portal, specify the following value for the `_artifactsLocation` parameter - `https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.automation/101-automation/azuredeploy.json`.

When you run from PowerShell, include the parameter and its value `-TemplateUri https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.automation/101-automation/azuredeploy.json`.

When you run from Azure CLI, include the parameter and its value - `--template-uri https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.automation/101-automation/azuredeploy.json`.

For reference about PowerShell/CLI, see the following - Create Azure Automation account (microsoft.com) under the **Use the template** section.

3. Select **Review + Create** and then **Create**. The deployment can take a few minutes to finish. When completed, the output is similar to the following image:

```
DeploymentName          : CreateAutomationAcct
ResourceGroupName       : Prod1
ProvisioningState       : Succeeded
Timestamp               : 4/15/2020 3:49:59 PM
Mode                    : Incremental
TemplateLink            :
Parameters              :
                          Name                                Type                        Value
                          ====================================  ==========================  ==========
                          workspaceName                       String                      Prod1-LA
                          sku                                 String                      pergb2018
                          dataRetention                       Int                         30
                          immediatePurgeDataOn30Days          Bool                        False
                          location                            String                      eastus
                          automationAccountName               String                      Prod1-AA
                          automationAccountLocation           String                      eastus2
                          sampleGraphicalRunbookName          String                      AzureAutomationTutorial
                          sampleGraphicalRunbookDescription   String                       An example runbook which gets
```

Review deployed resources

1. Once the deployment completes, you'll receive a **Deployment succeeded** notification with a **Go to resource** link. Your **Resource group** page will list your new resources. From the list, select your new Automation account.

2. From the left-side, under **Process Automation**, select **Runbooks**. The **Runbooks** page lists the three sample runbooks created with the Automation account.

3. From the left-side, under **Related Resources**, select **Linked workspace**. The **Linked workspace** page shows the Log Analytics workspace you specified earlier that is linked to your Automation account.

Next steps

Configure diagnostic settings for your Automation account to send runbook job status and job streams to the linked Log Analytics workspace.

Back up a virtual machine in Azure with an ARM template

Article • 03/08/2023

Azure Backup backs up on-premises machines and apps, and Azure VMs. This article shows you how to back up an Azure VM with an Azure Resource Manager template (ARM template) and Azure PowerShell. This quickstart focuses on the process of deploying an ARM template to create a Recovery Services vault. For more information on developing ARM templates, see the Azure Resource Manager documentation and the template reference.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

A Recovery Services vault is a logical container that stores backup data for protected resources, such as Azure VMs. When a backup job runs, it creates a recovery point inside the Recovery Services vault. You can then use one of these recovery points to restore data to a given point in time. Alternatively, you can back up a VM using Azure PowerShell, the Azure CLI, or in the Azure portal.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Review the template

The template used in this quickstart is from Azure quickstart Templates. This template allows you to deploy simple Windows VM and Recovery Services vault configured with the *DefaultPolicy* for *Protection*.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
```

```
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "12431143174203400392"
    }
  },
  "parameters": {
    "projectName": {
      "type": "string",
      "maxLength": 8,
      "metadata": {
        "description": "Specifies a name for generating resource names."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Specifies the location for all resources."
      }
    },
    "adminUsername": {
      "type": "string",
      "metadata": {
        "description": "Specifies the administrator username for the Virtual
Machine."
      }
    },
    "adminPassword": {
      "type": "secureString",
      "metadata": {
        "description": "Specifies the administrator password for the Virtual
Machine."
      }
    },
    "dnsLabelPrefix": {
      "type": "string",
      "metadata": {
        "description": "Specifies the unique DNS Name for the Public IP used
to access the Virtual Machine."
      }
    },
    "vmSize": {
      "type": "string",
      "defaultValue": "Standard_A2",
      "metadata": {
        "description": "Virtual machine size."
      }
    },
    "windowsOSVersion": {
      "type": "string",
      "defaultValue": "2016-Datacenter",
      "allowedValues": [
        "2008-R2-SP1",
        "2012-Datacenter",
```

```
        "2012-R2-Datacenter",
        "2016-Nano-Server",
        "2016-Datacenter-with-Containers",
        "2016-Datacenter",
        "2019-Datacenter",
        "2019-Datacenter-Core",
        "2019-Datacenter-Core-smalldisk",
        "2019-Datacenter-Core-with-Containers",
        "2019-Datacenter-Core-with-Containers-smalldisk",
        "2019-Datacenter-smalldisk",
        "2019-Datacenter-with-Containers",
        "2019-Datacenter-with-Containers-smalldisk"
      ],
      "metadata": {
        "description": "Specifies the Windows version for the VM. This will
pick a fully patched image of this given Windows version."
      }
    }
  },
  "variables": {
    "storageAccountName": "[format('{0}store', parameters('projectName'))]",
    "networkInterfaceName": "[format('{0}-nic',
parameters('projectName'))]",
    "vNetAddressPrefix": "10.0.0.0/16",
    "vNetSubnetName": "default",
    "vNetSubnetAddressPrefix": "10.0.0.0/24",
    "publicIPAddressName": "[format('{0}-ip', parameters('projectName'))]",
    "vmName": "[format('{0}-vm', parameters('projectName'))]",
    "vNetName": "[format('{0}-vnet', parameters('projectName'))]",
    "vaultName": "[format('{0}-vault', parameters('projectName'))]",
    "backupFabric": "Azure",
    "backupPolicyName": "DefaultPolicy",
    "protectionContainer": "[format('iaasvmcontainer;iaasvmcontainerv2;{0};
{1}', resourceGroup().name, variables('vmName'))]",
    "protectedItem": "[format('vm;iaasvmcontainerv2;{0};{1}',
resourceGroup().name, variables('vmName'))]",
    "networkSecurityGroupName": "default-NSG"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-08-01",
      "name": "[variables('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "Storage",
      "properties": {}
    },
    {
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2021-05-01",
      "name": "[variables('publicIPAddressName')]",
      "location": "[parameters('location')]",
```

```
      "properties": {
        "publicIPAllocationMethod": "Dynamic",
        "dnsSettings": {
          "domainNameLabel": "[parameters('dnsLabelPrefix')]"
        }
      }
    },
    {
      "type": "Microsoft.Network/networkSecurityGroups",
      "apiVersion": "2021-05-01",
      "name": "[variables('networkSecurityGroupName')]",
      "location": "[parameters('location')]",
      "properties": {
        "securityRules": [
          {
            "name": "default-allow-3389",
            "properties": {
              "priority": 1000,
              "access": "Allow",
              "direction": "Inbound",
              "destinationPortRange": "3389",
              "protocol": "Tcp",
              "sourceAddressPrefix": "*",
              "sourcePortRange": "*",
              "destinationAddressPrefix": "*"
            }
          }
        ]
      }
    },
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "2021-05-01",
      "name": "[variables('vNetName')]",
      "location": "[parameters('location')]",
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[variables('vNetAddressPrefix')]"
          ]
        },
        "subnets": [
          {
            "name": "[variables('vNetSubnetName')]",
            "properties": {
              "addressPrefix": "[variables('vNetSubnetAddressPrefix')]",
              "networkSecurityGroup": {
                "id": "
[resourceId('Microsoft.Network/networkSecurityGroups',
variables('networkSecurityGroupName'))]"
              }
            }
          }
        ]
      },
```

```
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('networkSecurityGroupName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/networkInterfaces",
      "apiVersion": "2021-05-01",
      "name": "[variables('networkInterfaceName')]",
      "location": "[parameters('location')]",
      "properties": {
        "ipConfigurations": [
          {
            "name": "ipconfig1",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses',
variables('publicIPAddressName'))]"
              },
              "subnet": {
                "id": "[format('{0}/subnets/{1}',
resourceId('Microsoft.Network/virtualNetworks', variables('vNetName')),
variables('vNetSubnetName'))]"
              }
            }
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/publicIPAddresses',
variables('publicIPAddressName'))]",
        "[resourceId('Microsoft.Network/virtualNetworks',
variables('vNetName'))]"
      ]
    },
    {
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2021-11-01",
      "name": "[variables('vmName')]",
      "location": "[parameters('location')]",
      "properties": {
        "hardwareProfile": {
          "vmSize": "[parameters('vmSize')]"
        },
        "osProfile": {
          "computerName": "[variables('vmName')]",
          "adminUsername": "[parameters('adminUsername')]",
          "adminPassword": "[parameters('adminPassword')]"
        },
        "storageProfile": {
          "imageReference": {
            "publisher": "MicrosoftWindowsServer",
            "offer": "WindowsServer",
            "sku": "[parameters('windowsOSVersion')]",
```

```
            "version": "latest"
          },
          "osDisk": {
            "createOption": "FromImage"
          },
          "dataDisks": [
            {
              "diskSizeGB": 1023,
              "lun": 0,
              "createOption": "Empty"
            }
          ]
        },
        "networkProfile": {
          "networkInterfaces": [
            {
              "id": "[resourceId('Microsoft.Network/networkInterfaces',
variables('networkInterfaceName'))]"
            }
          ]
        },
        "diagnosticsProfile": {
          "bootDiagnostics": {
            "enabled": true,
            "storageUri": "
[reference(resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))).primaryEndpoints.blob]"
          }
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkInterfaces',
variables('networkInterfaceName'))]",
        "[resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))]"
      ]
    },
    {
      "type": "Microsoft.RecoveryServices/vaults",
      "apiVersion": "2022-01-01",
      "name": "[variables('vaultName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "RS0",
        "tier": "Standard"
      },
      "properties": {}
    },
    {
      "type":
"Microsoft.RecoveryServices/vaults/backupFabrics/protectionContainers/protec
tedItems",
      "apiVersion": "2022-01-01",
      "name": "[format('{0}/{1}/{2}/{3}', variables('vaultName'),
variables('backupFabric'), variables('protectionContainer'),
```

```
variables('protectedItem'))]",
      "properties": {
        "protectedItemType": "Microsoft.Compute/virtualMachines",
        "policyId": "[format('{0}/backupPolicies/{1}',
resourceId('Microsoft.RecoveryServices/vaults', variables('vaultName')),
variables('backupPolicyName'))]",
        "sourceResourceId": "
[resourceId('Microsoft.Compute/virtualMachines', variables('vmName'))]"
      },
      "dependsOn": [
        "[resourceId('Microsoft.RecoveryServices/vaults',
variables('vaultName'))]",
        "[resourceId('Microsoft.Compute/virtualMachines',
variables('vmName'))]"
      ]
    }
  ]
}
```

The resources defined in the template are:

- Microsoft.Storage/storageAccounts
- Microsoft.Network/publicIPAddresses
- Microsoft.Network/networkSecurityGroups
- Microsoft.Network/virtualNetworks
- Microsoft.Network/networkInterfaces
- Microsoft.Compute/virtualMachines
- Microsoft.RecoveryServices/vaults
- Microsoft.RecoveryServices/vaults/backupFabrics/protectionContainers/protectedItems

Deploy the template

To deploy the template, select **Try it** to open the Azure Cloud Shell, and then paste the following PowerShell script into the shell window. To paste the code, right-click the shell window and then select **Paste**.

Azure PowerShell

```
$projectName = Read-Host -Prompt "Enter a project name (limited to eight
characters) that is used to generate Azure resource names"
$location = Read-Host -Prompt "Enter the location (for example, centralus)"
$adminUsername = Read-Host -Prompt "Enter the administrator username for the
virtual machine"
$adminPassword = Read-Host -Prompt "Enter the administrator password for the
virtual machine" -AsSecureString
$dnsPrefix = Read-Host -Prompt "Enter the unique DNS Name for the Public IP
used to access the virtual machine"
```

```
$resourceGroupName = "${projectName}rg"
$templateUri = "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.recoveryservices/recovery-services-create-vm-and-configure-backup/azuredeploy.json"

New-AzResourceGroup -Name $resourceGroupName -Location $location
New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName -TemplateUri $templateUri -projectName $projectName -adminUsername $adminUsername -adminPassword $adminPassword -dnsLabelPrefix $dnsPrefix
```

Azure PowerShell is used to deploy the ARM template in this quickstart. The Azure portal, Azure CLI, and REST API can also be used to deploy templates.

Validate the deployment

Start a backup job

The template creates a VM and enables backup on the VM. After you deploy the template, you need to start a backup job. For more information, see Start a backup job.

Monitor the backup job

To monitor the backup job, see Monitor the backup job.

Clean up resources

If you no longer need to back up the VM, you can clean it up.

- If you want to try out restoring the VM, skip the cleanup.
- If you used an existing VM, you can skip the final Remove-AzResourceGroup cmdlet to leave the resource group and VM in place.

Disable protection, remove the restore points and vault. Then delete the resource group and associated VM resources, as follows:

PowerShell

```
Disable-AzRecoveryServicesBackupProtection -Item $item -RemoveRecoveryPoints
$vault = Get-AzRecoveryServicesVault -Name "myRecoveryServicesVault"
Remove-AzRecoveryServicesVault -Vault $vault
Remove-AzResourceGroup -Name "myResourceGroup"
```

Next steps

In this quickstart, you created a Recovery Services vault, enabled protection on a VM, and created the initial recovery point.

- Learn how to back up VMs in the Azure portal.
- Learn how to quickly restore a VM
- Learn how to create ARM templates.

Quickstart: Create a budget with an ARM template

Article • 04/05/2023

Budgets in Cost Management help you plan for and drive organizational accountability. With budgets, you can account for the Azure services you consume or subscribe to during a specific period. They help you inform others about their spending to proactively manage costs, and to monitor how spending progresses over time. When the budget thresholds you've created are exceeded, notifications are triggered. None of your resources are affected and your consumption isn't stopped. You can use budgets to compare and track spending as you analyze costs. This quickstart shows you how to create a budget using three different Azure Resource Manager templates (ARM template).

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button for one of the following templates. The template will open in the Azure portal.

Template	Deployment button
No filter	Deploy to Azure
One filter	Deploy to Azure
Two or more filters	Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

If you have a new subscription, you can't immediately create a budget or use other Cost Management features. It might take up to 48 hours before you can use all Cost Management features.

Budgets are supported for the following types of Azure account types and scopes:

- Azure role-based access control (Azure RBAC) scopes
  - Management groups
  - Subscription
- Enterprise Agreement scopes
  - Billing account
  - Department
  - Enrollment account
- Individual agreements
  - Billing account
- Microsoft Customer Agreement scopes
  - Billing account
  - Billing profile
  - Invoice section
  - Customer
- AWS scopes
  - External account
  - External subscription

To view budgets, you need at least read access for your Azure account.

For Azure EA subscriptions, you must have read access to view budgets. To create and manage budgets, you must have contributor permission.

The following Azure permissions, or scopes, are supported per subscription for budgets by user and group. For more information about scopes, see Understand and work with scopes.

- Owner – Can create, modify, or delete budgets for a subscription.
- Contributor and Cost Management contributor – Can create, modify, or delete their own budgets. Can modify the budget amount for budgets created by others.
- Reader and Cost Management reader – Can view budgets that they have permission to.

For more information about assigning permission to Cost Management data, see Assign access to Cost Management data.

Use one of the following templates, based on your needs.

Template	Description
No filter	The ARM template doesn't have any filters.

Template	Description
One filter	The ARM template has a filter for resource groups.
Two or more filters	The ARM template has a filter for resource groups and a filter for meter categories.

Review and deploy the template

No filter

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2018-05-01/subscriptionDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "15851432186142062555"
    }
  },
  "parameters": {
    "budgetName": {
      "type": "string",
      "defaultValue": "MyBudget",
      "metadata": {
        "description": "Name of the Budget. It should be unique within a resource group."
      }
    },
    "amount": {
      "type": "int",
      "defaultValue": 1000,
      "metadata": {
        "description": "The total amount of cost or usage to track with the budget"
      }
    },
    "timeGrain": {
      "type": "string",
      "defaultValue": "Monthly",
```

```
      "allowedValues": [
        "Monthly",
        "Quarterly",
        "Annually"
      ],
      "metadata": {
        "description": "The time covered by a budget. Tracking of the
amount will be reset based on the time grain."
      }
    },
    "startDate": {
      "type": "string",
      "metadata": {
        "description": "The start date must be first of the month in
YYYY-MM-DD format. Future start date should not be more than three
months. Past start date should be selected within the timegrain preiod."
      }
    },
    "endDate": {
      "type": "string",
      "metadata": {
        "description": "The end date for the budget in YYYY-MM-DD
format. If not provided, we default this to 10 years from the start
date."
      }
    },
    "firstThreshold": {
      "type": "int",
      "defaultValue": 90,
      "metadata": {
        "description": "Threshold value associated with a notification.
Notification is sent when the cost exceeded the threshold. It is always
percent and has to be between 0.01 and 1000."
      }
    },
    "secondThreshold": {
      "type": "int",
      "defaultValue": 110,
      "metadata": {
        "description": "Threshold value associated with a notification.
Notification is sent when the cost exceeded the threshold. It is always
percent and has to be between 0.01 and 1000."
      }
    },
    "contactEmails": {
      "type": "array",
      "metadata": {
        "description": "The list of email addresses to send the budget
notification to when the threshold is exceeded."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Consumption/budgets",
```

```
      "apiVersion": "2021-10-01",
      "name": "[parameters('budgetName')]",
      "properties": {
        "timePeriod": {
          "startDate": "[parameters('startDate')]",
          "endDate": "[parameters('endDate')]"
        },
        "timeGrain": "[parameters('timeGrain')]",
        "amount": "[parameters('amount')]",
        "category": "Cost",
        "notifications": {
          "NotificationForExceededBudget1": {
            "enabled": true,
            "operator": "GreaterThan",
            "threshold": "[parameters('firstThreshold')]",
            "contactEmails": "[parameters('contactEmails')]"
          },
          "NotificationForExceededBudget2": {
            "enabled": true,
            "operator": "GreaterThan",
            "threshold": "[parameters('secondThreshold')]",
            "contactEmails": "[parameters('contactEmails')]"
          }
        }
      }
    }
  ]
}
```

One Azure resource is defined in the template:

- Microsoft.Consumption/budgets: Create an Azure budget.

Deploy the template

1. Select the following image to sign in to Azure and open a template. The template creates a budget without any filters.

   Deploy to Azure

2. Select or enter the following values.

- **Subscription**: select an Azure subscription.
- **Resource group**: if required, select an existing resource group, or **Create new**.
- **Region**: select an Azure region. For example, **Central US**.
- **Budget Name**: enter a name for the budget. It should be unique within a resource group. Only alphanumeric, underscore, and hyphen characters are allowed.
- **Amount**: enter the total amount of cost to track with the budget.
- **Time Grain**: enter the time covered by a budget. Allowed values are Monthly, Quarterly, or Annually. The budget resets at the end of the time grain.
- **Start Date**: enter the start date with the first day of the month in YYYY-MM-DD format. A future start date shouldn't be more than three months from today. You can specify a past start date with the Time Grain period.

- **End Date**: enter the end date for the budget in YYYY-MM-DD format.
- **First Threshold**: enter a threshold value for the first notification. A notification is sent when the cost exceeds the threshold. It's always percent and has to be between 0.01 and 1000.
- **Second Threshold**: enter a threshold value for the second notification. A notification is sent when the cost exceeds the threshold. It's always percent and has to be between 0.01 and 1000.
- **Contact Emails** enter a list of email addresses to send the budget notification to when a threshold is exceeded. It accepts an array of strings. Expected format is `["user1@domain.com","user2@domain.com"]`.

3. Depending on your Azure subscription type, do one of the following actions:

   - Select **Review + create**.
   - Review the terms and conditions, select **I agree to the terms and conditions stated above**, and then select **Purchase**.

4. If you selected **Review + create**, your template is validated. Select **Create**.

The Azure portal is used to deploy the template. In addition to the Azure portal, you can also use Azure PowerShell, Azure CLI, and REST API. To learn about other deployment templates, see Deploy templates.

Validate the deployment

Use one of the following ways to verify that the budget is created.

Azure portal

Navigate to **Cost Management + Billing** > select a scope > **Budgets**.

Clean up resources

When you no longer need a budget, delete it by using one the following methods:

Azure portal

Navigate to **Cost Management + Billing** > select a billing scope > **Budgets** > select a budget > then select **Delete budget**.

Next steps

In this quickstart, you created an Azure budget and deployed it. To learn more about Cost Management and Billing and Azure Resource Manager, continue on to the articles below.

- Read the Cost Management and Billing overview
- Create budgets in the Azure portal
- Learn more about Azure Resource Manager

Quickstart: Create and publish an Azure Managed Application definition

Article • 05/14/2023

This quickstart provides an introduction to working with Azure Managed Applications. You create and publish a managed application definition that's stored in your service catalog and is intended for members of your organization.

To publish a managed application to your service catalog, do the following tasks:

- Create an Azure Resource Manager template (ARM template) that defines the resources to deploy with the managed application.
- Define the user interface elements for the portal when deploying the managed application.
- Create a *.zip* package that contains the required JSON files. The *.zip* package file has a 120-MB limit for a service catalog's managed application definition.
- Publish the managed application definition so it's available in your service catalog.

If your managed application definition is more than 120 MB or if you want to use your own storage account for your organization's compliance reasons, go to Quickstart: Bring your own storage to create and publish an Azure Managed Application definition.

You can use Bicep to develop a managed application definition but it must be converted to ARM template JSON before you can publish the definition in Azure. For more information, go to Quickstart: Use Bicep to create and publish an Azure Managed Application definition.

You can also use Bicep deploy a managed application definition from your service catalog. For more information, go to Quickstart: Use Bicep to deploy an Azure Managed Application definition.

Prerequisites

To complete this quickstart, you need the following items:

- An Azure account with an active subscription and permissions to Azure Active Directory resources like users, groups, or service principals. If you don't have an account, create a free account before you begin.
- Visual Studio Code with the latest Azure Resource Manager Tools extension. For Bicep files, install the Bicep extension for Visual Studio Code.
- Install the latest version of Azure PowerShell or Azure CLI.

Create the ARM template

Every managed application definition includes a file named *mainTemplate.json*. The template defines the Azure resources to deploy and is no different than a regular ARM template.

Open Visual Studio Code, create a file with the case-sensitive name *mainTemplate.json* and save it.

Add the following JSON and save the file. It defines the resources to deploy an App Service, App Service plan, and storage account for the application. This storage account isn't used to store the managed application definition.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    },
    "appServicePlanName": {
      "type": "string",
      "maxLength": 40,
      "metadata": {
        "description": "App Service plan name."
      }
    },
    "appServiceNamePrefix": {
      "type": "string",
      "maxLength": 47,
      "metadata": {
        "description": "App Service name prefix."
      }
    },
    "storageAccountNamePrefix": {
      "type": "string",
      "maxLength": 11,
      "metadata": {
        "description": "Storage account name prefix."
      }
    },
    "storageAccountType": {
      "type": "string",
      "allowedValues": [
        "Premium_LRS",
        "Standard_LRS",
        "Standard_GRS"
```

```
      ],
      "metadata": {
        "description": "Storage account type allowed values"
      }
    }
  },
  "variables": {
    "appServicePlanSku": "F1",
    "appServicePlanCapacity": 1,
    "appServiceName": "[format('{0}{1}', parameters('appServiceNamePrefix'),
uniqueString(resourceGroup().id))]",
    "storageAccountName": "[format('{0}{1}',
parameters('storageAccountNamePrefix'), uniqueString(resourceGroup().id))]"
  },
  "resources": [
    {
      "type": "Microsoft.Web/serverfarms",
      "apiVersion": "2022-03-01",
      "name": "[parameters('appServicePlanName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[variables('appServicePlanSku')]",
        "capacity": "[variables('appServicePlanCapacity')]"
      }
    },
    {
      "type": "Microsoft.Web/sites",
      "apiVersion": "2022-03-01",
      "name": "[variables('appServiceName')]",
      "location": "[parameters('location')]",
      "properties": {
        "serverFarmId": "[resourceId('Microsoft.Web/serverfarms',
parameters('appServicePlanName'))]",
        "httpsOnly": true,
        "siteConfig": {
          "appSettings": [
            {
              "name": "AppServiceStorageConnectionString",
              "value": "[format('DefaultEndpointsProtocol=https;AccountName=
{0};EndpointSuffix={1};Key={2}', variables('storageAccountName'),
environment().suffixes.storage,
listKeys(resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName')), '2022-09-01').keys[0].value)]"
            }
          ]
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Web/serverfarms',
parameters('appServicePlanName'))]",
        "[resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))]"
      ]
    },
    {
```

```
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2022-09-01",
      "name": "[variables('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('storageAccountType')]"
      },
      "kind": "StorageV2",
      "properties": {
        "accessTier": "Hot"
      }
    }
  ],
  "outputs": {
    "appServicePlan": {
      "type": "string",
      "value": "[parameters('appServicePlanName')]"
    },
    "appServiceApp": {
      "type": "string",
      "value": "[reference(resourceId('Microsoft.Web/sites',
variables('appServiceName')), '2022-03-01').defaultHostName]"
    },
    "storageAccount": {
      "type": "string",
      "value": "[reference(resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName')), '2022-09-01').primaryEndpoints.blob]"
    }
  }
}
```

Define your portal experience

As a publisher, you define the portal experience to create the managed application. The *createUiDefinition.json* file generates the portal's user interface. You define how users provide input for each parameter using control elements like drop-downs and text boxes.

In this example, the user interface prompts you to input the App Service name prefix, App Service plan's name, storage account prefix, and storage account type. During deployment, the variables in *mainTemplate.json* use the `uniqueString` function to append a 13-character string to the name prefixes so the names are globally unique across Azure.

Open Visual Studio Code, create a file with the case-sensitive name *createUiDefinition.json* and save it.

Add the following JSON code to the file and save it.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-
preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {}
    ],
    "steps": [
      {
        "name": "webAppSettings",
        "label": "Web App settings",
        "subLabel": {
          "preValidation": "Configure the web app settings",
          "postValidation": "Completed"
        },
        "elements": [
          {
            "name": "appServicePlanName",
            "type": "Microsoft.Common.TextBox",
            "label": "App Service plan name",
            "placeholder": "App Service plan name",
            "defaultValue": "",
            "toolTip": "Use alphanumeric characters or hyphens with a
maximum of 40 characters.",
            "constraints": {
              "required": true,
              "regex": "^[a-z0-9A-Z-]{1,40}$",
              "validationMessage": "Only alphanumeric characters or hyphens
are allowed, with a maximum of 40 characters."
            },
            "visible": true
          },
          {
            "name": "appServiceName",
            "type": "Microsoft.Common.TextBox",
            "label": "App Service name prefix",
            "placeholder": "App Service name prefix",
            "defaultValue": "",
            "toolTip": "Use alphanumeric characters or hyphens with minimum
of 2 characters and maximum of 47 characters.",
            "constraints": {
              "required": true,
              "regex": "^[a-z0-9A-Z-]{2,47}$",
              "validationMessage": "Only alphanumeric characters or hyphens
are allowed, with a minimum of 2 characters and maximum of 47 characters."
            },
            "visible": true
          }
        ]
      },
```

```
      {
        "name": "storageConfig",
        "label": "Storage settings",
        "subLabel": {
          "preValidation": "Configure the storage settings",
          "postValidation": "Completed"
        },
        "elements": [
          {
            "name": "storageAccounts",
            "type": "Microsoft.Storage.MultiStorageAccountCombo",
            "label": {
              "prefix": "Storage account name prefix",
              "type": "Storage account type"
            },
            "toolTip": {
              "prefix": "Enter maximum of 11 lowercase letters or numbers.",
              "type": "Available choices are Standard_LRS, Standard_GRS, and
Premium_LRS."
            },
            "defaultValue": {
              "type": "Standard_LRS"
            },
            "constraints": {
              "allowedTypes": [
                "Premium_LRS",
                "Standard_LRS",
                "Standard_GRS"
              ]
            },
            "visible": true
          }
        ]
      }
    ],
    "outputs": {
      "location": "[location()]",
      "appServicePlanName": "[steps('webAppSettings').appServicePlanName]",
      "appServiceNamePrefix": "[steps('webAppSettings').appServiceName]",
      "storageAccountNamePrefix": "
[steps('storageConfig').storageAccounts.prefix]",
      "storageAccountType": "[steps('storageConfig').storageAccounts.type]"
    }
  }
}
```

To learn more, see Get started with CreateUiDefinition.

Package the files

Add the two files to a package file named *app.zip*. The two files must be at the root level of the *.zip* file. If the files are in a folder, when you create the managed application definition, you receive an error that states the required files aren't present.

Upload *app.zip* to an Azure storage account so you can use it when you deploy the managed application's definition. The storage account name must be globally unique across Azure and the length must be 3-24 characters with only lowercase letters and numbers. In the command, replace the placeholder `<demostorageaccount>` including the angle brackets (`<>`), with your unique storage account name.

PowerShell

In Visual Studio Code, open a new PowerShell terminal and sign in to your Azure subscription.

Azure PowerShell

```
Connect-AzAccount
```

The command opens your default browser and prompts you to sign in to Azure. For more information, go to Sign in with Azure PowerShell.

Azure PowerShell

```
New-AzResourceGroup -Name packageStorageGroup -Location westus3

$storageAccount = New-AzStorageAccount `
  -ResourceGroupName packageStorageGroup `
  -Name "<demostorageaccount>" `
  -Location westus3 `
  -SkuName Standard_LRS `
  -Kind StorageV2

$ctx = $storageAccount.Context

New-AzStorageContainer -Name appcontainer -Context $ctx -Permission blob

Set-AzStorageBlobContent `
  -File "app.zip" `
  -Container appcontainer `
  -Blob "app.zip" `
  -Context $ctx
```

Create the managed application definition

In this section, you get identity information from Azure Active Directory, create a resource group, and deploy the managed application definition.

Get group ID and role definition ID

The next step is to select a user, security group, or application for managing the resources for the customer. This identity has permissions on the managed resource group according to the assigned role. The role can be any Azure built-in role like Owner or Contributor.

PowerShell

This example uses a security group, and your Azure Active Directory account should be a member of the group. To get the group's object ID, replace the placeholder `<managedAppDemo>` including the angle brackets (`<>`), with your group's name. You use this variable's value when you deploy the managed application definition.

To create a new Azure Active Directory group, go to Manage Azure Active Directory groups and group membership.

Azure PowerShell

```
$principalid=(Get-AzADGroup -DisplayName <managedAppDemo>).Id
```

Next, get the role definition ID of the Azure built-in role you want to grant access to the user, group, or application. You use this variable's value when you deploy the managed application definition.

Azure PowerShell

```
$roleid=(Get-AzRoleDefinition -Name Owner).Id
```

Publish the managed application definition

PowerShell

Create a resource group for your managed application definition.

Azure PowerShell

```
New-AzResourceGroup -Name appDefinitionGroup -Location westus3
```

The `blob` command creates a variable to store the URL for the package *.zip* file. That variable is used in the command that creates the managed application definition.

Azure PowerShell

```
$blob = Get-AzStorageBlob -Container appcontainer -Blob app.zip -Context
$ctx

New-AzManagedApplicationDefinition `
  -Name "sampleManagedApplication" `
  -Location "westus3" `
  -ResourceGroupName appDefinitionGroup `
  -LockLevel ReadOnly `
  -DisplayName "Sample managed application" `
  -Description "Sample managed application that deploys web resources" `
  -Authorization "${principalid}:$roleid" `
  -PackageFileUri $blob.ICloudBlob.StorageUri.PrimaryUri.AbsoluteUri
```

When the command completes, you have a managed application definition in your resource group.

Some of the parameters used in the preceding example are:

- `ResourceGroupName`: The name of the resource group where the managed application definition is created.
- `LockLevel`: The `lockLevel` on the managed resource group prevents the customer from performing undesirable operations on this resource group. Currently, `ReadOnly` is the only supported lock level. `ReadOnly` specifies that the customer can only read the resources present in the managed resource group. The publisher identities that are granted access to the managed resource group are exempt from the lock level.
- `Authorization`: Describes the principal ID and the role definition ID that are used to grant permission to the managed resource group.
  - `"${principalid}:$roleid"` or you can use curly braces for each variable `"${principalid}:${roleid}"`.
  - Use a comma to separate multiple values: `"${principalid1}:$roleid1", "${principalid2}:$roleid2"`.
- `PackageFileUri`: The location of a *.zip* package file that contains the required files.

Make sure users can see your definition

You have access to the managed application definition, but you want to make sure other users in your organization can access it. Grant them at least the Reader role on the definition. They may have inherited this level of access from the subscription or resource group. To check who has access to the definition and add users or groups, see Assign Azure roles using the Azure portal.

Clean up resources

If you're going to deploy the definition, continue with the **Next steps** section that links to the article to deploy the definition.

If you're finished with the managed application definition, you can delete the resource groups you created named *packageStorageGroup* and *appDefinitionGroup*.

PowerShell

The command prompts you to confirm that you want to remove the resource group.

Azure PowerShell

```
Remove-AzResourceGroup -Name packageStorageGroup

Remove-AzResourceGroup -Name appDefinitionGroup
```

Next steps

You've published the managed application definition. The next step is to learn how to deploy an instance of that definition.

Quickstart: Deploy a service catalog managed application

Resource Manager template samples for diagnostic settings in Azure Monitor

Article • 03/16/2023

This article includes sample Azure Resource Manager templates to create diagnostic settings for an Azure resource. Each sample includes a template file and a parameters file with sample values to provide to the template.

To create a diagnostic setting for an Azure resource, add a resource of type `<resource namespace>/providers/diagnosticSettings` to the template. This article provides examples for some resource types, but the same pattern can be applied to other resource types. The collection of allowed logs and metrics will vary for each resource type.

> **Note**
>
> See **Azure Resource Manager samples** for Azure Monitor for a list of samples that are available and guidance on deploying them in your Azure subscription.

Diagnostic setting for an activity log

The following sample creates a diagnostic setting for an activity log by adding a resource of type `Microsoft.Insights/diagnosticSettings` to the template.

> **Important**
>
> Diagnostic settings for activity logs are created for a subscription, not for a resource group like settings for Azure resources. To deploy the Resource Manager template, use `New-AzSubscriptionDeployment` for PowerShell or `az deployment sub create` for the Azure CLI.

Template file

Bicep

Bicep

```
targetScope = 'subscription'

@description('The name of the diagnostic setting.')
param settingName string

@description('The resource Id for the workspace.')
param workspaceId string

@description('The resource Id for the storage account.')
param storageAccountId string

@description('The resource Id for the event hub authorization rule.')
param eventHubAuthorizationRuleId string

@description('The name of the event hub.')
param eventHubName string

resource setting 'Microsoft.Insights/diagnosticSettings@2021-05-01-
preview' = {
  name: settingName
  properties: {
    workspaceId: workspaceId
    storageAccountId: storageAccountId
    eventHubAuthorizationRuleId: eventHubAuthorizationRuleId
    eventHubName: eventHubName
    logs: [
      {
        category: 'Administrative'
        enabled: true
      }
      {
        category: 'Security'
        enabled: true
      }
      {
        category: 'ServiceHealth'
        enabled: true
      }
      {
        category: 'Alert'
        enabled: true
      }
      {
        category: 'Recommendation'
        enabled: true
      }
      {
        category: 'Policy'
        enabled: true
      }
      {
        category: 'Autoscale'
        enabled: true
      }
```

```
        {
          category: 'ResourceHealth'
          enabled: true
        }
      ]
    }
  }
```

Parameter file

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2018-05-01/subscriptionDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "settingName": {
      "value": "Send to all locations"
    },
    "workspaceId": {
      "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourcegroups/MyResourceGroup/providers/microsoft.operationalinsights/workspaces/MyWorkspace"
    },
    "storageAccountId": {
      "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/MyResourceGroup/providers/Microsoft.Storage/storageAccounts/mystorageaccount"
    },
    "eventHubAuthorizationRuleId": {
      "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/MyResourceGroup/providers/Microsoft.EventHub/namespaces/MyNameSpace/authorizationrules/RootManageSharedAccessKey"
    },
    "eventHubName": {
      "value": "my-eventhub"
    }
  }
}
```

Diagnostic setting for Azure Data Explorer

The following sample creates a diagnostic setting for an Azure Data Explorer cluster by adding a resource of type `Microsoft.Kusto/clusters/providers/diagnosticSettings` to the template.

Template file

Bicep

Bicep

```
param clusterName string
param settingName string
param workspaceId string
param storageAccountId string
param eventHubAuthorizationRuleId string
param eventHubName string

resource cluster 'Microsoft.Kusto/clusters@2022-02-01' existing = {
  name: clusterName
}

resource setting 'Microsoft.Insights/diagnosticSettings@2021-05-01-
preview' = {
  name: settingName
  scope: cluster
  properties: {
    workspaceId: workspaceId
    storageAccountId: storageAccountId
    eventHubAuthorizationRuleId: eventHubAuthorizationRuleId
    eventHubName: eventHubName
    metrics: []
    logs: [
      {
        category: 'Command'
        categoryGroup: null
        enabled: true
        retentionPolicy: {
          enabled: false
          days: 0
        }
      }
      {
        category: 'Query'
        categoryGroup: null
        enabled: true
        retentionPolicy: {
          enabled: false
          days: 0
        }
      }
      {
        category: 'Journal'
        categoryGroup: null
        enabled: true
        retentionPolicy: {
          enabled: false
          days: 0
```

```
          }
        }
        {
          category: 'SucceededIngestion'
          categoryGroup: null
          enabled: false
          retentionPolicy: {
            enabled: false
            days: 0
          }
        }
        {
          category: 'FailedIngestion'
          categoryGroup: null
          enabled: false
          retentionPolicy: {
            enabled: false
            days: 0
          }
        }
        {
          category: 'IngestionBatching'
          categoryGroup: null
          enabled: false
          retentionPolicy: {
            enabled: false
            days: 0
          }
        }
        {
          category: 'TableUsageStatistics'
          categoryGroup: null
          enabled: false
          retentionPolicy: {
            enabled: false
            days: 0
          }
        }
        {
          category: 'TableDetails'
          categoryGroup: null
          enabled: false
          retentionPolicy: {
            enabled: false
            days: 0
          }
        }
      ]
    }
  }
```

Parameter file

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "clusterName": {
      "value": "kustoClusterName"
    },
    "diagnosticSettingName": {
      "value": "A new Diagnostic Settings configuration"
    },
    "workspaceId": {
      "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourcegroups/MyResourceGroup/providers/microsoft.operationalinsights/workspaces/MyWorkspace"
    },
    "storageAccountId": {
      "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/MyResourceGroup/providers/Microsoft.Storage/storageAccounts/mystorageaccount"
    },
    "eventHubAuthorizationRuleId": {
      "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/MyResourceGroup/providers/Microsoft.EventHub/namespaces/MyNameSpace/authorizationrules/RootManageSharedAccessKey"
    },
    "eventHubName": {
      "value": "myEventhub"
    }
  }
}
```

Template file: Enabling the 'audit' category group

Bicep

Bicep

```
param clusterName string
param settingName string
param workspaceId string
param storageAccountId string
param eventHubAuthorizationRuleId string
param eventHubName string

resource cluster 'Microsoft.Kusto/clusters@2022-02-01' existing = {
  name: clusterName
}
```

```
resource setting 'Microsoft.Insights/diagnosticSettings@2021-05-01-preview' = {
  name: settingName
  scope: cluster
  properties: {
    workspaceId: workspaceId
    storageAccountId: storageAccountId
    eventHubAuthorizationRuleId: eventHubAuthorizationRuleId
    eventHubName: eventHubName
    logs: [
      {
        category: null
        categoryGroup: 'audit'
        enabled: true
        retentionPolicy: {
          enabled: false
          days: 0
        }
      }
    ]
  }
}
```

Diagnostic setting for Azure Key Vault

The following sample creates a diagnostic setting for an instance of Azure Key Vault by adding a resource of type `Microsoft.KeyVault/vaults/providers/diagnosticSettings` to the template.

> **Important**
>
> For Azure Key Vault, the event hub must be in the same region as the key vault.

Template file

Bicep

Bicep

```
@description('The name of the diagnostic setting.')
param settingName string

@description('The name of the key vault.')
param vaultName string
```

```
@description('The resource Id of the workspace.')
param workspaceId string

@description('The resource Id of the storage account.')
param storageAccountId string

@description('The resource Id for the event hub authorization rule.')
param eventHubAuthorizationRuleId string

@description('The name of the event hub.')
param eventHubName string

resource vault 'Microsoft.KeyVault/vaults@2021-11-01-preview' existing =
{
  name: vaultName
}

resource setting 'Microsoft.Insights/diagnosticSettings@2021-05-01-
preview' = {
  name: settingName
  scope: vault
  properties: {
    workspaceId: workspaceId
    storageAccountId: storageAccountId
    eventHubAuthorizationRuleId: eventHubAuthorizationRuleId
    eventHubName: eventHubName
    logs: [
      {
        category: 'AuditEvent'
        enabled: true
      }
    ]
    metrics: [
      {
        category: 'AllMetrics'
        enabled: true
      }
    ]
  }
}
```

Parameter file

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
```

```
    "settingName": {
        "value": "Send to all locations"
    },
    "vaultName": {
      "value": "MyVault"
    },
    "workspaceId": {
      "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourcegroups/MyResourceGroup/providers/microsoft.operationali
nsights/workspaces/MyWorkspace"
    },
    "storageAccountId": {
      "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourceGroups/MyResourceGroup/providers/Microsoft.Storage/stor
ageAccounts/mystorageaccount"
    },
    "eventHubAuthorizationRuleId": {
      "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourceGroups/MyResourceGroup/providers/Microsoft.EventHub/nam
espaces/MyNameSpace/authorizationrules/RootManageSharedAccessKey"
    },
    "eventHubName": {
      "value": "my-eventhub"
    }
  }
}
```

Diagnostic setting for Azure SQL Database

The following sample creates a diagnostic setting for an instance of Azure SQL Database by adding a resource of type `microsoft.sql/servers/databases/providers/diagnosticSettings` to the template.

Template file

Bicep

Bicep

```
@description('The name of the diagnostic setting.')
param settingName string

@description('The name of the Azure SQL database server.')
param serverName string

@description('The name of the SQL database.')
param dbName string
```

```
@description('The resource Id of the workspace.')
param workspaceId string

@description('The resource Id of the storage account.')
param storageAccountId string

@description('The resource Id of the event hub authorization rule.')
param eventHubAuthorizationRuleId string

@description('The name of the event hub.')
param eventHubName string

resource dbServer 'Microsoft.Sql/servers@2021-11-01-preview' existing =
{
  name: serverName
}

resource db 'Microsoft.Sql/servers/databases@2021-11-01-preview'
existing = {
  parent: dbServer
  name: dbName
}

resource setting 'Microsoft.Insights/diagnosticSettings@2021-05-01-
preview' = {
  name: settingName
  scope: db
  properties: {
    workspaceId: workspaceId
    storageAccountId: storageAccountId
    eventHubAuthorizationRuleId: eventHubAuthorizationRuleId
    eventHubName: eventHubName
    logs: [
      {
        category: 'SQLInsights'
        enabled: true
      }
      {
        category: 'AutomaticTuning'
        enabled: true
      }
      {
        category: 'QueryStoreRuntimeStatistics'
        enabled: true
      }
      {
        category: 'QueryStoreWaitStatistics'
        enabled: true
      }
      {
        category: 'Errors'
        enabled: true
      }
      {
        category: 'DatabaseWaitStatistics'
```

```
          enabled: true
        }
        {
          category: 'Timeouts'
          enabled: true
        }
        {
          category: 'Blocks'
          enabled: true
        }
        {
          category: 'Deadlocks'
          enabled: true
        }
      ]
      metrics: [
        {
          category: 'Basic'
          enabled: true
        }
        {
          category: 'InstanceAndAppAdvanced'
          enabled: true
        }
        {
          category: 'WorkloadManagement'
          enabled: true
        }
      ]
    }
  }
```

Parameter file

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "settingName": {
        "value": "Send to all locations"
    },
    "serverName": {
      "value": "MySqlServer"
    },
    "dbName": {
      "value": "MySqlDb"
    },
    "workspaceId": {
```

```
      "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourcegroups/MyResourceGroup/providers/microsoft.operationali
nsights/workspaces/MyWorkspace"
    },
    "storageAccountId": {
      "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourceGroups/MyResourceGroup/providers/Microsoft.Storage/stor
ageAccounts/mystorageaccount"
    },
    "eventHubAuthorizationRuleId": {
      "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourceGroups/MyResourceGroup/providers/Microsoft.EventHub/nam
espaces/MyNameSpace/authorizationrules/RootManageSharedAccessKey"
    },
    "eventHubName": {
      "value": "my-eventhub"
    }
  }
}
```

Diagnostic setting for Azure SQL Managed Instance

The following sample creates a diagnostic setting for an instance of Azure SQL Managed Instance by adding a resource of type `microsoft.sql/managedInstances/providers/diagnosticSettings` to the template.

Template file

Bicep

Bicep

```
param sqlManagedInstanceName string
param diagnosticSettingName string
param diagnosticWorkspaceId string
param storageAccountId string
param eventHubAuthorizationRuleId string
param eventHubName string

resource instance 'Microsoft.Sql/managedInstances@2021-11-01-preview'
existing = {
  name: sqlManagedInstanceName
}

resource setting 'Microsoft.Insights/diagnosticSettings@2021-05-01-
preview' = {
```

```
    name: diagnosticSettingName
    scope: instance
    properties: {
      workspaceId: diagnosticWorkspaceId
      storageAccountId: storageAccountId
      eventHubAuthorizationRuleId: eventHubAuthorizationRuleId
      eventHubName: eventHubName
      logs: [
        {
          category: 'ResourceUsageStats'
          enabled: true
        }
        {
          category: 'DevOpsOperationsAudit'
          enabled: true
        }
        {
          category: 'SQLSecurityAuditEvents'
          enabled: true
        }
      ]
    }
  }
```

Parameter file

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "sqlManagedInstanceName": {
        "value": "MyInstanceName"
    },
    "diagnosticSettingName": {
        "value": "Send to all locations"
    },
    "diagnosticWorkspaceId": {
        "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourcegroups/MyResourceGroup/providers/microsoft.operationali
nsights/workspaces/MyWorkspace"
    },
    "storageAccountId": {
        "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourceGroups/MyResourceGroup/providers/Microsoft.Storage/stor
ageAccounts/mystorageaccount"
    },
    "eventHubAuthorizationRuleId": {
        "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
```

```
xxxxxxxxxxxx/resourceGroups/MyResourceGroup/providers/Microsoft.EventHub/nam
espaces/MyNameSpace/authorizationrules/RootManageSharedAccessKey"
    },
    "eventHubName": {
        "value": "myEventhub"
    }
  }
}
```

Diagnostic setting for a managed instance of Azure SQL Database

The following sample creates a diagnostic setting for a managed instance of Azure SQL Database by adding a resource of type `microsoft.sql/managedInstances/databases/providers/diagnosticSettings` to the template.

Template file

Bicep

Bicep

```
param sqlManagedInstanceName string
param sqlManagedDatabaseName string
param diagnosticSettingName string
param diagnosticWorkspaceId string
param storageAccountId string
param eventHubAuthorizationRuleId string
param eventHubName string

resource dbInstance 'Microsoft.Sql/managedInstances@2021-11-01-preview'
existing = {
  name:sqlManagedInstanceName
}

resource db 'Microsoft.Sql/managedInstances/databases@2021-11-01-
preview' existing = {
  name: sqlManagedDatabaseName
  parent: dbInstance
}

resource setting 'Microsoft.Insights/diagnosticSettings@2021-05-01-
preview' = {
  name: diagnosticSettingName
  scope: db
  properties: {
```

```
      workspaceId: diagnosticWorkspaceId
      storageAccountId: storageAccountId
      eventHubAuthorizationRuleId: eventHubAuthorizationRuleId
      eventHubName: eventHubName
      logs: [
        {
          category: 'SQLInsights'
          enabled: true
        }
        {
          category: 'QueryStoreRuntimeStatistics'
          enabled: true
        }
        {
          category: 'QueryStoreWaitStatistics'
          enabled: true
        }
        {
          category: 'Errors'
          enabled: true
        }
      ]
    }
  }
```

Parameter file

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "sqlManagedInstanceName": {
      "value": "MyInstanceName"
    },
    "sqlManagedDatabaseName": {
      "value": "MyManagedDatabaseName"
    },
    "diagnosticSettingName": {
      "value": "Send to all locations"
    },
    "diagnosticWorkspaceId": {
      "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourcegroups/MyResourceGroup/providers/microsoft.operationali
nsights/workspaces/MyWorkspace"
    },
    "storageAccountId": {
      "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourceGroups/MyResourceGroup/providers/Microsoft.Storage/stor
```

```
ageAccounts/mystorageaccount"
    },
    "eventHubAuthorizationRuleId": {
      "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourceGroups/MyResourceGroup/providers/Microsoft.EventHub/nam
espaces/MyNameSpace/authorizationrules/RootManageSharedAccessKey"
    },
    "eventHubName": {
      "value": "myEventhub"
    }
  }
}
```

Diagnostic setting for Recovery Services vault

The following sample creates a diagnostic setting for an Azure Recovery Services vault by adding a resource of type `microsoft.recoveryservices/vaults/providers/diagnosticSettings` to the template. This example specifies the collection mode as described in Azure resource logs. Specify `Dedicated` or `AzureDiagnostics` for the `logAnalyticsDestinationType` property.

Template file

Bicep

Bicep

```
param recoveryServicesName string
param settingName string
param workspaceId string
param storageAccountId string
param eventHubAuthorizationRuleId string
param eventHubName string

resource vault 'Microsoft.RecoveryServices/vaults@2021-08-01' existing =
{
  name: recoveryServicesName
}

resource setting 'Microsoft.Insights/diagnosticSettings@2021-05-01-
preview' = {
  name: settingName
  scope: vault
  properties: {
    workspaceId: workspaceId
    storageAccountId: storageAccountId
    eventHubAuthorizationRuleId: eventHubAuthorizationRuleId
    eventHubName: eventHubName
```

```
    logs: [
      {
        category: 'AzureBackupReport'
        enabled: false
      }
      {
        category: 'CoreAzureBackup'
        enabled: true
      }
      {
        category: 'AddonAzureBackupJobs'
        enabled: true
      }
      {
        category: 'AddonAzureBackupAlerts'
        enabled: true
      }
      {
        category: 'AddonAzureBackupPolicy'
        enabled: true
      }
      {
        category: 'AddonAzureBackupStorage'
        enabled: true
      }
      {
        category: 'AddonAzureBackupProtectedInstance'
        enabled: true
      }
      {
        category: 'AzureSiteRecoveryJobs'
        enabled: false
      }
      {
        category: 'AzureSiteRecoveryEvents'
        enabled: false
      }
      {
        category: 'AzureSiteRecoveryReplicatedItems'
        enabled: false
      }
      {
        category: 'AzureSiteRecoveryReplicationStats'
        enabled: false
      }
      {
        category: 'AzureSiteRecoveryRecoveryPoints'
        enabled: false
      }
      {
        category: 'AzureSiteRecoveryReplicationDataUploadRate'
        enabled: false
      }
      {
        category: 'AzureSiteRecoveryProtectedDiskDataChurn'
```

```
          enabled: false
        }
      ]
      logAnalyticsDestinationType: 'Dedicated'
    }
  }
```

Parameter file

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "settingName": {
      "value": "Send to all locations"
    },
    "recoveryServicesName": {
      "value": "my-vault"
    },
    "workspaceId": {
      "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourcegroups/MyResourceGroup/providers/microsoft.operationalinsights/workspaces/MyWorkspace"
    },
    "storageAccountId": {
      "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/MyResourceGroup/providers/Microsoft.Storage/storageAccounts/mystorageaccount"
    },
    "eventHubAuthorizationRuleId": {
      "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/MyResourceGroup/providers/Microsoft.EventHub/namespaces/MyNameSpace/authorizationrules/RootManageSharedAccessKey"
    },
    "eventHubName": {
      "value": "my-eventhub"
    }
  }
}
```

Diagnostic setting for a Log Analytics workspace

The following sample creates a diagnostic setting for a Log Analytics workspace by adding a resource of type `Microsoft.OperationalInsights/workspaces/providers/diagnosticSettings` to the template. This example sends audit data about queries executed in the workspace to the same workspace.

Template file

Bicep

Bicep

```
param workspaceName string
param settingName string
param workspaceId string
param storageAccountId string
param eventHubAuthorizationRuleId string
param eventHubName string

resource workspace 'Microsoft.OperationalInsights/workspaces@2021-12-01-
preview' existing = {
  name: workspaceName
}
resource setting 'Microsoft.Insights/diagnosticSettings@2021-05-01-
preview' = {
  name: settingName
  scope: workspace
  properties: {
    workspaceId: workspaceId
    storageAccountId: storageAccountId
    eventHubAuthorizationRuleId: eventHubAuthorizationRuleId
    eventHubName: eventHubName
    logs: [
      {
        category: 'Audit'
        enabled: true
      }
    ]
  }
}
```

Parameter file

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
```

```
01/deployment Parameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "settingName": {
        "value": "Send to all locations"
    },
    "workspaceName": {
      "value": "MyWorkspace"
    },
    "workspaceId": {
      "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourcegroups/MyResourceGroup/providers/microsoft.operationali
nsights/workspaces/MyWorkspace"
    },
    "storageAccountId": {
      "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourceGroups/MyResourceGroup/providers/Microsoft.Storage/stor
ageAccounts/mystorageaccount"
    },
    "eventHubAuthorizationRuleId": {
      "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourceGroups/MyResourceGroup/providers/Microsoft.EventHub/nam
espaces/MyNameSpace/authorizationrules/RootManageSharedAccessKey"
    },
    "eventHubName": {
      "value": "my-eventhub"
    }
  }
}
```

Diagnostic setting for Azure Storage

The following sample creates a diagnostic setting for each storage service endpoint that's available in the Azure Storage account. A setting is applied to each individual storage service that's available on the account. The storage services that are available depend on the type of storage account.

This template creates a diagnostic setting for a storage service in the account only if it exists for the account. For each available service, the diagnostic setting enables transaction metrics, and the collection of resource logs for read, write, and delete operations.

Template file

Bicep

main.bicep

Bicep

```
param storageAccountName string
param settingName string
param storageSyncName string
param workspaceId string

module nested './module.bicep' = {
  name: 'nested'
  params: {
    endpoints: reference(resourceId('Microsoft.Storage/storageAccounts',
storageAccountName), '2019-06-01', 'Full').properties.primaryEndpoints
    settingName: settingName
    storageAccountName: storageAccountName
    storageSyncName: storageSyncName
    workspaceId: workspaceId
  }
}
```

module.bicep

Bicep

```
param endpoints object
param settingName string
param storageAccountName string
param storageSyncName string
param workspaceId string

var hasblob = contains(endpoints, 'blob')
var hastable = contains(endpoints, 'table')
var hasfile = contains(endpoints, 'file')
var hasqueue = contains(endpoints, 'queue')

resource storageAccount 'Microsoft.Storage/storageAccounts@2021-09-01'
existing = {
  name: storageAccountName
}

resource diagnosticSetting 'Microsoft.Insights/diagnosticSettings@2021-
05-01-preview' = {
  name: settingName
  scope: storageAccount
  properties: {
    workspaceId: workspaceId
    storageAccountId: resourceId('Microsoft.Storage/storageAccounts',
storageSyncName)
    metrics: [
      {
        category: 'Transaction'
        enabled: true
      }
    ]
```

```
  }
}

resource blob 'Microsoft.Storage/storageAccounts/blobServices@2021-09-
01' existing = {
  name:'default'
  parent:storageAccount
}

resource blobSetting 'Microsoft.Insights/diagnosticSettings@2021-05-01-
preview' = if (hasblob) {
  name: settingName
  scope: blob
  properties: {
    workspaceId: workspaceId
    storageAccountId: resourceId('Microsoft.Storage/storageAccounts',
storageSyncName)
    logs: [
      {
        category: 'StorageRead'
        enabled: true
      }
      {
        category: 'StorageWrite'
        enabled: true
      }
      {
        category: 'StorageDelete'
        enabled: true
      }
    ]
    metrics: [
      {
        category: 'Transaction'
        enabled: true
      }
    ]
  }
}

resource table 'Microsoft.Storage/storageAccounts/tableServices@2021-09-
01' existing = {
  name:'default'
  parent:storageAccount
}

resource tableSetting 'Microsoft.Insights/diagnosticSettings@2021-05-01-
preview' = if (hastable) {
  name: settingName
  scope: table
  properties: {
    workspaceId: workspaceId
    storageAccountId: resourceId('Microsoft.Storage/storageAccounts',
storageSyncName)
    logs: [
```

```
      {
        category: 'StorageRead'
        enabled: true
      }
      {
        category: 'StorageWrite'
        enabled: true
      }
      {
        category: 'StorageDelete'
        enabled: true
      }
    ]
    metrics: [
      {
        category: 'Transaction'
        enabled: true
      }
    ]
  }
}

resource file 'Microsoft.Storage/storageAccounts/fileServices@2021-09-
01' existing = {
  name:'default'
  parent:storageAccount
}

resource fileSetting 'Microsoft.Insights/diagnosticSettings@2021-05-01-
preview' = if (hasfile) {
  name: settingName
  scope: file
  properties: {
    workspaceId: workspaceId
    storageAccountId: resourceId('Microsoft.Storage/storageAccounts',
storageSyncName)
    logs: [
      {
        category: 'StorageRead'
        enabled: true
      }
      {
        category: 'StorageWrite'
        enabled: true
      }
      {
        category: 'StorageDelete'
        enabled: true
      }
    ]
    metrics: [
      {
        category: 'Transaction'
        enabled: true
      }
```

```
    ]
  }
}

resource queue 'Microsoft.Storage/storageAccounts/queueServices@2021-09-01' existing = {
  name:'default'
  parent:storageAccount
}


resource queueSetting 'Microsoft.Insights/diagnosticSettings@2021-05-01-preview' = if (hasqueue) {
  name: settingName
  scope: queue
  properties: {
    workspaceId: workspaceId
    storageAccountId: resourceId('Microsoft.Storage/storageAccounts', storageSyncName)
    logs: [
      {
        category: 'StorageRead'
        enabled: true
      }
      {
        category: 'StorageWrite'
        enabled: true
      }
      {
        category: 'StorageDelete'
        enabled: true
      }
    ]
    metrics: [
      {
        category: 'Transaction'
        enabled: true
      }
    ]
  }
}
```

Parameter file

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
```

```
    "storageAccountName": {
      "value": "mymonitoredstorageaccount"
    },
    "settingName": {
      "value": "Send to all locations"
    },
    "storageSyncName": {
      "value": "mystorageaccount"
    },
    "workspaceId": {
      "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourcegroups/MyResourceGroup/providers/microsoft.operationali
nsights/workspaces/MyWorkspace"
    }
  }
}
```

Next steps

- Get other sample templates for Azure Monitor
- Learn more about diagnostic settings

Quickstart: Create a policy assignment to identify non-compliant resources by using an ARM template

Article • 04/25/2023

The first step in understanding compliance in Azure is to identify the status of your resources. This quickstart steps you through the process of using an Azure Resource Manager template (ARM template) to create a policy assignment that identifies virtual machines that aren't using managed disks, and flags them as *non-compliant* to the policy assignment.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

In this quickstart, you create a policy assignment and assign a built-in policy definition called *Audit VMs that do not use managed disks*. For a partial list of available built-in policies, see Azure Policy samples.

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
```

```
  "parameters": {
    "policyAssignmentName": {
      "type": "string",
      "defaultValue": "[guid(parameters('policyDefinitionID'),
resourceGroup().name)]",
      "metadata": {
        "description": "Specifies the name of the policy assignment, can be
used defined or an idempotent name as the defaultValue provides."
      }
    },
    "policyDefinitionID": {
      "type": "string",
      "metadata": {
        "description": "Specifies the ID of the policy definition or policy
set definition being assigned."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Authorization/policyAssignments",
      "name": "[parameters('policyAssignmentName')]",
      "apiVersion": "2019-09-01",
      "properties": {
        "scope": "
[subscriptionResourceId('Microsoft.Resources/resourceGroups',
resourceGroup().name)]",
        "policyDefinitionId": "[parameters('policyDefinitionID')]"
      }
    }
  ]
}
```

The resource defined in the template is:

- Microsoft.Authorization/policyAssignments

Deploy the template

> **Note**
>
> Azure Policy service is free. For more information, see **Overview of Azure Policy**.

1. Select the following image to sign in to the Azure portal and open the template:

   Deploy to Azure

2. Select or enter the following values:

Name	Value
Subscription	Select your Azure subscription.
Resource group	Select **Create new**, specify a name, and then select **OK**. In the screenshot, the resource group name is *mypolicyquickstart<Date in MMDD>rg*.
Location	Select a region. For example, **Central US**.
Policy Assignment Name	Specify a policy assignment name. You can use the policy definition display if you want. For example, *Audit VMs that do not use managed disks*.
Resource Group Name	Specify a resource group name where you want to assign the policy to. In this quickstart, use the default value **[resourceGroup().name]**. **resourceGroup()** is a template function that retrieves the resource group.
Policy Definition ID	Specify **/providers/Microsoft.Authorization/policyDefinitions/0a914e76-4921-4c19-b460-a2d36003525a**.
I agree to the terms and conditions stated above	(Select)

3. Select **Purchase**.

Some other resources:

- To find more samples templates, see Azure Quickstart Template.
- To see the template reference, go to Azure template reference.
- To learn how to develop ARM templates, see Azure Resource Manager documentation.
- To learn subscription-level deployment, see Create resource groups and resources at the subscription level.

Validate the deployment

Select **Compliance** in the left side of the page. Then locate the *Audit VMs that do not use managed disks* policy assignment you created.

If there are any existing resources that aren't compliant with this new assignment, they appear under **Non-compliant resources**.

For more information, see How compliance works.

Clean up resources

To remove the assignment created, follow these steps:

1. Select **Compliance** (or **Assignments**) in the left side of the Azure Policy page and locate the *Audit VMs that do not use managed disks* policy assignment you created.

2. Right-click the *Audit VMs that do not use managed disks* policy assignment and select **Delete assignment**.

Next steps

In this quickstart, you assigned a built-in policy definition to a scope and evaluated its compliance report. The policy definition validates that all the resources in the scope are compliant and identifies which ones aren't.

To learn more about assigning policies to validate that new resources are compliant, continue to the tutorial for:

Creating and managing policies

Quickstart: Create a dashboard in the Azure portal by using an ARM template

Article • 04/09/2023

A dashboard in the Azure portal is a focused and organized view of your cloud resources. This quickstart focuses on the process of deploying an Azure Resource Manager template (ARM template) to create a dashboard. The dashboard shows the performance of a virtual machine (VM), and some static information and links.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

- An Azure account with an active subscription. Create an account for free.
- Azure PowerShell or Azure CLI.
- A virtual machine. The dashboard you create in the next part of this quickstart requires an existing VM. Create a VM by following these steps.

  1. In the Azure portal, select **Cloud Shell** from the global controls at the top of the page.

  2. In the **Cloud Shell** window, select **PowerShell**.

  3. Copy the following command and enter it at the command prompt to create a resource group.

```
PowerShell

New-AzResourceGroup -Name SimpleWinVmResourceGroup -Location
EastUS
```

4. Next, copy the following command and enter it at the command prompt to create a VM in your new resource group.

```
PowerShell

New-AzVm `
    -ResourceGroupName "SimpleWinVmResourceGroup" `
    -Name "myVM1" `
    -Location "East US"
```

5. Enter a username and password for the VM. This is a new user name and password; it's not, for example, the account you use to sign in to Azure. For more information, see username requirements and password requirements.

After the VM has been created, move on to the next section.

Review the template

The template used in this quickstart is from Azure Quickstart Templates. The template for this article is too long to show here. To view the template, see azuredeploy.json. The template defines one Azure resource, a dashboard that displays data about the VM you created.

Deploy the template

This example uses the Azure portal to deploy the template. You can also use other methods to deploy ARM templates, such as Azure PowerShell, Azure CLI, or REST API.

1. Select the following image to sign in to Azure and open a template.

   Deploy to Azure

2. Select or enter the following values, then select **Review + create**.

Unless it's specified, use the default values to create the dashboard.

- **Subscription**: select the Azure subscription.
- **Resource group**: select **SimpleWinVmResourceGroup**.
- **Location**: If not automatically selected, choose **East US**.
- **Virtual Machine Name**: enter **myVM1**.
- **Virtual Machine Resource Group**: enter **SimpleWinVmResourceGroup**.

3. Select **Create**. You'll see a notification confirming when the dashboard has been deployed successfully.

Review deployed resources

Verify that you can see data about your virtual machine in the Azure portal dashboard.

1. In the Azure portal menu, select **Dashboard**.

2. On the dashboard page, select **Simple VM Dashboard**.

3. Review the dashboard, which should look similar to the one shown here. While some of the content is static, there are also charts that show the performance of the VM you created at the beginning.

Clean up resources

If you want to remove the VM and associated dashboard, delete the resource group that contains them.

1. In the Azure portal, search for **SimpleWinVmResourceGroup**, then select it in the search results.
2. On the **SimpleWinVmResourceGroup** page, select **Delete resource group**, enter the resource group name to confirm, then select **Delete**.

> **Caution**
>
> Deleting a resource group will delete all of the resources contained within it. If the resource group contains additional resources aside from your virtual machine and dashboard, those resources will also be deleted.

Next steps

For more information about dashboards in the Azure portal, see:

Create and share dashboards in the Azure portal

Quickstart: Create a shared query by using an ARM template

Article • 04/13/2023

Resource Graph queries can be saved as a *private query* or a *shared query*. A private query is saved to the individuals portal profile and isn't visible to others. A shared query is a Resource Manager object that can be shared with others through permissions and role-based access. A shared query provides common and consistent execution of resource discovery. This quickstart uses an Azure Resource Manager template (ARM template) to create a shared query.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

In this quickstart, you create a shared query called *Count VMs by OS*. To try this query in SDK or in portal with Resource Graph Explorer, see Samples - Count virtual machines by OS type.

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
```

```
    "_generator": {
      "name": "bicep",
      "version": "0.6.1.6515",
      "templateHash": "3085789505121407565"
    }
  },
  "parameters": {
    "queryName": {
      "type": "string",
      "defaultValue": "Count VMs by OS",
      "metadata": {
        "description": "The name of the shared query."
      }
    },
    "queryCode": {
      "type": "string",
      "defaultValue": "Resources | where type =~
'Microsoft.Compute/virtualMachines' | summarize count() by
tostring(properties.storageProfile.osDisk.osType)",
      "metadata": {
        "description": "The Azure Resource Graph query to be saved to the
shared query."
      }
    },
    "queryDescription": {
      "type": "string",
      "defaultValue": "This shared query counts all virtual machine
resources and summarizes by the OS type.",
      "metadata": {
        "description": "The description of the saved Azure Resource Graph
query."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.ResourceGraph/queries",
      "apiVersion": "2018-09-01-preview",
      "name": "[parameters('queryName')]",
      "location": "global",
      "properties": {
        "query": "[parameters('queryCode')]",
        "description": "[parameters('queryDescription')]"
      }
    }
  ]
}
```

The resource defined in the template is:

- Microsoft.ResourceGraph/queries

Deploy the template

> **Note**
>
> Azure Resource Graph service is free. For more information, see **Overview of Azure Resource Graph**.

1. Select the following image to sign in to the Azure portal and open the template:

   Deploy to Azure

2. Select or enter the following values:

Name	Value
Subscription	Select your Azure subscription.
Resource group	Select **Create new**, specify a name, and then select **OK**.
Location	Select a region. For example, **Central US**.
Query name	Leave the default value: **Count VMs by OS**.
Query code	Leave the default value: `Resources | where type =~ 'Microsoft.Compute/virtualMachines' | summarize count() by tostring(properties.storageProfile.osDisk.osType)`
Query description	Leave the default value: **This shared query counts all virtual machine resources and summarizes by the OS type.**
I agree to the terms and conditions stated above	(Select)

3. Select **Purchase**.

Some other resources:

- To find more samples templates, see Azure Quickstart Template.
- To see the template reference, go to Azure template reference.
- To learn how to develop ARM templates, see Azure Resource Manager documentation.
- To learn subscription-level deployment, see Create resource groups and resources at the subscription level.

Validate the deployment

To run the new shared query, follow these steps:

1. From the portal search bar, search for **Resource Graph queries** and select it.
2. Select the shared query named **Count VMs by OS**, then select the **Results** tab on the **Overview** page.

The shared query can also be opened from Resource Graph Explorer:

1. From the portal search bar, search for **Resource Graph Explorer** and select it.
2. Select the **Open a query** button.
3. Change **Type** to *Shared queries*. If you don't see the **Count VMs by OS** in the list, use the filter box to limit the results. Once the **Count VMs by OS** shared query is visible, select its name.
4. Once the query is loaded, select the **Run query** button. Results are displayed in the **Results** tab.

Clean up resources

To remove the shared query created, follow these steps:

1. From the portal search bar, search for **Resource Graph queries** and select it.
2. Set the check box next to the shared query named **Count VMs by OS**.
3. Select the **Delete** button along the top of the page.

Next steps

In this quickstart, you created a Resource Graph shared query.

To learn more about shared queries, continue to the tutorial for:

Manage queries in Azure portal

Create or update Azure custom roles using an ARM template

Article • 04/13/2023

If the Azure built-in roles don't meet the specific needs of your organization, you can create your own custom roles. This article describes how to create or update a custom role using an Azure Resource Manager template (ARM template).

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

To create a custom role, you specify a role name, permissions, and where the role can be used. In this article, you create a role named *Custom Role - RG Reader* with resource permissions that can be assigned at a subscription scope or lower.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

To create a custom role, you must have:

- Permissions to create custom roles, such as Owner or User Access Administrator.

You must use the following version:

- `2018-07-01` or later

For more information, see API versions of Azure RBAC REST APIs.

Review the template

The template used in this article is from Azure Quickstart Templates. The template has four parameters and a resources section. The four parameters are:

- Array of actions with a default value of `["Microsoft.Resources/subscriptions/resourceGroups/read"]`.
- Array of `notActions` with an empty default value.
- Role name with a default value of `Custom Role - RG Reader`.
- Role description with a default value of `Subscription Level Deployment of a Role Definition`.

The scope where this custom role can be assigned is set to the current subscription.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2018-05-01/subscriptionDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "2227781763411200690"
    }
  },
  "parameters": {
    "actions": {
      "type": "array",
      "defaultValue": [
        "Microsoft.Resources/subscriptions/resourceGroups/read"
      ],
      "metadata": {
        "description": "Array of actions for the roleDefinition"
      }
    },
    "notActions": {
      "type": "array",
      "defaultValue": [],
      "metadata": {
        "description": "Array of notActions for the roleDefinition"
      }
    },
    "roleName": {
      "type": "string",
      "defaultValue": "Custom Role - RG Reader",
      "metadata": {
        "description": "Friendly name of the role definition"
      }
    },
    "roleDescription": {
      "type": "string",
      "defaultValue": "Subscription Level Deployment of a Role Definition",
      "metadata": {
        "description": "Detailed description of the role definition"
      }
```

```
    }
  },
  "variables": {
    "roleDefName": "[guid(subscription().id, string(parameters('actions')),
string(parameters('notActions')))]"
  },
  "resources": [
    {
      "type": "Microsoft.Authorization/roleDefinitions",
      "apiVersion": "2018-07-01",
      "name": "[variables('roleDefName')]",
      "properties": {
        "roleName": "[parameters('roleName')]",
        "description": "[parameters('roleDescription')]",
        "type": "customRole",
        "permissions": [
          {
            "actions": "[parameters('actions')]",
            "notActions": "[parameters('notActions')]"
          }
        ],
        "assignableScopes": [
          "[subscription().id]"
        ]
      }
    }
  ]
}
```

The resource defined in the template is:

- Microsoft.Authorization/roleDefinitions

Deploy the template

Follow these steps to deploy the previous template.

1. Sign in to the Azure portal.
2. Open Azure Cloud Shell for PowerShell.
3. Copy and paste the following script into Cloud Shell.

Azure PowerShell

```
$location = Read-Host -Prompt "Enter a location (i.e. centralus)"
[string[]]$actions = Read-Host -Prompt "Enter actions as a comma-
separated list (i.e. action1,action2)"
$actions = $actions.Split(',')
$templateUri = "https://raw.githubusercontent.com/Azure/azure-
quickstart-templates/master/subscription-deployments/create-role-
```

```
def/azuredeploy.json"
New-AzDeployment -Location $location -TemplateUri $templateUri -actions
$actions
```

4. Enter a location for the deployment such as `centralus`.

5. Enter a list of actions for the custom role as a comma-separated list such as `Microsoft.Resources/resources/read,Microsoft.Resources/subscriptions/resourceGroups/read`.

6. If necessary, press Enter to run the `New-AzDeployment` command.

   The New-AzDeployment command deploys the template to create the custom role.

   You should see output similar to the following:

Azure PowerShell

```
PS> New-AzDeployment -Location $location -TemplateUri $templateUri -
actions $actions

Id                      :
/subscriptions/{subscriptionId}/providers/Microsoft.Resources/deploymen
ts/azuredeploy
DeploymentName          : azuredeploy
Location                : centralus
ProvisioningState       : Succeeded
Timestamp               : 6/25/2020 8:08:32 PM
Mode                    : Incremental
TemplateLink            :
                          Uri            :
https://raw.githubusercontent.com/Azure/azure-quickstart-
templates/master/subscription-deployments/create-role-
def/azuredeploy.json
                          ContentVersion : 1.0.0.0

Parameters              :
                          Name               Type
Value
                          =================  ========================
==========
                          actions            Array
[
                            "Microsoft.Resources/resources/read",

"Microsoft.Resources/subscriptions/resourceGroups/read"
                          ]
                          notActions         Array
[]
                          roleName           String
```

```
Custom Role - RG Reader
                          roleDescription    String
Subscription Level Deployment of a Role Definition

Outputs                 :
DeploymentDebugLogLevel :
```

Review deployed resources

Follow these steps to verify that the custom role was created.

1. Run the Get-AzRoleDefinition command to list the custom role.

   Azure PowerShell

   ```
   Get-AzRoleDefinition "Custom Role - RG Reader" | ConvertTo-Json
   ```

   You should see output similar to the following:

   Azure PowerShell

   ```
   {
     "Name": "Custom Role - RG Reader",
     "Id": "11111111-1111-1111-1111-111111111111",
     "IsCustom": true,
     "Description": "Subscription Level Deployment of a Role Definition",
     "Actions": [
       "Microsoft.Resources/resources/read",
       "Microsoft.Resources/subscriptions/resourceGroups/read"
     ],
     "NotActions": [],
     "DataActions": [],
     "NotDataActions": [],
     "AssignableScopes": [
       "/subscriptions/{subscriptionId}"
     ]
   }
   ```

2. In the Azure portal, open your subscription.

3. In the left menu, select **Access control (IAM)**.

4. Select the **Roles** tab.

5. Set the **Type** list to **CustomRole**.

6. Verify that the **Custom Role - RG Reader** role is listed.

Update a custom role

Similar to creating a custom role, you can update an existing custom role by using a template. To update a custom role, you must specify the role you want to update.

Here are the changes you would need to make to the previous Quickstart template to update the custom role.

- Include the role ID as a parameter.

JSON

```
...
"roleDefName": {
  "type": "string",
  "metadata": {
    "description": "ID of the role definition"
  }
...
```

- Include the role ID parameter in the role definition.

JSON

```
...
"resources": [
  {
    "type": "Microsoft.Authorization/roleDefinitions",
    "apiVersion": "2022-04-01",
    "name": "[parameters('roleDefName')]",
    "properties": {
      ...
```

Here's an example of how to deploy the template.

Azure PowerShell

```
$location = Read-Host -Prompt "Enter a location (i.e. centralus)"
[string[]]$actions = Read-Host -Prompt "Enter actions as a comma-separated
list (i.e. action1,action2)"
$actions = $actions.Split(',')
$roleDefName = Read-Host -Prompt "Enter the role ID to update"
$templateFile = "rg-reader-update.json"
New-AzDeployment -Location $location -TemplateFile $templateFile -actions
$actions -roleDefName $roleDefName
```

Clean up resources

To remove the custom role, follow these steps.

1. Run the following command to remove the custom role.

   Azure PowerShell

   ```
   Get-AzRoleDefinition -Name "Custom Role - RG Reader" | Remove-
   AzRoleDefinition
   ```

2. Enter **Y** to confirm that you want to remove the custom role.

Next steps

- Understand Azure role definitions
- Quickstart: Assign an Azure role using an Azure Resource Manager template
- ARM template documentation

Quickstart: Assign an Azure role using an ARM template

Article • 03/10/2023

Azure role-based access control (Azure RBAC) is the way that you manage access to Azure resources. In this quickstart, you create a resource group and grant a user access to create and manage virtual machines in the resource group. This quickstart uses an Azure Resource Manager template (ARM template) to grant the access.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

To assign Azure roles and remove role assignments, you must have:

- If you don't have an Azure subscription, create a free account before you begin.
- `Microsoft.Authorization/roleAssignments/write` and `Microsoft.Authorization/roleAssignments/delete` permissions, such as User Access Administrator or Owner
- To assign a role, you must specify three elements: security principal, role definition, and scope. For this quickstart, the security principal is you or another user in your directory, the role definition is Virtual Machine Contributor, and the scope is a resource group that you specify.

Review the template

The template used in this quickstart is from Azure Quickstart Templates. The template has two parameters and a resources section. In the resources section, notice that it has the three elements of a role assignment: security principal, role definition, and scope.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "15014882513681156573"
    }
  },
  "parameters": {
    "roleDefinitionID": {
      "type": "string",
      "metadata": {
        "description": "Specifies the role definition ID used in the role assignment."
      }
    },
    "principalId": {
      "type": "string",
      "metadata": {
        "description": "Specifies the principal ID assigned to the role."
      }
    }
  },
  "variables": {
    "roleAssignmentName": "[guid(parameters('principalId'), parameters('roleDefinitionID'), resourceGroup().id)]"
  },
  "resources": [
    {
      "type": "Microsoft.Authorization/roleAssignments",
      "apiVersion": "2021-04-01-preview",
      "name": "[variables('roleAssignmentName')]",
      "properties": {
        "roleDefinitionId": "[resourceId('Microsoft.Authorization/roleDefinitions', parameters('roleDefinitionID'))]",
        "principalId": "[parameters('principalId')]"
      }
    }
  ]
}
```

The resource defined in the template is:

- Microsoft.Authorization/roleAssignments

Deploy the template

1. Sign in to the Azure portal.

2. Determine your email address that is associated with your Azure subscription. Or determine the email address of another user in your directory.

3. Open Azure Cloud Shell for PowerShell.

4. Copy and paste the following script into Cloud Shell.

   Azure PowerShell

   ```
   $resourceGroupName = Read-Host -Prompt "Enter a resource group name (i.e. ExampleGrouprg)"
   $emailAddress = Read-Host -Prompt "Enter an email address for a user in your directory"
   $location = Read-Host -Prompt "Enter a location (i.e. centralus)"

   $roleAssignmentName = New-Guid
   $principalId = (Get-AzAdUser -Mail $emailAddress).id
   $roleDefinitionId = (Get-AzRoleDefinition -name "Virtual Machine Contributor").id
   $templateUri = "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.authorization/rbac-builtinrole-resourcegroup/azuredeploy.json"

   New-AzResourceGroup -Name $resourceGroupName -Location $location
   New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName -TemplateUri $templateUri -roleDefinitionID $roleDefinitionId -principalId $principalId
   ```

5. Enter a resource group name such as ExampleGrouprg.

6. Enter an email address for yourself or another user in your directory.

7. Enter a location for the resource group such as centralus.

8. If necessary, press Enter to run the New-AzResourceGroupDeployment command.

   The New-AzResourceGroup command creates a new resource group and the New-AzResourceGroupDeployment command deploys the template to add the role assignment.

   You should see output similar to the following:

   Azure PowerShell

```
PS> New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName
-TemplateUri $templateUri -roleAssignmentName $roleAssignmentName -
roleDefinitionID $roleDefinitionId -principalId $principalId

DeploymentName          : azuredeploy
ResourceGroupName       : ExampleGrouprg
ProvisioningState       : Succeeded
Timestamp               : 5/22/2020 9:01:30 PM
Mode                    : Incremental
TemplateLink            :
                          Uri            :
https://raw.githubusercontent.com/Azure/azure-quickstart-
templates/master/quickstarts/microsoft.authorization/rbac-builtinrole-
resourcegroup/azuredeploy.json
                          ContentVersion : 1.0.0.0

Parameters              :
                          Name                   Type
Value
                          ====================
=========================  ==========
                          roleDefinitionID       String
9980e02c-c2be-4d73-94e8-173b1dc7cf3c
                          principalId            String
{principalId}

Outputs                 :
DeploymentDebugLogLevel :
```

Review deployed resources

1. In the Azure portal, open the resource group you created.
2. In the left menu, click **Access control (IAM)**.
3. Click the **Role assignments** tab.
4. Verify that the **Virtual Machine Contributor** role is assigned to the user you specified.

Clean up resources

To remove the role assignment and resource group you created, follow these steps.

1. Copy and paste the following script into Cloud Shell.

Azure PowerShell

```
$emailAddress = Read-Host -Prompt "Enter the email address of the user
with the role assignment to remove"
$resourceGroupName = Read-Host -Prompt "Enter the resource group name
to remove (i.e. ExampleGrouprg)"

$principalId = (Get-AzAdUser -Mail $emailAddress).id

Remove-AzRoleAssignment -ObjectId $principalId -RoleDefinitionName
"Virtual Machine Contributor" -ResourceGroupName $resourceGroupName
Remove-AzResourceGroup -Name $resourceGroupName
```

2. Enter the email address of the user with the role assignment to remove.

3. Enter the resource group name to remove such as ExampleGrouprg.

4. If necessary, press Enter to run the Remove-AzResourceGroup command.

5. Enter **Y** to confirm that you want to remove the resource group.

Next steps

Tutorial: Grant a user access to Azure resources using Azure PowerShell

Quickstart: Create activity log alerts on service notifications using an ARM template

Article • 03/10/2023

This article shows you how to set up activity log alerts for service health notifications by using an Azure Resource Manager template (ARM template).

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

Service health notifications are stored in the Azure activity log. Given the possibly large volume of information stored in the activity log, there is a separate user interface to make it easier to view and set up alerts on service health notifications.

You can receive an alert when Azure sends service health notifications to your Azure subscription. You can configure the alert based on:

- The class of service health notification (Service issues, Planned maintenance, Health advisories).
- The subscription affected.
- The service(s) affected.
- The region(s) affected.

> **Note**
>
> Service health notifications does not send an alert regarding resource health events.

You also can configure who the alert should be sent to:

- Select an existing action group.
- Create a new action group (that can be used for future alerts).

To learn more about action groups, see Create and manage action groups.

Prerequisites

- If you don't have an Azure subscription, create a free account before you begin.
- To run the commands from your local computer, install Azure CLI or the Azure PowerShell modules. For more information, see Install the Azure CLI and Install Azure PowerShell.

Review the template

The following template creates an action group with an email target and enables all service health notifications for the target subscription. Save this template as *CreateServiceHealthAlert.json*.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "actionGroups_name": {
      "type": "string",
      "defaultValue": "SubHealth"
    },
    "activityLogAlerts_name": {
      "type": "string",
      "defaultValue": "ServiceHealthActivityLogAlert"
    },
    "emailAddress": {
      "type": "string"
    }
  },
  "variables": {
    "alertScope": "[format('/subscriptions/{0}', subscription().subscriptionId)]"
  },
  "resources": [
    {
      "type": "microsoft.insights/actionGroups",
      "apiVersion": "2019-06-01",
      "name": "[parameters('actionGroups_name')]",
      "location": "Global",
      "properties": {
        "groupShortName": "[parameters('actionGroups_name')]",
        "enabled": true,
        "emailReceivers": [
          {
            "name": "[parameters('actionGroups_name')]",
            "emailAddress": "[parameters('emailAddress')]"
          }
        ],
        "smsReceivers": [],
```

```
        "webhookReceivers": []
      }
    },
    {
      "type": "microsoft.insights/activityLogAlerts",
      "apiVersion": "2017-04-01",
      "name": "[parameters('activityLogAlerts_name')]",
      "location": "Global",
      "properties": {
        "scopes": [
          "[variables('alertScope')]"
        ],
        "condition": {
          "allOf": [
            {
              "field": "category",
              "equals": "ServiceHealth"
            },
            {
              "field": "properties.incidentType",
              "equals": "Incident"
            }
          ]
        },
        "actions": {
          "actionGroups": [
            {
              "actionGroupId": "
[resourceId('microsoft.insights/actionGroups',
parameters('actionGroups_name'))]",
              "webhookProperties": {}
            }
          ]
        },
        "enabled": true
      },
      "dependsOn": [
        "[resourceId('microsoft.insights/actionGroups',
parameters('actionGroups_name'))]"
      ]
    }
  ]
}
```

The template defines two resources:

- Microsoft.Insights/actionGroups
- Microsoft.Insights/activityLogAlerts

Deploy the template

Deploy the template using any standard method for deploying an ARM template such as the following examples using CLI and PowerShell. Replace the sample values for **Resource Group** and **emailAddress** with appropriate values for your environment.

CLI

Azure CLI

```
az login
az deployment group create --name CreateServiceHealthAlert --resource-group my-resource-group --template-file CreateServiceHealthAlert.json --parameters emailAddress='user@contoso.com'
```

Validate the deployment

Verify that the workspace has been created using one of the following commands. Replace the sample values for **Resource Group** with the value you used above.

CLI

Azure CLI

```
az monitor activity-log alert show --resource-group my-resource-group --name ServiceHealthActivityLogAlert
```

Clean up resources

If you plan to continue working with subsequent quickstarts and tutorials, you might want to leave these resources in place. When no longer needed, delete the resource group, which deletes the alert rule and the related resources. To delete the resource group by using Azure CLI or Azure PowerShell

CLI

Azure CLI

```
az group delete --name my-resource-group
```

Next steps

- Learn about best practices for setting up Azure Service Health alerts.
- Learn how to setup mobile push notifications for Azure Service Health.
- Learn how to configure webhook notifications for existing problem management systems.
- Learn about service health notifications.
- Learn about notification rate limiting.
- Review the activity log alert webhook schema.
- Get an overview of activity log alerts, and learn how to receive alerts.
- Learn more about action groups.

Quickstart: Create a Recovery Services vault using an ARM template

Article • 03/10/2023

This quickstart describes how to set up a Recovery Services vault by using an Azure Resource Manager template (ARM template). The Azure Site Recovery service contributes to your business continuity and disaster recovery (BCDR) strategy so your business applications stay online during planned and unplanned outages. Site Recovery manages disaster recovery of on-premises machines and Azure virtual machines (VM), including replication, failover, and recovery.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

To protect VMware or physical server, see Modernized architecture.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an active Azure subscription, you can create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
```

```
      "name": "bicep",
      "version": "0.6.1.6515",
      "templateHash": "1347593202495112636"
    }
  },
  "parameters": {
    "vaultName": {
      "type": "string",
      "metadata": {
        "description": "Name of the Vault"
      }
    },
    "enableCRR": {
      "type": "bool",
      "defaultValue": true,
      "metadata": {
        "description": "Enable CRR (Works if vault has not registered any
backup instance)"
      }
    },
    "vaultStorageType": {
      "type": "string",
      "defaultValue": "GeoRedundant",
      "allowedValues": [
        "LocallyRedundant",
        "GeoRedundant"
      ],
      "metadata": {
        "description": "Change Vault Storage Type (Works if vault has not
registered any backup instance)"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    }
  },
  "variables": {
    "skuName": "RS0",
    "skuTier": "Standard"
  },
  "resources": [
    {
      "type": "Microsoft.RecoveryServices/vaults",
      "apiVersion": "2022-02-01",
      "name": "[parameters('vaultName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[variables('skuName')]",
        "tier": "[variables('skuTier')]"
      },
      "properties": {}
```

```
    },
    {
      "type": "Microsoft.RecoveryServices/vaults/backupstorageconfig",
      "apiVersion": "2022-02-01",
      "name": "[format('{0}/{1}', parameters('vaultName'),
'vaultstorageconfig')]",
      "properties": {
        "storageModelType": "[parameters('vaultStorageType')]",
        "crossRegionRestoreFlag": "[parameters('enableCRR')]"
      },
      "dependsOn": [
        "[resourceId('Microsoft.RecoveryServices/vaults',
parameters('vaultName'))]"
      ]
    }
  ]
}
```

Two Azure resources are defined in the template:

- Microsoft.RecoveryServices vaults: creates the vault.
- Microsoft.RecoveryServices/vaults/backupstorageconfig: configures the vault's backup redundancy settings.

The template includes optional parameters for the vault's backup configuration. The storage redundancy settings are locally-redundant storage (LRS) or geo-redundant storage (GRS). For more information, see Set storage redundancy.

For more Azure Recovery Services templates, see Azure Quickstart Templates.

Deploy the template

To deploy the template, the **Subscription**, **Resource group**, and **Vault name** are required.

1. To sign in to Azure and open the template, select the **Deploy to Azure** image.

   Deploy to Azure

2. Select or enter the following values:

- **Subscription**: select your Azure subscription.
- **Resource group**: select an existing group or select **Create new** to add a group.
- **Location**: defaults to the resource group's location and becomes unavailable after a resource group is selected.
- **Vault Name**: Provide a name for the vault.
- **Change Storage Type**: Default is **false**. Select **true** only if you need to change the vault's storage type.
- **Vault Storage Type**: Default is **GloballyRedundant**. If the storage type was set to **true**, select **LocallyRedundant**.
- **Location**: the function `[resourceGroup().location]` defaults to the resource group's location. To change the location, enter a value such as **westus**.
- Select the check box **I agree to the terms and conditions stated above**.

3. To begin the vault's deployment, select the **Purchase** button. After a successful deployment, a notification is displayed.

Validate the deployment

To confirm that the vault was created, use Azure CLI or Azure PowerShell.

CLI

Azure CLI

```
echo "Enter the resource group name:" &&
read resourceGroupName &&
echo "Enter the vault name:" &&
read vaultName &&
az backup vault show --name $vaultName --resource-group
$resourceGroupName &&
az backup vault backup-properties show --name $vaultName --resource-
group $resourceGroupName &&
echo "Press [ENTER] to continue ..."
```

The following output is an excerpt of the vault's information:

CLI

Output

```
"id": "/subscriptions/<Subscription Id>/resourceGroups/myResourceGroup
         /providers/Microsoft.RecoveryServices/vaults/myVault"
"location": "eastus"
"name": "myVault"
"resourceGroup": "myResourceGroup"
```

```
"storageModelType": "GeoRedundant"
"storageType": "GeoRedundant"
"type": "Microsoft.RecoveryServices/vaults/backupstorageconfig"
```

Clean up resources

If you plan to use the new resources, no action is needed. Otherwise, you can remove the resource group and vault that was created in this quickstart. To delete the resource group and its resources use Azure CLI or Azure PowerShell.

CLI

Azure CLI

```
echo "Enter the resource group name:" &&
read resourceGroupName &&
az group delete --name $resourceGroupName &&
echo "Press [ENTER] to continue ..."
```

Next steps

In this quickstart, you created a Recovery Services vault. To learn more about disaster recovery, continue to the next quickstart article.

Set up disaster recovery

Quickstart: Direct web traffic with Azure Application Gateway - ARM template

Article • 03/09/2023

In this quickstart, you use an Azure Resource Manager template (ARM template) to create an Azure Application Gateway. Then you test the application gateway to make sure it works correctly.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

You can also complete this quickstart using the Azure portal, Azure PowerShell, or Azure CLI.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

- An Azure account with an active subscription. Create an account for free.

Review the template

For the sake of simplicity, this template creates a simple setup with a public frontend IP, a basic listener to host a single site on the application gateway, a basic request routing rule, and two virtual machines in the backend pool.

The template used in this quickstart is from Azure Quickstart Templates

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "5093932345902341724"
    }
  },
  "parameters": {
    "adminUsername": {
      "type": "string",
      "metadata": {
        "description": "Admin username for the backend servers"
      }
    },
    "adminPassword": {
      "type": "secureString",
      "metadata": {
        "description": "Password for the admin account on the backend
servers"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "vmSize": {
      "type": "string",
      "defaultValue": "Standard_B2ms",
      "metadata": {
        "description": "Size of the virtual machine."
      }
    }
  },
  "variables": {
    "virtualMachineName": "myVM",
    "virtualNetworkName": "myVNet",
    "networkInterfaceName": "net-int",
    "ipconfigName": "ipconfig",
```

```
    "publicIPAddressName": "public_ip",
    "nsgName": "vm-nsg",
    "applicationGateWayName": "myAppGateway",
    "virtualNetworkPrefix": "10.0.0.0/16",
    "subnetPrefix": "10.0.0.0/24",
    "backendSubnetPrefix": "10.0.1.0/24"
  },
  "resources": [
    {
      "copy": {
        "name": "nsg",
        "count": "[length(range(0, 2))]"
      },
      "type": "Microsoft.Network/networkSecurityGroups",
      "apiVersion": "2021-05-01",
      "name": "[format('{0}{1}', variables('nsgName'), add(range(0, 2)
[copyIndex()], 1))]",
      "location": "[parameters('location')]",
      "properties": {
        "securityRules": [
          {
            "name": "RDP",
            "properties": {
              "protocol": "Tcp",
              "sourcePortRange": "*",
              "destinationPortRange": "3389",
              "sourceAddressPrefix": "*",
              "destinationAddressPrefix": "*",
              "access": "Allow",
              "priority": 300,
              "direction": "Inbound"
            }
          }
        ]
      }
    },
    {
      "copy": {
        "name": "publicIPAddress",
        "count": "[length(range(0, 3))]"
      },
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2021-05-01",
      "name": "[format('{0}{1}', variables('publicIPAddressName'), range(0,
3)[copyIndex()])]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard"
      },
      "properties": {
        "publicIPAddressVersion": "IPv4",
        "publicIPAllocationMethod": "Static",
        "idleTimeoutInMinutes": 4
      }
    },
```

```
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "2021-05-01",
      "name": "[variables('virtualNetworkName')]",
      "location": "[parameters('location')]",
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[variables('virtualNetworkPrefix')]"
          ]
        },
        "subnets": [
          {
            "name": "myAGSubnet",
            "properties": {
              "addressPrefix": "[variables('subnetPrefix')]",
              "privateEndpointNetworkPolicies": "Enabled",
              "privateLinkServiceNetworkPolicies": "Enabled"
            }
          },
          {
            "name": "myBackendSubnet",
            "properties": {
              "addressPrefix": "[variables('backendSubnetPrefix')]",
              "privateEndpointNetworkPolicies": "Enabled",
              "privateLinkServiceNetworkPolicies": "Enabled"
            }
          }
        ],
        "enableDdosProtection": false,
        "enableVmProtection": false
      }
    },
    {
      "copy": {
        "name": "virtualMachine",
        "count": "[length(range(0, 2))]"
      },
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2021-11-01",
      "name": "[format('{0}{1}', variables('virtualMachineName'),
add(range(0, 2)[copyIndex()], 1))]",
      "location": "[parameters('location')]",
      "properties": {
        "hardwareProfile": {
          "vmSize": "[parameters('vmSize')]"
        },
        "storageProfile": {
          "imageReference": {
            "publisher": "MicrosoftWindowsServer",
            "offer": "WindowsServer",
            "sku": "2016-Datacenter",
            "version": "latest"
          },
          "osDisk": {
```

```
            "osType": "Windows",
            "createOption": "FromImage",
            "caching": "ReadWrite",
            "managedDisk": {
              "storageAccountType": "StandardSSD_LRS"
            },
            "diskSizeGB": 127
          }
        },
        "osProfile": {
          "computerName": "[format('{0}{1}',
variables('virtualMachineName'), add(range(0, 2)[copyIndex()], 1))]",
          "adminUsername": "[parameters('adminUsername')]",
          "adminPassword": "[parameters('adminPassword')]",
          "windowsConfiguration": {
            "provisionVMAgent": true,
            "enableAutomaticUpdates": true
          },
          "allowExtensionOperations": true
        },
        "networkProfile": {
          "networkInterfaces": [
            {
              "id": "[resourceId('Microsoft.Network/networkInterfaces',
format('{0}{1}', variables('networkInterfaceName'), add(range(0, 2)
[copyIndex()], 1)))]"
            }
          ]
        }
      },
      "dependsOn": [
        "networkInterface"
      ]
    },
    {
      "copy": {
        "name": "virtualMachine_IIS",
        "count": "[length(range(0, 2))]"
      },
      "type": "Microsoft.Compute/virtualMachines/extensions",
      "apiVersion": "2021-11-01",
      "name": "[format('{0}{1}/IIS', variables('virtualMachineName'),
add(range(0, 2)[copyIndex()], 1))]",
      "location": "[parameters('location')]",
      "properties": {
        "autoUpgradeMinorVersion": true,
        "publisher": "Microsoft.Compute",
        "type": "CustomScriptExtension",
        "typeHandlerVersion": "1.4",
        "settings": {
          "commandToExecute": "powershell Add-WindowsFeature Web-Server;
powershell Add-Content -Path \"C:\\inetpub\\wwwroot\\Default.htm\" -Value
$($env:computername)"
        }
      },
```

```
      "dependsOn": [
        "virtualMachine"
      ]
    },
    {
      "type": "Microsoft.Network/applicationGateways",
      "apiVersion": "2021-05-01",
      "name": "[variables('applicationGateWayName')]",
      "location": "[parameters('location')]",
      "properties": {
        "sku": {
          "name": "Standard_v2",
          "tier": "Standard_v2"
        },
        "gatewayIPConfigurations": [
          {
            "name": "appGatewayIpConfig",
            "properties": {
              "subnet": {
                "id": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
variables('virtualNetworkName'), 'myAGSubnet')]"
              }
            }
          }
        ],
        "frontendIPConfigurations": [
          {
            "name": "appGwPublicFrontendIp",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses',
format('{0}0', variables('publicIPAddressName')))]"
              }
            }
          }
        ],
        "frontendPorts": [
          {
            "name": "port_80",
            "properties": {
              "port": 80
            }
          }
        ],
        "backendAddressPools": [
          {
            "name": "myBackendPool",
            "properties": {}
          }
        ],
        "backendHttpSettingsCollection": [
          {
            "name": "myHTTPSetting",
```

```
            "properties": {
              "port": 80,
              "protocol": "Http",
              "cookieBasedAffinity": "Disabled",
              "pickHostNameFromBackendAddress": false,
              "requestTimeout": 20
            }
          }
        ],
        "httpListeners": [
          {
            "name": "myListener",
            "properties": {
              "frontendIPConfiguration": {
                "id": "
[resourceId('Microsoft.Network/applicationGateways/frontendIPConfigurations'
, variables('applicationGateWayName'), 'appGwPublicFrontendIp')]"
              },
              "frontendPort": {
                "id": "
[resourceId('Microsoft.Network/applicationGateways/frontendPorts',
variables('applicationGateWayName'), 'port_80')]"
              },
              "protocol": "Http",
              "requireServerNameIndication": false
            }
          }
        ],
        "requestRoutingRules": [
          {
            "name": "myRoutingRule",
            "properties": {
              "ruleType": "Basic",
              "httpListener": {
                "id": "
[resourceId('Microsoft.Network/applicationGateways/httpListeners',
variables('applicationGateWayName'), 'myListener')]"
              },
              "backendAddressPool": {
                "id": "
[resourceId('Microsoft.Network/applicationGateways/backendAddressPools',
variables('applicationGateWayName'), 'myBackendPool')]"
              },
              "backendHttpSettings": {
                "id": "
[resourceId('Microsoft.Network/applicationGateways/backendHttpSettingsCollec
tion', variables('applicationGateWayName'), 'myHTTPSetting')]"
              }
            }
          }
        ],
        "enableHttp2": false,
        "autoscaleConfiguration": {
          "minCapacity": 0,
          "maxCapacity": 10
```

```
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/publicIPAddresses', format('{0}{1}',
variables('publicIPAddressName'), range(0, 3)[0]))]",
        "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]"
      ]
    },
    {
      "copy": {
        "name": "networkInterface",
        "count": "[length(range(0, 2))]"
      },
      "type": "Microsoft.Network/networkInterfaces",
      "apiVersion": "2021-05-01",
      "name": "[format('{0}{1}', variables('networkInterfaceName'),
add(range(0, 2)[copyIndex()], 1))]",
      "location": "[parameters('location')]",
      "properties": {
        "ipConfigurations": [
          {
            "name": "[format('{0}{1}', variables('ipconfigName'),
add(range(0, 2)[copyIndex()], 1))]",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses',
format('{0}{1}', variables('publicIPAddressName'), add(range(0, 2)
[copyIndex()], 1)))]"
              },
              "subnet": {
                "id": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
variables('virtualNetworkName'), 'myBackendSubnet')]"
              },
              "primary": true,
              "privateIPAddressVersion": "IPv4",
              "applicationGatewayBackendAddressPools": [
                {
                  "id": "
[resourceId('Microsoft.Network/applicationGateways/backendAddressPools',
variables('applicationGateWayName'), 'myBackendPool')]"
                }
              ]
            }
          }
        ],
        "enableAcceleratedNetworking": false,
        "enableIPForwarding": false,
        "networkSecurityGroup": {
          "id": "[resourceId('Microsoft.Network/networkSecurityGroups',
format('{0}{1}', variables('nsgName'), add(range(0, 2)[copyIndex()], 1)))]"
        }
      },
```

```
      "dependsOn": [
        "[resourceId('Microsoft.Network/applicationGateways',
variables('applicationGateWayName'))]",
        "nsg",
        "publicIPAddress"
      ]
    }
  ]
}
```

Multiple Azure resources are defined in the template:

- Microsoft.Network/applicationgateways
- Microsoft.Network/publicIPAddresses : one for the application gateway, and two for the virtual machines.
- Microsoft.Network/networkSecurityGroups
- Microsoft.Network/virtualNetworks
- Microsoft.Compute/virtualMachines : two virtual machines
- Microsoft.Network/networkInterfaces : two for the virtual machines
- Microsoft.Compute/virtualMachine/extensions : to configure IIS and the web pages

Deploy the template

Deploy the ARM template to Azure:

1. Select **Deploy to Azure** to sign in to Azure and open the template. The template creates an application gateway, the network infrastructure, and two virtual machines in the backend pool running IIS.

   Deploy to Azure

2. Select or create your resource group, type the virtual machine administrator user name and password.

3. Select **Review + Create** and then select **Create**.

   The deployment can take 20 minutes or longer to complete.

Validate the deployment

Although IIS isn't required to create the application gateway, it's installed to verify if Azure successfully created the application gateway. Use IIS to test the application

gateway:

1. Find the public IP address for the application gateway on its **Overview** page.

   Or, you can select **All resources**, enter *myAGPublicIPAddress* in the search box, and then select it in the search results. Azure displays the public IP address on the **Overview** page.

2. Copy the public IP address, and then paste it into the address bar of your browser to browse that IP address.

3. Check the response. A valid response verifies that the application gateway was successfully created and can successfully connect with the backend.

   Refresh the browser multiple times and you should see connections to both myVM1 and myVM2.

Clean up resources

When you no longer need the resources that you created with the application gateway, delete the resource group. This removes the application gateway and all the related resources.

To delete the resource group, call the `Remove-AzResourceGroup` cmdlet:

Azure PowerShell

```
Remove-AzResourceGroup -Name <your resource group name>
```

Next steps

Manage web traffic with an application gateway using the Azure CLI

Quickstart: Create an Azure CDN profile and endpoint - ARM template

Article • 03/08/2023

Get started with Azure Content Delivery Network (CDN) by using an Azure Resource Manager template (ARM template). The template deploys a profile and an endpoint.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template opens in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create an Azure free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

This template is configured to create a:

- Profile
- Endpoint

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "4893693976717968636"
```

```
      }
    },
    "parameters": {
      "profileName": {
        "type": "string",
        "metadata": {
          "description": "Name of the CDN Profile"
        }
      },
      "endpointName": {
        "type": "string",
        "metadata": {
          "description": "Name of the CDN Endpoint, must be unique"
        }
      },
      "originUrl": {
        "type": "string",
        "metadata": {
          "description": "Url of the origin"
        }
      },
      "CDNSku": {
        "type": "string",
        "defaultValue": "Standard_Microsoft",
        "allowedValues": [
          "Standard_Akamai",
          "Standard_Microsoft",
          "Standard_Verizon",
          "Premium_Verizon"
        ],
        "metadata": {
          "description": "CDN SKU names"
        }
      },
      "location": {
        "type": "string",
        "defaultValue": "[resourceGroup().location]",
        "metadata": {
          "description": "Location for all resources."
        }
      }
    },
    "resources": [
      {
        "type": "Microsoft.Cdn/profiles",
        "apiVersion": "2021-06-01",
        "name": "[parameters('profileName')]",
        "location": "[parameters('location')]",
        "sku": {
          "name": "[parameters('CDNSku')]"
        }
      },
      {
        "type": "Microsoft.Cdn/profiles/endpoints",
        "apiVersion": "2021-06-01",
```

```
      "name": "[format('{0}/{1}', parameters('profileName'),
parameters('endpointName'))]",
      "location": "[parameters('location')]",
      "properties": {
        "originHostHeader": "[parameters('originUrl')]",
        "isHttpAllowed": true,
        "isHttpsAllowed": true,
        "queryStringCachingBehavior": "IgnoreQueryString",
        "contentTypesToCompress": [
          "application/eot",
          "application/font",
          "application/font-sfnt",
          "application/javascript",
          "application/json",
          "application/opentype",
          "application/otf",
          "application/pkcs7-mime",
          "application/truetype",
          "application/ttf",
          "application/vnd.ms-fontobject",
          "application/xhtml+xml",
          "application/xml",
          "application/xml+rss",
          "application/x-font-opentype",
          "application/x-font-truetype",
          "application/x-font-ttf",
          "application/x-httpd-cgi",
          "application/x-javascript",
          "application/x-mpegurl",
          "application/x-opentype",
          "application/x-otf",
          "application/x-perl",
          "application/x-ttf",
          "font/eot",
          "font/ttf",
          "font/otf",
          "font/opentype",
          "image/svg+xml",
          "text/css",
          "text/csv",
          "text/html",
          "text/javascript",
          "text/js",
          "text/plain",
          "text/richtext",
          "text/tab-separated-values",
          "text/xml",
          "text/x-script",
          "text/x-component",
          "text/x-java-source"
        ],
        "isCompressionEnabled": true,
        "origins": [
          {
            "name": "origin1",
```

```
            "properties": {
              "hostName": "[parameters('originUrl')]"
            }
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Cdn/profiles', parameters('profileName'))]"
      ]
    }
  ]
}
```

One Azure resource is defined in the template:

- Microsoft.Cdn/profiles

Deploy the template

Azure CLI

Azure CLI

```
read -p "Enter the location (i.e. eastus): " location
resourceGroupName="myResourceGroupCDN"
templateUri="https://raw.githubusercontent.com/Azure/azure-quickstart-
templates/master/quickstarts/microsoft.cdn/cdn-with-custom-
origin/azuredeploy.json"

az group create \
--name $resourceGroupName \
--location $location

az deployment group create \
--resource-group $resourceGroupName \
--template-uri  $templateUri
```

PowerShell

Azure PowerShell

```
$location = Read-Host -Prompt "Enter the location (i.e. eastus)"
$templateUri = "https://raw.githubusercontent.com/Azure/azure-quickstart-
templates/master/quickstarts/microsoft.cdn/cdn-with-custom-
origin/azuredeploy.json"

$resourceGroupName = "myResourceGroupCDN"
```

```
New-AzResourceGroup -Name $resourceGroupName -Location $location
New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName -
TemplateUri $templateUri
```

Portal

Deploy to Azure

Review deployed resources

1. Sign in to the Azure portal.
2. Select **Resource groups** from the left pane.
3. Select the resource group that you created in the previous section. The default resource group name is **myResourceGroupCDN**
4. Verify the following resources were created in the resource group:

Clean up resources

Azure CLI

When no longer needed, you can use the az group delete command to remove the resource group and all resources contained within.

Azure CLI

```
az group delete \
  --name myResourceGroupCDN
```

PowerShell

When no longer needed, you can use the Remove-AzResourceGroup command to remove the resource group and all resources contained within.

Azure PowerShell

```
Remove-AzResourceGroup -Name myResourceGroupCDN
```

Portal

When no longer needed, delete the resource group, CDN profile, and all related resources. Select the resource group **myResourceGroupCDN** that contains the CDN profile and endpoint, and then select **Delete**.

Next steps

In this quickstart, you created a:

- CDN Profile
- Endpoint

To learn more about Azure CDN and Azure Resource Manager, continue to the next article:

Tutorial: Use CDN to serve static content from a web app

Quickstart: Create and configure Azure DDoS Network Protection using ARM template

Article • 03/08/2023

This quickstart describes how to use an Azure Resource Manager template (ARM template) to create a distributed denial of service (DDoS) protection plan and virtual network (VNet), then enables the protection plan for the VNet. An Azure DDoS Network Protection plan defines a set of virtual networks that have DDoS protection enabled across subscriptions. You can configure one DDoS protection plan for your organization and link virtual networks from multiple subscriptions to the same plan.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
```

```
      "version": "0.5.6.12127",
      "templateHash": "14909118711877377105"
    }
  },
  "parameters": {
    "ddosProtectionPlanName": {
      "type": "string",
      "metadata": {
        "description": "Specify a DDoS protection plan name."
      }
    },
    "virtualNetworkName": {
      "type": "string",
      "metadata": {
        "description": "Specify a DDoS virtual network name."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Specify a location for the resources."
      }
    },
    "vnetAddressPrefix": {
      "type": "string",
      "defaultValue": "172.17.0.0/16",
      "metadata": {
        "description": "Specify the virtual network address prefix"
      }
    },
    "subnetPrefix": {
      "type": "string",
      "defaultValue": "172.17.0.0/24",
      "metadata": {
        "description": "Specify the virtual network subnet prefix"
      }
    },
    "ddosProtectionPlanEnabled": {
      "type": "bool",
      "defaultValue": true,
      "metadata": {
        "description": "Enable DDoS protection plan."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Network/ddosProtectionPlans",
      "apiVersion": "2021-05-01",
      "name": "[parameters('ddosProtectionPlanName')]",
      "location": "[parameters('location')]"
    },
    {
      "type": "Microsoft.Network/virtualNetworks",
```

```
      "apiVersion": "2021-05-01",
      "name": "[parameters('virtualNetworkName')]",
      "location": "[parameters('location')]",
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[parameters('vnetAddressPrefix')]"
          ]
        },
        "subnets": [
          {
            "name": "default",
            "properties": {
              "addressPrefix": "[parameters('subnetPrefix')]"
            }
          }
        ],
        "enableDdosProtection": "[parameters('ddosProtectionPlanEnabled')]",
        "ddosProtectionPlan": {
          "id": "[resourceId('Microsoft.Network/ddosProtectionPlans',
parameters('ddosProtectionPlanName'))]"
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/ddosProtectionPlans',
parameters('ddosProtectionPlanName'))]"
      ]
    }
  ]
}
```

The template defines two resources:

- Microsoft.Network/ddosProtectionPlans
- Microsoft.Network/virtualNetworks

Deploy the template

In this example, the template creates a new resource group, a DDoS protection plan, and a VNet.

1. To sign in to Azure and open the template, select the **Deploy to Azure** button.

   Deploy to Azure

2. Enter the values to create a new resource group, DDoS protection plan, and VNet name.

- **Subscription**: Name of the Azure subscription where the resources will be deployed.
- **Resource group**: Select an existing resource group or create a new resource group.
- **Region**: The region where the resource group is deployed, such as East US.
- **Ddos Protection Plan Name**: The name of for the new DDoS protection plan.
- **Virtual Network Name**: Creates a name for the new VNet.
- **Location**: Function that uses the same region as the resource group for resource deployment.
- **Vnet Address Prefix**: Use the default value or enter your VNet address.
- **Subnet Prefix**: Use the default value or enter your VNet subnet.
- **Ddos Protection Plan Enabled**: Default is `true` to enable the DDoS protection plan.

3. Select **Review + create**.

4. Verify that template validation passed and select **Create** to begin the deployment.

Review deployed resources

To copy the Azure CLI or Azure PowerShell command, select the **Copy** button. The **Try it** button opens Azure Cloud Shell to run the command.

CLI

Azure CLI

```
az network ddos-protection show \
  --resource-group MyResourceGroup \
  --name MyDdosProtectionPlan
```

The output shows the new resources.

CLI

Output

```
{
  "etag": "W/\"abcdefgh-1111-2222-bbbb-987654321098\"",
  "id": "/subscriptions/b1111111-2222-3333-aaaa-
012345678912/resourceGroups/MyResourceGroup/providers/Microsoft.Network/
ddosProtectionPlans/MyDdosProtectionPlan",
  "location": "eastus",
  "name": "MyDdosProtectionPlan",
  "provisioningState": "Succeeded",
  "resourceGroup": "MyResourceGroup",
  "resourceGuid": null,
  "tags": null,
  "type": "Microsoft.Network/ddosProtectionPlans",
  "virtualNetworks": [
    {
      "id": "/subscriptions/b1111111-2222-3333-aaaa-
012345678912/resourceGroups/MyResourceGroup/providers/Microsoft.Network/
virtualNetworks/MyVNet",
      "resourceGroup": "MyResourceGroup"
    }
  ]
}
```

Clean up resources

When you're finished you can delete the resources. The command deletes the resource group and all the resources it contains.

CLI

Azure CLI

```
az group delete --name MyResourceGroup
```

Next steps

To learn how to view and configure telemetry for your DDoS protection plan, continue to the tutorials.

View and configure DDoS protection telemetry

Quickstart: Create an Azure DNS zone and record using an ARM template

Article • 03/08/2023

This quickstart describes how to use an Azure Resource Manager template (ARM Template) to create a DNS zone with an `A` record in it.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

In this quickstart, you'll create a unique DNS zone with a suffix of `azurequickstart.org`. An `A` record pointing to two IP addresses will also be placed in the zone.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.8.9.13224",
      "templateHash": "3207741835848035910"
    }
  },
```

```
  "parameters": {
    "zoneName": {
      "type": "string",
      "defaultValue": "[format('{0}.azurequickstart.org',
uniqueString(resourceGroup().id))]",
      "metadata": {
        "description": "The name of the DNS zone to be created.  Must have
at least 2 segments, e.g. hostname.org"
      }
    },
    "recordName": {
      "type": "string",
      "defaultValue": "www",
      "metadata": {
        "description": "The name of the DNS record to be created.  The name
is relative to the zone, not the FQDN."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Network/dnsZones",
      "apiVersion": "2018-05-01",
      "name": "[parameters('zoneName')]",
      "location": "global"
    },
    {
      "type": "Microsoft.Network/dnsZones/A",
      "apiVersion": "2018-05-01",
      "name": "[format('{0}/{1}', parameters('zoneName'),
parameters('recordName'))]",
      "properties": {
        "TTL": 3600,
        "ARecords": [
          {
            "ipv4Address": "1.2.3.4"
          },
          {
            "ipv4Address": "1.2.3.5"
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/dnsZones', parameters('zoneName'))]"
      ]
    }
  ],
  "outputs": {
    "nameServers": {
      "type": "array",
      "value": "[reference(resourceId('Microsoft.Network/dnsZones',
parameters('zoneName'))).nameServers]"
    }
  }
}
```

Two Azure resources have been defined in the template:

- **Microsoft.Network/dnsZones**
- **Microsoft.Network/dnsZones/A**: Used to create an `A` record in the zone.

To find more templates that are related to Azure Traffic Manager, see Azure Quickstart Templates.

Deploy the template

1. Select **Try it** from the following code block to open Azure Cloud Shell, and then follow the instructions to sign in to Azure.

   Azure PowerShell

   ```
   $projectName = Read-Host -Prompt "Enter a project name that is used for generating resource names"
   $location = Read-Host -Prompt "Enter the location (i.e. centralus)"
   $templateUri = "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.network/azure-dns-new-zone/azuredeploy.json"

   $resourceGroupName = "${projectName}rg"

   New-AzResourceGroup -Name $resourceGroupName -Location "$location"
   New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName -TemplateUri $templateUri

   Read-Host -Prompt "Press [ENTER] to continue ..."
   ```

   Wait until you see the prompt from the console.

2. Select **Copy** from the previous code block to copy the PowerShell script.

3. Right-click the shell console pane and then select **Paste**.

4. Enter the values.

   The template deployment creates a zone with one `A` record pointing to two IP addresses. The resource group name is the project name with **rg** appended.

   It takes a couple seconds to deploy the template. When completed, the output is similar to:

```
DeploymentName          : azuredeploy
ResourceGroupName       : devtestdnsrg
ProvisioningState       : Succeeded
Timestamp               : 9/8/2020 8:40:51 PM
Mode                    : Incremental
TemplateLink            :
                          Uri            :
                          https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101-azure-dns-new-zone/azuredeploy.json
                          ContentVersion : 1.0.0.0

Parameters              :
                          Name             Type                       Value
                          ===============  =========================  ==========
                          newZoneName      String                     havy3xctoclz2.azurequickstart.org
                          newRecordName    String                     www
```

Azure PowerShell is used to deploy the template. In addition to Azure PowerShell, you can also use the Azure portal, Azure CLI, and REST API. To learn other deployment methods, see Deploy templates.

Validate the deployment

1. Sign in to the Azure portal.

2. Select **Resource groups** from the left pane.

3. Select the resource group that you created in the previous section. The default resource group name is the project name with **rg** appended.

4. The resource group should contain the following resources seen here:

5. Select the DNS zone with the suffix of `azurequickstart.org` to verify that the zone is created properly with an `A` record referencing the value of `1.2.3.4` and `1.2.3.5`.

6. Copy one of the name server names from the previous step.

7. Open a command prompt, and run the following command:

Windows Command Prompt

```
nslookup www.<dns zone name> <name server name>
```

For example:

Windows Command Prompt

```
nslookup www.2lwynbseszpam.azurequickstart.org ns1-09.azure-dns.com.
```

You should see something like the following screenshot:

The host name `www.2lwynbseszpam.azurequickstart.org` resolves to `1.2.3.4` and `1.2.3.5`, just as you configured it. This result verifies that name resolution is working

correctly.

Clean up resources

When you no longer need the resources that you created with the DNS zone, delete the resource group. This action removes the DNS zone and all the related resources.

To delete the resource group, call the `Remove-AzResourceGroup` cmdlet:

Azure PowerShell

```
Remove-AzResourceGroup -Name <your resource group name>
```

Next steps

In this quickstart, you created a:

- DNS zone
- `A` record

Now that you've created your first DNS zone and record using an ARM template, you can create records for a web app in a custom domain.

Create DNS records for a web app in a custom domain

Quickstart: Create an ExpressRoute circuit with private peering using an ARM template

Article • 03/08/2023

This quickstart describes how to use an Azure Resource Manager template (ARM Template) to create an ExpressRoute circuit with private peering.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

In this quickstart, you'll create an ExpressRoute circuit with *Equinix* as the service provider. The circuit will be using a *Premium SKU*, with a bandwidth of *50 Mbps*, and the peering location of *Washington DC*. Private peering will be enabled with a primary and secondary subnet of *192.168.10.16/30* and *192.168.10.20/30* respectively. A virtual network will also be created along with a *HighPerformance ExpressRoute gateway*.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
```

```
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "7902308351795495256"
    }
  },
  "parameters": {
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources deployed in the Bicep
file"
      }
    },
    "erpeeringLocation": {
      "type": "string",
      "defaultValue": "Washington DC",
      "metadata": {
        "description": "ExpressRoute peering location"
      }
    },
    "erCircuitName": {
      "type": "string",
      "defaultValue": "er-ckt01",
      "metadata": {
        "description": "Name of the ExpressRoute circuit"
      }
    },
    "serviceProviderName": {
      "type": "string",
      "defaultValue": "Equinix",
      "metadata": {
        "description": "Name of the ExpressRoute provider"
      }
    },
    "erSKU_Tier": {
      "type": "string",
      "defaultValue": "Premium",
      "allowedValues": [
        "Premium",
        "Standard"
      ],
      "metadata": {
        "description": "Tier ExpressRoute circuit"
      }
    },
    "erSKU_Family": {
      "type": "string",
      "defaultValue": "MeteredData",
      "allowedValues": [
        "MeteredData",
        "UnlimitedData"
      ],
      "metadata": {
```

```
        "description": "Billing model ExpressRoute circuit"
      }
    },
    "bandwidthInMbps": {
      "type": "int",
      "defaultValue": 50,
      "allowedValues": [
        50,
        100,
        200,
        500,
        1000,
        2000,
        5000,
        10000
      ],
      "metadata": {
        "description": "Bandwidth ExpressRoute circuit"
      }
    },
    "peerASN": {
      "type": "int",
      "defaultValue": 65001,
      "metadata": {
        "description": "autonomous system number used to create private
peering between the customer edge router and MSEE routers"
      }
    },
    "primaryPeerAddressPrefix": {
      "type": "string",
      "defaultValue": "192.168.10.16/30",
      "metadata": {
        "description": "point-to-point network prefix of primary link
between the customer edge router and MSEE router"
      }
    },
    "secondaryPeerAddressPrefix": {
      "type": "string",
      "defaultValue": "192.168.10.20/30",
      "metadata": {
        "description": "point-to-point network prefix of secondary link
between the customer edge router and MSEE router"
      }
    },
    "vlanId": {
      "type": "int",
      "defaultValue": 100,
      "metadata": {
        "description": "VLAN Id used between the customer edge routers and
MSEE routers. primary and secondary link have the same VLAN Id"
      }
    },
    "vnetName": {
      "type": "string",
      "defaultValue": "vnet1",
```

```
      "metadata": {
        "description": "name of the Virtual Network"
      }
    },
    "subnet1Name": {
      "type": "string",
      "defaultValue": "subnet1",
      "metadata": {
        "description": "name of the subnet"
      }
    },
    "vnetAddressSpace": {
      "type": "string",
      "defaultValue": "10.10.10.0/24",
      "metadata": {
        "description": "address space assigned to the Virtual Network"
      }
    },
    "subnet1Prefix": {
      "type": "string",
      "defaultValue": "10.10.10.0/25",
      "metadata": {
        "description": "network prefix assigned to the subnet"
      }
    },
    "gatewaySubnetPrefix": {
      "type": "string",
      "defaultValue": "10.10.10.224/27",
      "metadata": {
        "description": "network prefixes assigned to the gateway subnet. It
has to be a network prefix with mask /27 or larger"
      }
    },
    "gatewayName": {
      "type": "string",
      "defaultValue": "er-gw",
      "metadata": {
        "description": "name of the ExpressRoute Gateway"
      }
    },
    "gatewaySku": {
      "type": "string",
      "defaultValue": "HighPerformance",
      "allowedValues": [
        "Standard",
        "HighPerformance",
        "UltraPerformance",
        "ErGw1AZ",
        "ErGw2AZ",
        "ErGw3AZ"
      ],
      "metadata": {
        "description": "ExpressRoute Gateway SKU"
      }
    }
```

```
  },
  "variables": {
    "erSKU_Name": "[format('{0}_{1}', parameters('erSKU_Tier'),
parameters('erSKU_Family'))]",
    "gatewayPublicIPName": "[format('{0}-pubIP',
parameters('gatewayName'))]",
    "nsgName": "nsg"
  },
  "resources": [
    {
      "type": "Microsoft.Network/expressRouteCircuits",
      "apiVersion": "2021-05-01",
      "name": "[parameters('erCircuitName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[variables('erSKU_Name')]",
        "tier": "[parameters('erSKU_Tier')]",
        "family": "[parameters('erSKU_Family')]"
      },
      "properties": {
        "serviceProviderProperties": {
          "serviceProviderName": "[parameters('serviceProviderName')]",
          "peeringLocation": "[parameters('erpeeringLocation')]",
          "bandwidthInMbps": "[parameters('bandwidthInMbps')]"
        },
        "allowClassicOperations": false
      }
    },
    {
      "type": "Microsoft.Network/expressRouteCircuits/peerings",
      "apiVersion": "2021-05-01",
      "name": "[format('{0}/{1}', parameters('erCircuitName'),
'AzurePrivatePeering')]",
      "properties": {
        "peeringType": "AzurePrivatePeering",
        "peerASN": "[parameters('peerASN')]",
        "primaryPeerAddressPrefix": "
[parameters('primaryPeerAddressPrefix')]",
        "secondaryPeerAddressPrefix": "
[parameters('secondaryPeerAddressPrefix')]",
        "vlanId": "[parameters('vlanId')]"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/expressRouteCircuits',
parameters('erCircuitName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/networkSecurityGroups",
      "apiVersion": "2021-05-01",
      "name": "[variables('nsgName')]",
      "location": "[parameters('location')]",
      "properties": {
        "securityRules": [
          {
```

```
            "name": "SSH-rule",
            "properties": {
              "description": "allow SSH",
              "protocol": "Tcp",
              "sourcePortRange": "*",
              "destinationPortRange": "22",
              "sourceAddressPrefix": "*",
              "destinationAddressPrefix": "VirtualNetwork",
              "access": "Allow",
              "priority": 500,
              "direction": "Inbound"
            }
          },
          {
            "name": "RDP-rule",
            "properties": {
              "description": "allow RDP",
              "protocol": "Tcp",
              "sourcePortRange": "*",
              "destinationPortRange": "3389",
              "sourceAddressPrefix": "*",
              "destinationAddressPrefix": "VirtualNetwork",
              "access": "Allow",
              "priority": 600,
              "direction": "Inbound"
            }
          }
        ]
      }
    },
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "2021-05-01",
      "name": "[parameters('vnetName')]",
      "location": "[parameters('location')]",
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[parameters('vnetAddressSpace')]"
          ]
        },
        "subnets": [
          {
            "name": "[parameters('subnet1Name')]",
            "properties": {
              "addressPrefix": "[parameters('subnet1Prefix')]",
              "networkSecurityGroup": {
                "id": "
[resourceId('Microsoft.Network/networkSecurityGroups',
variables('nsgName'))]"
              }
            }
          },
          {
            "name": "GatewaySubnet",
```

```
            "properties": {
              "addressPrefix": "[parameters('gatewaySubnetPrefix')]"
            }
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('nsgName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2021-05-01",
      "name": "[variables('gatewayPublicIPName')]",
      "location": "[parameters('location')]",
      "properties": {
        "publicIPAllocationMethod": "Dynamic"
      }
    },
    {
      "type": "Microsoft.Network/virtualNetworkGateways",
      "apiVersion": "2021-05-01",
      "name": "[parameters('gatewayName')]",
      "location": "[parameters('location')]",
      "properties": {
        "ipConfigurations": [
          {
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "subnet": {
                "id": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('vnetName'), 'GatewaySubnet')]"
              },
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses',
variables('gatewayPublicIPName'))]"
              }
            },
            "name": "gwIPconf"
          }
        ],
        "gatewayType": "ExpressRoute",
        "sku": {
          "name": "[parameters('gatewaySku')]",
          "tier": "[parameters('gatewaySku')]"
        },
        "vpnType": "RouteBased"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/publicIPAddresses',
variables('gatewayPublicIPName'))]",
        "[resourceId('Microsoft.Network/virtualNetworks',
parameters('vnetName'))]"
```

```
        ]
      }
    ],
    "outputs": {
      "erCircuitName": {
        "type": "string",
        "value": "[parameters('erCircuitName')]"
      },
      "gatewayName": {
        "type": "string",
        "value": "[parameters('gatewayName')]"
      },
      "gatewaySku": {
        "type": "string",
        "value": "[parameters('gatewaySku')]"
      }
    }
  }
```

Multiple Azure resources have been defined in the template:

- **Microsoft.Network/expressRouteCircuits**
- **Microsoft.Network/expressRouteCircuits/peerings** (Used to enabled private peering on the circuit)
- **Microsoft.Network/networkSecurityGroups** (network security group is applied to the subnets in the virtual network)
- **Microsoft.Network/virtualNetworks**
- **Microsoft.Network/publicIPAddresses** (Public IP is used by the ExpressRoute gateway)
- **Microsoft.Network/virtualNetworkGateways** (ExpressRoute gateway is used to link VNet to the circuit)

To find more templates that are related to ExpressRoute, see Azure Quickstart Templates.

Deploy the template

1. Select **Try it** from the following code block to open Azure Cloud Shell, and then follow the instructions to sign in to Azure.

   Azure PowerShell

   ```
   $projectName = Read-Host -Prompt "Enter a project name that is used for
   generating resource names"
   $location = Read-Host -Prompt "Enter the location (i.e. centralus)"
   $templateUri = "https://raw.githubusercontent.com/Azure/azure-
   quickstart-templates/master/quickstarts/microsoft.network/expressroute-
   ```

```
private-peering-vnet/azuredeploy.json"

$resourceGroupName = "${projectName}rg"

New-AzResourceGroup -Name $resourceGroupName -Location "$location"
New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName -
TemplateUri $templateUri

Read-Host -Prompt "Press [ENTER] to continue ..."
```

Wait until you see the prompt from the console.

2. Select **Copy** from the previous code block to copy the PowerShell script.

3. Right-click the shell console pane and then select **Paste**.

4. Enter the values.

   The resource group name is the project name with **rg** appended.

   It takes about 20 minutes to deploy the template. When completed, the output is similar to:

```
DeploymentName          : azuredeploy
ResourceGroupName       : erdevtestrg
ProvisioningState       : Succeeded
Timestamp               : 8/28/2020 7:18:02 PM
Mode                    : Incremental
TemplateLink            :
                          Uri            : https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101
                          -expressroute-private-peering-vnet/azuredeploy.json
                          ContentVersion : 1.0.0.0

Parameters              :
                          Name                        Type                       Value
                          ==========================  ========================   ==========
                          location                    String                     eastus
                          erpeeringLocation           String                     Washington DC
                          erCircuitName               String                     er-ckt01
                          serviceProviderName         String                     Equinix
                          erSKU_Tier                  String                     Premium
                          erSKU_Family                String                     MeteredData
                          bandwidthInMbps             Int                        50
                          peerASN                     Int                        65001
                          primaryPeerAddressPrefix    String                     192.168.10.16/30
                          secondaryPeerAddressPrefix  String                     192.168.10.20/30
                          vlanId                      Int                        100
                          vnetName                    String                     vnet1
                          subnet1Name                 String                     subnet1
                          vnetAddressSpace            String                     10.10.10.0/24
                          subnet1Prefix               String                     10.10.10.0/25
                          gatewaySubnetPrefix         String                     10.10.10.224/27
                          gatewayName                 String                     er-gw
                          gatewaySku                  String                     HighPerformance
```

Azure PowerShell is used to deploy the template. In addition to Azure PowerShell, you can also use the Azure portal, Azure CLI, and REST API. To learn other deployment methods, see Deploy templates.

Validate the deployment

1. Sign in to the Azure portal.

2. Select **Resource groups** from the left pane.

3. Select the resource group that you created in the previous section. The default resource group name is the project name with **rg** appended.

4. The resource group should contain the following resources seen here:

5. Select the ExpressRoute circuit **er-ck01** to verify that the circuit status is **Enabled**, provider status is **Not provisioned** and private peering has the status of **Provisioned**.

> ⓘ **Note**
>
> You will need to call the provider to complete the provisioning process before you can link the virtual network to the circuit.

Clean up resources

When you no longer need the resources that you created with the ExpressRoute circuit, delete the resource group. This removes the ExpressRoute circuit and all the related

resources.

To delete the resource group, call the `Remove-AzResourceGroup` cmdlet:

Azure PowerShell

```
Remove-AzResourceGroup -Name <your resource group name>
```

Next steps

In this quickstart, you created a:

- ExpressRoute circuit
- Virtual Network
- VPN Gateway
- Public IP
- network security groups

To learn how to link a virtual network to a circuit, continue to the ExpressRoute tutorials.

ExpressRoute tutorials

Quickstart: Deploy Azure Firewall with Availability Zones - ARM template

Article • 03/08/2023

In this quickstart, you use an Azure Resource Manager template (ARM template) to deploy an Azure Firewall in three Availability Zones.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

The template creates a test network environment with a firewall. The network has one virtual network (VNet) with three subnets: *AzureFirewallSubnet*, *ServersSubnet*, and *JumpboxSubnet*. The *ServersSubnet* and *JumpboxSubnet* subnet each have a single, two-core Windows Server virtual machine.

The firewall is in the *AzureFirewallSubnet* subnet, and has an application rule collection with a single rule that allows access to `www.microsoft.com`.

A user-defined route points network traffic from the *ServersSubnet* subnet through the firewall, where the firewall rules are applied.

For more information about Azure Firewall, see Deploy and configure Azure Firewall using the Azure portal.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

- An Azure account with an active subscription. Create an account for free.

Review the template

This template creates an Azure Firewall with Availability Zones, along with the necessary resources to support the Azure Firewall.

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.7.4.23292",
      "templateHash": "11311417953238012 57"
    }
  },
  "parameters": {
    "virtualNetworkName": {
      "type": "string",
      "defaultValue": "test-vnet",
      "metadata": {
        "description": "virtual network name"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "adminUsername": {
      "type": "string",
      "metadata": {
        "description": "Username for the Virtual Machine."
      }
    },
    "adminPassword": {
      "type": "secureString",
      "metadata": {
        "description": "Password for the Virtual Machine."
      }
    },
    "availabilityZones": {
      "type": "array",
      "defaultValue": [
        "1",
        "2",
        "3"
      ],
      "metadata": {
        "description": "Availability zone numbers e.g. 1,2,3."
      }
    },
    "numberOfFirewallPublicIPAddresses": {
```

```
      "type": "int",
      "defaultValue": 1,
      "maxValue": 100,
      "minValue": 1,
      "metadata": {
        "description": "Number of public IP addresses for the Azure
Firewall"
      }
    },
    "jumpBoxSize": {
      "type": "string",
      "defaultValue": "Standard_D2s_v3",
      "metadata": {
        "description": "Size of the virtual machine."
      }
    },
    "serverSize": {
      "type": "string",
      "defaultValue": "Standard_D2s_v3",
      "metadata": {
        "description": "Size of the virtual machine."
      }
    }
  },
  "variables": {
    "copy": [
      {
        "name": "azureFirewallIpConfigurations",
        "count": "[length(range(0,
parameters('numberOfFirewallPublicIPAddresses')))]",
        "input": {
          "name": "[format('IpConf{0}', range(0,
parameters('numberOfFirewallPublicIPAddresses'))
[copyIndex('azureFirewallIpConfigurations')])]",
          "properties": {
            "subnet": "[if(equals(range(0,
parameters('numberOfFirewallPublicIPAddresses'))
[copyIndex('azureFirewallIpConfigurations')], 0),
variables('azureFirewallSubnetJSON'), json('null'))]",
            "publicIPAddress": {
              "id": "[resourceId('Microsoft.Network/publicIPAddresses',
format('{0}{1}', variables('publicIPNamePrefix'), add(range(0,
parameters('numberOfFirewallPublicIPAddresses'))[range(0,
parameters('numberOfFirewallPublicIPAddresses'))
[copyIndex('azureFirewallIpConfigurations')]], 1)))]"
            }
          }
        }
      }
    ],
    "vnetAddressPrefix": "10.0.0.0/16",
    "serversSubnetPrefix": "10.0.2.0/24",
    "azureFirewallSubnetPrefix": "10.0.1.0/24",
    "jumpboxSubnetPrefix": "10.0.0.0/24",
    "nextHopIP": "10.0.1.4",
```

```
    "azureFirewallSubnetName": "AzureFirewallSubnet",
    "jumpBoxSubnetName": "JumpboxSubnet",
    "serversSubnetName": "ServersSubnet",
    "jumpBoxPublicIPAddressName": "JumpHostPublicIP",
    "jumpBoxNsgName": "JumpHostNSG",
    "jumpBoxNicName": "JumpHostNic",
    "jumpBoxSubnetId": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('virtualNetworkName'), variables('jumpBoxSubnetName'))]",
    "serverNicName": "ServerNic",
    "serverSubnetId": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('virtualNetworkName'), variables('serversSubnetName'))]",
    "storageAccountName": "[format('{0}sajumpbox',
uniqueString(resourceGroup().id))]",
    "azfwRouteTableName": "AzfwRouteTable",
    "firewallName": "firewall1",
    "publicIPNamePrefix": "publicIP",
    "azureFirewallSubnetId": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('virtualNetworkName'), variables('azureFirewallSubnetName'))]",
    "azureFirewallSubnetJSON": "[json(format('{{\"id\": \"{0}\"}}',
variables('azureFirewallSubnetId')))]",
    "networkSecurityGroupName": "[format('{0}-nsg',
variables('serversSubnetName'))]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-08-01",
      "name": "[variables('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "Storage",
      "properties": {}
    },
    {
      "type": "Microsoft.Network/routeTables",
      "apiVersion": "2021-03-01",
      "name": "[variables('azfwRouteTableName')]",
      "location": "[parameters('location')]",
      "properties": {
        "disableBgpRoutePropagation": false,
        "routes": [
          {
            "name": "AzfwDefaultRoute",
            "properties": {
              "addressPrefix": "0.0.0.0/0",
              "nextHopType": "VirtualAppliance",
              "nextHopIpAddress": "[variables('nextHopIP')]"
            }
          }
        ]
```

```
      }
    },
    {
      "type": "Microsoft.Network/networkSecurityGroups",
      "apiVersion": "2021-03-01",
      "name": "[variables('networkSecurityGroupName')]",
      "location": "[parameters('location')]",
      "properties": {}
    },
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "2021-05-01",
      "name": "[parameters('virtualNetworkName')]",
      "location": "[parameters('location')]",
      "tags": {
        "displayName": "[parameters('virtualNetworkName')]"
      },
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[variables('vnetAddressPrefix')]"
          ]
        },
        "subnets": [
          {
            "name": "[variables('jumpBoxSubnetName')]",
            "properties": {
              "addressPrefix": "[variables('jumpboxSubnetPrefix')]"
            }
          },
          {
            "name": "[variables('azureFirewallSubnetName')]",
            "properties": {
              "addressPrefix": "[variables('azureFirewallSubnetPrefix')]"
            }
          },
          {
            "name": "[variables('serversSubnetName')]",
            "properties": {
              "addressPrefix": "[variables('serversSubnetPrefix')]",
              "routeTable": {
                "id": "[resourceId('Microsoft.Network/routeTables',
variables('azfwRouteTableName'))]"
              },
              "networkSecurityGroup": {
                "id": "
[resourceId('Microsoft.Network/networkSecurityGroups',
variables('networkSecurityGroupName'))]"
              }
            }
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/routeTables',
```

```
variables('azfwRouteTableName'))]",
        "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('networkSecurityGroupName'))]"
      ]
    },
    {
      "copy": {
        "name": "publicIPAddress",
        "count": "[length(range(0,
parameters('numberOfFirewallPublicIPAddresses')))]"
      },
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2021-03-01",
      "name": "[format('{0}{1}', variables('publicIPNamePrefix'),
add(range(0, parameters('numberOfFirewallPublicIPAddresses'))[copyIndex()],
1))]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard"
      },
      "properties": {
        "publicIPAllocationMethod": "Static",
        "publicIPAddressVersion": "IPv4"
      },
      "zones": "[parameters('availabilityZones')]"
    },
    {
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2021-03-01",
      "name": "[variables('jumpBoxPublicIPAddressName')]",
      "location": "[parameters('location')]",
      "properties": {
        "publicIPAllocationMethod": "Dynamic"
      }
    },
    {
      "type": "Microsoft.Network/networkSecurityGroups",
      "apiVersion": "2021-05-01",
      "name": "[variables('jumpBoxNsgName')]",
      "location": "[parameters('location')]",
      "properties": {
        "securityRules": [
          {
            "name": "myNetworkSecurityGroupRuleRDP",
            "properties": {
              "protocol": "Tcp",
              "sourcePortRange": "*",
              "destinationPortRange": "3389",
              "sourceAddressPrefix": "*",
              "destinationAddressPrefix": "*",
              "access": "Allow",
              "priority": 1000,
              "direction": "Inbound"
            }
          }
```

```
        ]
      }
    },
    {
      "type": "Microsoft.Network/networkInterfaces",
      "apiVersion": "2021-05-01",
      "name": "[variables('jumpBoxNicName')]",
      "location": "[parameters('location')]",
      "properties": {
        "ipConfigurations": [
          {
            "name": "ipconfig1",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses',
variables('jumpBoxPublicIPAddressName'))]"
              },
              "subnet": {
                "id": "[variables('jumpBoxSubnetId')]"
              }
            }
          }
        ],
        "networkSecurityGroup": {
          "id": "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('jumpBoxNsgName'))]"
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('jumpBoxNsgName'))]",
        "[resourceId('Microsoft.Network/publicIPAddresses',
variables('jumpBoxPublicIPAddressName'))]",
        "[resourceId('Microsoft.Network/virtualNetworks',
parameters('virtualNetworkName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/networkInterfaces",
      "apiVersion": "2021-05-01",
      "name": "[variables('serverNicName')]",
      "location": "[parameters('location')]",
      "properties": {
        "ipConfigurations": [
          {
            "name": "ipconfig1",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "subnet": {
                "id": "[variables('serverSubnetId')]"
              }
            }
          }
        ]
```

```
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/virtualNetworks',
parameters('virtualNetworkName'))]"
      ]
    },
    {
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2021-11-01",
      "name": "JumpBox",
      "location": "[parameters('location')]",
      "properties": {
        "hardwareProfile": {
          "vmSize": "[parameters('jumpBoxSize')]"
        },
        "storageProfile": {
          "imageReference": {
            "publisher": "MicrosoftWindowsServer",
            "offer": "WindowsServer",
            "sku": "2019-Datacenter",
            "version": "latest"
          },
          "osDisk": {
            "osType": "Windows",
            "createOption": "FromImage",
            "diskSizeGB": 127
          }
        },
        "osProfile": {
          "computerName": "JumpBox",
          "adminUsername": "[parameters('adminUsername')]",
          "adminPassword": "[parameters('adminPassword')]"
        },
        "networkProfile": {
          "networkInterfaces": [
            {
              "id": "[resourceId('Microsoft.Network/networkInterfaces',
variables('jumpBoxNicName'))]"
            }
          ]
        },
        "diagnosticsProfile": {
          "bootDiagnostics": {
            "enabled": true,
            "storageUri": "
[reference(resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))).primaryEndpoints.blob]"
          }
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkInterfaces',
variables('jumpBoxNicName'))]",
        "[resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))]"
```

```
      ]
    },
    {
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2021-11-01",
      "name": "Server",
      "location": "[parameters('location')]",
      "properties": {
        "hardwareProfile": {
          "vmSize": "[parameters('serverSize')]"
        },
        "storageProfile": {
          "imageReference": {
            "publisher": "MicrosoftWindowsServer",
            "offer": "WindowsServer",
            "sku": "2019-Datacenter",
            "version": "latest"
          },
          "osDisk": {
            "osType": "Windows",
            "createOption": "FromImage",
            "diskSizeGB": 127
          }
        },
        "osProfile": {
          "computerName": "Server",
          "adminUsername": "[parameters('adminUsername')]",
          "adminPassword": "[parameters('adminPassword')]"
        },
        "networkProfile": {
          "networkInterfaces": [
            {
              "id": "[resourceId('Microsoft.Network/networkInterfaces',
variables('serverNicName'))]"
            }
          ]
        },
        "diagnosticsProfile": {
          "bootDiagnostics": {
            "enabled": true,
            "storageUri": "
[reference(resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))).primaryEndpoints.blob]"
          }
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkInterfaces',
variables('serverNicName'))]",
        "[resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/azureFirewalls",
```

```
      "apiVersion": "2021-05-01",
      "name": "[variables('firewallName')]",
      "location": "[parameters('location')]",
      "zones": "[if(equals(length(parameters('availabilityZones')), 0),
json('null'), parameters('availabilityZones'))]",
      "properties": {
        "ipConfigurations": "[variables('azureFirewallIpConfigurations')]",
        "applicationRuleCollections": [
          {
            "name": "appRc1",
            "properties": {
              "priority": 101,
              "action": {
                "type": "Allow"
              },
              "rules": [
                {
                  "name": "appRule1",
                  "protocols": [
                    {
                      "port": 80,
                      "protocolType": "Http"
                    },
                    {
                      "port": 443,
                      "protocolType": "Https"
                    }
                  ],
                  "targetFqdns": [
                    "www.microsoft.com"
                  ],
                  "sourceAddresses": [
                    "10.0.2.0/24"
                  ]
                }
              ]
            }
          }
        ],
        "networkRuleCollections": [
          {
            "name": "netRc1",
            "properties": {
              "priority": 200,
              "action": {
                "type": "Allow"
              },
              "rules": [
                {
                  "name": "netRule1",
                  "protocols": [
                    "TCP"
                  ],
                  "sourceAddresses": [
                    "10.0.2.0/24"
```

```
                    ],
                    "destinationAddresses": [
                      "*"
                    ],
                    "destinationPorts": [
                      "8000-8999"
                    ]
                  }
                ]
              }
            }
          ]
        },
        "dependsOn": [
          "publicIPAddress",
          "[resourceId('Microsoft.Network/virtualNetworks',
parameters('virtualNetworkName'))]"
        ]
      }
    ]
}
```

Multiple Azure resources are defined in the template:

- Microsoft.Storage/storageAccounts
- Microsoft.Network/routeTables
- Microsoft.Network/networkSecurityGroups
- Microsoft.Network/virtualNetworks
- Microsoft.Network/publicIPAddresses
- Microsoft.Network/networkInterfaces
- Microsoft.Compute/virtualMachines
- Microsoft.Network/azureFirewalls

Deploy the template

Deploy the ARM template to Azure:

1. Select **Deploy to Azure** to sign in to Azure and open the template. The template creates an Azure Firewall, the network infrastructure, and two virtual machines.

   Deploy to Azure

2. In the portal, on the **Create a sandbox setup of Azure Firewall with Zones** page, type or select the following values:

   - **Resource group**: Select **Create new**, type a name for the resource group, and select **OK**.

- **Virtual Network Name**: Type a name for the new VNet.
- **Admin Username**: Type a username for the administrator user account.
- **Admin Password**: Type an administrator password.

3. Read the terms and conditions, and then select **I agree to the terms and conditions stated above** and then select **Purchase**. The deployment can take 10 minutes or longer to complete.

Review deployed resources

Explore the resources that were created with the firewall.

To learn about the JSON syntax and properties for a firewall in a template, see Microsoft.Network/azureFirewalls.

Clean up resources

When you no longer need them, you can remove the resource group, firewall, and all related resources by running the `Remove-AzResourceGroup` PowerShell command. To remove a resource group named *MyResourceGroup*, run:

Azure PowerShell

```
Remove-AzResourceGroup -Name MyResourceGroup
```

Don't remove the resource group and firewall if you plan to continue on to the firewall monitoring tutorial.

Next steps

Next, you can monitor the Azure Firewall logs.

Tutorial: Monitor Azure Firewall logs

Quickstart: Create an Azure Firewall and IP Groups - ARM template

Article • 03/08/2023

In this quickstart, you use an Azure Resource Manager template (ARM template) to deploy an Azure Firewall with sample IP Groups used in a network rule and application rule. An IP Group is a top-level resource that allows you to define and group IP addresses, ranges, and subnets into a single object. This is useful for managing IP addresses in Azure Firewall rules. You can either manually enter IP addresses or import them from a file.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

- An Azure account with an active subscription. Create an account for free.

Review the template

This template creates an Azure Firewall and IP Groups, along with the necessary resources to support the Azure Firewall.

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
```

```
      "name": "bicep",
      "version": "0.9.1.41621",
      "templateHash": "7270757728234245791"
    }
  },
  "parameters": {
    "virtualNetworkName": {
      "type": "string",
      "defaultValue": "[format('vnet{0}',
uniqueString(resourceGroup().id))]",
      "metadata": {
        "description": "virtual network name"
      }
    },
    "ipgroups_name1": {
      "type": "string",
      "defaultValue": "[format('ipgroup1{0}',
uniqueString(resourceGroup().id))]"
    },
    "ipgroups_name2": {
      "type": "string",
      "defaultValue": "[format('ipgroup2{0}',
uniqueString(resourceGroup().id))]"
    },
    "adminUsername": {
      "type": "string",
      "metadata": {
        "description": "Username for the Virtual Machine."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "vmSize": {
      "type": "string",
      "defaultValue": "Standard_D2s_v3",
      "metadata": {
        "description": "Zone numbers e.g. 1,2,3."
      }
    },
    "numberOfFirewallPublicIPAddresses": {
      "type": "int",
      "defaultValue": 1,
      "maxValue": 100,
      "minValue": 1,
      "metadata": {
        "description": "Number of public IP addresses for the Azure
Firewall"
      }
    },
    "authenticationType": {
```

```
      "type": "string",
      "defaultValue": "sshPublicKey",
      "allowedValues": [
        "sshPublicKey",
        "password"
      ],
      "metadata": {
        "description": "Type of authentication to use on the Virtual
Machine. SSH key is recommended."
      }
    },
    "adminPasswordOrKey": {
      "type": "secureString",
      "metadata": {
        "description": "SSH Key or password for the Virtual Machine. SSH key
is recommended."
      }
    }
  },
  "variables": {
    "copy": [
      {
        "name": "azureFirewallIpConfigurations",
        "count": "[length(range(0,
parameters('numberOfFirewallPublicIPAddresses')))]",
        "input": {
          "name": "[format('IpConf{0}', range(0,
parameters('numberOfFirewallPublicIPAddresses'))
[copyIndex('azureFirewallIpConfigurations')])]",
          "properties": {
            "subnet": {
              "id": "[if(equals(range(0,
parameters('numberOfFirewallPublicIPAddresses'))
[copyIndex('azureFirewallIpConfigurations')], 0),
variables('azureFirewallSubnetId'), null())]"
            },
            "publicIPAddress": {
              "id": "[resourceId('Microsoft.Network/publicIPAddresses',
format('{0}{1}', variables('publicIPNamePrefix'), add(range(0,
parameters('numberOfFirewallPublicIPAddresses'))[range(0,
parameters('numberOfFirewallPublicIPAddresses'))
[copyIndex('azureFirewallIpConfigurations')]], 1)))]"
            }
          }
        }
      }
    ],
    "vnetAddressPrefix": "10.0.0.0/16",
    "serversSubnetPrefix": "10.0.2.0/24",
    "azureFirewallSubnetPrefix": "10.0.1.0/24",
    "jumpboxSubnetPrefix": "10.0.0.0/24",
    "nextHopIP": "10.0.1.4",
    "azureFirewallSubnetName": "AzureFirewallSubnet",
    "jumpBoxSubnetName": "JumpboxSubnet",
    "serversSubnetName": "ServersSubnet",
```

```
    "jumpBoxPublicIPAddressName": "JumpHostPublicIP",
    "jumpBoxNsgName": "JumpHostNSG",
    "jumpBoxNicName": "JumpHostNic",
    "jumpBoxSubnetId": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('virtualNetworkName'), variables('jumpBoxSubnetName'))]",
    "serverNicName": "ServerNic",
    "serverSubnetId": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('virtualNetworkName'), variables('serversSubnetName'))]",
    "storageAccountName": "[format('{0}sajumpbox',
uniqueString(resourceGroup().id))]",
    "azfwRouteTableName": "AzfwRouteTable",
    "firewallName": "firewall1",
    "publicIPNamePrefix": "publicIP",
    "azureFirewallSubnetId": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('virtualNetworkName'), variables('azureFirewallSubnetName'))]",
    "linuxConfiguration": {
      "disablePasswordAuthentication": true,
      "ssh": {
        "publicKeys": [
          {
            "path": "[format('/home/{0}/.ssh/authorized_keys',
parameters('adminUsername'))]",
            "keyData": "[parameters('adminPasswordOrKey')]"
          }
        ]
      }
    },
    "networkSecurityGroupName": "[format('{0}-nsg',
variables('serversSubnetName'))]"
  },
  "resources": [
    {
      "type": "Microsoft.Network/ipGroups",
      "apiVersion": "2021-08-01",
      "name": "[parameters('ipgroups_name1')]",
      "location": "[parameters('location')]",
      "properties": {
        "ipAddresses": [
          "13.73.64.64/26",
          "13.73.208.128/25",
          "52.126.194.0/23"
        ]
      }
    },
    {
      "type": "Microsoft.Network/ipGroups",
      "apiVersion": "2021-08-01",
      "name": "[parameters('ipgroups_name2')]",
      "location": "[parameters('location')]",
      "properties": {
        "ipAddresses": [
          "12.0.0.0/24",
```

```
          "13.9.0.0/24"
        ]
      }
    },
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-09-01",
      "name": "[variables('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "StorageV2",
      "properties": {}
    },
    {
      "type": "Microsoft.Network/routeTables",
      "apiVersion": "2021-08-01",
      "name": "[variables('azfwRouteTableName')]",
      "location": "[parameters('location')]",
      "properties": {
        "disableBgpRoutePropagation": false,
        "routes": [
          {
            "name": "AzfwDefaultRoute",
            "properties": {
              "addressPrefix": "0.0.0.0/0",
              "nextHopType": "VirtualAppliance",
              "nextHopIpAddress": "[variables('nextHopIP')]"
            }
          }
        ]
      }
    },
    {
      "type": "Microsoft.Network/networkSecurityGroups",
      "apiVersion": "2021-08-01",
      "name": "[variables('networkSecurityGroupName')]",
      "location": "[parameters('location')]",
      "properties": {}
    },
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "2021-08-01",
      "name": "[parameters('virtualNetworkName')]",
      "location": "[parameters('location')]",
      "tags": {
        "displayName": "[parameters('virtualNetworkName')]"
      },
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[variables('vnetAddressPrefix')]"
          ]
        },
```

```
        "subnets": [
          {
            "name": "[variables('jumpBoxSubnetName')]",
            "properties": {
              "addressPrefix": "[variables('jumpboxSubnetPrefix')]"
            }
          },
          {
            "name": "[variables('azureFirewallSubnetName')]",
            "properties": {
              "addressPrefix": "[variables('azureFirewallSubnetPrefix')]"
            }
          },
          {
            "name": "[variables('serversSubnetName')]",
            "properties": {
              "addressPrefix": "[variables('serversSubnetPrefix')]",
              "routeTable": {
                "id": "[resourceId('Microsoft.Network/routeTables',
variables('azfwRouteTableName'))]"
              },
              "networkSecurityGroup": {
                "id": "
[resourceId('Microsoft.Network/networkSecurityGroups',
variables('networkSecurityGroupName'))]"
              }
            }
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/routeTables',
variables('azfwRouteTableName'))]",
        "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('networkSecurityGroupName'))]"
      ]
    },
    {
      "copy": {
        "name": "publicIP",
        "count": "[length(range(0,
parameters('numberOfFirewallPublicIPAddresses')))]"
      },
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2021-08-01",
      "name": "[format('{0}{1}', variables('publicIPNamePrefix'),
add(range(0, parameters('numberOfFirewallPublicIPAddresses'))[copyIndex()],
1))]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard"
      },
      "properties": {
        "publicIPAllocationMethod": "Static",
        "publicIPAddressVersion": "IPv4"
```

```
      }
    },
    {
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2021-08-01",
      "name": "[variables('jumpBoxPublicIPAddressName')]",
      "location": "[parameters('location')]",
      "properties": {
        "publicIPAllocationMethod": "Dynamic"
      }
    },
    {
      "type": "Microsoft.Network/networkSecurityGroups",
      "apiVersion": "2021-08-01",
      "name": "[variables('jumpBoxNsgName')]",
      "location": "[parameters('location')]",
      "properties": {
        "securityRules": [
          {
            "name": "myNetworkSecurityGroupRuleSSH",
            "properties": {
              "protocol": "Tcp",
              "sourcePortRange": "*",
              "destinationPortRange": "22",
              "sourceAddressPrefix": "*",
              "destinationAddressPrefix": "*",
              "access": "Allow",
              "priority": 1000,
              "direction": "Inbound"
            }
          }
        ]
      }
    },
    {
      "type": "Microsoft.Network/networkInterfaces",
      "apiVersion": "2021-08-01",
      "name": "[variables('jumpBoxNicName')]",
      "location": "[parameters('location')]",
      "properties": {
        "ipConfigurations": [
          {
            "name": "ipconfig1",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses',
variables('jumpBoxPublicIPAddressName'))]"
              },
              "subnet": {
                "id": "[variables('jumpBoxSubnetId')]"
              }
            }
          }
        ],
```

```
        "networkSecurityGroup": {
          "id": "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('jumpBoxNsgName'))]"
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('jumpBoxNsgName'))]",
        "[resourceId('Microsoft.Network/publicIPAddresses',
variables('jumpBoxPublicIPAddressName'))]",
        "[resourceId('Microsoft.Network/virtualNetworks',
parameters('virtualNetworkName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/networkInterfaces",
      "apiVersion": "2021-08-01",
      "name": "[variables('serverNicName')]",
      "location": "[parameters('location')]",
      "properties": {
        "ipConfigurations": [
          {
            "name": "ipconfig1",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "subnet": {
                "id": "[variables('serverSubnetId')]"
              }
            }
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/virtualNetworks',
parameters('virtualNetworkName'))]"
      ]
    },
    {
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2022-03-01",
      "name": "JumpBox",
      "location": "[parameters('location')]",
      "properties": {
        "hardwareProfile": {
          "vmSize": "[parameters('vmSize')]"
        },
        "storageProfile": {
          "imageReference": {
            "publisher": "Canonical",
            "offer": "UbuntuServer",
            "sku": "18.04-LTS",
            "version": "latest"
          },
          "osDisk": {
            "createOption": "FromImage"
```

```
          }
        },
        "osProfile": {
          "computerName": "JumpBox",
          "adminUsername": "[parameters('adminUsername')]",
          "adminPassword": "[parameters('adminPasswordOrKey')]",
          "linuxConfiguration": "
[if(equals(parameters('authenticationType'), 'password'), json('null'),
variables('linuxConfiguration'))]"
        },
        "networkProfile": {
          "networkInterfaces": [
            {
              "id": "[resourceId('Microsoft.Network/networkInterfaces',
variables('jumpBoxNicName'))]"
            }
          ]
        },
        "diagnosticsProfile": {
          "bootDiagnostics": {
            "enabled": true,
            "storageUri": "
[reference(resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))).primaryEndpoints.blob]"
          }
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkInterfaces',
variables('jumpBoxNicName'))]",
        "[resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))]"
      ]
    },
    {
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2022-03-01",
      "name": "Server",
      "location": "[parameters('location')]",
      "properties": {
        "hardwareProfile": {
          "vmSize": "[parameters('vmSize')]"
        },
        "storageProfile": {
          "imageReference": {
            "publisher": "Canonical",
            "offer": "UbuntuServer",
            "sku": "18.04-LTS",
            "version": "latest"
          },
          "osDisk": {
            "createOption": "FromImage"
          }
        },
        "osProfile": {
```

```
          "computerName": "Server",
          "adminUsername": "[parameters('adminUsername')]",
          "adminPassword": "[parameters('adminPasswordOrKey')]",
          "linuxConfiguration": "
[if(equals(parameters('authenticationType'), 'password'), json('null'),
variables('linuxConfiguration'))]"
        },
        "networkProfile": {
          "networkInterfaces": [
            {
              "id": "[resourceId('Microsoft.Network/networkInterfaces',
variables('serverNicName'))]"
            }
          ]
        },
        "diagnosticsProfile": {
          "bootDiagnostics": {
            "enabled": true,
            "storageUri": "
[reference(resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))).primaryEndpoints.blob]"
          }
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkInterfaces',
variables('serverNicName'))]",
        "[resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/azureFirewalls",
      "apiVersion": "2021-08-01",
      "name": "[variables('firewallName')]",
      "location": "[parameters('location')]",
      "properties": {
        "ipConfigurations": "[variables('azureFirewallIpConfigurations')]",
        "applicationRuleCollections": [
          {
            "name": "appRc1",
            "properties": {
              "priority": 101,
              "action": {
                "type": "Allow"
              },
              "rules": [
                {
                  "name": "someAppRule",
                  "protocols": [
                    {
                      "protocolType": "Http",
                      "port": 8080
                    }
                  ],
```

```
                  "targetFqdns": [
                    "*bing.com"
                  ],
                  "sourceIpGroups": [
                    "[resourceId('Microsoft.Network/ipGroups',
parameters('ipgroups_name1'))]"
                  ]
                },
                {
                  "name": "someOtherAppRule",
                  "protocols": [
                    {
                      "protocolType": "Mssql",
                      "port": 1433
                    }
                  ],
                  "targetFqdns": [
                    "[format('sql1{0}',
environment().suffixes.sqlServerHostname)]"
                  ],
                  "sourceIpGroups": [
                    "[resourceId('Microsoft.Network/ipGroups',
parameters('ipgroups_name1'))]",
                    "[resourceId('Microsoft.Network/ipGroups',
parameters('ipgroups_name2'))]"
                  ]
                }
              ]
            }
          }
        ],
        "networkRuleCollections": [
          {
            "name": "netRc1",
            "properties": {
              "priority": 200,
              "action": {
                "type": "Allow"
              },
              "rules": [
                {
                  "name": "networkRule",
                  "description": "desc1",
                  "protocols": [
                    "UDP",
                    "TCP",
                    "ICMP"
                  ],
                  "sourceAddresses": [
                    "10.0.0.0",
                    "111.1.0.0/23"
                  ],
                  "sourceIpGroups": [
                    "[resourceId('Microsoft.Network/ipGroups',
parameters('ipgroups_name1'))]"
```

```
                    ],
                    "destinationIpGroups": [
                      "[resourceId('Microsoft.Network/ipGroups',
parameters('ipgroups_name2'))]"
                    ],
                    "destinationPorts": [
                      "90"
                    ]
                  }
                ]
              }
            }
          ]
        },
        "dependsOn": [
          "[resourceId('Microsoft.Network/ipGroups',
parameters('ipgroups_name1'))]",
          "[resourceId('Microsoft.Network/ipGroups',
parameters('ipgroups_name2'))]",
          "publicIP",
          "[resourceId('Microsoft.Network/virtualNetworks',
parameters('virtualNetworkName'))]"
        ]
      }
    ]
}
```

Multiple Azure resources are defined in the template:

- Microsoft.Network/ipGroups
- Microsoft.Storage/storageAccounts
- Microsoft.Network/routeTables
- Microsoft.Network/networkSecurityGroups
- Microsoft.Network/virtualNetworks
- Microsoft.Network/publicIPAddresses
- Microsoft.Network/networkInterfaces
- Microsoft.Compute/virtualMachines
- Microsoft.Network/azureFirewalls

Deploy the template

Deploy the ARM template to Azure:

1. Select **Deploy to Azure** to sign in to Azure and open the template. The template creates an Azure Firewall, the network infrastructure, and two virtual machines.

   Deploy to Azure

2. In the portal, on the **Create an Azure Firewall with IpGroups** page, type or select the following values:

    - Subscription: Select from existing subscriptions
    - Resource group: Select from existing resource groups or select **Create new**, and select **OK**.
    - Location: Select a location
    - Virtual Network Name: Type a name for the new virtual network (VNet)
    - IP Group Name 1: Type name for IP Group 1
    - IP Group Name 2: Type name for IP Group 2
    - Admin Username: Type username for the administrator user account
    - Authentication: Select sshPublicKey or password
    - Admin Password: Type an administrator password or key

3. Select **I agree to the terms and conditions stated above** and then select **Purchase**. The deployment can take 10 minutes or longer to complete.

Review deployed resources

In the Azure portal, review the deployed resources, especially the firewall rules that use IP Groups.

To learn about the JSON syntax and properties for a firewall in a template, see Microsoft.Network azureFirewalls template reference.

Clean up resources

When you no longer need the resources that you created with the firewall, delete the resource group. This removes the firewall and all the related resources.

To delete the resource group, call the `Remove-AzResourceGroup` cmdlet:

Azure PowerShell

```
Remove-AzResourceGroup -Name "<your resource group name>"
```

Next steps

Tutorial: Deploy and configure Azure Firewall in a hybrid network using the Azure portal

Quickstart: Create an Azure Firewall with multiple public IP addresses - ARM template

Article • 03/08/2023

In this quickstart, you use an Azure Resource Manager template (ARM template) to deploy an Azure Firewall with multiple public IP addresses from a public IP address prefix. The deployed firewall has NAT rule collection rules that allow RDP connections to two Windows Server 2019 virtual machines.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

For more information about Azure Firewall with multiple public IP addresses, see Deploy an Azure Firewall with multiple public IP addresses using Azure PowerShell.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

- An Azure account with an active subscription. Create an account for free.

Review the template

This template creates an Azure Firewall with two public IP addresses, along with the necessary resources to support the Azure Firewall.

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
```

```
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.8.9.13224",
      "templateHash": "15231316908103169117"
    }
  },
  "parameters": {
    "adminUsername": {
      "type": "string",
      "metadata": {
        "description": "Admin username for the backend servers"
      }
    },
    "adminPassword": {
      "type": "secureString",
      "metadata": {
        "description": "Password for the admin account on the backend
servers"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "vmSize": {
      "type": "string",
      "defaultValue": "Standard_B2ms",
      "metadata": {
        "description": "Size of the virtual machine."
      }
    }
  },
  "variables": {
    "copy": [
      {
        "name": "azureFirewallIpConfigurations",
        "count": "[length(range(0, 2))]",
        "input": {
          "name": "[format('IpConf{0}', add(range(0, 2)
[copyIndex('azureFirewallIpConfigurations')], 1))]",
          "properties": {
            "subnet": "[if(equals(range(0, 2)
[copyIndex('azureFirewallIpConfigurations')], 0), json(format('{{\"id\": \"
{0}\"}}', variables('azureFirewallSubnetId'))), json('null'))]",
            "publicIPAddress": {
              "id": "[resourceId('Microsoft.Network/publicIPAddresses',
format('{0}{1}', variables('publicIpAddressName'), add(range(0, 2)[range(0,
2)[copyIndex('azureFirewallIpConfigurations')]], 1)))]"
            }
          }
```

```
        }
      }
    ],
    "virtualMachineName": "myVM",
    "virtualNetworkName": "myVNet",
    "networkInterfaceName": "net-int",
    "ipConfigName": "ipconfig",
    "ipPrefixName": "public_ip_prefix",
    "ipPrefixSize": 31,
    "publicIpAddressName": "public_ip",
    "nsgName": "vm-nsg",
    "firewallName": "FW-01",
    "vnetPrefix": "10.0.0.0/16",
    "fwSubnetPrefix": "10.0.0.0/24",
    "backendSubnetPrefix": "10.0.1.0/24",
    "azureFirewallSubnetId": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
variables('virtualNetworkName'), 'AzureFirewallSubnet')]"
  },
  "resources": [
    {
      "copy": {
        "name": "nsg",
        "count": "[length(range(0, 2))]"
      },
      "type": "Microsoft.Network/networkSecurityGroups",
      "apiVersion": "2022-01-01",
      "name": "[format('{0}{1}', variables('nsgName'), add(range(0, 2)
[copyIndex()], 1))]",
      "location": "[parameters('location')]",
      "properties": {
        "securityRules": [
          {
            "name": "RDP",
            "properties": {
              "protocol": "Tcp",
              "sourcePortRange": "*",
              "destinationPortRange": "3389",
              "sourceAddressPrefix": "*",
              "destinationAddressPrefix": "*",
              "access": "Allow",
              "priority": 300,
              "direction": "Inbound"
            }
          }
        ]
      }
    },
    {
      "type": "Microsoft.Network/publicIPPrefixes",
      "apiVersion": "2022-01-01",
      "name": "[variables('ipPrefixName')]",
      "location": "[parameters('location')]",
      "properties": {
        "prefixLength": "[variables('ipPrefixSize')]",
```

```
        "publicIPAddressVersion": "IPv4"
      },
      "sku": {
        "name": "Standard"
      }
    },
    {
      "copy": {
        "name": "publicIPAddress",
        "count": "[length(range(0, 2))]"
      },
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2022-01-01",
      "name": "[format('{0}{1}', variables('publicIpAddressName'),
add(range(0, 2)[copyIndex()], 1))]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard"
      },
      "properties": {
        "publicIPAddressVersion": "IPv4",
        "publicIPAllocationMethod": "Static",
        "publicIPPrefix": {
          "id": "[resourceId('Microsoft.Network/publicIPPrefixes',
variables('ipPrefixName'))]"
        },
        "idleTimeoutInMinutes": 4
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/publicIPPrefixes',
variables('ipPrefixName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "2022-01-01",
      "name": "[variables('virtualNetworkName')]",
      "location": "[parameters('location')]",
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[variables('vnetPrefix')]"
          ]
        },
        "subnets": [
          {
            "name": "myBackendSubnet",
            "properties": {
              "addressPrefix": "[variables('backendSubnetPrefix')]",
              "routeTable": {
                "id": "[resourceId('Microsoft.Network/routeTables', 'rt-
01')]"
              },
              "privateEndpointNetworkPolicies": "Enabled",
              "privateLinkServiceNetworkPolicies": "Enabled"
```

```
            }
          }
        ],
        "enableDdosProtection": false,
        "enableVmProtection": false
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/routeTables', 'rt-01')]"
      ]
    },
    {
      "type": "Microsoft.Network/virtualNetworks/subnets",
      "apiVersion": "2022-01-01",
      "name": "[format('{0}/{1}', variables('virtualNetworkName'),
'AzureFirewallSubnet')]",
      "properties": {
        "addressPrefix": "[variables('fwSubnetPrefix')]",
        "privateEndpointNetworkPolicies": "Enabled",
        "privateLinkServiceNetworkPolicies": "Enabled"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]"
      ]
    },
    {
      "copy": {
        "name": "virtualMachine",
        "count": "[length(range(0, 2))]"
      },
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2022-03-01",
      "name": "[format('{0}{1}', variables('virtualMachineName'),
add(range(0, 2)[copyIndex()], 1))]",
      "location": "[parameters('location')]",
      "properties": {
        "hardwareProfile": {
          "vmSize": "[parameters('vmSize')]"
        },
        "storageProfile": {
          "imageReference": {
            "publisher": "MicrosoftWindowsServer",
            "offer": "WindowsServer",
            "sku": "2019-Datacenter",
            "version": "latest"
          },
          "osDisk": {
            "osType": "Windows",
            "createOption": "FromImage",
            "caching": "ReadWrite",
            "managedDisk": {
              "storageAccountType": "StandardSSD_LRS"
            },
            "diskSizeGB": 127
          }
```

```
        },
        "osProfile": {
          "computerName": "[format('{0}{1}',
variables('virtualMachineName'), add(range(0, 2)[copyIndex()], 1))]",
          "adminUsername": "[parameters('adminUsername')]",
          "adminPassword": "[parameters('adminPassword')]",
          "windowsConfiguration": {
            "provisionVMAgent": true,
            "enableAutomaticUpdates": true
          },
          "allowExtensionOperations": true
        },
        "networkProfile": {
          "networkInterfaces": [
            {
              "id": "[resourceId('Microsoft.Network/networkInterfaces',
format('{0}{1}', variables('networkInterfaceName'), add(range(0, 2)[range(0,
2)[copyIndex()]], 1)))]"
            }
          ]
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkInterfaces', format('{0}{1}',
variables('networkInterfaceName'), add(range(0, 2)[range(0, 2)
[copyIndex()]], 1)))]"
      ]
    },
    {
      "copy": {
        "name": "netInterface",
        "count": "[length(range(0, 2))]"
      },
      "type": "Microsoft.Network/networkInterfaces",
      "apiVersion": "2022-01-01",
      "name": "[format('{0}{1}', variables('networkInterfaceName'),
add(range(0, 2)[copyIndex()], 1))]",
      "location": "[parameters('location')]",
      "properties": {
        "ipConfigurations": [
          {
            "name": "[format('{0}{1}', variables('ipConfigName'),
add(range(0, 2)[copyIndex()], 1))]",
            "properties": {
              "subnet": {
                "id": "
[reference(resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))).subnets[0].id]"
              },
              "primary": true
            }
          }
        ],
        "enableAcceleratedNetworking": false,
        "enableIPForwarding": false,
```

```
        "networkSecurityGroup": {
          "id": "[resourceId('Microsoft.Network/networkSecurityGroups',
format('{0}{1}', variables('nsgName'), add(range(0, 2)[range(0, 2)
[copyIndex()]], 1)))]"
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkSecurityGroups', format('{0}
{1}', variables('nsgName'), add(range(0, 2)[range(0, 2)[copyIndex()]],
1)))]",
        "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/azureFirewalls",
      "apiVersion": "2022-01-01",
      "name": "[variables('firewallName')]",
      "location": "[parameters('location')]",
      "properties": {
        "sku": {
          "name": "AZFW_VNet",
          "tier": "Standard"
        },
        "threatIntelMode": "Alert",
        "ipConfigurations": "[variables('azureFirewallIpConfigurations')]",
        "applicationRuleCollections": [
          {
            "name": "web",
            "properties": {
              "priority": 100,
              "action": {
                "type": "Allow"
              },
              "rules": [
                {
                  "name": "wan-address",
                  "protocols": [
                    {
                      "protocolType": "Http",
                      "port": 80
                    },
                    {
                      "protocolType": "Https",
                      "port": 443
                    }
                  ],
                  "targetFqdns": [
                    "getmywanip.com"
                  ],
                  "sourceAddresses": [
                    "*"
                  ]
                },
                {
```

```
                "name": "google",
                "protocols": [
                  {
                    "protocolType": "Http",
                    "port": 80
                  },
                  {
                    "protocolType": "Https",
                    "port": 443
                  }
                ],
                "targetFqdns": [
                  "www.google.com"
                ],
                "sourceAddresses": [
                  "10.0.1.0/24"
                ]
              },
              {
                "name": "wupdate",
                "protocols": [
                  {
                    "protocolType": "Http",
                    "port": 80
                  },
                  {
                    "protocolType": "Https",
                    "port": 443
                  }
                ],
                "fqdnTags": [
                  "WindowsUpdate"
                ],
                "sourceAddresses": [
                  "*"
                ]
              }
            ]
          }
        }
      ],
      "natRuleCollections": [
        {
          "name": "Coll-01",
          "properties": {
            "priority": 100,
            "action": {
              "type": "Dnat"
            },
            "rules": [
              {
                "name": "rdp-01",
                "protocols": [
                  "TCP"
                ],
```

```
                  "translatedAddress": "10.0.1.4",
                  "translatedPort": "3389",
                  "sourceAddresses": [
                    "*"
                  ],
                  "destinationAddresses": [
                    "
[reference(resourceId('Microsoft.Network/publicIPAddresses', format('{0}
{1}', variables('publicIpAddressName'), add(range(0, 2)[0],
1)))).ipAddress]"
                  ],
                  "destinationPorts": [
                    "3389"
                  ]
                },
                {
                  "name": "rdp-02",
                  "protocols": [
                    "TCP"
                  ],
                  "translatedAddress": "10.0.1.5",
                  "translatedPort": "3389",
                  "sourceAddresses": [
                    "*"
                  ],
                  "destinationAddresses": [
                    "
[reference(resourceId('Microsoft.Network/publicIPAddresses', format('{0}
{1}', variables('publicIpAddressName'), add(range(0, 2)[1],
1)))).ipAddress]"
                  ],
                  "destinationPorts": [
                    "3389"
                  ]
                }
              ]
            }
          }
        ]
      },
      "dependsOn": [
        "publicIPAddress",
        "[resourceId('Microsoft.Network/virtualNetworks/subnets',
variables('virtualNetworkName'), 'AzureFirewallSubnet')]"
      ]
    },
    {
      "type": "Microsoft.Network/routeTables",
      "apiVersion": "2022-01-01",
      "name": "rt-01",
      "location": "[parameters('location')]",
      "properties": {
        "disableBgpRoutePropagation": false,
        "routes": [
          {
```

```
            "name": "fw",
            "properties": {
              "addressPrefix": "0.0.0.0/0",
              "nextHopType": "VirtualAppliance",
              "nextHopIpAddress": "10.0.0.4"
            }
          }
        ]
      }
    }
  ]
}
```

Multiple Azure resources are defined in the template:

- Microsoft.Network/networkSecurityGroups
- Microsoft.Network/publicIPPrefix
- Microsoft.Network/publicIPAddresses
- Microsoft.Network/virtualNetworks
- Microsoft.Compute/virtualMachines
- Microsoft.Storage/storageAccounts
- Microsoft.Network/networkInterfaces
- Microsoft.Network/azureFirewalls
- Microsoft.Network/routeTables

Deploy the template

Deploy the ARM template to Azure:

1. Select **Deploy to Azure** to sign in to Azure and open the template. The template creates an Azure Firewall, the network infrastructure, and two virtual machines.

   Deploy to Azure

2. In the portal, on the **Create an Azure Firewall with multiple IP public addresses** page, type or select the following values:

   - Subscription: Select from existing subscriptions
   - Resource group: Select from existing resource groups or select **Create new**, and select **OK**.
   - Location: Select a location
   - Admin Username: Type username for the administrator user account
   - Admin Password: Type an administrator password or key

3. Select **I agree to the terms and conditions stated above** and then select **Purchase**. The deployment can take 10 minutes or longer to complete.

Validate the deployment

In the Azure portal, review the deployed resources. Note the firewall public IP addresses.

Use Remote Desktop Connection to connect to the firewall public IP addresses. Successful connections demonstrate firewall NAT rules that allow the connection to the backend servers.

Clean up resources

When you no longer need the resources that you created with the firewall, delete the resource group. Deleting the resource group removes the firewall and all the related resources.

To delete the resource group, call the `Remove-AzResourceGroup` cmdlet:

Azure PowerShell

```
Remove-AzResourceGroup -Name "<your resource group name>"
```

Next steps

Tutorial: Deploy and configure Azure Firewall in a hybrid network using the Azure portal

Quickstart: Create an Azure Firewall and a firewall policy - ARM template

Article • 03/08/2023

In this quickstart, you use an Azure Resource Manager template (ARM template) to create an Azure Firewall and a firewall policy. The firewall policy has an application rule that allows connections to `www.microsoft.com` and a rule that allows connections to Windows Update using the **WindowsUpdate** FQDN tag. A network rule allows UDP connections to a time server at 13.86.101.172.

Also, IP Groups are used in the rules to define the **Source** IP addresses.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

For information about Azure Firewall Manager, see What is Azure Firewall Manager?.

For information about Azure Firewall, see What is Azure Firewall?.

For information about IP Groups, see IP Groups in Azure Firewall.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

- An Azure account with an active subscription. Create an account for free.

Review the template

This template creates a hub virtual network, along with the necessary resources to support the scenario.

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.8.9.13224",
      "templateHash": "2614956787969031174"
    }
  },
  "parameters": {
    "virtualNetworkName": {
      "type": "string",
      "defaultValue": "[format('vnet{0}',
uniqueString(resourceGroup().id))]",
      "metadata": {
        "description": "Virtual network name"
      }
    },
    "firewallName": {
      "type": "string",
      "defaultValue": "[format('fw{0}', uniqueString(resourceGroup().id))]",
      "metadata": {
        "description": "Azure Firewall name"
      }
    },
    "numberOfPublicIPAddresses": {
      "type": "int",
      "defaultValue": 2,
      "maxValue": 100,
      "minValue": 1,
      "metadata": {
        "description": "Number of public IP addresses for the Azure
Firewall"
      }
    },
    "availabilityZones": {
      "type": "array",
      "defaultValue": [],
      "metadata": {
        "description": "Zone numbers e.g. 1,2,3."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "infraIpGroupName": {
      "type": "string",
      "defaultValue": "[format('{0}-infra-ipgroup-{1}',
```

```
parameters('location'), uniqueString(resourceGroup().id))]"
    },
    "workloadIpGroupName": {
      "type": "string",
      "defaultValue": "[format('{0}-workload-ipgroup-{1}',
parameters('location'), uniqueString(resourceGroup().id))]"
    },
    "firewallPolicyName": {
      "type": "string",
      "defaultValue": "[format('{0}-firewallPolicy',
parameters('firewallName'))]"
    }
  },
  "variables": {
    "copy": [
      {
        "name": "azureFirewallIpConfigurations",
        "count": "[length(range(0,
parameters('numberOfPublicIPAddresses')))]",
        "input": {
          "name": "[format('IpConf{0}', range(0,
parameters('numberOfPublicIPAddresses'))
[copyIndex('azureFirewallIpConfigurations')])]",
          "properties": {
            "subnet": "[if(equals(range(0,
parameters('numberOfPublicIPAddresses'))
[copyIndex('azureFirewallIpConfigurations')], 0), json(format('{{\"id\": \"
{0}\"}}', variables('azureFirewallSubnetId'))), json('null'))]",
            "publicIPAddress": {
              "id": "[format('{0}{1}', variables('azureFirewallPublicIpId'),
add(range(0, parameters('numberOfPublicIPAddresses'))
[copyIndex('azureFirewallIpConfigurations')], 1))]"
            }
          }
        }
      }
    ],
    "vnetAddressPrefix": "10.10.0.0/24",
    "azureFirewallSubnetPrefix": "10.10.0.0/25",
    "publicIPNamePrefix": "publicIP",
    "azurepublicIpname": "[variables('publicIPNamePrefix')]",
    "azureFirewallSubnetName": "AzureFirewallSubnet",
    "azureFirewallSubnetId": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('virtualNetworkName'), variables('azureFirewallSubnetName'))]",
    "azureFirewallPublicIpId": "
[resourceId('Microsoft.Network/publicIPAddresses',
variables('publicIPNamePrefix'))]"
  },
  "resources": [
    {
      "type": "Microsoft.Network/ipGroups",
      "apiVersion": "2022-01-01",
      "name": "[parameters('workloadIpGroupName')]",
      "location": "[parameters('location')]",
```

```
      "properties": {
        "ipAddresses": [
          "10.20.0.0/24",
          "10.30.0.0/24"
        ]
      }
    },
    {
      "type": "Microsoft.Network/ipGroups",
      "apiVersion": "2022-01-01",
      "name": "[parameters('infraIpGroupName')]",
      "location": "[parameters('location')]",
      "properties": {
        "ipAddresses": [
          "10.40.0.0/24",
          "10.50.0.0/24"
        ]
      }
    },
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "2022-01-01",
      "name": "[parameters('virtualNetworkName')]",
      "location": "[parameters('location')]",
      "tags": {
        "displayName": "[parameters('virtualNetworkName')]"
      },
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[variables('vnetAddressPrefix')]"
          ]
        },
        "subnets": [
          {
            "name": "[variables('azureFirewallSubnetName')]",
            "properties": {
              "addressPrefix": "[variables('azureFirewallSubnetPrefix')]"
            }
          }
        ],
        "enableDdosProtection": false
      }
    },
    {
      "copy": {
        "name": "publicIpAddress",
        "count": "[length(range(0,
parameters('numberOfPublicIPAddresses')))]"
      },
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2022-01-01",
      "name": "[format('{0}{1}', variables('azurepublicIpname'),
add(range(0, parameters('numberOfPublicIPAddresses'))[copyIndex()], 1))]",
      "location": "[parameters('location')]",
```

```
      "sku": {
        "name": "Standard"
      },
      "properties": {
        "publicIPAllocationMethod": "Static",
        "publicIPAddressVersion": "IPv4"
      }
    },
    {
      "type": "Microsoft.Network/firewallPolicies",
      "apiVersion": "2022-01-01",
      "name": "[parameters('firewallPolicyName')]",
      "location": "[parameters('location')]",
      "properties": {
        "threatIntelMode": "Alert"
      }
    },
    {
      "type": "Microsoft.Network/firewallPolicies/ruleCollectionGroups",
      "apiVersion": "2022-01-01",
      "name": "[format('{0}/{1}', parameters('firewallPolicyName'),
'DefaultNetworkRuleCollectionGroup')]",
      "properties": {
        "priority": 200,
        "ruleCollections": [
          {
            "ruleCollectionType": "FirewallPolicyFilterRuleCollection",
            "action": {
              "type": "Allow"
            },
            "name": "azure-global-services-nrc",
            "priority": 1250,
            "rules": [
              {
                "ruleType": "NetworkRule",
                "name": "time-windows",
                "ipProtocols": [
                  "UDP"
                ],
                "destinationAddresses": [
                  "13.86.101.172"
                ],
                "sourceIpGroups": [
                  "[resourceId('Microsoft.Network/ipGroups',
parameters('workloadIpGroupName'))]",
                  "[resourceId('Microsoft.Network/ipGroups',
parameters('infraIpGroupName'))]"
                ],
                "destinationPorts": [
                  "123"
                ]
              }
            ]
          }
        ]
```

```
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/firewallPolicies',
parameters('firewallPolicyName'))]",
        "[resourceId('Microsoft.Network/ipGroups',
parameters('infraIpGroupName'))]",
        "[resourceId('Microsoft.Network/ipGroups',
parameters('workloadIpGroupName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/firewallPolicies/ruleCollectionGroups",
      "apiVersion": "2022-01-01",
      "name": "[format('{0}/{1}', parameters('firewallPolicyName'),
'DefaultApplicationRuleCollectionGroup')]",
      "properties": {
        "priority": 300,
        "ruleCollections": [
          {
            "ruleCollectionType": "FirewallPolicyFilterRuleCollection",
            "name": "global-rule-url-arc",
            "priority": 1000,
            "action": {
              "type": "Allow"
            },
            "rules": [
              {
                "ruleType": "ApplicationRule",
                "name": "winupdate-rule-01",
                "protocols": [
                  {
                    "protocolType": "Https",
                    "port": 443
                  },
                  {
                    "protocolType": "Http",
                    "port": 80
                  }
                ],
                "fqdnTags": [
                  "WindowsUpdate"
                ],
                "terminateTLS": false,
                "sourceIpGroups": [
                  "[resourceId('Microsoft.Network/ipGroups',
parameters('workloadIpGroupName'))]",
                  "[resourceId('Microsoft.Network/ipGroups',
parameters('infraIpGroupName'))]"
                ]
              }
            ]
          },
          {
            "ruleCollectionType": "FirewallPolicyFilterRuleCollection",
            "action": {
```

```
              "type": "Allow"
            },
            "name": "Global-rules-arc",
            "priority": 1202,
            "rules": [
              {
                "ruleType": "ApplicationRule",
                "name": "global-rule-01",
                "protocols": [
                  {
                    "protocolType": "Https",
                    "port": 443
                  }
                ],
                "targetFqdns": [
                  "www.microsoft.com"
                ],
                "terminateTLS": false,
                "sourceIpGroups": [
                  "[resourceId('Microsoft.Network/ipGroups',
parameters('workloadIpGroupName'))]",
                  "[resourceId('Microsoft.Network/ipGroups',
parameters('infraIpGroupName'))]"
                ]
              }
            ]
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/firewallPolicies',
parameters('firewallPolicyName'))]",
        "[resourceId('Microsoft.Network/ipGroups',
parameters('infraIpGroupName'))]",
        "
[resourceId('Microsoft.Network/firewallPolicies/ruleCollectionGroups',
parameters('firewallPolicyName'), 'DefaultNetworkRuleCollectionGroup')]",
        "[resourceId('Microsoft.Network/ipGroups',
parameters('workloadIpGroupName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/azureFirewalls",
      "apiVersion": "2021-03-01",
      "name": "[parameters('firewallName')]",
      "location": "[parameters('location')]",
      "zones": "[if(equals(length(parameters('availabilityZones')), 0),
null(), parameters('availabilityZones'))]",
      "properties": {
        "ipConfigurations": "[variables('azureFirewallIpConfigurations')]",
        "firewallPolicy": {
          "id": "[resourceId('Microsoft.Network/firewallPolicies',
parameters('firewallPolicyName'))]"
        }
      },
```

```
      "dependsOn": [
        "
[resourceId('Microsoft.Network/firewallPolicies/ruleCollectionGroups',
parameters('firewallPolicyName'),
'DefaultApplicationRuleCollectionGroup')]",
        "[resourceId('Microsoft.Network/firewallPolicies',
parameters('firewallPolicyName'))]",
        "[resourceId('Microsoft.Network/ipGroups',
parameters('infraIpGroupName'))]",
        "
[resourceId('Microsoft.Network/firewallPolicies/ruleCollectionGroups',
parameters('firewallPolicyName'), 'DefaultNetworkRuleCollectionGroup')]",
        "publicIpAddress",
        "[resourceId('Microsoft.Network/virtualNetworks',
parameters('virtualNetworkName'))]",
        "[resourceId('Microsoft.Network/ipGroups',
parameters('workloadIpGroupName'))]"
      ]
    }
  ]
}
```

Multiple Azure resources are defined in the template:

- Microsoft.Network/ipGroups
- Microsoft.Network/firewallPolicies
- Microsoft.Network/firewallPolicies/ruleCollectionGroups
- Microsoft.Network/azureFirewalls
- Microsoft.Network/virtualNetworks
- Microsoft.Network/publicIPAddresses

Deploy the template

Deploy the ARM template to Azure:

1. Select **Deploy to Azure** to sign in to Azure and open the template. The template creates an Azure Firewall, a virtual WAN and virtual hub, the network infrastructure, and two virtual machines.

   Deploy to Azure

2. In the portal, on the **Create a Firewall and FirewallPolicy with Rules and Ipgroups** page, type or select the following values:

   - Subscription: Select from existing subscriptions.
   - Resource group: Select from existing resource groups or select **Create new**, and select **OK**.

- Region: Select a region.
- Firewall Name: type a name for the firewall.

3. Select **Review + create** and then select **Create**. The deployment can take 10 minutes or longer to complete.

Review deployed resources

After deployment completes, you'll see the following similar resources.

Clean up resources

When you no longer need the resources that you created with the firewall, delete the resource group. This removes the firewall and all the related resources.

To delete the resource group, call the `Remove-AzResourceGroup` cmdlet:

Azure PowerShell

```
Remove-AzResourceGroup -Name "<your resource group name>"
```

Next steps

Azure Firewall Manager policy overview

Quickstart: Secure your virtual hub using Azure Firewall Manager - ARM template

Article • 03/08/2023

In this quickstart, you use an Azure Resource Manager template (ARM template) to secure your virtual hub using Azure Firewall Manager. The deployed firewall has an application rule that allows connections to `www.microsoft.com` . Two Windows Server 2019 virtual machines are deployed to test the firewall. One jump server is used to connect to the workload server. From the workload server, you can only connect to `www.microsoft.com`.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

For more information about Azure Firewall Manager, see What is Azure Firewall Manager?.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

- An Azure account with an active subscription. Create an account for free .

Review the template

This template creates a secured virtual hub using Azure Firewall Manager, along with the necessary resources to support the scenario.

The template used in this quickstart is from Azure Quickstart Templates .

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.9.1.41621",
      "templateHash": "1093777410793723808"
    }
  },
  "parameters": {
    "adminUsername": {
      "type": "string",
      "metadata": {
        "description": "Admin username for the servers"
      }
    },
    "adminPassword": {
      "type": "secureString",
      "metadata": {
        "description": "Password for the admin account on the servers"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "vmSize": {
      "type": "string",
      "defaultValue": "Standard_D2_v3",
      "metadata": {
        "description": "Size of the virtual machine."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Network/virtualWans",
      "apiVersion": "2021-08-01",
      "name": "VWan-01",
      "location": "[parameters('location')]",
      "properties": {
        "disableVpnEncryption": false,
        "allowBranchToBranchTraffic": true,
        "type": "Standard"
      }
    },
    {
      "type": "Microsoft.Network/virtualHubs",
      "apiVersion": "2021-08-01",
```

```
      "name": "Hub-01",
      "location": "[parameters('location')]",
      "properties": {
        "addressPrefix": "10.1.0.0/16",
        "virtualWan": {
          "id": "[resourceId('Microsoft.Network/virtualWans', 'VWan-01')]"
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/virtualWans', 'VWan-01')]"
      ]
    },
    {
      "type": "Microsoft.Network/virtualHubs/hubVirtualNetworkConnections",
      "apiVersion": "2021-08-01",
      "name": "[format('{0}/{1}', 'Hub-01', 'hub-spoke')]",
      "properties": {
        "remoteVirtualNetwork": {
          "id": "[resourceId('Microsoft.Network/virtualNetworks', 'Spoke-
01')]"
        },
        "allowHubToRemoteVnetTransit": true,
        "allowRemoteVnetToUseHubVnetGateways": false,
        "enableInternetSecurity": true,
        "routingConfiguration": {
          "associatedRouteTable": {
            "id": "
[resourceId('Microsoft.Network/virtualHubs/hubRouteTables', 'Hub-01',
'RT_VNet')]"
          },
          "propagatedRouteTables": {
            "labels": [
              "VNet"
            ],
            "ids": [
              {
                "id": "
[resourceId('Microsoft.Network/virtualHubs/hubRouteTables', 'Hub-01',
'RT_VNet')]"
              }
            ]
          }
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/azureFirewalls', 'AzfwTest')]",
        "[resourceId('Microsoft.Network/virtualHubs/hubRouteTables', 'Hub-
01', 'RT_VNet')]",
        "[resourceId('Microsoft.Network/virtualHubs', 'Hub-01')]",
        "[resourceId('Microsoft.Network/virtualNetworks', 'Spoke-01')]"
      ]
    },
    {
      "type": "Microsoft.Network/firewallPolicies",
      "apiVersion": "2021-08-01",
```

```
      "name": "Policy-01",
      "location": "[parameters('location')]",
      "properties": {
        "threatIntelMode": "Alert"
      }
    },
    {
      "type": "Microsoft.Network/firewallPolicies/ruleCollectionGroups",
      "apiVersion": "2021-08-01",
      "name": "[format('{0}/{1}', 'Policy-01',
'DefaultApplicationRuleCollectionGroup')]",
      "properties": {
        "priority": 300,
        "ruleCollections": [
          {
            "ruleCollectionType": "FirewallPolicyFilterRuleCollection",
            "name": "RC-01",
            "priority": 100,
            "action": {
              "type": "Allow"
            },
            "rules": [
              {
                "ruleType": "ApplicationRule",
                "name": "Allow-msft",
                "sourceAddresses": [
                  "*"
                ],
                "protocols": [
                  {
                    "port": 80,
                    "protocolType": "Http"
                  },
                  {
                    "port": 443,
                    "protocolType": "Https"
                  }
                ],
                "targetFqdns": [
                  "*.microsoft.com"
                ]
              }
            ]
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/firewallPolicies', 'Policy-01')]"
      ]
    },
    {
      "type": "Microsoft.Network/azureFirewalls",
      "apiVersion": "2021-08-01",
      "name": "AzfwTest",
      "location": "[parameters('location')]",
```

```
      "properties": {
        "sku": {
          "name": "AZFW_Hub",
          "tier": "Standard"
        },
        "hubIPAddresses": {
          "publicIPs": {
            "count": 1
          }
        },
        "virtualHub": {
          "id": "[resourceId('Microsoft.Network/virtualHubs', 'Hub-01')]"
        },
        "firewallPolicy": {
          "id": "[resourceId('Microsoft.Network/firewallPolicies', 'Policy-
01')]"
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/firewallPolicies', 'Policy-01')]",
        "[resourceId('Microsoft.Network/virtualHubs', 'Hub-01')]"
      ]
    },
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "2021-08-01",
      "name": "Spoke-01",
      "location": "[parameters('location')]",
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "10.0.0.0/16"
          ]
        },
        "enableDdosProtection": false,
        "enableVmProtection": false
      }
    },
    {
      "type": "Microsoft.Network/virtualNetworks/subnets",
      "apiVersion": "2021-08-01",
      "name": "[format('{0}/{1}', 'Spoke-01', 'Workload-SN')]",
      "properties": {
        "addressPrefix": "10.0.1.0/24",
        "privateEndpointNetworkPolicies": "Enabled",
        "privateLinkServiceNetworkPolicies": "Enabled"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/virtualNetworks', 'Spoke-01')]"
      ]
    },
    {
      "type": "Microsoft.Network/virtualNetworks/subnets",
      "apiVersion": "2021-08-01",
      "name": "[format('{0}/{1}', 'Spoke-01', 'Jump-SN')]",
```

```
      "properties": {
        "addressPrefix": "10.0.2.0/24",
        "routeTable": {
          "id": "[resourceId('Microsoft.Network/routeTables', 'RT-01')]"
        },
        "privateEndpointNetworkPolicies": "Enabled",
        "privateLinkServiceNetworkPolicies": "Enabled"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/routeTables', 'RT-01')]",
        "[resourceId('Microsoft.Network/virtualNetworks/subnets', 'Spoke-
01', 'Workload-SN')]",
        "[resourceId('Microsoft.Network/virtualNetworks', 'Spoke-01')]"
      ]
    },
    {
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2022-03-01",
      "name": "Jump-Srv",
      "location": "[parameters('location')]",
      "properties": {
        "hardwareProfile": {
          "vmSize": "[parameters('vmSize')]"
        },
        "storageProfile": {
          "imageReference": {
            "publisher": "MicrosoftWindowsServer",
            "offer": "WindowsServer",
            "sku": "2019-Datacenter",
            "version": "latest"
          },
          "osDisk": {
            "osType": "Windows",
            "createOption": "FromImage",
            "caching": "ReadWrite",
            "managedDisk": {
              "storageAccountType": "StandardSSD_LRS"
            },
            "diskSizeGB": 127
          }
        },
        "osProfile": {
          "computerName": "Jump-Srv",
          "adminUsername": "[parameters('adminUsername')]",
          "adminPassword": "[parameters('adminPassword')]",
          "windowsConfiguration": {
            "provisionVMAgent": true,
            "enableAutomaticUpdates": true
          },
          "allowExtensionOperations": true
        },
        "networkProfile": {
          "networkInterfaces": [
            {
              "id": "[resourceId('Microsoft.Network/networkInterfaces',
```

```
'netInterface-jump-srv')]"
            }
          ]
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkInterfaces', 'netInterface-
jump-srv')]"
      ]
    },
    {
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2022-03-01",
      "name": "Workload-Srv",
      "location": "[parameters('location')]",
      "properties": {
        "hardwareProfile": {
          "vmSize": "[parameters('vmSize')]"
        },
        "storageProfile": {
          "imageReference": {
            "publisher": "MicrosoftWindowsServer",
            "offer": "WindowsServer",
            "sku": "2019-Datacenter",
            "version": "latest"
          },
          "osDisk": {
            "osType": "Windows",
            "createOption": "FromImage",
            "caching": "ReadWrite",
            "managedDisk": {
              "storageAccountType": "StandardSSD_LRS"
            },
            "diskSizeGB": 127
          }
        },
        "osProfile": {
          "computerName": "Workload-Srv",
          "adminUsername": "[parameters('adminUsername')]",
          "adminPassword": "[parameters('adminPassword')]",
          "windowsConfiguration": {
            "provisionVMAgent": true,
            "enableAutomaticUpdates": true
          },
          "allowExtensionOperations": true
        },
        "networkProfile": {
          "networkInterfaces": [
            {
              "id": "[resourceId('Microsoft.Network/networkInterfaces',
'netInterface-workload-srv')]"
            }
          ]
        }
      },
```

```
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkInterfaces', 'netInterface-
workload-srv')]"
      ]
    },
    {
      "type": "Microsoft.Network/networkInterfaces",
      "apiVersion": "2021-08-01",
      "name": "netInterface-workload-srv",
      "location": "[parameters('location')]",
      "properties": {
        "ipConfigurations": [
          {
            "name": "ipconfig1",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "subnet": {
                "id": "
[resourceId('Microsoft.Network/virtualNetworks/subnets', 'Spoke-01',
'Workload-SN')]"
              },
              "primary": true,
              "privateIPAddressVersion": "IPv4"
            }
          }
        ],
        "enableAcceleratedNetworking": false,
        "enableIPForwarding": false,
        "networkSecurityGroup": {
          "id": "[resourceId('Microsoft.Network/networkSecurityGroups',
'nsg-workload-srv')]"
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkSecurityGroups', 'nsg-
workload-srv')]",
        "[resourceId('Microsoft.Network/virtualNetworks/subnets', 'Spoke-
01', 'Workload-SN')]"
      ]
    },
    {
      "type": "Microsoft.Network/networkInterfaces",
      "apiVersion": "2021-08-01",
      "name": "netInterface-jump-srv",
      "location": "[parameters('location')]",
      "properties": {
        "ipConfigurations": [
          {
            "name": "ipconfig1",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses',
'publicIP-jump-srv')]"
              },
```

```
              "subnet": {
                "id": "
[resourceId('Microsoft.Network/virtualNetworks/subnets', 'Spoke-01', 'Jump-
SN')]"
              },
              "primary": true,
              "privateIPAddressVersion": "IPv4"
            }
          }
        ],
        "enableAcceleratedNetworking": false,
        "enableIPForwarding": false,
        "networkSecurityGroup": {
          "id": "[resourceId('Microsoft.Network/networkSecurityGroups',
'nsg-jump-srv')]"
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkSecurityGroups', 'nsg-jump-
srv')]",
        "[resourceId('Microsoft.Network/publicIPAddresses', 'publicIP-jump-
srv')]",
        "[resourceId('Microsoft.Network/virtualNetworks/subnets', 'Spoke-
01', 'Jump-SN')]"
      ]
    },
    {
      "type": "Microsoft.Network/networkSecurityGroups",
      "apiVersion": "2021-08-01",
      "name": "nsg-jump-srv",
      "location": "[parameters('location')]",
      "properties": {
        "securityRules": [
          {
            "name": "RDP",
            "properties": {
              "protocol": "Tcp",
              "sourcePortRange": "*",
              "destinationPortRange": "3389",
              "sourceAddressPrefix": "*",
              "destinationAddressPrefix": "*",
              "access": "Allow",
              "priority": 300,
              "direction": "Inbound"
            }
          }
        ]
      }
    },
    {
      "type": "Microsoft.Network/networkSecurityGroups",
      "apiVersion": "2021-08-01",
      "name": "nsg-workload-srv",
      "location": "[parameters('location')]",
      "properties": {}
```

```
    },
    {
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2021-08-01",
      "name": "publicIP-jump-srv",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard"
      },
      "properties": {
        "publicIPAddressVersion": "IPv4",
        "publicIPAllocationMethod": "Static",
        "idleTimeoutInMinutes": 4
      }
    },
    {
      "type": "Microsoft.Network/routeTables",
      "apiVersion": "2021-08-01",
      "name": "RT-01",
      "location": "[parameters('location')]",
      "properties": {
        "disableBgpRoutePropagation": false,
        "routes": [
          {
            "name": "jump-to-inet",
            "properties": {
              "addressPrefix": "0.0.0.0/0",
              "nextHopType": "Internet"
            }
          }
        ]
      }
    },
    {
      "type": "Microsoft.Network/virtualHubs/hubRouteTables",
      "apiVersion": "2021-08-01",
      "name": "[format('{0}/{1}', 'Hub-01', 'RT_VNet')]",
      "properties": {
        "routes": [
          {
            "name": "Workload-SNToFirewall",
            "destinationType": "CIDR",
            "destinations": [
              "10.0.1.0/24"
            ],
            "nextHopType": "ResourceId",
            "nextHop": "[resourceId('Microsoft.Network/azureFirewalls',
'AzfwTest')]"
          },
          {
            "name": "InternetToFirewall",
            "destinationType": "CIDR",
            "destinations": [
              "0.0.0.0/0"
            ],
```

```
            "nextHopType": "ResourceId",
            "nextHop": "[resourceId('Microsoft.Network/azureFirewalls',
'AzfwTest')]"
          }
        ],
        "labels": [
          "VNet"
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/azureFirewalls', 'AzfwTest')]",
        "[resourceId('Microsoft.Network/virtualHubs', 'Hub-01')]"
      ]
    }
  ]
}
```

Multiple Azure resources are defined in the template:

- Microsoft.Network/virtualWans
- Microsoft.Network/virtualHubs
- Microsoft.Network/firewallPolicies
- Microsoft.Network/azureFirewalls
- Microsoft.Network/virtualNetworks
- Microsoft.Compute/virtualMachines
- Microsoft.Storage/storageAccounts
- Microsoft.Network/networkInterfaces
- Microsoft.Network/networkSecurityGroups
- Microsoft.Network/publicIPAddresses
- Microsoft.Network/routeTables

Deploy the template

Deploy the ARM template to Azure:

1. Select **Deploy to Azure** to sign in to Azure and open the template. The template creates an Azure Firewall, a virtual WAN and virtual hub, the network infrastructure, and two virtual machines.

   Deploy to Azure

2. In the portal, on the **Secured virtual hubs** page, type or select the following values:

   - Subscription: Select from existing subscriptions

- Resource group: Select from existing resource groups or select **Create new**, and select **OK**.
- Location: Select a location
- Admin Username: Type username for the administrator user account
- Admin Password: Type an administrator password or key

3. Select **Review + create** and then select **Create**. The deployment can take 10 minutes or longer to complete.

Validate the deployment

Now, test the firewall rules to confirm that it works as expected.

1. From the Azure portal, review the network settings for the **Workload-Srv** virtual machine and note the private IP address.

2. Connect a remote desktop to **Jump-Srv** virtual machine, and sign in. From there, open a remote desktop connection to the **Workload-Srv** private IP address.

3. Open Internet Explorer and browse to `www.microsoft.com`.

4. Select **OK** > **Close** on the Internet Explorer security alerts.

   You should see the Microsoft home page.

5. Browse to `www.google.com`.

   You should be blocked by the firewall.

So now you've verified that the firewall rules are working:

- You can browse to the one allowed FQDN, but not to any others.

Clean up resources

When you no longer need the resources that you created with the firewall, delete the resource group. This removes the firewall and all the related resources.

To delete the resource group, call the `Remove-AzResourceGroup` cmdlet:

Azure PowerShell

```
Remove-AzResourceGroup -Name "<your resource group name>"
```

Next steps

Learn about security partner providers

Quickstart: Create a Front Door using an ARM template

Article • 03/08/2023

This quickstart describes how to use an Azure Resource Manager template (ARM Template) to create a Front Door to set up high availability for a web endpoint.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

- If you don't have an Azure subscription, create a free account before you begin.
- IP or FQDN of a website or web application.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

In this quickstart, you'll create a Front Door configuration with a single backend and a single default path matching `/*`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "17606357911537037484"
    }
```

```
  },
  "parameters": {
    "frontDoorName": {
      "type": "string",
      "metadata": {
        "description": "The name of the frontdoor resource."
      }
    },
    "backendAddress": {
      "type": "string",
      "metadata": {
        "description": "The hostname of the backend. Must be an IP address
or FQDN."
      }
    }
  },
  "variables": {
    "frontEndEndpointName": "frontEndEndpoint",
    "loadBalancingSettingsName": "loadBalancingSettings",
    "healthProbeSettingsName": "healthProbeSettings",
    "routingRuleName": "routingRule",
    "backendPoolName": "backendPool"
  },
  "resources": [
    {
      "type": "Microsoft.Network/frontDoors",
      "apiVersion": "2020-05-01",
      "name": "[parameters('frontDoorName')]",
      "location": "global",
      "properties": {
        "enabledState": "Enabled",
        "frontendEndpoints": [
          {
            "name": "[variables('frontEndEndpointName')]",
            "properties": {
              "hostName": "[format('{0}.azurefd.net',
parameters('frontDoorName'))]",
              "sessionAffinityEnabledState": "Disabled"
            }
          }
        ],
        "loadBalancingSettings": [
          {
            "name": "[variables('loadBalancingSettingsName')]",
            "properties": {
              "sampleSize": 4,
              "successfulSamplesRequired": 2
            }
          }
        ],
        "healthProbeSettings": [
          {
            "name": "[variables('healthProbeSettingsName')]",
            "properties": {
              "path": "/",
```

```
              "protocol": "Http",
              "intervalInSeconds": 120
            }
          }
        ],
        "backendPools": [
          {
            "name": "[variables('backendPoolName')]",
            "properties": {
              "backends": [
                {
                  "address": "[parameters('backendAddress')]",
                  "backendHostHeader": "[parameters('backendAddress')]",
                  "httpPort": 80,
                  "httpsPort": 443,
                  "weight": 50,
                  "priority": 1,
                  "enabledState": "Enabled"
                }
              ],
              "loadBalancingSettings": {
                "id": "
[resourceId('Microsoft.Network/frontDoors/loadBalancingSettings',
parameters('frontDoorName'), variables('loadBalancingSettingsName'))]"
              },
              "healthProbeSettings": {
                "id": "
[resourceId('Microsoft.Network/frontDoors/healthProbeSettings',
parameters('frontDoorName'), variables('healthProbeSettingsName'))]"
              }
            }
          }
        ],
        "routingRules": [
          {
            "name": "[variables('routingRuleName')]",
            "properties": {
              "frontendEndpoints": [
                {
                  "id": "
[resourceId('Microsoft.Network/frontDoors/frontEndEndpoints',
parameters('frontDoorName'), variables('frontEndEndpointName'))]"
                }
              ],
              "acceptedProtocols": [
                "Http",
                "Https"
              ],
              "patternsToMatch": [
                "/*"
              ],
              "routeConfiguration": {
                "@odata.type":
"#Microsoft.Azure.FrontDoor.Models.FrontdoorForwardingConfiguration",
                "forwardingProtocol": "MatchRequest",
```

```
                "backendPool": {
                  "id": "
[resourceId('Microsoft.Network/frontDoors/backEndPools',
parameters('frontDoorName'), variables('backendPoolName'))]"
                }
              },
              "enabledState": "Enabled"
            }
          }
        ]
      }
    }
  ]
}
```

One Azure resource is defined in the template:

- Microsoft.Network/frontDoors

Deploy the template

1. Select **Try it** from the following code block to open Azure Cloud Shell, and then follow the instructions to sign in to Azure.

   Azure PowerShell

   ```
   $projectName = Read-Host -Prompt "Enter a project name that is used for
   generating resource names"
   $location = Read-Host -Prompt "Enter the location (i.e. centralus)"
   $templateUri = "https://raw.githubusercontent.com/Azure/azure-
   quickstart-templates/master/quickstarts/microsoft.network/front-door-
   create-basic/azuredeploy.json"

   $resourceGroupName = "${projectName}rg"

   New-AzResourceGroup -Name $resourceGroupName -Location "$location"
   New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName -
   TemplateUri $templateUri

   Read-Host -Prompt "Press [ENTER] to continue ..."
   ```

   Wait until you see the prompt from the console.

2. Select **Copy** from the previous code block to copy the PowerShell script.

3. Right-click the shell console pane and then select **Paste**.

4. Enter the values.

The template deployment creates a Front Door with a single backend. In this example `microsoft.com` is used as the **backendAddress**.

The resource group name is the project name with **rg** appended.

> ⓘ **Note**
>
> **frontDoorName** needs to be a globally unique name in order for the template to deploy successfully. If deployment fails, start over with Step 1.

It takes a few minutes to deploy the template. When completed, the output is similar to:

```
DeploymentName          : azuredeploy
ResourceGroupName       : devtestfrontdoorrg
ProvisioningState       : Succeeded
Timestamp               : 9/10/2020 10:30:29 PM
Mode                    : Incremental
TemplateLink            :
                          Uri            : https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101
                          -front-door-create-basic/azuredeploy.json
                          ContentVersion : 1.0.0.0

Parameters              :
                          Name               Type                       Value
                          ================   ========================   ==========
                          frontDoorName      String                     frontdoordevtest
                          backendAddress     String                     microsoft.com
```

Azure PowerShell is used to deploy the template. In addition to Azure PowerShell, you can also use the Azure portal, Azure CLI, and REST API. To learn other deployment methods, see Deploy templates.

Validate the deployment

1. Sign in to the Azure portal.
2. Select **Resource groups** from the left pane.
3. Select the resource group that you created in the previous section. The default resource group name is the project name with **rg** appended.
4. Select the Front Door you created previously and click on the **Frontend host** link. The link will open a web browser redirecting you to your backend FQDN you defined during creation.

Clean up resources

When you no longer need the Front Door service, delete the resource group. This removes the Front Door and all the related resources.

To delete the resource group, call the `Remove-AzResourceGroup` cmdlet:

Azure PowerShell

```
Remove-AzResourceGroup -Name <your resource group name>
```

Next steps

In this quickstart, you created a Front Door.

To learn how to add a custom domain to your Front Door, continue to the Front Door tutorials.

Front Door tutorials

Quickstart: Create a Front Door Standard/Premium using an ARM template

Article • 03/08/2023

This quickstart describes how to use an Azure Resource Manager template (ARM Template) to create an Azure Front Door Standard/Premium with a Web App as origin.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

- If you don't have an Azure subscription, create a free account before you begin.
- IP or FQDN of a website or web application.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

In this quickstart, you'll create a Front Door Standard/Premium, an App Service, and configure the App Service to validate that traffic has come through the Front Door origin.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
```

```
      "version": "0.5.6.12127",
      "templateHash": "1359495056007144414"
    }
  },
  "parameters": {
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "The location into which regionally scoped resources
should be deployed. Note that Front Door is a global resource."
      }
    },
    "appName": {
      "type": "string",
      "defaultValue": "[format('myapp-{0}',
uniqueString(resourceGroup().id))]",
      "metadata": {
        "description": "The name of the App Service application to create.
This must be globally unique."
      }
    },
    "appServicePlanSkuName": {
      "type": "string",
      "defaultValue": "S1",
      "metadata": {
        "description": "The name of the SKU to use when creating the App
Service plan."
      }
    },
    "appServicePlanCapacity": {
      "type": "int",
      "defaultValue": 1,
      "metadata": {
        "description": "The number of worker instances of your App Service
plan that should be provisioned."
      }
    },
    "frontDoorEndpointName": {
      "type": "string",
      "defaultValue": "[format('afd-{0}',
uniqueString(resourceGroup().id))]",
      "metadata": {
        "description": "The name of the Front Door endpoint to create. This
must be globally unique."
      }
    },
    "frontDoorSkuName": {
      "type": "string",
      "defaultValue": "Standard_AzureFrontDoor",
      "allowedValues": [
        "Standard_AzureFrontDoor",
        "Premium_AzureFrontDoor"
      ],
      "metadata": {
```

```
        "description": "The name of the SKU to use when creating the Front
Door profile."
      }
    }
  },
  "variables": {
    "appServicePlanName": "AppServicePlan",
    "frontDoorProfileName": "MyFrontDoor",
    "frontDoorOriginGroupName": "MyOriginGroup",
    "frontDoorOriginName": "MyAppServiceOrigin",
    "frontDoorRouteName": "MyRoute"
  },
  "resources": [
    {
      "type": "Microsoft.Cdn/profiles",
      "apiVersion": "2021-06-01",
      "name": "[variables('frontDoorProfileName')]",
      "location": "global",
      "sku": {
        "name": "[parameters('frontDoorSkuName')]"
      }
    },
    {
      "type": "Microsoft.Web/serverfarms",
      "apiVersion": "2020-06-01",
      "name": "[variables('appServicePlanName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('appServicePlanSkuName')]",
        "capacity": "[parameters('appServicePlanCapacity')]"
      },
      "kind": "app"
    },
    {
      "type": "Microsoft.Web/sites",
      "apiVersion": "2020-06-01",
      "name": "[parameters('appName')]",
      "location": "[parameters('location')]",
      "kind": "app",
      "identity": {
        "type": "SystemAssigned"
      },
      "properties": {
        "serverFarmId": "[resourceId('Microsoft.Web/serverfarms',
variables('appServicePlanName'))]",
        "httpsOnly": true,
        "siteConfig": {
          "detailedErrorLoggingEnabled": true,
          "httpLoggingEnabled": true,
          "requestTracingEnabled": true,
          "ftpsState": "Disabled",
          "minTlsVersion": "1.2",
          "ipSecurityRestrictions": [
            {
              "tag": "ServiceTag",
```

```
              "ipAddress": "AzureFrontDoor.Backend",
              "action": "Allow",
              "priority": 100,
              "headers": {
                "x-azure-fdid": [
                  "[reference(resourceId('Microsoft.Cdn/profiles',
variables('frontDoorProfileName'))).frontDoorId]"
                ]
              },
              "name": "Allow traffic from Front Door"
            }
          ]
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Web/serverfarms',
variables('appServicePlanName'))]",
        "[resourceId('Microsoft.Cdn/profiles',
variables('frontDoorProfileName'))]"
      ]
    },
    {
      "type": "Microsoft.Cdn/profiles/afdEndpoints",
      "apiVersion": "2021-06-01",
      "name": "[format('{0}/{1}', variables('frontDoorProfileName'),
parameters('frontDoorEndpointName'))]",
      "location": "global",
      "properties": {
        "enabledState": "Enabled"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Cdn/profiles',
variables('frontDoorProfileName'))]"
      ]
    },
    {
      "type": "Microsoft.Cdn/profiles/originGroups",
      "apiVersion": "2021-06-01",
      "name": "[format('{0}/{1}', variables('frontDoorProfileName'),
variables('frontDoorOriginGroupName'))]",
      "properties": {
        "loadBalancingSettings": {
          "sampleSize": 4,
          "successfulSamplesRequired": 3
        },
        "healthProbeSettings": {
          "probePath": "/",
          "probeRequestType": "HEAD",
          "probeProtocol": "Http",
          "probeIntervalInSeconds": 100
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Cdn/profiles',
variables('frontDoorProfileName'))]"
```

```
      ]
    },
    {
      "type": "Microsoft.Cdn/profiles/originGroups/origins",
      "apiVersion": "2021-06-01",
      "name": "[format('{0}/{1}/{2}', variables('frontDoorProfileName'),
variables('frontDoorOriginGroupName'), variables('frontDoorOriginName'))]",
      "properties": {
        "hostName": "[reference(resourceId('Microsoft.Web/sites',
parameters('appName'))).defaultHostName]",
        "httpPort": 80,
        "httpsPort": 443,
        "originHostHeader": "[reference(resourceId('Microsoft.Web/sites',
parameters('appName'))).defaultHostName]",
        "priority": 1,
        "weight": 1000
      },
      "dependsOn": [
        "[resourceId('Microsoft.Web/sites', parameters('appName'))]",
        "[resourceId('Microsoft.Cdn/profiles/originGroups',
variables('frontDoorProfileName'), variables('frontDoorOriginGroupName'))]"
      ]
    },
    {
      "type": "Microsoft.Cdn/profiles/afdEndpoints/routes",
      "apiVersion": "2021-06-01",
      "name": "[format('{0}/{1}/{2}', variables('frontDoorProfileName'),
parameters('frontDoorEndpointName'), variables('frontDoorRouteName'))]",
      "properties": {
        "originGroup": {
          "id": "[resourceId('Microsoft.Cdn/profiles/originGroups',
variables('frontDoorProfileName'), variables('frontDoorOriginGroupName'))]"
        },
        "supportedProtocols": [
          "Http",
          "Https"
        ],
        "patternsToMatch": [
          "/*"
        ],
        "forwardingProtocol": "HttpsOnly",
        "linkToDefaultDomain": "Enabled",
        "httpsRedirect": "Enabled"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Cdn/profiles/afdEndpoints',
variables('frontDoorProfileName'), parameters('frontDoorEndpointName'))]",
        "[resourceId('Microsoft.Cdn/profiles/originGroups/origins',
variables('frontDoorProfileName'), variables('frontDoorOriginGroupName'),
variables('frontDoorOriginName'))]",
        "[resourceId('Microsoft.Cdn/profiles/originGroups',
variables('frontDoorProfileName'), variables('frontDoorOriginGroupName'))]"
      ]
    }
  ],
```

```
  "outputs": {
    "appServiceHostName": {
      "type": "string",
      "value": "[reference(resourceId('Microsoft.Web/sites',
parameters('appName'))).defaultHostName]"
    },
    "frontDoorEndpointHostName": {
      "type": "string",
      "value": "[reference(resourceId('Microsoft.Cdn/profiles/afdEndpoints',
variables('frontDoorProfileName'),
parameters('frontDoorEndpointName'))).hostName]"
    }
  }
}
```

Multiple Azure resources are defined in the template:

- **Microsoft.Network/frontDoors**
- **Microsoft.Web/serverfarms** (App service plan to host web apps)
- **Microsoft.Web/sites** (Web app origin servicing request for Front Door)
-

Deploy the template

1. Select **Try it** from the following code block to open Azure Cloud Shell, and then follow the instructions to sign in to Azure.

> **Note**
>
> If you want to deploy Azure Front Door Premium instead of Standard substitute the value of the sku parameter with `Premium_AzureFrontDoor`. For detailed comparison, view **Azure Front Door tier comparison**.

Azure PowerShell

```
$projectName = Read-Host -Prompt "Enter a project name that is used for
generating resource names"
$location = Read-Host -Prompt "Enter the location (i.e. centralus)"
$templateUri = "https://raw.githubusercontent.com/Azure/azure-
quickstart-templates/master/quickstarts/microsoft.cdn/front-door-
standard-premium-app-service-public/azuredeploy.json"

$resourceGroupName = "${projectName}rg"

New-AzResourceGroup -Name $resourceGroupName -Location "$location"
New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName -
TemplateUri $templateUri -frontDoorSkuName Standard_AzureFrontDoor
```

```
Read-Host -Prompt "Press [ENTER] to continue ..."
```

Wait until you see the prompt from the console.

2. Select **Copy** from the previous code block to copy the PowerShell script.

3. Right-click the shell console pane and then select **Paste**.

4. Enter the values.

   The template deployment creates a Front Door with a web app as origin

   The resource group name is the project name with **rg** appended.

   > **Note**
   >
   > **frontDoorName** needs to be a globally unique name in order for the template to deploy successfully. If deployment fails, start over with Step 1.

   It takes a few minutes to deploy the template. When completed, the output is similar to:

```
DeploymentName          : azuredeploy
ResourceGroupName       : contosotestrg
ProvisioningState       : Succeeded
Timestamp               : 7/2/2022 7:42:06 AM
Mode                    : Incremental
TemplateLink            :
                          Uri            : https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.cdn/front-door-standa
                          rd-premium-app-service-public/azuredeploy.json
                          ContentVersion : 1.0.0.0

Parameters              :
                          Name                      Type                      Value
                          ======================    ========================  ==========
                          location                  String                    "centralus"
                          appName                   String                    "myapp-m3zjs3hppjaa6"
                          appServicePlanSkuName     String                    "S1"
                          appServicePlanCapacity    Int                       1
                          frontDoorEndpointName     String                    "afd-m3zjs3hppjaa6"
                          frontDoorSkuName          String                    "Standard_AzureFrontDoor"

Outputs                 :
                          Name                         Type                       Value
                          ==========================   ========================   ==========
                          appServiceHostName           String                     "myapp-m3zjs3hppjaa6.azurewebsites.net"
                          frontDoorEndpointHostName    String                     "afd-m3zjs3hppjaa6-cwfkd7d6ahcmfwhz.z01.azurefd.net"

DeploymentDebugLogLevel :
```

Azure PowerShell is used to deploy the template. In addition to Azure PowerShell, you can also use the Azure portal, Azure CLI, and REST API. To learn other deployment methods, see Deploy templates.

Validate the deployment

1. Sign in to the Azure portal.

2. Select **Resource groups** from the left pane.

3. Select the resource group that you created in the previous section. The default resource group name is the project name with **rg** appended.

4. Select the Front Door you created previously and you'll be able to see the endpoint hostname. Copy the hostname and paste it on to the address bar of a browser. Press enter and your request will automatically get routed to the web app.

Clean up resources

When you no longer need the Front Door service, delete the resource group. This will remove the Front Door and all the related resources.

To delete the resource group, call the `Remove-AzResourceGroup` cmdlet:

Azure PowerShell

```
Remove-AzResourceGroup -Name <your resource group name>
```

Next steps

In this quickstart, you created a:

- Front Door
- App Service plan

- Web App

To learn how to add a custom domain to your Front Door, continue to the Front Door tutorials.

Front Door tutorials

Quickstart: Create an internal load balancer to load balance VMs using an ARM template

Article • 05/01/2023

This quickstart describes how to use an Azure Resource Manager template (ARM template) to create an internal Azure load balancer.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template opens in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from the Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "12604669370283105560"
    }
  },
  "parameters": {
    "adminUsername": {
      "type": "string",
```

```
      "metadata": {
        "description": "Admin username"
      }
    },
    "adminPassword": {
      "type": "secureString",
      "metadata": {
        "description": "Admin password"
      }
    },
    "vmNamePrefix": {
      "type": "string",
      "defaultValue": "BackendVM",
      "metadata": {
        "description": "Prefix to use for VM names"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "vmSize": {
      "type": "string",
      "defaultValue": "Standard_D2s_v3",
      "metadata": {
        "description": "Size of the virtual machines"
      }
    }
  },
  "variables": {
    "availabilitySetName": "AvSet",
    "storageAccountType": "Standard_LRS",
    "storageAccountName": "[uniqueString(resourceGroup().id)]",
    "virtualNetworkName": "vNet",
    "subnetName": "backendSubnet",
    "loadBalancerName": "ilb",
    "networkInterfaceName": "nic",
    "subnetRef": "[resourceId('Microsoft.Network/virtualNetworks/subnets',
variables('virtualNetworkName'), variables('subnetName'))]",
    "numberOfInstances": 2
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-08-01",
      "name": "[variables('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[variables('storageAccountType')]"
      },
      "kind": "StorageV2"
    },
```

```
    {
      "type": "Microsoft.Compute/availabilitySets",
      "apiVersion": "2021-11-01",
      "name": "[variables('availabilitySetName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Aligned"
      },
      "properties": {
        "platformUpdateDomainCount": 2,
        "platformFaultDomainCount": 2
      }
    },
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "2021-05-01",
      "name": "[variables('virtualNetworkName')]",
      "location": "[parameters('location')]",
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "10.0.0.0/16"
          ]
        },
        "subnets": [
          {
            "name": "[variables('subnetName')]",
            "properties": {
              "addressPrefix": "10.0.2.0/24"
            }
          }
        ]
      }
    },
    {
      "copy": {
        "name": "networkInterface",
        "count": "[length(range(0, variables('numberOfInstances')))]"
      },
      "type": "Microsoft.Network/networkInterfaces",
      "apiVersion": "2021-05-01",
      "name": "[format('{0}{1}', variables('networkInterfaceName'), range(0,
variables('numberOfInstances'))[copyIndex()])]",
      "location": "[parameters('location')]",
      "properties": {
        "ipConfigurations": [
          {
            "name": "ipconfig1",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "subnet": {
                "id": "[variables('subnetRef')]"
              },
              "loadBalancerBackendAddressPools": [
                {
```

```
                  "id": "
[resourceId('Microsoft.Network/loadBalancers/backendAddressPools',
variables('loadBalancerName'), 'BackendPool1')]"
                }
              ]
            }
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/loadBalancers',
variables('loadBalancerName'))]",
        "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/loadBalancers",
      "apiVersion": "2021-05-01",
      "name": "[variables('loadBalancerName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard"
      },
      "properties": {
        "frontendIPConfigurations": [
          {
            "properties": {
              "subnet": {
                "id": "[variables('subnetRef')]"
              },
              "privateIPAddress": "10.0.2.6",
              "privateIPAllocationMethod": "Static"
            },
            "name": "LoadBalancerFrontend"
          }
        ],
        "backendAddressPools": [
          {
            "name": "BackendPool1"
          }
        ],
        "loadBalancingRules": [
          {
            "properties": {
              "frontendIPConfiguration": {
                "id": "
[resourceId('Microsoft.Network/loadBalancers/frontendIpConfigurations',
variables('loadBalancerName'), 'LoadBalancerFrontend')]"
              },
              "backendAddressPool": {
                "id": "
[resourceId('Microsoft.Network/loadBalancers/backendAddressPools',
variables('loadBalancerName'), 'BackendPool1')]"
              },
```

```
              "probe": {
                "id": "[resourceId('Microsoft.Network/loadBalancers/probes',
variables('loadBalancerName'), 'lbprobe')]"
              },
              "protocol": "Tcp",
              "frontendPort": 80,
              "backendPort": 80,
              "idleTimeoutInMinutes": 15
            },
            "name": "lbrule"
          }
        ],
        "probes": [
          {
            "properties": {
              "protocol": "Tcp",
              "port": 80,
              "intervalInSeconds": 15,
              "numberOfProbes": 2
            },
            "name": "lbprobe"
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]"
      ]
    },
    {
      "copy": {
        "name": "vm",
        "count": "[length(range(0, variables('numberOfInstances')))]"
      },
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2021-11-01",
      "name": "[format('{0}{1}', parameters('vmNamePrefix'), range(0,
variables('numberOfInstances'))[copyIndex()])]",
      "location": "[parameters('location')]",
      "properties": {
        "availabilitySet": {
          "id": "[resourceId('Microsoft.Compute/availabilitySets',
variables('availabilitySetName'))]"
        },
        "hardwareProfile": {
          "vmSize": "[parameters('vmSize')]"
        },
        "osProfile": {
          "computerName": "[format('{0}{1}', parameters('vmNamePrefix'),
range(0, variables('numberOfInstances'))[copyIndex()])]",
          "adminUsername": "[parameters('adminUsername')]",
          "adminPassword": "[parameters('adminPassword')]"
        },
        "storageProfile": {
          "imageReference": {
```

```
            "publisher": "MicrosoftWindowsServer",
            "offer": "WindowsServer",
            "sku": "2019-Datacenter",
            "version": "latest"
          },
          "osDisk": {
            "createOption": "FromImage"
          }
        },
        "networkProfile": {
          "networkInterfaces": [
            {
              "id": "[resourceId('Microsoft.Network/networkInterfaces',
format('{0}{1}', variables('networkInterfaceName'), range(0,
variables('numberOfInstances'))[range(0, variables('numberOfInstances'))
[copyIndex()]]))]"
            }
          ]
        },
        "diagnosticsProfile": {
          "bootDiagnostics": {
            "enabled": true,
            "storageUri": "
[reference(resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))).primaryEndpoints.blob]"
          }
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Compute/availabilitySets',
variables('availabilitySetName'))]",
        "[resourceId('Microsoft.Network/networkInterfaces', format('{0}{1}',
variables('networkInterfaceName'), range(0, variables('numberOfInstances'))
[range(0, variables('numberOfInstances'))[copyIndex()]]))]",
        "[resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))]"
      ]
    }
  ]
}
```

Multiple Azure resources have been defined in the template:

- **Microsoft.Storage/storageAccounts**: Virtual machine storage accounts for boot diagnostics.
- **Microsoft.Compute/availabilitySets**: Availability set for virtual machines.
- **Microsoft.Network/virtualNetworks**: Virtual network for load balancer and virtual machines.
- **Microsoft.Network/networkInterfaces**: Network interfaces for virtual machines.
- **Microsoft.Network/loadBalancers**: Internal load balancer.
- **Microsoft.Compute/virtualMachines**: Virtual machines.

To find more templates that are related to Azure Load Balancer, see Azure Quickstart Templates.

Deploy the template

Azure CLI

Azure CLI

```
read -p "Enter the location (i.e. westcentralus): " location
resourceGroupName="myResourceGroupLB"
templateUri="https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.compute/2-vms-internal-load-balancer/azuredeploy.json"

az group create \
--name $resourceGroupName \
--location $location

az deployment group create \
--resource-group $resourceGroupName \
--template-uri  $templateUri
```

Review deployed resources

1. Sign in to the Azure portal.
2. Select **Resource groups** from the left pane.
3. Select the resource group that you created in the previous section. The default resource group name is **myResourceGroupLB**
4. Verify the following resources were created in the resource group:

Clean up resources

When no longer needed, you can use the az group delete command to remove the resource group and all resources contained within.

Azure CLI

```
az group delete \
  --name myResourceGroupLB
```

Next steps

For a step-by-step tutorial that guides you through the process of creating a template, see:

Tutorial: Create and deploy your first ARM template

Quickstart: Create a public load balancer to load balance VMs using an ARM template

Article • 04/12/2023

Load balancing provides a higher level of availability and scale by spreading incoming requests across multiple virtual machines (VMs).

This quickstart shows you how to deploy a standard load balancer to load balance virtual machines.

Using an ARM template takes fewer steps comparing to other deployment methods.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

Load balancer and public IP SKUs must match. When you create a standard load balancer, you must also create a new standard public IP address that is configured as the frontend for the standard load balancer. If you want to create a basic load balancer, use this template. Microsoft recommends using standard SKU for production workloads.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.9.1.41621",
      "templateHash": "1833787209941575306"
    }
  },
  "parameters": {
    "projectName": {
      "type": "string",
      "metadata": {
        "description": "Specifies a project name that is used for generating
resource names."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Specifies the location for all of the resources
created by this template."
      }
    },
    "adminUsername": {
      "type": "string",
      "metadata": {
        "description": "Specifies the virtual machine administrator
username."
      }
    },
    "adminPassword": {
      "type": "secureString",
      "metadata": {
        "description": "Specifies the virtual machine administrator
password."
      }
    },
    "vmSize": {
      "type": "string",
      "defaultValue": "Standard_D2s_v3",
      "metadata": {
        "description": "Size of the virtual machine"
      }
    }
  },
  "variables": {
    "lbName": "[format('{0}-lb', parameters('projectName'))]",
    "lbSkuName": "Standard",
    "lbPublicIpAddressName": "[format('{0}-lbPublicIP',
parameters('projectName'))]",
```

```
    "lbPublicIPAddressNameOutbound": "[format('{0}-lbPublicIPOutbound',
parameters('projectName'))]",
    "lbFrontEndName": "LoadBalancerFrontEnd",
    "lbFrontEndNameOutbound": "LoadBalancerFrontEndOutbound",
    "lbBackendPoolName": "LoadBalancerBackEndPool",
    "lbBackendPoolNameOutbound": "LoadBalancerBackEndPoolOutbound",
    "lbProbeName": "loadBalancerHealthProbe",
    "nsgName": "[format('{0}-nsg', parameters('projectName'))]",
    "vNetName": "[format('{0}-vnet', parameters('projectName'))]",
    "vNetAddressPrefix": "10.0.0.0/16",
    "vNetSubnetName": "BackendSubnet",
    "vNetSubnetAddressPrefix": "10.0.0.0/24",
    "bastionName": "[format('{0}-bastion', parameters('projectName'))]",
    "bastionSubnetName": "AzureBastionSubnet",
    "vNetBastionSubnetAddressPrefix": "10.0.1.0/24",
    "bastionPublicIPAddressName": "[format('{0}-bastionPublicIP',
parameters('projectName'))]",
    "vmStorageAccountType": "Premium_LRS"
  },
  "resources": [
    {
      "copy": {
        "name": "project_vm_1_networkInterface",
        "count": "[length(range(0, 3))]"
      },
      "type": "Microsoft.Network/networkInterfaces",
      "apiVersion": "2021-08-01",
      "name": "[format('{0}-vm{1}-networkInterface',
parameters('projectName'), add(range(0, 3)[copyIndex()], 1))]",
      "location": "[parameters('location')]",
      "properties": {
        "ipConfigurations": [
          {
            "name": "ipconfig1",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "subnet": {
                "id": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
variables('vNetName'), variables('vNetSubnetName'))]"
              },
              "loadBalancerBackendAddressPools": [
                {
                  "id": "
[resourceId('Microsoft.Network/loadBalancers/backendAddressPools',
variables('lbName'), variables('lbBackendPoolName'))]"
                },
                {
                  "id": "
[resourceId('Microsoft.Network/loadBalancers/backendAddressPools',
variables('lbName'), variables('lbBackendPoolNameOutbound'))]"
                }
              ]
            }
          }
```

```
        ],
        "networkSecurityGroup": {
          "id": "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('nsgName'))]"
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/loadBalancers',
variables('lbName'))]",
        "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('nsgName'))]",
        "[resourceId('Microsoft.Network/virtualNetworks',
variables('vNetName'))]",
        "[resourceId('Microsoft.Network/virtualNetworks/subnets',
variables('vNetName'), variables('vNetSubnetName'))]"
      ]
    },
    {
      "copy": {
        "name": "project_vm_1_InstallWebServer",
        "count": "[length(range(0, 3))]"
      },
      "type": "Microsoft.Compute/virtualMachines/extensions",
      "apiVersion": "2021-11-01",
      "name": "[format('{0}-vm{1}/InstallWebServer',
parameters('projectName'), add(range(0, 3)[copyIndex()], 1))]",
      "location": "[parameters('location')]",
      "properties": {
        "publisher": "Microsoft.Compute",
        "type": "CustomScriptExtension",
        "typeHandlerVersion": "1.7",
        "autoUpgradeMinorVersion": true,
        "settings": {
          "commandToExecute": "powershell.exe Install-WindowsFeature -name
Web-Server -IncludeManagementTools && powershell.exe remove-item
'C:\\inetpub\\wwwroot\\iisstart.htm' && powershell.exe Add-Content -Path
'C:\\inetpub\\wwwroot\\iisstart.htm' -Value $('Hello World from ' +
$env:computername)"
        }
      },
      "dependsOn": [
        "project_vm_1"
      ]
    },
    {
      "copy": {
        "name": "project_vm_1",
        "count": "[length(range(1, 3))]"
      },
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2021-11-01",
      "name": "[format('{0}-vm{1}', parameters('projectName'), range(1, 3)
[copyIndex()])]",
      "location": "[parameters('location')]",
      "zones": [
```

```
        "[string(range(1, 3)[copyIndex()])]"
      ],
      "properties": {
        "hardwareProfile": {
          "vmSize": "[parameters('vmSize')]"
        },
        "storageProfile": {
          "imageReference": {
            "publisher": "MicrosoftWindowsServer",
            "offer": "WindowsServer",
            "sku": "2019-Datacenter",
            "version": "latest"
          },
          "osDisk": {
            "createOption": "FromImage",
            "managedDisk": {
              "storageAccountType": "[variables('vmStorageAccountType')]"
            }
          }
        },
        "networkProfile": {
          "networkInterfaces": [
            {
              "id": "[resourceId('Microsoft.Network/networkInterfaces',
format('{0}-vm{1}-networkInterface', parameters('projectName'), range(1, 3)
[copyIndex()]))]"
            }
          ]
        },
        "osProfile": {
          "computerName": "[format('{0}-vm{1}', parameters('projectName'),
range(1, 3)[copyIndex()])]",
          "adminUsername": "[parameters('adminUsername')]",
          "adminPassword": "[parameters('adminPassword')]",
          "windowsConfiguration": {
            "enableAutomaticUpdates": true,
            "provisionVMAgent": true
          }
        }
      },
      "dependsOn": [
        "project_vm_1_networkInterface"
      ]
    },
    {
      "type": "Microsoft.Network/virtualNetworks/subnets",
      "apiVersion": "2021-08-01",
      "name": "[format('{0}/{1}', variables('vNetName'),
variables('bastionSubnetName'))]",
      "properties": {
        "addressPrefix": "[variables('vNetBastionSubnetAddressPrefix')]"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/virtualNetworks',
variables('vNetName'))]",
```

```
        "[resourceId('Microsoft.Network/virtualNetworks/subnets',
variables('vNetName'), variables('vNetSubnetName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/virtualNetworks/subnets",
      "apiVersion": "2021-08-01",
      "name": "[format('{0}/{1}', variables('vNetName'),
variables('vNetSubnetName'))]",
      "properties": {
        "addressPrefix": "[variables('vNetSubnetAddressPrefix')]"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/virtualNetworks',
variables('vNetName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/bastionHosts",
      "apiVersion": "2021-08-01",
      "name": "[variables('bastionName')]",
      "location": "[parameters('location')]",
      "properties": {
        "ipConfigurations": [
          {
            "name": "IpConf",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses',
variables('bastionPublicIPAddressName'))]"
              },
              "subnet": {
                "id": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
variables('vNetName'), variables('bastionSubnetName'))]"
              }
            }
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/publicIPAddresses',
variables('bastionPublicIPAddressName'))]",
        "[resourceId('Microsoft.Network/virtualNetworks/subnets',
variables('vNetName'), variables('bastionSubnetName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2021-08-01",
      "name": "[variables('bastionPublicIPAddressName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[variables('lbSkuName')]"
```

```
      },
      "properties": {
        "publicIPAddressVersion": "IPv4",
        "publicIPAllocationMethod": "Static"
      }
    },
    {
      "type": "Microsoft.Network/loadBalancers",
      "apiVersion": "2021-08-01",
      "name": "[variables('lbName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[variables('lbSkuName')]"
      },
      "properties": {
        "frontendIPConfigurations": [
          {
            "name": "[variables('lbFrontEndName')]",
            "properties": {
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses',
variables('lbPublicIpAddressName'))]"
              }
            }
          },
          {
            "name": "[variables('lbFrontEndNameOutbound')]",
            "properties": {
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses',
variables('lbPublicIPAddressNameOutbound'))]"
              }
            }
          }
        ],
        "backendAddressPools": [
          {
            "name": "[variables('lbBackendPoolName')]"
          },
          {
            "name": "[variables('lbBackendPoolNameOutbound')]"
          }
        ],
        "loadBalancingRules": [
          {
            "name": "myHTTPRule",
            "properties": {
              "frontendIPConfiguration": {
                "id": "
[resourceId('Microsoft.Network/loadBalancers/frontendIPConfigurations',
variables('lbName'), variables('lbFrontEndName'))]"
              },
              "backendAddressPool": {
                "id": "
[resourceId('Microsoft.Network/loadBalancers/backendAddressPools',
```

```
variables('lbName'), variables('lbBackendPoolName'))]"
              },
              "frontendPort": 80,
              "backendPort": 80,
              "enableFloatingIP": false,
              "idleTimeoutInMinutes": 15,
              "protocol": "Tcp",
              "enableTcpReset": true,
              "loadDistribution": "Default",
              "disableOutboundSnat": true,
              "probe": {
                "id": "[resourceId('Microsoft.Network/loadBalancers/probes',
variables('lbName'), variables('lbProbeName'))]"
              }
            }
          }
        ],
        "probes": [
          {
            "name": "[variables('lbProbeName')]",
            "properties": {
              "protocol": "Tcp",
              "port": 80,
              "intervalInSeconds": 5,
              "numberOfProbes": 2
            }
          }
        ],
        "outboundRules": [
          {
            "name": "myOutboundRule",
            "properties": {
              "allocatedOutboundPorts": 10000,
              "protocol": "All",
              "enableTcpReset": false,
              "idleTimeoutInMinutes": 15,
              "backendAddressPool": {
                "id": "
[resourceId('Microsoft.Network/loadBalancers/backendAddressPools',
variables('lbName'), variables('lbBackendPoolNameOutbound'))]"
              },
              "frontendIPConfigurations": [
                {
                  "id": "
[resourceId('Microsoft.Network/loadBalancers/frontendIPConfigurations',
variables('lbName'), variables('lbFrontEndNameOutbound'))]"
                }
              ]
            }
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/publicIPAddresses',
variables('lbPublicIpAddressName'))]",
```

```
        "[resourceId('Microsoft.Network/publicIPAddresses',
variables('lbPublicIPAddressNameOutbound'))]"
      ]
    },
    {
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2021-08-01",
      "name": "[variables('lbPublicIpAddressName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[variables('lbSkuName')]"
      },
      "properties": {
        "publicIPAddressVersion": "IPv4",
        "publicIPAllocationMethod": "Static"
      }
    },
    {
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2021-08-01",
      "name": "[variables('lbPublicIPAddressNameOutbound')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[variables('lbSkuName')]"
      },
      "properties": {
        "publicIPAddressVersion": "IPv4",
        "publicIPAllocationMethod": "Static"
      }
    },
    {
      "type": "Microsoft.Network/networkSecurityGroups",
      "apiVersion": "2021-08-01",
      "name": "[variables('nsgName')]",
      "location": "[parameters('location')]",
      "properties": {
        "securityRules": [
          {
            "name": "AllowHTTPInbound",
            "properties": {
              "protocol": "*",
              "sourcePortRange": "*",
              "destinationPortRange": "80",
              "sourceAddressPrefix": "Internet",
              "destinationAddressPrefix": "*",
              "access": "Allow",
              "priority": 100,
              "direction": "Inbound"
            }
          }
        ]
      }
    },
    {
      "type": "Microsoft.Network/virtualNetworks",
```

```
      "apiVersion": "2021-08-01",
      "name": "[variables('vNetName')]",
      "location": "[parameters('location')]",
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[variables('vNetAddressPrefix')]"
          ]
        }
      }
    }
  ]
}
```

Multiple Azure resources have been defined in the template:

- **Microsoft.Network/loadBalancers**
- **Microsoft.Network/publicIPAddresses**: for the load balancer, bastion host, and for each of the three virtual machines.
- **Microsoft.Network/bastionHosts**
- **Microsoft.Network/networkSecurityGroups**
- **Microsoft.Network/virtualNetworks**
- **Microsoft.Compute/virtualMachines** (3).
- **Microsoft.Network/networkInterfaces** (3).
- **Microsoft.Compute/virtualMachine/extensions** (3): use to configure the Internet Information Server (IIS), and the web pages.

To find more templates that are related to Azure Load Balancer, see Azure Quickstart Templates.

Deploy the template

1. Select **Try it** from the following code block to open Azure Cloud Shell, and then follow the instructions to sign in to Azure.

   Azure PowerShell

   ```
   $projectName = Read-Host -Prompt "Enter a project name with 12 or less
   letters or numbers that is used to generate Azure resource names"
   $location = Read-Host -Prompt "Enter the location (i.e. centralus)"
   $adminUserName = Read-Host -Prompt "Enter the virtual machine
   administrator account name"
   $adminPassword = Read-Host -Prompt "Enter the virtual machine
   administrator password" -AsSecureString

   $resourceGroupName = "${projectName}rg"
   $templateUri = "https://raw.githubusercontent.com/Azure/azure-
   ```

```
quickstart-templates/master/quickstarts/microsoft.network/load-
balancer-standard-create/azuredeploy.json"

New-AzResourceGroup -Name $resourceGroupName -Location $location
New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName -
TemplateUri $templateUri -projectName $projectName -location $location
-adminUsername $adminUsername -adminPassword $adminPassword

Write-Host "Press [ENTER] to continue."
```

Wait until you see the prompt from the console.

2. Select **Copy** from the previous code block to copy the PowerShell script.

3. Right-click the shell console pane and then select **Paste**.

4. Enter the values.

   The template deployment creates three availability zones. Availability zones are supported only in certain regions. Use one of the supported regions. If you aren't sure, enter **centralus**.

   The resource group name is the project name with `rg` appended. You need the resource group name in the next section.

It takes about 10 minutes to deploy the template. When completed, the output is similar to:

```
DeploymentName          : azuredeploy
ResourceGroupName       : myslb0919rg
ProvisioningState       : Succeeded
Timestamp               : 9/19/19 2:17:47 PM
Mode                    : Incremental
TemplateLink            :
                          Uri            : https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master
                          /101-load-balancer-standard-create/azuredeploy.json
                          ContentVersion : 1.0.0.0

Parameters              :
                          Name             Type                       Value
                          ===============  =========================  ==========
                          projectName      String                     myslb0919
                          location         String                     centralus
                          adminUsername    String                     jgao
                          adminPassword    SecureString
```

Azure PowerShell is used to deploy the template. You can also use the Azure portal, Azure CLI, and REST API. To learn other deployment methods, see Deploy templates.

Review deployed resources

1. Sign in to the Azure portal.

2. Select **Resource groups** from the left pane.

3. Select the resource group that you created in the previous section. The default resource group name is the project name with **-rg** appended.

4. Select the load balancer. Its default name is the project name with **-lb** appended.

5. Copy only the IP address part of the public IP address, and then paste it into the address bar of your browser.

The browser displays the default page of the Internet Information Services (IIS) web server.

To see the load balancer distribute traffic across all three VMs, you can force a refresh of your web browser from the client machine.

Clean up resources

When you no longer need them, delete the:

- Resource group
- Load balancer
- Related resources

Go to the Azure portal, select the resource group that contains the load balancer, and then select **Delete resource group**.

Next steps

In this quickstart, you:

- Created a virtual network for the load balancer and virtual machines.
- Created an Azure Bastion host for management.
- Created a standard load balancer and attached VMs to it.
- Configured the load-balancer traffic rule, and the health probe.
- Tested the load balancer.

To learn more, continue to the tutorials for Azure Load Balancer.

Azure Load Balancer tutorials

Quickstart: Create a NAT gateway - ARM template

Article • 05/01/2023

Get started with Azure NAT Gateway by using an Azure Resource Manager template (ARM template). This template deploys a virtual network, a NAT gateway resource, and Ubuntu virtual machine. The Ubuntu virtual machine is deployed to a subnet that is associated with the NAT gateway resource.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template opens in the Azure portal.

Deploy to Azure

Prerequisites

Portal

- If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

This template is configured to create a:

- Virtual network
- NAT gateway resource
- Ubuntu virtual machine

The Ubuntu VM is deployed to a subnet that's associated with the NAT gateway resource.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "1908625783801438696"
    }
  },
  "parameters": {
    "vmname": {
      "type": "string",
      "defaultValue": "myVM",
      "metadata": {
        "description": "Name of the virtual machine"
      }
    },
    "vmsize": {
      "type": "string",
      "defaultValue": "Standard_D2s_v3",
      "metadata": {
        "description": "Size of the virtual machine"
      }
    },
    "vnetname": {
      "type": "string",
      "defaultValue": "myVnet",
      "metadata": {
        "description": "Name of the virtual network"
      }
    },
    "subnetname": {
      "type": "string",
      "defaultValue": "mySubnet",
      "metadata": {
        "description": "Name of the subnet for virtual network"
      }
    },
    "vnetaddressspace": {
      "type": "string",
      "defaultValue": "192.168.0.0/16",
      "metadata": {
        "description": "Address space for virtual network"
      }
    },
    "vnetsubnetprefix": {
      "type": "string",
```

```
        "defaultValue": "192.168.0.0/24",
        "metadata": {
          "description": "Subnet prefix for virtual network"
        }
      },
      "natgatewayname": {
        "type": "string",
        "defaultValue": "myNATgateway",
        "metadata": {
          "description": "Name of the NAT gateway"
        }
      },
      "networkinterfacename": {
        "type": "string",
        "defaultValue": "myvmNIC",
        "metadata": {
          "description": "Name of the virtual machine nic"
        }
      },
      "publicipname": {
        "type": "string",
        "defaultValue": "myPublicIP",
        "metadata": {
          "description": "Name of the NAT gateway public IP"
        }
      },
      "nsgname": {
        "type": "string",
        "defaultValue": "myVMnsg",
        "metadata": {
          "description": "Name of the virtual machine NSG"
        }
      },
      "publicipvmname": {
        "type": "string",
        "defaultValue": "myPublicIPVM",
        "metadata": {
          "description": "Name of the virtual machine public IP"
        }
      },
      "publicipprefixname": {
        "type": "string",
        "defaultValue": "myPublicIPPrefix",
        "metadata": {
          "description": "Name of the NAT gateway public IP"
        }
      },
      "adminusername": {
        "type": "string",
        "metadata": {
          "description": "Administrator username for virtual machine"
        }
      },
      "adminpassword": {
        "type": "secureString",
```

```
      "metadata": {
        "description": "Administrator password for virtual machine"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Name of resource group"
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Network/networkSecurityGroups",
      "apiVersion": "2021-05-01",
      "name": "[parameters('nsgname')]",
      "location": "[parameters('location')]",
      "properties": {
        "securityRules": [
          {
            "name": "SSH",
            "properties": {
              "protocol": "Tcp",
              "sourcePortRange": "*",
              "destinationPortRange": "22",
              "sourceAddressPrefix": "*",
              "destinationAddressPrefix": "*",
              "access": "Allow",
              "priority": 300,
              "direction": "Inbound"
            }
          }
        ]
      }
    },
    {
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2021-05-01",
      "name": "[parameters('publicipname')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard"
      },
      "properties": {
        "publicIPAddressVersion": "IPv4",
        "publicIPAllocationMethod": "Static",
        "idleTimeoutInMinutes": 4
      }
    },
    {
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2021-05-01",
      "name": "[parameters('publicipvmname')]",
      "location": "[parameters('location')]",
```

```
      "sku": {
        "name": "Standard"
      },
      "properties": {
        "publicIPAddressVersion": "IPv4",
        "publicIPAllocationMethod": "Static",
        "idleTimeoutInMinutes": 4
      }
    },
    {
      "type": "Microsoft.Network/publicIPPrefixes",
      "apiVersion": "2021-05-01",
      "name": "[parameters('publicipprefixname')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard"
      },
      "properties": {
        "prefixLength": 31,
        "publicIPAddressVersion": "IPv4"
      }
    },
    {
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2021-11-01",
      "name": "[parameters('vmname')]",
      "location": "[parameters('location')]",
      "properties": {
        "hardwareProfile": {
          "vmSize": "[parameters('vmsize')]"
        },
        "storageProfile": {
          "imageReference": {
            "publisher": "Canonical",
            "offer": "UbuntuServer",
            "sku": "18.04-LTS",
            "version": "latest"
          },
          "osDisk": {
            "osType": "Linux",
            "name": "[format('{0}_disk1', parameters('vmname'))]",
            "createOption": "FromImage",
            "caching": "ReadWrite",
            "managedDisk": {
              "storageAccountType": "Premium_LRS"
            },
            "diskSizeGB": 30
          }
        },
        "osProfile": {
          "computerName": "[parameters('vmname')]",
          "adminUsername": "[parameters('adminusername')]",
          "adminPassword": "[parameters('adminpassword')]",
          "linuxConfiguration": {
            "disablePasswordAuthentication": false,
```

```
            "provisionVMAgent": true
          },
          "allowExtensionOperations": true
        },
        "networkProfile": {
          "networkInterfaces": [
            {
              "id": "[resourceId('Microsoft.Network/networkInterfaces',
parameters('networkinterfacename'))]"
            }
          ]
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkInterfaces',
parameters('networkinterfacename'))]"
      ]
    },
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "2021-05-01",
      "name": "[parameters('vnetname')]",
      "location": "[parameters('location')]",
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[parameters('vnetaddressspace')]"
          ]
        },
        "subnets": [
          {
            "name": "[parameters('subnetname')]",
            "properties": {
              "addressPrefix": "[parameters('vnetsubnetprefix')]",
              "natGateway": {
                "id": "[resourceId('Microsoft.Network/natGateways',
parameters('natgatewayname'))]"
              },
              "privateEndpointNetworkPolicies": "Enabled",
              "privateLinkServiceNetworkPolicies": "Enabled"
            }
          }
        ],
        "enableDdosProtection": false,
        "enableVmProtection": false
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/natGateways',
parameters('natgatewayname'))]"
      ]
    },
    {
      "type": "Microsoft.Network/natGateways",
      "apiVersion": "2021-05-01",
      "name": "[parameters('natgatewayname')]",
```

```
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard"
      },
      "properties": {
        "idleTimeoutInMinutes": 4,
        "publicIpAddresses": [
          {
            "id": "[resourceId('Microsoft.Network/publicIPAddresses',
parameters('publicipname'))]"
          }
        ],
        "publicIpPrefixes": [
          {
            "id": "[resourceId('Microsoft.Network/publicIPPrefixes',
parameters('publicipprefixname'))]"
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/publicIPAddresses',
parameters('publicipname'))]",
        "[resourceId('Microsoft.Network/publicIPPrefixes',
parameters('publicipprefixname'))]"
      ]
    },
    {
      "type": "Microsoft.Network/virtualNetworks/subnets",
      "apiVersion": "2021-05-01",
      "name": "[format('{0}/{1}', parameters('vnetname'), 'mySubnet')]",
      "properties": {
        "addressPrefix": "[parameters('vnetsubnetprefix')]",
        "natGateway": {
          "id": "[resourceId('Microsoft.Network/natGateways',
parameters('natgatewayname'))]"
        },
        "privateEndpointNetworkPolicies": "Enabled",
        "privateLinkServiceNetworkPolicies": "Enabled"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/natGateways',
parameters('natgatewayname'))]",
        "[resourceId('Microsoft.Network/virtualNetworks',
parameters('vnetname'))]"
      ]
    },
    {
      "type": "Microsoft.Network/networkInterfaces",
      "apiVersion": "2021-05-01",
      "name": "[parameters('networkinterfacename')]",
      "location": "[parameters('location')]",
      "properties": {
        "ipConfigurations": [
          {
            "name": "ipconfig1",
```

```
            "properties": {
              "privateIPAddress": "192.168.0.4",
              "privateIPAllocationMethod": "Dynamic",
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses',
parameters('publicipvmname'))]"
              },
              "subnet": {
                "id": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('vnetname'), 'mySubnet')]"
              },
              "primary": true,
              "privateIPAddressVersion": "IPv4"
            }
          }
        ],
        "enableAcceleratedNetworking": false,
        "enableIPForwarding": false,
        "networkSecurityGroup": {
          "id": "[resourceId('Microsoft.Network/networkSecurityGroups',
parameters('nsgname'))]"
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('vnetname'), 'mySubnet')]",
        "[resourceId('Microsoft.Network/networkSecurityGroups',
parameters('nsgname'))]",
        "[resourceId('Microsoft.Network/publicIPAddresses',
parameters('publicipvmname'))]"
      ]
    }
  ]
}
```

Nine Azure resources are defined in the template:

- **Microsoft.Network/networkSecurityGroups**: Creates a network security group.
- **Microsoft.Network/networkSecurityGroups/securityRules**: Creates a security rule.
- **Microsoft.Network/publicIPAddresses**: Creates a public IP address.
- **Microsoft.Network/publicIPPrefixes**: Creates a public IP prefix.
- **Microsoft.Compute/virtualMachines**: Creates a virtual machine.
- **Microsoft.Network/virtualNetworks**: Creates a virtual network.
- **Microsoft.Network/natGateways**: Creates a NAT gateway resource.

- Microsoft.Network/virtualNetworks/subnets: Creates a virtual network subnet.
- Microsoft.Network/networkinterfaces: Creates a network interface.

Deploy the template

Portal

Deploy to Azure

Review deployed resources

1. Sign in to the Azure portal.
2. Select **Resource groups** from the left pane.
3. Select the resource group that you created in the previous section. The default resource group name is **myResourceGroupNAT**
4. Verify the following resources were created in the resource group:

Clean up resources

Portal

When no longer needed, delete the resource group, NAT gateway, and all related resources. Select the resource group **myResourceGroupNAT** that contains the NAT gateway, and then select **Delete**.

Next steps

In this quickstart, you created a:

- NAT gateway resource
- Virtual network
- Ubuntu virtual machine

The virtual machine is deployed to a virtual network subnet associated with the NAT gateway.

To learn more about Azure NAT Gateway and Azure Resource Manager, continue to the following articles.

- Read an Overview of Azure NAT Gateway
- Read about the NAT Gateway resource
- Learn more about Azure Resource Manager

Quickstart: Create a private endpoint by using an ARM template

Article • 03/10/2023

In this quickstart, you'll use an Azure Resource Manager template (ARM template) to create a private endpoint.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

You can also create a private endpoint by using the Azure portal, Azure PowerShell, or the Azure CLI.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button here. The ARM template will open in the Azure portal.

Deploy to Azure

Prerequisites

You need an Azure account with an active subscription. If you don't already have an Azure account, create an account for free.

Review the template

This template creates a private endpoint for an instance of Azure SQL Database.

The template that this quickstart uses is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
```

```
      "templateHash": "14846974543330599630"
    }
  },
  "parameters": {
    "sqlAdministratorLogin": {
      "type": "string",
      "metadata": {
        "description": "The administrator username of the SQL logical
server"
      }
    },
    "sqlAdministratorLoginPassword": {
      "type": "secureString",
      "metadata": {
        "description": "The administrator password of the SQL logical
server."
      }
    },
    "vmAdminUsername": {
      "type": "string",
      "metadata": {
        "description": "Username for the Virtual Machine."
      }
    },
    "vmAdminPassword": {
      "type": "secureString",
      "metadata": {
        "description": "Password for the Virtual Machine. The password must
be at least 12 characters long and have lower case, upper characters, digit
and a special character (Regex match)"
      }
    },
    "VmSize": {
      "type": "string",
      "defaultValue": "Standard_D2_v3",
      "metadata": {
        "description": "The size of the VM"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    }
  },
  "variables": {
    "vnetName": "myVirtualNetwork",
    "vnetAddressPrefix": "10.0.0.0/16",
    "subnet1Prefix": "10.0.0.0/24",
    "subnet1Name": "mySubnet",
    "sqlServerName": "[format('sqlserver{0}',
uniqueString(resourceGroup().id))]",
    "databaseName": "[format('{0}/sample-db', variables('sqlServerName'))]",
```

```
    "privateEndpointName": "myPrivateEndpoint",
    "privateDnsZoneName": "[format('privatelink{0}',
environment().suffixes.sqlServerHostname)]",
    "pvtEndpointDnsGroupName": "[format('{0}/mydnsgroupname',
variables('privateEndpointName'))]",
    "vmName": "[take(format('myVm{0}', uniqueString(resourceGroup().id)),
15)]",
    "publicIpAddressName": "[format('{0}PublicIP', variables('vmName'))]",
    "networkInterfaceName": "[format('{0}NetInt', variables('vmName'))]",
    "osDiskType": "StandardSSD_LRS"
  },
  "resources": [
    {
      "type": "Microsoft.Sql/servers",
      "apiVersion": "2021-11-01-preview",
      "name": "[variables('sqlServerName')]",
      "location": "[parameters('location')]",
      "tags": {
        "displayName": "[variables('sqlServerName')]"
      },
      "properties": {
        "administratorLogin": "[parameters('sqlAdministratorLogin')]",
        "administratorLoginPassword": "
[parameters('sqlAdministratorLoginPassword')]",
        "version": "12.0",
        "publicNetworkAccess": "Disabled"
      }
    },
    {
      "type": "Microsoft.Sql/servers/databases",
      "apiVersion": "2021-11-01-preview",
      "name": "[variables('databaseName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Basic",
        "tier": "Basic",
        "capacity": 5
      },
      "tags": {
        "displayName": "[variables('databaseName')]"
      },
      "properties": {
        "collation": "SQL_Latin1_General_CP1_CI_AS",
        "maxSizeBytes": 104857600,
        "sampleName": "AdventureWorksLT"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Sql/servers', variables('sqlServerName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "2021-05-01",
      "name": "[variables('vnetName')]",
      "location": "[parameters('location')]",
```

```
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[variables('vnetAddressPrefix')]"
          ]
        }
      }
    },
    {
      "type": "Microsoft.Network/virtualNetworks/subnets",
      "apiVersion": "2021-05-01",
      "name": "[format('{0}/{1}', variables('vnetName'),
variables('subnet1Name'))]",
      "properties": {
        "addressPrefix": "[variables('subnet1Prefix')]",
        "privateEndpointNetworkPolicies": "Disabled"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/virtualNetworks',
variables('vnetName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/privateEndpoints",
      "apiVersion": "2021-05-01",
      "name": "[variables('privateEndpointName')]",
      "location": "[parameters('location')]",
      "properties": {
        "subnet": {
          "id": "[resourceId('Microsoft.Network/virtualNetworks/subnets',
variables('vnetName'), variables('subnet1Name'))]"
        },
        "privateLinkServiceConnections": [
          {
            "name": "[variables('privateEndpointName')]",
            "properties": {
              "privateLinkServiceId": "[resourceId('Microsoft.Sql/servers',
variables('sqlServerName'))]",
              "groupIds": [
                "sqlServer"
              ]
            }
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Sql/servers', variables('sqlServerName'))]",
        "[resourceId('Microsoft.Network/virtualNetworks/subnets',
variables('vnetName'), variables('subnet1Name'))]",
        "[resourceId('Microsoft.Network/virtualNetworks',
variables('vnetName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/privateDnsZones",
```

```
      "apiVersion": "2020-06-01",
      "name": "[variables('privateDnsZoneName')]",
      "location": "global",
      "properties": {},
      "dependsOn": [
        "[resourceId('Microsoft.Network/virtualNetworks',
variables('vnetName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/privateDnsZones/virtualNetworkLinks",
      "apiVersion": "2020-06-01",
      "name": "[format('{0}/{1}', variables('privateDnsZoneName'),
format('{0}-link', variables('privateDnsZoneName')))]",
      "location": "global",
      "properties": {
        "registrationEnabled": false,
        "virtualNetwork": {
          "id": "[resourceId('Microsoft.Network/virtualNetworks',
variables('vnetName'))]"
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/privateDnsZones',
variables('privateDnsZoneName'))]",
        "[resourceId('Microsoft.Network/virtualNetworks',
variables('vnetName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/privateEndpoints/privateDnsZoneGroups",
      "apiVersion": "2021-05-01",
      "name": "[variables('pvtEndpointDnsGroupName')]",
      "properties": {
        "privateDnsZoneConfigs": [
          {
            "name": "config1",
            "properties": {
              "privateDnsZoneId": "
[resourceId('Microsoft.Network/privateDnsZones',
variables('privateDnsZoneName'))]"
            }
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/privateDnsZones',
variables('privateDnsZoneName'))]",
        "[resourceId('Microsoft.Network/privateEndpoints',
variables('privateEndpointName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2021-05-01",
```

```
      "name": "[variables('publicIpAddressName')]",
      "location": "[parameters('location')]",
      "tags": {
        "displayName": "[variables('publicIpAddressName')]"
      },
      "properties": {
        "publicIPAllocationMethod": "Dynamic"
      }
    },
    {
      "type": "Microsoft.Network/networkInterfaces",
      "apiVersion": "2021-05-01",
      "name": "[variables('networkInterfaceName')]",
      "location": "[parameters('location')]",
      "tags": {
        "displayName": "[variables('networkInterfaceName')]"
      },
      "properties": {
        "ipConfigurations": [
          {
            "name": "ipConfig1",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses',
variables('publicIpAddressName'))]"
              },
              "subnet": {
                "id": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
variables('vnetName'), variables('subnet1Name'))]"
              }
            }
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/publicIPAddresses',
variables('publicIpAddressName'))]",
        "[resourceId('Microsoft.Network/virtualNetworks/subnets',
variables('vnetName'), variables('subnet1Name'))]",
        "[resourceId('Microsoft.Network/virtualNetworks',
variables('vnetName'))]"
      ]
    },
    {
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2021-11-01",
      "name": "[variables('vmName')]",
      "location": "[parameters('location')]",
      "tags": {
        "displayName": "[variables('vmName')]"
      },
      "properties": {
        "hardwareProfile": {
```

```
          "vmSize": "[parameters('VmSize')]"
        },
        "osProfile": {
          "computerName": "[variables('vmName')]",
          "adminUsername": "[parameters('vmAdminUsername')]",
          "adminPassword": "[parameters('vmAdminPassword')]"
        },
        "storageProfile": {
          "imageReference": {
            "publisher": "MicrosoftWindowsServer",
            "offer": "WindowsServer",
            "sku": "2019-Datacenter",
            "version": "latest"
          },
          "osDisk": {
            "name": "[format('{0}OsDisk', variables('vmName'))]",
            "caching": "ReadWrite",
            "createOption": "FromImage",
            "managedDisk": {
              "storageAccountType": "[variables('osDiskType')]"
            },
            "diskSizeGB": 128
          }
        },
        "networkProfile": {
          "networkInterfaces": [
            {
              "id": "[resourceId('Microsoft.Network/networkInterfaces',
variables('networkInterfaceName'))]"
            }
          ]
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkInterfaces',
variables('networkInterfaceName'))]"
      ]
    }
  ]
}
```

The template defines multiple Azure resources:

- **Microsoft.Sql/servers**: The instance of SQL Database with the sample database.
- **Microsoft.Sql/servers/databases**: The sample database.
- **Microsoft.Network/virtualNetworks**: The virtual network where the private endpoint is deployed.
- **Microsoft.Network/privateEndpoints**: The private endpoint that you use to access the instance of SQL Database.
- **Microsoft.Network/privateDnsZones**: The zone that you use to resolve the private endpoint IP address.

- Microsoft.Network/privateDnsZones/virtualNetworkLinks
- Microsoft.Network/privateEndpoints/privateDnsZoneGroups: The zone group that you use to associate the private endpoint with a private DNS zone.
- Microsoft.Network/publicIpAddresses: The public IP address that you use to access the virtual machine.
- Microsoft.Network/networkInterfaces: The network interface for the virtual machine.
- Microsoft.Compute/virtualMachines: The virtual machine that you use to test the connection of the private endpoint to the instance of SQL Database.

Deploy the template

Deploy the ARM template to Azure by doing the following:

1. Sign in to Azure and open the ARM template by selecting the **Deploy to Azure** button here. The template creates the private endpoint, the instance of SQL Database, the network infrastructure, and a virtual machine to be validated.

   Deploy to Azure

2. Select your resource group or create a new one.

3. Enter the SQL administrator sign-in name and password.

4. Enter the virtual machine administrator username and password.

5. Read the terms and conditions statement. If you agree, select **I agree to the terms and conditions stated above**, and then select **Purchase**. The deployment can take 20 minutes or longer to complete.

Validate the deployment

> **Note**
>
> The ARM template generates a unique name for the virtual machine myVm**{uniqueid}** resource, and for the SQL Database sqlserver**{uniqueid}** resource. Substitute your generated value for **{uniqueid}**.

Connect to a VM from the internet

Connect to the VM *myVm{uniqueid}* from the internet by doing the following:

1. In the portal's search bar, enter *myVm{uniqueid}*.

2. Select **Connect**. **Connect to virtual machine** opens.

3. Select **Download RDP File**. Azure creates a Remote Desktop Protocol (RDP) file and downloads it to your computer.

4. Open the downloaded RDP file.

   a. If you're prompted, select **Connect**.
   b. Enter the username and password that you specified when you created the VM.

   > ⓘ **Note**
   >
   > You might need to select **More choices** > **Use a different account** to specify the credentials you entered when you created the VM.

5. Select **OK**.

   You might receive a certificate warning during the sign-in process. If you do, select **Yes** or **Continue**.

6. After the VM desktop appears, minimize it to go back to your local desktop.

Access the SQL Database server privately from the VM

To connect to the SQL Database server from the VM by using the private endpoint, do the following:

1. On the Remote Desktop of *myVM{uniqueid}*, open PowerShell.

2. Run the following command:

   `nslookup sqlserver{uniqueid}.database.windows.net`

   You'll receive a message that's similar to this one:

```
Server:  UnKnown
Address:  168.63.129.16
Non-authoritative answer:
Name:    sqlserver.privatelink.database.windows.net
```

```
    Address:  10.0.0.5
    Aliases:  sqlserver.database.windows.net
```

3. Install SQL Server Management Studio.

4. On the **Connect to server** pane, do the following:

   - For **Server type**, select **Database Engine**.
   - For **Server name**, select **sqlserver{uniqueid}.database.windows.net**.
   - For **Username**, enter the username that was provided earlier.
   - For **Password**, enter the password that was provided earlier.
   - For **Remember password**, select **Yes**.

5. Select **Connect**.

6. On the left pane, select **Databases**. Optionally, you can create or query information from *sample-db*.

7. Close the Remote Desktop connection to *myVm{uniqueid}*.

Clean up resources

When you no longer need the resources that you created with the private endpoint, delete the resource group. Doing so removes the private endpoint and all the related resources.

To delete the resource group, run the `Remove-AzResourceGroup` cmdlet:

Azure PowerShell

```
Remove-AzResourceGroup -Name <your resource group name>
```

Next steps

For more information about the services that support private endpoints, see:

What is Azure Private Link?

Quickstart: Create a private link service using an ARM template

Article • 03/30/2023

In this quickstart, you use an Azure Resource Manager template (ARM template) to create a private link service.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

You can also complete this quickstart by using the Azure portal, Azure PowerShell, or the Azure CLI.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template opens in the Azure portal.

Deploy to Azure

Prerequisites

You need an Azure account with an active subscription. Create an account for free.

Review the template

This template creates a private link service.

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "4187161334981532249"
    }
  },
```

```
  "parameters": {
    "vmAdminUsername": {
      "type": "string",
      "metadata": {
        "description": "Username for the Virtual Machine."
      }
    },
    "vmAdminPassword": {
      "type": "secureString",
      "metadata": {
        "description": "Password for the Virtual Machine. The password must
be at least 12 characters long and have lower case, upper characters, digit
and a special character (Regex match)"
      }
    },
    "vmSize": {
      "type": "string",
      "defaultValue": "Standard_D2_v3",
      "metadata": {
        "description": "The size of the VM"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    }
  },
  "variables": {
    "vnetName": "myVirtualNetwork",
    "vnetConsumerName": "myPEVnet",
    "vnetAddressPrefix": "10.0.0.0/16",
    "frontendSubnetPrefix": "10.0.1.0/24",
    "frontendSubnetName": "frontendSubnet",
    "backendSubnetPrefix": "10.0.2.0/24",
    "backendSubnetName": "backendSubnet",
    "consumerSubnetPrefix": "10.0.0.0/24",
    "consumerSubnetName": "myPESubnet",
    "loadbalancerName": "myILB",
    "backendPoolName": "myBackEndPool",
    "loadBalancerFrontEndIpConfigurationName": "myFrontEnd",
    "healthProbeName": "myHealthProbe",
    "privateEndpointName": "myPrivateEndpoint",
    "vmName": "[take(format('myVm{0}', uniqueString(resourceGroup().id)),
15)]",
    "networkInterfaceName": "[format('{0}NetInt', variables('vmName'))]",
    "vmConsumerName": "[take(format('myConsumerVm{0}',
uniqueString(resourceGroup().id)), 15)]",
    "publicIpAddressConsumerName": "[format('{0}PublicIP',
variables('vmConsumerName'))]",
    "networkInterfaceConsumerName": "[format('{0}NetInt',
variables('vmConsumerName'))]",
    "osDiskType": "StandardSSD_LRS",
```

```
    "privatelinkServiceName": "myPLS",
    "loadbalancerId": "[resourceId('Microsoft.Network/loadBalancers',
variables('loadbalancerName'))]"
  },
  "resources": [
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "2021-05-01",
      "name": "[variables('vnetName')]",
      "location": "[parameters('location')]",
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[variables('vnetAddressPrefix')]"
          ]
        },
        "subnets": [
          {
            "name": "[variables('frontendSubnetName')]",
            "properties": {
              "addressPrefix": "[variables('frontendSubnetPrefix')]",
              "privateLinkServiceNetworkPolicies": "Disabled"
            }
          },
          {
            "name": "[variables('backendSubnetName')]",
            "properties": {
              "addressPrefix": "[variables('backendSubnetPrefix')]"
            }
          }
        ]
      }
    },
    {
      "type": "Microsoft.Network/loadBalancers",
      "apiVersion": "2021-05-01",
      "name": "[variables('loadbalancerName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard"
      },
      "properties": {
        "frontendIPConfigurations": [
          {
            "name": "
[variables('loadBalancerFrontEndIpConfigurationName')]",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "subnet": {
                "id": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
variables('vnetName'), variables('frontendSubnetName'))]"
              }
            }
          }
```

```
        ],
        "backendAddressPools": [
          {
            "name": "[variables('backendPoolName')]"
          }
        ],
        "inboundNatRules": [
          {
            "name": "RDP-VM0",
            "properties": {
              "frontendIPConfiguration": {
                "id": "
[resourceId('Microsoft.Network/loadBalancers/frontendIpConfigurations',
variables('loadbalancerName'),
variables('loadBalancerFrontEndIpConfigurationName'))]"
              },
              "protocol": "Tcp",
              "frontendPort": 3389,
              "backendPort": 3389,
              "enableFloatingIP": false
            }
          }
        ],
        "loadBalancingRules": [
          {
            "name": "myHTTPRule",
            "properties": {
              "frontendIPConfiguration": {
                "id": "
[resourceId('Microsoft.Network/loadBalancers/frontendIpConfigurations',
variables('loadbalancerName'),
variables('loadBalancerFrontEndIpConfigurationName'))]"
              },
              "backendAddressPool": {
                "id": "
[resourceId('Microsoft.Network/loadBalancers/backendAddressPools',
variables('loadbalancerName'), variables('backendPoolName'))]"
              },
              "probe": {
                "id": "[resourceId('Microsoft.Network/loadBalancers/probes',
variables('loadbalancerName'), variables('healthProbeName'))]"
              },
              "protocol": "Tcp",
              "frontendPort": 80,
              "backendPort": 80,
              "idleTimeoutInMinutes": 15
            }
          }
        ],
        "probes": [
          {
            "properties": {
              "protocol": "Tcp",
              "port": 80,
              "intervalInSeconds": 15,
```

```
              "numberOfProbes": 2
            },
            "name": "[variables('healthProbeName')]"
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/virtualNetworks',
variables('vnetName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/networkInterfaces",
      "apiVersion": "2021-05-01",
      "name": "[variables('networkInterfaceName')]",
      "location": "[parameters('location')]",
      "tags": {
        "displayName": "[variables('networkInterfaceName')]"
      },
      "properties": {
        "ipConfigurations": [
          {
            "name": "ipConfig1",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "subnet": {
                "id": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
variables('vnetName'), variables('backendSubnetName'))]"
              },
              "loadBalancerBackendAddressPools": [
                {
                  "id": "
[resourceId('Microsoft.Network/loadBalancers/backendAddressPools',
variables('loadbalancerName'), variables('backendPoolName'))]"
                }
              ],
              "loadBalancerInboundNatRules": [
                {
                  "id": "
[resourceId('Microsoft.Network/loadBalancers/inboundNatRules/',
variables('loadbalancerName'), 'RDP-VM0')]"
                }
              ]
            }
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/loadBalancers',
variables('loadbalancerName'))]"
      ]
    },
    {
      "type": "Microsoft.Compute/virtualMachines",
```

```
      "apiVersion": "2021-11-01",
      "name": "[variables('vmName')]",
      "location": "[parameters('location')]",
      "tags": {
        "displayName": "[variables('vmName')]"
      },
      "properties": {
        "hardwareProfile": {
          "vmSize": "[parameters('vmSize')]"
        },
        "osProfile": {
          "computerName": "[variables('vmName')]",
          "adminUsername": "[parameters('vmAdminUsername')]",
          "adminPassword": "[parameters('vmAdminPassword')]"
        },
        "storageProfile": {
          "imageReference": {
            "publisher": "MicrosoftWindowsServer",
            "offer": "WindowsServer",
            "sku": "2019-Datacenter",
            "version": "latest"
          },
          "osDisk": {
            "name": "[format('{0}OsDisk', variables('vmName'))]",
            "caching": "ReadWrite",
            "createOption": "FromImage",
            "managedDisk": {
              "storageAccountType": "[variables('osDiskType')]"
            },
            "diskSizeGB": 128
          }
        },
        "networkProfile": {
          "networkInterfaces": [
            {
              "id": "[resourceId('Microsoft.Network/networkInterfaces',
variables('networkInterfaceName'))]"
            }
          ]
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkInterfaces',
variables('networkInterfaceName'))]"
      ]
    },
    {
      "type": "Microsoft.Compute/virtualMachines/extensions",
      "apiVersion": "2021-11-01",
      "name": "[format('{0}/{1}', variables('vmName'),
'installcustomscript')]",
      "location": "[parameters('location')]",
      "tags": {
        "displayName": "install software for Windows VM"
      },
```

```
      "properties": {
        "publisher": "Microsoft.Compute",
        "type": "CustomScriptExtension",
        "typeHandlerVersion": "1.9",
        "autoUpgradeMinorVersion": true,
        "protectedSettings": {
          "commandToExecute": "powershell -ExecutionPolicy Unrestricted
Install-WindowsFeature -Name Web-Server"
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Compute/virtualMachines',
variables('vmName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/privateLinkServices",
      "apiVersion": "2021-05-01",
      "name": "[variables('privatelinkServiceName')]",
      "location": "[parameters('location')]",
      "properties": {
        "enableProxyProtocol": false,
        "loadBalancerFrontendIpConfigurations": [
          {
            "id": "
[resourceId('Microsoft.Network/loadBalancers/frontendIpConfigurations',
variables('loadbalancerName'),
variables('loadBalancerFrontEndIpConfigurationName'))]"
          }
        ],
        "ipConfigurations": [
          {
            "name": "snet-provider-default-1",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "privateIPAddressVersion": "IPv4",
              "subnet": {
                "id": "[reference(variables('loadbalancerId'), '2019-06-
01').frontendIPConfigurations[0].properties.subnet.id]"
              },
              "primary": false
            }
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/loadBalancers',
variables('loadbalancerName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "2021-05-01",
      "name": "[variables('vnetConsumerName')]",
      "location": "[parameters('location')]",
```

```
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[variables('vnetAddressPrefix')]"
          ]
        },
        "subnets": [
          {
            "name": "[variables('consumerSubnetName')]",
            "properties": {
              "addressPrefix": "[variables('consumerSubnetPrefix')]",
              "privateEndpointNetworkPolicies": "Disabled"
            }
          },
          {
            "name": "[variables('backendSubnetName')]",
            "properties": {
              "addressPrefix": "[variables('backendSubnetPrefix')]"
            }
          }
        ]
      }
    },
    {
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2021-05-01",
      "name": "[variables('publicIpAddressConsumerName')]",
      "location": "[parameters('location')]",
      "tags": {
        "displayName": "[variables('publicIpAddressConsumerName')]"
      },
      "properties": {
        "publicIPAllocationMethod": "Dynamic",
        "dnsSettings": {
          "domainNameLabel": "[toLower(variables('vmConsumerName'))]"
        }
      }
    },
    {
      "type": "Microsoft.Network/networkInterfaces",
      "apiVersion": "2021-05-01",
      "name": "[variables('networkInterfaceConsumerName')]",
      "location": "[parameters('location')]",
      "tags": {
        "displayName": "[variables('networkInterfaceConsumerName')]"
      },
      "properties": {
        "ipConfigurations": [
          {
            "name": "ipConfig1",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses',
variables('publicIpAddressConsumerName'))]"
```

```
              },
              "subnet": {
                "id": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
variables('vnetConsumerName'), variables('consumerSubnetName'))]"
              }
            }
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/publicIPAddresses',
variables('publicIpAddressConsumerName'))]",
        "[resourceId('Microsoft.Network/virtualNetworks',
variables('vnetConsumerName'))]"
      ]
    },
    {
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2021-11-01",
      "name": "[variables('vmConsumerName')]",
      "location": "[parameters('location')]",
      "tags": {
        "displayName": "[variables('vmConsumerName')]"
      },
      "properties": {
        "hardwareProfile": {
          "vmSize": "[parameters('vmSize')]"
        },
        "osProfile": {
          "computerName": "[variables('vmConsumerName')]",
          "adminUsername": "[parameters('vmAdminUsername')]",
          "adminPassword": "[parameters('vmAdminPassword')]"
        },
        "storageProfile": {
          "imageReference": {
            "publisher": "MicrosoftWindowsServer",
            "offer": "WindowsServer",
            "sku": "2019-Datacenter",
            "version": "latest"
          },
          "osDisk": {
            "name": "[format('{0}OsDisk', variables('vmConsumerName'))]",
            "caching": "ReadWrite",
            "createOption": "FromImage",
            "managedDisk": {
              "storageAccountType": "[variables('osDiskType')]"
            },
            "diskSizeGB": 128
          }
        },
        "networkProfile": {
          "networkInterfaces": [
            {
              "id": "[resourceId('Microsoft.Network/networkInterfaces',
```

```
variables('networkInterfaceConsumerName'))]"
            }
          ]
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/networkInterfaces',
variables('networkInterfaceConsumerName'))]"
      ]
    },
    {
      "type": "Microsoft.Network/privateEndpoints",
      "apiVersion": "2021-05-01",
      "name": "[variables('privateEndpointName')]",
      "location": "[parameters('location')]",
      "properties": {
        "subnet": {
          "id": "[resourceId('Microsoft.Network/virtualNetworks/subnets',
variables('vnetConsumerName'), variables('consumerSubnetName'))]"
        },
        "privateLinkServiceConnections": [
          {
            "name": "[variables('privateEndpointName')]",
            "properties": {
              "privateLinkServiceId": "
[resourceId('Microsoft.Network/privateLinkServices',
variables('privatelinkServiceName'))]"
            }
          }
        ]
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/privateLinkServices',
variables('privatelinkServiceName'))]",
        "[resourceId('Microsoft.Network/virtualNetworks',
variables('vnetConsumerName'))]"
      ]
    }
  ]
}
```

Multiple Azure resources are defined in the template:

- **Microsoft.Network/virtualNetworks**: There's one virtual network for each virtual machine.
- **Microsoft.Network/loadBalancers**: The load balancer that exposes the virtual machines that host the service.
- **Microsoft.Network/networkInterfaces**: There are two network interfaces, one for each virtual machine.

- Microsoft.Compute/virtualMachines: There are two virtual machines, one that hosts the service and one that tests the connection to the private endpoint.
- Microsoft.Compute/virtualMachines/extensions: The extension that installs a web server.
- Microsoft.Network/privateLinkServices: The private link service to expose the service.
- Microsoft.Network/publicIpAddresses: There are two public IP addresses, one for each virtual machine.
- Microsoft.Network/privateendpoints: The private endpoint to access the service.

Deploy the template

Here's how to deploy the ARM template to Azure:

1. To sign in to Azure and open the template, select **Deploy to Azure**. The template creates a virtual machine, standard load balancer, private link service, private endpoint, networking, and a virtual machine to validate.

   Deploy to Azure

2. Select or create your resource group.
3. Enter the virtual machine administrator username and password.
4. Select **Review + create**.
5. Select **Create**.

   The deployment takes a few minutes to complete.

Validate the deployment

> **Note**
>
> The ARM template generates a unique name for the virtual machine myConsumerVm**{uniqueid}** resource. Substitute your generated value for **{uniqueid}**.

Connect to a VM from the internet

Connect to the VM *myConsumerVm{uniqueid}* from the internet as follows:

1. In the portal's search bar, enter *myConsumerVm{uniqueid}*.
2. Select **Connect**. **Connect to virtual machine** opens.
3. Select **Download RDP File**. Azure creates a Remote Desktop Protocol (*.rdp*) file and downloads it to your computer.
4. Open the RDP file that was downloaded to your computer.

   a. If prompted, select **Connect**.

   b. Enter the username and password you specified when you created the VM.

   > **Note**
   >
   > You might need to select **More choices** > **Use a different account**, to specify the credentials you entered when you created the VM.

5. Select **OK**.
6. You might receive a certificate warning during the sign-in process. If you receive a certificate warning, select **Yes** or **Continue**.
7. After the VM desktop appears, minimize it to go back to your local desktop.

Access the http service privately from the VM

Here's how to connect to the http service from the VM by using the private endpoint.

1. Go to the Remote Desktop of *myConsumerVm{uniqueid}*.
2. Open a browser, and enter the private endpoint address: `http://10.0.0.5/`.
3. The default IIS page appears.

Clean up resources

When you no longer need the resources that you created with the private link service, delete the resource group. This operation removes the private link service and all the related resources.

To delete the resource group, call the `Remove-AzResourceGroup` cmdlet:

Azure PowerShell

```
Remove-AzResourceGroup -Name <your resource group name>
```

Next steps

For more information on the services that support a private endpoint, see:

Private Link availability

Quickstart: Create a Traffic Manager profile using an ARM template

Article • 03/10/2023

This quickstart describes how to use an Azure Resource Manager template (ARM Template) to create a Traffic Manager profile with external endpoints using the performance routing method.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.6.18.56646",
      "templateHash": "14278858930741668940"
    }
  },
  "parameters": {
    "uniqueDnsName": {
      "type": "string",
```

```
      "metadata": {
        "description": "Relative DNS name for the traffic manager profile,
must be globally unique."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Network/trafficmanagerprofiles",
      "apiVersion": "2018-08-01",
      "name": "ExternalEndpointExample",
      "location": "global",
      "properties": {
        "profileStatus": "Enabled",
        "trafficRoutingMethod": "Performance",
        "dnsConfig": {
          "relativeName": "[parameters('uniqueDnsName')]",
          "ttl": 30
        },
        "monitorConfig": {
          "protocol": "HTTPS",
          "port": 443,
          "path": "/",
          "expectedStatusCodeRanges": [
            {
              "min": 200,
              "max": 202
            },
            {
              "min": 301,
              "max": 302
            }
          ]
        },
        "endpoints": [
          {
            "type":
"Microsoft.Network/TrafficManagerProfiles/ExternalEndpoints",
            "name": "endpoint1",
            "properties": {
              "target": "www.microsoft.com",
              "endpointStatus": "Enabled",
              "endpointLocation": "northeurope"
            }
          },
          {
            "type":
"Microsoft.Network/TrafficManagerProfiles/ExternalEndpoints",
            "name": "endpoint2",
            "properties": {
              "target": "docs.microsoft.com",
              "endpointStatus": "Enabled",
              "endpointLocation": "southcentralus"
            }
          }
```

```
        ]
      }
    }
  ]
}
```

One Azure resource is defined in the template:

- **Microsoft.Network/trafficManagerProfiles**

To find more templates that are related to Azure Traffic Manager, see Azure Quickstart Templates.

Deploy the template

1. Select **Try it** from the following code block to open Azure Cloud Shell, and then follow the instructions to sign in to Azure.

   Azure PowerShell

   ```
   $projectName = Read-Host -Prompt "Enter a project name that is used for generating resource names"
   $location = Read-Host -Prompt "Enter the location (i.e. centralus)"
   $templateUri = "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.network/traffic-manager-external-endpoint/azuredeploy.json"

   $resourceGroupName = "${projectName}rg"

   New-AzResourceGroup -Name $resourceGroupName -Location "$location"
   New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName -TemplateUri $templateUri

   Read-Host -Prompt "Press [ENTER] to continue ..."
   ```

   Wait until you see the prompt from the console.

2. Select **Copy** from the previous code block to copy the PowerShell script.

3. Right-click the shell console pane and then select **Paste**.

4. Enter the values.

   The template deployment creates a profile with two external endpoints. **Endpoint1** uses a target endpoint of `www.microsoft.com` with the location in **North Europe**. **Endpoint2** uses a target endpoint of `learn.microsoft.com` with the location in **South Central US**.

The resource group name is the project name with **rg** appended.

> ⓘ **Note**
>
> **uniqueDNSname** needs to be a globally unique name in order for the template to deploy successfully. If deployment fails, start over with Step 1.

It takes a few minutes to deploy the template. When completed, the output is similar to:

```
DeploymentName          : azuredeploy
ResourceGroupName       : myTM0919rg
ProvisioningState       : Succeeded
Timestamp               : 8/26/2020 4:52:29 PM
Mode                    : Incremental
TemplateLink            :
                          Uri            : https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101
                          -traffic-manager-external-endpoint/azuredeploy.json
                          ContentVersion : 1.0.0.0

Parameters              :
                          Name            Type                       Value
                          ===============  =========================  ==========
                          uniqueDnsName    String                     mytm0919
```

Azure PowerShell is used to deploy the template. In addition to Azure PowerShell, you can also use the Azure portal, Azure CLI, and REST API. To learn other deployment methods, see Deploy templates.

Validate the deployment

1. Determine the DNS name of the Traffic Manager profile using Get-AzTrafficManagerProfile.

   Azure PowerShell

   ```
   Get-AzTrafficManagerProfile -Name ExternalEndpointExample -
   ResourceGroupName $resourceGroupName | Select RelativeDnsName
   ```

   Copy the **RelativeDnsName** value. The DNS name of your Traffic Manager profile is `<relativednsname>.trafficmanager.net`.

2. From a local PowerShell run the following command by replacing the **{relativeDNSname}** variable with `<relativednsname>.trafficmanager.net`.

   PowerShell

   ```
   Resolve-DnsName -Name {relativeDNSname} | Select-Object NameHost |
   Select -First 1
   ```

You should get a NameHost of either `www.microsoft.com` or `learn.microsoft.com` depending on which region is closer to you.

3. To check if you can resolve to the other endpoint, disable the endpoint for the target you got in the last step. Replace the **{endpointName}** with either **endpoint1** or **endpoint2** to disable the target for `www.microsoft.com` or `learn.microsoft.com` respectively.

   Azure PowerShell

   ```
   Disable-AzTrafficManagerEndpoint -Name {endpointName} -Type
   ExternalEndpoints -ProfileName ExternalEndpointExample -
   ResourceGroupName $resourceGroupName -Force
   ```

4. Run the command from Step 2 again in a local PowerShell. This time you should get the other NameHost for the other endpoint.

Clean up resources

When you no longer need the Traffic Manager profile, delete the resource group. This removes the Traffic Manager profile and all the related resources.

To delete the resource group, call the `Remove-AzResourceGroup` cmdlet:

Azure PowerShell

```
Remove-AzResourceGroup -Name <your resource group name>
```

Next steps

In this quickstart, you created a Traffic Manager profile.

To learn more about routing traffic, continue to the Traffic Manager tutorials.

Traffic Manager tutorials

Quickstart: Create an Azure Web Application Firewall v2 by using an ARM template

Article • 03/31/2023

In this quickstart, you use an Azure Resource Manager template (ARM template) to create an Azure Web Application Firewall (WAF) v2 on Azure Application Gateway.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

> **Note**
>
> We recommend that you use the Azure Az PowerShell module to interact with Azure. See **Install Azure PowerShell** to get started. To learn how to migrate to the Az PowerShell module, see **Migrate Azure PowerShell from AzureRM to Az**.

If your environment meets the prerequisites and you're familiar with using ARM templates, you can select the **Deploy to Azure** button to open the template in the Azure portal.

Deploy to Azure

Prerequisites

- An Azure account with an active subscription. If you don't have one, you can create an account for free.

Review the template

This template creates a simple Web Application Firewall v2 on Azure Application Gateway. The template creates a public IP frontend IP address, HTTP settings, a rule with a basic listener on port 80, and a backend pool. A WAF policy with a custom rule blocks traffic to the backend pool based on an IP address match type.

The template defines the following Azure resources:

- Microsoft.Network/applicationgateways
- Microsoft.Network/ApplicationGatewayWebApplicationFirewallPolicies
- Microsoft.Network/publicIPAddresses, one for the application gateway and two for the virtual machines (VMs)
- Microsoft.Network/networkSecurityGroups
- Microsoft.Network/virtualNetworks
- Microsoft.Compute/virtualMachines, two VMs
- Microsoft.Network/networkInterfaces, one for each VM
- Microsoft.Compute/virtualMachine/extensions to configure IIS and the web pages

This template is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.15.31.15270",
      "templateHash": "7253194970749033988"
    }
  },
  "parameters": {
    "adminUsername": {
      "type": "string",
      "metadata": {
        "description": "Admin username for the backend servers"
      }
    },
    "adminPassword": {
      "type": "securestring",
      "metadata": {
        "description": "Password for the admin account on the backend servers"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "vmSize": {
      "type": "string",
      "defaultValue": "Standard_B2ms",
      "metadata": {
        "description": "Size of the virtual machine."
```

```
      }
    }
  },
  "variables": {
    "virtualMachines_myVM_name": "myVM",
    "virtualNetworks_myVNet_name": "myVNet",
    "myNic_name": "net-int",
    "ipconfig_name": "ipconfig",
    "publicIPAddress_name": "public_ip",
    "nsg_name": "vm-nsg",
    "applicationGateways_myAppGateway_name": "myAppGateway",
    "vnet_prefix": "10.0.0.0/16",
    "ag_subnet_prefix": "10.0.0.0/24",
    "backend_subnet_prefix": "10.0.1.0/24",
    "AppGW_AppFW_Pol_name": "WafPol01"
  },
  "resources": [
    {
      "copy": {
        "name": "nsg",
        "count": "[length(range(0, 2))]"
      },
      "type": "Microsoft.Network/networkSecurityGroups",
      "apiVersion": "2021-08-01",
      "name": "[format('{0}{1}', variables('nsg_name'), add(range(0, 2)
[copyIndex()], 1))]",
      "location": "[parameters('location')]",
      "properties": {
        "securityRules": [
          {
            "name": "RDP",
            "properties": {
              "protocol": "Tcp",
              "sourcePortRange": "*",
              "destinationPortRange": "3389",
              "sourceAddressPrefix": "*",
              "destinationAddressPrefix": "*",
              "access": "Allow",
              "priority": 300,
              "direction": "Inbound"
            }
          }
        ]
      }
    },
    {
      "copy": {
        "name": "publicIPAddress",
        "count": "[length(range(0, 3))]"
      },
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2021-08-01",
      "name": "[format('{0}{1}', variables('publicIPAddress_name'), range(0,
3)[copyIndex()])]",
      "location": "[parameters('location')]",
```

```
      "sku": {
        "name": "Standard"
      },
      "properties": {
        "publicIPAddressVersion": "IPv4",
        "publicIPAllocationMethod": "Static",
        "idleTimeoutInMinutes": 4
      }
    },
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "2021-08-01",
      "name": "[variables('virtualNetworks_myVNet_name')]",
      "location": "[parameters('location')]",
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[variables('vnet_prefix')]"
          ]
        },
        "subnets": [
          {
            "name": "myAGSubnet",
            "properties": {
              "addressPrefix": "[variables('ag_subnet_prefix')]",
              "privateEndpointNetworkPolicies": "Enabled",
              "privateLinkServiceNetworkPolicies": "Enabled"
            }
          },
          {
            "name": "myBackendSubnet",
            "properties": {
              "addressPrefix": "[variables('backend_subnet_prefix')]",
              "privateEndpointNetworkPolicies": "Enabled",
              "privateLinkServiceNetworkPolicies": "Enabled"
            }
          }
        ],
        "enableDdosProtection": false,
        "enableVmProtection": false
      }
    },
    {
      "copy": {
        "name": "myVM",
        "count": "[length(range(0, 2))]"
      },
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2021-11-01",
      "name": "[format('{0}{1}', variables('virtualMachines_myVM_name'),
add(range(0, 2)[copyIndex()], 1))]",
      "location": "[parameters('location')]",
      "properties": {
        "hardwareProfile": {
          "vmSize": "[parameters('vmSize')]"
```

```
        },
        "storageProfile": {
          "imageReference": {
            "publisher": "MicrosoftWindowsServer",
            "offer": "WindowsServer",
            "sku": "2019-Datacenter",
            "version": "latest"
          },
          "osDisk": {
            "osType": "Windows",
            "createOption": "FromImage",
            "caching": "ReadWrite",
            "managedDisk": {
              "storageAccountType": "StandardSSD_LRS"
            },
            "diskSizeGB": 127
          }
        },
        "osProfile": {
          "computerName": "[format('{0}{1}',
variables('virtualMachines_myVM_name'), add(range(0, 2)[copyIndex()], 1))]",
          "adminUsername": "[parameters('adminUsername')]",
          "adminPassword": "[parameters('adminPassword')]",
          "windowsConfiguration": {
            "provisionVMAgent": true,
            "enableAutomaticUpdates": true
          },
          "allowExtensionOperations": true
        },
        "networkProfile": {
          "networkInterfaces": [
            {
              "id": "[resourceId('Microsoft.Network/networkInterfaces',
format('{0}{1}', variables('myNic_name'), add(range(0, 2)[copyIndex()],
1)))]"
            }
          ]
        }
      },
      "dependsOn": [
        "myNic"
      ]
    },
    {
      "copy": {
        "name": "myVM_IIS",
        "count": "[length(range(0, 2))]"
      },
      "type": "Microsoft.Compute/virtualMachines/extensions",
      "apiVersion": "2021-11-01",
      "name": "[format('{0}{1}/IIS', variables('virtualMachines_myVM_name'),
add(range(0, 2)[copyIndex()], 1))]",
      "location": "[parameters('location')]",
      "properties": {
        "autoUpgradeMinorVersion": true,
```

```
        "publisher": "Microsoft.Compute",
        "type": "CustomScriptExtension",
        "typeHandlerVersion": "1.4",
        "settings": {
          "commandToExecute": "powershell Add-WindowsFeature Web-Server;
powershell Add-Content -Path \"C:\\inetpub\\wwwroot\\Default.htm\" -Value
$($env:computername)"
        }
      },
      "dependsOn": [
        "myVM"
      ]
    },
    {
      "type": "Microsoft.Network/applicationGateways",
      "apiVersion": "2021-08-01",
      "name": "[variables('applicationGateways_myAppGateway_name')]",
      "location": "[parameters('location')]",
      "properties": {
        "sku": {
          "name": "WAF_v2",
          "tier": "WAF_v2",
          "capacity": 2
        },
        "gatewayIPConfigurations": [
          {
            "name": "appGatewayIpConfig",
            "properties": {
              "subnet": {
                "id": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
variables('virtualNetworks_myVNet_name'), 'myAGSubnet')]"
              }
            }
          }
        ],
        "frontendIPConfigurations": [
          {
            "name": "appGwPublicFrontendIp",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses',
format('{0}0', variables('publicIPAddress_name')))]"
              }
            }
          }
        ],
        "frontendPorts": [
          {
            "name": "port_80",
            "properties": {
              "port": 80
            }
          }
```

```
        ],
        "backendAddressPools": [
          {
            "name": "myBackendPool",
            "properties": {}
          }
        ],
        "backendHttpSettingsCollection": [
          {
            "name": "myHTTPSetting",
            "properties": {
              "port": 80,
              "protocol": "Http",
              "cookieBasedAffinity": "Disabled",
              "pickHostNameFromBackendAddress": false,
              "requestTimeout": 20
            }
          }
        ],
        "httpListeners": [
          {
            "name": "myListener",
            "properties": {
              "firewallPolicy": {
                "id": "
[resourceId('Microsoft.Network/ApplicationGatewayWebApplicationFirewallPolic
ies', variables('AppGW_AppFW_Pol_name'))]"
              },
              "frontendIPConfiguration": {
                "id": "
[resourceId('Microsoft.Network/applicationGateways/frontendIPConfigurations'
, variables('applicationGateways_myAppGateway_name'),
'appGwPublicFrontendIp')]"
              },
              "frontendPort": {
                "id": "
[resourceId('Microsoft.Network/applicationGateways/frontendPorts',
variables('applicationGateways_myAppGateway_name'), 'port_80')]"
              },
              "protocol": "Http",
              "requireServerNameIndication": false
            }
          }
        ],
        "requestRoutingRules": [
          {
            "name": "myRoutingRule",
            "properties": {
              "ruleType": "Basic",
              "priority": 10,
              "httpListener": {
                "id": "
[resourceId('Microsoft.Network/applicationGateways/httpListeners',
variables('applicationGateways_myAppGateway_name'), 'myListener')]"
              },
```

```
              "backendAddressPool": {
                "id": "
[resourceId('Microsoft.Network/applicationGateways/backendAddressPools',
variables('applicationGateways_myAppGateway_name'), 'myBackendPool')]"
              },
              "backendHttpSettings": {
                "id": "
[resourceId('Microsoft.Network/applicationGateways/backendHttpSettingsCollec
tion', variables('applicationGateways_myAppGateway_name'),
'myHTTPSetting')]"
              }
            }
          }
        ],
        "enableHttp2": false,
        "firewallPolicy": {
          "id": "
[resourceId('Microsoft.Network/ApplicationGatewayWebApplicationFirewallPolic
ies', variables('AppGW_AppFW_Pol_name'))]"
        }
      },
      "dependsOn": [
        "
[resourceId('Microsoft.Network/ApplicationGatewayWebApplicationFirewallPolic
ies', variables('AppGW_AppFW_Pol_name'))]",
        "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworks_myVNet_name'))]",
        "publicIPAddress"
      ]
    },
    {
      "type":
"Microsoft.Network/ApplicationGatewayWebApplicationFirewallPolicies",
      "apiVersion": "2021-08-01",
      "name": "[variables('AppGW_AppFW_Pol_name')]",
      "location": "[parameters('location')]",
      "properties": {
        "customRules": [
          {
            "name": "CustRule01",
            "priority": 100,
            "ruleType": "MatchRule",
            "action": "Block",
            "matchConditions": [
              {
                "matchVariables": [
                  {
                    "variableName": "RemoteAddr"
                  }
                ],
                "operator": "IPMatch",
                "negationConditon": true,
                "matchValues": [
                  "10.10.10.0/24"
                ]
```

```
              }
            ]
          }
        ],
        "policySettings": {
          "requestBodyCheck": true,
          "maxRequestBodySizeInKb": 128,
          "fileUploadLimitInMb": 100,
          "state": "Enabled",
          "mode": "Prevention"
        },
        "managedRules": {
          "managedRuleSets": [
            {
              "ruleSetType": "OWASP",
              "ruleSetVersion": "3.1"
            }
          ]
        }
      }
    },
    {
      "copy": {
        "name": "myNic",
        "count": "[length(range(0, 2))]"
      },
      "type": "Microsoft.Network/networkInterfaces",
      "apiVersion": "2021-08-01",
      "name": "[format('{0}{1}', variables('myNic_name'), add(range(0, 2)
[copyIndex()], 1))]",
      "location": "[parameters('location')]",
      "properties": {
        "ipConfigurations": [
          {
            "name": "[format('{0}{1}', variables('ipconfig_name'),
add(range(0, 2)[copyIndex()], 1))]",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses',
format('{0}{1}', variables('publicIPAddress_name'), add(range(0, 2)
[copyIndex()], 1)))]"
              },
              "subnet": {
                "id": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
variables('virtualNetworks_myVNet_name'), 'myBackendSubnet')]"
              },
              "primary": true,
              "privateIPAddressVersion": "IPv4",
              "applicationGatewayBackendAddressPools": [
                {
                  "id": "
[resourceId('Microsoft.Network/applicationGateways/backendAddressPools',
variables('applicationGateways_myAppGateway_name'), 'myBackendPool')]"
```

```
                }
              ]
            }
          }
        ],
        "enableAcceleratedNetworking": false,
        "enableIPForwarding": false,
        "networkSecurityGroup": {
          "id": "[resourceId('Microsoft.Network/networkSecurityGroups',
format('{0}{1}', variables('nsg_name'), add(range(0, 2)[copyIndex()], 1)))]"
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/applicationGateways',
variables('applicationGateways_myAppGateway_name'))]",
        "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworks_myVNet_name'))]",
        "nsg",
        "publicIPAddress"
      ]
    }
  ]
}
```

Deploy the template

Deploy the ARM template to Azure:

1. Select **Deploy to Azure** to sign in to Azure and open the template. The template creates an application gateway, the network infrastructure, and two VMs in the backend pool running IIS.

   Deploy to Azure

2. Select or create a resource group.

3. Select **Review + create**, and when validation passes, select **Create**. The deployment can take 10 minutes or longer to complete.

Validate the deployment

Although IIS isn't required, the template installs IIS on the backend servers so you can verify that Azure successfully created a WAF v2 on the application gateway.

Use IIS to test the application gateway:

1. Copy the public IP address for the application gateway from its **Overview** page.

You can also search for *application gateways* in the Azure search box. The list of application gateways shows the public IP addresses in the **Public IP address** column.

2. Paste the IP address into the address bar of your browser to browse that address.

3. Check the response. A **403 Forbidden** response verifies that the WAF is successfully blocking connections to the backend pool.

4. To change the custom rule to allow traffic, run the following Azure PowerShell script, replacing your resource group name:

Azure PowerShell

```
$rg = "<your resource group name>"
$AppGW = Get-AzApplicationGateway -Name myAppGateway -ResourceGroupName
$rg
$pol = Get-AzApplicationGatewayFirewallPolicy -Name WafPol01 -
ResourceGroupName $rg
$pol[0].customrules[0].action = "allow"
$rule = $pol.CustomRules
Set-AzApplicationGatewayFirewallPolicy -Name WafPol01 -
ResourceGroupName $rg -CustomRule $rule
$AppGW.FirewallPolicy = $pol
Set-AzApplicationGateway -ApplicationGateway $AppGW
```

5. Refresh your browser several times. You should see connections to both myVM1 and myVM2.

Clean up resources

When you no longer need the resources you created in this quickstart, delete the resource group to remove the application gateway and all its related resources.

To delete the resource group, call the `Remove-AzResourceGroup` cmdlet:

Azure PowerShell

```
Remove-AzResourceGroup -Name "<your resource group name>"
```

Next steps

Tutorial: Create an application gateway with a Web Application Firewall by using the Azure portal

Quickstart: Create an Azure Attestation provider with an ARM template

Article • 03/09/2023

Microsoft Azure Attestation is a solution for attesting Trusted Execution Environments (TEEs). This quickstart focuses on the process of deploying an Azure Resource Manager template (ARM template) to create a Microsoft Azure Attestation policy.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "9350596888842621478"
    }
  },
  "parameters": {
    "attestationProviderName": {
      "type": "string",
```

```
      "defaultValue": "[uniqueString(resourceGroup().name)]",
      "metadata": {
        "description": "Name of the Attestation provider. Must be between 3
and 24 characters in length and use numbers and lower-case letters only."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "policySigningCertificates": {
      "type": "string",
      "defaultValue": ""
    }
  },
  "variables": {
    "PolicySigningCertificates": {
      "PolicySigningCertificates": {
        "keys": [
          {
            "kty": "RSA",
            "use": "sig",
            "x5c": [
              "[parameters('policySigningCertificates')]"
            ]
          }
        ]
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Attestation/attestationProviders",
      "apiVersion": "2021-06-01-preview",
      "name": "[parameters('attestationProviderName')]",
      "location": "[parameters('location')]",
      "properties": "[if(empty(parameters('policySigningCertificates')),
json('{}'), variables('PolicySigningCertificates'))]"
    }
  ],
  "outputs": {
    "attestationName": {
      "type": "string",
      "value": "[parameters('attestationProviderName')]"
    }
  }
}
```

Azure resources defined in the template:

- Microsoft.Attestation/attestationProviders

Deploy the template

1. Select the following image to sign in to Azure and open the template.

   Deploy to Azure

2. Select or enter the following values.

   Unless it's specified, use the default value to create the attestation provider.

   - **Attestation Provider Name**: Select a name for your Azure Attestation provider.
   - **Location**: Select a location. For example, **Central US**.
   - **Tags**: Select a location. For example, **Central US**.

3. Select **Purchase**. After the attestation resource has been deployed successfully, you get a notification.

The Azure portal is used to deploy the template. In addition to the Azure portal, you can also use the Azure PowerShell, Azure CLI, and REST API. To learn other deployment methods, see Deploy templates.

Review deployed resources

You can use the Azure portal to check the attestation resource.

Clean up resources

Other Azure Attestation build upon this quickstart. If you plan to continue on to work with subsequent quickstarts and tutorials, you may wish to leave these resources in place.

When no longer needed, delete the resource group, which deletes the Attestation resource. To delete the resource group by using Azure CLI or Azure PowerShell:

CLI

Azure CLI

```
echo "Enter the Resource Group name:" &&
read resourceGroupName &&
az group delete --name $resourceGroupName &&
echo "Press [ENTER] to continue ..."
```

Next steps

In this quickstart, you created an attestation resource using an ARM template, and validated the deployment. To learn more about Azure Attestation, see Overview of Azure Attestation.

Quickstart: Create an automatic response to a specific security alert using an ARM template or Bicep

Article • 04/13/2023

In this quickstart, you'll learn how to use an Azure Resource Manager template (ARM template) or a Bicep file to create a workflow automation. The workflow automation will trigger a logic app when specific security alerts are received by Microsoft Defender for Cloud.

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

For a list of the roles and permissions required to work with Microsoft Defender for Cloud's workflow automation feature, see workflow automation.

The examples in this quickstart assume you have an existing Logic App. To deploy the example, you pass in parameters that contain the logic app name and resource group. For information about deploying a logic app, see Quickstart: Create and deploy a Consumption logic app workflow in multi-tenant Azure Logic Apps with Bicep or Quickstart: Create and deploy a Consumption logic app workflow in multi-tenant Azure Logic Apps with an ARM template.

ARM template tutorial

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "5191074894407113732"
    }
  },
  "parameters": {
    "automationName": {
      "type": "string",
      "maxLength": 24,
      "minLength": 3
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for the automation"
      }
    },
    "logicAppName": {
      "type": "string",
      "minLength": 3
    },
    "logicAppResourceGroupName": {
      "type": "string",
      "minLength": 3
    },
    "subscriptionId": {
      "type": "string",
      "defaultValue": "[subscription().subscriptionId]",
      "metadata": {
        "description": "The Azure resource GUID id of the subscription"
      }
    },
    "alertSettings": {
      "type": "object",
      "metadata": {
        "description": "The alert settings object used for deploying the automation"
      }
    }
  },
  "variables": {
    "automationDescription": "automation description for subscription {0}",
    "scopeDescription": "automation scope for subscription {0}"
```

```
  },
  "resources": [
    {
      "type": "Microsoft.Security/automations",
      "apiVersion": "2019-01-01-preview",
      "name": "[parameters('automationName')]",
      "location": "[parameters('location')]",
      "properties": {
        "description": "[format(variables('automationDescription'),
parameters('subscriptionId'))]",
        "isEnabled": true,
        "actions": [
          {
            "actionType": "LogicApp",
            "logicAppResourceId": "[resourceId('Microsoft.Logic/workflows',
parameters('logicAppName'))]",
            "uri": "
[listCallbackURL(resourceId(parameters('subscriptionId'),
parameters('logicAppResourceGroupName'),
'Microsoft.Logic/workflows/triggers', parameters('logicAppName'), 'manual'),
'2019-05-01').value]"
          }
        ],
        "scopes": [
          {
            "description": "[format(variables('scopeDescription'),
parameters('subscriptionId'))]",
            "scopePath": "[subscription().id]"
          }
        ],
        "sources": [
          {
            "copy": [
              {
                "name": "ruleSets",
                "count": "[length(range(0,
length(parameters('alertSettings').alertSeverityMapping)))]",
                "input": {
                  "rules": [
                    {
                      "propertyJPath": "
[parameters('alertSettings').alertSeverityMapping[range(0,
length(parameters('alertSettings').alertSeverityMapping))
[copyIndex('ruleSets')]].jpath]",
                      "propertyType": "String",
                      "expectedValue": "
[parameters('alertSettings').alertSeverityMapping[range(0,
length(parameters('alertSettings').alertSeverityMapping))
[copyIndex('ruleSets')]].expectedValue]",
                      "operator": "
[parameters('alertSettings').alertSeverityMapping[range(0,
length(parameters('alertSettings').alertSeverityMapping))
[copyIndex('ruleSets')]].operator]"
                    },
                    {
```

```
                        "propertyJPath": "Severity",
                        "propertyType": "String",
                        "expectedValue": "
[parameters('alertSettings').alertSeverityMapping[range(0,
length(parameters('alertSettings').alertSeverityMapping))
[copyIndex('ruleSets')]].severity]",
                        "operator": "Equals"
                      }
                    ]
                  }
                }
              ],
              "eventSource": "Alerts"
            }
          ]
        }
      }
    ]
  }
```

Relevant resources

- **Microsoft.Security/automations**: The automation that will trigger the logic app, upon receiving a Microsoft Defender for Cloud alert that contains a specific string.
- **Microsoft.Logic/workflows**: An empty triggerable Logic App.

For other Defender for Cloud quickstart templates, see these community contributed templates.

Deploy the template

- **PowerShell**:

  Azure PowerShell

  ```
  New-AzResourceGroup -Name <resource-group-name> -Location <resource-group-location> #use this command when you need to create a new resource group for your deployment
  New-AzResourceGroupDeployment -ResourceGroupName <resource-group-name> -TemplateUri https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.security/securitycenter-create-automation-for-alertnamecontains/azuredeploy.json
  ```

- **CLI**:

  Azure CLI

```
az group create --name <resource-group-name> --location <resource-group-location> #use this command when you need to create a new resource group for your deployment
az deployment group create --resource-group <my-resource-group> --template-uri https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.security/securitycenter-create-automation-for-alertnamecontains/azuredeploy.json
```

- **Portal**:

  Deploy to Azure

  To find more information about this deployment option, see Use a deployment button to deploy templates from GitHub repository.

Review deployed resources

Use the Azure portal to check the workflow automation has been deployed.

1. Sign in to the Azure portal.
2. Search for and select **Microsoft Defender for Cloud**.
3. Select **filter**.
4. Select the specific subscription on which you deployed the new workflow automation.
5. From Microsoft Defender for Cloud's menu, open **workflow automation** and check for your new automation.

Tip

If you have many workflow automations on your subscription, use the **filter by name** option.

Clean up resources

When no longer needed, delete the workflow automation using the Azure portal.

1. Sign in to the Azure portal.
2. Search for and select **Microsoft Defender for Cloud**.
3. Select **filter**.
4. Select the specific subscription on which you deployed the new workflow automation.
5. From Microsoft Defender for Cloud's menu, open **workflow automation** and find the automation to be deleted.

6. Select the checkbox for the item to be deleted.
7. From the toolbar, select **Delete**.

Bicep tutorial

Bicep is a domain-specific language (DSL) that uses declarative syntax to deploy Azure resources. It provides concise syntax, reliable type safety, and support for code reuse. Bicep offers the best authoring experience for your infrastructure-as-code solutions in Azure.

Review the Bicep file

The Bicep file used in this quickstart is from Azure Quickstart Templates.

Bicep

```
@minLength(3)
@maxLength(24)
param automationName string

@description('Location for the automation')
param location string = resourceGroup().location

@minLength(3)
param logicAppName string

@minLength(3)
param logicAppResourceGroupName string

@description('The Azure resource GUID id of the subscription')
param subscriptionId string = subscription().subscriptionId

@description('The alert settings object used for deploying the automation')
param alertSettings object

var automationDescription = 'automation description for subscription {0}'
var scopeDescription = 'automation scope for subscription {0}'

resource automation 'Microsoft.Security/automations@2019-01-01-preview' = {
  name: automationName
  location: location
  properties: {
    description: format(automationDescription, subscriptionId)
    isEnabled: true
    actions: [
      {
        actionType: 'LogicApp'
        logicAppResourceId: resourceId('Microsoft.Logic/workflows',
logicAppName)
        uri: listCallbackURL(resourceId(subscriptionId,
logicAppResourceGroupName, 'Microsoft.Logic/workflows/triggers',
logicAppName, 'manual'), '2019-05-01').value
      }
    ]
    scopes: [
      {
        description: format(scopeDescription, subscriptionId)
        scopePath: subscription().id
      }
    ]
    sources: [
      {
        eventSource: 'Alerts'
        ruleSets: [for j in range(0,
length(alertSettings.alertSeverityMapping)): {
          rules: [
            {
              propertyJPath: alertSettings.alertSeverityMapping[j].jpath
```

```
                propertyType: 'String'
                expectedValue:
alertSettings.alertSeverityMapping[j].expectedValue
                operator: alertSettings.alertSeverityMapping[j].operator
              }
              {
                propertyJPath: 'Severity'
                propertyType: 'String'
                expectedValue: alertSettings.alertSeverityMapping[j].severity
                operator: 'Equals'
              }
            ]
          }]
        }
      ]
    }
}
```

Relevant resources

- **Microsoft.Security/automations**: The automation that will trigger the logic app, upon receiving a Microsoft Defender for Cloud alert that contains a specific string.
- **Microsoft.Logic/workflows**: An empty triggerable Logic App.

For other Defender for Cloud quickstart templates, see these community contributed templates.

Deploy the Bicep file

1. Save the Bicep file as **main.bicep** to your local computer.

2. Deploy the Bicep file using either Azure CLI or Azure PowerShell.

   CLI

   Azure CLI

   ```
   az group create --name exampleRG --location eastus
   az deployment group create --resource-group exampleRG --template-
   file main.bicep --parameters automationName=<automation-name>
   logicAppName=<logic-name> logicAppResourceGroupName=<group-name>
   alertSettings={alert-settings}
   ```

   You're required to enter the following parameters:

- **automationName**: Replace **<automation-name>** with the name of the automation. It has a minimum length of three characters and a maximum length of 24 characters.
- **logicAppName**: Replace **<logic-name>** with the name of the logic app. It has a minimum length of three characters.
- **logicAppResourceGroupName**: Replace **<group-name>** with the name of the resource group in which the resources are located. It has a minimum length of three characters.
- **alertSettings**: Replace **{alert-settings}** with the alert settings object used for deploying the automation.

> **Note**
>
> When the deployment finishes, you should see a message indicating the deployment succeeded.

Review deployed resources

Use the Azure portal, Azure CLI, or Azure PowerShell to list the deployed resources in the resource group.

CLI

Azure CLI

```
az resource list --resource-group exampleRG
```

Clean up resources

When no longer needed, use the Azure portal, Azure CLI, or Azure PowerShell to delete the resource group and all of its resources.

CLI

Azure CLI

```
az group delete --name exampleRG
```

Next steps

For step-by-step tutorials that guide you through the process of creating an ARM template or a Bicep file, see:

Tutorial: Create and deploy your first ARM template

Quickstart: Create Bicep files with Visual Studio Code

Quickstart: Create an Azure key vault and a key by using ARM template

Article • 03/08/2023

Azure Key Vault is a cloud service that provides a secure store for secrets, such as keys, passwords, and certificate. This quickstart focuses on the process of deploying an Azure Resource Manager template (ARM template) to create a key vault and a key.

Prerequisites

To complete this article:

- If you don't have an Azure subscription, create a free account before you begin.
- User would need to have an Azure built-in role assigned, recommended role **contributor**. Learn more here

Review the template

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "vaultName": {
      "type": "string",
      "metadata": {
        "description": "The name of the key vault to be created."
      }
    },
    "keyName": {
      "type": "string",
      "metadata": {
        "description": "The name of the key to be created."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "The location of the resources"
      }
    },
    "skuName": {
```

```
      "type": "string",
      "defaultValue": "standard",
      "allowedValues": [
        "standard",
        "premium"
      ],
      "metadata": {
        "description": "The SKU of the vault to be created."
      }
    },
    "keyType": {
      "type": "string",
      "defaultValue": "RSA",
      "allowedValues": [
        "EC",
        "EC-HSM",
        "RSA",
        "RSA-HSM"
      ],
      "metadata": {
        "description": "The JsonWebKeyType of the key to be created."
      }
    },
    "keyOps": {
      "type": "array",
      "defaultValue": [],
      "metadata": {
        "description": "The permitted JSON web key operations of the key to
be created."
      }
    },
    "keySize": {
      "type": "int",
      "defaultValue": 2048,
      "metadata": {
        "description": "The size in bits of the key to be created."
      }
    },
    "curveName": {
      "type": "string",
      "defaultValue": "",
      "allowedValues": [
        "",
        "P-256",
        "P-256K",
        "P-384",
        "P-521"
      ],
      "metadata": {
        "description": "The JsonWebKeyCurveName of the key to be created."
      }
    }
  },
  "resources": [
    {
```

```
      "type": "Microsoft.KeyVault/vaults",
      "apiVersion": "2021-11-01-preview",
      "name": "[parameters('vaultName')]",
      "location": "[parameters('location')]",
      "properties": {
        "accessPolicies": [],
        "enableRbacAuthorization": false,
        "enableSoftDelete": false,
        "enabledForDeployment": false,
        "enabledForDiskEncryption": false,
        "enabledForTemplateDeployment": false,
        "tenantId": "[subscription().tenantId]",
        "sku": {
          "name": "[parameters('skuName')]",
          "family": "A"
        },
        "networkAcls": {
          "defaultAction": "Allow",
          "bypass": "AzureServices"
        }
      }
    },
    {
      "type": "Microsoft.KeyVault/vaults/keys",
      "apiVersion": "2021-11-01-preview",
      "name": "[format('{0}/{1}', parameters('vaultName'),
parameters('keyName'))]",
      "properties": {
        "kty": "[parameters('keyType')]",
        "keyOps": "[parameters('keyOps')]",
        "keySize": "[parameters('keySize')]",
        "curveName": "[parameters('curveName')]"
      },
      "dependsOn": [
        "[resourceId('Microsoft.KeyVault/vaults', parameters('vaultName'))]"
      ]
    }
  ],
  "outputs": {
    "proxyKey": {
      "type": "object",
      "value": "[reference(resourceId('Microsoft.KeyVault/vaults/keys',
parameters('vaultName'), parameters('keyName')))]"
    }
  }
}
```

Two resources are defined in the template:

- Microsoft.KeyVault/vaults
- Microsoft.KeyVault/vaults/keys

More Azure Key Vault template samples can be found in Azure Quickstart Templates.

Parameters and definitions

Parameter	Definition
keyOps	Specifies operations that can be performed by using the key. If you don't specify this parameter, all operations can be performed. The acceptable values for this parameter are a comma-separated list of key operations as defined by the JSON Web Key (JWK) specification: `["sign", "verify", "encrypt", "decrypt", " wrapKey", "unwrapKey"]`
CurveName	Elliptic curve (EC) name for EC key type. See JsonWebKeyCurveName
Kty	The type of key to create. For valid values, see JsonWebKeyType
Tags	Application-specific metadata in the form of key-value pairs.
nbf	Specifies the time, as a DateTime object, before which the key can't be used. The format would be Unix time stamp (the number of seconds after Unix Epoch on January 1st, 1970 at UTC).
exp	Specifies the expiration time, as a DateTime object. The format would be Unix time stamp (the number of seconds after Unix Epoch on January 1st, 1970 at UTC).

Deploy the template

You can use Azure portal, Azure PowerShell, Azure CLI, or REST API. To learn about deployment methods, see Deploy templates.

Review deployed resources

You can use the Azure portal to check the key vault and the key. Alternatively, use the following Azure CLI or Azure PowerShell script to list the key created.

CLI

Azure CLI

```
echo "Enter your key vault name:" &&
read keyVaultName &&
az keyvault key list --vault-name $keyVaultName &&
echo "Press [ENTER] to continue ..."
```

Creating key using ARM template is different from creating key via data plane

Creating a key via ARM

- It's only possible to create *new* keys. It isn't possible to update existing keys, nor create new versions of existing keys. If the key already exists, then the existing key is retrieved from storage and used (no write operations will occur).

- To be authorized to use this API, the caller needs to have the "**Microsoft.KeyVault/vaults/keys/write**" role-based access control (RBAC) Action. The built-in "Key Vault Contributor" role is sufficient, since it authorizes all RBAC Actions that match the pattern "Microsoft.KeyVault/*".

Existing API (creating key via data plane)

- It's possible to create new keys, update existing keys, and create new versions of existing keys.
- The caller must be authorized to use this API. If the vault uses access policies, the caller must have "create" key permission; if the vault is enabled for RBAC, the caller must have "Microsoft.KeyVault/vaults/keys/create/action" RBAC DataAction.

Clean up resources

Other Key Vault quickstarts and tutorials build upon this quickstart. If you plan to continue on to work with subsequent quickstarts and tutorials, you may wish to leave these resources in place. When no longer needed, delete the resource group, which deletes the Key Vault and related resources. To delete the resource group by using Azure CLI or Azure PowerShell:

CLI

Azure CLI

```
echo "Enter the Resource Group name:" &&
read resourceGroupName &&
az group delete --name $resourceGroupName &&
echo "Press [ENTER] to continue ..."
```

Next steps

In this quickstart, you created a key vault and a key using an ARM template, and validated the deployment. To learn more about Key Vault and Azure Resource Manager, see these articles.

- Read an Overview of Azure Key Vault
- Learn more about Azure Resource Manager
- Review the Key Vault security overview

Quickstart: Set and retrieve a secret from Azure Key Vault using an ARM template

Article • 04/23/2023

Azure Key Vault is a cloud service that provides a secure store for secrets, such as keys, passwords, certificates, and other secrets. This quickstart focuses on the process of deploying an Azure Resource Manager template (ARM template) to create a key vault and a secret.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

To complete this article:

- If you don't have an Azure subscription, create a free account before you begin.
- Your Azure AD user object ID is needed by the template to configure permissions. The following procedure gets the object ID (GUID).
    1. Run the following Azure PowerShell or Azure CLI command by select **Try it**, and then paste the script into the shell pane. To paste the script, right-click the shell, and then select **Paste**.

CLI

Azure CLI

```
echo "Enter your email address that is used to sign in to
Azure:" &&
```

```
read upn &&
az ad user show --id $upn --query "objectId" &&
echo "Press [ENTER] to continue ..."
```

2. Write down the object ID. You need it in the next section of this quickstart.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.16.2.56959",
      "templateHash": "3049149085309005966"
    }
  },
  "parameters": {
    "keyVaultName": {
      "type": "string",
      "metadata": {
        "description": "Specifies the name of the key vault."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Specifies the Azure location where the key vault should be created."
      }
    },
    "enabledForDeployment": {
      "type": "bool",
      "defaultValue": false,
      "metadata": {
        "description": "Specifies whether Azure Virtual Machines are permitted to retrieve certificates stored as secrets from the key vault."
      }
    },
    "enabledForDiskEncryption": {
      "type": "bool",
      "defaultValue": false,
      "metadata": {
```

```
        "description": "Specifies whether Azure Disk Encryption is permitted
to retrieve secrets from the vault and unwrap keys."
      }
    },
    "enabledForTemplateDeployment": {
      "type": "bool",
      "defaultValue": false,
      "metadata": {
        "description": "Specifies whether Azure Resource Manager is
permitted to retrieve secrets from the key vault."
      }
    },
    "tenantId": {
      "type": "string",
      "defaultValue": "[subscription().tenantId]",
      "metadata": {
        "description": "Specifies the Azure Active Directory tenant ID that
should be used for authenticating requests to the key vault. Get it by using
Get-AzSubscription cmdlet."
      }
    },
    "objectId": {
      "type": "string",
      "metadata": {
        "description": "Specifies the object ID of a user, service principal
or security group in the Azure Active Directory tenant for the vault. The
object ID must be unique for the list of access policies. Get it by using
Get-AzADUser or Get-AzADServicePrincipal cmdlets."
      }
    },
    "keysPermissions": {
      "type": "array",
      "defaultValue": [
        "list"
      ],
      "metadata": {
        "description": "Specifies the permissions to keys in the vault.
Valid values are: all, encrypt, decrypt, wrapKey, unwrapKey, sign, verify,
get, list, create, update, import, delete, backup, restore, recover, and
purge."
      }
    },
    "secretsPermissions": {
      "type": "array",
      "defaultValue": [
        "list"
      ],
      "metadata": {
        "description": "Specifies the permissions to secrets in the vault.
Valid values are: all, get, list, set, delete, backup, restore, recover, and
purge."
      }
    },
    "skuName": {
      "type": "string",
```

```
      "defaultValue": "standard",
      "allowedValues": [
        "standard",
        "premium"
      ],
      "metadata": {
        "description": "Specifies whether the key vault is a standard vault
or a premium vault."
      }
    },
    "secretName": {
      "type": "string",
      "metadata": {
        "description": "Specifies the name of the secret that you want to
create."
      }
    },
    "secretValue": {
      "type": "securestring",
      "metadata": {
        "description": "Specifies the value of the secret that you want to
create."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.KeyVault/vaults",
      "apiVersion": "2021-11-01-preview",
      "name": "[parameters('keyVaultName')]",
      "location": "[parameters('location')]",
      "properties": {
        "enabledForDeployment": "[parameters('enabledForDeployment')]",
        "enabledForDiskEncryption": "
[parameters('enabledForDiskEncryption')]",
        "enabledForTemplateDeployment": "
[parameters('enabledForTemplateDeployment')]",
        "tenantId": "[parameters('tenantId')]",
        "enableSoftDelete": true,
        "softDeleteRetentionInDays": 90,
        "accessPolicies": [
          {
            "objectId": "[parameters('objectId')]",
            "tenantId": "[parameters('tenantId')]",
            "permissions": {
              "keys": "[parameters('keysPermissions')]",
              "secrets": "[parameters('secretsPermissions')]"
            }
          }
        ],
        "sku": {
          "name": "[parameters('skuName')]",
          "family": "A"
        },
        "networkAcls": {
```

```
            "defaultAction": "Allow",
            "bypass": "AzureServices"
          }
        }
      },
      {
        "type": "Microsoft.KeyVault/vaults/secrets",
        "apiVersion": "2021-11-01-preview",
        "name": "[format('{0}/{1}', parameters('keyVaultName'),
parameters('secretName'))]",
        "properties": {
          "value": "[parameters('secretValue')]"
        },
        "dependsOn": [
          "[resourceId('Microsoft.KeyVault/vaults',
parameters('keyVaultName'))]"
        ]
      }
    ]
}
```

Two Azure resources are defined in the template:

- **Microsoft.KeyVault/vaults**: create an Azure key vault.
- **Microsoft.KeyVault/vaults/secrets**: create a key vault secret.

More Azure Key Vault template samples can be found in Azure Quickstart Templates.

Deploy the template

1. Select the following image to sign in to Azure and open a template. The template creates a key vault and a secret.

   Deploy to Azure

2. Select or enter the following values.

Unless it's specified, use the default value to create the key vault and a secret.

- **Subscription**: select an Azure subscription.

- **Resource group**: select **Create new**, enter a unique name for the resource group, and then click **OK**.
- **Location**: select a location. For example, **Central US**.
- **Key Vault Name**: enter a name for the key vault, which must be globally unique within the .vault.azure.net namespace. You need the name in the next section when you validate the deployment.
- **Tenant Id**: the template function automatically retrieves your tenant ID. Don't change the default value.
- **Ad User Id**: enter your Azure AD user object ID that you retrieved from Prerequisites.
- **Secret Name**: enter a name for the secret that you store in the key vault. For example, **adminpassword**.
- **Secret Value**: enter the secret value. If you store a password, it's recommended to use the generated password you created in Prerequisites.
- **I agree to the terms and conditions state above**: Select.

3. Select **Purchase**. After the key vault has been deployed successfully, you get a notification:

The Azure portal is used to deploy the template. In addition to the Azure portal, you can also use the Azure PowerShell, Azure CLI, and REST API. To learn other deployment methods, see Deploy templates.

Review deployed resources

You can either use the Azure portal to check the key vault and the secret, or use the following Azure CLI or Azure PowerShell script to list the secret created.

CLI

Azure CLI

```
echo "Enter your key vault name:" &&
read keyVaultName &&
az keyvault secret list --vault-name $keyVaultName &&
echo "Press [ENTER] to continue ..."
```

The output looks similar to:

CLI

```
[
  {
    "attributes": {
      "created": "2019-09-17T19:10:09+00:00",
      "enabled": true,
      "expires": null,
      "notBefore": null,
      "recoveryLevel": "Purgeable",
      "updated": "2019-09-17T19:10:09+00:00"
    },
    "contentType": null,
    "id": "https://keyvault0917.vault.azure.net/secrets/adminpassword",
    "managed": null,
    "tags": null
  }
]
```

Clean up resources

Other Key Vault quickstarts and tutorials build upon this quickstart. If you plan to continue on to work with subsequent quickstarts and tutorials, you may wish to leave these resources in place. When no longer needed, delete the resource group, which deletes the Key Vault and related resources. To delete the resource group by using Azure CLI or Azure PowerShell:

CLI

Azure CLI

```
echo "Enter the Resource Group name:" &&
read resourceGroupName &&
az group delete --name $resourceGroupName &&
echo "Press [ENTER] to continue ..."
```

Next steps

In this quickstart, you created a key vault and a secret using an ARM template, and validated the deployment. To learn more about Key Vault and Azure Resource Manager, continue on to the articles below.

- Read an Overview of Azure Key Vault
- Learn more about Azure Resource Manager
- Review the Key Vault security overview

Quickstart: Share data using Azure Data Share and ARM template

Article • 03/08/2023

Learn how to set up a new Azure Data Share from an Azure storage account by using an Azure Resource Manager template (ARM template). And, start sharing your data with customers and partners outside of your Azure organization. For a list of the supported data stores, see Supported data stores in Azure Data Share.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "9068130575551523333"
    }
  },
  "parameters": {
```

```
    "projectName": {
      "type": "string",
      "metadata": {
        "description": "Specify a project name that is used to generate
resource names."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Specify the location for the resources."
      }
    },
    "invitationEmail": {
      "type": "string",
      "metadata": {
        "description": "Specify an email address for receiving data share
invitations."
      }
    },
    "syncInterval": {
      "type": "string",
      "defaultValue": "Day",
      "allowedValues": [
        "Day",
        "Hour"
      ],
      "metadata": {
        "description": "Specify snapshot schedule recurrence."
      }
    },
    "syncTime": {
      "type": "string",
      "defaultValue": "[utcNow('yyyy-MM-ddTHH:mm:ssZ')]",
      "metadata": {
        "description": "Specify snapshot schedule start time."
      }
    }
  },
  "variables": {
    "storageAccountName": "[format('{0}store', parameters('projectName'))]",
    "containerName": "[format('{0}container', parameters('projectName'))]",
    "dataShareAccountName": "[format('{0}shareaccount',
parameters('projectName'))]",
    "dataShareName": "[format('{0}share', parameters('projectName'))]",
    "roleAssignmentName": "
[guid(resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName')),
variables('storageBlobDataReaderRoleDefinitionId'),
resourceId('Microsoft.DataShare/accounts',
variables('dataShareAccountName')))]",
    "inviteName": "[format('{0}invite', variables('dataShareName'))]",
    "storageBlobDataReaderRoleDefinitionId": "
[resourceId('Microsoft.Authorization/roleDefinitions', '2a2b9908-6ea1-4ae2-
```

```
8e65-a410df84e7d1')]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-04-01",
      "name": "[variables('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "StorageV2",
      "properties": {
        "accessTier": "Hot"
      }
    },
    {
      "type": "Microsoft.Storage/storageAccounts/blobServices/containers",
      "apiVersion": "2021-04-01",
      "name": "[format('{0}/default/{1}', variables('storageAccountName'),
variables('containerName'))]",
      "dependsOn": [
        "[resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))]"
      ]
    },
    {
      "type": "Microsoft.DataShare/accounts",
      "apiVersion": "2021-08-01",
      "name": "[variables('dataShareAccountName')]",
      "location": "[parameters('location')]",
      "identity": {
        "type": "SystemAssigned"
      },
      "properties": {}
    },
    {
      "type": "Microsoft.DataShare/accounts/shares",
      "apiVersion": "2021-08-01",
      "name": "[format('{0}/{1}', variables('dataShareAccountName'),
variables('dataShareName'))]",
      "properties": {
        "shareKind": "CopyBased"
      },
      "dependsOn": [
        "[resourceId('Microsoft.DataShare/accounts',
variables('dataShareAccountName'))]"
      ]
    },
    {
      "type": "Microsoft.Authorization/roleAssignments",
      "apiVersion": "2020-04-01-preview",
      "scope": "[format('Microsoft.Storage/storageAccounts/{0}',
variables('storageAccountName'))]",
      "name": "[variables('roleAssignmentName')]",
```

```
      "properties": {
        "roleDefinitionId": "
[variables('storageBlobDataReaderRoleDefinitionId')]",
        "principalId": "
[reference(resourceId('Microsoft.DataShare/accounts',
variables('dataShareAccountName')), '2021-08-01',
'full').identity.principalId]",
        "principalType": "ServicePrincipal"
      },
      "dependsOn": [
        "[resourceId('Microsoft.DataShare/accounts',
variables('dataShareAccountName'))]",
        "[resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))]"
      ]
    },
    {
      "type": "Microsoft.DataShare/accounts/shares/dataSets",
      "apiVersion": "2021-08-01",
      "name": "[format('{0}/{1}/{2}', variables('dataShareAccountName'),
variables('dataShareName'), variables('containerName'))]",
      "kind": "Container",
      "properties": {
        "subscriptionId": "[subscription().subscriptionId]",
        "resourceGroup": "[resourceGroup().name]",
        "storageAccountName": "[variables('storageAccountName')]",
        "containerName": "[variables('containerName')]"
      },
      "dependsOn": [
        "
[resourceId('Microsoft.Storage/storageAccounts/blobServices/containers',
split(format('{0}/default/{1}', variables('storageAccountName'),
variables('containerName')), '/')[0], split(format('{0}/default/{1}',
variables('storageAccountName'), variables('containerName')), '/')[1],
split(format('{0}/default/{1}', variables('storageAccountName'),
variables('containerName')), '/')[2])]",
        "[resourceId('Microsoft.DataShare/accounts/shares',
variables('dataShareAccountName'), variables('dataShareName'))]",
        "[resourceId('Microsoft.DataShare/accounts/shares/invitations',
variables('dataShareAccountName'), variables('dataShareName'),
variables('inviteName'))]",
        "[resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))]",
        "
[resourceId('Microsoft.DataShare/accounts/shares/synchronizationSettings',
variables('dataShareAccountName'), variables('dataShareName'),
format('{0}_synchronizationSetting', variables('dataShareName')))]"
      ]
    },
    {
      "type": "Microsoft.DataShare/accounts/shares/invitations",
      "apiVersion": "2021-08-01",
      "name": "[format('{0}/{1}/{2}', variables('dataShareAccountName'),
variables('dataShareName'), variables('inviteName'))]",
      "properties": {
```

```
        "targetEmail": "[parameters('invitationEmail')]"
      },
      "dependsOn": [
        "[resourceId('Microsoft.DataShare/accounts/shares',
variables('dataShareAccountName'), variables('dataShareName'))]"
      ]
    },
    {
      "type": "Microsoft.DataShare/accounts/shares/synchronizationSettings",
      "apiVersion": "2021-08-01",
      "name": "[format('{0}/{1}/{2}', variables('dataShareAccountName'),
variables('dataShareName'), format('{0}_synchronizationSetting',
variables('dataShareName')))]",
      "kind": "ScheduleBased",
      "properties": {
        "recurrenceInterval": "[parameters('syncInterval')]",
        "synchronizationTime": "[parameters('syncTime')]"
      },
      "dependsOn": [
        "[resourceId('Microsoft.DataShare/accounts/shares',
variables('dataShareAccountName'), variables('dataShareName'))]"
      ]
    }
  ]
}
```

The following resources are defined in the template:

- Microsoft.Storage/storageAccounts:
- Microsoft.Storage/storageAccounts/blobServices/containers
- Microsoft.DataShare/accounts
- Microsoft.DataShare/accounts/shares
- Microsoft.Storage/storageAccounts/providers/roleAssignments
- Microsoft.DataShare/accounts/shares/dataSets
- Microsoft.DataShare/accounts/shares/invitations
- Microsoft.DataShare/accounts/shares/synchronizationSettings

The template performs the following tasks:

- Create a storage account and a container used as the Data share data source.
- Create a Data share account and a Data share.
- Create a role assignment to grant the Storage Blob Data Reader role to the source data share resource. See Roles and requirements for Azure Data Share.
- Add a dataset to the Date share.
- Add recipients to the Data share.
- Enable a snapshot schedule for the Data share.

This template is created for learning purposes. In practice, you usually have some data in an existing storage account. You would need to create the role assignment before running a template or a script to create the dataset. Sometimes, you might get the following error message when you deploy the template:

```plaintext
"Missing permissions for DataShareAcccount on resource
'subscriptions/<SUBSCRIPTION ID>/resourceGroups/<RESOURCE GROUP
NAME>/providers/Microsoft.Storage/storageAccounts/<STORAGE ACCOUNT NAME>'
(Code: 5006)"
```

It's because the deployment is trying to create the dataset before the Azure role assignment gets finalized. Despite the error message, the deployment could be successful. You would still be able to walk through Review deployed resources.

Deploy the template

1. Select the following image to sign in to Azure and open the template.

   Deploy to Azure

2. Select or enter the following values:

   - **Subscription**: select an Azure subscription used to create the data share and the other resources.
   - **Resource group**: select **Create new** to create a new resource group or select an existing resource group.
   - **Location**: select a location for the resource group.
   - **Project Name**: enter a project name. The project name is used for generating resource names. See the variable definitions in the previous template.
   - **location**: select a location for the resources. You can use the same location for the resource group.
   - **Invitation Email**: enter the data share recipient's Azure login email address. Email alias doesn't work.

   Use the default value for the rest of the settings.

3. select **I agree to the terms and conditions stated above**, and the select **Purchase**.

Review deployed resources

1. Sign in to the Azure portal.

2. Open the data share account that you created.

3. From the left menu, select **Send Shares**. You shall see the storage account listed.

4. Select the storage account. Under **Details**, you shall see the synchronization setting as you configured in the template.

5. Select **Invitations** from the top. You shall see the email address that you specified when you deploy the template. The **Status** shall be **Pending**.

Clean up resources

When no longer needed, delete the resource group, which deletes the resources in the resource group.

Azure PowerShell

```
$resourceGroupName = Read-Host -Prompt "Enter the resource group name"
Remove-AzResourceGroup -Name $resourceGroupName
Write-Host "Press [ENTER] to continue..."
```

Next steps

In this quickstart, you learnt how to create an Azure data share and invite recipients. To learn more about how a data consumer can accept and receive a data share, continue to the accept and receive data tutorial.

Create a storage account

Article • 05/02/2023

An Azure storage account contains all of your Azure Storage data objects: blobs, files, queues, and tables. The storage account provides a unique namespace for your Azure Storage data that is accessible from anywhere in the world over HTTP or HTTPS. For more information about Azure storage accounts, see Storage account overview.

In this how-to article, you learn to create a storage account using the Azure portal, Azure PowerShell, Azure CLI, or an Azure Resource Manager template.

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Portal

None.

Next, sign in to Azure.

Portal

Sign in to the Azure portal.

Create a storage account

A storage account is an Azure Resource Manager resource. Resource Manager is the deployment and management service for Azure. For more information, see Azure Resource Manager overview.

Every Resource Manager resource, including an Azure storage account, must belong to an Azure resource group. A resource group is a logical container for grouping your Azure services. When you create a storage account, you have the option to either create a new resource group, or use an existing resource group. This how-to shows how to create a new resource group.

Portal

To create an Azure storage account with the Azure portal, follow these steps:

1. From the left portal menu, select **Storage accounts** to display a list of your storage accounts. If the portal menu isn't visible, click the menu button to toggle it on.

2. On the **Storage accounts** page, select **Create**.

Options for your new storage account are organized into tabs in the **Create a storage account** page. The following sections describe each of the tabs and their options.

Basics tab

On the **Basics** tab, provide the essential information for your storage account. After you complete the **Basics** tab, you can choose to further customize your new storage account by setting options on the other tabs, or you can select **Review + create** to accept the default options and proceed to validate and create the account.

The following table describes the fields on the **Basics** tab.

Section	Field	Required or optional	Description
Project details	Subscription	Required	Select the subscription for the new storage account.
Project details	Resource group	Required	Create a new resource group for this storage account, or select an existing one. For more information, see Resource groups.
Instance details	Storage account name	Required	Choose a unique name for your storage account. Storage account names must be between 3 and 24 characters in length and may contain numbers and lowercase letters only.
Instance details	Region	Required	Select the appropriate region for your storage account. For more information, see Regions and Availability Zones in Azure. Not all regions are supported for all types of storage accounts or redundancy configurations. For more information, see Azure Storage redundancy. The choice of region can have a billing impact. For more information, see Storage account billing. If your subscription supports Azure public multi-access edge zones (Azure MEC), you can deploy your storage account to an edge zone. For more information about edge zones, see What is Azure public MEC?.

Section	Field	Required or optional	Description
Instance details	Performance	Required	Select **Standard** performance for general-purpose v2 storage accounts (default). This type of account is recommended by Microsoft for most scenarios. For more information, see Types of storage accounts. Select **Premium** for scenarios requiring low latency. After selecting **Premium**, select the type of premium storage account to create. The following types of premium storage accounts are available: • Block blobs • File shares • Page blobs Microsoft recommends creating a general-purpose v2, premium block blob, or premium file share account for most scenarios. To select a legacy account type, use the link provided beneath **Instance details**. For more information about legacy account types, see Legacy storage account types.
Instance details	Redundancy	Required	Select your desired redundancy configuration. Not all redundancy options are available for all types of storage accounts in all regions. For more information about redundancy configurations, see Azure Storage redundancy. If you select a geo-redundant configuration (GRS or GZRS), your data is replicated to a data center in a different region. For read access to data in the secondary region, select **Make read access to data available in the event of regional unavailability**.

The following image shows a standard configuration of the basic properties for a new storage account.

Advanced tab

On the **Advanced** tab, you can configure additional options and modify default settings for your new storage account. Some of these options can also be configured after the storage account is created, while others must be configured at the time of creation.

The following table describes the fields on the **Advanced** tab.

Section	Field	Required or optional	Description

Section	Field	Required or optional	Description
Security	Require secure transfer for REST API operations	Optional	Require secure transfer to ensure that incoming requests to this storage account are made only via HTTPS (default). Recommended for optimal security. For more information, see Require secure transfer to ensure secure connections.
Security	Allow enabling public access on containers	Optional	When enabled, this setting allows a user with the appropriate permissions to enable anonymous public access to a container in the storage account (default). Disabling this setting prevents all anonymous public access to the storage account. For more information, see Prevent anonymous public read access to containers and blobs. Enabling blob public access does not make blob data available for public access unless the user takes the additional step to explicitly configure the container's public access setting.
Security	Enable storage account key access	Optional	When enabled, this setting allows clients to authorize requests to the storage account using either the account access keys or an Azure Active Directory (Azure AD) account (default). Disabling this setting prevents authorization with the account access keys. For more information, see Prevent Shared Key authorization for an Azure Storage account.
Security	Default to Azure Active Directory authorization in the Azure portal	Optional	When enabled, the Azure portal authorizes data operations with the user's Azure AD credentials by default. If the user does not have the appropriate permissions assigned via Azure role-based access control (Azure RBAC) to perform data operations, then the portal will use the account access keys for data access instead. The user can also choose to switch to using the account access keys. For more information, see Default to Azure AD authorization in the Azure portal.
Security	Minimum TLS version	Required	Select the minimum version of Transport Layer Security (TLS) for incoming requests to the storage account. The default value is TLS version 1.2. When set to the default value, incoming requests made using TLS 1.0 or TLS 1.1 are rejected. For more information, see Enforce a minimum required version of Transport Layer Security (TLS) for requests to a storage account.

Section	Field	Required or optional	Description
Security	Permitted scope for copy operations (preview)	Required	Select the scope of storage accounts from which data can be copied to the new account. The default value is `From any storage account`. When set to the default value, users with the appropriate permissions can copy data from any storage account to the new account. Select `From storage accounts in the same Azure AD tenant` to only allow copy operations from storage accounts within the same Azure AD tenant. Select `From storage accounts that have a private endpoint to the same virtual network` to only allow copy operations from storage accounts with private endpoints on the same virtual network. For more information, see Restrict the source of copy operations to a storage account.
Data Lake Storage Gen2	Enable hierarchical namespace	Optional	To use this storage account for Azure Data Lake Storage Gen2 workloads, configure a hierarchical namespace. For more information, see Introduction to Azure Data Lake Storage Gen2.
Blob storage	Enable SFTP	Optional	Enable the use of Secure File Transfer Protocol (SFTP) to securely transfer of data over the internet. For more information, see Secure File Transfer (SFTP) protocol support in Azure Blob Storage.
Blob storage	Enable network file system (NFS) v3	Optional	NFS v3 provides Linux file system compatibility at object storage scale enables Linux clients to mount a container in Blob storage from an Azure Virtual Machine (VM) or a computer on-premises. For more information, see Network File System (NFS) 3.0 protocol support in Azure Blob Storage.
Blob storage	Allow cross-tenant replication	Required	By default, users with appropriate permissions can configure object replication across Azure AD tenants. To prevent replication across tenants, deselect this option. For more information, see Prevent replication across Azure AD tenants.

Section	Field	Required or optional	Description
Blob storage	Access tier	Required	Blob access tiers enable you to store blob data in the most cost-effective manner, based on usage. Select the hot tier (default) for frequently accessed data. Select the cool tier for infrequently accessed data. For more information, see Hot, Cool, and Archive access tiers for blob data.
Azure Files	Enable large file shares	Optional	Available only for standard file shares with the LRS or ZRS redundancies.

The following image shows a standard configuration of the advanced properties for a new storage account.

Home > Resource groups > SampleRG > Marketplace > Storage account >

Create a storage account ...

Basics | **Advanced** | Networking | Data protection | Encryption | Tags | Review

Security

Configure security settings that impact your storage account.

Require secure transfer for REST API operations ☑

Allow enabling public access on containers ☑

Enable storage account key access ☑

Default to Azure Active Directory authorization in the Azure portal ☐

Minimum TLS version: Version 1.2

Permitted scope for copy operations (preview): From any storage account

Data Lake Storage Gen2

The Data Lake Storage Gen2 hierarchical namespace accelerates big data analytics workloads and enables file-level access control lists (ACLs). Learn more

Enable hierarchical namespace ☐

Blob storage

Enable SFTP ☐

To enable SFTP, 'hierarchical namespace' must be enabled.

Enable network file system v3 ☐

To enable NFS v3 'hierarchical namespace' must be enabled. Learn more about NFS v3

Allow cross-tenant replication ☑

Access tier

- (•) Hot: Frequently accessed data and day-to-day usage scenarios
- ( ) Cool: Infrequently accessed data and backup scenarios

Azure Files

Enable large file shares ☐

Review | < Previous | Next : Networking >

Networking tab

On the **Networking** tab, you can configure network connectivity and routing preference settings for your new storage account. These options can also be configured after the storage account is created.

The following table describes the fields on the **Networking** tab.

Section	Field	Required or optional	Description
Network connectivity	Network access	Required	By default, incoming network traffic is routed to the public endpoint for your storage account. You can specify that traffic must be routed to the public endpoint through an Azure virtual network. You can also configure private endpoints for your storage account. For more information, see Use private endpoints for Azure Storage.
Network connectivity	Endpoint type	Required	Azure Storage supports two types of endpoints: standard endpoints (the default) and Azure DNS zone endpoints (preview). Within a given subscription, you can create up to 250[1] accounts with standard endpoints per region, and up to 5000 accounts with Azure DNS zone endpoints per region, for a total of 5250 storage accounts. To register for the preview, see About the preview.
Network routing	Routing preference	Required	The network routing preference specifies how network traffic is routed to the public endpoint of your storage account from clients over the internet. By default, a new storage account uses Microsoft network routing. You can also choose to route network traffic through the POP closest to the storage account, which may lower networking costs. For more information, see Network routing preference for Azure Storage.

[1]With a quota increase, you can create up to 500 storage accounts with standard endpoints per region in a given subscription, for a total of 5500 storage accounts per region. For more information, see Increase Azure Storage account quotas.

The following image shows a standard configuration of the networking properties for a new storage account.

> **Important**
>
> Azure DNS zone endpoints are currently in PREVIEW. See the **Supplemental Terms of Use for Microsoft Azure Previews** for legal terms that apply to Azure features that are in beta, preview, or otherwise not yet released into general availability.

Data protection tab

On the **Data protection** tab, you can configure data protection options for blob data in your new storage account. These options can also be configured after the storage account is created. For an overview of data protection options in Azure Storage, see Data protection overview.

The following table describes the fields on the **Data protection** tab.

Section	Field	Required or optional	Description

Section	Field	Required or optional	Description
Recovery	Enable point-in-time restore for containers	Optional	Point-in-time restore provides protection against accidental deletion or corruption by enabling you to restore block blob data to an earlier state. For more information, see Point-in-time restore for block blobs. Enabling point-in-time restore also enables blob versioning, blob soft delete, and blob change feed. These prerequisite features may have a cost impact. For more information, see Pricing and billing for point-in-time restore.
Recovery	Enable soft delete for blobs	Optional	Blob soft delete protects an individual blob, snapshot, or version from accidental deletes or overwrites by maintaining the deleted data in the system for a specified retention period. During the retention period, you can restore a soft-deleted object to its state at the time it was deleted. For more information, see Soft delete for blobs. Microsoft recommends enabling blob soft delete for your storage accounts and setting a minimum retention period of seven days.
Recovery	Enable soft delete for containers	Optional	Container soft delete protects a container and its contents from accidental deletes by maintaining the deleted data in the system for a specified retention period. During the retention period, you can restore a soft-deleted container to its state at the time it was deleted. For more information, see Soft delete for containers. Microsoft recommends enabling container soft delete for your storage accounts and setting a minimum retention period of seven days.

Section	Field	Required or optional	Description
Recovery	Enable soft delete for file shares	Optional	Soft delete for file shares protects a file share and its contents from accidental deletes by maintaining the deleted data in the system for a specified retention period. During the retention period, you can restore a soft-deleted file share to its state at the time it was deleted. For more information, see Prevent accidental deletion of Azure file shares. Microsoft recommends enabling soft delete for file shares for Azure Files workloads and setting a minimum retention period of seven days.
Tracking	Enable versioning for blobs	Optional	Blob versioning automatically saves the state of a blob in a previous version when the blob is overwritten. For more information, see Blob versioning. Microsoft recommends enabling blob versioning for optimal data protection for the storage account.
Tracking	Enable blob change feed	Optional	The blob change feed provides transaction logs of all changes to all blobs in your storage account, as well as to their metadata. For more information, see Change feed support in Azure Blob Storage.
Access control	Enable version-level immutability support	Optional	Enable support for immutability policies that are scoped to the blob version. If this option is selected, then after you create the storage account, you can configure a default time-based retention policy for the account or for the container, which blob versions within the account or container will inherit by default. For more information, see Enable version-level immutability support on a storage account.

The following image shows a standard configuration of the data protection properties for a new storage account.

Encryption tab

On the **Encryption** tab, you can configure options that relate to how your data is encrypted when it is persisted to the cloud. Some of these options can be configured only when you create the storage account.

Field	Required or optional	Description

Field	Required or optional	Description
Encryption type	Required	By default, data in the storage account is encrypted by using Microsoft-managed keys. You can rely on Microsoft-managed keys for the encryption of your data, or you can manage encryption with your own keys. For more information, see Azure Storage encryption for data at rest.
Enable support for customer-managed keys	Required	By default, customer managed keys can be used to encrypt only blobs and files. Set this option to **All service types (blobs, files, tables, and queues)** to enable support for customer-managed keys for all services. You are not required to use customer-managed keys if you choose this option. For more information, see Customer-managed keys for Azure Storage encryption.
Encryption key	Required if **Encryption type** field is set to **Customer-managed keys**.	If you choose **Select a key vault and key**, you are presented with the option to navigate to the key vault and key that you wish to use. If you choose **Enter key from URI**, then you are presented with a field to enter the key URI and the subscription.
User-assigned identity	Required if **Encryption type** field is set to **Customer-managed keys**.	If you are configuring customer-managed keys at create time for the storage account, you must provide a user-assigned identity to use for authorizing access to the key vault.
Enable infrastructure encryption	Optional	By default, infrastructure encryption is not enabled. Enable infrastructure encryption to encrypt your data at both the service level and the infrastructure level. For more information, see Create a storage account with infrastructure encryption enabled for double encryption of data.

The following image shows a standard configuration of the encryption properties for a new storage account.

Tags tab

On the **Tags** tab, you can specify Resource Manager tags to help organize your Azure resources. For more information, see Tag resources, resource groups, and subscriptions for logical organization.

The following image shows a standard configuration of the index tag properties for a new storage account.

Review + create tab

When you navigate to the **Review + create** tab, Azure runs validation on the storage account settings that you have chosen. If validation passes, you can proceed to create the storage account.

If validation fails, then the portal indicates which settings need to be modified.

The following image shows the **Review** tab data prior to the creation of a new storage account.

Delete a storage account

Deleting a storage account deletes the entire account, including all data in the account. Be sure to back up any data you want to save before you delete the account.

Under certain circumstances, a deleted storage account may be recovered, but recovery is not guaranteed. For more information, see Recover a deleted storage account.

If you try to delete a storage account associated with an Azure virtual machine, you may get an error about the storage account still being in use. For help troubleshooting this error, see Troubleshoot errors when you delete storage accounts.

Portal

1. Navigate to the storage account in the Azure portal.
2. Click **Delete**.

Alternately, you can delete the resource group, which deletes the storage account and any other resources in that resource group. For more information about deleting a resource group, see Delete resource group and resources.

Next steps

- Storage account overview
- Upgrade to a general-purpose v2 storage account
- Move a storage account to another region
- Recover a deleted storage account
- Migrate a classic storage account

Quickstart: Create a new Azure API Management service instance using an ARM template

Article • 04/09/2023

This quickstart describes how to use an Azure Resource Manager template (ARM template) to create an Azure API Management (APIM) service instance. APIM helps organizations publish APIs to external, partner, and internal developers to unlock the potential of their data and services. API Management provides the core competencies to ensure a successful API program through developer engagement, business insights, analytics, security, and protection. APIM enables you to create and manage modern API gateways for existing backend services hosted anywhere. For more information, see the Overview.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
```

```
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "6751554658084889632"
    }
  },
  "parameters": {
    "apiManagementServiceName": {
      "type": "string",
      "defaultValue": "[format('apiservice{0}',
uniqueString(resourceGroup().id))]",
      "metadata": {
        "description": "The name of the API Management service instance"
      }
    },
    "publisherEmail": {
      "type": "string",
      "minLength": 1,
      "metadata": {
        "description": "The email address of the owner of the service"
      }
    },
    "publisherName": {
      "type": "string",
      "minLength": 1,
      "metadata": {
        "description": "The name of the owner of the service"
      }
    },
    "sku": {
      "type": "string",
      "defaultValue": "Developer",
      "allowedValues": [
        "Developer",
        "Standard",
        "Premium"
      ],
      "metadata": {
        "description": "The pricing tier of this API Management service"
      }
    },
    "skuCount": {
      "type": "int",
      "defaultValue": 1,
      "allowedValues": [
        1,
        2
      ],
      "metadata": {
        "description": "The instance size of this API Management service."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
```

```
      "metadata": {
        "description": "Location for all resources."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.ApiManagement/service",
      "apiVersion": "2021-08-01",
      "name": "[parameters('apiManagementServiceName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('sku')]",
        "capacity": "[parameters('skuCount')]"
      },
      "properties": {
        "publisherEmail": "[parameters('publisherEmail')]",
        "publisherName": "[parameters('publisherName')]"
      }
    }
  ]
}
```

The following resource is defined in the template:

- **Microsoft.ApiManagement/service**

More Azure API Management template samples can be found in Azure Quickstart Templates.

Deploy the template

1. Select the following image to sign in to Azure and open a template. The template creates an API Management service instance with an automatically generated name.

   Deploy to Azure

   In this example, the instance is configured in the Developer tier, an economical option to evaluate Azure API Management. This tier isn't for production use. For more information about scaling the API Management tiers, see upgrade and scale.

2. Select or enter the following values.

   - **Subscription**: select an Azure subscription.
   - **Resource group**: select **Create new**, enter a unique name for the resource group, and then select **OK**.

- **Region**: select a location for the resource group. Example: **Central US**.
- **Publisher Email**: enter an email address to receive notifications.
- **Publisher Name**: enter a name you choose for the API publisher.
- **Sku**: accept the default value of **Developer**.
- **Sku Count**: accept the default value.
- **Location**: accept the generated location for the API Management service.

3. Select **Review + Create**, then review the terms and conditions. If you agree, select **Create**.

> **Tip**
>
> It can take between 30 and 40 minutes to create and activate an API Management service in the Developer tier.

4. After the instance has been created successfully, you get a notification:

The Azure portal is used to deploy the template. In addition to the Azure portal, you can also use the Azure PowerShell, Azure CLI, and REST API. To learn other deployment methods, see Deploy templates.

Review deployed resources

Use the Azure portal to check the deployed resources, or use tools such as the Azure CLI or Azure PowerShell to list the deployed resources.

1. In the Azure portal, search for and select **API Management services**, and select the service instance you created.
2. Review the properties of your service on the **Overview** page.

When your API Management service instance is online, you're ready to use it. Start with the tutorial to import and publish your first API.

Clean up resources

If you plan to continue working with subsequent tutorials, you might want to leave the API Management instance in place. When no longer needed, delete the resource group, which deletes the resources in the resource group.

1. In the Azure portal, search for and select **Resource groups**. You can also select **Resource groups** on the **Home** page.

2. On the **Resource groups** page, select your resource group.

3. On the resource group page, select **Delete resource group**.

4. Type the name of your resource group, and then select **Delete**.

Next steps

Tutorial: Import and publish your first API

Quickstart: Create App Service app using an ARM template

Article • 04/09/2023

Get started with Azure App Service by deploying an app to the cloud using an Azure Resource Manager template (ARM template) and Azure CLI in Cloud Shell. Because you use a free App Service tier, you incur no costs to complete this quickstart.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Use the following button to deploy on **Linux**:

Deploy to Azure

Use the following button to deploy on **Windows**:

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create an Azure free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates. It deploys an App Service plan and an App Service app on Linux. It's compatible with all supported programming languages on App Service.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
```

```
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "10602523904429381366"
    }
  },
  "parameters": {
    "webAppName": {
      "type": "string",
      "defaultValue": "[format('webApp-{0}',
uniqueString(resourceGroup().id))]",
      "minLength": 2,
      "metadata": {
        "description": "Web app name."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "sku": {
      "type": "string",
      "defaultValue": "F1",
      "metadata": {
        "description": "The SKU of App Service Plan."
      }
    },
    "linuxFxVersion": {
      "type": "string",
      "defaultValue": "DOTNETCORE|3.0",
      "metadata": {
        "description": "The Runtime stack of current web app"
      }
    },
    "repoUrl": {
      "type": "string",
      "defaultValue": " ",
      "metadata": {
        "description": "Optional Git Repo URL"
      }
    }
  },
  "variables": {
    "appServicePlanPortalName": "[format('AppServicePlan-{0}',
parameters('webAppName'))]"
  },
  "resources": [
    {
      "type": "Microsoft.Web/serverfarms",
```

```
      "apiVersion": "2021-02-01",
      "name": "[variables('appServicePlanPortalName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('sku')]"
      },
      "kind": "linux",
      "properties": {
        "reserved": true
      }
    },
    {
      "type": "Microsoft.Web/sites",
      "apiVersion": "2021-02-01",
      "name": "[parameters('webAppName')]",
      "location": "[parameters('location')]",
      "properties": {
        "httpsOnly": true,
        "serverFarmId": "[resourceId('Microsoft.Web/serverfarms',
variables('appServicePlanPortalName'))]",
        "siteConfig": {
          "linuxFxVersion": "[parameters('linuxFxVersion')]",
          "minTlsVersion": "1.2",
          "ftpsState": "FtpsOnly"
        }
      },
      "identity": {
        "type": "SystemAssigned"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Web/serverfarms',
variables('appServicePlanPortalName'))]"
      ]
    },
    {
      "condition": "[contains(parameters('repoUrl'), 'http')]",
      "type": "Microsoft.Web/sites/sourcecontrols",
      "apiVersion": "2021-02-01",
      "name": "[format('{0}/{1}', parameters('webAppName'), 'web')]",
      "properties": {
        "repoUrl": "[parameters('repoUrl')]",
        "branch": "master",
        "isManualIntegration": true
      },
      "dependsOn": [
        "[resourceId('Microsoft.Web/sites', parameters('webAppName'))]"
      ]
    }
  ]
}
```

Two Azure resources are defined in the template:

- Microsoft.Web/serverfarms: create an App Service plan.
- Microsoft.Web/sites: create an App Service app.

This template contains several parameters that are predefined for your convenience. See the table below for parameter defaults and their descriptions:

Parameters	Type	Default value	Description
webAppName	string	"webApp-<uniqueString>"	App name
location	string	"[resourceGroup().location]"	App region
sku	string	"F1"	Instance size (F1 = Free Tier)
linuxFxVersion	string	"DOTNETCORE|3.0	"Programming language stack | Version"
repoUrl	string	" "	External Git repo (optional)

Deploy the template

Azure CLI is used here to deploy the template. You can also use the Azure portal, Azure PowerShell, and REST API. To learn other deployment methods, see Deploy templates.

The following code creates a resource group, an App Service plan, and a web app. A default resource group, App Service plan, and location have been set for you. Replace `<app-name>` with a globally unique app name (valid characters are `a-z`, `0-9`, and `-`).

Run the code below to create a Python app on Linux.

Azure CLI

```
az group create --name myResourceGroup --location "southcentralus" &&
az deployment group create --resource-group myResourceGroup --parameters
webAppName="<app-name>" linuxFxVersion="PYTHON|3.7" \
--template-uri "https://raw.githubusercontent.com/Azure/azure-quickstart-
templates/master/quickstarts/microsoft.web/app-service-docs-
linux/azuredeploy.json"
```

To deploy a different language stack, update `linuxFxVersion` with appropriate values. Samples are shown below. To show current versions, run the following command in the Cloud Shell: `az webapp config show --resource-group myResourceGroup --name <app-name> --query linuxFxVersion`

Language	Example

Language	Example
.NET	linuxFxVersion="DOTNETCORE|3.0"
PHP	linuxFxVersion="PHP|7.4"
Node.js	linuxFxVersion="NODE|10.15"
Java	linuxFxVersion="JAVA|1.8 |TOMCAT|9.0"
Python	linuxFxVersion="PYTHON|3.7"
Ruby	linuxFxVersion="RUBY|2.6"

> **Note**
>
> You can find more Azure App Service template samples here.

Validate the deployment

Browse to `http://<app_name>.azurewebsites.net/` and verify it's been created.

Clean up resources

When no longer needed, delete the resource group.

Next steps

- Deploy from local Git
- ASP.NET Core with SQL Database
- Python with Postgres
- PHP with MySQL
- Connect to Azure SQL database with Java
- Secure with custom domain and certificate

Quickstart: Create a notification hub using a Resource Manager template

Article • 03/10/2023

Azure Notification Hubs provides an easy-to-use and scaled-out push engine that enables you to send notifications to any platform (iOS, Android, Windows, Kindle, etc.) from any backend (cloud or on-premises). For more information about the service, see What is Azure Notification Hubs.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

This quickstart uses an Azure Resource Manager template to create an Azure Notification Hubs namespace, and a notification hub named **MyHub** within that namespace.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.8.9.13224",
```

```
      "templateHash": "2713724900359552876"
    }
  },
  "parameters": {
    "namespaceName": {
      "type": "string",
      "metadata": {
        "description": "The name of the Notification Hubs namespace."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "The location in which the Notification Hubs
resources should be deployed."
      }
    }
  },
  "variables": {
    "hubName": "MyHub"
  },
  "resources": [
    {
      "type": "Microsoft.NotificationHubs/namespaces",
      "apiVersion": "2017-04-01",
      "name": "[parameters('namespaceName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Free"
      }
    },
    {
      "type": "Microsoft.NotificationHubs/namespaces/notificationHubs",
      "apiVersion": "2017-04-01",
      "name": "[format('{0}/{1}', parameters('namespaceName'),
variables('hubName'))]",
      "location": "[parameters('location')]",
      "properties": {},
      "dependsOn": [
        "[resourceId('Microsoft.NotificationHubs/namespaces',
parameters('namespaceName'))]"
      ]
    }
  ]
}
```

- Microsoft.NotificationHubs/namespaces
- Microsoft.NotificationHubs/namespaces/notificationHubs

Deploy the template

Select the following image to sign in to Azure and open a template. The template takes a Notification Hubs namespace name as a parameter. The template then creates a namespace with that name and a notification hub named **MyHub** within that namespace.

Deploy to Azure

Review deployed resources

You can either use the Azure portal to check the deployed resources, or use Azure CLI or Azure PowerShell script to list the deployed Notification Hubs namespace and hub:

PowerShell

Azure PowerShell

```
Get-AzNotificationHub -Namespace "nhtestns123" -ResourceGroup
"ContosoNotificationsGroup"
Get-AzNotificationHubsNamespace -Namespace "nhtestns123"
```

The output looks similar to:

```
Id                  : /subscriptions/subscription ID here
                      165/resourceGroups/myRG1/providers/Microsoft.Not
                      ificationHubs/namespaces/nhtestns123/notificatio
                      nHubs/myhub
Type                : Microsoft.NotificationHubs/namespaces/notificati
                      onHubs
Location            : East US
Tags                :
AdmCredential       :
ApnsCredential      :
AuthorizationRules  :
BaiduCredential     :
GcmCredential       :
MpnsCredential      :
Name                : myhub
RegistrationTtl     : RegistrationTtl here
WnsCredential       :
```

Clean up resources

When no longer needed, delete the resource group, which deletes the resources in the resource group.

Azure PowerShell

```
$resourceGroupName = Read-Host -Prompt "Enter the resource group name"
Remove-AzResourceGroup -Name $resourceGroupName
Write-Host "Press [ENTER] to continue..."
```

Next steps

For a step-by-step tutorial that guides you through the process of creating a template, see:

Tutorial: Create and deploy your first ARM template

Create a Web App plus Azure Cache for Redis using a template

Article • 03/09/2023

> **Note**
>
> We recommend that you use the Azure Az PowerShell module to interact with Azure. See **Install Azure PowerShell** to get started. To learn how to migrate to the Az PowerShell module, see **Migrate Azure PowerShell from AzureRM to Az**.

In this article, you learn how to create an Azure Resource Manager template that deploys an Azure Web App with Azure Cache for Redis. You'll learn the following deployment details:

- How to define which resources are deployed
- How to define parameters that are specified when the deployment is executed

You can use this template for your own deployments, or customize it to meet your requirements.

For more information about creating templates, see Authoring Azure Resource Manager templates. To learn about the JSON syntax and properties for cache resource types, see Microsoft.Cache resource types.

For the complete template, see Web App with Azure Cache for Redis template.

What you will deploy

In this template, you deploy:

- Azure Web App
- Azure Cache for Redis

To run the deployment automatically, select the following button:

Deploy to Azure

Parameters to specify

With Azure Resource Manager, you define parameters for values you want to specify when the template is deployed. The template includes a section called Parameters that contains all of the parameter values. You should define a parameter for those values that vary based on the project you're deploying or the environment you're deploying to. Do not define parameters for values that are constant. Each parameter value is used in the template to define the resources that are deployed.

When defining parameters, use the **allowedValues** field to specify which values a user can provide during deployment. Use the **defaultValue** field to assign a value to the parameter, if no value is provided during deployment.

We will describe each parameter in the template.

siteName

The name of the web app that you wish to create.

```
config

"siteName":{
  "type":"string"
}
```

hostingPlanName

The name of the App Service plan to use for hosting the web app.

```
config

"hostingPlanName":{
  "type":"string"
}
```

sku

The pricing tier for the hosting plan.

```
JSON

"sku": {
  "type": "string",
  "allowedValues": [
    "F1",
    "D1",
```

```
    "B1",
    "B2",
    "B3",
    "S1",
    "S2",
    "S3",
    "P1",
    "P2",
    "P3",
    "P4"
  ],
  "defaultValue": "S1",
  "metadata": {
    "description": "The pricing tier for the hosting plan."
  }
}
```

The template defines the values that are permitted for this parameter, and assigns a default value of `S1` if no value is specified.

workerSize

The instance size of the hosting plan (small, medium, or large).

JSON

```
"workerSize":{
  "type":"string",
  "allowedValues":[
    "0",
    "1",
    "2"
  ],
  "defaultValue":"0"
}
```

The template defines the values that are permitted for this parameter (`0`, `1`, or `2`), and assigns a default value of `0` if no value is specified. The values correspond to small, medium, and large.

cacheSKUName

The pricing tier of the new Azure Cache for Redis.

JSON

```
    "cacheSKUName": {
      "type": "string",
      "allowedValues": [
        "Basic",
        "Standard",
        "Premium"
      ],
      "defaultValue": "Basic",
      "metadata": {
        "description": "The pricing tier of the new Azure Cache for Redis."
      }
    },
```

The template defines the values that are permitted for this parameter (Basic, Standard, or Premium), and assigns a default value (Basic) if no value is specified. Basic provides a single node with multiple sizes available up to 53 GB. Standard provides two-node Primary/Replica with multiple sizes available up to 53 GB and 99.9% SLA.

cacheSKUFamily

The family for the sku.

JSON

```
    "cacheSKUFamily": {
      "type": "string",
      "allowedValue/s": [
        "C",
        "P"
      ],
      "defaultValue": "C",
      "metadata": {
        "description": "The family for the sku."
      }
    },
```

cacheSKUCapacity

The size of the new Azure Cache for Redis instance.

For the Basic and Standard families:

JSON

```
    "cacheSKUCapacity": {
      "type": "int",
      "allowedValues": [
```

```
        0,
        1,
        2,
        3,
        4,
        5,
        6
      ],
      "defaultValue": 0,
      "metadata": {
        "description": "The size of the new Azure Cache for Redis instance.
"
      }
    }
```

The Premium value cache capacity is defined the same, except the allowed values run from 1 to 5 instead of from 0 to 6.

The template defines the integer values that are permitted for this parameter (0 through 6 for the Basic and Standard families; 1 through 5 for the Premium family). If no value is specified, the template assigns a default value of 0 for Basic and Standard, 1 for Premium.

The values correspond to following cache sizes:

Value	Basic and Standard cache size	Premium cache size
0	250 MB (default)	n/a
1	1 GB	6 GB (default)
2	2.5 GB	13 GB
3	6 GB	26 GB
4	13 GB	53 GB
5	26 GB	120 GB
6	53 GB	n/a

Variables for names

This template uses variables to construct names for the resources. It uses the uniqueString function to construct a value based on the resource group id.

JSON

```
"variables": {
  "hostingPlanName": "[concat('hostingplan',
uniqueString(resourceGroup().id))]",
  "webSiteName": "[concat('webSite', uniqueString(resourceGroup().id))]",
  "cacheName": "[concat('cache', uniqueString(resourceGroup().id))]"
},
```

Resources to deploy

App Service plan

Creates the service plan for hosting the web app. You provide the name of the plan through the **hostingPlanName** parameter. The location of the plan is the same location used for the resource group. The pricing tier and worker size are specified in the **sku** and **workerSize** parameters.

config

```
{
  "apiVersion": "2015-08-01",
  "name": "[parameters('hostingPlanName')]",
  "type": "Microsoft.Web/serverfarms",
  "location": "[resourceGroup().location]",
  "sku": {
    "name": "[parameters('sku')]",
    "capacity": "[parameters('workerSize')]"
  },
  "properties": {
    "name": "[parameters('hostingPlanName')]"
  }
},
```

Azure Cache for Redis

Creates the Azure Cache for Redis that is used with the web app. The name of the cache is specified in the **cacheName** variable.

The template creates the cache in the same location as the resource group.

JSON

```
{
  "name": "[variables('cacheName')]",
  "type": "Microsoft.Cache/Redis",
  "location": "[resourceGroup().location]",
```

```
  "apiVersion": "2015-08-01",
  "dependsOn": [ ],
  "tags": {
    "displayName": "cache"
  },
  "properties": {
    "sku": {
      "name": "[parameters('cacheSKUName')]",
      "family": "[parameters('cacheSKUFamily')]",
      "capacity": "[parameters('cacheSKUCapacity')]"
    }
  }
}
```

Web app (Azure Cache for Redis)

Creates the web app with name specified in the **webSiteName** variable.

Notice that the web app is configured with app setting properties that enable it to work with the Azure Cache for Redis. These app settings are dynamically created based on values provided during deployment.

JSON

```
{
  "apiVersion": "2015-08-01",
  "name": "[variables('webSiteName')]",
  "type": "Microsoft.Web/sites",
  "location": "[resourceGroup().location]",
  "dependsOn": [
    "[concat('Microsoft.Web/serverFarms/', variables('hostingPlanName'))]"
  ],
  "tags": {
    "[concat('hidden-related:', resourceGroup().id,
'/providers/Microsoft.Web/serverfarms/', variables('hostingPlanName'))]":
"empty",
    "displayName": "Website"
  },
  "properties": {
    "name": "[variables('webSiteName')]",
    "serverFarmId": "[resourceId('Microsoft.Web/serverfarms',
variables('hostingPlanName'))]"
  },
  "resources": [
    {
      "apiVersion": "2015-08-01",
      "type": "config",
      "name": "appsettings",
      "dependsOn": [
        "[concat('Microsoft.Web/Sites/', variables('webSiteName'))]",
        "[concat('Microsoft.Cache/Redis/', variables('cacheName'))]"
```

```
      ],
      "properties": {
       "CacheConnection": "
[concat(variables('cacheHostName'),'.redis.cache.windows.net,abortConnect=fa
lse,ssl=true,password=', listKeys(resourceId('Microsoft.Cache/Redis',
variables('cacheName')), '2015-08-01').primaryKey)]"
      }
    }
  ]
}
```

Web app (RedisEnterprise)

For RedisEnterprise, because the resource types are slightly different, the way to do **listKeys** is different:

JSON

```
{
  "apiVersion": "2015-08-01",
  "name": "[variables('webSiteName')]",
  "type": "Microsoft.Web/sites",
  "location": "[resourceGroup().location]",
  "dependsOn": [
    "[concat('Microsoft.Web/serverFarms/', variables('hostingPlanName'))]"
  ],
  "tags": {
    "[concat('hidden-related:', resourceGroup().id,
'/providers/Microsoft.Web/serverfarms/', variables('hostingPlanName'))]":
"empty",
    "displayName": "Website"
  },
  "properties": {
    "name": "[variables('webSiteName')]",
    "serverFarmId": "[resourceId('Microsoft.Web/serverfarms',
variables('hostingPlanName'))]"
  },
  "resources": [
    {
      "apiVersion": "2015-08-01",
      "type": "config",
      "name": "appsettings",
      "dependsOn": [
        "[concat('Microsoft.Web/Sites/', variables('webSiteName'))]",
        "[concat('Microsoft.Cache/RedisEnterprise/databases/',
variables('cacheName'), "/default")]",
      ],
      "properties": {
       "CacheConnection": "
[concat(variables('cacheHostName'),abortConnect=false,ssl=true,password=',
listKeys(resourceId('Microsoft.Cache/RedisEnterprise',
variables('cacheName'), 'default'), '2020-03-01').primaryKey)]"
```

```
      }
    }
  ]
}
```

Commands to run deployment

To deploy the resources to Azure, you must be signed in to your Azure account and you must use the Azure Resource Manager module. To learn about using Azure Resource Manager with either Azure PowerShell or Azure CLI, see:

- Manage Azure resources by using Azure PowerShell
- Manage Azure resources by using Azure CLI.

The following examples assume you already have a resource group in your account with the specified name.

PowerShell

Azure PowerShell

```
New-AzResourceGroupDeployment -TemplateUri
https://raw.githubusercontent.com/Azure/azure-quickstart-
templates/master/quickstarts/microsoft.web/web-app-with-redis-
cache/azuredeploy.json -ResourceGroupName ExampleDeployGroup
```

Azure CLI

Azure CLI

```
azure group deployment create --template-uri
https://raw.githubusercontent.com/Azure/azure-quickstart-
templates/master/quickstarts/microsoft.web/web-app-with-redis-
cache/azuredeploy.json -g ExampleDeployGroup
```

Quickstart: Use an ARM template to deploy Azure SignalR Service

Article • 03/09/2023

This quickstart describes how to use an Azure Resource Manager template (ARM template) to create an Azure SignalR Service. You can deploy the Azure SignalR Service through the Azure portal, PowerShell, or CLI.

A resource manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal once you sign in.

Deploy to Azure

Prerequisites

Portal

An Azure account with an active subscription. Create one for free.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.9.1.41621",
      "templateHash": "7185938028783900711"
    }
```

```
  },
  "parameters": {
    "name": {
      "type": "string",
      "defaultValue": "[uniqueString(resourceGroup().id)]",
      "metadata": {
        "description": "The globally unique name of the SignalR resource to
create."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for the SignalR resource."
      }
    },
    "pricingTier": {
      "type": "string",
      "defaultValue": "Standard_S1",
      "allowedValues": [
        "Free_F1",
        "Standard_S1",
        "Premium_P1"
      ],
      "metadata": {
        "description": "The pricing tier of the SignalR resource."
      }
    },
    "capacity": {
      "type": "int",
      "defaultValue": 1,
      "allowedValues": [
        1,
        2,
        5,
        10,
        20,
        50,
        100
      ],
      "metadata": {
        "description": "The number of SignalR Unit."
      }
    },
    "serviceMode": {
      "type": "string",
      "defaultValue": "Default",
      "allowedValues": [
        "Default",
        "Serverless",
        "Classic"
      ],
      "metadata": {
        "description": "Visit https://github.com/Azure/azure-
```

```
signalr/blob/dev/docs/faq.md#service-mode to understand SignalR Service
Mode."
      }
    },
    "enableConnectivityLogs": {
      "type": "bool",
      "defaultValue": true
    },
    "enableMessagingLogs": {
      "type": "bool",
      "defaultValue": true
    },
    "enableLiveTrace": {
      "type": "bool",
      "defaultValue": true
    },
    "allowedOrigins": {
      "type": "array",
      "defaultValue": [
        "*"
      ],
      "metadata": {
        "description": "Set the list of origins that should be allowed to
make cross-origin calls."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.SignalRService/signalR",
      "apiVersion": "2022-02-01",
      "name": "[parameters('name')]",
      "location": "[parameters('location')]",
      "sku": {
        "capacity": "[parameters('capacity')]",
        "name": "[parameters('pricingTier')]"
      },
      "kind": "SignalR",
      "identity": {
        "type": "SystemAssigned"
      },
      "properties": {
        "tls": {
          "clientCertEnabled": false
        },
        "features": [
          {
            "flag": "ServiceMode",
            "value": "[parameters('serviceMode')]"
          },
          {
            "flag": "EnableConnectivityLogs",
            "value": "[string(parameters('enableConnectivityLogs'))]"
          },
          {
```

```
            "flag": "EnableMessagingLogs",
            "value": "[string(parameters('enableMessagingLogs'))]"
          },
          {
            "flag": "EnableLiveTrace",
            "value": "[string(parameters('enableLiveTrace'))]"
          }
        ],
        "cors": {
          "allowedOrigins": "[parameters('allowedOrigins')]"
        },
        "networkACLs": {
          "defaultAction": "Deny",
          "publicNetwork": {
            "allow": [
              "ClientConnection"
            ]
          },
          "privateEndpoints": [
            {
              "name": "mySignalRService.1fa229cd-bf3f-47f0-8c49-
afb36723997e",
              "allow": [
                "ServerConnection"
              ]
            }
          ]
        },
        "upstream": {
          "templates": [
            {
              "categoryPattern": "*",
              "eventPattern": "connect,disconnect",
              "hubPattern": "*",
              "urlTemplate": "https://example.com/chat/api/connect"
            }
          ]
        }
      }
    }
  ]
}
```

The template defines one Azure resource:

- Microsoft.SignalRService/SignalR

Deploy the template

Portal

Select the following link to deploy the Azure SignalR Service using the ARM template in the Azure portal:

Deploy to Azure

On the **Deploy an Azure SignalR Service** page:

1. If you want, change the **Subscription** from the default.
2. For **Resource group**, select **Create new**, enter a name for the new resource group, and select **OK**.
3. If you created a new resource group, select a **Region** for the resource group.
4. If you want, enter a new **Name** and the **Location** (such as **eastus2**) of the Azure SignalR Service. If you don't specify a name, it's automatically generated. The location for the Azure SignalR Service can be the same as or different from the region of the resource group. If you don't specify a location, it's set to the same region as the resource group.
5. Choose the **Pricing Tier** (**Free_F1** or **Standard_S1**), enter the **Capacity** (number of SignalR units), and choose a **Service Mode** of **Default** (requires hub server), **Serverless** (doesn't allow any server connection), or **Classic** (routed to hub server only if hub has server connection). Then choose whether to **Enable Connectivity Logs** or **Enable Messaging Logs**.

   > **Note**
   >
   > For the **Free_F1** pricing tier, the capacity is limited to 1 unit.

6. Select **Review + create**.

7. Read the terms and conditions, and then select **Create**.

> **Note**
>
> The deployment may take a few minutes to complete. Note the names for the Azure SignalR Service and the resource group, which you use to review the deployed resources later.

Review deployed resources

Portal

Follow these steps to see an overview of your new Azure SignalR Service:

1. In the Azure portal, search for and select **SignalR**.
2. In the SignalR list, select your new service. The **Overview** page for the new Azure SignalR Service appears.

Clean up resources

When it's no longer needed, delete the resource group, which deletes the resources in the resource group.

Portal

1. In the Azure portal, search for and select **Resource groups**.
2. In the resource group list, choose the name of your resource group.
3. In the **Overview** page of your resource group, select **Delete resource group**.
4. In the confirmation dialog box, type the name of your resource group, and then select **Delete**.

Next steps

For a step-by-step tutorial that guides you through the process of creating an ARM template, see:

Tutorial: Create and deploy your first ARM template

ARM template best practices

Article • 04/09/2023

This article shows you how to use recommended practices when constructing your Azure Resource Manager template (ARM template). These recommendations help you avoid common problems when using an ARM template to deploy a solution.

Template limits

Limit the size of your template to 4 MB. The 4-MB limit applies to the final state of the template after it has been expanded with iterative resource definitions, and values for variables and parameters. The parameter file is also limited to 4 MB. You may get an error with a template or parameter file of less than 4 MB if the total size of the request is too large. For more information about how to simplify your template to avoid a large request, see Resolve errors for job size exceeded.

You're also limited to:

- 256 parameters
- 256 variables
- 800 resources (including copy count)
- 64 output values
- 10 unique locations per subscription/tenant/management group scope
- 24,576 characters in a template expression

You can exceed some template limits by using a nested template. For more information, see Using linked and nested templates when deploying Azure resources. To reduce the number of parameters, variables, or outputs, you can combine several values into an object. For more information, see Objects as parameters.

Resource group

When you deploy resources to a resource group, the resource group stores metadata about the resources. The metadata is stored in the location of the resource group.

If the resource group's region is temporarily unavailable, you can't update resources in the resource group because the metadata is unavailable. The resources in other regions will still function as expected, but you can't update them. To minimize risk, locate your resource group and resources in the same region.

Parameters

The information in this section can be helpful when you work with parameters.

General recommendations for parameters

- Minimize your use of parameters. Instead, use variables or literal values for properties that don't need to be specified during deployment.
- Use camel case for parameter names.
- Use parameters for settings that vary according to the environment, like SKU, size, or capacity.
- Use parameters for resource names that you want to specify for easy identification.
- Provide a description of every parameter in the metadata.

  JSON

  ```
  "parameters": {
    "storageAccountType": {
      "type": "string",
      "metadata": {
        "description": "The type of the new storage account created to
  store the VM disks."
      }
    }
  }
  ```

- Define default values for parameters that aren't sensitive. By specifying a default value, it's easier to deploy the template, and users of your template see an example of an appropriate value. Any default value for a parameter must be valid for all users in the default deployment configuration.

  JSON

  ```
  "parameters": {
    "storageAccountType": {
      "type": "string",
      "defaultValue": "Standard_GRS",
      "metadata": {
        "description": "The type of the new storage account created to
  store the VM disks."
      }
    }
  }
  ```

- To specify an optional parameter, don't use an empty string as a default value. Instead, use a literal value or a language expression to construct a value.

```JSON
"storageAccountName": {
   "type": "string",
   "defaultValue": "[concat('storage',
uniqueString(resourceGroup().id))]",
   "metadata": {
     "description": "Name of the storage account"
   }
}
```

- Use `allowedValues` sparingly. Use it only when you must make sure some values aren't included in the permitted options. If you use `allowedValues` too broadly, you might block valid deployments by not keeping your list up to date.

- When a parameter name in your template matches a parameter in the PowerShell deployment command, Resource Manager resolves this naming conflict by adding the postfix **FromTemplate** to the template parameter. For example, if you include a parameter named **ResourceGroupName** in your template, it conflicts with the **ResourceGroupName** parameter in the New-AzResourceGroupDeployment cmdlet. During deployment, you're prompted to provide a value for **ResourceGroupNameFromTemplate**.

Security recommendations for parameters

- Always use parameters for user names and passwords (or secrets).

- Use `securestring` for all passwords and secrets. If you pass sensitive data in a JSON object, use the `secureObject` type. Template parameters with secure string or secure object types can't be read after resource deployment.

```JSON
"parameters": {
  "secretValue": {
    "type": "securestring",
    "metadata": {
      "description": "The value of the secret to store in the vault."
    }
  }
}
```

- Don't provide default values for user names, passwords, or any value that requires a `secureString` type.
- Don't provide default values for properties that increase the attack surface area of the application.

Location recommendations for parameters

- Use a parameter to specify the location for resources, and set the default value to `resourceGroup().location`. Providing a location parameter enables users of the template to specify a location where they have permission to deploy resources.

  JSON

  ```
  "parameters": {
     "location": {
       "type": "string",
       "defaultValue": "[resourceGroup().location]",
       "metadata": {
         "description": "The location in which the resources should be
  deployed."
       }
     }
  }
  ```

- Don't specify `allowedValues` for the location parameter. The locations you specify might not be available in all clouds.
- Use the location parameter value for resources that are likely to be in the same location. This approach minimizes the number of times users are asked to provide location information.
- For resources that aren't available in all locations, use a separate parameter or specify a literal location value.

Variables

The following information can be helpful when you work with variables:

- Use camel case for variable names.
- Use variables for values that you need to use more than once in a template. If a value is used only once, a hard-coded value makes your template easier to read.

- Use variables for values that you construct from a complex arrangement of template functions. Your template is easier to read when the complex expression only appears in variables.
- You can't use the reference function in the `variables` section of the template. The `reference` function derives its value from the resource's runtime state. However, variables are resolved during the initial parsing of the template. Construct values that need the `reference` function directly in the `resources` or `outputs` section of the template.
- Include variables for resource names that must be unique.
- Use a copy loop in variables to create a repeated pattern of JSON objects.
- Remove unused variables.

API version

Set the `apiVersion` property to a hard-coded API version for the resource type. When creating a new template, we recommend you use the latest API version for a resource type. To determine available values, see template reference.

When your template works as expected, we recommend you continue using the same API version. By using the same API version, you don't have to worry about breaking changes that might be introduced in later versions.

Don't use a parameter for the API version. Resource properties and values can vary by API version. IntelliSense in a code editor can't determine the correct schema when the API version is set to a parameter. If you pass in an API version that doesn't match the properties in your template, the deployment will fail.

Don't use variables for the API version.

Resource dependencies

When deciding what dependencies to set, use the following guidelines:

- Use the `reference` function and pass in the resource name to set an implicit dependency between resources that need to share a property. Don't add an explicit `dependsOn` element when you've already defined an implicit dependency. This approach reduces the risk of having unnecessary dependencies. For an example of setting an implicit dependency, see reference and list functions.

- Set a child resource as dependent on its parent resource.
- Resources with the condition element set to `false` are automatically removed from the dependency order. Set the dependencies as if the resource is always deployed.
- Let dependencies cascade without setting them explicitly. For example, your virtual machine depends on a virtual network interface, and the virtual network interface depends on a virtual network and public IP addresses. Therefore, the virtual machine is deployed after all three resources, but don't explicitly set the virtual machine as dependent on all three resources. This approach clarifies the dependency order and makes it easier to change the template later.
- If a value can be determined before deployment, try deploying the resource without a dependency. For example, if a configuration value needs the name of another resource, you might not need a dependency. This guidance doesn't always work because some resources verify the existence of the other resource. If you receive an error, add a dependency.

Resources

The following information can be helpful when you work with resources:

- To help other contributors understand the purpose of the resource, specify `comments` for each resource in the template.

  JSON

  ```
  "resources": [
    {
      "name": "[variables('storageAccountName')]",
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2019-06-01",
      "location": "[resourceGroup().location]",
      "comments": "This storage account is used to store the VM disks.",
        ...
    }
  ]
  ```

  If your ARM template is stored in a `.jsonc` file, comments using the `//` syntax are supported, as shown here.

  JavaScript

```
"resources": [
  {
    // This storage account is used to store the VM disks.
    "name": "[variables('storageAccountName')]",
    "type": "Microsoft.Storage/storageAccounts",
    "apiVersion": "2019-06-01",
    "location": "[resourceGroup().location]",
      ...
  }
]
```

For more details about comments and metadata see Understand the structure and syntax of ARM templates.

- If you use a *public endpoint* in your template (such as an Azure Blob storage public endpoint), *don't hard-code* the namespace. Use the `reference` function to dynamically retrieve the namespace. You can use this approach to deploy the template to different public namespace environments without manually changing the endpoint in the template. Set the API version to the same version that you're using for the storage account in your template.

JSON

```
"diagnosticsProfile": {
  "bootDiagnostics": {
    "enabled": "true",
    "storageUri": "
[reference(resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName')), '2019-06-01').primaryEndpoints.blob]"
  }
}
```

If the storage account is deployed in the same template that you're creating and the name of the storage account isn't shared with another resource in the template, you don't need to specify the provider namespace or the `apiVersion` when you reference the resource. The following example shows the simplified syntax.

JSON

```
"diagnosticsProfile": {
  "bootDiagnostics": {
    "enabled": "true",
    "storageUri": "
[reference(variables('storageAccountName')).primaryEndpoints.blob]"
  }
}
```

You also can reference an existing storage account that's in a different resource group.

JSON

```
"diagnosticsProfile": {
  "bootDiagnostics": {
    "enabled": "true",
    "storageUri": "
[reference(resourceId(parameters('existingResourceGroup'),
'Microsoft.Storage/storageAccounts',
parameters('existingStorageAccountName')), '2019-06-
01').primaryEndpoints.blob]"
  }
}
```

- Assign public IP addresses to a virtual machine only when an application requires it. To connect to a virtual machine for administrative purposes, use inbound NAT rules, a virtual network gateway, or a jumpbox.

  For more information about connecting to virtual machines, see:
  - What is Azure Bastion?
  - How to connect and sign on to an Azure virtual machine running Windows
  - Setting up WinRM access for Virtual Machines in Azure Resource Manager
  - Connect to a Linux VM

- The `domainNameLabel` property for public IP addresses must be unique. The `domainNameLabel` value must be between 3 and 63 characters long, and follow the rules specified by this regular expression: `^[a-z][a-z0-9-]{1,61}[a-z0-9]$`. Because the `uniqueString` function generates a string that is 13 characters long, the `dnsPrefixString` parameter is limited to 50 characters.

  JSON

```
"parameters": {
  "dnsPrefixString": {
    "type": "string",
    "maxLength": 50,
    "metadata": {
      "description": "The DNS label for the public IP address. It must
be lowercase. It should match the following regular expression, or it
will raise an error: ^[a-z][a-z0-9-]{1,61}[a-z0-9]$"
    }
  }
},
"variables": {
```

```
  "dnsPrefix": "
[concat(parameters('dnsPrefixString'),uniquestring(resourceGroup().id))
]"
}
```

- When you add a password to a custom script extension, use the `commandToExecute` property in the `protectedSettings` property.

JSON

```
"properties": {
  "publisher": "Microsoft.Azure.Extensions",
  "type": "CustomScript",
  "typeHandlerVersion": "2.0",
  "autoUpgradeMinorVersion": true,
  "settings": {
    "fileUris": [
      "[concat(variables('template').assets, '/lamp-
app/install_lamp.sh')]"
    ]
  },
  "protectedSettings": {
    "commandToExecute": "[concat('sh install_lamp.sh ',
parameters('mySqlPassword'))]"
  }
}
```

> ⓘ **Note**
>
> To ensure that secrets are encrypted when they are passed as parameters to VMs and extensions, use the `protectedSettings` property of the relevant extensions.

- Specify explicit values for properties that have default values that could change over time. For example, if you're deploying an AKS cluster, you can either specify or omit the `kubernetesVersion` property. If you don't specify it, then the cluster is defaulted to the N-1 minor version and latest patch. When you deploy the cluster using an ARM template, this default behavior might not be what you expect. Redeploying your template may result in the cluster being upgraded to a new Kubernetes version unexpectedly. Instead, consider specifying an explicit version number and then manually changing it when you're ready to upgrade your cluster.

Comments

In addition to the `comments` property, comments using the `//` syntax are supported. For more details about comments and metadata see Understand the structure and syntax of ARM templates. You may choose to save JSON files that contain `//` comments using the `.jsonc` file extension, to indicate the JSON file contains comments. The ARM service will also accept comments in any JSON file including parameters files.

Visual Studio Code ARM Tools

Working with ARM templates is much easier with the Azure Resource Manager (ARM) Tools for Visual Studio Code. This extension provides language support, resource snippets, and resource auto-completion to help you create and validate Azure Resource Manager templates. To learn more and install the extension, see Azure Resource Manager (ARM) Tools.

Use test toolkit

The ARM template test toolkit is a script that checks whether your template uses recommended practices. When your template isn't compliant with recommended practices, it returns a list of warnings with suggested changes. The test toolkit can help you learn how to implement best practices in your template.

After you've completed your template, run the test toolkit to see if there are ways you can improve its implementation. For more information, see Use ARM template test toolkit.

Next steps

- For information about the structure of the template file, see Understand the structure and syntax of ARM templates.
- For recommendations about how to build templates that work in all Azure cloud environments, see Develop ARM templates for cloud consistency.

Data types in ARM templates

Article • 04/09/2023

This article describes the data types supported in Azure Resource Manager templates (ARM templates).

Supported types

Within an ARM template, you can use these data types:

- array
- bool
- int
- object
- secureObject
- securestring
- string

Arrays

Arrays start with a left bracket (`[`) and end with a right bracket (`]`). An array can be declared in a single line or multiple lines. Each element is separated by a comma.

JSON

```
"parameters": {
  "exampleArray": {
    "type": "array",
    "defaultValue": [
      1,
      2,
      3
    ]
  }
},

"outputs": {
  "arrayOutput": {
    "type": "array",
    "value": "[variables('exampleArray')]"
  },
  "firstExampleArrayElement": {
    "type": "int",
    "value": "[parameters('exampleArray')[0]]"
```

```
  }
}
```

The elements of an array can be the same type or different types.

JSON

```
"variables": {
  "mixedArray": [
    "[resourceGroup().name]",
    1,
    true,
    "example string"
  ]
}

"outputs": {
  "arrayOutput": {
    "type": "array",
    "value": "[variables('mixedArray')]"
  },
  "firstMixedArrayElement": {
    "type": "string",
    "value": "[variables('mixedArray')[0]]"
  }
}
```

Booleans

When specifying boolean values, use `true` or `false`. Don't surround the value with quotation marks.

JSON

```
"parameters": {
  "exampleBool": {
    "type": "bool",
    "defaultValue": true
  }
},
```

Integers

When specifying integer values, don't use quotation marks.

JSON

```
"parameters": {
  "exampleInt": {
    "type": "int",
    "defaultValue": 1
  }
}
```

For integers passed as inline parameters, the range of values may be limited by the SDK or command-line tool you use for deployment. For example, when using PowerShell to deploy a template, integer types can range from -2147483648 to 2147483647. To avoid this limitation, specify large integer values in a parameter file. Resource types apply their own limits for integer properties.

Objects

Objects start with a left brace (`{`) and end with a right brace (`}`). Each property in an object consists of `key` and `value`. The `key` and `value` are enclosed in double quotes and separated by a colon (`:`). Each property is separated by a comma.

JSON

```
"parameters": {
  "exampleObject": {
    "type": "object",
    "defaultValue": {
      "name": "test name",
      "id": "123-abc",
      "isCurrent": true,
      "tier": 1
    }
  }
}
```

You can get a property from an object with dot notation.

JSON

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "exampleObject": {
            "type": "object",
            "defaultValue": {
                "name": "test name",
                "id": "123-abc",
```

```
                "isCurrent": true,
                "tier": 1
            }
        }
    },
    "resources": [
    ],
    "outputs": {
        "nameFromObject": {
            "type": "string",
            "value": "[parameters('exampleObject').name]"
        }
    }
}
```

In JSON, an object is an unordered collection of zero or more key/value pairs. The ordering can be different depending on the implementations. For example, the Bicep items() function sorts the objects in the alphabetical order. In other places, the original ordering can be preserved. Because of this non-determinism, avoid making any assumptions about the ordering of object keys when writing code, which interacts with deployments parameters & outputs.

Strings

Strings are marked with double quotes.

JSON

```
"parameters": {
  "exampleString": {
    "type": "string",
    "defaultValue": "test value"
  }
},
```

Secure strings and objects

Secure string uses the same format as string, and secure object uses the same format as object. When you set a parameter to a secure string or secure object, the value of the parameter isn't saved to the deployment history and isn't logged. However, if you set that secure value to a property that isn't expecting a secure value, the value isn't protected. For example, if you set a secure string to a tag, that value is stored as plain text. Use secure strings for passwords and secrets.

The following example shows two secure parameters.

JSON

```
"parameters": {
  "password": {
    "type": "securestring"
  },
  "configValues": {
    "type": "secureObject"
  }
}
```

> **Note**
>
> Don't use secure strings or objects as output values. If you include a secure value as an output value, the value isn't displayed in the deployment history and can't be retrieved from another template. Instead, save the secure value in a key vault, and **pass as a parameter from the key vault**.

Next steps

To learn about the template syntax, see Understand the structure and syntax of ARM templates.

Frequently asked questions about ARM templates

FAQ

This article answers frequently asked questions about Azure Resource Manager templates (ARM templates).

Getting started

What are ARM templates, and why should I use them?

ARM templates are JSON files where you define what you want to deploy to Azure. Templates help you implement an infrastructure-as-code solution for Azure. Your organization can repeatedly and reliably deploy the required infrastructure to different environments.

To learn more about how ARM templates help you manage your Azure infrastructure, see What are ARM templates?

How do I get started with templates?

To simplify authoring ARM templates, you need the right tools. We recommend installing Visual Studio Code and the Azure Resource Manager tools extension. For a quick introduction to these tools, see Quickstart: Create ARM templates with Visual Studio Code.

When you're ready to learn about creating ARM templates, start the beginner tutorial series on ARM templates. These tutorials take you step by step through the process of constructing an ARM template. You learn about the different sections of the template and how to they work together. This content is also available as a Learn module.

Should I use ARM templates or Terraform to deploy to Azure?

Use the option that you like the best. Both services assist you with automating deployments to Azure.

We believe there are benefits to using ARM templates over other infrastructure-as-code services. To learn about those benefits, see Why choose ARM templates?

Build 2020

I missed your presentation at Microsoft Build 2020. Is the presentation available for viewing?

Yes, please watch it anytime.

Where can I get more information about the new features you announced at Build?

For general information about features we're working, join our Azure Advisors Deployments Yammer group.

To learn about the new template language, sign up for notifications.

To learn about template specs, see Azure Resource Manager template specs.

Creating and testing templates

Where can I learn about best practices for ARM templates?

For recommendations about how you implement your templates, see ARM template best practices. After creating a template, run the ARM test toolkit. It checks whether your template matches recommended practices.

I have set up my environment through the portal. Is there some way to get the template from an existing resource group?

Yes, you can export the template from a resource group. The exported template is a good starting point for learning about templates, but you'll probably want to revise it before using it in a production environment.

When exporting the template, you can select which resources you want to include in the template.

Can I create a resource group in an ARM template and deploy resources to it?

Yes, you can create a resource group in a template when you deploy the template at the level of your Azure subscription. For an example of creating a resource group and deploying resources, see Resource group and resources.

Can I create a subscription in an ARM template?

Yes, for more information, see Programmatically create Azure subscriptions with the latest APIs.

How can I test my template before deploying it?

We recommend running the ARM test toolkit and the what-if operation on your templates before deploying them. The test toolkit checks whether your template uses best practices. It provides warnings when it identifies changes that could improve how you've implemented your template.

The what-if operation shows the changes your template will make to your environment. You can see unintended changes before they're deployed. What-if also returns any errors it can detect during preflight validation. For example, if your template contains a syntactical error, it returns that error. It also returns any errors it can determine about the final state of the deployed resources. For example, if your template deploys a storage account with a name that is already in use, what-if returns that error.

Where can I find information about the properties that are available for each resource type?

VS Code provides intellisense for working with the resource properties. You can also view the template reference for properties and descriptions.

I need to create multiple instances of a resource type. How do I create an iterator in my template?

Use the copy element to specify more than one instance. You can use copy on resources, properties, variables, and outputs.

Template language

I've heard you're working on a new template language. Where can I find out more about it?

To learn about the new language, see What is Bicep (Preview)?.

Is there a plan to support creating templates in YAML?

Currently, there's no plan to support YAML. We believe the new template language will offer a solution that is easier to use than YAML or JSON.

Can I still write templates in JSON after the new template language has been released?

Yes, you can continue using JSON templates.

Will you offer a tool to convert my JSON templates to the new template language?

Yes. See Converting ARM templates between JSON and Bicep.

Template Specs

How are template specs and Azure Blueprints related?

Azure Blueprints will use template specs in its implementation by replacing the `blueprint definition` resource with a `template spec` resource. We'll provide a migration path to convert the blueprint definition into a template spec, but the blueprint definition APIs will still be supported. There are no changes to the `blueprint assignment` resource. Blueprints will remain a user-experience to compose a governed environment in Azure.

Do template specs replace linked templates?

No, but template specs are designed to work well with linked templates. You don't have to move the linked template to a publicly accessible endpoint before deploying the parent template. Instead, you package the parent template and its artifacts together when creating the template spec.

Can template specs be shared across subscriptions?

Yes, they can be used across subscriptions as long as the user has read access to the template spec. Template specs can't be used across tenants.

Scripts in templates

Can I include a script in my template to do tasks that aren't possible in a template?

Yes, use deployment scripts. You can include Azure PowerShell or Azure CLI scripts in your templates.

Can I still use custom script extensions and desired state configuration (DSC)?

Those options are still available and haven't changed. Deployment scripts are designed to perform actions that aren't related to the VM guest. If you need to run a script on a host operating system in a VM, then the custom script extension and/or DSC would be a better choice. However, deployment scripts have advantages, such as setting the timeout duration.

Are deployment scripts supported in Azure Government?

Yes, you can use deployment scripts in US Gov Arizona and US Gov Virginia.

Preview changes before deployment

Can I preview the changes that will happen before deploying a template?

Yes, use the what-if feature. It evaluates the current state of your environment and compares it to the state that will exist after deployment. You can examine the summarized changes to make sure the template doesn't have any unexpected results.

Can I use what-if with both incremental and complete modes?

Yes, both deployment modes are supported. For an example of using incremental mode, see Run what-if operation. For an example of using complete mode, see Confirm deletion.

Does what-if work with linked templates?

Yes, what-if evaluates the state of the parent template and its linked templates.

Can I use what-if in an Azure Pipeline?

Yes, you can use what-if to verify that the Pipeline should continue.

When I use what-if, I see changes in properties that aren't in my template. Is this "noise" expected?

We're working on reducing the noise. You help us improve by submitting issues in our GitHub repo here: https://aka.ms/WhatIfIssues

Template visualizer

Is there a way for me to visualize my ARM template and its resources?

We have a community-contributed VS Code extension that does a great job of visualizing your ARM template. It shows the resources you're deploying and the relationships between them.

Can I use the template visualizer outside of VS Code?

The template visualizer is being previewed in the portal. For more information, watch this short session from Build.

Deployment limits

How many resource groups can I deploy to in a single deployment operation?

In the past, this limit was five resource groups. It has recently been increased to 800 resource groups. For more information, see Create resource groups and resources at the subscription level.

I got an error about being limited to 800 deployments in the deployment history. What should I do?

We're changing how the deployment history for a resource group is maintained. In the past, you had to manually delete deployments from this history to avoid this error. Starting in June 2020, we'll automatically delete deployments from the history as you get near the limit. For more information, see Automatic deletions from deployment history.

Deleting a deployment from the history doesn't affect the deployed resources.

Templates and DevOps

Can I integrate ARM templates into Azure Pipelines?

Yes. For an explanation of how to use template and pipelines, see Tutorial: Continuous integration of ARM templates with Azure Pipelines and Integrate ARM templates with Azure Pipelines.

Can I use GitHub Actions to deploy a template?

Yes, see Deploy ARM templates by using GitHub Actions.

Next steps

For an introduction to ARM templates, see What are ARM templates?.

Develop ARM templates for cloud consistency

Article • 03/09/2023

> **Important**
>
> Using this Azure feature from PowerShell requires the `AzureRM` module installed. This is an older module only available for Windows PowerShell 5.1 that no longer receives new features. The `Az` and `AzureRM` modules are **not** compatible when installed for the same versions of PowerShell. If you need both versions:
>
> 1. **Uninstall the Az module** from a PowerShell 5.1 session.
> 2. **Install the AzureRM module** from a PowerShell 5.1 session.
> 3. **Download and install PowerShell Core 6.x or later**.
> 4. **Install the Az module** in a PowerShell Core session.

A key benefit of Azure is consistency. Development investments for one location are reusable in another. An Azure Resource Manager template (ARM template) makes your deployments consistent and repeatable across environments, including the global Azure, Azure sovereign clouds, and Azure Stack. To reuse templates across clouds, however, you need to consider cloud-specific dependencies as this guide explains.

Microsoft offers intelligent, enterprise-ready cloud services in many locations, including:

- The global Azure platform supported by a growing network of Microsoft-managed datacenters in regions around the world.
- Isolated sovereign clouds like Azure Germany, Azure Government, and Azure China 21Vianet. Sovereign clouds provide a consistent platform with most of the same great features that global Azure customers have access to.
- Azure Stack, a hybrid cloud platform that lets you deliver Azure services from your organization's datacenter. Enterprises can set up Azure Stack in their own datacenters, or consume Azure Services from service providers, running Azure Stack in their facilities (sometimes known as hosted regions).

At the core of all these clouds, Azure Resource Manager provides an API that allows a wide variety of user interfaces to communicate with the Azure platform. This API gives you powerful infrastructure-as-code capabilities. Any type of resource that is available on the Azure cloud platform can be deployed and configured with Azure Resource

Manager. With a single template, you can deploy and configure your complete application to an operational end state.

The consistency of global Azure, the sovereign clouds, hosted clouds, and a cloud in your datacenter helps you benefit from Azure Resource Manager. You can reuse your development investments across these clouds when you set up template-based resource deployment and configuration.

However, even though the global, sovereign, hosted, and hybrid clouds provide consistent services, not all clouds are identical. As a result, you can create a template with dependencies on features available only in a specific cloud.

The rest of this guide describes the areas to consider when planning to develop new or updating existing ARM templates for Azure Stack. In general, your checklist should include the following items:

- Verify that the functions, endpoints, services, and other resources in your template are available in the target deployment locations.
- Store nested templates and configuration artifacts in accessible locations, ensuring access across clouds.
- Use dynamic references instead of hard-coding links and elements.
- Ensure the template parameters you use work in the target clouds.
- Verify that resource-specific properties are available the target clouds.

For an introduction to ARM templates, see Template deployment.

Ensure template functions work

The basic syntax of an ARM template is JSON. Templates use a superset of JSON, extending the syntax with expressions and functions. The template language processor is frequently updated to support additional template functions. For a detailed explanation of the available template functions, see ARM template functions.

New template functions that are introduced to Azure Resource Manager aren't immediately available in the sovereign clouds or Azure Stack. To deploy a template successfully, all functions referenced in the template must be available in the target cloud.

Azure Resource Manager capabilities will always be introduced to global Azure first. You can use the following PowerShell script to verify whether newly introduced template functions are also available in Azure Stack:

1. Make a clone of the GitHub repository: https://github.com/marcvaneijk/arm-template-functions.
2. Once you have a local clone of the repository, connect to the destination's Azure Resource Manager with PowerShell.
3. Import the psm1 module and execute the Test-AzureRmTemplateFunctions cmdlet:

   PowerShell

   ```
   # Import the module
   Import-module <path to local clone>\AzTemplateFunctions.psm1

   # Execute the Test-AzureRmTemplateFunctions cmdlet
   Test-AzureRmTemplateFunctions -path <path to local clone>
   ```

The script deploys multiple, minimized templates, each containing only unique template functions. The output of the script reports the supported and unavailable template functions.

Working with linked artifacts

A template can contain references to linked artifacts and contain a deployment resource that links to another template. The linked templates (also referred to as nested template) are retrieved by Resource Manager at runtime. A template can also contain references to artifacts for virtual machine (VM) extensions. These artifacts are retrieved

by the VM extension running inside of the VM for configuration of the VM extension during the template deployment.

The following sections describe considerations for cloud consistency when developing templates that include artifacts that are external to the main deployment template.

Use nested templates across regions

Templates can be decomposed into small, reusable templates, each of which has a specific purpose and can be reused across deployment scenarios. To execute a deployment, you specify a single template known as the main or master template. It specifies the resources to deploy, such as virtual networks, VMs, and web apps. The main template can also contain a link to another template, meaning you can nest templates. Likewise, a nested template can contain links to other templates. You can nest up to five levels deep.

The following code shows how the templateLink parameter refers to a nested template:

JSON

```
"resources": [
  {
     "type": "Microsoft.Resources/deployments",
     "apiVersion": "2020-10-01",
     "name": "linkedTemplate",
     "properties": {
       "mode": "incremental",
       "templateLink": {

"uri":"https://mystorageaccount.blob.core.windows.net/AzureTemplates/vNet.js
on",
          "contentVersion":"1.0.0.0"
       }
     }
  }
]
```

Azure Resource Manager evaluates the main template at runtime and retrieves and evaluates each nested template. After all nested templates are retrieved, the template is flattened, and further processing is initiated.

Make linked templates accessible across clouds

Consider where and how to store any linked templates you use. At runtime, Azure Resource Manager fetches—and therefore requires direct access to—any linked

templates. A common practice is to use GitHub to store the nested templates. A GitHub repository can contain files that are accessible publicly through a URL. Although this technique works well for the public cloud and the sovereign clouds, an Azure Stack environment might be located on a corporate network or on a disconnected remote location, without any outbound Internet access. In those cases, Azure Resource Manager would fail to retrieve the nested templates.

A better practice for cross-cloud deployments is to store your linked templates in a location that is accessible for the target cloud. Ideally all deployment artifacts are maintained in and deployed from a continuous integration/continuous development (CI/CD) pipeline. Alternatively, you can store nested templates in a blob storage container, from which Azure Resource Manager can retrieve them.

Since the blob storage on each cloud uses a different endpoint fully qualified domain name (FQDN), configure the template with the location of the linked templates with two parameters. Parameters can accept user input at deployment time. Templates are typically authored and shared by multiple people, so a best practice is to use a standard name for these parameters. Naming conventions help make templates more reusable across regions, clouds, and authors.

In the following code, `_artifactsLocation` is used to point to a single location, containing all deployment-related artifacts. Notice that a default value is provided. At deployment time, if no input value is specified for `_artifactsLocation`, the default value is used. The `_artifactsLocationSasToken` is used as input for the `sasToken`. The default value should be an empty string for scenarios where the `_artifactsLocation` isn't secured — for example, a public GitHub repository.

JSON

```
"parameters": {
  "_artifactsLocation": {
    "type": "string",
    "metadata": {
      "description": "The base URI where artifacts required by this template
are located."
    },
    "defaultValue": "https://raw.githubusercontent.com/Azure/azure-
quickstart-templates/master/quickstarts/microsoft.compute/vm-custom-script-
windows/"
  },
  "_artifactsLocationSasToken": {
    "type": "securestring",
    "metadata": {
      "description": "The sasToken required to access _artifactsLocation."
    },
    "defaultValue": ""
```

```
  }
}
```

Throughout the template, links are generated by combining the base URI (from the `_artifactsLocation` parameter) with an artifact-relative path and the `_artifactsLocationSasToken`. The following code shows how to specify the link to the nested template using the uri template function:

JSON

```
"resources": [
  {
    "type": "Microsoft.Resources/deployments",
    "apiVersion": "2020-10-01",
    "name": "shared",
    "properties": {
      "mode": "Incremental",
      "templateLink": {
        "uri": "[uri(parameters('_artifactsLocation'),
concat('nested/vnet.json', parameters('_artifactsLocationSasToken')))]",
        "contentVersion": "1.0.0.0"
      }
    }
  }
]
```

By using this approach, the default value for the `_artifactsLocation` parameter is used. If the linked templates need to be retrieved from a different location, the parameter input can be used at deployment time to override the default value—no change to the template itself is needed.

Use _artifactsLocation instead of hardcoding links

Besides being used for nested templates, the URL in the `_artifactsLocation` parameter is used as a base for all related artifacts of a deployment template. Some VM extensions include a link to a script stored outside the template. For these extensions, you should not hardcode the links. For example, the Custom Script and PowerShell DSC extensions may link to an external script on GitHub as shown:

JSON

```
"properties": {
  "publisher": "Microsoft.Compute",
  "type": "CustomScriptExtension",
  "typeHandlerVersion": "1.9",
  "autoUpgradeMinorVersion": true,
```

```
  "settings": {
    "fileUris": [
      "https://raw.githubusercontent.com/Microsoft/dotnet-core-sample-
templates/master/dotnet-core-music-windows/scripts/configure-music-app.ps1"
    ]
  }
}
```

Hardcoding the links to the script potentially prevents the template from deploying successfully to another location. During configuration of the VM resource, the VM agent running inside the VM initiates a download of all the scripts linked in the VM extension, and then stores the scripts on the VM's local disk. This approach functions like the nested template links explained earlier in the "Use nested templates across regions" section.

Resource Manager retrieves nested templates at runtime. For VM extensions, the retrieval of any external artifacts is performed by the VM agent. Besides the different initiator of the artifact retrieval, the solution in the template definition is the same. Use the _artifactsLocation parameter with a default value of the base path where all the artifacts are stored (including the VM extension scripts) and the `_artifactsLocationSasToken` parameter for the input for the sasToken.

JSON

```
"parameters": {
  "_artifactsLocation": {
    "type": "string",
    "metadata": {
      "description": "The base URI where artifacts required by this template
are located."
    },
    "defaultValue": "https://raw.githubusercontent.com/Microsoft/dotnet-
core-sample-templates/master/dotnet-core-music-windows/"
  },
  "_artifactsLocationSasToken": {
    "type": "securestring",
    "metadata": {
      "description": "The sasToken required to access _artifactsLocation."
    },
    "defaultValue": ""
  }
}
```

To construct the absolute URI of an artifact, the preferred method is to use the uri template function, instead of the concat template function. By replacing hardcoded links to the scripts in the VM extension with the uri template function, this functionality in the template is configured for cloud consistency.

JSON

```
"properties": {
  "publisher": "Microsoft.Compute",
  "type": "CustomScriptExtension",
  "typeHandlerVersion": "1.9",
  "autoUpgradeMinorVersion": true,
  "settings": {
    "fileUris": [
      "[uri(parameters('_artifactsLocation'), concat('scripts/configure-
music-app.ps1', parameters('_artifactsLocationSasToken')))]"
    ]
  }
}
```

With this approach, all deployment artifacts, including configuration scripts, can be stored in the same location with the template itself. To change the location of all the links, you only need to specify a different base URL for the *artifactsLocation parameters*.

Factor in differing regional capabilities

With the agile development and continuous flow of updates and new services introduced to Azure, regions can differ in availability of services or updates. After rigorous internal testing, new services or updates to existing services are usually introduced to a small audience of customers participating in a validation program. After successful customer validation, the services or updates are made available within a subset of Azure regions, then introduced to more regions, rolled out to the sovereign clouds, and potentially made available for Azure Stack customers as well.

Knowing that Azure regions and clouds may differ in their available services, you can make some proactive decisions about your templates. A good place to start is by examining the available resource providers for a cloud. A resource provider tells you the set of resources and operations that are available for an Azure service.

A template deploys and configures resources. A resource type is provided by a resource provider. For example, the compute resource provider (Microsoft.Compute), provides multiple resource types such as virtualMachines and availabilitySets. Each resource provider provides an API to Azure Resource Manager defined by a common contract, enabling a consistent, unified authoring experience across all resource providers. However, a resource provider that is available in global Azure may not be available in a sovereign cloud or an Azure Stack region.

To verify the resource providers that are available in a given cloud, run the following script in the Azure CLI:

Azure CLI

```
az provider list --query "[].{Provider:namespace, Status:registrationState}"
--out table
```

You can also use the following PowerShell cmdlet to see available resource providers:

Azure PowerShell

```
Get-AzureRmResourceProvider -ListAvailable | Select-Object
ProviderNamespace, RegistrationState
```

Verify the version of all resource types

A set of properties is common for all resource types, but each resource also has its own specific properties. New features and related properties are added to existing resource types at times through a new API version. A resource in a template has its own API version property - `apiVersion`. This versioning ensures that an existing resource configuration in a template is not affected by changes on the platform.

New API versions introduced to existing resource types in global Azure might not immediately be available in all regions, sovereign clouds, or Azure Stack. To view a list of the available resource providers, resource types, and API versions for a cloud, you can use Resource Explorer in Azure portal. Search for Resource Explorer in the All Services

menu. Expand the Providers node in Resource Explorer to return all the available resource providers, their resource types, and API versions in that cloud.

To list the available API version for all resource types in a given cloud in Azure CLI, run the following script:

Azure CLI

```
az provider list --query "[].{namespace:namespace,
resourceType:resourceType[]}"
```

You can also use the following PowerShell cmdlet:

Azure PowerShell

```
Get-AzureRmResourceProvider | select-object ProviderNamespace -
ExpandProperty ResourceTypes | ft ProviderNamespace, ResourceTypeName,
ApiVersions
```

Refer to resource locations with a parameter

A template is always deployed into a resource group that resides in a region. Besides the deployment itself, each resource in a template also has a location property that you use to specify the region to deploy in. To develop your template for cloud consistency, you need a dynamic way to refer to resource locations, because each Azure Stack can contain unique location names. Usually resources are deployed in the same region as the resource group, but to support scenarios such as cross-region application availability, it can be useful to spread resources across regions.

Even though you could hardcode the region names when specifying the resource properties in a template, this approach doesn't guarantee that the template can be deployed to other Azure Stack environments, because the region name most likely doesn't exist there.

To accommodate different regions, add an input parameter location to the template with a default value. The default value will be used if no value is specified during deployment.

The template function `[resourceGroup()]` returns an object that contains the following key/value pairs:

JSON

```
{
  "id":
"/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}",
  "name": "{resourceGroupName}",
  "location": "{resourceGroupLocation}",
  "tags": {
  },
  "properties": {
    "provisioningState": "{status}"
  }
}
```

By referencing the location key of the object in the defaultValue of the input parameter, Azure Resource Manager will, at runtime, replace the `[resourceGroup().location]` template function with the name of the location of the resource group the template is deployed to.

JSON

```
"parameters": {
  "location": {
    "type": "string",
    "metadata": {
      "description": "Location the resources will be deployed to."
    },
    "defaultValue": "[resourceGroup().location]"
  }
},
"resources": [
  {
    "type": "Microsoft.Storage/storageAccounts",
    "apiVersion": "2015-06-15",
    "name": "storageaccount1",
    "location": "[parameters('location')]",
    ...
```

With this template function, you can deploy your template to any cloud without even knowing the region names in advance. In addition, a location for a specific resource in the template can differ from the resource group location. In this case, you can configure it by using additional input parameters for that specific resource, while the other resources in the same template still use the initial location input parameter.

Track versions using API profiles

It can be very challenging to keep track of all the available resource providers and related API versions that are present in Azure Stack. For example, at the time of writing,

the latest API version for **Microsoft.Compute/availabilitySets** in Azure is `2018-04-01`, while the available API version common to Azure and Azure Stack is `2016-03-30`. The common API version for **Microsoft.Storage/storageAccounts** shared among all Azure and Azure Stack locations is `2016-01-01`, while the latest API version in Azure is `2018-02-01`.

For this reason, Resource Manager introduced the concept of API profiles to templates. Without API profiles, each resource in a template is configured with an `apiVersion` element that describes the API version for that specific resource.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "location": {
      "type": "string",
      "metadata": {
          "description": "Location the resources will be deployed to."
      },
      "defaultValue": "[resourceGroup().location]"
    }
  },
  "variables": {},
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2016-01-01",
      "name": "mystorageaccount",
      "location": "[parameters('location')]",
      "properties": {
        "accountType": "Standard_LRS"
      }
    },
    {
      "type": "Microsoft.Compute/availabilitySets",
      "apiVersion": "2016-03-30",
      "name": "myavailabilityset",
      "location": "[parameters('location')]",
      "properties": {
        "platformFaultDomainCount": 2,
        "platformUpdateDomainCount": 2
      }
    }
  ],
  "outputs": {}
}
```

An API profile version acts as an alias for a single API version per resource type common to Azure and Azure Stack. Instead of specifying an API version for each resource in a template, you specify only the API profile version in a new root element called `apiProfile` and omit the `apiVersion` element for the individual resources.

JSON

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "apiProfile": "2018-03-01-hybrid",
    "parameters": {
        "location": {
            "type": "string",
            "metadata": {
                "description": "Location the resources will be deployed to."
            },
            "defaultValue": "[resourceGroup().location]"
        }
    },
    "variables": {},
    "resources": [
        {
            "type": "Microsoft.Storage/storageAccounts",
            "name": "mystorageaccount",
            "location": "[parameters('location')]",
            "properties": {
                "accountType": "Standard_LRS"
            }
        },
        {
            "type": "Microsoft.Compute/availabilitySets",
            "name": "myavailabilityset",
            "location": "[parameters('location')]",
            "properties": {
                "platformFaultDomainCount": 2,
                "platformUpdateDomainCount": 2
            }
        }
    ],
    "outputs": {}
}
```

The API profile ensures that the API versions are available across locations, so you do not have to manually verify the apiVersions that are available in a specific location. To ensure the API versions referenced by your API profile are present in an Azure Stack environment, the Azure Stack operators must keep the solution up-to-date based on the policy for support. If a system is more than six months out of date, it is considered out of compliance, and the environment must be updated.

The API profile isn't a required element in a template. Even if you add the element, it will only be used for resources for which no `apiVersion` is specified. This element allows for gradual changes but doesn't require any changes to existing templates.

JSON

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "apiProfile": "2018-03-01-hybrid",
    "parameters": {
        "location": {
            "type": "string",
            "metadata": {
                "description": "Location the resources will be deployed to."
            },
            "defaultValue": "[resourceGroup().location]"
        }
    },
    "variables": {},
    "resources": [
        {
            "type": "Microsoft.Storage/storageAccounts",
            "apiVersion": "2016-01-01",
            "name": "mystorageaccount",
            "location": "[parameters('location')]",
            "properties": {
                "accountType": "Standard_LRS"
            }
        },
        {
            "type": "Microsoft.Compute/availabilitySets",
            "name": "myavailabilityset",
            "location": "[parameters('location')]",
            "properties": {
                "platformFaultDomainCount": 2,
                "platformUpdateDomainCount": 2
            }
        }
    ],
    "outputs": {}
}
```

Check endpoint references

Resources can have references to other services on the platform. For example, a public IP can have a public DNS name assigned to it. The public cloud, the sovereign clouds, and Azure Stack solutions have their own distinct endpoint namespaces. In most cases, a

resource requires only a prefix as input in the template. During runtime, Azure Resource Manager appends the endpoint value to it. Some endpoint values need to be explicitly specified in the template.

> **Note**
>
> To develop templates for cloud consistency, don't hardcode endpoint namespaces.

The following two examples are common endpoint namespaces that need to be explicitly specified when creating a resource:

- Storage accounts (blob, queue, table and file)
- Connection strings for databases and Azure Cache for Redis

Endpoint namespaces can also be used in the output of a template as information for the user when the deployment completes. The following are common examples:

- Storage accounts (blob, queue, table and file)
- Connection strings (MySql, SQLServer, SQLAzure, Custom, NotificationHub, ServiceBus, EventHub, ApiHub, DocDb, RedisCache, PostgreSQL)
- Traffic Manager
- domainNameLabel of a public IP address
- Cloud services

In general, avoid hardcoded endpoints in a template. The best practice is to use the reference template function to retrieve the endpoints dynamically. For example, the endpoint most commonly hardcoded is the endpoint namespace for storage accounts. Each storage account has a unique FQDN that is constructed by concatenating the name of the storage account with the endpoint namespace. A blob storage account named mystorageaccount1 results in different FQDNs depending on the cloud:

- `mystorageaccount1.blob.core.windows.net` when created on the global Azure cloud.
- `mystorageaccount1.blob.core.chinacloudapi.cn` when created in the Azure China 21Vianet cloud.

The following reference template function retrieves the endpoint namespace from the storage resource provider:

JSON

```
"diskUri":"[concat(reference(resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))).primaryEndpoints.blob,
'container/myosdisk.vhd')]"
```

By replacing the hardcoded value of the storage account endpoint with the `reference` template function, you can use the same template to deploy to different environments successfully without making any changes to the endpoint reference.

Refer to existing resources by unique ID

You can also refer to an existing resource from the same or another resource group, and within the same subscription or another subscription, within the same tenant in the same cloud. To retrieve the resource properties, you must use the unique identifier for the resource itself. The `resourceId` template function retrieves the unique ID of a resource such as SQL Server as the following code shows:

JSON

```
"outputs": {
  "resourceId":{
    "type": "string",
    "value": "[resourceId('otherResourceGroup', 'Microsoft.Sql/servers',
parameters('serverName'))]"
  }
}
```

You can then use the `resourceId` function inside the `reference` template function to retrieve the properties of a database. The return object contains the `fullyQualifiedDomainName` property that holds the full endpoint value. This value is retrieved at runtime and provides the cloud environment-specific endpoint namespace. To define the connection string without hardcoding the endpoint namespace, you can refer to the property of the return object directly in the connection string as shown:

JSON

```
"[concat('Server=tcp:', reference(resourceId('sql', 'Microsoft.Sql/servers',
parameters('test')), '2015-05-01-preview').fullyQualifiedDomainName,
',1433;Initial Catalog=', parameters('database'),';User ID=',
parameters('username'), ';Password=', parameters('pass'),
';Encrypt=True;')]"
```

Consider resource properties

Specific resources within Azure Stack environments have unique properties you must consider in your template.

Ensure VM images are available

Azure provides a rich selection of VM images. These images are created and prepared for deployment by Microsoft and partners. The images form the foundation for VMs on the platform. However, a cloud-consistent template should refer to available parameters only — in particular, the publisher, offer, and SKU of the VM images available to the global Azure, Azure sovereign clouds, or an Azure Stack solution.

To retrieve a list of the available VM images in a location, run the following Azure CLI command:

Azure CLI

```
az vm image list -all
```

You can retrieve the same list with the Azure PowerShell cmdlet Get-AzureRmVMImagePublisher and specify the location you want with the `-Location` parameter. For example:

Azure PowerShell

```
Get-AzureRmVMImagePublisher -Location "West Europe" | Get-
AzureRmVMImageOffer | Get-AzureRmVMImageSku | Get-AzureRmVMImage
```

This command takes a couple of minutes to return all the available images in the West Europe region of the global Azure cloud.

If you made these VM images available to Azure Stack, all the available storage would be consumed. To accommodate even the smallest scale unit, Azure Stack allows you to select the images you want to add to an environment.

The following code sample shows a consistent approach to refer to the publisher, offer, and SKU parameters in your ARM templates:

JSON

```
"storageProfile": {
    "imageReference": {
    "publisher": "MicrosoftWindowsServer",
    "offer": "WindowsServer",
    "sku": "2016-Datacenter",
    "version": "latest"
    }
}
```

Check local VM sizes

To develop your template for cloud consistency, you need to make sure the VM size you want is available in all target environments. VM sizes are a grouping of performance characteristics and capabilities. Some VM sizes depend on the hardware that the VM runs on. For example, if you want to deploy a GPU-optimized VM, the hardware that runs the hypervisor needs to have the hardware GPUs.

When Microsoft introduces a new size of VM that has certain hardware dependencies, the VM size is usually made available first in a small subset of regions in the Azure cloud. Later, it is made available to other regions and clouds. To make sure the VM size exists in each cloud you deploy to, you can retrieve the available sizes with the following Azure CLI command:

Azure CLI

```
az vm list-sizes --location "West Europe"
```

For Azure PowerShell, use:

Azure PowerShell

```
Get-AzureRmVMSize -Location "West Europe"
```

For a full list of available services, see Products available by region.

Check use of Azure Managed Disks in Azure Stack

Managed disks handle the storage for an Azure tenant. Instead of explicitly creating a storage account and specifying the URI for a virtual hard disk (VHD), you can use managed disks to implicitly perform these actions when you deploy a VM. Managed disks enhance availability by placing all the disks from VMs in the same availability set into different storage units. Additionally, existing VHDs can be converted from Standard to Premium storage with significantly less downtime.

Although managed disks are on the roadmap for Azure Stack, they are currently not supported. Until they are, you can develop cloud-consistent templates for Azure Stack by explicitly specifying VHDs using the `vhd` element in the template for the VM resource as shown:

JSON

```
"storageProfile": {
  "imageReference": {
    "publisher": "MicrosoftWindowsServer",
    "offer": "WindowsServer",
    "sku": "[parameters('windowsOSVersion')]",
    "version": "latest"
  },
  "osDisk": {
    "name": "osdisk",
    "vhd": {
      "uri": "
[concat(reference(resourceId('Microsoft.Storage/storageAccounts/',
variables('storageAccountName')), '2015-06-15').primaryEndpoints.blob,
'vhds/osdisk.vhd')]"
    },
    "caching": "ReadWrite",
    "createOption": "FromImage"
  }
}
```

In contrast, to specify a managed disk configuration in a template, remove the `vhd` element from the disk configuration.

JSON

```
"storageProfile": {
  "imageReference": {
    "publisher": "MicrosoftWindowsServer",
    "offer": "WindowsServer",
    "sku": "[parameters('windowsOSVersion')]",
    "version": "latest"
  },
  "osDisk": {
    "caching": "ReadWrite",
    "createOption": "FromImage"
  }
}
```

The same changes also apply data disks.

Verify that VM extensions are available in Azure Stack

Another consideration for cloud consistency is the use of virtual machine extensions to configure the resources inside a VM. Not all VM extensions are available in Azure Stack. A template can specify the resources dedicated to the VM extension, creating dependencies and conditions within the template.

For example, if you want to configure a VM running Microsoft SQL Server, the VM extension can configure SQL Server as part the template deployment. Consider what happens if the deployment template also contains an application server configured to create a database on the VM running SQL Server. Besides also using a VM extension for the application servers, you can configure the dependency of the application server on the successful return of the SQL Server VM extension resource. This approach ensures the VM running SQL Server is configured and available when the application server is instructed to create the database.

The declarative approach of the template allows you to define the end state of the resources and their inter-dependencies, while the platform takes care of the logic required for the dependencies.

Check that VM extensions are available

Many types of VM extensions exist. When developing template for cloud consistency, make sure to use only the extensions that are available in all the regions the template targets.

To retrieve a list of the VM extensions that are available for a specific region (in this example, `myLocation`), run the following Azure CLI command:

Azure CLI

```
az vm extension image list --location myLocation
```

You can also execute the Azure PowerShell Get-AzureRmVmImagePublisher cmdlet and use `-Location` to specify the location of the virtual machine image. For example:

Azure PowerShell

```
Get-AzureRmVmImagePublisher -Location myLocation | Get-
AzureRmVMExtensionImageType | Get-AzureRmVMExtensionImage | Select Type,
Version
```

Ensure that versions are available

Since VM extensions are first-party Resource Manager resources, they have their own API versions. As the following code shows, the VM extension type is a nested resource in the Microsoft.Compute resource provider.

JSON

```
{
    "type": "Microsoft.Compute/virtualMachines/extensions",
    "apiVersion": "2015-06-15",
    "name": "myExtension",
    "location": "[parameters('location')]",
    ...
```

The API version of the VM extension resource must be present in all the locations you plan to target with your template. The location dependency works like the resource provider API version availability discussed earlier in the "Verify the version of all resource types" section.

To retrieve a list of the available API versions for the VM extension resource, use the Get-AzureRmResourceProvider cmdlet with the **Microsoft.Compute** resource provider as shown:

Azure PowerShell

```
Get-AzureRmResourceProvider -ProviderNamespace "Microsoft.Compute" | Select-Object -ExpandProperty ResourceTypes | Select ResourceTypeName, Locations, ApiVersions | where {$_.ResourceTypeName -eq "virtualMachines/extensions"}
```

You can also use VM extensions in virtual machine scale sets. The same location conditions apply. To develop your template for cloud consistency, make sure the API versions are available in all the locations you plan on deploying to. To retrieve the API versions of the VM extension resource for scale sets, use the same cmdlet as before, but specify the virtual machine scale sets resource type as shown:

Azure PowerShell

```
Get-AzureRmResourceProvider -ProviderNamespace "Microsoft.Compute" | Select-Object -ExpandProperty ResourceTypes | Select ResourceTypeName, Locations, ApiVersions | where {$_.ResourceTypeName -eq "virtualMachineScaleSets/extensions"}
```

Each specific extension is also versioned. This version is shown in the `typeHandlerVersion` property of the VM extension. Make sure that the version specified in the `typeHandlerVersion` element of your template's VM extensions are available in the locations where you plan to deploy the template. For example, the following code specifies version 1.7:

JSON

```
{
    "type": "extensions",
```

```
    "apiVersion": "2016-03-30",
    "name": "MyCustomScriptExtension",
    "location": "[parameters('location')]",
    "dependsOn": [
        "[concat('Microsoft.Compute/virtualMachines/myVM', copyindex())]"
    ],
    "properties": {
        "publisher": "Microsoft.Compute",
        "type": "CustomScriptExtension",
        "typeHandlerVersion": "1.7",
        ...
```

To retrieve a list of the available versions for a specific VM extension, use the Get-AzureRmVMExtensionImage cmdlet. The following example retrieves the available versions for the PowerShell DSC (Desired State Configuration) VM extension from **myLocation**:

Azure PowerShell

```
Get-AzureRmVMExtensionImage -Location myLocation -PublisherName
Microsoft.PowerShell -Type DSC | FT
```

To get a list of publishers, use the Get-AzureRmVmImagePublisher command. To request type, use the Get-AzureRmVMExtensionImageType commend.

Tips for testing and automation

It's a challenge to keep track of all related settings, capabilities, and limitations while authoring a template. The common approach is to develop and test templates against a single cloud before other locations are targeted. However, the earlier that tests are performed in the authoring process, the less troubleshooting and code rewriting your development team will have to do. Deployments that fail because of location dependencies can be time-consuming to troubleshoot. That's why we recommend automated testing as early as possible in the authoring cycle. Ultimately, you'll need less development time and fewer resources, and your cloud-consistent artifacts will become even more valuable.

The following image shows a typical example of a development process for a team using an integrated development environment (IDE). At different stages in the timeline, different test types are executed. Here, two developers are working on the same solution, but this scenario applies equally to a single developer or a large team. Each developer typically creates a local copy of a central repository, enabling each one to work on the local copy without impacting the others who may be working on the same files.

Consider the following tips for testing and automation:

- Do make use of testing tools. For example, Visual Studio Code and Visual Studio include IntelliSense and other features that can help you validate your templates.
- To improve the code quality during development on the local IDE, perform static code analysis with unit tests and integration tests.
- For an even better experience during initial development, unit tests and integration tests should only warn when an issue is found and proceed with the tests. That way, you can identify the issues to addressed and prioritize the order of the changes, also referred to as test-driven deployment (TDD).
- Be aware that some tests can be performed without being connected to Azure Resource Manager. Others, like testing template deployment, require Resource Manager to perform certain actions that cannot be performed offline.
- Testing a deployment template against the validation API isn't equal to an actual deployment. Also, even if you deploy a template from a local file, any references to nested templates in the template are retrieved by Resource Manager directly, and artifacts referenced by VM extensions are retrieved by the VM agent running inside the deployed VM.

Next steps

- Azure Resource Manager template considerations
- Best practices for ARM templates

Azure Resource Manager deployment modes

Article • 04/09/2023

When deploying your resources, you specify that the deployment is either an incremental update or a complete update. The difference between these two modes is how Resource Manager handles existing resources in the resource group that aren't in the template.

For both modes, Resource Manager tries to create all resources specified in the template. If the resource already exists in the resource group and its settings are unchanged, no operation is taken for that resource. If you change the property values for a resource, the resource is updated with those new values. If you try to update the location or type of an existing resource, the deployment fails with an error. Instead, deploy a new resource with the location or type that you need.

The default mode is incremental.

Complete mode

In complete mode, Resource Manager **deletes** resources that exist in the resource group but aren't specified in the template.

> ⓘ **Note**
>
> Always use the **what-if operation** before deploying a template in complete mode. What-if shows you which resources will be created, deleted, or modified. Use what-if to avoid unintentionally deleting resources.

If your template includes a resource that isn't deployed because condition evaluates to false, the result depends on which REST API version you use to deploy the template. If you use a version earlier than 2019-05-10, the resource **isn't deleted**. With 2019-05-10 or later, the resource **is deleted**. The latest versions of Azure PowerShell and Azure CLI delete the resource.

Be careful using complete mode with copy loops. Any resources that aren't specified in the template after resolving the copy loop are deleted.

If you deploy to more than one resource group in a template, resources in the resource group specified in the deployment operation are eligible to be deleted. Resources in the

secondary resource groups aren't deleted.

There are some differences in how resource types handle complete mode deletions. Parent resources are automatically deleted when not in a template that's deployed in complete mode. Some child resources aren't automatically deleted when not in the template. However, these child resources are deleted if the parent resource is deleted.

For example, if your resource group contains a DNS zone (`Microsoft.Network/dnsZones` resource type) and a CNAME record (`Microsoft.Network/dnsZones/CNAME` resource type), the DNS zone is the parent resource for the CNAME record. If you deploy with complete mode and don't include the DNS zone in your template, the DNS zone and the CNAME record are both deleted. If you include the DNS zone in your template but don't include the CNAME record, the CNAME isn't deleted.

For a list of how resource types handle deletion, see Deletion of Azure resources for complete mode deployments.

If the resource group is locked, complete mode doesn't delete the resources.

> **Note**
>
> Only root-level templates support the complete deployment mode. For **linked or nested templates**, you must use incremental mode.
>
> **Subscription level deployments** don't support complete mode.
>
> Currently, the portal doesn't support complete mode.

Incremental mode

In incremental mode, Resource Manager **leaves unchanged** resources that exist in the resource group but aren't specified in the template. Resources in the template **are added** to the resource group.

> **Important**
>
> When redeploying an existing resource in incremental mode, all properties are reapplied. The **properties aren't incrementally added**. A common misunderstanding is to think properties that aren't specified in the template are left unchanged. If you don't specify certain properties, Resource Manager interprets the deployment as overwriting those values. Properties that aren't included in the template are reset to the default values. Specify all non-default values for the

resource, not just the ones you're updating. The resource definition in the template always contains the final state of the resource. It can't represent a partial update to an existing resource.

> ⚠ **Warning**
>
> In rare cases, you can specify properties either on a resource or on one of its child resources. Two common examples are **subnets on virtual networks** and **site configuration values for web apps**. In these cases, you must handle incremental updates carefully.
>
> For subnets, specify the values through the `subnets` property on the **Microsoft.Network/virtualNetworks** resource. Don't define the values through the child resource **Microsoft.Network/virtualNetworks/subnets**. As long as the subnets are defined on the virtual network, you can redeploy the virtual network and not lose the subnets.
>
> For site configuration values, the values are implemented in the child resource type `Microsoft.Web/sites/config`. If you redeploy the web app and specify an empty object for the site configuration values, the child resource isn't updated. However, if you provide new site configuration values, the child resource type is updated.

Example result

To illustrate the difference between incremental and complete modes, consider the following scenario.

Resource Group contains:

- Resource A
- Resource B
- Resource C

Template contains:

- Resource A
- Resource B
- Resource D

When deployed in **incremental** mode, the resource group has:

- Resource A

- Resource B
- Resource C
- Resource D

When deployed in **complete** mode, Resource C is deleted. The resource group has:

- Resource A
- Resource B
- Resource D

Set deployment mode

To set the deployment mode when deploying with PowerShell, use the `Mode` parameter.

Azure PowerShell

```
New-AzResourceGroupDeployment `
  -Mode Complete `
  -Name ExampleDeployment `
  -ResourceGroupName ExampleResourceGroup `
  -TemplateFile c:\MyTemplates\storage.json
```

To set the deployment mode when deploying with Azure CLI, use the `mode` parameter.

Azure CLI

```
az deployment group create \
  --mode Complete \
  --name ExampleDeployment \
  --resource-group ExampleResourceGroup \
  --template-file storage.json
```

The following example shows a linked template set to incremental deployment mode:

JSON

```
"resources": [
  {
    "type": "Microsoft.Resources/deployments",
    "apiVersion": "2020-10-01",
    "name": "linkedTemplate",
    "properties": {
      "mode": "Incremental",
          <nested-template-or-external-template>
    }
  }
]
```

Next steps

- To learn about creating Resource Manager templates, see Understand the structure and syntax of ARM templates.
- To learn about deploying resources, see Deploy resources with ARM templates and Azure PowerShell.
- To view the operations for a resource provider, see Azure REST API.

Automatic deletions from deployment history

Article • 03/09/2023

Every time you deploy a template, information about the deployment is written to the deployment history. Each resource group is limited to 800 deployments in its deployment history.

Azure Resource Manager automatically deletes deployments from your history as you near the limit. Automatic deletion is a change from past behavior. Previously, you had to manually delete deployments from the deployment history to avoid getting an error. This change was implemented on August 6, 2020.

Automatic deletions are supported for resource group and subscription deployments. Currently, deployments in the history for management group and tenant deployments aren't automatically deleted.

> **Note**
>
> Deleting a deployment from the history doesn't affect any of the resources that were deployed.

When deployments are deleted

Deployments are deleted from your history when you exceed 700 deployments. Azure Resource Manager deletes deployments until the history is down to 600. The oldest deployments are always deleted first.

> **Important**
>
> If your resource group is already at the 800 limit, your next deployment fails with an error. The automatic deletion process starts immediately. You can try your

deployment again after a short wait.

In addition to deployments, you also trigger deletions when you run the what-if operation or validate a deployment.

When you give a deployment the same name as one in the history, you reset its place in the history. The deployment moves to the most recent place in the history. You also reset a deployment's place when you roll back to that deployment after an error.

Remove locks that block deletions

If you have a CanNotDelete lock on a resource group, the deployments for that resource group can't be deleted. You must remove the lock to take advantage of automatic deletions in the deployment history.

To use PowerShell to delete a lock, run the following commands:

Azure PowerShell

```
$lockId = (Get-AzResourceLock -ResourceGroupName lockedRG).LockId
Remove-AzResourceLock -LockId $lockId
```

To use Azure CLI to delete a lock, run the following commands:

Azure CLI

```
lockid=$(az lock show --resource-group lockedRG --name deleteLock --output
tsv --query id)
az lock delete --ids $lockid
```

Required permissions

The deletions are requested under the identity of the user who deployed the template. To delete deployments, the user must have access to the **Microsoft.Resources/deployments/delete** action. If the user doesn't have the required permissions, deployments aren't deleted from the history.

If the current user doesn't have the required permissions, automatic deletion is attempted again during the next deployment.

Opt out of automatic deletions

You can opt out of automatic deletions from the history. **Use this option only when you want to manage the deployment history yourself.** The limit of 800 deployments in the history is still enforced. If you exceed 800 deployments, you'll receive an error and your deployment will fail.

To disable automatic deletions, register the `Microsoft.Resources/DisableDeploymentGrooming` feature flag. When you register the feature flag, you opt out of automatic deletions for the entire Azure subscription. You can't opt out for only a particular resource group. To reenable automatic deletions, unregister the feature flag.

PowerShell

For PowerShell, use Register-AzProviderFeature.

Azure PowerShell

```
Register-AzProviderFeature -ProviderNamespace Microsoft.Resources -
FeatureName DisableDeploymentGrooming
```

To see the current status of your subscription, use:

Azure PowerShell

```
Get-AzProviderFeature -ProviderNamespace Microsoft.Resources -
FeatureName DisableDeploymentGrooming
```

To reenable automatic deletions, use Azure REST API or Azure CLI.

Next steps

- To learn about viewing the deployment history, see View deployment history with Azure Resource Manager.

Deletion of Azure resources for complete mode deployments

Article • 04/09/2023

This article describes how resource types handle deletion when not in a template that is deployed in complete mode.

The resource types marked with **Yes** are deleted when the type isn't in the template deployed with complete mode.

The resource types marked with **No** aren't automatically deleted when not in the template; however, they're deleted if the parent resource is deleted. For a full description of the behavior, see Azure Resource Manager deployment modes.

If you deploy to more than one resource group in a template, resources in the resource group specified in the deployment operation are eligible to be deleted. Resources in the secondary resource groups aren't deleted.

The resources are listed by resource provider namespace. To match a resource provider namespace with its Azure service name, see Resource providers for Azure services.

> **Note**
>
> Always use the **what-if operation** before deploying a template in complete mode. What-if shows you which resources will be created, deleted, or modified. Use what-if to avoid unintentionally deleting resources.

Microsoft.AAD

Resource type	Complete mode deletion
DomainServices	Yes
DomainServices / oucontainer	No

microsoft.aadiam

Resource type	Complete mode deletion
azureADMetrics	Yes

Resource type	Complete mode deletion
diagnosticSettings	No
diagnosticSettingsCategories	No
privateLinkForAzureAD	Yes
tenants	Yes

Microsoft.ADHybridHealthService

Resource type	Complete mode deletion
aadsupportcases	No
addsservices	No
agents	No
anonymousapiusers	No
configuration	No
logs	No
reports	No
servicehealthmetrics	No
services	No

Microsoft.Advisor

Resource type	Complete mode deletion
advisorScore	No
configurations	No
generateRecommendations	No
metadata	No
predict	No
recommendations	No
suppressions	No

Microsoft.AgFoodPlatform

Resource type	Complete mode deletion
farmBeats	Yes
farmBeats / eventGridFilters	No
farmBeats / extensions	No
farmBeats / solutions	No
farmBeatsExtensionDefinitions	No
farmBeatsSolutionDefinitions	No

Microsoft.AlertsManagement

Resource type	Complete mode deletion
actionRules	Yes
alerts	No
alertsMetaData	No
migrateFromSmartDetection	No
smartDetectorAlertRules	Yes
smartGroups	No

Microsoft.AnalysisServices

Resource type	Complete mode deletion
servers	Yes

Microsoft.AnyBuild

Resource type	Complete mode deletion
clusters	Yes

Microsoft.ApiManagement

Resource type	Complete mode deletion
deletedServices	No
getDomainOwnershipIdentifier	No
reportFeedback	No
service	Yes
service / eventGridFilters	No
validateServiceName	No

Microsoft.App

Resource type	Complete mode deletion
connectedEnvironments	Yes
connectedEnvironments / certificates	Yes
containerApps	Yes
managedEnvironments	Yes
managedEnvironments / certificates	Yes

Microsoft.AppAssessment

Resource type	Complete mode deletion
migrateProjects	Yes
migrateProjects / assessments	No
migrateProjects / assessments / assessedApplications	No
migrateProjects / assessments / assessedApplications / machines	No
migrateProjects / assessments / assessedMachines	No

Resource type	Complete mode deletion
migrateProjects / assessments / assessedMachines / applications	No
migrateProjects / assessments / machinesToAssess	No
migrateProjects / sites	No
migrateProjects / sites / applianceConfigurations	No
migrateProjects / sites / machines	No

Microsoft.AppConfiguration

Resource type	Complete mode deletion
configurationStores	Yes
configurationStores / eventGridFilters	No
configurationStores / keyValues	No
configurationStores / replicas	No
deletedConfigurationStores	No

Microsoft.AppPlatform

Resource type	Complete mode deletion
runtimeVersions	No
Spring	Yes
Spring / apps	No
Spring / apps / deployments	No

Microsoft.Attestation

Resource type	Complete mode deletion
attestationProviders	Yes

Resource type	Complete mode deletion
defaultProviders	No

Microsoft.Authorization

Resource type	Complete mode deletion
accessReviewHistoryDefinitions	No
classicAdministrators	No
dataAliases	No
dataPolicyManifests	No
denyAssignments	No
diagnosticSettings	No
diagnosticSettingsCategories	No
elevateAccess	No
eligibleChildResources	No
locks	No
policyAssignments	No
policyDefinitions	No
policyExemptions	No
policySetDefinitions	No
privateLinkAssociations	No
resourceManagementPrivateLinks	Yes
roleAssignmentApprovals	No
roleAssignments	No
roleAssignmentScheduleInstances	No
roleAssignmentScheduleRequests	No
roleAssignmentSchedules	No
roleDefinitions	No

Resource type	Complete mode deletion
roleEligibilityScheduleInstances	No
roleEligibilityScheduleRequests	No
roleEligibilitySchedules	No
roleManagementPolicies	No
roleManagementPolicyAssignments	No

Microsoft.Automanage

Resource type	Complete mode deletion
accounts	Yes
bestPractices	No
bestPractices / versions	No
configurationProfileAssignmentIntents	No
configurationProfileAssignments	No
configurationProfilePreferences	Yes
configurationProfiles	Yes
configurationProfiles / versions	Yes
patchJobConfigurations	Yes
patchJobConfigurations / patchJobs	No
patchSchedules	Yes
patchSchedules / associations	Yes
patchTiers	Yes
servicePrincipals	No

Microsoft.Automation

Resource type	Complete mode deletion
automationAccounts	Yes

Resource type	Complete mode deletion
automationAccounts / agentRegistrationInformation	No
automationAccounts / configurations	Yes
automationAccounts / hybridRunbookWorkerGroups	No
automationAccounts / hybridRunbookWorkerGroups / hybridRunbookWorkers	No
automationAccounts / jobs	No
automationAccounts / privateEndpointConnectionProxies	No
automationAccounts / privateEndpointConnections	No
automationAccounts / privateLinkResources	No
automationAccounts / runbooks	Yes
automationAccounts / runtimes	Yes
automationAccounts / softwareUpdateConfigurationMachineRuns	No
automationAccounts / softwareUpdateConfigurationRuns	No
automationAccounts / softwareUpdateConfigurations	No
automationAccounts / webhooks	No
deletedAutomationAccounts	No

Microsoft.AutonomousDevelopmentPlatform

Resource type	Complete mode deletion
accounts	Yes
accounts / datapools	No
workspaces	Yes

Resource type	Complete mode deletion
workspaces / eventgridfilters	No

Microsoft.AutonomousSystems

Resource type	Complete mode deletion
workspaces	Yes
workspaces / validateCreateRequest	No

Microsoft.AVS

Resource type	Complete mode deletion
privateClouds	Yes
privateClouds / addons	No
privateClouds / authorizations	No
privateClouds / cloudLinks	No
privateClouds / clusters	No
privateClouds / clusters / datastores	No
privateClouds / clusters / placementPolicies	No
privateClouds / clusters / virtualMachines	No
privateClouds / globalReachConnections	No
privateClouds / hcxEnterpriseSites	No
privateClouds / scriptExecutions	No
privateClouds / scriptPackages	No
privateClouds / scriptPackages / scriptCmdlets	No
privateClouds / workloadNetworks	No
privateClouds / workloadNetworks / dhcpConfigurations	No

Resource type	Complete mode deletion
privateClouds / workloadNetworks / dnsServices	No
privateClouds / workloadNetworks / dnsZones	No
privateClouds / workloadNetworks / gateways	No
privateClouds / workloadNetworks / portMirroringProfiles	No
privateClouds / workloadNetworks / publicIPs	No
privateClouds / workloadNetworks / segments	No
privateClouds / workloadNetworks / virtualMachines	No
privateClouds / workloadNetworks / vmGroups	No

Microsoft.AzureActiveDirectory

Resource type	Complete mode deletion
b2cDirectories	Yes
b2ctenants	No
ciamDirectories	Yes
guestUsages	Yes

Microsoft.AzureArcData

Resource type	Complete mode deletion
DataControllers	Yes
DataControllers / ActiveDirectoryConnectors	No
PostgresInstances	Yes
SqlManagedInstances	Yes
SqlServerInstances	Yes
SqlServerInstances / Databases	Yes

Microsoft.AzureCIS

Resource type	Complete mode deletion
autopilotEnvironments	Yes
dstsServiceAccounts	Yes
dstsServiceClientIdentities	Yes

Microsoft.AzureData

Resource type	Complete mode deletion
sqlServerRegistrations	Yes
sqlServerRegistrations / sqlServers	No

Microsoft.AzureScan

Resource type	Complete mode deletion
scanningAccounts	Yes

Microsoft.AzureSphere

Resource type	Complete mode deletion
catalogs	Yes
catalogs / certificates	No
catalogs / deployments	No
catalogs / devices	No
catalogs / images	No
catalogs / products	No
catalogs / products / devicegroups	No

Microsoft.AzureStack

Resource type	Complete mode deletion
cloudManifestFiles	No
generateDeploymentLicense	No
linkedSubscriptions	Yes
registrations	Yes
registrations / customerSubscriptions	No
registrations / products	No

Microsoft.AzureStackHCI

Resource type	Complete mode deletion
clusters	Yes
clusters / arcSettings	No
clusters / arcSettings / extensions	No
clusters / offers	No
clusters / publishers	No
clusters / publishers / offers	No
clusters / updates	No
clusters / updates / updateRuns	No
clusters / updateSummaries	No
galleryImages	Yes
marketplaceGalleryImages	Yes
networkinterfaces	Yes
registeredSubscriptions	No
storageContainers	Yes
virtualharddisks	Yes
virtualmachines	Yes
virtualMachines / extensions	Yes

Resource type	Complete mode deletion
virtualmachines / hybrididentitymetadata	No
virtualnetworks	Yes

Microsoft.BackupSolutions

Resource type	Complete mode deletion
VMwareApplications	Yes

Microsoft.BareMetalInfrastructure

Resource type	Complete mode deletion
bareMetalInstances	Yes

Microsoft.Batch

Resource type	Complete mode deletion
batchAccounts	Yes
batchAccounts / certificates	No
batchAccounts / detectors	No
batchAccounts / pools	No

Microsoft.Billing

Resource type	Complete mode deletion
billingAccounts	No
billingAccounts / agreements	No
billingAccounts / appliedReservationOrders	No
billingAccounts / associatedTenants	No
billingAccounts / billingPermissions	No

Resource type	Complete mode deletion
billingAccounts / billingProfiles	No
billingAccounts / billingProfiles / billingPermissions	No
billingAccounts / billingProfiles / billingRoleAssignments	No
billingAccounts / billingProfiles / billingRoleDefinitions	No
billingAccounts / billingProfiles / billingSubscriptions	No
billingAccounts / billingProfiles / createBillingRoleAssignment	No
billingAccounts / billingProfiles / customers	No
billingAccounts / billingProfiles / instructions	No
billingAccounts / billingProfiles / invoices	No
billingAccounts / billingProfiles / invoices / pricesheet	No
billingAccounts / billingProfiles / invoices / transactions	No
billingAccounts / billingProfiles / invoiceSections	No
billingAccounts / billingProfiles / invoiceSections / billingPermissions	No
billingAccounts / billingProfiles / invoiceSections / billingRoleAssignments	No
billingAccounts / billingProfiles / invoiceSections / billingRoleDefinitions	No
billingAccounts / billingProfiles / invoiceSections / billingSubscriptions	No
billingAccounts / billingProfiles / invoiceSections / createBillingRoleAssignment	No
billingAccounts / billingProfiles / invoiceSections / initiateTransfer	No

Resource type	Complete mode deletion
billingAccounts / billingProfiles / invoiceSections / products	No
billingAccounts / billingProfiles / invoiceSections / products / transfer	No
billingAccounts / billingProfiles / invoiceSections / products / updateAutoRenew	No
billingAccounts / billingProfiles / invoiceSections / transactions	No
billingAccounts / billingProfiles / invoiceSections / transfers	No
billingAccounts / billingProfiles / invoiceSections / validateDeleteInvoiceSectionEligibility	No
billingAccounts / billingProfiles / paymentMethodLinks	No
billingAccounts / billingProfiles / paymentMethods	No
billingAccounts / billingProfiles / policies	No
billingAccounts / billingProfiles / pricesheet	No
billingAccounts / billingProfiles / products	No
billingAccounts / billingProfiles / reservations	No
billingAccounts / billingProfiles / transactions	No
billingAccounts / billingProfiles / validateDeleteBillingProfileEligibility	No
billingAccounts / billingProfiles / validateDetachPaymentMethodEligibility	No
billingAccounts / billingRoleAssignments	No
billingAccounts / billingRoleDefinitions	No
billingAccounts / billingSubscriptionAliases	No
billingAccounts / billingSubscriptions	No

Resource type	Complete mode deletion
billingAccounts / billingSubscriptions / elevateRole	No
billingAccounts / billingSubscriptions / invoices	No
billingAccounts / createBillingRoleAssignment	No
billingAccounts / customers	No
billingAccounts / customers / billingPermissions	No
billingAccounts / customers / billingRoleAssignments	No
billingAccounts / customers / billingRoleDefinitions	No
billingAccounts / customers / billingSubscriptions	No
billingAccounts / customers / createBillingRoleAssignment	No
billingAccounts / customers / initiateTransfer	No
billingAccounts / customers / policies	No
billingAccounts / customers / products	No
billingAccounts / customers / transactions	No
billingAccounts / customers / transfers	No
billingAccounts / customers / transferSupportedAccounts	No
billingAccounts / departments	No
billingAccounts / departments / billingPermissions	No
billingAccounts / departments / billingRoleAssignments	No
billingAccounts / departments / billingRoleDefinitions	No
billingAccounts / departments / billingSubscriptions	No

Resource type	Complete mode deletion
billingAccounts / departments / enrollmentAccounts	No
billingAccounts / enrollmentAccounts	No
billingAccounts / enrollmentAccounts / billingPermissions	No
billingAccounts / enrollmentAccounts / billingRoleAssignments	No
billingAccounts / enrollmentAccounts / billingRoleDefinitions	No
billingAccounts / enrollmentAccounts / billingSubscriptions	No
billingAccounts / invoices	No
billingAccounts / invoices / summary	No
billingAccounts / invoices / transactions	No
billingAccounts / invoices / transactionSummary	No
billingAccounts / invoiceSections	No
billingAccounts / invoiceSections / billingSubscriptions	No
billingAccounts / invoiceSections / billingSubscriptions / transfer	No
billingAccounts / invoiceSections / elevate	No
billingAccounts / invoiceSections / initiateTransfer	No
billingAccounts / invoiceSections / products	No
billingAccounts / invoiceSections / products / transfer	No
billingAccounts / invoiceSections / products / updateAutoRenew	No
billingAccounts / invoiceSections / transactions	No
billingAccounts / invoiceSections / transfers	No

Resource type	Complete mode deletion
billingAccounts / lineOfCredit	No
billingAccounts / notificationContacts	No
billingAccounts / payableOverage	No
billingAccounts / paymentMethods	No
billingAccounts / payNow	No
billingAccounts / permissionRequests	No
billingAccounts / policies	No
billingAccounts / products	No
billingAccounts / promotionalCredits	No
billingAccounts / reservationOrders	No
billingAccounts / reservationOrders / reservations	No
billingAccounts / reservations	No
billingAccounts / savingsPlanOrders	No
billingAccounts / savingsPlanOrders / savingsPlans	No
billingAccounts / savingsPlans	No
billingAccounts / transactions	No
billingPeriods	No
billingPermissions	No
billingProperty	No
billingRoleAssignments	No
billingRoleDefinitions	No
createBillingRoleAssignment	No
departments	No
enrollmentAccounts	No
invoices	No

Resource type	Complete mode deletion
paymentMethods	No
permissionRequests	No
promotionalCredits	No
promotions	No
transfers	No
transfers / acceptTransfer	No
transfers / declineTransfer	No
transfers / validateTransfer	No
validateAddress	No

Microsoft.BillingBenefits

Resource type	Complete mode deletion
calculateMigrationCost	No
savingsPlanOrderAliases	No
savingsPlanOrders	No
savingsPlanOrders / savingsPlans	No
savingsPlans	No
validate	No

Microsoft.Bing

Resource type	Complete mode deletion
accounts	Yes
accounts / usages	No
registeredSubscriptions	No

Microsoft.BlockchainTokens

Resource type	Complete mode deletion
TokenServices	Yes
TokenServices / BlockchainNetworks	No
TokenServices / Groups	No
TokenServices / Groups / Accounts	No
TokenServices / TokenTemplates	No

Microsoft.Blueprint

Resource type	Complete mode deletion
blueprintAssignments	No
blueprints	No
blueprints / artifacts	No
blueprints / versions	No
blueprints / versions / artifacts	No

Microsoft.BotService

Resource type	Complete mode deletion
botServices	Yes
botServices / channels	No
botServices / connections	No
botServices / privateEndpointConnectionProxies	No
botServices / privateEndpointConnections	No
botServices / privateLinkResources	No
hostSettings	No
languages	No
templates	No

Microsoft.Cache

Resource type	Complete mode deletion
Redis	Yes
Redis / EventGridFilters	No
Redis / privateEndpointConnectionProxies	No
Redis / privateEndpointConnectionProxies / validate	No
Redis / privateEndpointConnections	No
Redis / privateLinkResources	No
redisEnterprise	Yes
redisEnterprise / databases	No
RedisEnterprise / privateEndpointConnectionProxies	No
RedisEnterprise / privateEndpointConnectionProxies / validate	No
RedisEnterprise / privateEndpointConnections	No
RedisEnterprise / privateLinkResources	No

Microsoft.Capacity

Resource type	Complete mode deletion
appliedReservations	No
autoQuotaIncrease	No
calculateExchange	No
calculatePrice	No
calculatePurchasePrice	No
catalogs	No
commercialReservationOrders	No
exchange	No

Resource type	Complete mode deletion
ownReservations	No
placePurchaseOrder	No
reservationOrders	No
reservationOrders / calculateRefund	No
reservationOrders / merge	No
reservationOrders / reservations	No
reservationOrders / reservations / revisions	No
reservationOrders / return	No
reservationOrders / split	No
reservationOrders / swap	No
reservations	No
resourceProviders	No
resources	No
validateReservationOrder	No

Microsoft.Cascade

Resource type	Complete mode deletion
sites	Yes

Microsoft.Cdn

Resource type	Complete mode deletion
canMigrate	No
CdnWebApplicationFirewallManagedRuleSets	No
CdnWebApplicationFirewallPolicies	Yes
edgenodes	No
migrate	No

Resource type	Complete mode deletion
profiles	Yes
profiles / afdendpoints	Yes
profiles / afdendpoints / routes	No
profiles / customdomains	No
profiles / endpoints	Yes
profiles / endpoints / customdomains	No
profiles / endpoints / origingroups	No
profiles / endpoints / origins	No
profiles / origingroups	No
profiles / origingroups / origins	No
profiles / policies	No
profiles / rulesets	No
profiles / rulesets / rules	No
profiles / secrets	No
profiles / securitypolicies	No
validateProbe	No
validateSecret	No

Microsoft.CertificateRegistration

Resource type	Complete mode deletion
certificateOrders	Yes
certificateOrders / certificates	No
validateCertificateRegistrationInformation	No

Microsoft.ChangeAnalysis

Resource type	Complete mode deletion

Resource type	Complete mode deletion
changes	No
changeSnapshots	No
computeChanges	No
profile	No

Microsoft.Chaos

Resource type	Complete mode deletion
artifactSetDefinitions	No
artifactSetSnapshots	No
experiments	Yes
targets	No

Microsoft.ClassicCompute

Resource type	Complete mode deletion
capabilities	No
domainNames	Yes
domainNames / capabilities	No
domainNames / internalLoadBalancers	No
domainNames / serviceCertificates	No
domainNames / slots	No
domainNames / slots / roles	No
domainNames / slots / roles / metricDefinitions	No
domainNames / slots / roles / metrics	No
moveSubscriptionResources	No
operatingSystemFamilies	No
operatingSystems	No

Resource type	Complete mode deletion
quotas	No
resourceTypes	No
validateSubscriptionMoveAvailability	No
virtualMachines	Yes
virtualMachines / diagnosticSettings	No
virtualMachines / metricDefinitions	No
virtualMachines / metrics	No

Microsoft.ClassicInfrastructureMigrate

Resource type	Complete mode deletion
classicInfrastructureResources	No

Microsoft.ClassicNetwork

Resource type	Complete mode deletion
capabilities	No
expressRouteCrossConnections	No
expressRouteCrossConnections / peerings	No
gatewaySupportedDevices	No
networkSecurityGroups	Yes
quotas	No
reservedIps	Yes
virtualNetworks	Yes
virtualNetworks / remoteVirtualNetworkPeeringProxies	No
virtualNetworks / virtualNetworkPeerings	No

Microsoft.ClassicStorage

Resource type	Complete mode deletion
capabilities	No
disks	No
images	No
osImages	No
osPlatformImages	No
publicImages	No
quotas	No
storageAccounts	Yes
storageAccounts / blobServices	No
storageAccounts / fileServices	No
storageAccounts / metricDefinitions	No
storageAccounts / metrics	No
storageAccounts / queueServices	No
storageAccounts / services	No
storageAccounts / services / diagnosticSettings	No
storageAccounts / services / metricDefinitions	No
storageAccounts / services / metrics	No
storageAccounts / tableServices	No
storageAccounts / vmImages	No
vmImages	No

Microsoft.CloudTest

Resource type	Complete mode deletion
accounts	Yes

Resource type	Complete mode deletion
hostedpools	Yes
images	Yes
pools	Yes

Microsoft.CodeSigning

Resource type	Complete mode deletion
codeSigningAccounts	Yes
codeSigningAccounts / certificateProfiles	No

Microsoft.Codespaces

Resource type	Complete mode deletion
plans	Yes
registeredSubscriptions	No

Microsoft.CognitiveServices

Resource type	Complete mode deletion
accounts	Yes
accounts / networkSecurityPerimeterAssociationProxies	No
accounts / privateEndpointConnectionProxies	No
accounts / privateEndpointConnections	No
accounts / privateLinkResources	No
deletedAccounts	No

Microsoft.Commerce

Resource type	Complete mode deletion
RateCard	No
UsageAggregates	No

Microsoft.Communication

Resource type	Complete mode deletion
CommunicationServices	Yes
CommunicationServices / eventGridFilters	No
EmailServices	Yes
EmailServices / Domains	Yes
registeredSubscriptions	No

Microsoft.Compute

Resource type	Complete mode deletion
availabilitySets	Yes
capacityReservationGroups	Yes
capacityReservationGroups / capacityReservations	Yes
cloudServices	Yes
cloudServices / networkInterfaces	No
cloudServices / publicIPAddresses	No
cloudServices / roleInstances	No
cloudServices / roleInstances / networkInterfaces	No
cloudServices / roles	No
diskAccesses	Yes
diskEncryptionSets	Yes

Resource type	Complete mode deletion
disks	Yes
galleries	Yes
galleries / applications	Yes
galleries / applications / versions	Yes
galleries / images	Yes
galleries / images / versions	Yes
galleries / serviceArtifacts	Yes
hostGroups	Yes
hostGroups / hosts	Yes
images	Yes
proximityPlacementGroups	Yes
restorePointCollections	Yes
restorePointCollections / restorePoints	No
restorePointCollections / restorePoints / diskRestorePoints	No
sharedVMExtensions	Yes
sharedVMExtensions / versions	Yes
sharedVMImages	Yes
sharedVMImages / versions	Yes
snapshots	Yes
sshPublicKeys	Yes
virtualMachines	Yes
virtualMachines / applications	Yes
virtualMachines / extensions	Yes
virtualMachines / metricDefinitions	No
virtualMachines / runCommands	Yes
virtualMachineScaleSets	Yes

Resource type	Complete mode deletion
virtualMachineScaleSets / applications	No
virtualMachineScaleSets / extensions	No
virtualMachineScaleSets / networkInterfaces	No
virtualMachineScaleSets / publicIPAddresses	No
virtualMachineScaleSets / virtualMachines	No
virtualMachineScaleSets / virtualMachines / extensions	No
virtualMachineScaleSets / virtualMachines / networkInterfaces	No

Microsoft.ConfidentialLedger

Resource type	Complete mode deletion
Ledgers	Yes
ManagedCCF	Yes

Microsoft.Confluent

Resource type	Complete mode deletion
agreements	No
organizations	Yes
validations	No

Microsoft.ConnectedCache

Resource type	Complete mode deletion
CacheNodes	Yes
enterpriseCustomers	Yes
ispCustomers	Yes

Resource type	Complete mode deletion
ispCustomers / ispCacheNodes	Yes

microsoft.connectedopenstack

Resource type	Complete mode deletion
flavors	Yes
heatStacks	Yes
heatStackTemplates	Yes
images	Yes
keypairs	Yes
networkPorts	Yes
networks	Yes
openStackIdentities	Yes
securityGroupRules	Yes
securityGroups	Yes
subnets	Yes
virtualMachines	Yes
volumes	Yes
volumeSnapshots	Yes
volumeTypes	Yes

Microsoft.ConnectedVehicle

Resource type	Complete mode deletion
platformAccounts	Yes
registeredSubscriptions	No

Microsoft.ConnectedVMwarevSphere

Resource type	Complete mode deletion
clusters	Yes
datastores	Yes
hosts	Yes
resourcepools	Yes
VCenters	Yes
vcenters / inventoryitems	No
virtualmachines	Yes
VirtualMachines / AssessPatches	No
virtualmachines / extensions	Yes
virtualmachines / guestagents	No
virtualmachines / hybrididentitymetadata	No
VirtualMachines / InstallPatches	No
VirtualMachines / UpgradeExtensions	No
virtualmachinetemplates	Yes
virtualnetworks	Yes

Microsoft.Consumption

Resource type	Complete mode deletion
AggregatedCost	No
Balances	No
Budgets	No
Charges	No
CostTags	No
credits	No
events	No
Forecasts	No

Resource type	Complete mode deletion
lots	No
Marketplaces	No
Pricesheets	No
products	No
ReservationDetails	No
ReservationRecommendationDetails	No
ReservationRecommendations	No
ReservationSummaries	No
ReservationTransactions	No

Microsoft.ContainerInstance

Resource type	Complete mode deletion
containerGroupProfiles	Yes
containerGroups	Yes
containerScaleSets	Yes
serviceAssociationLinks	No

Microsoft.ContainerRegistry

Resource type	Complete mode deletion
registries	Yes
registries / agentPools	Yes
registries / builds	No
registries / builds / cancel	No
registries / builds / getLogLink	No
registries / buildTasks	Yes
registries / buildTasks / steps	No

Resource type	Complete mode deletion
registries / connectedRegistries	No
registries / connectedRegistries / deactivate	No
registries / eventGridFilters	No
registries / exportPipelines	No
registries / generateCredentials	No
registries / getBuildSourceUploadUrl	No
registries / GetCredentials	No
registries / importImage	No
registries / importPipelines	No
registries / pipelineRuns	No
registries / privateEndpointConnectionProxies	No
registries / privateEndpointConnectionProxies / validate	No
registries / privateEndpointConnections	No
registries / privateLinkResources	No
registries / queueBuild	No
registries / regenerateCredential	No
registries / regenerateCredentials	No
registries / replications	Yes
registries / runs	No
registries / runs / cancel	No
registries / scheduleRun	No
registries / scopeMaps	No
registries / taskRuns	No
registries / tasks	Yes
registries / tokens	No
registries / updatePolicies	No

Resource type	Complete mode deletion
registries / webhooks	Yes
registries / webhooks / getCallbackConfig	No
registries / webhooks / ping	No

Microsoft.ContainerService

Resource type	Complete mode deletion
containerServices	Yes
fleetMemberships	No
fleets	Yes
fleets / members	No
managedClusters	Yes
ManagedClusters / eventGridFilters	No
managedclustersnapshots	Yes
snapshots	Yes

Microsoft.CostManagement

Resource type	Complete mode deletion
Alerts	No
BenefitRecommendations	No
BenefitUtilizationSummaries	No
BillingAccounts	No
Budgets	No
CloudConnectors	No
Connectors	Yes
Departments	No
Dimensions	No

Resource type	Complete mode deletion
EnrollmentAccounts	No
Exports	No
ExternalBillingAccounts	No
ExternalBillingAccounts / Alerts	No
ExternalBillingAccounts / Dimensions	No
ExternalBillingAccounts / Forecast	No
ExternalBillingAccounts / Query	No
ExternalSubscriptions	No
ExternalSubscriptions / Alerts	No
ExternalSubscriptions / Dimensions	No
ExternalSubscriptions / Forecast	No
ExternalSubscriptions / Query	No
fetchMarketplacePrices	No
fetchPrices	No
Forecast	No
GenerateCostDetailsReport	No
GenerateDetailedCostReport	No
Insights	No
Pricesheets	No
Publish	No
Query	No
register	No
Reportconfigs	No
Reports	No
ScheduledActions	No
Settings	No

Resource type	Complete mode deletion
Views	No

Microsoft.CustomerLockbox

Resource type	Complete mode deletion
DisableLockbox	No
EnableLockbox	No
requests	No
TenantOptedIn	No

Microsoft.CustomProviders

Resource type	Complete mode deletion
associations	No
resourceProviders	Yes

Microsoft.D365CustomerInsights

Resource type	Complete mode deletion
instances	Yes

Microsoft.Dashboard

Resource type	Complete mode deletion
grafana	Yes
grafana / privateEndpointConnections	No
grafana / privateLinkResources	No

Microsoft.DataBox

Resource type	Complete mode deletion
jobs	Yes
jobs / eventGridFilters	No

Microsoft.DataBoxEdge

Resource type	Complete mode deletion
DataBoxEdgeDevices	Yes

Microsoft.Databricks

Resource type	Complete mode deletion
accessConnectors	Yes
workspaces	Yes
workspaces / dbWorkspaces	No
workspaces / virtualNetworkPeerings	No

Microsoft.DataCatalog

Resource type	Complete mode deletion
catalogs	Yes
datacatalogs	Yes

Microsoft.DataCollaboration

Resource type	Complete mode deletion
workspaces	Yes
workspaces / constrainedResources	No
workspaces / contracts	No
workspaces / contracts / entitlements	No

Resource type	Complete mode deletion
workspaces / dataAssets	No
workspaces / dataAssets / dataSets	No
workspaces / pipelineRuns	No
workspaces / pipelineRuns / pipelineStepRuns	No
workspaces / pipelines	No
workspaces / pipelines / pipelineSteps	No
workspaces / pipelines / runs	No
workspaces / proposals	No
workspaces / proposals / dataAssetReferences	No
workspaces / proposals / entitlements	No
workspaces / proposals / entitlements / constraints	No
workspaces / proposals / entitlements / policies	No
workspaces / proposals / invitations	No
workspaces / proposals / scriptReferences	No
workspaces / proposals / virtualOutputReferences	No
workspaces / resourceReferences	No
workspaces / scripts	No
workspaces / scripts / scriptrevisions	No

Microsoft.Datadog

Resource type	Complete mode deletion
agreements	No
monitors	Yes
monitors / getDefaultKey	No
monitors / refreshSetPasswordLink	No

Resource type	Complete mode deletion
monitors / setDefaultKey	No
monitors / singleSignOnConfigurations	No
monitors / tagRules	No
registeredSubscriptions	No

Microsoft.DataFactory

Resource type	Complete mode deletion
dataFactories	Yes
dataFactories / diagnosticSettings	No
dataFactories / metricDefinitions	No
dataFactorySchema	No
factories	Yes
factories / integrationRuntimes	No
factories / pipelines	No

Microsoft.DataLakeAnalytics

Resource type	Complete mode deletion
accounts	Yes
accounts / dataLakeStoreAccounts	No
accounts / storageAccounts	No
accounts / storageAccounts / containers	No
accounts / transferAnalyticsUnits	No

Microsoft.DataLakeStore

Resource type	Complete mode deletion

Resource type	Complete mode deletion
accounts	Yes
accounts / eventGridFilters	No
accounts / firewallRules	No

Microsoft.DataMigration

Resource type	Complete mode deletion
DatabaseMigrations	No
services	Yes
services / projects	Yes
slots	Yes
SqlMigrationServices	Yes

Microsoft.DataProtection

Resource type	Complete mode deletion
backupInstances	No
BackupVaults	Yes
ResourceGuards	Yes

Microsoft.DataReplication

Resource type	Complete mode deletion
replicationFabrics	Yes
replicationVaults	Yes

Microsoft.DataShare

Resource type	Complete mode deletion

Resource type	Complete mode deletion
accounts	Yes
accounts / shares	No
accounts / shares / datasets	No
accounts / shares / invitations	No
accounts / shares / providersharesubscriptions	No
accounts / shares / synchronizationSettings	No
accounts / sharesubscriptions	No
accounts / sharesubscriptions / consumerSourceDataSets	No
accounts / sharesubscriptions / datasetmappings	No
accounts / sharesubscriptions / triggers	No

Microsoft.DBforMariaDB

Resource type	Complete mode deletion
servers	Yes
servers / advisors	No
servers / keys	No
servers / privateEndpointConnectionProxies	No
servers / privateEndpointConnections	No
servers / privateLinkResources	No
servers / queryTexts	No
servers / recoverableServers	No
servers / resetQueryPerformanceInsightData	No
servers / topQueryStatistics	No
servers / virtualNetworkRules	No
servers / waitStatistics	No

Microsoft.DBforMySQL

Resource type	Complete mode deletion
assessForMigration	No
flexibleServers	Yes
getPrivateDnsZoneSuffix	No
servers	Yes
servers / advisors	No
servers / keys	No
servers / privateEndpointConnectionProxies	No
servers / privateEndpointConnections	No
servers / privateLinkResources	No
servers / queryTexts	No
servers / recoverableServers	No
servers / resetQueryPerformanceInsightData	No
servers / topQueryStatistics	No
servers / virtualNetworkRules	No
servers / waitStatistics	No

Microsoft.DBforPostgreSQL

Resource type	Complete mode deletion
flexibleServers	Yes
getPrivateDnsZoneSuffix	No
serverGroups	Yes
serverGroupsv2	Yes
servers	Yes
servers / advisors	No

Resource type	Complete mode deletion
servers / keys	No
servers / privateEndpointConnectionProxies	No
servers / privateEndpointConnections	No
servers / privateLinkResources	No
servers / queryTexts	No
servers / recoverableServers	No
servers / resetQueryPerformanceInsightData	No
servers / topQueryStatistics	No
servers / virtualNetworkRules	No
servers / waitStatistics	No
serversv2	Yes

Microsoft.DelegatedNetwork

Resource type	Complete mode deletion
controller	Yes
delegatedSubnets	Yes
orchestrators	Yes

Microsoft.DeploymentManager

Resource type	Complete mode deletion
artifactSources	Yes
rollouts	Yes
serviceTopologies	Yes
serviceTopologies / services	Yes
serviceTopologies / services / serviceUnits	Yes
steps	Yes

Microsoft.DesktopVirtualization

Resource type	Complete mode deletion
applicationgroups	Yes
applicationgroups / applications	No
applicationgroups / desktops	No
applicationgroups / startmenuitems	No
hostpools	Yes
hostpools / msixpackages	No
hostpools / sessionhosts	No
hostpools / sessionhosts / usersessions	No
hostpools / usersessions	No
scalingplans	Yes
workspaces	Yes

Microsoft.DevAI

Resource type	Complete mode deletion
instances	Yes
instances / experiments	Yes
instances / sandboxes	Yes
instances / sandboxes / experiments	Yes

Microsoft.DevCenter

Resource type	Complete mode deletion
devcenters	Yes
devcenters / attachednetworks	No
devcenters / catalogs	No

Resource type	Complete mode deletion
devcenters / devboxdefinitions	Yes
devcenters / environmentTypes	No
devcenters / galleries	No
devcenters / galleries / images	No
devcenters / galleries / images / versions	No
devcenters / images	No
networkconnections	Yes
projects	Yes
projects / allowedEnvironmentTypes	No
projects / attachednetworks	No
projects / devboxdefinitions	No
projects / environmentTypes	No
projects / pools	Yes
projects / pools / schedules	No
registeredSubscriptions	No

Microsoft.DevHub

Resource type	Complete mode deletion
workflows	Yes

Microsoft.Devices

Resource type	Complete mode deletion
IotHubs	Yes
IotHubs / eventGridFilters	No
IotHubs / failover	No
IotHubs / securitySettings	No

Resource type	Complete mode deletion
ProvisioningServices	Yes
usages	No

Microsoft.DeviceUpdate

Resource type	Complete mode deletion
accounts	Yes
accounts / instances	Yes
accounts / privateEndpointConnectionProxies	No
accounts / privateEndpointConnections	No
accounts / privateLinkResources	No
registeredSubscriptions	No

Microsoft.DevOps

Resource type	Complete mode deletion
pipelines	Yes

Microsoft.DevTestLab

Resource type	Complete mode deletion
labcenters	Yes
labs	Yes
labs / environments	Yes
labs / serviceRunners	Yes
labs / virtualMachines	Yes
schedules	Yes

Microsoft.DigitalTwins

Resource type	Complete mode deletion
digitalTwinsInstances	Yes
digitalTwinsInstances / endpoints	No
digitalTwinsInstances / timeSeriesDatabaseConnections	No

Microsoft.DocumentDB

Resource type	Complete mode deletion
cassandraClusters	Yes
databaseAccountNames	No
databaseAccounts	Yes
databaseAccounts / encryptionScopes	No
restorableDatabaseAccounts	No

Microsoft.DomainRegistration

Resource type	Complete mode deletion
domains	Yes
domains / domainOwnershipIdentifiers	No
generateSsoRequest	No
topLevelDomains	No
validateDomainRegistrationInformation	No

Microsoft.EdgeOrder

Resource type	Complete mode deletion
addresses	Yes

Resource type	Complete mode deletion
orderItems	Yes
orders	No
productFamiliesMetadata	No

Microsoft.Elastic

Resource type	Complete mode deletion
monitors	Yes
monitors / tagRules	No

Microsoft.EventGrid

Resource type	Complete mode deletion
domains	Yes
domains / topics	No
eventSubscriptions	No
extensionTopics	No
namespaces	Yes
partnerConfigurations	Yes
partnerDestinations	Yes
partnerNamespaces	Yes
partnerNamespaces / channels	No
partnerNamespaces / eventChannels	No
partnerRegistrations	Yes
partnerTopics	Yes
partnerTopics / eventSubscriptions	No
systemTopics	Yes
systemTopics / eventSubscriptions	No

Resource type	Complete mode deletion
topics	Yes
topicTypes	No
verifiedPartners	No

Microsoft.EventHub

Resource type	Complete mode deletion
clusters	Yes
namespaces	Yes
namespaces / applicationGroups	No
namespaces / authorizationrules	No
namespaces / disasterrecoveryconfigs	No
namespaces / eventhubs	No
namespaces / eventhubs / authorizationrules	No
namespaces / eventhubs / consumergroups	No
namespaces / networkrulesets	No
namespaces / privateEndpointConnections	No

Microsoft.Falcon

Resource type	Complete mode deletion
namespaces	Yes

Microsoft.Features

Resource type	Complete mode deletion
featureConfigurations	No
featureProviderNamespaces	No

Resource type	Complete mode deletion
featureProviders	No
features	No
providers	No
subscriptionFeatureRegistrations	No

Microsoft.Fidalgo

Resource type	Complete mode deletion
devcenters	Yes
devcenters / attachednetworks	No
devcenters / catalogs	No
devcenters / catalogs / items	No
devcenters / devboxdefinitions	Yes
devcenters / environmentTypes	No
devcenters / galleries	No
devcenters / galleries / images	No
devcenters / galleries / images / versions	No
devcenters / images	No
devcenters / mappings	No
machinedefinitions	Yes
networksettings	Yes
projects	Yes
projects / attachednetworks	No
projects / catalogItems	No
projects / devboxdefinitions	No
projects / environments	Yes
projects / environments / deployments	No

Resource type	Complete mode deletion
projects / environmentTypes	No
projects / pools	Yes
registeredSubscriptions	No

Microsoft.FluidRelay

Resource type	Complete mode deletion
fluidRelayServers	Yes
fluidRelayServers / fluidRelayContainers	No

Microsoft.Graph

Resource type	Complete mode deletion
AzureADApplication	Yes
AzureADApplicationPrototype	Yes
registeredSubscriptions	No

Microsoft.GuestConfiguration

Resource type	Complete mode deletion
guestConfigurationAssignments	No

Microsoft.HanaOnAzure

Resource type	Complete mode deletion
hanaInstances	Yes
sapMonitors	Yes

Microsoft.HardwareSecurityModules

Resource type	Complete mode deletion
dedicatedHSMs	Yes

Microsoft.HDInsight

Resource type	Complete mode deletion
clusterPools	Yes
clusterPools / clusters	Yes
clusterPools / clusters / instanceViews	No
clusterPools / clusters / serviceConfigs	No
clusterPools / clusters / sessionClusters	Yes
clusterPools / clusters / sessionClusters / instanceViews	No
clusterPools / clusters / sessionClusters / serviceConfigs	No
clusters	Yes
clusters / applications	No

Microsoft.HealthBot

Resource type	Complete mode deletion
healthBots	Yes

Microsoft.HealthcareApis

Resource type	Complete mode deletion
services	Yes
services / iomtconnectors	No
services / iomtconnectors / connections	No
services / iomtconnectors / mappings	No

Resource type	Complete mode deletion
services / privateEndpointConnectionProxies	No
services / privateEndpointConnections	No
services / privateLinkResources	No
validateMedtechMappings	No
workspaces	Yes
workspaces / analyticsconnectors	Yes
workspaces / dicomservices	Yes
workspaces / eventGridFilters	No
workspaces / fhirservices	Yes
workspaces / iotconnectors	Yes
workspaces / iotconnectors / destinations	No
workspaces / iotconnectors / fhirdestinations	No
workspaces / privateEndpointConnectionProxies	No
workspaces / privateEndpointConnections	No
workspaces / privateLinkResources	No

Microsoft.HpcWorkbench

Resource type	Complete mode deletion
instances	Yes
instances / chambers	Yes
instances / chambers / accessProfiles	Yes
instances / chambers / fileRequests	No
instances / chambers / files	No
instances / chambers / workloads	Yes
instances / consortiums	Yes

Microsoft.HybridCompute

Resource type	Complete mode deletion
machines	Yes
machines / assessPatches	No
machines / extensions	Yes
machines / installPatches	No
machines / privateLinkScopes	No
privateLinkScopes	Yes
privateLinkScopes / privateEndpointConnectionProxies	No
privateLinkScopes / privateEndpointConnections	No

Microsoft.HybridConnectivity

Resource type	Complete mode deletion
endpoints	No

Microsoft.HybridContainerService

Resource type	Complete mode deletion
provisionedClusters	Yes
provisionedClusters / agentPools	Yes
provisionedClusters / hybridIdentityMetadata	No
provisionedClusters / upgradeProfiles	No
storageSpaces	Yes
virtualNetworks	Yes

Microsoft.HybridData

Resource type	Complete mode deletion
dataManagers	Yes

Microsoft.HybridNetwork

Resource type	Complete mode deletion
configurationGroupValues	Yes
devices	Yes
networkFunctionPublishers	No
networkFunctionPublishers / networkFunctionDefinitionGroups	No
networkFunctionPublishers / networkFunctionDefinitionGroups / publisherNetworkFunctionDefinitionVersions	No
networkfunctions	Yes
networkFunctions / components	No
networkFunctionVendors	No
publishers	Yes
publishers / artifactStores	Yes
publishers / artifactStores / artifactManifests	Yes
publishers / configurationGroupSchemas	Yes
publishers / networkFunctionDefinitionGroups	Yes
publishers / networkFunctionDefinitionGroups / networkFunctionDefinitionVersions	Yes
publishers / networkFunctionDefinitionGroups / previewSubscriptions	Yes
publishers / networkServiceDesignGroups	Yes
publishers / networkServiceDesignGroups / networkServiceDesignVersions	Yes
registeredSubscriptions	No

Resource type	Complete mode deletion
siteNetworkServices	Yes
sites	Yes
vendors	No

Microsoft.ImportExport

Resource type	Complete mode deletion
jobs	Yes

microsoft.insights

Resource type	Complete mode deletion
actiongroups	Yes
actiongroups / networkSecurityPerimeterAssociationProxies	No
actiongroups / networkSecurityPerimeterConfigurations	No
activityLogAlerts	Yes
alertrules	Yes
autoscalesettings	Yes
components	Yes
components / aggregate	No
components / analyticsItems	No
components / annotations	No
components / api	No
components / apiKeys	No
components / currentBillingFeatures	No
components / defaultWorkItemConfig	No

Resource type	Complete mode deletion
components / events	No
components / exportConfiguration	No
components / extendQueries	No
components / favorites	No
components / featureCapabilities	No
components / generateDiagnosticServiceReadOnlyToken	No
components / generateDiagnosticServiceReadWriteToken	No
components / linkedstorageaccounts	No
components / metadata	No
components / metricDefinitions	No
components / metrics	No
components / move	No
components / myAnalyticsItems	No
components / myFavorites	No
components / pricingPlans	No
components / proactiveDetectionConfigs	No
components / purge	No
components / query	No
components / quotaStatus	No
components / webtests	No
components / workItemConfigs	No
createnotifications	No
dataCollectionEndpoints	Yes
dataCollectionEndpoints / networkSecurityPerimeterAssociationProxies	No

Resource type	Complete mode deletion
dataCollectionEndpoints / networkSecurityPerimeterConfigurations	No
dataCollectionEndpoints / scopedPrivateLinkProxies	No
dataCollectionRuleAssociations	No
dataCollectionRules	Yes
diagnosticSettings	No
diagnosticSettingsCategories	No
eventCategories	No
eventtypes	No
extendedDiagnosticSettings	No
generateDiagnosticServiceReadOnlyToken	No
generateDiagnosticServiceReadWriteToken	No
guestDiagnosticSettings	Yes
guestDiagnosticSettingsAssociation	No
logDefinitions	No
logprofiles	No
logs	No
metricalerts	Yes
metricbaselines	No
metricbatch	No
metricDefinitions	No
metricNamespaces	No
metrics	No
migratealertrules	No
migrateToNewPricingModel	No
monitoredObjects	No

Resource type	Complete mode deletion
myWorkbooks	No
notificationgroups	Yes
notificationstatus	No
privateLinkScopes	Yes
privateLinkScopes / privateEndpointConnectionProxies	No
privateLinkScopes / privateEndpointConnections	No
privateLinkScopes / scopedResources	No
rollbackToLegacyPricingModel	No
scheduledqueryrules	Yes
scheduledqueryrules / networkSecurityPerimeterAssociationProxies	No
scheduledqueryrules / networkSecurityPerimeterConfigurations	No
topology	No
transactions	No
webtests	Yes
webtests / getTestResultFile	No
workbooks	Yes
workbooktemplates	Yes

Microsoft.IntelligentITDigitalTwin

Resource type	Complete mode deletion
digitalTwins	Yes
digitalTwins / assets	Yes
digitalTwins / executionPlans	Yes
digitalTwins / testPlans	Yes

Resource type	Complete mode deletion
digitalTwins / tests	Yes

Microsoft.IoTCentral

Resource type	Complete mode deletion
appTemplates	No
IoTApps	Yes

Microsoft.IoTFirmwareDefense

Resource type	Complete mode deletion
firmwareGroups	No
firmwareGroups / firmwares	No

Microsoft.IoTSecurity

Resource type	Complete mode deletion
alertTypes	No
defenderSettings	No
onPremiseSensors	No
recommendationTypes	No
sensors	No
sites	No

Microsoft.KeyVault

Resource type	Complete mode deletion
deletedManagedHSMs	No
deletedVaults	No

Resource type	Complete mode deletion
hsmPools	Yes
managedHSMs	Yes
vaults	Yes
vaults / accessPolicies	No
vaults / eventGridFilters	No
vaults / keys	No
vaults / keys / versions	No
vaults / secrets	No

Microsoft.Kubernetes

Resource type	Complete mode deletion
connectedClusters	Yes
registeredSubscriptions	No

Microsoft.KubernetesConfiguration

Resource type	Complete mode deletion
extensions	No
extensionTypes	No
fluxConfigurations	No
namespaces	No
privateLinkScopes	Yes
privateLinkScopes / privateEndpointConnectionProxies	No
privateLinkScopes / privateEndpointConnections	No
sourceControlConfigurations	No

Microsoft.Kusto

Resource type	Complete mode deletion
clusters	Yes
clusters / attacheddatabaseconfigurations	No
clusters / databases	No
clusters / databases / dataconnections	No
clusters / databases / eventhubconnections	No
clusters / databases / principalassignments	No
clusters / databases / scripts	No
clusters / dataconnections	No
clusters / principalassignments	No
clusters / sharedidentities	No

Microsoft.LabServices

Resource type	Complete mode deletion
labaccounts	Yes
labplans	Yes
labs	Yes
users	No

Microsoft.LoadTestService

Resource type	Complete mode deletion
loadtests	Yes
loadtests / outboundNetworkDependenciesEndpoints	No
registeredSubscriptions	No

Microsoft.Logic

Resource type	Complete mode deletion
hostingEnvironments	Yes
integrationAccounts	Yes
integrationServiceEnvironments	Yes
integrationServiceEnvironments / managedApis	Yes
isolatedEnvironments	Yes
workflows	Yes

Microsoft.Logz

Resource type	Complete mode deletion
monitors	Yes
monitors / accounts	Yes
monitors / accounts / tagRules	No
monitors / metricsSource	Yes
monitors / metricsSource / tagRules	No
monitors / singleSignOnConfigurations	No
monitors / tagRules	No
registeredSubscriptions	No

Microsoft.MachineLearning

Resource type	Complete mode deletion
commitmentPlans	Yes
webServices	Yes
Workspaces	Yes

Microsoft.MachineLearningServices

Resource type	Complete mode deletion
aisysteminventories	Yes
registries	Yes
registries / codes	No
registries / codes / versions	No
registries / components	No
registries / components / versions	No
registries / environments	No
registries / environments / versions	No
registries / models	No
registries / models / versions	No
virtualclusters	Yes
workspaces	Yes
workspaces / batchEndpoints	Yes
workspaces / batchEndpoints / deployments	Yes
workspaces / batchEndpoints / deployments / jobs	No
workspaces / batchEndpoints / jobs	No
workspaces / codes	No
workspaces / codes / versions	No
workspaces / components	No
workspaces / components / versions	No
workspaces / computes	No
workspaces / data	No
workspaces / data / versions	No
workspaces / datasets	No

Resource type	Complete mode deletion
workspaces / datastores	No
workspaces / environments	No
workspaces / environments / versions	No
workspaces / eventGridFilters	No
workspaces / jobs	No
workspaces / labelingJobs	No
workspaces / linkedServices	No
workspaces / models	No
workspaces / models / versions	No
workspaces / onlineEndpoints	Yes
workspaces / onlineEndpoints / deployments	Yes
workspaces / schedules	No
workspaces / services	No

Microsoft.Maintenance

Resource type	Complete mode deletion
applyUpdates	No
configurationAssignments	No
maintenanceConfigurations	Yes
publicMaintenanceConfigurations	No
updates	No

Microsoft.ManagedIdentity

Resource type	Complete mode deletion
Identities	No
userAssignedIdentities	Yes

Resource type	Complete mode deletion
userAssignedIdentities / federatedIdentityCredentials	No

Microsoft.ManagedServices

Resource type	Complete mode deletion
marketplaceRegistrationDefinitions	No
registrationAssignments	No
registrationDefinitions	No

Microsoft.Management

Resource type	Complete mode deletion
getEntities	No
managementGroups	No
managementGroups / settings	No
resources	No
startTenantBackfill	No
tenantBackfillStatus	No

Microsoft.Maps

Resource type	Complete mode deletion
accounts	Yes
accounts / creators	Yes
accounts / eventGridFilters	No

Microsoft.Marketplace

Resource type	Complete mode deletion
macc	No
offers	No
offerTypes	No
offerTypes / publishers	No
offerTypes / publishers / offers	No
offerTypes / publishers / offers / plans	No
offerTypes / publishers / offers / plans / agreements	No
offerTypes / publishers / offers / plans / configs	No
offerTypes / publishers / offers / plans / configs / importImage	No
privategalleryitems	No
privateStoreClient	No
privateStores	No
privateStores / AdminRequestApprovals	No
privateStores / anyExistingOffersInTheCollections	No
privateStores / billingAccounts	No
privateStores / bulkCollectionsAction	No
privateStores / collections	No
privateStores / collections / approveAllItems	No
privateStores / collections / disableApproveAllItems	No
privateStores / collections / offers	No
privateStores / collections / offers / upsertOfferWithMultiContext	No
privateStores / collections / transferOffers	No
privateStores / collectionsToSubscriptionsMapping	No

Resource type	Complete mode deletion
privateStores / fetchAllSubscriptionsInTenant	No
privateStores / offers	No
privateStores / offers / acknowledgeNotification	No
privateStores / queryApprovedPlans	No
privateStores / queryNotificationsState	No
privateStores / queryOffers	No
privateStores / queryUserOffers	No
privateStores / RequestApprovals	No
privateStores / requestApprovals / query	No
privateStores / requestApprovals / withdrawPlan	No
products	No
publishers	No
publishers / offers	No
publishers / offers / amendments	No
register	No
search	No

Microsoft.MarketplaceNotifications

Resource type	Complete mode deletion
reviewsnotifications	No

Microsoft.MarketplaceOrdering

Resource type	Complete mode deletion
agreements	No

Resource type	Complete mode deletion
offertypes	No

Microsoft.Media

Resource type	Complete mode deletion
mediaservices	Yes
mediaservices / accountFilters	No
mediaservices / assets	No
mediaservices / assets / assetFilters	No
mediaservices / assets / tracks	No
mediaservices / contentKeyPolicies	No
mediaservices / eventGridFilters	No
mediaservices / graphInstances	No
mediaservices / graphTopologies	No
mediaservices / liveEvents	Yes
mediaservices / liveEvents / liveOutputs	No
mediaservices / mediaGraphs	No
mediaservices / privateEndpointConnectionProxies	No
mediaservices / privateEndpointConnections	No
mediaservices / streamingEndpoints	Yes
mediaservices / streamingLocators	No
mediaservices / streamingPolicies	No
mediaservices / transforms	No
mediaservices / transforms / jobs	No
videoAnalyzers	Yes
videoAnalyzers / accessPolicies	No

Resource type	Complete mode deletion
videoAnalyzers / edgeModules	No
videoAnalyzers / livePipelines	No
videoAnalyzers / pipelineJobs	No
videoAnalyzers / pipelineTopologies	No
videoAnalyzers / videos	No

Microsoft.Migrate

Resource type	Complete mode deletion
assessmentProjects	Yes
migrateprojects	Yes
modernizeProjects	Yes
moveCollections	Yes
projects	Yes

Microsoft.MixedReality

Resource type	Complete mode deletion
objectAnchorsAccounts	Yes
objectUnderstandingAccounts	Yes
remoteRenderingAccounts	Yes
spatialAnchorsAccounts	Yes

Microsoft.MobileNetwork

Resource type	Complete mode deletion
mobileNetworks	Yes
mobileNetworks / dataNetworks	Yes

Resource type	Complete mode deletion
mobileNetworks / services	Yes
mobileNetworks / simPolicies	Yes
mobileNetworks / sites	Yes
mobileNetworks / slices	Yes
packetCoreControlPlanes	Yes
packetCoreControlPlanes / packetCoreDataPlanes	Yes
packetCoreControlPlanes / packetCoreDataPlanes / attachedDataNetworks	Yes
packetCoreControlPlaneVersions	No
simGroups	Yes
simGroups / sims	No
sims	Yes

Microsoft.Monitor

Resource type	Complete mode deletion
accounts	Yes

Microsoft.NetApp

Resource type	Complete mode deletion
netAppAccounts	Yes
netAppAccounts / accountBackups	No
netAppAccounts / backupPolicies	Yes
netAppAccounts / capacityPools	Yes
netAppAccounts / capacityPools / volumes	Yes
netAppAccounts / capacityPools / volumes / backups	No

Resource type	Complete mode deletion
netAppAccounts / capacityPools / volumes / mountTargets	No
netAppAccounts / capacityPools / volumes / snapshots	No
netAppAccounts / capacityPools / volumes / subvolumes	No
netAppAccounts / capacityPools / volumes / volumeQuotaRules	No
netAppAccounts / snapshotPolicies	Yes
netAppAccounts / vaults	No
netAppAccounts / volumeGroups	No

Microsoft.Network

Resource type	Complete mode deletion
applicationGateways	Yes
applicationGatewayWebApplicationFirewallPolicies	Yes
applicationSecurityGroups	Yes
azureFirewallFqdnTags	No
azureFirewalls	Yes
azureWebCategories	No
bastionHosts	Yes
bgpServiceCommunities	No
cloudServiceSlots	No
connections	Yes
customIpPrefixes	Yes
ddosCustomPolicies	Yes
ddosProtectionPlans	Yes

Resource type	Complete mode deletion
dnsForwardingRulesets	Yes
dnsForwardingRulesets / forwardingRules	No
dnsForwardingRulesets / virtualNetworkLinks	No
dnsResolvers	Yes
dnsResolvers / inboundEndpoints	Yes
dnsResolvers / outboundEndpoints	Yes
dnszones	Yes
dnszones / A	No
dnszones / AAAA	No
dnszones / all	No
dnszones / CAA	No
dnszones / CNAME	No
dnszones / MX	No
dnszones / NS	No
dnszones / PTR	No
dnszones / recordsets	No
dnszones / SOA	No
dnszones / SRV	No
dnszones / TXT	No
dscpConfigurations	Yes
expressRouteCircuits	Yes
expressRouteCrossConnections	Yes
expressRouteGateways	Yes
expressRoutePorts	Yes
expressRouteProviderPorts	No
expressRouteServiceProviders	No

Resource type	Complete mode deletion
firewallPolicies	Yes
frontdoors	Yes
frontdoors / frontendEndpoints	No
frontdoors / frontendEndpoints / customHttpsConfiguration	No
frontdoorWebApplicationFirewallManagedRuleSets	No
frontdoorWebApplicationFirewallPolicies	Yes
getDnsResourceReference	No
internalNotify	No
internalPublicIpAddresses	No
ipGroups	Yes
loadBalancers	Yes
localNetworkGateways	Yes
natGateways	Yes
networkExperimentProfiles	Yes
networkIntentPolicies	Yes
networkInterfaces	Yes
networkManagerConnections	No
networkManagers	Yes
networkProfiles	Yes
networkSecurityGroups	Yes
networkSecurityPerimeters	Yes
networkVirtualAppliances	Yes
networkWatchers	Yes
networkWatchers / connectionMonitors	Yes
networkWatchers / flowLogs	Yes

Resource type	Complete mode deletion
networkWatchers / lenses	Yes
networkWatchers / pingMeshes	Yes
p2sVpnGateways	Yes
privateDnsZones	Yes
privateDnsZones / A	No
privateDnsZones / AAAA	No
privateDnsZones / all	No
privateDnsZones / CNAME	No
privateDnsZones / MX	No
privateDnsZones / PTR	No
privateDnsZones / SOA	No
privateDnsZones / SRV	No
privateDnsZones / TXT	No
privateDnsZones / virtualNetworkLinks	Yes
privateDnsZonesInternal	No
privateEndpointRedirectMaps	Yes
privateEndpoints	Yes
privateEndpoints / privateLinkServiceProxies	No
privateLinkServices	Yes
publicIPAddresses	Yes
publicIPPrefixes	Yes
routeFilters	Yes
routeTables	Yes
securityPartnerProviders	Yes
serviceEndpointPolicies	Yes
trafficManagerGeographicHierarchies	No

Resource type	Complete mode deletion
trafficmanagerprofiles	Yes
trafficmanagerprofiles / azureendpoints	No
trafficmanagerprofiles / externalendpoints	No
trafficmanagerprofiles / heatMaps	No
trafficmanagerprofiles / nestedendpoints	No
trafficManagerUserMetricsKeys	No
virtualHubs	Yes
virtualNetworkGateways	Yes
virtualNetworks	Yes
virtualNetworks / privateDnsZoneLinks	No
virtualNetworks / taggedTrafficConsumers	No
virtualNetworkTaps	Yes
virtualRouters	Yes
virtualWans	Yes
vpnGateways	Yes
vpnServerConfigurations	Yes
vpnSites	Yes

Microsoft.NetworkCloud

Resource type	Complete mode deletion
bareMetalMachines	Yes
cloudServicesNetworks	Yes
clusterManagers	Yes
clusters	Yes
clusters / admissions	No
defaultCniNetworks	Yes

Resource type	Complete mode deletion
disks	Yes
hybridAksClusters	Yes
hybridAksManagementDomains	Yes
hybridAksVirtualMachines	Yes
l2Networks	Yes
l3Networks	Yes
rackManifests	Yes
racks	Yes
storageAppliances	Yes
trunkedNetworks	Yes
virtualMachines	Yes
workloadNetworks	Yes

Microsoft.NetworkFunction

Resource type	Complete mode deletion
azureTrafficCollectors	Yes
azureTrafficCollectors / collectorPolicies	Yes
meshVpns	Yes
meshVpns / connectionPolicies	Yes
meshVpns / privateEndpointConnectionProxies	No
meshVpns / privateEndpointConnections	No

Microsoft.NotificationHubs

Resource type	Complete mode deletion
namespaces	Yes
namespaces / notificationHubs	Yes

Microsoft.ObjectStore

Resource type	Complete mode deletion
osNamespaces	Yes

Microsoft.OffAzure

Resource type	Complete mode deletion
HyperVSites	Yes
ImportSites	Yes
MasterSites	Yes
ServerSites	Yes
VMwareSites	Yes

Microsoft.OpenEnergyPlatform

Resource type	Complete mode deletion
energyServices	Yes
energyServices / privateEndpointConnectionProxies	No
energyServices / privateEndpointConnections	No
energyServices / privateLinkResources	No

Microsoft.OpenLogisticsPlatform

Resource type	Complete mode deletion
applicationManagers	Yes
applicationManagers / applicationRegistrations	No
applicationManagers / eventGridFilters	No
applicationRegistrationInvites	No

Resource type	Complete mode deletion
applicationWorkspaces	Yes
applicationWorkspaces / applications	No
applicationWorkspaces / applications / applicationRegistrationInvites	No
shareInvites	No
workspaces	Yes
workspaces / applicationRegistrations	No
workspaces / applications	No
workspaces / eventGridFilters	No
workspaces / shares	No
workspaces / shareSubscriptions	No

Microsoft.OperationalInsights

Resource type	Complete mode deletion
clusters	Yes
deletedWorkspaces	No
linkTargets	No
querypacks	Yes
storageInsightConfigs	No
workspaces	Yes
workspaces / dataExports	No
workspaces / dataSources	No
workspaces / linkedServices	No
workspaces / linkedStorageAccounts	No
workspaces / metadata	No
workspaces / networkSecurityPerimeterAssociationProxies	No

Resource type	Complete mode deletion
workspaces / networkSecurityPerimeterConfigurations	No
workspaces / query	No
workspaces / scopedPrivateLinkProxies	No
workspaces / storageInsightConfigs	No
workspaces / tables	No

Microsoft.Orbital

Resource type	Complete mode deletion
contactProfiles	Yes
edgeSites	Yes
globalCommunicationsSites	No
groundStations	Yes
l2Connections	Yes
l3Connections	Yes
orbitalGateways	Yes
spacecrafts	Yes
spacecrafts / contacts	No

Microsoft.Peering

Resource type	Complete mode deletion
cdnPeeringPrefixes	No
legacyPeerings	No
lookingGlass	No
peerAsns	No
peerings	Yes

Resource type	Complete mode deletion
peeringServiceCountries	No
peeringServiceProviders	No
peeringServices	Yes

Microsoft.Pki

Resource type	Complete mode deletion
pkis	Yes
pkis / certificateAuthorities	No
pkis / enrollmentPolicies	Yes

Microsoft.PlayFab

Resource type	Complete mode deletion
playerAccountPools	Yes
titles	Yes
titles / automationRules	No
titles / segments	No
titles / titleDataSets	No
titles / titleInternalDataKeyValues	No
titles / titleInternalDataSets	No

Microsoft.PolicyInsights

Resource type	Complete mode deletion
attestations	No
componentPolicyStates	No
eventGridFilters	No

Resource type	Complete mode deletion
policyEvents	No
policyMetadata	No
policyStates	No
policyTrackedResources	No
remediations	No

Microsoft.Portal

Resource type	Complete mode deletion
consoles	No
dashboards	Yes
tenantconfigurations	No
userSettings	No

Microsoft.PowerBI

Resource type	Complete mode deletion
privateLinkServicesForPowerBI	Yes
tenants	Yes
tenants / workspaces	No
workspaceCollections	Yes

Microsoft.PowerBIDedicated

Resource type	Complete mode deletion
autoScaleVCores	Yes
capacities	Yes
servers	Yes

Microsoft.PowerPlatform

Resource type	Complete mode deletion
accounts	Yes
enterprisePolicies	Yes

Microsoft.ProviderHub

Resource type	Complete mode deletion
providerRegistrations	No
providerRegistrations / customRollouts	No
providerRegistrations / defaultRollouts	No
providerRegistrations / resourceActions	No
providerRegistrations / resourceTypeRegistrations	No

Microsoft.Purview

Resource type	Complete mode deletion
accounts	Yes
accounts / kafkaConfigurations	No
getDefaultAccount	No
removeDefaultAccount	No
setDefaultAccount	No

Microsoft.Quantum

Resource type	Complete mode deletion
Workspaces	Yes

Microsoft.Quota

Resource type	Complete mode deletion
quotaRequests	No
quotas	No
usages	No

Microsoft.RecommendationsService

Resource type	Complete mode deletion
accounts	Yes
accounts / modeling	Yes
accounts / serviceEndpoints	Yes

Microsoft.RecoveryServices

Resource type	Complete mode deletion
backupProtectedItems	No
vaults	Yes

Microsoft.RedHatOpenShift

Resource type	Complete mode deletion
OpenShiftClusters	Yes

Microsoft.Relay

Resource type	Complete mode deletion
namespaces	Yes
namespaces / authorizationrules	No
namespaces / hybridconnections	No

Resource type	Complete mode deletion
namespaces / hybridconnections / authorizationrules	No
namespaces / privateEndpointConnections	No
namespaces / wcfrelays	No
namespaces / wcfrelays / authorizationrules	No

Microsoft.ResourceConnector

Resource type	Complete mode deletion
appliances	Yes
telemetryconfig	No

Microsoft.ResourceGraph

Resource type	Complete mode deletion
queries	Yes
resourceChangeDetails	No
resourceChanges	No
resources	No
resourcesHistory	No
subscriptionsStatus	No

Microsoft.ResourceHealth

Resource type	Complete mode deletion
childResources	No
emergingissues	No
events	No
impactedResources	No

Resource type	Complete mode deletion
metadata	No

Microsoft.Resources

Resource type	Complete mode deletion
deployments	No
deploymentScripts	Yes
deploymentScripts / logs	No
deploymentStacks / snapshots	No
links	No
resourceGroups	No
snapshots	No
subscriptions	No
tags	No
templateSpecs	Yes
templateSpecs / versions	Yes
tenants	No
validateResources	No

Microsoft.SaaS

Resource type	Complete mode deletion
applications	Yes
resources	Yes
saasresources	No

Microsoft.Scom

Resource type	Complete mode deletion
managedInstances	Yes

Microsoft.ScVmm

Resource type	Complete mode deletion
AvailabilitySets	Yes
Clouds	Yes
VirtualMachines	Yes
VirtualMachines / HybridIdentityMetadata	No
VirtualMachineTemplates	Yes
VirtualNetworks	Yes
VMMServers	Yes
VMMServers / InventoryItems	No

Microsoft.Search

Resource type	Complete mode deletion
resourceHealthMetadata	No
searchServices	Yes

Microsoft.Security

Resource type	Complete mode deletion
adaptiveNetworkHardenings	No
advancedThreatProtectionSettings	No
alerts	No
alertsSuppressionRules	No
allowedConnections	No

Resource type	Complete mode deletion
antiMalwareSettings	No
applications	No
assessmentMetadata	No
assessments	No
assessments / governanceAssignments	No
assignments	Yes
attackPaths	No
autoDismissAlertsRules	No
automations	Yes
AutoProvisioningSettings	No
Compliances	No
connectedContainerRegistries	No
connectors	No
customAssessmentAutomations	Yes
customEntityStoreAssignments	Yes
dataCollectionAgents	No
dataScanners	Yes
dataSensitivitySettings	No
deviceSecurityGroups	No
discoveredSecuritySolutions	No
externalSecuritySolutions	No
governanceRules	No
InformationProtectionPolicies	No
ingestionSettings	No
insights	No
iotSecuritySolutions	Yes

Resource type	Complete mode deletion
iotSecuritySolutions / analyticsModels	No
iotSecuritySolutions / analyticsModels / aggregatedAlerts	No
iotSecuritySolutions / analyticsModels / aggregatedRecommendations	No
iotSecuritySolutions / iotAlerts	No
iotSecuritySolutions / iotAlertTypes	No
iotSecuritySolutions / iotRecommendations	No
iotSecuritySolutions / iotRecommendationTypes	No
jitNetworkAccessPolicies	No
jitPolicies	No
MdeOnboardings	No
policies	No
pricings	No
query	No
regulatoryComplianceStandards	No
regulatoryComplianceStandards / regulatoryComplianceControls	No
regulatoryComplianceStandards / regulatoryComplianceControls / regulatoryComplianceAssessments	No
secureScoreControlDefinitions	No
secureScoreControls	No
secureScores	No
secureScores / secureScoreControls	No
securityConnectors	Yes
securityContacts	No
securitySolutions	No

Resource type	Complete mode deletion
securitySolutionsReferenceData	No
securityStatuses	No
securityStatusesSummaries	No
serverVulnerabilityAssessments	No
serverVulnerabilityAssessmentsSettings	No
settings	No
sqlVulnerabilityAssessments	No
standards	Yes
subAssessments	No
tasks	No
topologies	No
vmScanners	No
workspaceSettings	No

Microsoft.SecurityDetonation

Resource type	Complete mode deletion
chambers	Yes

Microsoft.SecurityDevOps

Resource type	Complete mode deletion
azureDevOpsConnectors	Yes
azureDevOpsConnectors / orgs	No
azureDevOpsConnectors / orgs / projects	No
azureDevOpsConnectors / orgs / projects / repos	No
azureDevOpsConnectors / repos	No

Resource type	Complete mode deletion
azureDevOpsConnectors / stats	No
gitHubConnectors	Yes
gitHubConnectors / gitHubRepos	No
gitHubConnectors / owners	No
gitHubConnectors / owners / repos	No
gitHubConnectors / repos	No
gitHubConnectors / stats	No

Microsoft.SecurityInsights

Resource type	Complete mode deletion
aggregations	No
alertRules	No
alertRuleTemplates	No
automationRules	No
bookmarks	No
cases	No
contentPackages	No
contentTemplates	No
dataConnectorDefinitions	No
dataConnectors	No
enrichment	No
entities	No
entityQueryTemplates	No
fileImports	No
huntsessions	No
incidents	No

Resource type	Complete mode deletion
metadata	No
MitreCoverageRecords	No
onboardingStates	No
overview	No
recommendations	No
securityMLAnalyticsSettings	No
settings	No
sourceControls	No
threatIntelligence	No

Microsoft.SerialConsole

Resource type	Complete mode deletion
consoleServices	No
serialPorts	No

Microsoft.ServiceBus

Resource type	Complete mode deletion
namespaces	Yes
namespaces / authorizationrules	No
namespaces / disasterrecoveryconfigs	No
namespaces / eventgridfilters	No
namespaces / networkrulesets	No
namespaces / privateEndpointConnections	No
namespaces / queues	No
namespaces / queues / authorizationrules	No
namespaces / topics	No

Resource type	Complete mode deletion
namespaces / topics / authorizationrules	No
namespaces / topics / subscriptions	No
namespaces / topics / subscriptions / rules	No
premiumMessagingRegions	No

Microsoft.ServiceFabric

Resource type	Complete mode deletion
clusters	Yes
clusters / applications	No
clusters / applications / services	No
clusters / applicationTypes	No
clusters / applicationTypes / versions	No
edgeclusters	Yes
edgeclusters / applications	No
managedclusters	Yes
managedclusters / applications	No
managedclusters / applications / services	No
managedclusters / applicationTypes	No
managedclusters / applicationTypes / versions	No
managedclusters / nodetypes	No

Microsoft.ServiceLinker

Resource type	Complete mode deletion
dryruns	No
linkers	No

Microsoft.ServicesHub

Resource type	Complete mode deletion
connectors	Yes
supportOfferingEntitlement	No
workspaces	No

Microsoft.SignalRService

Resource type	Complete mode deletion
SignalR	Yes
SignalR / eventGridFilters	No
WebPubSub	Yes

Microsoft.Singularity

Resource type	Complete mode deletion
accounts	Yes
accounts / accountQuotaPolicies	No
accounts / groupPolicies	No
accounts / jobs	No
accounts / models	No
accounts / networks	No
accounts / storageContainers	No
images	No
quotas	No

Microsoft.SoftwarePlan

Resource type	Complete mode deletion

Resource type	Complete mode deletion
hybridUseBenefits	No

Microsoft.Solutions

Resource type	Complete mode deletion
applicationDefinitions	Yes
applications	Yes
jitRequests	Yes

Microsoft.Sql

Resource type	Complete mode deletion
instancePools	Yes
managedInstances	Yes
managedInstances / administrators	No
managedInstances / advancedThreatProtectionSettings	No
managedInstances / databases	Yes
managedInstances / databases / advancedThreatProtectionSettings	No
managedInstances / databases / backupLongTermRetentionPolicies	No
managedInstances / databases / vulnerabilityAssessments	No
managedInstances / dnsAliases	No
managedInstances / metricDefinitions	No
managedInstances / metrics	No
managedInstances / recoverableDatabases	No
managedInstances / sqlAgent	No

Resource type	Complete mode deletion
managedInstances / startStopSchedules	No
managedInstances / tdeCertificates	No
managedInstances / vulnerabilityAssessments	No
servers	Yes
servers / administrators	No
servers / advancedThreatProtectionSettings	No
servers / advisors	No
servers / aggregatedDatabaseMetrics	No
servers / auditingSettings	No
servers / automaticTuning	No
servers / communicationLinks	No
servers / connectionPolicies	No
servers / databases	Yes
servers / databases / activate	No
servers / databases / activatedatabase	No
servers / databases / advancedThreatProtectionSettings	No
servers / databases / advisors	No
servers / databases / auditingSettings	No
servers / databases / auditRecords	No
servers / databases / automaticTuning	No
servers / databases / backupLongTermRetentionPolicies	No
servers / databases / backupShortTermRetentionPolicies	No
servers / databases / databaseState	No
servers / databases / dataMaskingPolicies	No

Resource type	Complete mode deletion
servers / databases / dataMaskingPolicies / rules	No
servers / databases / deactivate	No
servers / databases / deactivatedatabase	No
servers / databases / extensions	No
servers / databases / geoBackupPolicies	No
servers / databases / ledgerDigestUploads	No
servers / databases / metricDefinitions	No
servers / databases / metrics	No
servers / databases / recommendedSensitivityLabels	No
servers / databases / securityAlertPolicies	No
servers / databases / sqlvulnerabilityassessments	No
servers / databases / syncGroups	No
servers / databases / syncGroups / syncMembers	No
servers / databases / topQueries	No
servers / databases / topQueries / queryText	No
servers / databases / transparentDataEncryption	No
servers / databases / VulnerabilityAssessment	No
servers / databases / vulnerabilityAssessments	No
servers / databases / VulnerabilityAssessmentScans	No
servers / databases / VulnerabilityAssessmentSettings	No
servers / databases / workloadGroups	No
servers / databaseSecurityPolicies	No

Resource type	Complete mode deletion
servers / devOpsAuditingSettings	No
servers / disasterRecoveryConfiguration	No
servers / dnsAliases	No
servers / elasticPoolEstimates	No
servers / elasticpools	Yes
servers / elasticPools / advisors	No
servers / elasticpools / metricdefinitions	No
servers / elasticpools / metrics	No
servers / encryptionProtector	No
servers / extendedAuditingSettings	No
servers / failoverGroups	No
servers / import	No
servers / jobAccounts	Yes
servers / jobAgents	Yes
servers / jobAgents / jobs	No
servers / jobAgents / jobs / executions	No
servers / jobAgents / jobs / steps	No
servers / keys	No
servers / recommendedElasticPools	No
servers / recoverableDatabases	No
servers / restorableDroppedDatabases	No
servers / securityAlertPolicies	No
servers / serviceObjectives	No
servers / sqlvulnerabilityassessments	No
servers / syncAgents	No
servers / tdeCertificates	No

Resource type	Complete mode deletion
servers / usages	No
servers / virtualNetworkRules	No
servers / vulnerabilityAssessments	No
virtualClusters	Yes

Microsoft.SqlVirtualMachine

Resource type	Complete mode deletion
SqlVirtualMachineGroups	Yes
SqlVirtualMachines	Yes

Microsoft.Storage

Resource type	Complete mode deletion
dataMovers	Yes
dataMovers / agents	No
dataMovers / endpoints	No
dataMovers / projects	No
dataMovers / projects / jobDefinitions	No
dataMovers / projects / jobDefinitions / jobRuns	No
deletedAccounts	No
storageAccounts	Yes
storageAccounts / blobServices	No
storageAccounts / encryptionScopes	No
storageAccounts / fileServices	No
storageAccounts / queueServices	No
storageAccounts / services	No

Resource type	Complete mode deletion
storageAccounts / services / metricDefinitions	No
storageAccounts / storageTaskAssignments	No
storageAccounts / tableServices	No
storageTasks	Yes
usages	No

Microsoft.StorageCache

Resource type	Complete mode deletion
amlFilesystems	Yes
caches	Yes
caches / storageTargets	No
usageModels	No

Microsoft.StorageMover

Resource type	Complete mode deletion
storageMovers	Yes
storageMovers / agents	No
storageMovers / endpoints	No
storageMovers / projects	No
storageMovers / projects / jobDefinitions	No
storageMovers / projects / jobDefinitions / jobRuns	No

Microsoft.StoragePool

Resource type	Complete mode deletion
diskPools	Yes

Resource type	Complete mode deletion
diskPools / iscsiTargets	No

Microsoft.StorageSync

Resource type	Complete mode deletion
storageSyncServices	Yes
storageSyncServices / registeredServers	No
storageSyncServices / syncGroups	No
storageSyncServices / syncGroups / cloudEndpoints	No
storageSyncServices / syncGroups / serverEndpoints	No
storageSyncServices / workflows	No

Microsoft.StorSimple

Resource type	Complete mode deletion
managers	Yes

Microsoft.StreamAnalytics

Resource type	Complete mode deletion
clusters	Yes
clusters / privateEndpoints	No
streamingjobs	Yes

Microsoft.Subscription

Resource type	Complete mode deletion
acceptChangeTenant	No

Resource type	Complete mode deletion
acceptOwnership	No
acceptOwnershipStatus	No
aliases	No
cancel	No
changeTenantRequest	No
changeTenantStatus	No
enable	No
policies	No
rename	No
SubscriptionDefinitions	No
subscriptions	No

microsoft.support

Resource type	Complete mode deletion
lookUpResourceId	No
services	No
services / problemclassifications	No
supporttickets	No

Microsoft.Synapse

Resource type	Complete mode deletion
privateLinkHubs	Yes
workspaces	Yes
workspaces / bigDataPools	Yes
workspaces / kustoPools	Yes

Resource type	Complete mode deletion
workspaces / kustoPools / attacheddatabaseconfigurations	No
workspaces / kustoPools / databases	No
workspaces / kustoPools / databases / dataconnections	No
workspaces / sqlDatabases	Yes
workspaces / sqlPools	Yes

Microsoft.TestBase

Resource type	Complete mode deletion
testBaseAccounts	Yes
testBaseAccounts / customerEvents	No
testBaseAccounts / emailEvents	No
testBaseAccounts / externalTestTools	No
testBaseAccounts / externalTestTools / testCases	No
testBaseAccounts / featureUpdateSupportedOses	No
testBaseAccounts / firstPartyApps	No
testBaseAccounts / flightingRings	No
testBaseAccounts / packages	Yes
testBaseAccounts / packages / favoriteProcesses	No
testBaseAccounts / packages / osUpdates	No
testBaseAccounts / testSummaries	No
testBaseAccounts / testTypes	No
testBaseAccounts / usages	No

Microsoft.TimeSeriesInsights

Resource type	Complete mode deletion
environments	Yes
environments / accessPolicies	No
environments / eventsources	Yes
environments / privateEndpointConnectionProxies	No
environments / privateEndpointConnections	No
environments / privateLinkResources	No
environments / referenceDataSets	Yes

Microsoft.UsageBilling

Resource type	Complete mode deletion
accounts	Yes

Microsoft.VideoIndexer

Resource type	Complete mode deletion
accounts	Yes

Microsoft.VirtualMachineImages

Resource type	Complete mode deletion
imageTemplates	Yes
imageTemplates / runOutputs	No
imageTemplates / triggers	No

microsoft.visualstudio

Resource type	Complete mode deletion
account	Yes
account / extension	Yes
account / project	Yes

Microsoft.VMware

Resource type	Complete mode deletion
arczones	Yes
resourcepools	Yes
vcenters	Yes
VCenters / InventoryItems	No
virtualmachines	Yes
virtualmachinetemplates	Yes
virtualnetworks	Yes

Microsoft.VMwareCloudSimple

Resource type	Complete mode deletion
dedicatedCloudNodes	Yes
dedicatedCloudServices	Yes
virtualMachines	Yes

Microsoft.VSOnline

Resource type	Complete mode deletion
accounts	Yes
plans	Yes
registeredSubscriptions	No

Microsoft.Web

Resource type	Complete mode deletion
apiManagementAccounts	No
apiManagementAccounts / apiAcls	No
apiManagementAccounts / apis	No
apiManagementAccounts / apis / apiAcls	No
apiManagementAccounts / apis / connectionAcls	No
apiManagementAccounts / apis / connections	No
apiManagementAccounts / apis / connections / connectionAcls	No
apiManagementAccounts / apis / localizedDefinitions	No
apiManagementAccounts / connectionAcls	No
apiManagementAccounts / connections	No
billingMeters	No
certificates	Yes
connectionGateways	Yes
connections	Yes
containerApps	Yes
customApis	Yes
customhostnameSites	No
deletedSites	No
functionAppStacks	No
generateGithubAccessTokenForAppserviceCLI	No
hostingEnvironments	Yes
hostingEnvironments / eventGridFilters	No
hostingEnvironments / multiRolePools	No

Resource type	Complete mode deletion
hostingEnvironments / workerPools	No
kubeEnvironments	Yes
publishingUsers	No
recommendations	No
resourceHealthMetadata	No
runtimes	No
serverFarms	Yes
serverFarms / eventGridFilters	No
serverFarms / firstPartyApps	No
serverFarms / firstPartyApps / keyVaultSettings	No
sites	Yes
sites / eventGridFilters	No
sites / hostNameBindings	No
sites / networkConfig	No
sites / premieraddons	Yes
sites / slots	Yes
sites / slots / eventGridFilters	No
sites / slots / hostNameBindings	No
sites / slots / networkConfig	No
sourceControls	No
staticSites	Yes
staticSites / builds	No
staticSites / builds / linkedBackends	No
staticSites / builds / userProvidedFunctionApps	No
staticSites / linkedBackends	No
staticSites / userProvidedFunctionApps	No

Resource type	Complete mode deletion
validate	No
verifyHostingEnvironmentVnet	No
webAppStacks	No
workerApps	Yes

Microsoft.WindowsESU

Resource type	Complete mode deletion
multipleActivationKeys	Yes

Microsoft.WindowsIoT

Resource type	Complete mode deletion
DeviceServices	Yes

Microsoft.WorkloadBuilder

Resource type	Complete mode deletion
migrationAgents	Yes
workloads	Yes
workloads / instances	No
workloads / versions	No
workloads / versions / artifacts	No

Microsoft.WorkloadMonitor

Resource type	Complete mode deletion
monitors	No

Microsoft.Workloads

Resource type	Complete mode deletion
monitors	Yes
monitors / providerInstances	No
phpWorkloads	Yes
phpWorkloads / wordpressInstances	No
sapVirtualInstances	Yes
sapVirtualInstances / applicationInstances	Yes
sapVirtualInstances / centralInstances	Yes
sapVirtualInstances / databaseInstances	Yes

Next steps

To get the same data as a file of comma-separated values, download complete-mode-deletion.csv .

Understand the structure and syntax of ARM templates

Article • 04/09/2023

This article describes the structure of an Azure Resource Manager template (ARM template). It presents the different sections of a template and the properties that are available in those sections.

This article is intended for users who have some familiarity with ARM templates. It provides detailed information about the structure of the template. For a step-by-step tutorial that guides you through the process of creating a template, see Tutorial: Create and deploy your first ARM template. To learn about ARM templates through a guided set of Learn modules, see Deploy and manage resources in Azure by using ARM templates.

> **Tip**
>
> Bicep is a new language that offers the same capabilities as ARM templates but with a syntax that's easier to use. If you're considering infrastructure as code options, we recommend looking at Bicep.
>
> To learn about the elements of a Bicep file, see **Understand the structure and syntax of Bicep files**.

Template format

In its simplest structure, a template has the following elements:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "",
  "apiProfile": "",
  "parameters": {  },
  "variables": {  },
  "functions": [  ],
  "resources": [  ],
  "outputs": {  }
}
```

Element name	Required	Description
$schema	Yes	Location of the JavaScript Object Notation (JSON) schema file that describes the version of the template language. The version number you use depends on the scope of the deployment and your JSON editor. If you're using Visual Studio Code with the Azure Resource Manager tools extension, use the latest version for resource group deployments: `https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#` Other editors (including Visual Studio) may not be able to process this schema. For those editors, use: `https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#` For subscription deployments, use: `https://schema.management.azure.com/schemas/2018-05-01/subscriptionDeploymentTemplate.json#` For management group deployments, use: `https://schema.management.azure.com/schemas/2019-08-01/managementGroupDeploymentTemplate.json#` For tenant deployments, use: `https://schema.management.azure.com/schemas/2019-08-01/tenantDeploymentTemplate.json#`
contentVersion	Yes	Version of the template (such as 1.0.0.0). You can provide any value for this element. Use this value to document significant changes in your template. When deploying resources using the template, this value can be used to make sure that the right template is being used.

Element name	Required	Description
apiProfile	No	An API version that serves as a collection of API versions for resource types. Use this value to avoid having to specify API versions for each resource in the template. When you specify an API profile version and don't specify an API version for the resource type, Resource Manager uses the API version for that resource type that is defined in the profile. The API profile property is especially helpful when deploying a template to different environments, such as Azure Stack and global Azure. Use the API profile version to make sure your template automatically uses versions that are supported in both environments. For a list of the current API profile versions and the resources API versions defined in the profile, see API Profile. For more information, see Track versions using API profiles.
parameters	No	Values that are provided when deployment is executed to customize resource deployment.
variables	No	Values that are used as JSON fragments in the template to simplify template language expressions.
functions	No	User-defined functions that are available within the template.
resources	Yes	Resource types that are deployed or updated in a resource group or subscription.
outputs	No	Values that are returned after deployment.

Each element has properties you can set. This article describes the sections of the template in greater detail.

Parameters

In the `parameters` section of the template, you specify which values you can input when deploying the resources. You're limited to 256 parameters in a template. You can reduce the number of parameters by using objects that contain multiple properties.

The available properties for a parameter are:

JSON

```
"parameters": {
  "<parameter-name>" : {
    "type" : "<type-of-parameter-value>",
```

```
      "defaultValue": "<default-value-of-parameter>",
      "allowedValues": [ "<array-of-allowed-values>" ],
      "minValue": <minimum-value-for-int>,
      "maxValue": <maximum-value-for-int>,
      "minLength": <minimum-length-for-string-or-array>,
      "maxLength": <maximum-length-for-string-or-array-parameters>,
      "metadata": {
        "description": "<description-of-the parameter>"
      }
    }
  }
```

Element name	Required	Description
parameter-name	Yes	Name of the parameter. Must be a valid JavaScript identifier.
type	Yes	Type of the parameter value. The allowed types and values are **string**, **securestring**, **int**, **bool**, **object**, **secureObject**, and **array**. See Data types in ARM templates.
defaultValue	No	Default value for the parameter, if no value is provided for the parameter.
allowedValues	No	Array of allowed values for the parameter to make sure that the right value is provided.
minValue	No	The minimum value for int type parameters, this value is inclusive.
maxValue	No	The maximum value for int type parameters, this value is inclusive.
minLength	No	The minimum length for string, secure string, and array type parameters, this value is inclusive.
maxLength	No	The maximum length for string, secure string, and array type parameters, this value is inclusive.
description	No	Description of the parameter that is displayed to users through the portal. For more information, see Comments in templates.

For examples of how to use parameters, see Parameters in ARM templates.

In Bicep, see parameters.

Variables

In the `variables` section, you construct values that can be used throughout your template. You don't need to define variables, but they often simplify your template by

reducing complex expressions. The format of each variable matches one of the data types. You are limited to 256 variables in a template.

The following example shows the available options for defining a variable:

JSON

```
"variables": {
  "<variable-name>": "<variable-value>",
  "<variable-name>": {
    <variable-complex-type-value>
  },
  "<variable-object-name>": {
    "copy": [
      {
        "name": "<name-of-array-property>",
        "count": <number-of-iterations>,
        "input": <object-or-value-to-repeat>
      }
    ]
  },
  "copy": [
    {
      "name": "<variable-array-name>",
      "count": <number-of-iterations>,
      "input": <object-or-value-to-repeat>
    }
  ]
}
```

For information about using `copy` to create several values for a variable, see Variable iteration.

For examples of how to use variables, see Variables in ARM template.

In Bicep, see variables.

Functions

Within your template, you can create your own functions. These functions are available for use in your template. Typically, you define complicated expressions that you don't want to repeat throughout your template. You create the user-defined functions from expressions and functions that are supported in templates.

When defining a user function, there are some restrictions:

- The function can't access variables.

- The function can only use parameters that are defined in the function. When you use the parameters function within a user-defined function, you're restricted to the parameters for that function.
- The function can't call other user-defined functions.
- The function can't use the reference function.
- Parameters for the function can't have default values.

JSON

```
"functions": [
  {
    "namespace": "<namespace-for-functions>",
    "members": {
      "<function-name>": {
        "parameters": [
          {
            "name": "<parameter-name>",
            "type": "<type-of-parameter-value>"
          }
        ],
        "output": {
          "type": "<type-of-output-value>",
          "value": "<function-return-value>"
        }
      }
    }
  }
],
```

Element name	Required	Description
namespace	Yes	Namespace for the custom functions. Use to avoid naming conflicts with template functions.
function-name	Yes	Name of the custom function. When calling the function, combine the function name with the namespace. For example, to call a function named `uniqueName` in the namespace contoso, use `"[contoso.uniqueName()]"`.
parameter-name	No	Name of the parameter to be used within the custom function.
parameter-value	No	Type of the parameter value. The allowed types and values are **string**, **securestring**, **int**, **bool**, **object**, **secureObject**, and **array**.
output-type	Yes	Type of the output value. Output values support the same types as function input parameters.

Element name	Required	Description
output-value	Yes	Template language expression that is evaluated and returned from the function.

For examples of how to use custom functions, see User-defined functions in ARM template.

In Bicep, user-defined functions aren't supported. Bicep does support a variety of functions and operators.

Resources

In the `resources` section, you define the resources that are deployed or updated. You are limited to 800 resources in a template.

You define resources with the following structure:

JSON

```
"resources": [
  {
      "condition": "<true-to-deploy-this-resource>",
      "type": "<resource-provider-namespace/resource-type-name>",
      "apiVersion": "<api-version-of-resource>",
      "name": "<name-of-the-resource>",
      "comments": "<your-reference-notes>",
      "location": "<location-of-resource>",
      "dependsOn": [
          "<array-of-related-resource-names>"
      ],
      "tags": {
          "<tag-name1>": "<tag-value1>",
          "<tag-name2>": "<tag-value2>"
      },
      "identity": {
        "type": "<system-assigned-or-user-assigned-identity>",
        "userAssignedIdentities": {
          "<resource-id-of-identity>": {}
        }
      },
      "sku": {
          "name": "<sku-name>",
          "tier": "<sku-tier>",
          "size": "<sku-size>",
          "family": "<sku-family>",
          "capacity": <sku-capacity>
      },
```

```
        "kind": "<type-of-resource>",
        "scope": "<target-scope-for-extension-resources>",
        "copy": {
            "name": "<name-of-copy-loop>",
            "count": <number-of-iterations>,
            "mode": "<serial-or-parallel>",
            "batchSize": <number-to-deploy-serially>
        },
        "plan": {
            "name": "<plan-name>",
            "promotionCode": "<plan-promotion-code>",
            "publisher": "<plan-publisher>",
            "product": "<plan-product>",
            "version": "<plan-version>"
        },
        "properties": {
            "<settings-for-the-resource>",
            "copy": [
                {
                    "name": ,
                    "count": ,
                    "input": {}
                }
            ]
        },
        "resources": [
            "<array-of-child-resources>"
        ]
    }
  ]
```

Element name	Required	Description
condition	No	Boolean value that indicates whether the resource will be provisioned during this deployment. When `true`, the resource is created during deployment. When `false`, the resource is skipped for this deployment. See condition.
type	Yes	Type of the resource. This value is a combination of the namespace of the resource provider and the resource type (such as `Microsoft.Storage/storageAccounts`). To determine available values, see template reference. For a child resource, the format of the type depends on whether it's nested within the parent resource or defined outside of the parent resource. See Set name and type for child resources.

Element name	Required	Description
apiVersion	Yes	Version of the REST API to use for creating the resource. When creating a new template, set this value to the latest version of the resource you're deploying. As long as the template works as needed, keep using the same API version. By continuing to use the same API version, you minimize the risk of a new API version changing how your template works. Consider updating the API version only when you want to use a new feature that is introduced in a later version. To determine available values, see template reference.
name	Yes	Name of the resource. The name must follow URI component restrictions defined in RFC3986. Azure services that expose the resource name to outside parties validate the name to make sure it isn't an attempt to spoof another identity. For a child resource, the format of the name depends on whether it's nested within the parent resource or defined outside of the parent resource. See Set name and type for child resources.
comments	No	Your notes for documenting the resources in your template. For more information, see Comments in templates.
location	Varies	Supported geo-locations of the provided resource. You can select any of the available locations, but typically it makes sense to pick one that is close to your users. Usually, it also makes sense to place resources that interact with each other in the same region. Most resource types require a location, but some types (such as a role assignment) don't require a location. See Set resource location.
dependsOn	No	Resources that must be deployed before this resource is deployed. Resource Manager evaluates the dependencies between resources and deploys them in the correct order. When resources aren't dependent on each other, they're deployed in parallel. The value can be a comma-separated list of a resource names or resource unique identifiers. Only list resources that are deployed in this template. Resources that aren't defined in this template must already exist. Avoid adding unnecessary dependencies as they can slow your deployment and create circular dependencies. For guidance on setting dependencies, see Define the order for deploying resources in ARM templates.
tags	No	Tags that are associated with the resource. Apply tags to logically organize resources across your subscription.

Element name	Required	Description
identity	No	Some resources support managed identities for Azure resources. Those resources have an identity object at the root level of the resource declaration. You can set whether the identity is user-assigned or system-assigned. For user-assigned identities, provide a list of resource IDs for the identities. Set the key to the resource ID and the value to an empty object. For more information, see Configure managed identities for Azure resources on an Azure VM using templates.
sku	No	Some resources allow values that define the SKU to deploy. For example, you can specify the type of redundancy for a storage account.
kind	No	Some resources allow a value that defines the type of resource you deploy. For example, you can specify the type of Azure Cosmos DB instance to create.
scope	No	The scope property is only available for extension resource types. Use it when specifying a scope that is different than the deployment scope. See Setting scope for extension resources in ARM templates.
copy	No	If more than one instance is needed, the number of resources to create. The default mode is parallel. Specify serial mode when you don't want all or the resources to deploy at the same time. For more information, see Create several instances of resources in Azure Resource Manager.
plan	No	Some resources allow values that define the plan to deploy. For example, you can specify the marketplace image for a virtual machine.
properties	No	Resource-specific configuration settings. The values for the properties are the same as the values you provide in the request body for the REST API operation (PUT method) to create the resource. You can also specify a copy array to create several instances of a property. To determine available values, see template reference.
resources	No	Child resources that depend on the resource being defined. Only provide resource types that are permitted by the schema of the parent resource. Dependency on the parent resource isn't implied. You must explicitly define that dependency. See Set name and type for child resources.

In Bicep, see resources.

Outputs

In the `outputs` section, you specify values that are returned from deployment. Typically, you return values from resources that were deployed. You are limited to 64 outputs in a

template.

The following example shows the structure of an output definition:

JSON

```
"outputs": {
  "<output-name>": {
    "condition": "<boolean-value-whether-to-output-value>",
    "type": "<type-of-output-value>",
    "value": "<output-value-expression>",
    "copy": {
      "count": <number-of-iterations>,
      "input": <values-for-the-variable>
    }
  }
}
```

Element name	Required	Description
output-name	Yes	Name of the output value. Must be a valid JavaScript identifier.
condition	No	Boolean value that indicates whether this output value is returned. When `true`, the value is included in the output for the deployment. When `false`, the output value is skipped for this deployment. When not specified, the default value is `true`.
type	Yes	Type of the output value. Output values support the same types as template input parameters. If you specify **securestring** for the output type, the value isn't displayed in the deployment history and can't be retrieved from another template. To use a secret value in more than one template, store the secret in a Key Vault and reference the secret in the parameter file. For more information, see Use Azure Key Vault to pass secure parameter value during deployment.
value	No	Template language expression that is evaluated and returned as output value. Specify either **value** or **copy**.
copy	No	Used to return more than one value for an output. Specify **value** or **copy**. For more information, see Output iteration in ARM templates.

For examples of how to use outputs, see Outputs in ARM template.

In Bicep, see outputs.

Comments and metadata

You have a few options for adding comments and metadata to your template.

Comments

For inline comments, you can use either `//` or `/* ... */`. In Visual Studio Code, save the parameter files with comments as the **JSON with comments (JSONC)** file type, otherwise you will get an error message saying "Comments not permitted in JSON".

> **Note**
>
> When using Azure CLI to deploy templates with comments, use version 2.3.0 or later, and specify the `--handle-extended-json-format` switch.

JSON

```
{
  "type": "Microsoft.Compute/virtualMachines",
  "apiVersion": "2018-10-01",
  "name": "[variables('vmName')]", // to customize name, change it in
variables
  "location": "[parameters('location')]", //defaults to resource group
location
  "dependsOn": [ /* storage account and network interface must be deployed
first */
    "[resourceId('Microsoft.Storage/storageAccounts/',
variables('storageAccountName'))]",
    "[resourceId('Microsoft.Network/networkInterfaces/',
variables('nicName'))]"
  ],
```

In Visual Studio Code, the Azure Resource Manager Tools extension can automatically detect an ARM template and change the language mode. If you see **Azure Resource Manager Template** at the bottom-right corner of Visual Studio Code, you can use the inline comments. The inline comments are no longer marked as invalid.

In Bicep, see comments.

Metadata

You can add a `metadata` object almost anywhere in your template. Resource Manager ignores the object, but your JSON editor may warn you that the property isn't valid. In the object, define the properties you need.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "comments": "This template was developed for demonstration purposes.",
    "author": "Example Name"
  },
```

For `parameters`, add a `metadata` object with a `description` property.

JSON

```
"parameters": {
  "adminUsername": {
    "type": "string",
    "metadata": {
      "description": "User name for the Virtual Machine."
    }
  },
```

When deploying the template through the portal, the text you provide in the description is automatically used as a tip for that parameter.

For `resources`, add a `comments` element or a `metadata` object. The following example shows both a `comments` element and a `metadata` object.

JSON

```
"resources": [
  {
    "type": "Microsoft.Storage/storageAccounts",
    "apiVersion": "2018-07-01",
    "name": "[concat('storage', uniqueString(resourceGroup().id))]",
    "comments": "Storage account used to store VM disks",
    "location": "[parameters('location')]",
    "metadata": {
      "comments": "These tags are needed for policy compliance."
    },
    "tags": {
      "Dept": "[parameters('deptName')]",
      "Environment": "[parameters('environment')]"
    },
    "sku": {
      "name": "Standard_LRS"
    },
    "kind": "Storage",
    "properties": {}
  }
]
```

For `outputs`, add a `metadata` object to the output value.

JSON

```
"outputs": {
  "hostname": {
    "type": "string",
    "value": "
[reference(variables('publicIPAddressName')).dnsSettings.fqdn]",
    "metadata": {
      "comments": "Return the fully qualified domain name"
```

```
    }
  },
```

You can't add a `metadata` object to user-defined functions.

Multi-line strings

You can break a string into multiple lines. For example, see the `location` property and one of the comments in the following JSON example.

> **Note**
>
> To deploy templates with multi-line strings, use Azure PowerShell or Azure CLI. For CLI, use version 2.3.0 or later, and specify the `--handle-extended-json-format` switch.
>
> Multi-line strings aren't supported when you deploy the template through the Azure portal, a DevOps pipeline, or the REST API.

JSON

```
{
  "type": "Microsoft.Compute/virtualMachines",
  "apiVersion": "2018-10-01",
  "name": "[variables('vmName')]", // to customize name, change it in
variables
  "location": "[
    parameters('location')
    ]", //defaults to resource group location
  /*
    storage account and network interface
    must be deployed first
  */
  "dependsOn": [
    "[resourceId('Microsoft.Storage/storageAccounts/',
variables('storageAccountName'))]",
    "[resourceId('Microsoft.Network/networkInterfaces/',
variables('nicName'))]"
  ],
```

In Bicep, see multi-line strings.

Next steps

- To view complete templates for many different types of solutions, see the Azure Quickstart Templates.
- For details about the functions you can use from within a template, see ARM template functions.
- To combine several templates during deployment, see Using linked and nested templates when deploying Azure resources.
- For recommendations about creating templates, see ARM template best practices.
- For answers to common questions, see Frequently asked questions about ARM templates.

Syntax and expressions in ARM templates

Article • 03/09/2023

The basic syntax of the Azure Resource Manager template (ARM template) is JavaScript Object Notation (JSON). However, you can use expressions to extend the JSON values available within the template. Expressions start and end with brackets: `[` and `]`, respectively. The value of the expression is evaluated when the template is deployed. An expression can return a string, integer, boolean, array, or object.

A template expression can't exceed 24,576 characters.

Use functions

Azure Resource Manager provides functions that you can use in a template. The following example shows an expression that uses a function in the default value of a parameter:

JSON

```
"parameters": {
  "location": {
    "type": "string",
    "defaultValue": "[resourceGroup().location]"
  }
},
```

Within the expression, the syntax `resourceGroup()` calls one of the functions that Resource Manager provides for use within a template. In this case, it's the resourceGroup function. Just like in JavaScript, function calls are formatted as `functionName(arg1,arg2,arg3)`. The syntax `.location` retrieves one property from the object returned by that function.

Template functions and their parameters are case-insensitive. For example, Resource Manager resolves `variables('var1')` and `VARIABLES('VAR1')` as the same. When evaluated, unless the function expressly modifies case (such as `toUpper` or `toLower`), the function preserves the case. Certain resource types may have case requirements that are separate from how functions are evaluated.

To pass a string value as a parameter to a function, use single quotes.

```JSON
"name": "[concat('storage', uniqueString(resourceGroup().id))]"
```

Most functions work the same whether deployed to a resource group, subscription, management group, or tenant. The following functions have restrictions based on the scope:

- resourceGroup - can only be used in deployments to a resource group.
- resourceId - can be used at any scope, but the valid parameters change depending on the scope.
- subscription - can only be used in deployments to a resource group or subscription.

Escape characters

To have a literal string start with a left bracket `[` and end with a right bracket `]`, but not have it interpreted as an expression, add an extra bracket to start the string with `[[`. For example, the variable:

```JSON
"demoVar1": "[[test value]"
```

Resolves to `[test value]`.

However, if the literal string doesn't end with a bracket, don't escape the first bracket. For example, the variable:

```JSON
"demoVar2": "[test] value"
```

Resolves to `[test] value`.

To escape double quotes in an expression, such as adding a JSON object in the template, use the backslash.

```JSON
"tags": {
    "CostCenter": "{\"Dept\":\"Finance\",\"Environment\":\"Production\"}"
},
```

To escape single quotes in an ARM expression output, double up the single quotes. The output of the following template will result in JSON value `{"abc":"'quoted'"}`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {},
  "resources": [],
  "outputs": {
    "foo": {
      "type": "object",
      "value": "[createObject('abc', '''quoted''')]"
    }
  }
}
```

When passing in parameter values, the use of escape characters depends on where the parameter value is specified. If you set a default value in the template, you need the extra left bracket.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "demoParam1": {
      "type": "string",
      "defaultValue": "[[test value]"
    }
  },
  "resources": [],
  "outputs": {
    "exampleOutput": {
      "type": "string",
      "value": "[parameters('demoParam1')]"
    }
  }
}
```

If you use the default value, the template returns `[test value]`.

However, if you pass in a parameter value through the command line, the characters are interpreted literally. Deploying the previous template with:

Azure PowerShell

```
New-AzResourceGroupDeployment -ResourceGroupName demoGroup -TemplateFile
azuredeploy.json -demoParam1 "[[test value]"
```

Returns `[[test value]`. Instead, use:

Azure PowerShell

```
New-AzResourceGroupDeployment -ResourceGroupName demoGroup -TemplateFile
azuredeploy.json -demoParam1 "[test value]"
```

The same formatting applies when passing values in from a parameter file. The characters are interpreted literally. When used with the preceding template, the following parameter file returns `[test value]`:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "demoParam1": {
      "value": "[test value]"
    }
  }
}
```

Null values

To set a property to null, you can use `null` or `[json('null')]`. The json function returns an empty object when you provide `null` as the parameter. In both cases, Resource Manager templates treat it as if the property isn't present.

JSON

```
"stringValue": null,
"objectValue": "[json('null')]"
```

To totally remove an element, you can use the filter() function. For example:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "deployCaboodle": {
      "type": "bool",
      "defaultValue": false
    }
  },
  "variables": {
    "op": [
      {
        "name": "ODB"
      },
      {
        "name": "ODBRPT"
      },
      {
        "name": "Caboodle"
      }
    ]
  },
  "resources": {},
  "outputs": {
    "backendAddressPools": {
      "type": "array",
      "value": "[if(parameters('deployCaboodle'), variables('op'), filter(variables('op'), lambda('on', not(equals(lambdaVariables('on').name, 'Caboodle')))))]"
    }
  }
}
```

Next steps

- For the full list of template functions, see ARM template functions.
- For more information about template files, see Understand the structure and syntax of ARM templates.

Parameters in ARM templates

Article • 04/09/2023

This article describes how to define and use parameters in your Azure Resource Manager template (ARM template). By providing different values for parameters, you can reuse a template for different environments.

Resource Manager resolves parameter values before starting the deployment operations. Wherever the parameter is used in the template, Resource Manager replaces it with the resolved value.

Each parameter must be set to one of the data types.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **parameters**.

You are limited to 256 parameters in a template. For more information, see Template limits.

For parameter best practices, see Parameters.

Minimal declaration

At a minimum, every parameter needs a name and type.

When you deploy a template via the Azure portal, camel-cased parameter names are turned into space-separated names. For example, *demoString* in the following example is shown as *Demo String*. For more information, see Use a deployment button to deploy templates from GitHub repository and Deploy resources with ARM templates and Azure portal.

JSON

```
"parameters": {
  "demoString": {
    "type": "string"
  },
  "demoInt": {
    "type": "int"
  },
  "demoBool": {
```

```
    "type": "bool"
  },
  "demoObject": {
    "type": "object"
  },
  "demoArray": {
    "type": "array"
  }
}
```

Secure parameters

You can mark string or object parameters as secure. The value of a secure parameter isn't saved to the deployment history and isn't logged.

JSON

```
"parameters": {
  "demoPassword": {
    "type": "secureString"
  },
  "demoSecretObject": {
    "type": "secureObject"
  }
}
```

Allowed values

You can define allowed values for a parameter. You provide the allowed values in an array. The deployment fails during validation if a value is passed in for the parameter that isn't one of the allowed values.

JSON

```
"parameters": {
  "demoEnum": {
    "type": "string",
    "allowedValues": [
      "one",
      "two"
    ]
  }
}
```

Default value

You can specify a default value for a parameter. The default value is used when a value isn't provided during deployment.

JSON

```
"parameters": {
  "demoParam": {
    "type": "string",
    "defaultValue": "Contoso"
  }
}
```

To specify a default value along with other properties for the parameter, use the following syntax.

JSON

```
"parameters": {
  "demoParam": {
    "type": "string",
    "defaultValue": "Contoso",
    "allowedValues": [
      "Contoso",
      "Fabrikam"
    ]
  }
}
```

You can use expressions with the default value. You can't use the reference function or any of the list functions in the parameters section. These functions get the runtime state of a resource, and can't be executed before deployment when parameters are resolved.

Expressions aren't allowed with other parameter properties.

JSON

```
"parameters": {
  "location": {
    "type": "string",
    "defaultValue": "[resourceGroup().location]"
  }
}
```

You can use another parameter value to build a default value. The following template constructs a host plan name from the site name.

JSON

```
"parameters": {
  "siteName": {
    "type": "string",
    "defaultValue": "[concat('site', uniqueString(resourceGroup().id))]"
  },
  "hostingPlanName": {
    "type": "string",
    "defaultValue": "[concat(parameters('siteName'),'-plan')]"
  }
}
```

Length constraints

You can specify minimum and maximum lengths for string and array parameters. You can set one or both constraints. For strings, the length indicates the number of characters. For arrays, the length indicates the number of items in the array.

The following example declares two parameters. One parameter is for a storage account name that must have 3-24 characters. The other parameter is an array that must have from 1-5 items.

JSON

```
"parameters": {
  "storageAccountName": {
    "type": "string",
    "minLength": 3,
    "maxLength": 24
  },
  "appNames": {
    "type": "array",
    "minLength": 1,
    "maxLength": 5
  }
}
```

Integer constraints

You can set minimum and maximum values for integer parameters. You can set one or both constraints.

JSON

```
"parameters": {
  "month": {
    "type": "int",
```

```
    "minValue": 1,
    "maxValue": 12
  }
}
```

Description

You can add a description to a parameter to help users of your template understand the value to provide. When deploying the template through the portal, the text you provide in the description is automatically used as a tip for that parameter. Only add a description when the text provides more information than can be inferred from the parameter name.

JSON

```
"parameters": {
  "virtualMachineSize": {
    "type": "string",
    "metadata": {
      "description": "Must be at least Standard_A3 to support 2 NICs."
    },
    "defaultValue": "Standard_DS1_v2"
  }
}
```

Use parameter

To reference a parameter's value, use the parameters function. The following example uses a parameter value for a key vault name.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "vaultName": {
      "type": "string",
      "defaultValue": "[format('keyVault{0}',
uniqueString(resourceGroup().id))]"
    }
  },
  "resources": [
    {
      "type": "Microsoft.KeyVault/vaults",
      "apiVersion": "2021-06-01-preview",
```

```
      "name": "[parameters('vaultName')]",
      ...
    }
  ]
}
```

Objects as parameters

You can organize related values by passing them in as an object. This approach also reduces the number of parameters in the template.

The following example shows a parameter that is an object. The default value shows the expected properties for the object. Those properties are used when defining the resource to deploy.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "vNetSettings": {
      "type": "object",
      "defaultValue": {
        "name": "VNet1",
        "location": "eastus",
        "addressPrefixes": [
          {
            "name": "firstPrefix",
            "addressPrefix": "10.0.0.0/22"
          }
        ],
        "subnets": [
          {
            "name": "firstSubnet",
            "addressPrefix": "10.0.0.0/24"
          },
          {
            "name": "secondSubnet",
            "addressPrefix": "10.0.1.0/24"
          }
        ]
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Network/virtualNetworks",
      "apiVersion": "2021-02-01",
```

```
      "name": "[parameters('vNetSettings').name]",
      "location": "[parameters('vNetSettings').location]",
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[parameters('vNetSettings').addressPrefixes[0].addressPrefix]"
          ]
        },
        "subnets": [
          {
            "name": "[parameters('vNetSettings').subnets[0].name]",
            "properties": {
              "addressPrefix": "
[parameters('vNetSettings').subnets[0].addressPrefix]"
            }
          },
          {
            "name": "[parameters('vNetSettings').subnets[1].name]",
            "properties": {
              "addressPrefix": "
[parameters('vNetSettings').subnets[1].addressPrefix]"
            }
          }
        ]
      }
    }
  ]
}
```

Example templates

The following examples demonstrate scenarios for using parameters.

Template	Description
parameters with functions for default values	Demonstrates how to use template functions when defining default values for parameters. The template doesn't deploy any resources. It constructs parameter values and returns those values.
parameter object	Demonstrates using an object for a parameter. The template doesn't deploy any resources. It constructs parameter values and returns those values.

Next steps

- To learn about the available properties for parameters, see Understand the structure and syntax of ARM templates.
- To learn about passing in parameter values as a file, see Create Resource Manager parameter file.

- For recommendations about creating parameters, see Best practices - parameters.

Variables in ARM templates

Article • 04/09/2023

This article describes how to define and use variables in your Azure Resource Manager template (ARM template). You use variables to simplify your template. Rather than repeating complicated expressions throughout your template, you define a variable that contains the complicated expression. Then, you use that variable as needed throughout your template.

Resource Manager resolves variables before starting the deployment operations. Wherever the variable is used in the template, Resource Manager replaces it with the resolved value.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **variables**.

You are limited to 256 variables in a template. For more information, see Template limits.

Define variable

When defining a variable, you don't specify a data type for the variable. Instead provide a value or template expression. The variable type is inferred from the resolved value. The following example sets a variable to a string.

JSON

```
"variables": {
  "stringVar": "example value"
},
```

To construct the variable, you can use the value from a parameter or another variable.

JSON

```
"parameters": {
  "inputValue": {
    "defaultValue": "deployment parameter",
    "type": "string"
  }
},
"variables": {
```

```
  "stringVar": "myVariable",
  "concatToVar": "[concat(variables('stringVar'), '-addtovar') ]",
  "concatToParam": "[concat(parameters('inputValue'), '-addtoparam')]"
}
```

You can use template functions to construct the variable value.

The following example creates a string value for a storage account name. It uses several template functions to get a parameter value, and concatenates it to a unique string.

JSON

```
"variables": {
  "storageName": "[concat(toLower(parameters('storageNamePrefix')),
uniqueString(resourceGroup().id))]"
},
```

You can't use the reference function or any of the list functions in the variable declaration. These functions get the runtime state of a resource, and can't be executed before deployment when variables are resolved.

Use variable

The following example shows how to use the variable for a resource property.

To reference the value for the variable, use the variables function.

JSON

```
"variables": {
  "storageName": "[concat(toLower(parameters('storageNamePrefix')),
uniqueString(resourceGroup().id))]"
},
"resources": [
  {
    "type": "Microsoft.Storage/storageAccounts",
    "name": "[variables('storageName')]",
    ...
  }
]
```

Example template

The following template doesn't deploy any resources. It shows some ways of declaring variables of different types.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "inputValue": {
      "defaultValue": "deployment parameter",
      "type": "string"
    }
  },
  "variables": {
    "stringVar": "myVariable",
    "concatToVar": "[concat(variables('stringVar'), '-addtovar') ]",
    "concatToParam": "[concat(parameters('inputValue'), '-addtoparam')]",
    "arrayVar": [
      1,
      2,
      3,
      4
    ],
    "objectVar": {
      "property1": "value1",
      "property2": "value2"
    },
    "copyWithinVar": {
      "copy": [
        {
          "name": "disks",
          "count": 5,
          "input": {
            "name": "[concat('myDataDisk', copyIndex('disks', 1))]",
            "diskSizeGB": "1",
            "diskIndex": "[copyIndex('disks')]"
          }
        },
        {
          "name": "diskNames",
          "count": 5,
          "input": "[concat('myDataDisk', copyIndex('diskNames', 1))]"
        }
      ]
    },
    "copy": [
      {
        "name": "topLevelCopy1",
        "count": 5,
        "input": {
          "name": "[concat('oneDataDisk', copyIndex('topLevelCopy1', 1))]",
          "diskSizeGB": "1",
          "diskIndex": "[copyIndex('topLevelCopy1')]"
        }
      },
```

```
      {
        "name": "topLevelCopy2",
        "count": 3,
        "input": {
          "name": "[concat('twoDataDisk', copyIndex('topLevelCopy2', 1))]",
          "diskSizeGB": "1",
          "diskIndex": "[copyIndex('topLevelCopy2')]"
        }
      },
      {
        "name": "topLevelCopy3",
        "count": 4,
        "input": "[concat('stringValue', copyIndex('topLevelCopy3'))]"
      },
      {
        "name": "topLevelCopy4",
        "count": 4,
        "input": "[copyIndex('topLevelCopy4')]"
      }
    ]
  },
  "resources": [],
  "outputs": {
    "stringOutput": {
      "type": "string",
      "value": "[variables('stringVar')]"
    },
    "concatToVariableOutput": {
      "type": "string",
      "value": "[variables('concatToVar')]"
    },
    "concatToParameterOutput": {
      "type": "string",
      "value": "[variables('concatToParam')]"
    },
    "arrayOutput": {
      "type": "array",
      "value": "[variables('arrayVar')]"
    },
    "arrayElementOutput": {
      "type": "int",
      "value": "[variables('arrayVar')[0]]"
    },
    "objectOutput": {
      "type": "object",
      "value": "[variables('objectVar')]"
    },
    "copyWithinVariableOutput": {
      "type": "object",
      "value": "[variables('copyWithinVar')]"
    },
    "topLevelCopyOutput1": {
      "type": "array",
      "value": "[variables('topLevelCopy1')]"
    },
```

```
    "topLevelCopyOutput2": {
      "type": "array",
      "value": "[variables('topLevelCopy2')]"
    },
    "topLevelCopyOutput3": {
      "type": "array",
      "value": "[variables('topLevelCopy3')]"
    },
    "topLevelCopyOutput4": {
      "type": "array",
      "value": "[variables('topLevelCopy4')]"
    }
  }
}
```

Configuration variables

You can define variables that hold related values for configuring an environment. You define the variable as an object with the values. The following example shows an object that holds values for two environments - **test** and **prod**. Pass in one of these values during deployment.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "environmentName": {
      "type": "string",
      "allowedValues": [
        "test",
        "prod"
      ],
      "metadata": {
        "description": "Specify either test or prod for configuration
values."
      }
    }
  },
  "variables": {
    "environmentSettings": {
      "test": {
        "instanceSize": "Small",
        "instanceCount": 1
      },
      "prod": {
        "instanceSize": "Large",
        "instanceCount": 4
```

```
      }
    }
  },
  "resources": [],
  "outputs": {
    "instanceSize": {
      "value": "[variables('environmentSettings')
[parameters('environmentName')].instanceSize]",
      "type": "string"
    },
    "instanceCount": {
      "value": "[variables('environmentSettings')
[parameters('environmentName')].instanceCount]",
      "type": "int"
    }
  }
}
```

Next steps

- To learn about the available properties for variables, see Understand the structure and syntax of ARM templates.
- For recommendations about creating variables, see Best practices - variables.
- For an example template that assigns security rules to a network security group, see network security rules and parameter file.

User-defined functions in ARM template

Article • 05/05/2023

Within your template, you can create your own functions. These functions are available for use in your template. User-defined functions are separate from the standard template functions that are automatically available within your template. Create your own functions when you have complicated expressions that are used repeatedly in your template.

This article describes how to add user-defined functions in your Azure Resource Manager template (ARM template).

Define the function

Your functions require a namespace value to avoid naming conflicts with template functions. The following example shows a function that returns a unique name:

JSON

```
"functions": [
  {
    "namespace": "contoso",
    "members": {
      "uniqueName": {
        "parameters": [
          {
            "name": "namePrefix",
            "type": "string"
          }
        ],
        "output": {
          "type": "string",
          "value": "[concat(toLower(parameters('namePrefix')),
uniqueString(resourceGroup().id))]"
        }
      }
    }
  }
],
```

Use the function

The following example shows a template that includes a user-defined function to get a unique name for a storage account. The template has a parameter named

`storageNamePrefix` that is passed as a parameter to the function.

JSON

```
{
 "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
 "contentVersion": "1.0.0.0",
 "parameters": {
   "storageNamePrefix": {
     "type": "string",
     "maxLength": 11
   }
 },
 "functions": [
  {
    "namespace": "contoso",
    "members": {
      "uniqueName": {
        "parameters": [
          {
            "name": "namePrefix",
            "type": "string"
          }
        ],
        "output": {
          "type": "string",
          "value": "[concat(toLower(parameters('namePrefix')), uniqueString(resourceGroup().id))]"
        }
      }
    }
  }
],
 "resources": [
   {
     "type": "Microsoft.Storage/storageAccounts",
     "apiVersion": "2019-04-01",
     "name": "[contoso.uniqueName(parameters('storageNamePrefix'))]",
     "location": "South Central US",
     "sku": {
       "name": "Standard_LRS"
     },
     "kind": "StorageV2",
     "properties": {
       "supportsHttpsTrafficOnly": true
     }
   }
 ]
}
```

During deployment, the `storageNamePrefix` parameter is passed to the function:

- The template defines a parameter named `storageNamePrefix`.
- The function uses `namePrefix` because you can only use parameters defined in the function. For more information, see Limitations.
- In the template's `resources` section, the `name` element uses the function and passes the `storageNamePrefix` value to the function's `namePrefix`.

Limitations

When defining a user function, there are some restrictions:

- The function can't access variables.
- The function can only use parameters that are defined in the function. When you use the parameters function within a user-defined function, you're restricted to the parameters for that function.
- The function can't call other user-defined functions.
- The function can't use the reference function or any of the list functions.
- Parameters for the function can't have default values.

Next steps

- To learn about the available properties for user-defined functions, see Understand the structure and syntax of ARM templates.
- For a list of the available template functions, see ARM template functions.

Resource declaration in ARM templates

Article • 04/09/2023

To deploy a resource through an Azure Resource Manager template (ARM template), you add a resource declaration. Use the `resources` array in a JSON template.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **resource declaration**.

You are limited to 800 resources in a template. For more information, see Template limits.

Set resource type and version

When adding a resource to your template, start by setting the resource type and API version. These values determine the other properties that are available for the resource.

The following example shows how to set the resource type and API version for a storage account. The example doesn't show the full resource declaration.

JSON

```
"resources": [
  {
    "type": "Microsoft.Storage/storageAccounts",
    "apiVersion": "2019-06-01",
    ...
  }
]
```

Set resource name

Each resource has a name. When setting the resource name, pay attention to the rules and restrictions for resource names.

JSON

```
"parameters": {
  "storageAccountName": {
    "type": "string",
```

```
    "minLength": 3,
    "maxLength": 24
  }
},
"resources": [
  {
    "type": "Microsoft.Storage/storageAccounts",
    "apiVersion": "2019-06-01",
    "name": "[parameters('storageAccountName')]",
    ...
  }
]
```

Set location

Many resources require a location. You can determine if the resource needs a location either through intellisense or template reference. The following example adds a location parameter that is used for the storage account.

JSON

```
"parameters": {
  "storageAccountName": {
    "type": "string",
    "minLength": 3,
    "maxLength": 24
  },
  "location": {
    "type": "string",
    "defaultValue": "[resourceGroup().location]"
  }
},
"resources": [
  {
    "type": "Microsoft.Storage/storageAccounts",
    "apiVersion": "2019-06-01",
    "name": "[parameters('storageAccountName')]",
    "location": "[parameters('location')]",
    ...
  }
]
```

For more information, see Set resource location in ARM template.

Set tags

You can apply tags to a resource during deployment. Tags help you logically organize your deployed resources. For examples of the different ways you can specify the tags, see ARM template tags.

Set resource-specific properties

The preceding properties are generic to most resource types. After setting those values, you need to set the properties that are specific to the resource type you're deploying.

Use intellisense or template reference to determine which properties are available and which ones are required. The following example sets the remaining properties for a storage account.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageAccountName": {
      "type": "string",
      "minLength": 3,
      "maxLength": 24
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    }
  },
  "functions": [],
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2019-06-01",
      "name": "[parameters('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard_LRS",
        "tier": "Standard"
      },
      "kind": "StorageV2",
      "properties": {
        "accessTier": "Hot"
      }
    }
  ]
}
```

Next steps

- To conditionally deploy a resource, see Conditional deployment in ARM templates.
- To set resource dependencies, see Define the order for deploying resources in ARM templates.

Set name and type for child resources

Article • 04/09/2023

Child resources are resources that exist only within the context of another resource. For example, a virtual machine extension can't exist without a virtual machine. The extension resource is a child of the virtual machine.

Each parent resource accepts only certain resource types as child resources. The resource type for the child resource includes the resource type for the parent resource. For example, `Microsoft.Web/sites/config` and `Microsoft.Web/sites/extensions` are both child resources of the `Microsoft.Web/sites`. The accepted resource types are specified in the template schema of the parent resource.

In an Azure Resource Manager template (ARM template), you can specify the child resource either within the parent resource or outside of the parent resource. The values you provide for the resource name and resource type vary based on whether the child resource is defined inside or outside of the parent resource.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **child resources**.

Within parent resource

The following example shows the child resource included within the resources property of the parent resource.

JSON

```
"resources": [
  {
    <parent-resource>
    "resources": [
      <child-resource>
    ]
  }
]
```

Child resources can only be defined five levels deep.

When defined within the parent resource type, you format the type and name values as a single segment without slashes.

JSON

```
"type": "{child-resource-type}",
"name": "{child-resource-name}",
```

The following example shows a virtual network and with a subnet. Notice that the subnet is included within the resources array for the virtual network. The name is set to **Subnet1** and the type is set to **subnets**. The child resource is marked as dependent on the parent resource because the parent resource must exist before the child resource can be deployed.

JSON

```
"resources": [
  {
    "type": "Microsoft.Network/virtualNetworks",
    "apiVersion": "2018-10-01",
    "name": "VNet1",
    "location": "[parameters('location')]",
    "properties": {
      "addressSpace": {
        "addressPrefixes": [
          "10.0.0.0/16"
        ]
      }
    },
    "resources": [
      {
        "type": "subnets",
        "apiVersion": "2018-10-01",
        "name": "Subnet1",
        "dependsOn": [
          "VNet1"
        ],
        "properties": {
          "addressPrefix": "10.0.0.0/24"
        }
      }
    ]
  }
]
```

The full resource type is still `Microsoft.Network/virtualNetworks/subnets`. You don't provide `Microsoft.Network/virtualNetworks/` because it's assumed from the parent resource type.

The child resource name is set to **Subnet1** but the full name includes the parent name. You don't provide **VNet1** because it's assumed from the parent resource.

Outside parent resource

The following example shows the child resource outside of the parent resource. You might use this approach if the parent resource isn't deployed in the same template, or if want to use copy to create more than one child resource.

JSON

```
"resources": [
  {
    <parent-resource>
  },
  {
    <child-resource>
  }
]
```

When defined outside of the parent resource, you format the type and name values with slashes to include the parent type and name.

JSON

```
"type": "{resource-provider-namespace}/{parent-resource-type}/{child-resource-type}",
"name": "{parent-resource-name}/{child-resource-name}",
```

The following example shows a virtual network and subnet that are both defined at the root level. Notice that the subnet isn't included within the resources array for the virtual network. The name is set to **VNet1/Subnet1** and the type is set to `Microsoft.Network/virtualNetworks/subnets`. The child resource is marked as dependent on the parent resource because the parent resource must exist before the child resource can be deployed.

JSON

```
"resources": [
  {
    "type": "Microsoft.Network/virtualNetworks",
    "apiVersion": "2018-10-01",
    "name": "VNet1",
    "location": "[parameters('location')]",
    "properties": {
      "addressSpace": {
```

```
        "addressPrefixes": [
          "10.0.0.0/16"
        ]
      }
    }
  },
  {
    "type": "Microsoft.Network/virtualNetworks/subnets",
    "apiVersion": "2018-10-01",
    "name": "VNet1/Subnet1",
    "dependsOn": [
      "VNet1"
    ],
    "properties": {
      "addressPrefix": "10.0.0.0/24"
    }
  }
]
```

Next steps

- To learn about creating ARM templates, see Understand the structure and syntax of ARM templates.
- To learn about the format of the resource name when referencing the resource, see the reference function.

Conditional deployment in ARM templates

Article • 05/12/2023

Sometimes you need to optionally deploy a resource in an Azure Resource Manager template (ARM template). Use the `condition` element to specify whether the resource is deployed. The value for the condition resolves to true or false. When the value is true, the resource is created. When the value is false, the resource isn't created. The value can only be applied to the whole resource.

> **Note**
>
> Conditional deployment doesn't cascade to **child resources**. If you want to conditionally deploy a resource and its child resources, you must apply the same condition to each resource type.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **conditional deployments**.

Deploy condition

You can pass in a parameter value that indicates whether a resource is deployed. The following example conditionally deploys a DNS zone.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "deployZone": {
      "type": "bool"
    }
  },
  "functions": [],
  "resources": [
    {
      "condition": "[parameters('deployZone')]",
```

```
      "type": "Microsoft.Network/dnsZones",
      "apiVersion": "2018-05-01",
      "name": "myZone",
      "location": "global"
    }
  ]
}
```

For a more complex example, see Azure SQL logical server.

New or existing resource

You can use conditional deployment to create a new resource or use an existing one. The following example shows how to either deploy a new storage account or use an existing storage account.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageAccountName": {
      "type": "string"
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    },
    "newOrExisting": {
      "type": "string",
      "defaultValue": "new",
      "allowedValues": [
        "new",
        "existing"
      ]
    }
  },
  "resources": {
    "saNew": {
      "condition": "[equals(parameters('newOrExisting'), 'new')]",
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2022-09-01",
      "name": "[parameters('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "StorageV2"
    },
```

```
    "saExisting": {
      "condition": "[equals(parameters('newOrExisting'), 'existing')]",
      "existing": true,
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2022-09-01",
      "name": "[parameters('storageAccountName')]"
    }
  },
  "outputs": {
    "storageAccountId": {
      "type": "string",
      "value": "[if(equals(parameters('newOrExisting'), 'new'),
resourceId('Microsoft.Storage/storageAccounts',
parameters('storageAccountName')),
resourceId('Microsoft.Storage/storageAccounts',
parameters('storageAccountName')))]"
    }
  }
}
```

When the parameter `newOrExisting` is set to **new**, the condition evaluates to true. The storage account is deployed. Otherwise the existing storage account is used.

For a complete example template that uses the `condition` element, see VM with a new or existing Virtual Network, Storage, and Public IP.

Runtime functions

If you use a reference or list function with a resource that is conditionally deployed, the function is evaluated even if the resource isn't deployed. You get an error if the function refers to a resource that doesn't exist.

Use the if function to make sure the function is only evaluated for conditions when the resource is deployed. See the if function for a sample template that uses `if` and `reference` with a conditionally deployed resource.

You set a resource as dependent on a conditional resource exactly as you would any other resource. When a conditional resource isn't deployed, Azure Resource Manager automatically removes it from the required dependencies.

Complete mode

If you deploy a template with complete mode and a resource isn't deployed because `condition` evaluates to false, the result depends on which REST API version you use to deploy the template. If you use a version earlier than 2019-05-10, the resource **isn't**

deleted. With 2019-05-10 or later, the resource **is deleted**. The latest versions of Azure PowerShell and Azure CLI delete the resource when condition is false.

Next steps

- For a Learn module that covers conditional deployment, see Manage complex cloud deployments by using advanced ARM template features.
- For recommendations about creating templates, see ARM template best practices.
- To create multiple instances of a resource, see Resource iteration in ARM templates.

Define the order for deploying resources in ARM templates

Article • 09/20/2022

When deploying resources, you may need to make sure some resources exist before other resources. For example, you need a logical SQL server before deploying a database. You establish this relationship by marking one resource as dependent on the other resource. Use the `dependsOn` element to define an explicit dependency. Use the **reference** or **list** functions to define an implicit dependency.

Azure Resource Manager evaluates the dependencies between resources, and deploys them in their dependent order. When resources aren't dependent on each other, Resource Manager deploys them in parallel. You only need to define dependencies for resources that are deployed in the same template.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **resource dependencies**.

dependsOn

Within your Azure Resource Manager template (ARM template), the `dependsOn` element enables you to define one resource as a dependent on one or more resources. Its value is a JavaScript Object Notation (JSON) array of strings, each of which is a resource name or ID. The array can include resources that are conditionally deployed. When a conditional resource isn't deployed, Azure Resource Manager automatically removes it from the required dependencies.

The following example shows a network interface that depends on a virtual network, network security group, and public IP address. For the full template, see the quickstart template for a Linux VM.

JSON

```
{
    "type": "Microsoft.Network/networkInterfaces",
    "apiVersion": "2020-06-01",
    "name": "[variables('networkInterfaceName')]",
    "location": "[parameters('location')]",
```

```
    "dependsOn": [
      "[resourceId('Microsoft.Network/networkSecurityGroups/',
parameters('networkSecurityGroupName'))]",
      "[resourceId('Microsoft.Network/virtualNetworks/',
parameters('virtualNetworkName'))]",
      "[resourceId('Microsoft.Network/publicIpAddresses/',
variables('publicIpAddressName'))]"
    ],
    ...
}
```

While you may be inclined to use `dependsOn` to map relationships between your resources, it's important to understand why you're doing it. For example, to document how resources are interconnected, `dependsOn` isn't the right approach. After deployment, the resource doesn't retain deployment dependencies in its properties, so there are no commands or operations that let you see dependencies. Setting unnecessary dependencies slows deployment time because Resource Manager can't deploy those resources in parallel.

Child resources

An implicit deployment dependency isn't automatically created between a child resource and the parent resource. If you need to deploy the child resource after the parent resource, set the `dependsOn` property.

The following example shows a logical SQL server and database. Notice that an explicit dependency is defined between the database and the server, even though the database is a child of the server.

JSON

```
"resources": [
  {
    "type": "Microsoft.Sql/servers",
    "apiVersion": "2020-02-02-preview",
    "name": "[parameters('serverName')]",
    "location": "[parameters('location')]",
    "properties": {
      "administratorLogin": "[parameters('administratorLogin')]",
      "administratorLoginPassword": "
[parameters('administratorLoginPassword')]"
    },
    "resources": [
      {
        "type": "databases",
        "apiVersion": "2020-08-01-preview",
        "name": "[parameters('sqlDBName')]",
```

```
        "location": "[parameters('location')]",
        "sku": {
          "name": "Standard",
          "tier": "Standard"
          },
        "dependsOn": [
          "[resourceId('Microsoft.Sql/servers',
concat(parameters('serverName')))]"
        ]
      }
    ]
  }
]
```

For the full template, see quickstart template for Azure SQL Database.

reference and list functions

The reference function enables an expression to derive its value from other JSON name and value pairs or runtime resources. The list* functions return values for a resource from a list operation.

Reference and list expressions implicitly declare that one resource depends on another. Whenever possible, use an implicit reference to avoid adding an unnecessary dependency.

To enforce an implicit dependency, refer to the resource by name, not resource ID. If you pass the resource ID into the reference or list functions, an implicit reference isn't created.

The general format of the `reference` function is:

JSON

```
reference('resourceName').propertyPath
```

The general format of the `listKeys` function is:

JSON

```
listKeys('resourceName', 'yyyy-mm-dd')
```

In the following example, a CDN endpoint explicitly depends on the CDN profile, and implicitly depends on a web app.

JSON

```
{
    "name": "[variables('endpointName')]",
    "apiVersion": "2016-04-02",
    "type": "endpoints",
    "location": "[resourceGroup().location]",
    "dependsOn": [
      "[variables('profileName')]"
    ],
    "properties": {
      "originHostHeader": "
[reference(variables('webAppName')).hostNames[0]]",
      ...
    }
    ...
}
```

To learn more, see reference function.

Depend on resources in a loop

To deploy resources that depend on resources in a copy loop, you have two options. You can either set a dependency on individual resources in the loop or on the whole loop.

> **Note**
>
> For most scenarios, you should set the dependency on individual resources within the copy loop. Only depend on the whole loop when you need all of the resources in the loop to exist before creating the next resource. Setting the dependency on the whole loop causes the dependencies graph to expand significantly, especially if those looped resources depend on other resources. The expanded dependencies make it difficult for the deployment to complete efficiently.

The following example shows how to deploy multiple virtual machines. The template creates the same number of network interfaces. Each virtual machine is dependent on one network interface, rather than the whole loop.

JSON

```
{
  "type": "Microsoft.Network/networkInterfaces",
  "apiVersion": "2020-05-01",
  "name": "[concat(variables('nicPrefix'),'-',copyIndex())]",
  "location": "[parameters('location')]",
  "copy": {
```

```
    "name": "nicCopy",
    "count": "[parameters('vmCount')]"
  },
  ...
},
{
  "type": "Microsoft.Compute/virtualMachines",
  "apiVersion": "2020-06-01",
  "name": "[concat(variables('vmPrefix'),copyIndex())]",
  "location": "[parameters('location')]",
  "dependsOn": [
    "
[resourceId('Microsoft.Network/networkInterfaces',concat(variables('nicPrefi
x'),'-',copyIndex()))]"
  ],
  "copy": {
    "name": "vmCopy",
    "count": "[parameters('vmCount')]"
  },
  "properties": {
    "networkProfile": {
      "networkInterfaces": [
        {
          "id": "
[resourceId('Microsoft.Network/networkInterfaces',concat(variables('nicPrefi
x'),'-',copyIndex()))]",
          "properties": {
            "primary": "true"
          }
        }
      ]
    },
    ...
  }
}
```

The following example shows how to deploy three storage accounts before deploying the virtual machine. Notice that the `copy` element has `name` set to `storagecopy` and the `dependsOn` element for the virtual machine is also set to `storagecopy`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {},
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2019-04-01",
      "name": "[concat(copyIndex(),'storage',
```

```
uniqueString(resourceGroup().id))]",
      "location": "[resourceGroup().location]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "Storage",
      "copy": {
        "name": "storagecopy",
        "count": 3
      },
      "properties": {}
    },
    {
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2015-06-15",
      "name": "[concat('VM', uniqueString(resourceGroup().id))]",
      "dependsOn": ["storagecopy"],
      ...
    }
  ],
  "outputs": {}
}
```

Circular dependencies

Resource Manager identifies circular dependencies during template validation. If you receive an error for a circular dependency, evaluate your template to see if any dependencies can be removed. If removing dependencies doesn't work, you can avoid circular dependencies by moving some deployment operations into child resources. Deploy the child resources after the resources that have the circular dependency. For example, suppose you're deploying two virtual machines but you must set properties on each one that refer to the other. You can deploy them in the following order:

1. vm1
2. vm2
3. Extension on vm1 depends on vm1 and vm2. The extension sets values on vm1 that it gets from vm2.
4. Extension on vm2 depends on vm1 and vm2. The extension sets values on vm2 that it gets from vm1.

For information about assessing the deployment order and resolving dependency errors, see Troubleshoot common Azure deployment errors with Azure Resource Manager.

Next steps

- To go through a tutorial, see Tutorial: Create ARM templates with dependent resources.
- For a Learn module that covers resource dependencies, see Manage complex cloud deployments by using advanced ARM template features.
- For recommendations when setting dependencies, see ARM template best practices.
- To learn about troubleshooting dependencies during deployment, see Troubleshoot common Azure deployment errors with Azure Resource Manager.
- To learn about creating Azure Resource Manager templates, see Understand the structure and syntax of ARM templates.
- For a list of the available functions in a template, see ARM template functions.

Set resource location in ARM template

Article • 03/09/2023

When deploying an Azure Resource Manager template (ARM template), you must provide a location for each resource. The location doesn't need to be the same location as the resource group location.

Get available locations

Different resource types are supported in different locations. To get the supported locations for a resource type, use Azure PowerShell or Azure CLI.

PowerShell

Azure PowerShell

```
((Get-AzResourceProvider -ProviderNamespace
Microsoft.Batch).ResourceTypes `
  | Where-Object ResourceTypeName -eq batchAccounts).Locations
```

Use location parameter

To allow for flexibility when deploying your template, use a parameter to specify the location for resources. Set the default value of the parameter to `resourceGroup().location`.

The following example shows a storage account that is deployed to a location specified as a parameter:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageAccountType": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
```

```
        "Standard_ZRS",
        "Premium_LRS"
      ],
      "metadata": {
        "description": "Storage Account type"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    }
  },
  "variables": {
    "storageAccountName": "[concat('storage',
uniquestring(resourceGroup().id))]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2018-07-01",
      "name": "[variables('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('storageAccountType')]"
      },
      "kind": "StorageV2",
      "properties": {}
    }
  ],
  "outputs": {
    "storageAccountName": {
      "type": "string",
      "value": "[variables('storageAccountName')]"
    }
  }
}
```

Next steps

- For the full list of template functions, see ARM template functions.
- For more information about template files, see Understand the structure and syntax of ARM templates.

Post-deployment configurations by using extensions

Article • 04/09/2023

Azure Resource Manager template (ARM template) extensions are small applications that provide post-deployment configuration and automation tasks on Azure resources. The most popular one is virtual machine extensions. See Virtual machine extensions and features for Windows, and Virtual machine extensions and features for Linux.

Extensions

The existing extensions are:

- Microsoft.Compute/virtualMachines/extensions
- Microsoft.Compute virtualMachineScaleSets/extensions
- Microsoft.HDInsight clusters/extensions
- Microsoft.Sql servers/databases/extensions
- Microsoft.Web/sites/siteextensions

To find out the available extensions, browse to the template reference. In **Filter by title**, enter **extension**.

To learn how to use these extensions, see:

- Tutorial: Deploy virtual machine extensions with ARM templates.
- Tutorial: Import SQL BACPAC files with ARM templates

Next steps

Tutorial: Deploy virtual machine extensions with ARM templates

Outputs in ARM templates

Article • 03/09/2023

This article describes how to define output values in your Azure Resource Manager template (ARM template). You use outputs when you need to return values from the deployed resources.

The format of each output value must resolve to one of the data types.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **outputs**.

You are limited to 64 outputs in a template. For more information, see Template limits.

Define output values

The following example shows how to return a property from a deployed resource.

Add the `outputs` section to the template. The output value gets the fully qualified domain name for a public IP address.

JSON

```
"outputs": {
  "hostname": {
    "type": "string",
    "value": "[reference(resourceId('Microsoft.Network/publicIPAddresses',
variables('publicIPAddressName'))).dnsSettings.fqdn]"
  },
}
```

If you need to output a property that has a hyphen in the name, use brackets around the name instead of dot notation. For example, use `['property-name']` instead of `.property-name`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
```

```
  "variables": {
    "user": {
      "user-name": "Test Person"
    }
  },
  "resources": [
  ],
  "outputs": {
    "nameResult": {
      "type": "string",
      "value": "[variables('user')['user-name']]"
    }
  }
}
```

Conditional output

You can use the `condition` element to conditionally return a value. Typically, you use a conditional output when you've conditionally deployed a resource. The following example shows how to conditionally return the resource ID for a public IP address based on whether a new one was deployed:

JSON

```
"outputs": {
  "resourceID": {
    "condition": "[equals(parameters('publicIpNewOrExisting'), 'new')]",
    "type": "string",
    "value": "[resourceId('Microsoft.Network/publicIPAddresses',
parameters('publicIPAddresses_name'))]"
  }
}
```

For a simple example of conditional output, see conditional output template.

Dynamic number of outputs

In some scenarios, you don't know the number of instances of a value you need to return when creating the template. You can return a variable number of values by using iterative output. Add the `copy` element to iterate an output.

JSON

```
"outputs": {
  "storageEndpoints": {
    "type": "array",
```

```
    "copy": {
      "count": "[parameters('storageCount')]",
      "input": "[reference(concat(copyIndex(),
variables('baseName'))).primaryEndpoints.blob]"
    }
  }
}
```

For more information, see Output iteration in ARM templates.

Linked templates

You can deploy related templates by using linked templates. To retrieve the output value from a linked template, use the reference function in the parent template. The syntax in the parent template is:

JSON

```
"[reference('<deploymentName>').outputs.<propertyName>.value]"
```

The following example shows how to set the IP address on a load balancer by retrieving a value from a linked template.

JSON

```
"publicIPAddress": {
  "id": "[reference('linkedTemplate').outputs.resourceID.value]"
}
```

If the property name has a hyphen, use brackets around the name instead of dot notation.

JSON

```
"publicIPAddress": {
  "id": "[reference('linkedTemplate').outputs['resource-ID'].value]"
}
```

You can't use the `reference` function in the outputs section of a nested template. To return the values for a deployed resource in a nested template, convert your nested template to a linked template.

The Public IP address template creates a public IP address and outputs the resource ID. The Load balancer template links to the preceding template. It uses the resource

ID in the output when creating the load balancer.

Example template

The following template doesn't deploy any resources. It shows some ways of returning outputs of different types.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [],
  "outputs": {
    "stringOutput": {
      "type": "string",
      "value": "[deployment().name]"
    },
    "integerOutput": {
      "type": "int",
      "value": "[length(environment().authentication.audiences)]"
    },
    "booleanOutput": {
      "type": "bool",
      "value": "[contains(deployment().name, 'demo')]"
    },
    "arrayOutput": {
      "type": "array",
      "value": "[environment().authentication.audiences]"
    },
    "objectOutput": {
      "type": "object",
      "value": "[subscription()]"
    }
  }
}
```

Get output values

When the deployment succeeds, the output values are automatically returned in the results of the deployment.

To get output values from the deployment history, you can use script.

PowerShell

Azure PowerShell

```
(Get-AzResourceGroupDeployment `
  -ResourceGroupName <resource-group-name> `
  -Name <deployment-name>).Outputs.resourceID.value
```

Object sorting in outputs

In JSON, an object is an unordered collection of zero or more key/value pairs. The ordering can be different depending on the implementations. For example, the Bicep items() function sorts the objects in the alphabetical order. In other places, the original ordering can be preserved. Because of this non-determinism, avoid making any assumptions about the ordering of object keys when writing code, which interacts with deployments parameters & outputs.

Next steps

- To learn about the available properties for outputs, see Understand the structure and syntax of ARM templates.

Use deployment scripts in ARM templates

Article • 05/11/2023

Learn how to use deployment scripts in Bicep. With the deploymentScripts resource, users can execute scripts in Bicep deployments and review execution results.

These scripts can be used for performing custom steps such as:

- Add users to a directory.
- Perform data plane operations, for example, copy blobs or seed database.
- Look up and validate a license key.
- Create a self-signed certificate.
- Create an object in Azure Active Directory (Azure AD).
- Look up IP Address blocks from custom system.

The benefits of deployment script:

- Easy to code, use, and debug. You can develop deployment scripts in your favorite development environments. The scripts can be embedded in templates or in external script files.
- You can specify the script language and platform. Currently, Azure PowerShell and Azure CLI deployment scripts on the Linux environment are supported.
- Allow passing command-line arguments to the script.
- Can specify script outputs and pass them back to the deployment.

The deployment script resource is only available in the regions where Azure Container Instance is available. See Resource availability for Azure Container Instances in Azure regions. Currently, deployment script only uses public networking.

> **Important**
>
> The deployment script service requires two supporting resources for script execution and troubleshooting: a storage account and a container instance. You can specify an existing storage account, otherwise the script service creates one for you. The two automatically-created supporting resources are usually deleted by the script service when the deployment script execution gets in a terminal state. You are billed for the supporting resources until they are deleted. For the price information, see **Container Instances pricing** and **Azure Storage pricing**. To learn more, see **Clean-up deployment script resources**.

> **Note**
>
> Retry logic for Azure sign in is now built in to the wrapper script. If you grant permissions in the same template as your deployment scripts, the deployment script service retries sign in for 10 minutes with 10-second interval until the managed identity role assignment is replicated.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **Deployment script**.

Training resources

If you would rather learn about deployment scripts through step-by-step guidance, see Extend ARM templates by using deployment scripts.

Configure the minimum permissions

For deployment script API version 2020-10-01 or later, there are two principals involved in deployment script execution:

- **Deployment principal** (the principal used to deploy the template): this principal is used to create underlying resources required for the deployment script resource to execute — a storage account and an Azure container instance. To configure the least-privilege permissions, assign a custom role with the following properties to the deployment principal:

  ```
  JSON
  ```

```
{
  "roleName": "deployment-script-minimum-privilege-for-deployment-principal",
  "description": "Configure least privilege for the deployment principal in deployment script",
  "type": "customRole",
  "IsCustom": true,
  "permissions": [
    {
      "actions": [
        "Microsoft.Storage/storageAccounts/*",
        "Microsoft.ContainerInstance/containerGroups/*",
        "Microsoft.Resources/deployments/*",
        "Microsoft.Resources/deploymentScripts/*"
      ],
    }
  ],
  "assignableScopes": [
    "[subscription().id]"
  ]
}
```

If the Azure Storage and the Azure Container Instance resource providers haven't been registered, you also need to add `Microsoft.Storage/register/action` and `Microsoft.ContainerInstance/register/action`.

- **Deployment script principal**: This principal is only required if the deployment script needs to authenticate to Azure and call Azure CLI/PowerShell. There are two ways to specify the deployment script principal:
  - Specify a user-assigned managed identity in the `identity` property (see Sample templates). When specified, the script service calls `Connect-AzAccount -Identity` before invoking the deployment script. The managed identity must have the required access to complete the operation in the script. Currently, only user-assigned managed identity is supported for the `identity` property. To log in with a different identity, use the second method in this list.
  - Pass the service principal credentials as secure environment variables, and then can call Connect-AzAccount or az login in the deployment script.

  If a managed identity is used, the deployment principal needs the **Managed Identity Operator** role (a built-in role) assigned to the managed identity resource.

Sample templates

The following JSON is an example. For more information, see the latest template schema.

JSON

```
{
  "type": "Microsoft.Resources/deploymentScripts",
  "apiVersion": "2020-10-01",
  "name": "runPowerShellInline",
  "location": "[resourceGroup().location]",
  "tags": {
    "tagName1": "tagValue1",
    "tagName2": "tagValue2"
  },
  "kind": "AzurePowerShell", // or "AzureCLI"
  "identity": {
    "type": "userAssigned",
    "userAssignedIdentities": {
      "/subscriptions/01234567-89AB-CDEF-0123-
456789ABCDEF/resourceGroups/myResourceGroup/providers/Microsoft.ManagedIdentity/userAssignedIdentities/myID": {}
    }
  },
  "properties": {
    "forceUpdateTag": "1",
    "containerSettings": {
      "containerGroupName": "mycustomaci"
    },
    "storageAccountSettings": {
      "storageAccountName": "myStorageAccount",
      "storageAccountKey": "myKey"
    },
    "azPowerShellVersion": "9.7",  // or "azCliVersion": "2.47.0",
    "arguments": "-name \\\"John Dole\\\"",
    "environmentVariables": [
      {
        "name": "UserName",
        "value": "jdole"
      },
```

```
      {
        "name": "Password",
        "secureValue": "jDolePassword"
      }
    ],
    "scriptContent": "
      param([string] $name)
      $output = 'Hello {0}. The username is {1}, the password is {2}.' -f $name,${Env:UserName},${Env:Password}
      Write-Output $output
      $DeploymentScriptOutputs = @{}
      $DeploymentScriptOutputs['text'] = $output
    ", // or "primaryScriptUri": "https://raw.githubusercontent.com/Azure/azure-docs-json-samples/master/deployment-
script/deploymentscript-helloworld.ps1",
    "supportingScriptUris":[],
    "timeout": "PT30M",
    "cleanupPreference": "OnSuccess",
    "retentionInterval": "P1D"
  }
}
```

> **Note**
>
> The example is for demonstration purposes. The properties `scriptContent` and `primaryScriptUri` can't coexist in a template.

> **Note**
>
> The *scriptContent* shows a script with multiple lines. The Azure portal and Azure DevOps pipeline can't parse a deployment script with multiple lines. You can either chain the PowerShell commands (by using semicolons or *\r\n* or *\n*) into one line, or use the `primaryScriptUri` property with an external script file. There are many free JSON string escape/unescape tools available. For example, **https://www.freeformatter.com/json-escape.html**.

Property value details:

- `identity`: For deployment script API version 2020-10-01 or later, a user-assigned managed identity is optional unless you need to perform any Azure-specific actions in the script. For the API version 2019-10-01-preview, a managed identity is required as the deployment script service uses it to execute the scripts. When the identity property is specified, the script service calls `Connect-AzAccount -Identity` before invoking the user script. Currently, only user-assigned managed identity is supported. To log in with a different identity, you can call Connect-AzAccount in the script.
- `tags`: Deployment script tags. If the deployment script service generates a storage account and a container instance, the tags are passed to both resources, which can be used to identify them. Another way to identify these resources is through their suffixes, which contain "azscripts". For more information, see Monitor and troubleshoot deployment scripts.
- `kind`: Specify the type of script. Currently, Azure PowerShell and Azure CLI scripts are supported. The values are **AzurePowerShell** and **AzureCLI**.
- `forceUpdateTag`: Changing this value between template deployments forces the deployment script to re-execute. If you use the `newGuid()` or the `utcNow()` functions, both functions can only be used in the default value for a parameter. To learn more, see Run script more than once.
- `containerSettings`: Specify the settings to customize Azure Container Instance. Deployment script requires a new Azure Container Instance. You can't specify an existing Azure Container Instance. However, you can customize the container group name by using `containerGroupName`. If not specified, the group name is automatically generated.
- `storageAccountSettings`: Specify the settings to use an existing storage account. If `storageAccountName` isn't specified, a storage account is automatically created. See Use an existing storage account.
- `azPowerShellVersion`/`azCliVersion`: Specify the module version to be used. See a list of supported Azure PowerShell versions. The version determines which container image to use:
  - **Az version greater than or equal to 9** uses Ubuntu 22.04.
  - **Az version greater than or equal to 6 but less than 9** uses Ubuntu 20.04.
  - **Az version less than 6** uses Ubuntu 18.04.

  > **Important**

It is advisable to upgrade to the latest version of Ubuntu, as Ubuntu 18.04 is nearing its end of life and will no longer receive security updates beyond **May 31st, 2023**.

See a list of supported Azure CLI versions.

> **Important**
>
> Deployment script uses the available CLI images from Microsoft Container Registry (MCR). It typically takes approximatedly one month to certify a CLI image for deployment script. Don't use the CLI versions that were released within 30 days. To find the release dates for the images, see **Azure CLI release notes**. If an unsupported version is used, the error message lists the supported versions.

- `arguments`: Specify the parameter values. The values are separated by spaces.

  Deployment Scripts splits the arguments into an array of strings by invoking the CommandLineToArgvW system call. This step is necessary because the arguments are passed as a command property to Azure Container Instance, and the command property is an array of string.

  If the arguments contain escaped characters, use JsonEscaper to double escaped the characters. Paste your original escaped string into the tool, and then select **Escape**. The tool outputs a double escaped string. For example, in the previous sample template, The argument is `-name \"John Dole\"`. The escaped string is `-name \\\"John Dole\\\"`.

  To pass an ARM template parameter of type object as an argument, convert the object to a string by using the string() function, and then use the replace() function to replace any `\"` into `\\\"`. For example:

  ```
  JSON
  replace(string(parameters('tables')), '\"', '\\\"')
  ```

  For more information, see the sample template.

- `environmentVariables`: Specify the environment variables to pass over to the script. For more information, see Develop deployment scripts.

- `scriptContent`: Specify the script content. To run an external script, use `primaryScriptUri` instead. For examples, see Use inline script and Use external script.

- `primaryScriptUri`: Specify a publicly accessible URL to the primary deployment script with supported file extensions. For more information, see Use external scripts.

- `supportingScriptUris`: Specify an array of publicly accessible URLs to supporting files that are called in either `scriptContent` or `primaryScriptUri`. For more information, see Use external scripts.

- `timeout`: Specify the maximum allowed script execution time specified in the ISO 8601 format. Default value is **P1D**.

- `cleanupPreference`. Specify the preference of cleaning up the two supporting deployment resources, the storage account and the container instance, when the script execution gets in a terminal state. Default setting is **Always**, which means deleting the supporting resources despite the terminal state (Succeeded, Failed, Canceled). To learn more, see Clean up deployment script resources.

- `retentionInterval`: Specify the interval for which the service retains the deployment script resource after the deployment script execution reaches a terminal state. The deployment script resource is deleted when this duration expires. Duration is based on the ISO 8601 pattern. The retention interval is between 1 and 26 hours (PT26H). This property is used when `cleanupPreference` is set to **OnExpiration**. To learn more, see Clean up deployment script resources.

More samples

- Sample 1: create a key vault and use deployment script to assign a certificate to the key vault.
- Sample 2: create a resource group at the subscription level, create a key vault in the resource group, and then use deployment script to assign a certificate to the key vault.
- Sample 3: create a user-assigned managed identity, assign the contributor role to the identity at the resource group level, create a key vault, and then use deployment script to assign a certificate to the key vault.
- Sample 4: it is the same scenario as Sample 1 in this list. A new resource group is created to run the deployment script. This template is a subscription level template.
- Sample 5: it is the same scenario as Sample 4. This template is a resource group level template.

- Sample 6: manually create a user-assigned managed identity and assign it permission to use the Microsoft Graph API to create Azure AD applications; in the Bicep file, use a deployment script to create an Azure AD application and service principal, and output the object IDs and client ID.

Use inline scripts

The following template has one resource defined with the `Microsoft.Resources/deploymentScripts` type. The highlighted part is the inline script.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "name": {
      "type": "string",
      "defaultValue": "\\\"John Dole\\\""
    },
    "utcValue": {
      "type": "string",
      "defaultValue": "[utcNow()]"
    }
  },
  "resources": [
    {
      "type": "Microsoft.Resources/deploymentScripts",
      "apiVersion": "2020-10-01",
      "name": "runPowerShellInlineWithOutput",
      "location": "[resourceGroup().location]",
      "kind": "AzurePowerShell",
      "properties": {
        "forceUpdateTag": "[parameters('utcValue')]",
        "azPowerShellVersion": "8.3",
        "scriptContent": "
          param([string] $name)
          $output = \"Hello {0}\" -f $name
          Write-Output $output
          $DeploymentScriptOutputs = @{}
          $DeploymentScriptOutputs['text'] = $output
        ",
        "arguments": "[concat('-name', ' ', parameters('name'))]",
        "timeout": "PT1H",
        "cleanupPreference": "OnSuccess",
        "retentionInterval": "P1D"
      }
    }
  ],
  "outputs": {
    "result": {
      "value": "[reference('runPowerShellInlineWithOutput').outputs.text]",
      "type": "string"
    }
  }
}
```

> **Note**
>
> Because the inline deployment scripts are enclosed in double quotes, the strings inside the deployment scripts need to be escaped by using a backslash (\) or enclosed in single quotes. You can also consider using string substitution as it is shown in the previous JSON sample.

The script takes one parameter, and output the parameter value. `DeploymentScriptOutputs` is used for storing outputs. In the outputs section, the `value` line shows how to access the stored values. `Write-Output` is used for debugging purpose. To learn how to access the output file, see Monitor and troubleshoot deployment scripts. For the property descriptions, see Sample templates.

To run the script, select **Try it** to open the Cloud Shell, and then paste the following code into the shell pane.

Azure PowerShell

```
$resourceGroupName = Read-Host -Prompt "Enter the name of the resource group to be created"
$location = Read-Host -Prompt "Enter the location (i.e. centralus)"
```

```
New-AzResourceGroup -Name $resourceGroupName -Location $location

New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName -TemplateUri
"https://raw.githubusercontent.com/Azure/azure-docs-json-samples/master/deployment-script/deploymentscript-helloworld.json"

Write-Host "Press [ENTER] to continue ..."
```

The output looks like:

```
Outputs                 :
                          Name             Type                       Value
                          ===============  =========================  ==========
                          result           String                     Hello John Dole
```

Use external scripts

In addition to inline scripts, you can also use external script files. Only primary PowerShell scripts with the *ps1* file extension are supported. For CLI scripts, the primary scripts can have any extensions (or without an extension), as long as the scripts are valid bash scripts. To use external script files, replace `scriptContent` with `primaryScriptUri`. For example:

JSON

```
"primaryScriptUri": "https://raw.githubusercontent.com/Azure/azure-docs-json-samples/master/deployment-
script/deploymentscript-helloworld.ps1",
```

For more information, see the example template.

The external script files must be accessible. To secure your script files that are stored in Azure storage accounts, generate a SAS token and include it in the URI for the template. Set the expiry time to allow enough time to complete the deployment. For more information, see Deploy private ARM template with SAS token.

You're responsible for ensuring the integrity of the scripts that are referenced by deployment script, either `primaryScriptUri` or `supportingScriptUris`. Reference only scripts that you trust.

Use supporting scripts

You can separate complicated logics into one or more supporting script files. The `supportingScriptUris` property allows you to provide an array of URIs to the supporting script files if needed:

JSON

```
"scriptContent": "
    ...
    ./Create-Cert.ps1
    ...
"

"supportingScriptUris": [
  "https://raw.githubusercontent.com/Azure/azure-docs-json-samples/master/deployment-script/create-cert.ps1"
],
```

Supporting script files can be called from both inline scripts and primary script files. Supporting script files have no restrictions on the file extension.

The supporting files are copied to `azscripts/azscriptinput` at the runtime. Use relative path to reference the supporting files from inline scripts and primary script files.

Work with outputs from PowerShell script

The following template shows how to pass values between two `deploymentScripts` resources:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "name": {
```

```
      "type": "string",
      "defaultValue": "John Dole"
    },
    "utcValue": {
      "type": "string",
      "defaultValue": "[utcNow()]"
    }
  },
  "resources": [
    {
      "type": "Microsoft.Resources/deploymentScripts",
      "apiVersion": "2020-10-01",
      "name": "scriptInTemplate1",
      "location": "[resourceGroup().location]",
      "kind": "AzurePowerShell",
      "properties": {
        "forceUpdateTag": "[parameters('utcValue')]",
        "azPowerShellVersion": "8.3",
        "timeout": "PT1H",
        "arguments": "[concat('-name', ' ', concat('\\\"', parameters('name'), '\\\"'))]",
        "scriptContent": "
          param([string] $name)
          $output = 'Hello {0}' -f $name
          Write-Output $output
          $DeploymentScriptOutputs = @{}
          $DeploymentScriptOutputs['text'] = $output
        ",
        "cleanupPreference": "Always",
        "retentionInterval": "P1D"
      }
    },
    {
      "type": "Microsoft.Resources/deploymentScripts",
      "apiVersion": "2020-10-01",
      "name": "scriptInTemplate2",
      "location": "[resourceGroup().location]",
      "kind": "AzurePowerShell",
      "dependsOn": [
        "scriptInTemplate1"
      ],
      "properties": {
        "forceUpdateTag": "[parameters('utcValue')]",
        "azPowerShellVersion": "8.3",
        "timeout": "PT1H",
        "arguments": "[concat('-textToEcho', ' ', concat('\\\"', reference('scriptInTemplate1').outputs.text, '\\\"'))]",
        "scriptContent": "
          param([string] $textToEcho)
          Write-Output $textToEcho
          $DeploymentScriptOutputs = @{}
          $DeploymentScriptOutputs['text'] = $textToEcho
        ",
        "cleanupPreference": "Always",
        "retentionInterval": "P1D"
      }
    }
  ],
  "outputs": {
    "result": {
      "value": "[reference('scriptInTemplate2').outputs.text]",
      "type": "string"
    }
  }
}
```

In the first resource, you define a variable called `$DeploymentScriptOutputs`, and use it to store the output values. To access the output value from another resource within the template, use:

```
JSON

reference('<ResourceName>').outputs.text
```

Work with outputs from CLI script

Different from the PowerShell deployment script, CLI/bash support doesn't expose a common variable to store script outputs, instead, there's an environment variable called `AZ_SCRIPTS_OUTPUT_PATH` that stores the location where the script outputs file resides. If a deployment

script is run from a Resource Manager template, this environment variable is set automatically for you by the Bash shell. The value of `AZ_SCRIPTS_OUTPUT_PATH` is */mnt/azscripts/azscriptoutput/scriptoutputs.json*.

Deployment script outputs must be saved in the `AZ_SCRIPTS_OUTPUT_PATH` location, and the outputs must be a valid JSON string object. The contents of the file must be saved as a key-value pair. For example, an array of strings is stored as `{ "MyResult": [ "foo", "bar"] }`. Storing just the array results, for example `[ "foo", "bar" ]`, is invalid.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "identity": {
      "type": "string"
    },
    "utcValue": {
      "type": "string",
      "defaultValue": "[utcNow()]"
    }
  },
  "resources": [
    {
      "type": "Microsoft.Resources/deploymentScripts",
      "apiVersion": "2020-10-01",
      "name": "runBashWithOutputs",
      "location": "[resourceGroup().location]",
      "kind": "AzureCLI",
      "identity": {
        "type": "UserAssigned",
        "userAssignedIdentities": {
          "[parameters('identity')]": {
          }
        }
      },
      "properties": {
        "forceUpdateTag": "[parameters('utcValue')]",
        "AzCliVersion": "2.40.0",
        "timeout": "PT30M",
        "arguments": "'foo' 'bar'",
        "environmentVariables": [
          {
            "name": "UserName",
            "value": "jdole"
          },
          {
            "name": "Password",
            "secureValue": "jDolePassword"
          }
        ],
        "scriptContent": "result=$(az keyvault list); echo \"arg1 is: $1\"; echo \"arg2 is: $2\"; echo \"Username is:
$UserName\"; echo \"password is: $Password\"; echo $result | jq -c '{Result: map({id: .id})}' > $AZ_SCRIPTS_OUTPUT_PATH",
        "cleanupPreference": "OnSuccess",
        "retentionInterval": "P1D"
```

jq is used in the previous sample. It comes with the container images. See Configure development environment.

Use existing storage account

A storage account and a container instance are needed for script execution and troubleshooting. You have the options to specify an existing storage account, otherwise the storage account along with the container instance are automatically created by the script service. The requirements for using an existing storage account:

- Supported storage account kinds are:

SKU	Supported Kind
Premium_LRS	FileStorage
Premium_ZRS	FileStorage
Standard_GRS	Storage, StorageV2
Standard_GZRS	StorageV2

SKU	Supported Kind
Standard_LRS	Storage, StorageV2
Standard_RAGRS	Storage, StorageV2
Standard_RAGZRS	StorageV2
Standard_ZRS	StorageV2

These combinations support file shares. For more information, see Create an Azure file share and Types of storage accounts.

- Storage account firewall rules aren't supported yet. For more information, see Configure Azure Storage firewalls and virtual networks.
- Deployment principal must have permissions to manage the storage account, which includes read, create, delete file shares.

To specify an existing storage account, add the following JSON to the property element of `Microsoft.Resources/deploymentScripts`:

JSON

```
"storageAccountSettings": {
  "storageAccountName": "myStorageAccount",
  "storageAccountKey": "myKey"
},
```

- `storageAccountName`: specify the name of the storage account.
- `storageAccountKey`: specify one of the storage account keys. You can use the listKeys() function to retrieve the key. For example:

  JSON

  ```
  "storageAccountSettings": {
      "storageAccountName": "[variables('storageAccountName')]",
      "storageAccountKey": "[listKeys(resourceId('Microsoft.Storage/storageAccounts', variables('storageAccountName')),
  '2019-06-01').keys[0].value]"
  }
  ```

See Sample templates for a complete `Microsoft.Resources/deploymentScripts` definition sample.

When an existing storage account is used, the script service creates a file share with a unique name. See Clean up deployment script resources for how the script service cleans up the file share.

Develop deployment scripts

Handle nonterminating errors

You can control how PowerShell responds to nonterminating errors by using the `$ErrorActionPreference` variable in your deployment script. If the variable isn't set in your deployment script, the script service uses the default value **Continue**.

The script service sets the resource provisioning state to **Failed** when the script encounters an error despite the setting of `$ErrorActionPreference`.

Use environment variables

Deployment script uses these environment variables:

Environment variable	Default value	System reserved
AZ_SCRIPTS_AZURE_ENVIRONMENT	AzureCloud	N
AZ_SCRIPTS_CLEANUP_PREFERENCE	OnExpiration	N
AZ_SCRIPTS_OUTPUT_PATH	<AZ_SCRIPTS_PATH_OUTPUT_DIRECTORY>/<AZ_SCRIPTS_PATH_SCRIPT_OUTPUT_FILE_NAME>	Y
AZ_SCRIPTS_PATH_INPUT_DIRECTORY	/mnt/azscripts/azscriptinput	Y
AZ_SCRIPTS_PATH_OUTPUT_DIRECTORY	/mnt/azscripts/azscriptoutput	Y
AZ_SCRIPTS_PATH_USER_SCRIPT_FILE_NAME	Azure PowerShell: userscript.ps1; Azure CLI: userscript.sh	Y

Environment variable	Default value	System reserved
AZ_SCRIPTS_PATH_PRIMARY_SCRIPT_URI_FILE_NAME	primaryscripturi.config	Y
AZ_SCRIPTS_PATH_SUPPORTING_SCRIPT_URI_FILE_NAME	supportingscripturi.config	Y
AZ_SCRIPTS_PATH_SCRIPT_OUTPUT_FILE_NAME	scriptoutputs.json	Y
AZ_SCRIPTS_PATH_EXECUTION_RESULTS_FILE_NAME	executionresult.json	Y
AZ_SCRIPTS_USER_ASSIGNED_IDENTITY	/subscriptions/	N

For more information about using `AZ_SCRIPTS_OUTPUT_PATH`, see Work with outputs from CLI script.

Pass secured strings to deployment script

Setting environment variables (EnvironmentVariable) in your container instances allows you to provide dynamic configuration of the application or script run by the container. Deployment script handles nonsecured and secured environment variables in the same way as Azure Container Instance. For more information, see Set environment variables in container instances. For an example, see Sample templates.

The max allowed size for environment variables is 64 KB.

Monitor and troubleshoot deployment scripts

The script service creates a storage account (unless you specify an existing storage account) and a container instance for script execution. If these resources are automatically created by the script service, both resources have the `azscripts` suffix in the resource names.

The user script, the execution results, and the stdout file are stored in the files shares of the storage account. There's a folder called `azscripts`. In the folder, there are two more folders for the input and the output files: `azscriptinput` and `azscriptoutput`.

The output folder contains a *executionresult.json* and the script output file. You can see the script execution error message in *executionresult.json*. The output file is created only when the script is executed successfully. The input folder contains a system PowerShell script file and the user deployment script files. You can replace the user deployment script file with a revised one, and rerun the deployment script from the Azure container instance.

Use the Azure portal

After you deploy a deployment script resource, the resource is listed under the resource group in the Azure portal. The following screenshot shows the **Overview** page of a deployment script resource:

The overview page displays some important information of the resource, such as **Provisioning state**, **Storage account**, **Container instance**, and **Logs**.

From the left menu, you can view the deployment script content, the arguments passed to the script, and the output. You can also export a template for the deployment script including the deployment script.

Use PowerShell

Using Azure PowerShell, you can manage deployment scripts at subscription or resource group scope:

- Get-AzDeploymentScript: Gets or lists deployment scripts.
- Get-AzDeploymentScriptLog: Gets the log of a deployment script execution.
- Remove-AzDeploymentScript: Removes a deployment script and its associated resources.
- Save-AzDeploymentScriptLog: Saves the log of a deployment script execution to disk.

The `Get-AzDeploymentScript` output is similar to:

Output

```
Name                : runPowerShellInlineWithOutput
Id                  : /subscriptions/01234567-89AB-CDEF-0123-
456789ABCDEF/resourceGroups/myds0618rg/providers/Microsoft.Resources/deploymentScripts/runPowerShellInlineWithOutput
ResourceGroupName   : myds0618rg
Location            : centralus
SubscriptionId      : 01234567-89AB-CDEF-0123-456789ABCDEF
ProvisioningState   : Succeeded
Identity            : /subscriptions/01234567-89AB-CDEF-0123-
456789ABCDEF/resourceGroups/mydentity1008rg/providers/Microsoft.ManagedIdentity/userAssignedIdentities/myuami
ScriptKind          : AzurePowerShell
AzPowerShellVersion : 9.7
StartTime           : 5/11/2023 7:46:45 PM
EndTime             : 5/11/2023 7:49:45 PM
ExpirationDate      : 5/12/2023 7:49:45 PM
CleanupPreference   : OnSuccess
StorageAccountId    : /subscriptions/01234567-89AB-CDEF-0123-
456789ABCDEF/resourceGroups/myds0618rg/providers/Microsoft.Storage/storageAccounts/ftnlvo6rlrvo2azscripts
ContainerInstanceId : /subscriptions/01234567-89AB-CDEF-0123-
456789ABCDEF/resourceGroups/myds0618rg/providers/Microsoft.ContainerInstance/containerGroups/ftnlvo6rlrvo2azscripts
Outputs             :
                      Key                 Value
                      ==================  ==================
                      text                Hello John Dole
```

```
RetentionInterval   : P1D
Timeout             : PT1H
```

Use Azure CLI

Using Azure CLI, you can manage deployment scripts at subscription or resource group scope:

- az deployment-scripts delete: Delete a deployment script.
- az deployment-scripts list: List all deployment scripts.
- az deployment-scripts show: Retrieve a deployment script.
- az deployment-scripts show-log: Show deployment script logs.

The list command output is similar to:

JSON


```
[
  {
    "arguments": "-name \\\"John Dole\\\"",
    "azPowerShellVersion": "9.7",
    "cleanupPreference": "OnSuccess",
    "containerSettings": {
      "containerGroupName": null
    },
    "environmentVariables": null,
    "forceUpdateTag": "20230511T025902Z",
    "id": "/subscriptions/01234567-89AB-CDEF-0123-
456789ABCDEF/resourceGroups/myds0624rg/providers/Microsoft.Resources/deploymentScripts/runPowerShellInlineWithOutput",
    "identity": {
      "tenantId": "01234567-89AB-CDEF-0123-456789ABCDEF",
      "type": "userAssigned",
      "userAssignedIdentities": {
        "/subscriptions/01234567-89AB-CDEF-0123-
456789ABCDEF/resourceGroups/myidentity1008rg/providers/Microsoft.ManagedIdentity/userAssignedIdentities/myuami": {
          "clientId": "01234567-89AB-CDEF-0123-456789ABCDEF",
          "principalId": "01234567-89AB-CDEF-0123-456789ABCDEF"
        }
      }
    },
    "kind": "AzurePowerShell",
    "location": "centralus",
    "name": "runPowerShellInlineWithOutput",
    "outputs": {
      "text": "Hello John Dole"
    },
    "primaryScriptUri": null,
    "provisioningState": "Succeeded",
    "resourceGroup": "myds0624rg",
    "retentionInterval": "1 day, 0:00:00",
    "scriptContent": "\r\n          param([string] $name)\r\n          $output = \"Hello {0}\" -f $name\r\n          Write-
Output $output\r\n          $DeploymentScriptOutputs = @{}\r\n          $DeploymentScriptOutputs['text'] = $output\r\n
",
    "status": {
      "containerInstanceId": "/subscriptions/01234567-89AB-CDEF-0123-
456789ABCDEF/resourceGroups/myds0624rg/providers/Microsoft.ContainerInstance/containerGroups/64lxews2qfa5uazscripts",
      "endTime": "2023-05-11T03:00:16.796923+00:00",
      "error": null,
      "expirationTime": "2023-05-12T03:00:16.796923+00:00",
      "startTime": "2023-05-11T02:59:07.595140+00:00",
      "storageAccountId": "/subscriptions/01234567-89AB-CDEF-0123-
456789ABCDEF/resourceGroups/myds0624rg/providers/Microsoft.Storage/storageAccounts/64lxews2qfa5uazscripts"
    },
    "storageAccountSettings": null,
    "supportingScriptUris": null,
    "systemData": {
      "createdAt": "2023-05-11T02:59:04.750195+00:00",
      "createdBy": "someone@contoso.com",
      "createdByType": "User",
      "lastModifiedAt": "2023-05-11T02:59:04.750195+00:00",
      "lastModifiedBy": "someone@contoso.com",
      "lastModifiedByType": "User"
    },
    "tags": null,
    "timeout": "1:00:00",
    "type": "Microsoft.Resources/deploymentScripts"
  }
]
```

Use REST API

You can get the deployment script resource deployment information at the resource group level and the subscription level by using REST API:

HTTP

```
/subscriptions/<SubscriptionID>/resourcegroups/<ResourceGroupName>/providers/microsoft.resources/deploymentScripts/<DeploymentScriptResourceName>?api-version=2020-10-01
```

HTTP

```
/subscriptions/<SubscriptionID>/providers/microsoft.resources/deploymentScripts?api-version=2020-10-01
```

The following example uses ARMClient:

Azure PowerShell

```
armclient login
armclient get /subscriptions/01234567-89AB-CDEF-0123-456789ABCDEF/resourcegroups/myrg/providers/microsoft.resources/deploymentScripts/myDeployementScript?api-version=2020-10-01
```

The output is similar to:

JSON


```
{
  "kind": "AzurePowerShell",
  "identity": {
    "type": "userAssigned",
    "tenantId": "01234567-89AB-CDEF-0123-456789ABCDEF",
    "userAssignedIdentities": {
      "/subscriptions/01234567-89AB-CDEF-0123-456789ABCDEF/resourceGroups/myidentity1008rg/providers/Microsoft.ManagedIdentity/userAssignedIdentities/myuami": {
        "principalId": "01234567-89AB-CDEF-0123-456789ABCDEF",
        "clientId": "01234567-89AB-CDEF-0123-456789ABCDEF"
      }
    }
  },
  "location": "centralus",
  "systemData": {
    "createdBy": "someone@contoso.com",
    "createdByType": "User",
    "createdAt": "2023-05-11T02:59:04.7501955Z",
    "lastModifiedBy": "someone@contoso.com",
    "lastModifiedByType": "User",
    "lastModifiedAt": "2023-05-11T02:59:04.7501955Z"
  },
  "properties": {
    "provisioningState": "Succeeded",
    "forceUpdateTag": "20220625T025902Z",
    "azPowerShellVersion": "9.7",
    "scriptContent": "\r\n          param([string] $name)\r\n          $output = \"Hello {0}\" -f $name\r\n          Write-Output $output\r\n          $DeploymentScriptOutputs = @{}\r\n          $DeploymentScriptOutputs['text'] = $output\r\n",
    "arguments": "-name \\\"John Dole\\\"",
    "retentionInterval": "P1D",
    "timeout": "PT1H",
    "containerSettings": {},
    "status": {
      "containerInstanceId": "/subscriptions/01234567-89AB-CDEF-0123-456789ABCDEF/resourceGroups/myds0624rg/providers/Microsoft.ContainerInstance/containerGroups/64lxews2qfa5uazscripts",
      "storageAccountId": "/subscriptions/01234567-89AB-CDEF-0123-456789ABCDEF/resourceGroups/myds0624rg/providers/Microsoft.Storage/storageAccounts/64lxews2qfa5uazscripts",
      "startTime": "2023-05-11T02:59:07.5951401Z",
      "endTime": "2023-05-11T03:00:16.7969234Z",
      "expirationTime": "2023-05-12T03:00:16.7969234Z"
    },
    "outputs": {
      "text": "Hello John Dole"
    },
    "cleanupPreference": "OnSuccess"
  },
  "id": "/subscriptions/01234567-89AB-CDEF-0123-456789ABCDEF/resourceGroups/myds0624rg/providers/Microsoft.Resources/deploymentScripts/runPowerShellInlineWithOutput",
  "type": "Microsoft.Resources/deploymentScripts",
```

```
  "name": "runPowerShellInlineWithOutput"
}
```

The following REST API returns the log:

```HTTP
/subscriptions/<SubscriptionID>/resourcegroups/<ResourceGroupName>/providers/microsoft.resources/deploymentScripts/<DeploymentScriptResourceName>/logs?api-version=2020-10-01
```

It only works before the deployment script resources are deleted.

To see the deploymentScripts resource in the portal, select **Show hidden types**:

Clean up deployment script resources

The two automatically created supporting resources can never outlive the `deploymentScript` resource, unless there are failures deleting them. The life cycle of the supporting resources is controlled by the `cleanupPreference` property, the life cycle of the `deploymentScript` resource is controlled by the `retentionInterval` property:

- `cleanupPreference`: Specify the clean-up preference of the two supporting resources when the script execution gets in a terminal state. The supported values are:
  - **Always**: Delete the two supporting resources once script execution gets in a terminal state. If an existing storage account is used, the script service deletes the file share created by the service. Because the `deploymentScripts` resource may still be present after the supporting resources are cleaned up, the script service persists the script execution results, for example, stdout, outputs, and return value before the resources are deleted.
  - **OnSuccess**: Delete the two supporting resources only when the script execution is successful. If an existing storage account is used, the script service removes the file share only when the script execution is successful.

    If the script execution isn't successful, the script service waits until the `retentionInterval` expires before it cleans up the supporting resources and then the deployment script resource.
  - **OnExpiration**: Delete the two supporting resources only when the `retentionInterval` setting is expired. If an existing storage account is used, the script service removes the file share, but retains the storage account.

  The container instance and storage account are deleted according to the `cleanupPreference`. However, if the script fails and `cleanupPreference` isn't set to **Always**, the deployment process automatically keeps the container running for one hour or until the

container is cleaned up. You can use the time to troubleshoot the script. If you want to keep the container running after successful deployments, add a sleep step to your script. For example, add Start-Sleep to the end of your script. If you don't add the sleep step, the container is set to a terminal state and can't be accessed even if it hasn't been deleted yet.

- `retentionInterval`: Specify the time interval that a `deploymentScript` resource will be retained and after which will be expired and deleted.

> **Note**
>
> It isn't recommended to use the storage account and the container instance that are generated by the script service for other purposes. The two resources might be removed depending on the script life cycle.

The automatically created storage account and container instance can't be deleted if the deployment script is deployed to a resource group with a CanNotDelete lock. To solve this problem, you can deploy the deployment script to another resource group without locks. See Sample 4 and Sample 5 in Sample templates.

Run script more than once

Deployment script execution is an idempotent operation. If none of the `deploymentScripts` resource properties (including the inline script) are changed, the script doesn't execute when you redeploy the template. The deployment script service compares the resource names in the template with the existing resources in the same resource group. There are two options if you want to execute the same deployment script multiple times:

- Change the name of your `deploymentScripts` resource. For example, use the utcNow template function as the resource name or as a part of the resource name. Changing the resource name creates a new `deploymentScripts` resource. It's good for keeping a history of script execution.

  > **Note**
  >
  > The `utcNow` function can only be used in the default value for a parameter.

- Specify a different value in the `forceUpdateTag` template property. For example, use `utcNow` as the value.

> **Note**
>
> Write the deployment scripts that are idempotent. This ensures that if they run again accidentally, it will not cause system changes. For example, if the deployment script is used to create an Azure resource, verify the resource doesn't exist before creating it, so the script will succeed or you don't create the resource again.

Configure development environment

You can use a preconfigured container image as your deployment script development environment. For more information, see Configure development environment for deployment scripts in templates.

After the script is tested successfully, you can use it as a deployment script in your templates.

Deployment script error codes

Error code	Description
DeploymentScriptInvalidOperation	The deployment script resource definition in the template contains invalid property names.
DeploymentScriptResourceConflict	Can't delete a deployment script resource that is in nonterminal state and the execution hasn't exceeded 1 hour. Or can't rerun the same deployment script with the same resource identifier (same subscription, resource group name, and resource name) but different script body content at the same time.
DeploymentScriptOperationFailed	The deployment script operation failed internally. Contact Microsoft support.
DeploymentScriptStorageAccountAccessKeyNotSpecified	The access key hasn't been specified for the existing storage account.

Error code	Description
DeploymentScriptContainerGroupContainsInvalidContainers	A container group created by the deployment script service got externally modified, and invalid containers got added.
DeploymentScriptContainerGroupInNonterminalState	Two or more deployment script resources use the same Azure container instance name in the same resource group, and one of them hasn't finished its execution yet.
DeploymentScriptStorageAccountInvalidKind	The existing storage account of the BlobBlobStorage or BlobStorage type doesn't support file shares, and can't be used.
DeploymentScriptStorageAccountInvalidKindAndSku	The existing storage account doesn't support file shares. For a list of supported storage account kinds, see Use existing storage account.
DeploymentScriptStorageAccountNotFound	The storage account doesn't exist or has been deleted by an external process or tool.
DeploymentScriptStorageAccountWithServiceEndpointEnabled	The storage account specified has a service endpoint. A storage account with a service endpoint isn't supported.
DeploymentScriptStorageAccountInvalidAccessKey	Invalid access key specified for the existing storage account.
DeploymentScriptStorageAccountInvalidAccessKeyFormat	Invalid storage account key format. See Manage storage account access keys.
DeploymentScriptExceededMaxAllowedTime	Deployment script execution time exceeded the timeout value specified in the deployment script resource definition.
DeploymentScriptInvalidOutputs	The deployment script output isn't a valid JSON object.
DeploymentScriptContainerInstancesServiceLoginFailure	The user-assigned managed identity wasn't able to sign in after 10 attempts with 1-minute interval.
DeploymentScriptContainerGroupNotFound	A Container group created by deployment script service got deleted by an external tool or process.
DeploymentScriptDownloadFailure	Failed to download a supporting script. See Use supporting script.
DeploymentScriptError	The user script threw an error.
DeploymentScriptBootstrapScriptExecutionFailed	The bootstrap script threw an error. Bootstrap script is the system script that orchestrates the deployment script execution.
DeploymentScriptExecutionFailed	Unknown error during the deployment script execution.
DeploymentScriptContainerInstancesServiceUnavailable	When creating the Azure container instance (ACI), ACI threw a service unavailable error.
DeploymentScriptContainerGroupInNonterminalState	When creating the Azure container instance (ACI), another deployment script is using the same ACI name in the same scope (same subscription, resource group name, and resource name).
DeploymentScriptContainerGroupNameInvalid	The Azure container instance name (ACI) specified doesn't meet the ACI requirements. See Troubleshoot common issues in Azure Container Instances.

Use Microsoft Graph within a deployment script

A deployment script can use Microsoft Graph to create and work with objects in Azure AD.

Commands

When you use Azure CLI deployment scripts, you can use commands within the `az ad` command group to work with applications, service principals, groups, and users. You can also directly invoke Microsoft Graph APIs by using the `az rest` command.

When you use Azure PowerShell deployment scripts, you can use the `Invoke-RestMethod` cmdlet to directly invoke the Microsoft Graph APIs.

Permissions

The identity that your deployment script uses needs to be authorized to work with the Microsoft Graph API, with the appropriate permissions for the operations it performs. You must authorize the identity outside of your template deployment, such as by precreating a user-assigned managed identity and assigning it an app role for Microsoft Graph. For more information, see this quickstart example.

Access private virtual network

The supporting resources including the container instance can't be deployed to a private virtual network. To access a private virtual network from your deployment script, you can create another virtual network with a publicly accessible virtual machine or a container instance, and create a peering from this virtual network to the private virtual network.

Next steps

In this article, you learned how to use deployment scripts. To walk through a deployment script tutorial:

Tutorial: Use deployment scripts in Azure Resource Manager templates

Learn module: Extend ARM templates by using deployment scripts

Configure development environment for deployment scripts in ARM templates

Article • 04/09/2023

Learn how to create a development environment for developing and testing ARM template deployment scripts with a deployment script image. You can either create an Azure container instance or use Docker. Both options are covered in this article.

Prerequisites

Azure PowerShell container

If you don't have an Azure PowerShell deployment script, you can create a *hello.ps1* file by using the following content:

PowerShell

```
param([string] $name)
$output = 'Hello {0}' -f $name
Write-Output $output
$DeploymentScriptOutputs = @{}
$DeploymentScriptOutputs['text'] = $output
```

Azure CLI container

For an Azure CLI container image, you can create a *hello.sh* file by using the following content:

Bash

```
FIRSTNAME=$1
LASTNAME=$2
OUTPUT="{\"name\":{\"displayName\":\"$FIRSTNAME
$LASTNAME\",\"firstName\":\"$FIRSTNAME\",\"lastName\":\"$LASTNAME\"}}"
echo -n "Hello "
echo $OUTPUT | jq -r '.name.displayName'
```

Note

When you run an Azure CLI deployment script, an environment variable called `AZ_SCRIPTS_OUTPUT_PATH` stores the location of the script output file. The environment variable isn't available in the development environment container. For more information about working with Azure CLI outputs, see **Work with outputs from CLI script**.

Use Azure PowerShell container instance

To author your scripts on your computer, you need to create a storage account and mount the storage account to the container instance. So that you can upload your script to the storage account and run the script on the container instance.

> **Note**
>
> The storage account that you create to test your script is not the same storage account that the deployment script service uses to execute the script. Deployment script service creates a unique name as a file share on every execution.

Create an Azure PowerShell container instance

The following Azure Resource Manager template (ARM template) creates a container instance and a file share, and then mounts the file share to the container image.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "projectName": {
      "type": "string",
      "metadata": {
        "description": "Specify a project name that is used for generating resource names."
      }
    },
    "containerImage": {
      "type": "string",
      "defaultValue": "mcr.microsoft.com/azuredeploymentscripts-powershell:az5.2",
      "metadata": {
        "description": "Specify the container image."
      }
```

```
    },
    "mountPath": {
      "type": "string",
      "defaultValue": "/mnt/azscripts/azscriptinput",
      "metadata": {
        "description": "Specify the mount path."
      }
    }
  },
  "variables": {
    "storageAccountName": "[tolower(concat(parameters('projectName'),
'store'))]",
    "fileShareName": "[concat(parameters('projectName'), 'share')]",
    "containerGroupName": "[concat(parameters('projectName'), 'cg')]",
    "containerName": "[concat(parameters('projectName'), 'container')]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2019-06-01",
      "name": "[variables('storageAccountName')]",
      "location": "[resourceGroup().location]",
      "sku": {
        "name": "Standard_LRS",
        "tier": "Standard"
      },
      "kind": "StorageV2",
      "properties": {
        "accessTier": "Hot"
      }
    },
    {
      "type": "Microsoft.Storage/storageAccounts/fileServices/shares",
      "apiVersion": "2019-06-01",
      "name": "[concat(variables('storageAccountName'), '/default/',
variables('fileShareName'))]",
      "dependsOn": [
        "[resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))]"
      ]
    },
    {
      "type": "Microsoft.ContainerInstance/containerGroups",
      "apiVersion": "2019-12-01",
      "name": "[variables('containerGroupName')]",
      "location": "[resourceGroup().location]",
      "dependsOn": [
        "[resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))]"
      ],
      "properties": {
        "containers": [
          {
            "name": "[variables('containerName')]",
            "properties": {
```

```
                "image": "[parameters('containerImage')]",
                "resources": {
                  "requests": {
                    "cpu": 1,
                    "memoryInGb": 1.5
                  }
                },
                "ports": [
                  {
                    "protocol": "TCP",
                    "port": 80
                  }
                ],
                "volumeMounts": [
                  {
                    "name": "filesharevolume",
                    "mountPath": "[parameters('mountPath')]"
                  }
                ],
                "command": [
                  "/bin/sh",
                  "-c",
                  "pwsh -c 'Start-Sleep -Seconds 1800'"
                ]
              }
            }
          ],
          "osType": "Linux",
          "volumes": [
            {
              "name": "filesharevolume",
              "azureFile": {
                "readOnly": false,
                "shareName": "[variables('fileShareName')]",
                "storageAccountName": "[variables('storageAccountName')]",
                "storageAccountKey": "
[listKeys(resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName')), '2019-06-01').keys[0].value]"
              }
            }
          ]
        }
      }
    ]
}
```

The default value for the mount path is `/mnt/azscripts/azscriptinput`. This is the path in the container instance where it's mounted to the file share.

The default container image specified in the template is **mcr.microsoft.com/azuredeploymentscripts-powershell:az5.2**. See a list of all supported Azure PowerShell versions.

The template suspends the container instance after 1,800 seconds. You have 30 minutes before the container instance goes into a terminated state and the session ends.

To deploy the template:

Azure PowerShell

```
$projectName = Read-Host -Prompt "Enter a project name that is used to
generate resource names"
$location = Read-Host -Prompt "Enter the location (i.e. centralus)"
$templateFile = Read-Host -Prompt "Enter the template file path and file
name"
$resourceGroupName = "${projectName}rg"

New-AzResourceGroup -Location $location -name $resourceGroupName
New-AzResourceGroupDeployment -resourceGroupName $resourceGroupName -
TemplateFile $templatefile -projectName $projectName
```

Upload the deployment script

Upload your deployment script to the storage account. Here's an example of a PowerShell script:

Azure PowerShell

```
$projectName = Read-Host -Prompt "Enter the same project name that you used
earlier"
$fileName = Read-Host -Prompt "Enter the deployment script file name with
the path"

$resourceGroupName = "${projectName}rg"
$storageAccountName = "${projectName}store"
$fileShareName = "${projectName}share"

$context = (Get-AzStorageAccount -ResourceGroupName $resourceGroupName -Name
$storageAccountName).Context
Set-AzStorageFileContent -Context $context -ShareName $fileShareName -Source
$fileName -Force
```

You also can upload the file by using the Azure portal or the Azure CLI.

Test the deployment script

1. In the Azure portal, open the resource group where you deployed the container instance and the storage account.

2. Open the container group. The default container group name is the project name appended with *cg*. The container instance is in the **Running** state.

3. In the resource menu, select **Containers**. The container instance name is the project name appended with *container*.

4. Select **Connect**, and then select **Connect**. If you can't connect to the container instance, restart the container group and try again.

5. In the console pane, run the following commands:

Console

```
cd /mnt/azscripts/azscriptinput
ls
pwsh ./hello.ps1 "John Dole"
```

The output is **Hello John Dole**.

```
root@SandboxHost-637453877092393449:/# cd /mnt/azscripts/azscriptinput
root@SandboxHost-637453877092393449:/mnt/azscripts/azscriptinput# ls
hello.ps1
root@SandboxHost-637453877092393449:/mnt/azscripts/azscriptinput# pwsh./hello.ps1 "John Dole"
Hello John Dole
root@SandboxHost-637453877092393449:/mnt/azscripts/azscriptinput#
```

Use an Azure CLI container instance

To author your scripts on your computer, create a storage account and mount the storage account to the container instance. Then, you can upload your script to the storage account and run the script on the container instance.

> **Note**
>
> The storage account that you create to test your script isn't the same storage account that the deployment script service uses to execute the script. The deployment script service creates a unique name as a file share on every execution.

Create an Azure CLI container instance

The following ARM template creates a container instance and a file share, and then mounts the file share to the container image:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "projectName": {
      "type": "string",
      "metadata": {
        "description": "Specify a project name that is used for generating resource names."
      }
    },
    "containerImage": {
      "type": "string",
      "defaultValue": "mcr.microsoft.com/azure-cli:2.9.1",
      "metadata": {
        "description": "Specify the container image."
      }
    },
    "mountPath": {
      "type": "string",
      "defaultValue": "/mnt/azscripts/azscriptinput",
      "metadata": {
        "description": "Specify the mount path."
      }
    }
  },
  "variables": {
    "storageAccountName": "[tolower(concat(parameters('projectName'), 'store'))]",
    "fileShareName": "[concat(parameters('projectName'), 'share')]",
    "containerGroupName": "[concat(parameters('projectName'), 'cg')]",
    "containerName": "[concat(parameters('projectName'), 'container')]"
  },
  "resources": [
    {
```

```
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2019-06-01",
      "name": "[variables('storageAccountName')]",
      "location": "[resourceGroup().location]",
      "sku": {
        "name": "Standard_LRS",
        "tier": "Standard"
      },
      "kind": "StorageV2",
      "properties": {
        "accessTier": "Hot"
      }
    },
    {
      "type": "Microsoft.Storage/storageAccounts/fileServices/shares",
      "apiVersion": "2019-06-01",
      "name": "[concat(variables('storageAccountName'), '/default/',
variables('fileShareName'))]",
      "dependsOn": [
        "[resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))]"
      ]
    },
    {
      "type": "Microsoft.ContainerInstance/containerGroups",
      "apiVersion": "2019-12-01",
      "name": "[variables('containerGroupName')]",
      "location": "[resourceGroup().location]",
      "dependsOn": [
        "[resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName'))]"
      ],
      "properties": {
        "containers": [
          {
            "name": "[variables('containerName')]",
            "properties": {
              "image": "[parameters('containerImage')]",
              "resources": {
                "requests": {
                  "cpu": 1,
                  "memoryInGb": 1.5
                }
              },
              "ports": [
                {
                  "protocol": "TCP",
                  "port": 80
                }
              ],
              "volumeMounts": [
                {
                  "name": "filesharevolume",
                  "mountPath": "[parameters('mountPath')]"
                }
```

```
                ],
                "command": [
                  "/bin/bash",
                  "-c",
                  "echo hello; sleep 1800"
                ]
              }
            }
          ],
          "osType": "Linux",
          "volumes": [
            {
              "name": "filesharevolume",
              "azureFile": {
                "readOnly": false,
                "shareName": "[variables('fileShareName')]",
                "storageAccountName": "[variables('storageAccountName')]",
                "storageAccountKey": "
[listKeys(resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName')), '2019-06-01').keys[0].value]"
              }
            }
          ]
        }
      }
    ]
}
```

The default value for the mount path is `/mnt/azscripts/azscriptinput`. This is the path in the container instance where it's mounted to the file share.

The default container image specified in the template is **mcr.microsoft.com/azure-cli:2.9.1**. See a list of supported Azure CLI versions.

> **Important**
>
> The deployment script uses the available CLI images from Microsoft Container Registry (MCR). It takes about one month to certify a CLI image for a deployment script. Don't use the CLI versions that were released within 30 days. To find the release dates for the images, see **Azure CLI release notes**. If you use an unsupported version, the error message lists the supported versions.

The template suspends the container instance after 1,800 seconds. You have 30 minutes before the container instance goes into a terminal state and the session ends.

To deploy the template:

Azure PowerShell

```
$projectName = Read-Host -Prompt "Enter a project name that is used to
generate resource names"
$location = Read-Host -Prompt "Enter the location (i.e. centralus)"
$templateFile = Read-Host -Prompt "Enter the template file path and file
name"
$resourceGroupName = "${projectName}rg"

New-AzResourceGroup -Location $location -name $resourceGroupName
New-AzResourceGroupDeployment -resourceGroupName $resourceGroupName -
TemplateFile $templatefile -projectName $projectName
```

Upload the deployment script

Upload your deployment script to the storage account. The following is a PowerShell example:

Azure PowerShell

```
$projectName = Read-Host -Prompt "Enter the same project name that you used
earlier"
$fileName = Read-Host -Prompt "Enter the deployment script file name with
the path"

$resourceGroupName = "${projectName}rg"
$storageAccountName = "${projectName}store"
$fileShareName = "${projectName}share"

$context = (Get-AzStorageAccount -ResourceGroupName $resourceGroupName -Name
$storageAccountName).Context
Set-AzStorageFileContent -Context $context -ShareName $fileShareName -Source
$fileName -Force
```

You also can upload the file by using the Azure portal or the Azure CLI.

Test the deployment script

1. In the Azure portal, open the resource group where you deployed the container instance and the storage account.

2. Open the container group. The default container group name is the project name appended with *cg*. The container instance is shown in the **Running** state.

3. In the resource menu, select **Containers**. The container instance name is the project name appended with *container*.

4. Select **Connect**, and then select **Connect**. If you can't connect to the container instance, restart the container group and try again.

5. In the console pane, run the following commands:

Console

```
cd /mnt/azscripts/azscriptinput
ls
./hello.sh John Dole
```

The output is **Hello John Dole**.

```
bash-5.0# cd /mnt/azscripts/azscriptinput
bash-5.0# ls
hello.sh
bash-5.0# ./hello.sh John Dole
Hello John Dole
bash-5.0#
```

Use Docker

You can use a pre-configured Docker container image as your deployment script development environment. To install Docker, see Get Docker. You also need to configure file sharing to mount the directory, which contains the deployment scripts into Docker container.

1. Pull the deployment script container image to the local computer:

command

```
docker pull mcr.microsoft.com/azuredeploymentscripts-powershell:az4.3
```

The example uses version PowerShell 4.3.0.

To pull a CLI image from an MCR:

```
command

docker pull mcr.microsoft.com/azure-cli:2.0.80
```

This example uses version CLI 2.0.80. Deployment script uses the default CLI containers images found here.

2. Run the Docker image locally.

```
command

docker run -v <host drive letter>:/<host directory name>:/data -it
mcr.microsoft.com/azuredeploymentscripts-powershell:az4.3
```

Replace **<host driver letter>** and **<host directory name>** with an existing folder on the shared drive. It maps the folder to the */data* folder in the container. For example, to map *D:\docker*:

```
command

docker run -v d:/docker:/data -it
mcr.microsoft.com/azuredeploymentscripts-powershell:az4.3
```

-it means keeping the container image alive.

A CLI example:

```
command

docker run -v d:/docker:/data -it mcr.microsoft.com/azure-cli:2.0.80
```

3. The following screenshot shows how to run a PowerShell script, given that you have a *helloworld.ps1* file in the shared drive.

```
PS D:\> docker run -v d:/Docker:/data -it mcr.microsoft.com/azuredeploymentscripts-powershell:az2.7
PowerShell 6.2.3
Copyright (c) Microsoft Corporation. All rights reserved.

https://aka.ms/pscore6-docs
Type 'help' to get help.

PS /> ls
bin  boot  data  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
PS /> cd data
PS /data> ls
helloWorld.ps1
PS /data> pwsh -command "./helloWorld.ps1"
```

After the script is tested successfully, you can use it as a deployment script in your templates.

Next steps

In this article, you learned how to use deployment scripts. To walk through a deployment script tutorial:

Tutorial: Use deployment scripts in ARM templates

Using linked and nested templates when deploying Azure resources

Article • 04/26/2023

To deploy complex solutions, you can break your Azure Resource Manager template (ARM template) into many related templates, and then deploy them together through a main template. The related templates can be separate files or template syntax that is embedded within the main template. This article uses the term **linked template** to refer to a separate template file that is referenced via a link from the main template. It uses the term **nested template** to refer to embedded template syntax within the main template.

For small to medium solutions, a single template is easier to understand and maintain. You can see all the resources and values in a single file. For advanced scenarios, linked templates enable you to break down the solution into targeted components. You can easily reuse these templates for other scenarios.

For a tutorial, see Tutorial: Deploy a linked template.

> **Note**
>
> For linked or nested templates, you can only set the deployment mode to **Incremental**. However, the main template can be deployed in complete mode. If you deploy the main template in the complete mode, and the linked or nested template targets the same resource group, the resources deployed in the linked or nested template are included in the evaluation for complete mode deployment. The combined collection of resources deployed in the main template and linked or nested templates is compared against the existing resources in the resource group. Any resources not included in this combined collection are deleted.
>
> If the linked or nested template targets a different resource group, that deployment uses incremental mode. For more information, see **Deployment Scope**.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **modules**.

Nested template

To nest a template, add a deployments resource to your main template. In the `template` property, specify the template syntax.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {},
  "variables": {},
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "nestedTemplate1",
      "properties": {
        "mode": "Incremental",
        "template": {
          <nested-template-syntax>
        }
      }
    }
  ],
  "outputs": {
  }
}
```

The following example deploys a storage account through a nested template.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageAccountName": {
      "type": "string"
    }
  },
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "nestedTemplate1",
      "properties": {
        "mode": "Incremental",
        "template": {
```

```
          "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
          "contentVersion": "1.0.0.0",
          "resources": [
            {
              "type": "Microsoft.Storage/storageAccounts",
              "apiVersion": "2021-04-01",
              "name": "[parameters('storageAccountName')]",
              "location": "West US",
              "sku": {
                "name": "Standard_LRS"
              },
              "kind": "StorageV2"
            }
          ]
        }
      }
    }
  ],
  "outputs": {
  }
}
```

Expression evaluation scope in nested templates

When using a nested template, you can specify whether template expressions are evaluated within the scope of the parent template or the nested template. The scope determines how parameters, variables, and functions like resourceGroup and subscription are resolved.

You set the scope through the `expressionEvaluationOptions` property. By default, the `expressionEvaluationOptions` property is set to `outer`, which means it uses the parent template scope. Set the value to `inner` to cause expressions to be evaluated within the scope of the nested template.

JSON

```
{
  "type": "Microsoft.Resources/deployments",
  "apiVersion": "2021-04-01",
  "name": "nestedTemplate1",
  "properties": {
    "expressionEvaluationOptions": {
      "scope": "inner"
    },
  ...
```

> **Note**
>
> When scope is set to `outer`, you can't use the `reference` function in the outputs section of a nested template for a resource you have deployed in the nested template. To return the values for a deployed resource in a nested template, either use `inner` scope or convert your nested template to a linked template.

The following template demonstrates how template expressions are resolved according to the scope. It contains a variable named `exampleVar` that is defined in both the parent template and the nested template. It returns the value of the variable.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
  },
  "variables": {
    "exampleVar": "from parent template"
  },
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "nestedTemplate1",
      "properties": {
        "expressionEvaluationOptions": {
          "scope": "inner"
        },
        "mode": "Incremental",
        "template": {
          "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
          "contentVersion": "1.0.0.0",
          "variables": {
            "exampleVar": "from nested template"
          },
          "resources": [
          ],
          "outputs": {
            "testVar": {
              "type": "string",
              "value": "[variables('exampleVar')]"
            }
          }
        }
      }
    }
```

```
  ],
  "outputs": {
    "messageFromLinkedTemplate": {
      "type": "string",
      "value": "[reference('nestedTemplate1').outputs.testVar.value]"
    }
  }
}
```

The value of `exampleVar` changes depending on the value of the `scope` property in `expressionEvaluationOptions`. The following table shows the results for both scopes.

Evaluation scope	Output
inner	from nested template
outer (or default)	from parent template

The following example deploys a SQL server and retrieves a key vault secret to use for the password. The scope is set to `inner` because it dynamically creates the key vault ID (see `adminPassword.reference.keyVault` in the outer templates `parameters`) and passes it as a parameter to the nested template.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "The location where the resources will be deployed."
      }
    },
    "vaultName": {
      "type": "string",
      "metadata": {
        "description": "The name of the keyvault that contains the secret."
      }
    },
    "secretName": {
      "type": "string",
      "metadata": {
        "description": "The name of the secret."
      }
    },
    "vaultResourceGroupName": {
```

```
      "type": "string",
      "metadata": {
        "description": "The name of the resource group that contains the
keyvault."
      }
    },
    "vaultSubscription": {
      "type": "string",
      "defaultValue": "[subscription().subscriptionId]",
      "metadata": {
        "description": "The name of the subscription that contains the
keyvault."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "dynamicSecret",
      "properties": {
        "mode": "Incremental",
        "expressionEvaluationOptions": {
          "scope": "inner"
        },
        "parameters": {
          "location": {
            "value": "[parameters('location')]"
          },
          "adminLogin": {
            "value": "ghuser"
          },
          "adminPassword": {
            "reference": {
              "keyVault": {
                "id": "[resourceId(parameters('vaultSubscription'),
parameters('vaultResourceGroupName'), 'Microsoft.KeyVault/vaults',
parameters('vaultName'))]"
              },
              "secretName": "[parameters('secretName')]"
            }
          }
        },
        "template": {
          "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
          "contentVersion": "1.0.0.0",
          "parameters": {
            "adminLogin": {
              "type": "string"
            },
            "adminPassword": {
              "type": "securestring"
            },
            "location": {
```

```
              "type": "string"
            }
          },
          "variables": {
            "sqlServerName": "[concat('sql-',
uniqueString(resourceGroup().id, 'sql'))]"
          },
          "resources": [
            {
              "type": "Microsoft.Sql/servers",
              "apiVersion": "2021-02-01-preview",
              "name": "[variables('sqlServerName')]",
              "location": "[parameters('location')]",
              "properties": {
                "administratorLogin": "[parameters('adminLogin')]",
                "administratorLoginPassword": "
[parameters('adminPassword')]"
              }
            }
          ],
          "outputs": {
            "sqlFQDN": {
              "type": "string",
              "value": "
[reference(variables('sqlServerName')).fullyQualifiedDomainName]"
            }
          }
        }
      }
    }
  ],
  "outputs": {
  }
}
```

Be careful when using secure parameter values in a nested template. If you set the scope to outer, the secure values are stored as plain text in the deployment history. A user viewing the template in the deployment history could see the secure values. Instead use the inner scope or add to the parent template the resources that need secure values.

The following excerpt shows which values are secure and which aren't secure.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "adminUsername": {
      "type": "string",
      "metadata": {
```

```
        "description": "Username for the Virtual Machine."
      }
    },
    "adminPasswordOrKey": {
      "type": "securestring",
      "metadata": {
        "description": "SSH Key or password for the Virtual Machine. SSH key
is recommended."
      }
    }
  },
  ...
  "resources": [
    {
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2021-04-01",
      "name": "mainTemplate",
      "properties": {
        ...
        "osProfile": {
          "computerName": "mainTemplate",
          "adminUsername": "[parameters('adminUsername')]",
          "adminPassword": "[parameters('adminPasswordOrKey')]" // Yes,
secure because resource is in parent template
        }
      }
    },
    {
      "name": "outer",
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "properties": {
        "expressionEvaluationOptions": {
          "scope": "outer"
        },
        "mode": "Incremental",
        "template": {
          "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
          "contentVersion": "1.0.0.0",
          "resources": [
            {
              "type": "Microsoft.Compute/virtualMachines",
              "apiVersion": "2021-04-01",
              "name": "outer",
              "properties": {
                ...
                "osProfile": {
                  "computerName": "outer",
                  "adminUsername": "[parameters('adminUsername')]",
                  "adminPassword": "[parameters('adminPasswordOrKey')]" //
No, not secure because resource is in nested template with outer scope
                }
              }
            }
```

```
          ]
        }
      }
    },
    {
      "name": "inner",
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "properties": {
        "expressionEvaluationOptions": {
          "scope": "inner"
        },
        "mode": "Incremental",
        "parameters": {
          "adminPasswordOrKey": {
              "value": "[parameters('adminPasswordOrKey')]"
          },
          "adminUsername": {
              "value": "[parameters('adminUsername')]"
          }
        },
        "template": {
          "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
          "contentVersion": "1.0.0.0",
          "parameters": {
            "adminUsername": {
              "type": "string",
              "metadata": {
                "description": "Username for the Virtual Machine."
              }
            },
            "adminPasswordOrKey": {
              "type": "securestring",
              "metadata": {
                "description": "SSH Key or password for the Virtual Machine.
SSH key is recommended."
              }
            }
          },
          "resources": [
            {
              "type": "Microsoft.Compute/virtualMachines",
              "apiVersion": "2021-04-01",
              "name": "inner",
              "properties": {
                ...
                "osProfile": {
                  "computerName": "inner",
                  "adminUsername": "[parameters('adminUsername')]",
                  "adminPassword": "[parameters('adminPasswordOrKey')]" //
Yes, secure because resource is in nested template and scope is inner
                }
              }
            }
```

```
          ]
        }
      }
    }
  ]
}
```

Linked template

To link a template, add a deployments resource to your main template. In the `templateLink` property, specify the URI of the template to include. The following example links to a template that is in a storage account.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {},
  "variables": {},
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "linkedTemplate",
      "properties": {
        "mode": "Incremental",
        "templateLink": {

"uri":"https://mystorageaccount.blob.core.windows.net/AzureTemplates/newStor
ageAccount.json",
          "contentVersion":"1.0.0.0"
        }
      }
    }
  ],
  "outputs": {
  }
}
```

When referencing a linked template, the value of `uri` can't be a local file or a file that is only available on your local network. Azure Resource Manager must be able to access the template. Provide a URI value that downloadable as HTTP or HTTPS.

You may reference templates using parameters that include HTTP or HTTPS. For example, a common pattern is to use the `_artifactsLocation` parameter. You can set the linked template with an expression like:

JSON

```
"uri": "[concat(parameters('_artifactsLocation'), '/shared/os-disk-parts-
md.json', parameters('_artifactsLocationSasToken'))]"
```

If you're linking to a template in GitHub, use the raw URL. The link has the format: `https://raw.githubusercontent.com/Azure/azure-docs-json-samples/master/get-started-with-templates/quickstart-template/azuredeploy.json`. To get the raw link, select **Raw**.

> **Note**
>
> To deploy a template or reference a linked template that is stored in a private GitHub repo, see a custom solution documented in **Creating a Custom and Secure Azure Portal Offering**. You can create an **Azure function** that pulls the GitHub token out of Azure Key Vault.

Parameters for linked template

You can provide the parameters for your linked template either in an external file or inline. When providing an external parameter file, use the `parametersLink` property:

JSON

```
"resources": [
  {
    "type": "Microsoft.Resources/deployments",
    "apiVersion": "2021-04-01",
    "name": "linkedTemplate",
    "properties": {
      "mode": "Incremental",
      "templateLink": {
        "uri":
"https://mystorageaccount.blob.core.windows.net/AzureTemplates/newStorageAcc
ount.json",
        "contentVersion": "1.0.0.0"
      },
      "parametersLink": {
        "uri":
"https://mystorageaccount.blob.core.windows.net/AzureTemplates/newStorageAcc
```

```
ount.parameters.json",
        "contentVersion": "1.0.0.0"
      }
    }
  }
]
```

To pass parameter values inline, use the `parameters` property.

JSON

```
"resources": [
  {
    "type": "Microsoft.Resources/deployments",
    "apiVersion": "2021-04-01",
    "name": "linkedTemplate",
    "properties": {
      "mode": "Incremental",
      "templateLink": {
        "uri":
"https://mystorageaccount.blob.core.windows.net/AzureTemplates/newStorageAcc
ount.json",
        "contentVersion": "1.0.0.0"
      },
      "parameters": {
        "storageAccountName": {
          "value": "[parameters('storageAccountName')]"
        }
      }
    }
  }
]
```

You can't use both inline parameters and a link to a parameter file. The deployment fails with an error when both `parametersLink` and `parameters` are specified.

Use relative path for linked templates

The `relativePath` property of `Microsoft.Resources/deployments` makes it easier to author linked templates. This property can be used to deploy a remote linked template at a location relative to the parent. This feature requires all template files to be staged and available at a remote URI, such as GitHub or Azure storage account. When the main template is called by using a URI from Azure PowerShell or Azure CLI, the child deployment URI is a combination of the parent and relativePath.

Note

When creating a templateSpec, any templates referenced by the `relativePath` property is packaged in the templateSpec resource by Azure PowerShell or Azure CLI. It do not require the files to be staged. For more information, see **Create a template spec with linked templates**.

Assume a folder structure like this:

The following template shows how *mainTemplate.json* deploys *nestedChild.json* illustrated in the preceding image.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {},
  "functions": [],
  "variables": {},
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "childLinked",
      "properties": {
        "mode": "Incremental",
        "templateLink": {
          "relativePath": "children/nestedChild.json"
        }
      }
    }
  ],
  "outputs": {}
}
```

In the following deployment, the URI of the linked template in the preceding template is https://raw.githubusercontent.com/Azure/azure-docs-json-samples/master/linked-template-relpath/children/nestedChild.json.

PowerShell

Azure PowerShell

```
New-AzResourceGroupDeployment `
  -Name linkedTemplateWithRelativePath `
  -ResourceGroupName "myResourceGroup" `
  -TemplateUri "https://raw.githubusercontent.com/Azure/azure-docs-json-samples/master/linked-template-relpath/mainTemplate.json"
```

To deploy linked templates with relative path stored in an Azure storage account, use the `QueryString`/`query-string` parameter to specify the SAS token to be used with the TemplateUri parameter. This parameter is only supported by Azure CLI version 2.18 or later and Azure PowerShell version 5.4 or later.

PowerShell

Azure PowerShell

```
New-AzResourceGroupDeployment `
  -Name linkedTemplateWithRelativePath `
  -ResourceGroupName "myResourceGroup" `
  -TemplateUri "https://stage20210126.blob.core.windows.net/template-staging/mainTemplate.json" `
  -QueryString $sasToken
```

Make sure there is no leading "?" in QueryString. The deployment adds one when assembling the URI for the deployments.

Template specs

Instead of maintaining your linked templates at an accessible endpoint, you can create a template spec that packages the main template and its linked templates into a single entity you can deploy. The template spec is a resource in your Azure subscription. It makes it easy to securely share the template with users in your organization. You use Azure role-based access control (Azure RBAC) to grant access to the template spec. This feature is currently in preview.

For more information, see:

- Tutorial: Create a template spec with linked templates.
- Tutorial: Deploy a template spec as a linked template.

Dependencies

As with other resource types, you can set dependencies between the nested/linked templates. If the resources in one nested/linked template must be deployed before resources in a second nested/linked template, set the second template dependent on the first.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {},
  "variables": {},
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "linkedTemplate1",
      "properties": {
        "mode": "Incremental",
        "templateLink": {
          "uri": "[uri(deployment().properties.templateLink.uri, 'firstresources.json')]",
          "contentVersion": "1.0.0.0"
        }
      }
    },
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "linkedTemplate2",
      "dependsOn": [
        "[resourceId('Microsoft.Resources/deployments', 'linkedTemplate1')]"
      ],
      "properties": {
        "mode": "Incremental",
        "templateLink": {
          "uri": "[uri(deployment().properties.templateLink.uri, 'secondresources.json')]",
          "contentVersion": "1.0.0.0"
        }
      }
    }
  ]
}
```

contentVersion

You don't have to provide the `contentVersion` property for the `templateLink` or `parametersLink` property. If you don't provide a `contentVersion`, the current version of the template is deployed. If you provide a value for content version, it must match the version in the linked template; otherwise, the deployment fails with an error.

Using variables to link templates

The previous examples showed hard-coded URL values for the template links. This approach might work for a simple template, but it doesn't work well for a large set of modular templates. Instead, you can create a static variable that stores a base URL for the main template and then dynamically create URLs for the linked templates from that base URL. The benefit of this approach is that you can easily move or fork the template because you need to change only the static variable in the main template. The main template passes the correct URIs throughout the decomposed template.

The following example shows how to use a base URL to create two URLs for linked templates (`sharedTemplateUrl` and `vmTemplateUrl`).

JSON

```
"variables": {
  "templateBaseUrl": "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/application-workloads/postgre/postgresql-on-ubuntu/",
  "sharedTemplateUrl": "[uri(variables('templateBaseUrl'), 'shared-resources.json')]",
  "vmTemplateUrl": "[uri(variables('templateBaseUrl'), 'database-2disk-resources.json')]"
}
```

You can also use deployment() to get the base URL for the current template, and use that to get the URL for other templates in the same location. This approach is useful if your template location changes or you want to avoid hard coding URLs in the template file. The `templateLink` property is only returned when linking to a remote template with a URL. If you're using a local template, that property isn't available.

JSON

```
"variables": {
  "sharedTemplateUrl": "[uri(deployment().properties.templateLink.uri, 'shared-resources.json')]"
}
```

Ultimately, you would use the variable in the `uri` property of a `templateLink` property.

JSON

```
"templateLink": {
 "uri": "[variables('sharedTemplateUrl')]",
 "contentVersion":"1.0.0.0"
}
```

Using copy

To create multiple instances of a resource with a nested template, add the `copy` element at the level of the `Microsoft.Resources/deployments` resource. Or, if the scope is `inner`, you can add the copy within the nested template.

The following example template shows how to use `copy` with a nested template.

JSON

```
"resources": [
  {
    "type": "Microsoft.Resources/deployments",
    "apiVersion": "2021-04-01",
    "name": "[concat('nestedTemplate', copyIndex())]",
    // yes, copy works here
    "copy": {
      "name": "storagecopy",
      "count": 2
    },
    "properties": {
      "mode": "Incremental",
      "expressionEvaluationOptions": {
        "scope": "inner"
      },
      "template": {
        "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
        "contentVersion": "1.0.0.0",
        "resources": [
          {
            "type": "Microsoft.Storage/storageAccounts",
            "apiVersion": "2021-04-01",
            "name": "[concat(variables('storageName'), copyIndex())]",
            "location": "West US",
            "sku": {
              "name": "Standard_LRS"
            },
            "kind": "StorageV2"
            // Copy works here when scope is inner
            // But, when scope is default or outer, you get an error
            // "copy": {
            //   "name": "storagecopy",
```

```
            //   "count": 2
            // }
          }
        ]
      }
    }
  }
]
```

Get values from linked template

To get an output value from a linked template, retrieve the property value with syntax like: `"[reference('deploymentName').outputs.propertyName.value]"`.

When getting an output property from a linked template, the property name must not include a dash.

The following examples demonstrate how to reference a linked template and retrieve an output value. The linked template returns a simple message. First, the linked template:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {},
  "variables": {},
  "resources": [],
  "outputs": {
    "greetingMessage": {
      "value": "Hello World",
      "type": "string"
    }
  }
}
```

The main template deploys the linked template and gets the returned value. Notice that it references the deployment resource by name, and it uses the name of the property returned by the linked template.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {},
```

```
  "variables": {},
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "linkedTemplate",
      "properties": {
        "mode": "incremental",
        "templateLink": {
          "uri": "[uri(deployment().properties.templateLink.uri,
'helloworld.json')]",
          "contentVersion": "1.0.0.0"
        }
      }
    }
  ],
  "outputs": {
    "messageFromLinkedTemplate": {
      "type": "string",
      "value": "[reference('linkedTemplate').outputs.greetingMessage.value]"
    }
  }
}
```

The following example shows a template that deploys a public IP address and returns the resource ID of the Azure resource for that public IP:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "publicIPAddresses_name": {
      "type": "string"
    }
  },
  "variables": {},
  "resources": [
    {
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2021-02-01",
      "name": "[parameters('publicIPAddresses_name')]",
      "location": "eastus",
      "properties": {
        "publicIPAddressVersion": "IPv4",
        "publicIPAllocationMethod": "Dynamic",
        "idleTimeoutInMinutes": 4
      },
      "dependsOn": []
    }
  ],
```

```
  "outputs": {
    "resourceID": {
      "type": "string",
      "value": "[resourceId('Microsoft.Network/publicIPAddresses',
parameters('publicIPAddresses_name'))]"
    }
  }
}
```

To use the public IP address from the preceding template when deploying a load balancer, link to the template and declare a dependency on the `Microsoft.Resources/deployments` resource. The public IP address on the load balancer is set to the output value from the linked template.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "loadBalancers_name": {
      "defaultValue": "mylb",
      "type": "string"
    },
    "publicIPAddresses_name": {
      "defaultValue": "myip",
      "type": "string"
    }
  },
  "variables": {},
  "resources": [
    {
      "type": "Microsoft.Network/loadBalancers",
      "apiVersion": "2021-02-01",
      "name": "[parameters('loadBalancers_name')]",
      "location": "eastus",
      "properties": {
        "frontendIPConfigurations": [
          {
            "name": "LoadBalancerFrontEnd",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "publicIPAddress": {
                "id": "
[reference('linkedTemplate').outputs.resourceID.value]"
              }
            }
          }
        ],
        "backendAddressPools": [],
        "loadBalancingRules": [],
```

```
          "probes": [],
          "inboundNatRules": [],
          "outboundNatRules": [],
          "inboundNatPools": []
        },
        "dependsOn": [
          "[resourceId('Microsoft.Resources/deployments', 'linkedTemplate')]"
        ]
      },
      {
        "type": "Microsoft.Resources/deployments",
        "apiVersion": "2021-04-01",
        "name": "linkedTemplate",
        "properties": {
          "mode": "Incremental",
          "templateLink": {
            "uri": "[uri(deployment().properties.templateLink.uri, 'public-
ip.json')]",
            "contentVersion": "1.0.0.0"
          },
          "parameters": {
            "publicIPAddresses_name": { "value": "
[parameters('publicIPAddresses_name')]" }
          }
        }
      }
    ]
}
```

Deployment history

Resource Manager processes each template as a separate deployment in the deployment history. A main template with three linked or nested templates appears in the deployment history as:

Search for deployments by name...

DEPLOYMENT NAME	STATUS	TIMESTAMP	DURATION
parentTemplate	Succeeded	11/28/2017 2:01:23 PM	19 seconds
linkedTemplate2	Succeeded	11/28/2017 2:01:18 PM	7 seconds
linkedTemplate1	Succeeded	11/28/2017 2:01:18 PM	7 seconds
linkedTemplate0	Succeeded	11/28/2017 2:01:18 PM	6 seconds

You can use these separate entries in the history to retrieve output values after the deployment. The following template creates a public IP address and outputs the IP

address:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "publicIPAddresses_name": {
      "type": "string"
    }
  },
  "variables": {},
  "resources": [
    {
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2021-02-01",
      "name": "[parameters('publicIPAddresses_name')]",
      "location": "southcentralus",
      "properties": {
        "publicIPAddressVersion": "IPv4",
        "publicIPAllocationMethod": "Static",
        "idleTimeoutInMinutes": 4,
        "dnsSettings": {
          "domainNameLabel": "[concat(parameters('publicIPAddresses_name'),
uniqueString(resourceGroup().id))]"
        }
      },
      "dependsOn": []
    }
  ],
  "outputs": {
    "returnedIPAddress": {
      "type": "string",
      "value": "[reference(parameters('publicIPAddresses_name')).ipAddress]"
    }
  }
}
```

The following template links to the preceding template. It creates three public IP addresses.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
  },
  "variables": {},
```

```
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "[concat('linkedTemplate', copyIndex())]",
      "copy": {
        "count": 3,
        "name": "ip-loop"
      },
      "properties": {
        "mode": "Incremental",
        "templateLink": {
        "uri": "[uri(deployment().properties.templateLink.uri, 'static-
public-ip.json')]",
        "contentVersion": "1.0.0.0"
        },
        "parameters":{
          "publicIPAddresses_name":{"value": "[concat('myip-',
copyIndex())]"}
        }
      }
    }
  ]
}
```

After the deployment, you can retrieve the output values with the following PowerShell script:

Azure PowerShell

```
$loopCount = 3
for ($i = 0; $i -lt $loopCount; $i++)
{
  $name = 'linkedTemplate' + $i;
  $deployment = Get-AzResourceGroupDeployment -ResourceGroupName
examplegroup -Name $name
  Write-Output "deployment $($deployment.DeploymentName) returned
$($deployment.Outputs.returnedIPAddress.value)"
}
```

Or, Azure CLI script in a Bash shell:

Azure CLI

```
#!/bin/bash

for i in 0 1 2;
do
  name="linkedTemplate$i";
  deployment=$(az deployment group show -g examplegroup -n $name);
  ip=$(echo $deployment | jq .properties.outputs.returnedIPAddress.value);
```

```
  echo "deployment $name returned $ip";
done
```

Securing an external template

Although the linked template must be externally available, it doesn't need to be generally available to the public. You can add your template to a private storage account that is accessible to only the storage account owner. Then, you create a shared access signature (SAS) token to enable access during deployment. You add that SAS token to the URI for the linked template. Even though the token is passed in as a secure string, the URI of the linked template, including the SAS token, is logged in the deployment operations. To limit exposure, set an expiration for the token.

The parameter file can also be limited to access through a SAS token.

Currently, you can't link to a template in a storage account that is behind an Azure Storage firewall.

> **Important**
>
> Instead of securing your linked template with a SAS token, consider creating a **template spec**. The template spec securely stores the main template and its linked templates as a resource in your Azure subscription. You use Azure RBAC to grant access to users who need to deploy the template.

The following example shows how to pass a SAS token when linking to a template:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "containerSasToken": { "type": "securestring" }
  },
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "linkedTemplate",
      "properties": {
        "mode": "Incremental",
        "templateLink": {
          "uri": "[concat(uri(deployment().properties.templateLink.uri,
```

```
'helloworld.json'), parameters('containerSasToken'))]",
          "contentVersion": "1.0.0.0"
        }
      }
    }
  ],
  "outputs": {
  }
}
```

In PowerShell, you get a token for the container and deploy the templates with the following commands. Notice that the `containerSasToken` parameter is defined in the template. It isn't a parameter in the `New-AzResourceGroupDeployment` command.

Azure PowerShell

```
Set-AzCurrentStorageAccount -ResourceGroupName ManageGroup -Name
storagecontosotemplates
$token = New-AzStorageContainerSASToken -Name templates -Permission r -
ExpiryTime (Get-Date).AddMinutes(30.0)
$url = (Get-AzStorageBlob -Container templates -Blob
parent.json).ICloudBlob.uri.AbsoluteUri
New-AzResourceGroupDeployment -ResourceGroupName ExampleGroup -TemplateUri
($url + $token) -containerSasToken $token
```

For Azure CLI in a Bash shell, you get a token for the container and deploy the templates with the following code:

Azure CLI

```
#!/bin/bash

expiretime=$(date -u -d '30 minutes' +%Y-%m-%dT%H:%MZ)
connection=$(az storage account show-connection-string \
  --resource-group ManageGroup \
  --name storagecontosotemplates \
  --query connectionString)
token=$(az storage container generate-sas \
  --name templates \
  --expiry $expiretime \
  --permissions r \
  --output tsv \
  --connection-string $connection)
url=$(az storage blob url \
  --container-name templates \
  --name parent.json \
  --output tsv \
  --connection-string $connection)
parameter='{"containerSasToken":{"value":"?'$token'"}}'
```

```
az deployment group create --resource-group ExampleGroup --template-uri
$url?$token --parameters $parameter
```

Example templates

The following examples show common uses of linked templates.

Main template	Linked template	Description
Hello World	linked template	Returns string from linked template.
Load Balancer with public IP address	linked template	Returns public IP address from linked template and sets that value in load balancer.
Multiple IP addresses	linked template	Creates several public IP addresses in linked template.

Next steps

- To go through a tutorial, see Tutorial: Deploy a linked template.
- To learn about the defining the deployment order for your resources, see Define the order for deploying resources in ARM templates.
- To learn how to define one resource but create many instances of it, see Resource iteration in ARM templates.
- For steps on setting up a template in a storage account and generating a SAS token, see Deploy resources with ARM templates and Azure PowerShell or Deploy resources with ARM templates and Azure CLI.

Resource iteration in ARM templates

Article • 04/09/2023

This article shows you how to create more than one instance of a resource in your Azure Resource Manager template (ARM template). By adding copy loop to the resources section of your template, you can dynamically set the number of resources to deploy. You also avoid having to repeat template syntax.

You can also use copy loop with properties, variables, and outputs.

If you need to specify whether a resource is deployed at all, see condition element.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **loops**.

Syntax

Add the `copy` element to the resources section of your template to deploy multiple instances of the resource. The `copy` element has the following general format:

JSON

```
"copy": {
  "name": "<name-of-loop>",
  "count": <number-of-iterations>,
  "mode": "serial" <or> "parallel",
  "batchSize": <number-to-deploy-serially>
}
```

The `name` property is any value that identifies the loop. The `count` property specifies the number of iterations you want for the resource type.

Use the `mode` and `batchSize` properties to specify if the resources are deployed in parallel or in sequence. These properties are described in Serial or Parallel.

Copy limits

The count can't exceed 800.

The count can't be a negative number. It can be zero if you deploy the template with a recent version of Azure CLI, PowerShell, or REST API. Specifically, you must use:

- Azure PowerShell **2.6** or later
- Azure CLI **2.0.74** or later
- REST API version **2019-05-10** or later
- Linked deployments must use API version **2019-05-10** or later for the deployment resource type

Earlier versions of PowerShell, CLI, and the REST API don't support zero for count.

Be careful using complete mode deployment with copy loop. If you redeploy with complete mode to a resource group, any resources that aren't specified in the template after resolving the copy loop are deleted.

Resource iteration

The following example creates the number of storage accounts specified in the `storageCount` parameter.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageCount": {
      "type": "int",
      "defaultValue": 3
    }
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2019-04-01",
      "name": "[concat(copyIndex(),'storage', uniqueString(resourceGroup().id))]",
      "location": "[resourceGroup().location]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "Storage",
      "properties": {},
      "copy": {
        "name": "storagecopy",
        "count": "[parameters('storageCount')]"
      }
    }
```

```
  ]
}
```

Notice that the name of each resource includes the `copyIndex()` function, which returns the current iteration in the loop. `copyIndex()` is zero-based. So, the following example:

JSON

```
"name": "[concat('storage', copyIndex())]",
```

Creates these names:

- storage0
- storage1
- storage2

To offset the index value, you can pass a value in the `copyIndex()` function. The number of iterations is still specified in the copy element, but the value of `copyIndex` is offset by the specified value. So, the following example:

JSON

```
"name": "[concat('storage', copyIndex(1))]",
```

Creates these names:

- storage1
- storage2
- storage3

The copy operation is helpful when working with arrays because you can iterate through each element in the array. Use the `length` function on the array to specify the count for iterations, and `copyIndex` to retrieve the current index in the array.

The following example creates one storage account for each name provided in the parameter.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
      "storageNames": {
```

```
            "type": "array",
            "defaultValue": [
              "contoso",
              "fabrikam",
              "coho"
            ]
        }
    },
    "resources": [
      {
        "type": "Microsoft.Storage/storageAccounts",
        "apiVersion": "2019-04-01",
        "name": "[concat(parameters('storageNames')[copyIndex()],
uniqueString(resourceGroup().id))]",
        "location": "[resourceGroup().location]",
        "sku": {
          "name": "Standard_LRS"
        },
        "kind": "Storage",
        "properties": {},
        "copy": {
          "name": "storagecopy",
          "count": "[length(parameters('storageNames'))]"
        }
      }
    ],
    "outputs": {}
}
```

If you want to return values from the deployed resources, you can use copy in the outputs section.

Serial or Parallel

By default, Resource Manager creates the resources in parallel. It applies no limit to the number of resources deployed in parallel, other than the total limit of 800 resources in the template. The order in which they're created isn't guaranteed.

However, you may want to specify that the resources are deployed in sequence. For example, when updating a production environment, you may want to stagger the updates so only a certain number are updated at any one time.

To serially deploy more than one instance of a resource, set `mode` to **serial** and `batchSize` to the number of instances to deploy at a time. With serial mode, Resource Manager creates a dependency on earlier instances in the loop, so it doesn't start one batch until the previous batch completes.

The value for `batchSize` can't exceed the value for `count` in the copy element.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2019-04-01",
      "name": "[concat(copyIndex(),'storage', uniqueString(resourceGroup().id))]",
      "location": "[resourceGroup().location]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "Storage",
      "copy": {
        "name": "storagecopy",
        "count": 4,
        "mode": "serial",
        "batchSize": 2
      },
      "properties": {}
    }
  ],
  "outputs": {}
}
```

The `mode` property also accepts **parallel**, which is the default value.

Iteration for a child resource

You can't use a copy loop for a child resource. To create more than one instance of a resource that you typically define as nested within another resource, you must instead create that resource as a top-level resource. You define the relationship with the parent resource through the type and name properties.

For example, suppose you typically define a dataset as a child resource within a data factory.

JSON

```
{
  "resources": [
    {
      "type": "Microsoft.DataFactory/factories",
      "name": "exampleDataFactory",
      ...
```

```
      "resources": [
        {
          "type": "datasets",
          "name": "exampleDataSet",
          "dependsOn": [
            "exampleDataFactory"
          ],
          ...
        }
      ]
      ...
    }
  ]
}
```

To create more than one data set, move it outside of the data factory. The dataset must be at the same level as the data factory, but it's still a child resource of the data factory. You preserve the relationship between data set and data factory through the type and name properties. Since type can no longer be inferred from its position in the template, you must provide the fully qualified type in the format: `{resource-provider-namespace}/{parent-resource-type}/{child-resource-type}`.

To establish a parent/child relationship with an instance of the data factory, provide a name for the data set that includes the parent resource name. Use the format: `{parent-resource-name}/{child-resource-name}`.

The following example shows the implementation.

JSON

```
"resources": [
{
  "type": "Microsoft.DataFactory/factories",
  "name": "exampleDataFactory",
  ...
},
{
  "type": "Microsoft.DataFactory/factories/datasets",
  "name": "[concat('exampleDataFactory', '/', 'exampleDataSet',
copyIndex())]",
  "dependsOn": [
    "exampleDataFactory"
  ],
  "copy": {
    "name": "datasetcopy",
    "count": "3"
  },
  ...
}]
```

Example templates

The following examples show common scenarios for creating more than one instance of a resource or property.

Template	Description
Copy storage	Deploys more than one storage account with an index number in the name.
Serial copy storage	Deploys several storage accounts one at time. The name includes the index number.
Copy storage with array	Deploys several storage accounts. The name includes a value from an array.
Copy resource group	Deploys multiple resource groups.

Next steps

- To set dependencies on resources that are created in a copy loop, see Define the order for deploying resources in ARM templates.
- To go through a tutorial, see Tutorial: Create multiple resource instances with ARM templates.
- For a Learn module that covers resource copy, see Manage complex cloud deployments by using advanced ARM template features.
- For other uses of the copy loop, see:
  - Property iteration in ARM templates
  - Variable iteration in ARM templates
  - Output iteration in ARM templates
- For information about using copy with nested templates, see Using copy.

Property iteration in ARM templates

Article • 04/09/2023

This article shows you how to create more than one instance of a property in your Azure Resource Manager template (ARM template). By adding copy loop to the properties section of a resource in your template, you can dynamically set the number of items for a property during deployment. You also avoid having to repeat template syntax.

You can only use copy loop with top-level resources, even when applying copy loop to a property. To learn about changing a child resource to a top-level resource, see Iteration for a child resource.

You can also use copy loop with resources, variables, and outputs.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **loops**.

Syntax

Add the `copy` element to the resources section of your template to set the number of items for a property. The copy element has the following general format:

JSON

```
"copy": [
  {
    "name": "<name-of-property>",
    "count": <number-of-iterations>,
    "input": <values-for-the-property>
  }
]
```

For `name`, provide the name of the resource property that you want to create.

The `count` property specifies the number of iterations you want for the property.

The `input` property specifies the properties that you want to repeat. You create an array of elements constructed from the value in the `input` property.

Copy limits

The count can't exceed 800.

The count can't be a negative number. It can be zero if you deploy the template with a recent version of Azure CLI, PowerShell, or REST API. Specifically, you must use:

- Azure PowerShell **2.6** or later
- Azure CLI **2.0.74** or later
- REST API version **2019-05-10** or later
- Linked deployments must use API version **2019-05-10** or later for the deployment resource type

Earlier versions of PowerShell, CLI, and the REST API don't support zero for count.

Property iteration

The following example shows how to apply copy loop to the `dataDisks` property on a virtual machine:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "numberOfDataDisks": {
      "type": "int",
      "minValue": 0,
      "maxValue": 16,
      "defaultValue": 3,
      "metadata": {
        "description": "The number of dataDisks to create."
      }
    },
    ...
  },
  "resources": [
    {
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2020-06-01",
      ...
      "properties": {
        "storageProfile": {
          ...
          "copy": [
            {
              "name": "dataDisks",
```

```
              "count": "[parameters('numberOfDataDisks')]",
              "input": {
                "lun": "[copyIndex('dataDisks')]",
                "createOption": "Empty",
                "diskSizeGB": 1023
              }
            }
          ]
        }
        ...
      }
    }
  ]
}
```

Notice that when using copyIndex inside a property iteration, you must provide the name of the iteration. Property iteration also supports an offset argument. The offset must come after the name of the iteration, such as `copyIndex('dataDisks', 1)`.

The deployed template becomes:

JSON

```
{
  "name": "examplevm",
  "type": "Microsoft.Compute/virtualMachines",
  "apiVersion": "2020-06-01",
  "properties": {
    "storageProfile": {
      "dataDisks": [
        {
          "lun": 0,
          "createOption": "Empty",
          "diskSizeGB": 1023
        },
        {
          "lun": 1,
          "createOption": "Empty",
          "diskSizeGB": 1023
        },
        {
          "lun": 2,
          "createOption": "Empty",
          "diskSizeGB": 1023
        }
      ],
      ...
```

The copy operation is helpful when working with arrays because you can iterate through each element in the array. Use the length function on the array to specify the count for

iterations, and `copyIndex` to retrieve the current index in the array.

The following example template creates a failover group for databases that are passed in as an array.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "primaryServerName": {
      "type": "string"
    },
    "secondaryServerName": {
      "type": "string"
    },
    "databaseNames": {
      "type": "array",
      "defaultValue": [
        "mydb1",
        "mydb2",
        "mydb3"
      ]
    }
  },
  "variables": {
    "failoverName": "[concat(parameters('primaryServerName'),'/',parameters('primaryServerName'),'failovergroups')]"
  },
  "resources": [
    {
      "type": "Microsoft.Sql/servers/failoverGroups",
      "apiVersion": "2015-05-01-preview",
      "name": "[variables('failoverName')]",
      "properties": {
        "readWriteEndpoint": {
          "failoverPolicy": "Automatic",
          "failoverWithDataLossGracePeriodMinutes": 60
        },
        "readOnlyEndpoint": {
          "failoverPolicy": "Disabled"
        },
        "partnerServers": [
          {
            "id": "[resourceId('Microsoft.Sql/servers',parameters('secondaryServerName'))]"
          }
        ],
        "copy": [
          {
            "name": "databases",
```

```
            "count": "[length(parameters('databaseNames'))]",
            "input": "[resourceId('Microsoft.Sql/servers/databases',
parameters('primaryServerName'), parameters('databaseNames')
[copyIndex('databases')])]"
          }
        ]
      }
    }
  ],
  "outputs": {
  }
}
```

The `copy` element is an array so you can specify more than one property for the resource.

JSON

```
{
  "type": "Microsoft.Network/loadBalancers",
  "apiVersion": "2017-10-01",
  "name": "exampleLB",
  "properties": {
    "copy": [
      {
        "name": "loadBalancingRules",
        "count": "[length(parameters('loadBalancingRules'))]",
        "input": {
          ...
        }
      },
      {
        "name": "probes",
        "count": "[length(parameters('loadBalancingRules'))]",
        "input": {
          ...
        }
      }
    ]
  }
}
```

You can use resource and property iterations together. Reference the property iteration by name.

JSON

```
{
  "type": "Microsoft.Network/virtualNetworks",
  "apiVersion": "2018-04-01",
  "name": "[concat(parameters('vnetname'), copyIndex())]",
```

```
  "copy":{
    "count": 2,
    "name": "vnetloop"
  },
  "location": "[resourceGroup().location]",
  "properties": {
    "addressSpace": {
      "addressPrefixes": [
        "[parameters('addressPrefix')]"
      ]
    },
    "copy": [
      {
        "name": "subnets",
        "count": 2,
        "input": {
          "name": "[concat('subnet-', copyIndex('subnets'))]",
          "properties": {
            "addressPrefix": "[variables('subnetAddressPrefix')
[copyIndex('subnets')]]"
          }
        }
      }
    ]
  }
}
```

Example templates

The following example shows a common scenario for creating more than one value for a property.

Template	Description
VM deployment with a variable number of data disks	Deploys several data disks with a virtual machine.

Next steps

- To go through a tutorial, see Tutorial: Create multiple resource instances with ARM templates.
- For other uses of the copy loop, see:
  - Resource iteration in ARM templates
  - Variable iteration in ARM templates
  - Output iteration in ARM templates

- If you want to learn about the sections of a template, see Understand the structure and syntax of ARM templates.
- To learn how to deploy your template, see Deploy resources with ARM templates and Azure PowerShell.

Variable iteration in ARM templates

Article • 04/09/2023

This article shows you how to create more than one value for a variable in your Azure Resource Manager template (ARM template). By adding the `copy` element to the variables section of your template, you can dynamically set the number of items for a variable during deployment. You also avoid having to repeat template syntax.

You can also use copy with resources, properties in a resource, and outputs.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **loops**.

Syntax

The copy element has the following general format:

JSON

```
"copy": [
  {
    "name": "<name-of-loop>",
    "count": <number-of-iterations>,
    "input": <values-for-the-variable>
  }
]
```

The `name` property is any value that identifies the loop. The `count` property specifies the number of iterations you want for the variable.

The `input` property specifies the properties that you want to repeat. You create an array of elements constructed from the value in the `input` property. It can be a single property (like a string), or an object with several properties.

Copy limits

The count can't exceed 800.

The count can't be a negative number. It can be zero if you deploy the template with a recent version of Azure CLI, PowerShell, or REST API. Specifically, you must use:

- Azure PowerShell **2.6** or later
- Azure CLI **2.0.74** or later
- REST API version **2019-05-10** or later
- Linked deployments must use API version **2019-05-10** or later for the deployment resource type

Earlier versions of PowerShell, CLI, and the REST API don't support zero for count.

Variable iteration

The following example shows how to create an array of string values:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "itemCount": {
      "type": "int",
      "defaultValue": 5
    }
  },
  "variables": {
    "copy": [
      {
        "name": "stringArray",
        "count": "[parameters('itemCount')]",
        "input": "[concat('item', copyIndex('stringArray', 1))]"
      }
    ]
  },
  "resources": [],
  "outputs": {
    "arrayResult": {
      "type": "array",
      "value": "[variables('stringArray')]"
    }
  }
}
```

The preceding template returns an array with the following values:

JSON

```
[
  "item1",
  "item2",
  "item3",
  "item4",
  "item5"
]
```

The next example shows how to create an array of objects with three properties - `name`, `diskSizeGB`, and `diskIndex`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "itemCount": {
      "type": "int",
      "defaultValue": 5
    }
  },
  "variables": {
    "copy": [
      {
        "name": "objectArray",
        "count": "[parameters('itemCount')]",
        "input": {
          "name": "[concat('myDataDisk', copyIndex('objectArray', 1))]",
          "diskSizeGB": "1",
          "diskIndex": "[copyIndex('objectArray')]"
        }
      }
    ]
  },
  "resources": [],
  "outputs": {
    "arrayResult": {
      "type": "array",
      "value": "[variables('objectArray')]"
    }
  }
}
```

The preceding example returns an array with the following values:

JSON

```
[
  {
```

```
    "name": "myDataDisk1",
    "diskSizeGB": "1",
    "diskIndex": 0
  },
  {
    "name": "myDataDisk2",
    "diskSizeGB": "1",
    "diskIndex": 1
  },
  {
    "name": "myDataDisk3",
    "diskSizeGB": "1",
    "diskIndex": 2
  },
  {
    "name": "myDataDisk4",
    "diskSizeGB": "1",
    "diskIndex": 3
  },
  {
    "name": "myDataDisk5",
    "diskSizeGB": "1",
    "diskIndex": 4
  }
]
```

> **Note**
>
> Variable iteration supports an offset argument. The offset must come after the name of the iteration, such as `copyIndex('diskNames', 1)`. If you don't provide an offset value, it defaults to 0 for the first instance.

You can also use the `copy` element within a variable. The following example creates an object that has an array as one of its values.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "itemCount": {
      "type": "int",
      "defaultValue": 5
    }
  },
  "variables": {
    "topLevelObject": {
      "sampleProperty": "sampleValue",
```

```
      "copy": [
        {
          "name": "disks",
          "count": "[parameters('itemCount')]",
          "input": {
            "name": "[concat('myDataDisk', copyIndex('disks', 1))]",
            "diskSizeGB": "1",
            "diskIndex": "[copyIndex('disks')]"
          }
        }
      ]
    }
  },
  "resources": [],
  "outputs": {
    "objectResult": {
      "type": "object",
      "value": "[variables('topLevelObject')]"
    }
  }
}
```

The preceding example returns an object with the following values:

JSON

```
{
  "sampleProperty": "sampleValue",
  "disks": [
    {
      "name": "myDataDisk1",
      "diskSizeGB": "1",
      "diskIndex": 0
    },
    {
      "name": "myDataDisk2",
      "diskSizeGB": "1",
      "diskIndex": 1
    },
    {
      "name": "myDataDisk3",
      "diskSizeGB": "1",
      "diskIndex": 2
    },
    {
      "name": "myDataDisk4",
      "diskSizeGB": "1",
      "diskIndex": 3
    },
    {
      "name": "myDataDisk5",
      "diskSizeGB": "1",
      "diskIndex": 4
```

```
    }
  ]
}
```

The next example shows the different ways you can use `copy` with variables.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {},
  "variables": {
    "disk-array-on-object": {
      "copy": [
        {
          "name": "disks",
          "count": 5,
          "input": {
            "name": "[concat('myDataDisk', copyIndex('disks', 1))]",
            "diskSizeGB": "1",
            "diskIndex": "[copyIndex('disks')]"
          }
        },
        {
          "name": "diskNames",
          "count": 5,
          "input": "[concat('myDataDisk', copyIndex('diskNames', 1))]"
        }
      ]
    },
    "copy": [
      {
        "name": "top-level-object-array",
        "count": 5,
        "input": {
          "name": "[concat('myDataDisk', copyIndex('top-level-object-array',
1))]",
          "diskSizeGB": "1",
          "diskIndex": "[copyIndex('top-level-object-array')]"
        }
      },
      {
        "name": "top-level-string-array",
        "count": 5,
        "input": "[concat('myDataDisk', copyIndex('top-level-string-array',
1))]"
      },
      {
        "name": "top-level-integer-array",
        "count": 5,
        "input": "[copyIndex('top-level-integer-array')]"
```

```
      }
    ]
  },
  "resources": [],
  "outputs": {
    "exampleObject": {
      "value": "[variables('disk-array-on-object')]",
      "type": "object"
    },
    "exampleArrayOnObject": {
      "value": "[variables('disk-array-on-object').disks]",
      "type" : "array"
    },
    "exampleObjectArray": {
      "value": "[variables('top-level-object-array')]",
      "type" : "array"
    },
    "exampleStringArray": {
      "value": "[variables('top-level-string-array')]",
      "type" : "array"
    },
    "exampleIntegerArray": {
      "value": "[variables('top-level-integer-array')]",
      "type" : "array"
    }
  }
}
```

Example templates

The following examples show common scenarios for creating more than one value for a variable.

Template	Description
Copy variables	Demonstrates the different ways of iterating on variables.
Multiple security rules	Deploys several security rules to a network security group. It constructs the security rules from a parameter. For the parameter, see multiple NSG parameter file.
Copy storage with variables	Example of how to iterate a variable and create multiple storage accounts.

Next steps

- To go through a tutorial, see Tutorial: Create multiple resource instances with ARM templates.
- For other uses of the copy element, see:
  - Resource iteration in ARM templates
  - Property iteration in ARM templates
  - Output iteration in ARM templates
- If you want to learn about the sections of a template, see Understand the structure and syntax of ARM templates.
- To learn how to deploy your template, see Deploy resources with ARM templates and Azure PowerShell.

Output iteration in ARM templates

Article • 04/09/2023

This article shows you how to create more than one value for an output in your Azure Resource Manager template (ARM template). By adding copy loop to the outputs section of your template, you can dynamically return a number of items during deployment.

You can also use copy loop with resources, properties in a resource, and variables.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **loops**.

Syntax

Add the `copy` element to the output section of your template to return a number of items. The copy element has the following general format:

```JSON
"copy": {
  "count": <number-of-iterations>,
  "input": <values-for-the-output>
}
```

The `count` property specifies the number of iterations you want for the output value.

The `input` property specifies the properties that you want to repeat. You create an array of elements constructed from the value in the `input` property. It can be a single property (like a string), or an object with several properties.

Copy limits

The count can't exceed 800.

The count can't be a negative number. It can be zero if you deploy the template with a recent version of Azure CLI, PowerShell, or REST API. Specifically, you must use:

- Azure PowerShell **2.6** or later

- Azure CLI **2.0.74** or later
- REST API version **2019-05-10** or later
- Linked deployments must use API version **2019-05-10** or later for the deployment resource type

Earlier versions of PowerShell, CLI, and the REST API don't support zero for count.

Outputs iteration

The following example creates a variable number of storage accounts and returns an endpoint for each storage account:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageCount": {
      "type": "int",
      "defaultValue": 2
    }
  },
  "variables": {
    "baseName": "[concat('storage', uniqueString(resourceGroup().id))]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2019-04-01",
      "name": "[concat(copyIndex(), variables('baseName'))]",
      "location": "[resourceGroup().location]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "Storage",
      "properties": {},
      "copy": {
        "name": "storagecopy",
        "count": "[parameters('storageCount')]"
      }
    }
  ],
  "outputs": {
    "storageEndpoints": {
      "type": "array",
      "copy": {
        "count": "[parameters('storageCount')]",
        "input": "[reference(concat(copyIndex(), variables('baseName'))).primaryEndpoints.blob]"
```

```
      }
    }
  }
}
```

The preceding template returns an array with the following values:

JSON

```
[
  "https://0storagecfrbqnnmpeudi.blob.core.windows.net/",
  "https://1storagecfrbqnnmpeudi.blob.core.windows.net/"
]
```

The next example returns three properties from the new storage accounts.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageCount": {
      "type": "int",
      "defaultValue": 2
    }
  },
  "variables": {
    "baseName": "[concat('storage', uniqueString(resourceGroup().id))]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2019-04-01",
      "name": "[concat(copyIndex(), variables('baseName'))]",
      "location": "[resourceGroup().location]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "Storage",
      "properties": {},
      "copy": {
        "name": "storagecopy",
        "count": "[parameters('storageCount')]"
      }
    }
  ],
  "outputs": {
    "storageInfo": {
      "type": "array",
      "copy": {
```

```
        "count": "[parameters('storageCount')]",
        "input": {
          "id": "[reference(concat(copyIndex(), variables('baseName')),
'2019-04-01', 'Full').resourceId]",
          "blobEndpoint": "[reference(concat(copyIndex(),
variables('baseName'))).primaryEndpoints.blob]",
          "status": "[reference(concat(copyIndex(),
variables('baseName'))).statusOfPrimary]"
        }
      }
    }
  }
}
```

The preceding example returns an array with the following values:

JSON

```
[
  {
    "id": "Microsoft.Storage/storageAccounts/0storagecfrbqnnmpeudi",
    "blobEndpoint": "https://0storagecfrbqnnmpeudi.blob.core.windows.net/",
    "status": "available"
  },
  {
    "id": "Microsoft.Storage/storageAccounts/1storagecfrbqnnmpeudi",
    "blobEndpoint": "https://1storagecfrbqnnmpeudi.blob.core.windows.net/",
    "status": "available"
  }
]
```

Next steps

- To go through a tutorial, see Tutorial: Create multiple resource instances with ARM templates.
- For other uses of the copy loop, see:
  - Resource iteration in ARM templates
  - Property iteration in ARM templates
  - Variable iteration in ARM templates
- If you want to learn about the sections of a template, see Understand the structure and syntax of ARM templates.
- To learn how to deploy your template, see Deploy resources with ARM templates and Azure PowerShell.

Deploy resources with ARM templates and Azure portal

Article • 09/10/2021

Learn how to use the Azure portal with Azure Resource Manager templates (ARM templates) to deploy your Azure resources. To learn about managing your resources, see Manage Azure resources by using the Azure portal.

Deploying Azure resources by using the Azure portal usually involves two steps:

- Create a resource group.
- Deploy resources to the resource group.

Also, you can create a customized ARM template to deploy Azure resources.

This article shows both methods.

Create a resource group

1. To create a new resource group, select **Resource groups** from the Azure portal.

2. Under Resource groups, select **Add**.

3. Select or enter the following property values:

   - **Subscription**: Select an Azure subscription.
   - **Resource group**: Give the resource group a name.
   - **Region**: Specify an Azure location. This location is where the resource group stores metadata about the resources. For compliance reasons, you may want to specify where that metadata is stored. In general, we recommend that you specify a location where most of your resources will be. Using the same location can simplify your template.

4. Select **Review + create**.

5. Review the values, and then select **Create**.

6. Select **Refresh** before you can see the new resource group in the list.

Deploy resources to a resource group

After you create a resource group, you can deploy resources to the group from the Marketplace. The Marketplace provides pre-defined solutions for common scenarios.

1. To start a deployment, select **Create a resource** from the Azure portal.

2. Find the type of resource you would like to deploy. The resources are organized in categories. If you don't see the particular solution you would like to deploy, you can search the Marketplace for it. The following screenshot shows that Ubuntu Server is selected.

3. Depending on the type of selected resource, you have a collection of relevant properties to set before deployment. For all types, you must select a destination resource group. The following image shows how to create a Linux virtual machine and deploy it to the resource group you created.

You can decide to create a resource group when deploying your resources. Select **Create new** and give the resource group a name.

4. Your deployment begins. The deployment could take several minutes. Some resources take longer time than other resources. When the deployment has finished, you see a notification. Select **Go to resource** to open

5. After deploying your resources, you can add more resources to the resource group by selecting **Add**.

Although you didn't see it, the portal used an ARM template to deploy the resources you selected. You can find the template from the deployment history. For more information, see Export template after deployment.

Deploy resources from custom template

If you want to execute a deployment but not use any of the templates in the Marketplace, you can create a customized template that defines the infrastructure for your solution. To learn about creating templates, see Understand the structure and syntax of ARM templates.

> **Note**
>
> The portal interface doesn't support referencing a **secret from a Key Vault**. Instead, use **PowerShell** or **Azure CLI** to deploy your template locally or from an external URI.

1. To deploy a customized template through the portal, select **Create a resource**, search for **template**. and then select **Template deployment**.

2. Select **Create**.

3. You see several options for creating a template:

- **Build your own template in editor**: Create your own template in the portal template editor.
- **Common templates**: Select from common solutions.
- **Load a GitHub quickstart template**: Select from quickstart templates.

This tutorial provides the instruction for loading a quickstart template.

4. Under **Load a GitHub quickstart template**, type or select **storage-account-create**.

   You have two options:

   - **Select template**: deploy the template.
   - **Edit template**: edit the quickstart template before you deploy it.

5. Select **Edit template** to explore the portal template editor. The template is loaded in the editor. Notice there are two parameters: `storageAccountType` and `location`.

6. Make a minor change to the template. For example, update the `storageAccountName` variable to:

```
JSON

"storageAccountName": "[concat('azstore',
uniquestring(resourceGroup().id))]"
```

7. Select **Save**. Now you see the portal template deployment interface. Notice the two parameters that you defined in the template.

8. Enter or select the property values:

   - **Subscription**: Select an Azure subscription.
   - **Resource group**: Select **Create new** and give a name.
   - **Location**: Select an Azure location.
   - **Storage Account Type**: Use the default value. The camel-cased parameter name, *storageAccountType*, defined in the template is turned into a space-separated string when displayed on the portal.
   - **Location**: Use the default value.
   - **I agree to the terms and conditions stated above**: (select)

9. Select **Purchase**.

Next steps

- To troubleshoot deployment errors, see View deployment operations.
- To export a template from a deployment or resource group, see Export ARM templates.

How to use Azure Resource Manager (ARM) deployment templates with Azure CLI

Article • 04/09/2023

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

The deployment commands changed in Azure CLI version 2.2.0. The examples in this article require Azure CLI version 2.20.0 or later.

To run this sample, install the latest version of the Azure CLI. To start, run `az login` to create a connection with Azure.

Samples for the Azure CLI are written for the `bash` shell. To run this sample in Windows PowerShell or Command Prompt, you may need to change elements of the script.

If you don't have Azure CLI installed, you can use Azure Cloud Shell. For more information, see Deploy ARM templates from Azure Cloud Shell.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **How to deploy resources with Bicep and Azure CLI**.

Required permissions

To deploy a Bicep file or ARM template, you need write access on the resources you're deploying and access to all operations on the Microsoft.Resources/deployments resource type. For example, to deploy a virtual machine, you need `Microsoft.Compute/virtualMachines/write` and `Microsoft.Resources/deployments/*` permissions. The what-if operation has the same permission requirements.

For a list of roles and permissions, see Azure built-in roles.

Deployment scope

You can target your Azure deployment template to a resource group, subscription, management group, or tenant. Depending on the scope of the deployment, you use different commands.

- To deploy to a **resource group**, use az deployment group create:

  Azure CLI

  ```
  az deployment group create --resource-group <resource-group-name> --template-file <path-to-template>
  ```

- To deploy to a **subscription**, use az deployment sub create:

  Azure CLI

  ```
  az deployment sub create --location <location> --template-file <path-to-template>
  ```

  For more information about subscription level deployments, see Create resource groups and resources at the subscription level.

- To deploy to a **management group**, use az deployment mg create:

  Azure CLI

  ```
  az deployment mg create --location <location> --template-file <path-to-template>
  ```

  For more information about management group level deployments, see Create resources at the management group level.

- To deploy to a **tenant**, use az deployment tenant create:

  Azure CLI

  ```
  az deployment tenant create --location <location> --template-file <path-to-template>
  ```

  For more information about tenant level deployments, see Create resources at the tenant level.

For every scope, the user deploying the template must have the required permissions to create resources.

Deploy local template

You can deploy an ARM template from your local machine or one that is stored externally. This section describes deploying a local template.

If you're deploying to a resource group that doesn't exist, create the resource group. The name of the resource group can only include alphanumeric characters, periods, underscores, hyphens, and parenthesis. It can be up to 90 characters. The name can't end in a period.

Azure CLI

```
az group create --name ExampleGroup --location "Central US"
```

To deploy a local template, use the `--template-file` parameter in the deployment command. The following example also shows how to set a parameter value.

Azure CLI

```
az deployment group create \
  --name ExampleDeployment \
  --resource-group ExampleGroup \
  --template-file <path-to-template> \
  --parameters storageAccountType=Standard_GRS
```

The value of the `--template-file` parameter must be a Bicep file or a `.json` or `.jsonc` file. The `.jsonc` file extension indicates the file can contain `//` style comments. The ARM system accepts `//` comments in `.json` files. It does not care about the file extension. For more details about comments and metadata see Understand the structure and syntax of ARM templates.

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

Output

```
"provisioningState": "Succeeded",
```

Deploy remote template

Instead of storing ARM 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.

> **Note**
>
> To deploy a template or reference a linked template that is stored in a private GitHub repo, see a custom solution documented in **Creating a Custom and Secure Azure Portal Offering**. You can create an **Azure function** that pulls the GitHub token out of Azure Key Vault.

If you're deploying to a resource group that doesn't exist, create the resource group. The name of the resource group can only include alphanumeric characters, periods, underscores, hyphens, and parenthesis. It can be up to 90 characters. The name can't end in a period.

Azure CLI

```
az group create --name ExampleGroup --location "Central US"
```

To deploy an external template, use the `template-uri` parameter.

Azure CLI

```
az deployment group create \
  --name ExampleDeployment \
  --resource-group ExampleGroup \
  --template-uri "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.storage/storage-account-create/azuredeploy.json" \
  --parameters storageAccountType=Standard_GRS
```

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 want to manage access to the template, consider using template specs.

To deploy remote linked templates with relative path that are stored in a storage account, use `query-string` to specify the SAS token:

Azure CLI

```
az deployment group create \
  --name linkedTemplateWithRelativePath \
  --resource-group myResourceGroup \
```

```
  --template-uri "https://stage20210126.blob.core.windows.net/template-staging/mainTemplate.json" \
  --query-string $sasToken
```

For more information, see Use relative path for linked templates.

Azure deployment template name

When deploying an ARM template, you can give the Azure deployment template a name. This name can help you retrieve the deployment from the deployment history. If you don't provide a name for the deployment, the name of the template file is used. For example, if you deploy a template named *azuredeploy.json* and don't specify a deployment name, the deployment is named `azuredeploy`.

Every time you run a deployment, an entry is added to the resource group's deployment history with the deployment name. If you run another deployment and give it the same name, the earlier entry is replaced with the current deployment. If you want to maintain unique entries in the deployment history, give each deployment a unique name.

To create a unique name, you can assign a random number.

Azure CLI

```
deploymentName='ExampleDeployment'$RANDOM
```

Or, add a date value.

Azure CLI

```
deploymentName='ExampleDeployment'$(date +"%d-%b-%Y")
```

If you run concurrent deployments to the same resource group with the same deployment name, only the last deployment is completed. Any deployments with the same name that haven't finished are replaced by the last deployment. For example, if you run a deployment named `newStorage` that deploys a storage account named `storage1`, and at the same time run another deployment named `newStorage` that deploys a storage account named `storage2`, you deploy only one storage account. The resulting storage account is named `storage2`.

However, if you run a deployment named `newStorage` that deploys a storage account named `storage1`, and immediately after it completes you run another deployment named `newStorage` that deploys a storage account named `storage2`, then you have two

storage accounts. One is named `storage1`, and the other is named `storage2`. But, you only have one entry in the deployment history.

When you specify a unique name for each deployment, you can run them concurrently without conflict. If you run a deployment named `newStorage1` that deploys a storage account named `storage1`, and at the same time run another deployment named `newStorage2` that deploys a storage account named `storage2`, then you have two storage accounts and two entries in the deployment history.

To avoid conflicts with concurrent deployments and to ensure unique entries in the deployment history, give each deployment a unique name.

Deploy template spec

Instead of deploying a local or remote template, you can create a template spec. The template spec is a resource in your Azure subscription that contains an ARM template. It makes it easy to securely share the template with users in your organization. You use Azure role-based access control (Azure RBAC) to grant access to the template spec. This feature is currently in preview.

The following examples show how to create and deploy a template spec.

First, create the template spec by providing the ARM template.

Azure CLI

```
az ts create \
  --name storageSpec \
  --version "1.0" \
  --resource-group templateSpecRG \
  --location "westus2" \
  --template-file "./mainTemplate.json"
```

Then, get the ID for template spec and deploy it.

Azure CLI

```
id = $(az ts show --name storageSpec --resource-group templateSpecRG --
version "1.0" --query "id")

az deployment group create \
  --resource-group demoRG \
  --template-spec $id
```

For more information, see Azure Resource Manager template specs.

Preview changes

Before deploying your ARM template, you can preview the changes the template will make to your environment. Use the what-if operation to verify that the template makes the changes that you expect. What-if also validates the template for errors.

Parameters

To pass parameter values, you can use either inline parameters or a parameter file.

Inline parameters

To pass inline parameters, provide the values in `parameters`. For example, to pass a string and array to a template in a Bash shell, use:

Azure CLI

```
az deployment group create \
  --resource-group testgroup \
  --template-file <path-to-template> \
  --parameters exampleString='inline string' exampleArray='("value1",
"value2")'
```

If you're using Azure CLI with Windows Command Prompt (CMD) or PowerShell, pass the array in the format: `exampleArray="['value1','value2']"`.

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

Azure CLI

```
az deployment group create \
  --resource-group testgroup \
  --template-file <path-to-template> \
  --parameters exampleString=@stringContent.txt
exampleArray=@arrayContent.json
```

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.

The *arrayContent.json* format is:

JSON

```
[
    "value1",
    "value2"
]
```

To pass in an object, for example, to set tags, use JSON. For example, your template might include a parameter like this one:

JSON

```
    "resourceTags": {
      "type": "object",
      "defaultValue": {
        "Cost Center": "IT Department"
      }
    }
```

In this case, you can pass in a JSON string to set the parameter as shown in the following Bash script:

Azure CLI

```
tags='{"Owner":"Contoso","Cost Center":"2345-324"}'
az deployment group create --name addstorage  --resource-group
myResourceGroup \
--template-file $templateFile \
--parameters resourceName=abcdef4556 resourceTags="$tags"
```

Use double quotes around the JSON that you want to pass into the object.

You can use a variable to contain the parameter values. In Bash, set the variable to all of the parameter values and add it to the deployment command.

Azure CLI

```
params="prefix=start suffix=end"

az deployment group create \
  --resource-group testgroup \
  --template-file <path-to-template> \
  --parameters $params
```

However, if you're using Azure CLI with Windows Command Prompt (CMD) or PowerShell, set the variable to a JSON string. Escape the quotation marks: `$params = '{ \"prefix\": {\"value\":\"start\"}, \"suffix\": {\"value\":\"end\"} }'`.

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 must be a local file. External parameter files aren't supported with Azure CLI.

For more information about the parameter file, see Create Resource Manager parameter file.

To pass a local parameter file, use `@` to specify a local file named *storage.parameters.json*.

Azure CLI

```
az deployment group create \
  --name ExampleDeployment \
  --resource-group ExampleGroup \
  --template-file storage.json \
  --parameters '@storage.parameters.json'
```

Comments and the extended JSON format

You can include `//` style comments in your parameter file, but you must name the file with a `.jsonc` extension.

Azure CLI

```
az deployment group create \
  --name ExampleDeployment \
  --resource-group ExampleGroup \
  --template-file storage.json \
  --parameters '@storage.parameters.jsonc'
```

For more details about comments and metadata see Understand the structure and syntax of ARM templates.

If you are using Azure CLI with version 2.3.0 or older, you can deploy a template with multi-line strings or comments using the `--handle-extended-json-format` switch. For example:

JSON

```
{
  "type": "Microsoft.Compute/virtualMachines",
  "apiVersion": "2018-10-01",
```

```
  "name": "[variables('vmName')]", // to customize name, change it in
variables
  "location": "[
    parameters('location')
    ]", //defaults to resource group location
  /*
    storage account and network interface
    must be deployed first
  */
  "dependsOn": [
    "[resourceId('Microsoft.Storage/storageAccounts/',
variables('storageAccountName'))]",
    "[resourceId('Microsoft.Network/networkInterfaces/',
variables('nicName'))]"
  ],
```

Next steps

- To roll back to a successful deployment when you get an error, see Rollback on error to successful deployment.
- 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 ARM templates.
- For tips on resolving common deployment errors, see Troubleshoot common Azure deployment errors with Azure Resource Manager.

Deploy resources with ARM templates and Azure PowerShell

Article • 04/09/2023

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

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **Deploy resources with Bicep and Azure PowerShell**.

Prerequisites

You need a template to deploy. If you don't already have one, download and save an example template from the Azure Quickstart templates repo. The local file name used in this article is *C:\MyTemplates\azuredeploy.json*.

You need to install Azure PowerShell and connect to Azure:

- **Install Azure PowerShell cmdlets on your local computer.** For more information, see Get started with Azure PowerShell.
- **Connect to Azure by using Connect-AZAccount**. If you have multiple Azure subscriptions, you might also need to run Set-AzContext. For more information, see Use multiple Azure subscriptions.

If you don't have PowerShell installed, you can use Azure Cloud Shell. For more information, see Deploy ARM templates from Azure Cloud Shell.

Required permissions

To deploy a Bicep file or ARM template, you need write access on the resources you're deploying and access to all operations on the Microsoft.Resources/deployments resource type. For example, to deploy a virtual machine, you need `Microsoft.Compute/virtualMachines/write` and `Microsoft.Resources/deployments/*` permissions. The what-if operation has the same permission requirements.

For a list of roles and permissions, see Azure built-in roles.

Deployment scope

You can target your deployment to a resource group, subscription, management group, or tenant. Depending on the scope of the deployment, you use different commands.

- To deploy to a **resource group**, use New-AzResourceGroupDeployment:

  Azure PowerShell

  ```
  New-AzResourceGroupDeployment -ResourceGroupName <resource-group-name>
  -TemplateFile <path-to-template>
  ```

- To deploy to a **subscription**, use New-AzSubscriptionDeployment which is an alias of the `New-AzDeployment` cmdlet:

  Azure PowerShell

  ```
  New-AzSubscriptionDeployment -Location <location> -TemplateFile <path-
  to-template>
  ```

  For more information about subscription level deployments, see Create resource groups and resources at the subscription level.

- To deploy to a **management group**, use New-AzManagementGroupDeployment.

  Azure PowerShell

  ```
  New-AzManagementGroupDeployment -Location <location> -TemplateFile
  <path-to-template>
  ```

  For more information about management group level deployments, see Create resources at the management group level.

- To deploy to a **tenant**, use New-AzTenantDeployment.

  Azure PowerShell

  ```
  New-AzTenantDeployment -Location <location> -TemplateFile <path-to-
  template>
  ```

  For more information about tenant level deployments, see Create resources at the tenant level.

For every scope, the user deploying the template must have the required permissions to create resources.

Deployment name

When deploying an ARM template, you can give the deployment a name. This name can help you retrieve the deployment from the deployment history. If you don't provide a name for the deployment, the name of the template file is used. For example, if you deploy a template named `azuredeploy.json` and don't specify a deployment name, the deployment is named `azuredeploy`.

Every time you run a deployment, an entry is added to the resource group's deployment history with the deployment name. If you run another deployment and give it the same name, the earlier entry is replaced with the current deployment. If you want to maintain unique entries in the deployment history, give each deployment a unique name.

To create a unique name, you can assign a random number.

Azure PowerShell

```
$suffix = Get-Random -Maximum 1000
$deploymentName = "ExampleDeployment" + $suffix
```

Or, add a date value.

Azure PowerShell

```
$today=Get-Date -Format "MM-dd-yyyy"
$deploymentName="ExampleDeployment"+"$today"
```

If you run concurrent deployments to the same resource group with the same deployment name, only the last deployment is completed. Any deployments with the same name that haven't finished are replaced by the last deployment. For example, if you run a deployment named `newStorage` that deploys a storage account named `storage1`, and at the same time run another deployment named `newStorage` that deploys a storage account named `storage2`, you deploy only one storage account. The resulting storage account is named `storage2`.

However, if you run a deployment named `newStorage` that deploys a storage account named `storage1`, and immediately after it completes you run another deployment named `newStorage` that deploys a storage account named `storage2`, then you have two

storage accounts. One is named `storage1`, and the other is named `storage2`. But, you only have one entry in the deployment history.

When you specify a unique name for each deployment, you can run them concurrently without conflict. If you run a deployment named `newStorage1` that deploys a storage account named `storage1`, and at the same time run another deployment named `newStorage2` that deploys a storage account named `storage2`, then you have two storage accounts and two entries in the deployment history.

To avoid conflicts with concurrent deployments and to ensure unique entries in the deployment history, give each deployment a unique name.

Deploy local template

You can deploy a template from your local machine or one that is stored externally. This section describes deploying a local template.

If you're deploying to a resource group that doesn't exist, create the resource group. The name of the resource group can only include alphanumeric characters, periods, underscores, hyphens, and parenthesis. It can be up to 90 characters. The name can't end in a period.

Azure PowerShell

```
New-AzResourceGroup -Name ExampleGroup -Location "Central US"
```

To deploy a local template, use the `-TemplateFile` parameter in the deployment command. The following example also shows how to set a parameter value that comes from the template.

Azure PowerShell

```
New-AzResourceGroupDeployment `
  -Name ExampleDeployment `
  -ResourceGroupName ExampleGroup `
  -TemplateFile <path-to-template>
```

The deployment can take several minutes to complete.

Deploy remote template

Instead of storing ARM 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.

> **Note**
>
> To deploy a template or reference a linked template that is stored in a private GitHub repo, see a custom solution documented in **Creating a Custom and Secure Azure Portal Offering**. You can create an **Azure function** that pulls the GitHub token out of Azure Key Vault.

If you're deploying to a resource group that doesn't exist, create the resource group. The name of the resource group can only include alphanumeric characters, periods, underscores, hyphens, and parenthesis. It can be up to 90 characters. The name can't end in a period.

Azure PowerShell

```
New-AzResourceGroup -Name ExampleGroup -Location "Central US"
```

To deploy an external template, use the `-TemplateUri` parameter.

Azure PowerShell

```
New-AzResourceGroupDeployment `
  -Name remoteTemplateDeployment `
  -ResourceGroupName ExampleGroup `
  -TemplateUri https://raw.githubusercontent.com/Azure/azure-quickstart-
templates/master/quickstarts/microsoft.storage/storage-account-
create/azuredeploy.json
```

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 want to manage access to the template, consider using template specs.

To deploy remote linked templates with relative path that are stored in a storage account, use `QueryString` to specify the SAS token:

Azure PowerShell

```
New-AzResourceGroupDeployment `
  -Name linkedTemplateWithRelativePath `
```

```
  -ResourceGroupName "myResourceGroup" `
  -TemplateUri "https://stage20210126.blob.core.windows.net/template-
staging/mainTemplate.json" `
  -QueryString "$sasToken"
```

For more information, see Use relative path for linked templates.

Deploy template spec

Instead of deploying a local or remote template, you can create a template spec. The template spec is a resource in your Azure subscription that contains an ARM template. It makes it easy to securely share the template with users in your organization. You use Azure role-based access control (Azure RBAC) to grant access to the template spec. This feature is currently in preview.

The following examples show how to create and deploy a template spec.

First, create the template spec by providing the ARM template.

Azure PowerShell

```
New-AzTemplateSpec `
  -Name storageSpec `
  -Version 1.0 `
  -ResourceGroupName templateSpecsRg `
  -Location westus2 `
  -TemplateJsonFile ./mainTemplate.json
```

Then, get the ID for template spec and deploy it.

Azure PowerShell

```
$id = (Get-AzTemplateSpec -Name storageSpec -ResourceGroupName
templateSpecsRg -Version 1.0).Versions.Id

New-AzResourceGroupDeployment `
  -ResourceGroupName demoRG `
  -TemplateSpecId $id
```

For more information, see Azure Resource Manager template specs.

Preview changes

Before deploying your template, you can preview the changes the template will make to your environment. Use the what-if operation to verify that the template makes the

changes that you expect. What-if also validates the template for errors.

Pass parameter values

To pass parameter values, you can use either inline parameters or a parameter file.

Inline parameters

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

PowerShell

```
$arrayParam = "value1", "value2"
New-AzResourceGroupDeployment -ResourceGroupName testgroup `
  -TemplateFile <path-to-template> `
  -exampleString "inline string" `
  -exampleArray $arrayParam
```

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

PowerShell

```
$arrayParam = "value1", "value2"
New-AzResourceGroupDeployment -ResourceGroupName testgroup `
  -TemplateFile <path-to-template> `
  -exampleString $(Get-Content -Path c:\MyTemplates\stringcontent.txt -Raw)
`
  -exampleArray $arrayParam
```

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.

If you need to pass in an array of objects, create hash tables in PowerShell and add them to an array. Pass that array as a parameter during deployment.

PowerShell

```
$hash1 = @{ Name = "firstSubnet"; AddressPrefix = "10.0.0.0/24"}
$hash2 = @{ Name = "secondSubnet"; AddressPrefix = "10.0.1.0/24"}
$subnetArray = $hash1, $hash2
New-AzResourceGroupDeployment -ResourceGroupName testgroup `
  -TemplateFile <path-to-template> `
  -exampleArray $subnetArray
```

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.

For more information about the parameter file, see Create Resource Manager parameter file.

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

PowerShell

```
New-AzResourceGroupDeployment -Name ExampleDeployment -ResourceGroupName
ExampleResourceGroup `
  -TemplateFile <path-to-template> `
  -TemplateParameterFile c:\MyTemplates\storage.parameters.json
```

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

PowerShell

```
New-AzResourceGroupDeployment -Name ExampleDeployment -ResourceGroupName
ExampleResourceGroup `
  -TemplateUri https://raw.githubusercontent.com/Azure/azure-quickstart-
templates/master/quickstarts/microsoft.storage/storage-account-
create/azuredeploy.json `
  -TemplateParameterUri https://raw.githubusercontent.com/Azure/azure-
quickstart-templates/master/quickstarts/microsoft.storage/storage-account-
create/azuredeploy.parameters.json
```

Next steps

- To roll back to a successful deployment when you get an error, see Rollback on error to successful deployment.
- 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 ARM templates.
- For information about deploying a template that requires a SAS token, see Deploy private ARM template with SAS token.

Deploy resources with ARM templates and Python

Article • 04/25/2023

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

> **Note**
>
> This article was partially created with the help of artificial intelligence. Before publishing, an author reviewed and revised the content as needed. See **Our principles for using AI-generated content in Microsoft Learn**.

Prerequisites

- A template to deploy. If you don't already have one, download and save an example template from the Azure Quickstart templates repo.
- Python 3.7 or later installed. To install the latest, see Python.org
- The following Azure library packages for Python installed in your virtual environment. To install any of the packages, use `pip install {package-name}`
  - azure-identity
  - azure-mgmt-resource

  If you have older versions of these packages already installed in your virtual environment, you may need to update them with `pip install --upgrade {package-name}`
- The examples in this article use CLI-based authentication (`AzureCliCredential`). Depending on your environment, you may need to run `az login` first to authenticate.

Required permissions

To deploy a Bicep file or ARM template, you need write access on the resources you're deploying and access to all operations on the Microsoft.Resources/deployments resource type. For example, to deploy a virtual machine, you need

`Microsoft.Compute/virtualMachines/write` and `Microsoft.Resources/deployments/*` permissions. The what-if operation has the same permission requirements.

For a list of roles and permissions, see Azure built-in roles.

Deployment scope

You can target your deployment to a resource group, subscription, management group, or tenant. Depending on the scope of the deployment, you use different methods.

- To deploy to a **resource group**, use ResourceManagementClient.deployments.begin_create_or_update:
- To deploy to a **subscription**, use ResourceManagementClient.deployments.begin_create_or_update_at_subscription_scope:

  For more information about subscription level deployments, see Create resource groups and resources at the subscription level.
- To deploy to a **management group**, use ResourceManagementClient.deployments.begin_create_or_update_at_management_group_scope.

  For more information about management group level deployments, see Create resources at the management group level.
- To deploy to a **tenant**, use ResourceManagementClient.deployments.begin_create_or_update_at_tenant_scope.

  For more information about tenant level deployments, see Create resources at the tenant level.

For every scope, the user deploying the template must have the required permissions to create resources.

Deployment name

When deploying an ARM template, you can give the deployment a name. This name can help you retrieve the deployment from the deployment history. If you don't provide a name for the deployment, the name of the template file is used. For example, if you

deploy a template named `azuredeploy.json` and don't specify a deployment name, the deployment is named `azuredeploy`.

Every time you run a deployment, an entry is added to the resource group's deployment history with the deployment name. If you run another deployment and give it the same name, the earlier entry is replaced with the current deployment. If you want to maintain unique entries in the deployment history, give each deployment a unique name.

To create a unique name, you can assign a random number.

Python

```python
import random

suffix = random.randint(1, 1000)
deployment_name = f"ExampleDeployment{suffix}"
```

Or, add a date value.

Python

```python
from datetime import datetime

today = datetime.now().strftime("%m-%d-%Y")
deployment_name = f"ExampleDeployment{today}"
```

If you run concurrent deployments to the same resource group with the same deployment name, only the last deployment is completed. Any deployments with the same name that haven't finished are replaced by the last deployment. For example, if you run a deployment named `newStorage` that deploys a storage account named `storage1`, and at the same time run another deployment named `newStorage` that deploys a storage account named `storage2`, you deploy only one storage account. The resulting storage account is named `storage2`.

However, if you run a deployment named `newStorage` that deploys a storage account named `storage1`, and immediately after it completes you run another deployment named `newStorage` that deploys a storage account named `storage2`, then you have two storage accounts. One is named `storage1`, and the other is named `storage2`. But, you only have one entry in the deployment history.

When you specify a unique name for each deployment, you can run them concurrently without conflict. If you run a deployment named `newStorage1` that deploys a storage account named `storage1`, and at the same time run another deployment named

`newStorage2` that deploys a storage account named `storage2`, then you have two storage accounts and two entries in the deployment history.

To avoid conflicts with concurrent deployments and to ensure unique entries in the deployment history, give each deployment a unique name.

Deploy local template

You can deploy a template from your local machine or one that is stored externally. This section describes deploying a local template.

If you're deploying to a resource group that doesn't exist, create the resource group. The name of the resource group can only include alphanumeric characters, periods, underscores, hyphens, and parenthesis. It can be up to 90 characters. The name can't end in a period.

Python

```python
import os
from azure.identity import AzureCliCredential
from azure.mgmt.resource import ResourceManagementClient

credential = AzureCliCredential()
subscription_id = os.environ["AZURE_SUBSCRIPTION_ID"]

resource_client = ResourceManagementClient(credential, subscription_id)

rg_result = resource_client.resource_groups.create_or_update(
    "exampleGroup",
    {
        "location": "Central US"
    }
)

print(f"Provisioned resource group with ID: {rg_result.id}")
```

To deploy an ARM template, use ResourceManagementClient.deployments.begin_create_or_update. The following example requires a local template named `storage.json`.

Python

```python
import os
import json
from azure.identity import AzureCliCredential
from azure.mgmt.resource import ResourceManagementClient
from azure.mgmt.resource.resources.models import DeploymentMode
```

```
credential = AzureCliCredential()
subscription_id = os.environ["AZURE_SUBSCRIPTION_ID"]

resource_client = ResourceManagementClient(credential, subscription_id)

with open("storage.json", "r") as template_file:
    template_body = json.load(template_file)

rg_deployment_result = resource_client.deployments.begin_create_or_update(
    "exampleGroup",
    "exampleDeployment",
    {
        "properties": {
            "template": template_body,
            "parameters": {
                "storagePrefix": {
                    "value": "demostore"
                },
            },
            "mode": DeploymentMode.incremental
        }
    }
)
```

The deployment can take several minutes to complete.

Deploy remote template

Instead of storing ARM 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.

If you're deploying to a resource group that doesn't exist, create the resource group. The name of the resource group can only include alphanumeric characters, periods, underscores, hyphens, and parenthesis. It can be up to 90 characters. The name can't end in a period.

Python

```
import os
from azure.identity import AzureCliCredential
from azure.mgmt.resource import ResourceManagementClient

credential = AzureCliCredential()
subscription_id = os.environ["AZURE_SUBSCRIPTION_ID"]

resource_client = ResourceManagementClient(credential, subscription_id)
```

```
rg_result = resource_client.resource_groups.create_or_update(
    "exampleGroup",
    {
        "location": "Central US"
    }
)

print(f"Provisioned resource group with ID: {rg_result.id}")
```

To deploy an ARM template, use ResourceManagementClient.deployments.begin_create_or_update. The following example deploys a remote template. That template creates a storage account.

Python

```
import os
from azure.identity import AzureCliCredential
from azure.mgmt.resource import ResourceManagementClient
from azure.mgmt.resource.resources.models import DeploymentMode

credential = AzureCliCredential()
subscription_id = os.environ["AZURE_SUBSCRIPTION_ID"]

resource_client = ResourceManagementClient(credential, subscription_id)

resource_group_name = "exampleGroup"
location = "westus"
template_uri = "https://raw.githubusercontent.com/Azure/azure-quickstart-
templates/master/quickstarts/microsoft.storage/storage-account-
create/azuredeploy.json"

rg_deployment_result = resource_client.deployments.begin_create_or_update(
    resource_group_name,
    "exampleDeployment",
    {
        "properties": {
            "templateLink": {
                "uri": template_uri
            },
            "parameters": {
                "location": {
                    "value": location
                }
            },
            "mode": DeploymentMode.incremental
        }
    }
)
```

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. If you keep your templates in a storage account that doesn't allow anonymous access, you need to provide a SAS token.

Python

```python
import os
from azure.identity import AzureCliCredential
from azure.mgmt.resource import ResourceManagementClient
from azure.mgmt.resource.resources.models import DeploymentMode

credential = AzureCliCredential()
subscription_id = os.environ["AZURE_SUBSCRIPTION_ID"]
sas_token = os.environ["SAS_TOKEN"]

resource_client = ResourceManagementClient(credential, subscription_id)

resource_group_name = "exampleGroup"
location = "westus"
template_uri =
f"https://stage20230425.blob.core.windows.net/templates/storage.json?
{sas_token}"

rg_deployment_result = resource_client.deployments.begin_create_or_update(
    resource_group_name,
    "exampleDeployment",
    {
        "properties": {
            "templateLink": {
                "uri": template_uri
            },
            "parameters": {
                "location": {
                    "value": location
                }
            },
            "mode": DeploymentMode.incremental
        }
    }
)
```

For more information, see Use relative path for linked templates.

Deploy template spec

Instead of deploying a local or remote template, you can create a template spec. The template spec is a resource in your Azure subscription that contains an ARM template. It

makes it easy to securely share the template with users in your organization. You use Azure role-based access control (Azure RBAC) to grant access to the template spec.

The following examples show how to create and deploy a template spec.

First, create the template spec by providing the ARM template.

Python

```python
import os
import json
from azure.identity import AzureCliCredential
from azure.mgmt.resource.templatespecs import TemplateSpecsClient
from azure.mgmt.resource.templatespecs.models import TemplateSpecVersion,
TemplateSpec

credential = AzureCliCredential()
subscription_id = os.environ["AZURE_SUBSCRIPTION_ID"]

template_specs_client = TemplateSpecsClient(credential, subscription_id)

template_spec = TemplateSpec(
    location="westus2",
    description="Storage Spec"
)

template_specs_client.template_specs.create_or_update(
    "templateSpecsRG",
    "storageSpec",
    template_spec
)

with open("storage.json", "r") as template_file:
    template_body = json.load(template_file)

version = TemplateSpecVersion(
    location="westus2",
    description="Storage Spec",
    main_template=template_body
)

template_spec_result =
template_specs_client.template_spec_versions.create_or_update(
     "templateSpecsRG",
    "storageSpec",
    "1.0.0",
    version
)

print(f"Provisioned template spec with ID: {template_spec_result.id}")
```

Then, get the ID for template spec and deploy it.

Python

```python
import os
from azure.identity import AzureCliCredential
from azure.mgmt.resource import ResourceManagementClient
from azure.mgmt.resource.resources.models import DeploymentMode
from azure.mgmt.resource.templatespecs import TemplateSpecsClient

credential = AzureCliCredential()
subscription_id = os.environ["AZURE_SUBSCRIPTION_ID"]

resource_client = ResourceManagementClient(credential, subscription_id)
template_specs_client = TemplateSpecsClient(credential, subscription_id)

template_spec = template_specs_client.template_spec_versions.get(
    "templateSpecsRg",
    "storageSpec",
    "1.0.0"
)

rg_deployment_result = resource_client.deployments.begin_create_or_update(
    "exampleGroup",
    "exampleDeployment",
    {
        "properties": {
            "template_link": {
                "id": template_spec.id
            },
            "mode": DeploymentMode.incremental
        }
    }
)
```

For more information, see Azure Resource Manager template specs.

Preview changes

Before deploying your template, you can preview the changes the template will make to your environment. Use the what-if operation to verify that the template makes the changes that you expect. What-if also validates the template for errors.

Next steps

- To roll back to a successful deployment when you get an error, see Rollback on error to successful deployment.
- 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 ARM templates.
- For information about deploying a template that requires a SAS token, see Deploy private ARM template with SAS token.

Deploy resources with ARM templates and Azure Resource Manager REST API

Article • 04/09/2023

This article explains how to use the Azure Resource Manager REST API with Azure Resource Manager templates (ARM templates) to deploy your resources to Azure.

You can either include your template in the request body or link to a file. When using a file, it can be a local file or an external file that is available through a URI. When your template is in a storage account, you can restrict access to the template and provide a shared access signature (SAS) token during deployment.

Required permissions

To deploy a Bicep file or ARM template, you need write access on the resources you're deploying and access to all operations on the Microsoft.Resources/deployments resource type. For example, to deploy a virtual machine, you need `Microsoft.Compute/virtualMachines/write` and `Microsoft.Resources/deployments/*` permissions. The what-if operation has the same permission requirements.

For a list of roles and permissions, see Azure built-in roles.

Deployment scope

You can target your deployment to a resource group, Azure subscription, management group, or tenant. Depending on the scope of the deployment, you use different commands.

- To deploy to a **resource group**, use Deployments - Create. The request is sent to:

  HTTP

  ```
  PUT
  https://management.azure.com/subscriptions/{subscriptionId}/resourcegroups/{resourceGroupName}/providers/Microsoft.Resources/deployments/{deploymentName}?api-version=2020-10-01
  ```

- To deploy to a **subscription**, use Deployments - Create At Subscription Scope. The request is sent to:

  HTTP

```
PUT
https://management.azure.com/subscriptions/{subscriptionId}/providers/M
icrosoft.Resources/deployments/{deploymentName}?api-version=2020-10-01
```

For more information about subscription level deployments, see Create resource groups and resources at the subscription level.

- To deploy to a **management group**, use Deployments - Create At Management Group Scope. The request is sent to:

  ```HTTP
  PUT
  https://management.azure.com/providers/Microsoft.Management/managementG
  roups/{groupId}/providers/Microsoft.Resources/deployments/{deploymentNa
  me}?api-version=2020-10-01
  ```

  For more information about management group level deployments, see Create resources at the management group level.

- To deploy to a **tenant**, use Deployments - Create Or Update At Tenant Scope. The request is sent to:

  ```HTTP
  PUT
  https://management.azure.com/providers/Microsoft.Resources/deployments/
  {deploymentName}?api-version=2020-10-01
  ```

  For more information about tenant level deployments, see Create resources at the tenant level.

The examples in this article use resource group deployments.

Deploy with the REST API

1. Set common parameters and headers, including authentication tokens.

2. If you're deploying to a resource group that doesn't exist, create the resource group. Provide your subscription ID, the name of the new resource group, and location that you need for your solution. For more information, see Create a resource group.

   ```HTTP
   ```

```
PUT
https://management.azure.com/subscriptions/<YourSubscriptionId>/resourc
egroups/<YourResourceGroupName>?api-version=2020-06-01
```

With a request body like:

```JSON
{
 "location": "West US",
 "tags": {
   "tagname1": "tagvalue1"
 }
}
```

3. Before deploying your template, you can preview the changes the template will make to your environment. Use the what-if operation to verify that the template makes the changes that you expect. What-if also validates the template for errors.

4. To deploy a template, provide your subscription ID, the name of the resource group, the name of the deployment in the request URI.

```HTTP
PUT
https://management.azure.com/subscriptions/<YourSubscriptionId>/resourc
egroups/<YourResourceGroupName>/providers/Microsoft.Resources/deploymen
ts/<YourDeploymentName>?api-version=2020-10-01
```

In the request body, provide a link to your template and parameter file. For more information about the parameter file, see Create Resource Manager parameter file.

Notice the `mode` is set to **Incremental**. To run a complete deployment, set `mode` to **Complete**. Be careful when using the complete mode as you can inadvertently delete resources that aren't in your template.

```JSON
{
 "properties": {
   "templateLink": {
     "uri":
"http://mystorageaccount.blob.core.windows.net/templates/template.json"
,
     "contentVersion": "1.0.0.0"
   },
   "parametersLink": {
```

```
      "uri":
"http://mystorageaccount.blob.core.windows.net/templates/parameters.jso
n",
      "contentVersion": "1.0.0.0"
    },
    "mode": "Incremental"
  }
}
```

If you want to log response content, request content, or both, include `debugSetting` in the request.

JSON

```
{
  "properties": {
    "templateLink": {
      "uri":
"http://mystorageaccount.blob.core.windows.net/templates/template.json"
,
      "contentVersion": "1.0.0.0"
    },
    "parametersLink": {
      "uri":
"http://mystorageaccount.blob.core.windows.net/templates/parameters.jso
n",
      "contentVersion": "1.0.0.0"
    },
    "mode": "Incremental",
    "debugSetting": {
      "detailLevel": "requestContent, responseContent"
    }
  }
}
```

You can set up your storage account to use a shared access signature (SAS) token. For more information, see Delegate access with a shared access signature.

If you need to provide a sensitive value for a parameter (such as a password), add that value to a key vault. Retrieve the key vault during deployment as shown in the previous example. For more information, see Use Azure Key Vault to pass secure parameter value during deployment.

5. Instead of linking to files for the template and parameters, you can include them in the request body. The following example shows the request body with the template and parameter inline:

JSON

```
{
   "properties": {
   "mode": "Incremental",
   "template": {
     "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
     "contentVersion": "1.0.0.0",
     "parameters": {
       "storageAccountType": {
         "type": "string",
         "defaultValue": "Standard_LRS",
         "allowedValues": [
           "Standard_LRS",
           "Standard_GRS",
           "Standard_ZRS",
           "Premium_LRS"
         ],
         "metadata": {
           "description": "Storage Account type"
         }
       },
       "location": {
         "type": "string",
         "defaultValue": "[resourceGroup().location]",
         "metadata": {
           "description": "Location for all resources."
         }
       }
     },
     "variables": {
       "storageAccountName": "[concat(uniquestring(resourceGroup().id),
'standardsa')]"
     },
     "resources": [
       {
         "type": "Microsoft.Storage/storageAccounts",
         "apiVersion": "2018-02-01",
         "name": "[variables('storageAccountName')]",
         "location": "[parameters('location')]",
         "sku": {
           "name": "[parameters('storageAccountType')]"
         },
         "kind": "StorageV2",
         "properties": {}
       }
     ],
     "outputs": {
       "storageAccountName": {
         "type": "string",
         "value": "[variables('storageAccountName')]"
       }
     }
   },
   "parameters": {
```

```
      "location": {
        "value": "eastus2"
      }
    }
  }
}
```

6. To get the status of the template deployment, use Deployments - Get.

HTTP

```
GET
https://management.azure.com/subscriptions/{subscriptionId}/resourcegro
ups/{resourceGroupName}/providers/Microsoft.Resources/deployments/{depl
oymentName}?api-version=2020-10-01
```

Deploy with ARMClient

ARMClient is a simple command line tool to invoke the Azure Resource Manager API. To install the tool, see ARMClient.

To list your subscriptions:

Windows Command Prompt

```
armclient GET /subscriptions?api-version=2021-04-01
```

To list your resource groups:

Windows Command Prompt

```
armclient GET /subscriptions/<subscription-id>/resourceGroups?api-
version=2021-04-01
```

Replace **<subscription-id>** with your Azure subscription ID.

To create a resource group in the *Central US* region:

Windows Command Prompt

```
armclient PUT /subscriptions/<subscription-id>/resourceGroups/<resource-
group-name>?api-version=2021-04-01  "{location: 'central us', properties:
{}}"
```

Alternatively, you can put the body into a JSON file called **CreateRg.json**:

JSON

```
{
  "location": "Central US",
  "properties": { }
}
```

Windows Command Prompt

```
armclient PUT /subscriptions/<subscription-id>/resourceGroups/<resource-group-name>?api-version=2021-04-01 '@CreateRg.json'
```

For more information, see ARMClient: a command line tool for the Azure API .

Deployment name

You can give your deployment a name such as `ExampleDeployment`.

Every time you run a deployment, an entry is added to the resource group's deployment history with the deployment name. If you run another deployment and give it the same name, the earlier entry is replaced with the current deployment. If you want to maintain unique entries in the deployment history, give each deployment a unique name.

To create a unique name, you can assign a random number. Or, add a date value.

If you run concurrent deployments to the same resource group with the same deployment name, only the last deployment is completed. Any deployments with the same name that haven't finished are replaced by the last deployment. For example, if you run a deployment named `newStorage` that deploys a storage account named `storage1`, and at the same time run another deployment named `newStorage` that deploys a storage account named `storage2`, you deploy only one storage account. The resulting storage account is named `storage2`.

However, if you run a deployment named `newStorage` that deploys a storage account named `storage1`, and immediately after it completes you run another deployment named `newStorage` that deploys a storage account named `storage2`, then you have two storage accounts. One is named `storage1`, and the other is named `storage2`. But, you only have one entry in the deployment history.

When you specify a unique name for each deployment, you can run them concurrently without conflict. If you run a deployment named `newStorage1` that deploys a storage account named `storage1`, and at the same time run another deployment named

`newStorage2` that deploys a storage account named `storage2`, then you have two storage accounts and two entries in the deployment history.

To avoid conflicts with concurrent deployments and to ensure unique entries in the deployment history, give each deployment a unique name.

Next steps

- To roll back to a successful deployment when you get an error, see Rollback on error to successful deployment.
- 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 learn about handling asynchronous REST operations, see Track asynchronous Azure operations.
- To learn more about templates, see Understand the structure and syntax of ARM templates.

Deploy ARM templates from Azure Cloud Shell

Article • 04/09/2023

You can use Azure Cloud Shell to deploy an Azure Resource Manager template (ARM template). You can deploy either an ARM template that is stored remotely, or an ARM template that is stored on the local storage account for Cloud Shell.

You can deploy to any scope. This article shows deploying to a resource group.

Required permissions

To deploy a Bicep file or ARM template, you need write access on the resources you're deploying and access to all operations on the Microsoft.Resources/deployments resource type. For example, to deploy a virtual machine, you need `Microsoft.Compute/virtualMachines/write` and `Microsoft.Resources/deployments/*` permissions. The what-if operation has the same permission requirements.

For a list of roles and permissions, see Azure built-in roles.

Deploy remote template

To deploy an external template, provide the URI of the template exactly as you would for any external deployment. The external template could be in a GitHub repository or and an external storage account.

1. Open the Cloud Shell prompt.

2. To deploy the template, use the following commands:

Azure CLI

Azure CLI

```
az group create --name ExampleGroup --location "Central US"
az deployment group create \
  --name ExampleDeployment \
  --resource-group ExampleGroup \
```

```
  --template-uri "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.storage/storage-account-create/azuredeploy.json" \
  --parameters storageAccountType=Standard_GRS
```

Deploy local template

To deploy a local template, you must first upload your template to the storage account that is connected to your Cloud Shell session.

1. Sign in to the Cloud Shell.

2. Select either **PowerShell** or **Bash**.

3. Select **Upload/Download files**, and then select **Upload**.

4. Select the ARM template you want to upload, and then select **Open**.

5. To deploy the template, use the following commands:

Azure CLI

Azure CLI

```
az group create --name ExampleGroup --location "South Central US"
az deployment group create \
  --resource-group ExampleGroup \
```

```
    --template-file azuredeploy.json \
    --parameters storageAccountType=Standard_GRS
```

Next steps

- For more information about deployment commands, see Deploy resources with ARM templates and Azure CLI and Deploy resources with ARM templates and Azure PowerShell.
- To preview changes before deploying a template, see ARM template deployment what-if operation.

Use a deployment button to deploy remote templates

Article • 04/09/2023

This article describes how to use the **Deploy to Azure** button to deploy remote ARM JSON templates from a GitHub repository or an Azure storage account. You can add the button directly to the *README.md* file in your GitHub repository. Or, you can add the button to a web page that references the repository. This method doesn't support deploying remote Bicep files.

The deployment scope is determined by the template schema. For more information, see:

- resource groups
- subscriptions
- management groups
- tenants

Required permissions

To deploy a Bicep file or ARM template, you need write access on the resources you're deploying and access to all operations on the Microsoft.Resources/deployments resource type. For example, to deploy a virtual machine, you need `Microsoft.Compute/virtualMachines/write` and `Microsoft.Resources/deployments/*` permissions. The what-if operation has the same permission requirements.

For a list of roles and permissions, see Azure built-in roles.

Use common image

To add the button to your web page or repository, use the following image:

markdown

```markdown
![Deploy to Azure](https://aka.ms/deploytoazurebutton)
```

HTML

```html
<img src="https://aka.ms/deploytoazurebutton"/>
```

The image appears as:

Create URL for deploying template

This section shows how to get the URLs for the templates stored in GitHub and Azure storage account, and how to format the URLs.

Template stored in GitHub

To create the URL for your template, start with the raw URL to the template in your GitHub repo. To see the raw URL, select **Raw**.

The format of the URL is:

HTML

```
https://raw.githubusercontent.com/Azure/azure-quickstart-
templates/master/quickstarts/microsoft.storage/storage-account-
create/azuredeploy.json
```

> **Note**
>
> To deploy a template or reference a linked template that is stored in a private GitHub repo, see a custom solution documented in **Creating a Custom and Secure Azure Portal Offering**. You can create an **Azure function** that pulls the GitHub token out of Azure Key Vault.

If you're using Git with Azure Repos instead of a GitHub repo, you can still use the **Deploy to Azure** button. Make sure your repo is public. Use the Items operation to get the template. Your request should be in the following format:

HTTP

```
https://dev.azure.com/{organization-name}/{project-name}/_apis/git/repositories/{repository-name}/items?scopePath={url-encoded-path}&api-version=6.0
```

Template stored in Azure storage account

The format of the URLs for the templates stored in a public container is:

HTML

```
https://{storage-account-name}.blob.core.windows.net/{container-name}/{template-file-name}
```

For example:

HTML

```
https://demostorage0215.blob.core.windows.net/democontainer/azuredeploy.json
```

You can secure the template with SAS token. For more information, see How to deploy private ARM template with SAS token. The following url is an example with SAS token:

HTML

```
https://demostorage0215.blob.core.windows.net/privatecontainer/azuredeploy.json?sv=2019-07-07&sr=b&sig=rnI8%2FvKoCHmvmP7XvfspfyzdHjtN4GPsSqB8qMI9FAo%3D&se=2022-02-16T17%3A47%3A46Z&sp=r
```

Format the URL

Once you have the URL, you need to convert the URL to a URL-encoded value. You can use an online encoder or run a command. The following PowerShell example shows how to URL encode a value.

PowerShell

```
$url = "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.storage/storage-account-create/azuredeploy.json"
[uri]::EscapeDataString($url)
```

The example URL has the following value when URL encoded.

```HTML
https%3A%2F%2Fraw.githubusercontent.com%2FAzure%2Fazure-quickstart-templates%2Fmaster%2Fquickstarts%2Fmicrosoft.storage%2Fstorage-account-create%2Fazuredeploy.json
```

Each link starts with the same base URL:

```HTML
https://portal.azure.com/#create/Microsoft.Template/uri/
```

Add your URL-encoded template link to the end of the base URL.

```HTML
https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2Fraw.githubusercontent.com%2FAzure%2Fazure-quickstart-templates%2Fmaster%2Fquickstarts%2Fmicrosoft.storage%2Fstorage-account-create%2Fazuredeploy.json
```

You have your full URL for the link.

Create Deploy to Azure button

Finally, put the link and image together.

To add the button with Markdown in the *README.md* file in your GitHub repository or a web page, use:

```markdown
[![Deploy to Azure](https://aka.ms/deploytoazurebutton)](https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2Fraw.githubusercontent.com%2FAzure%2Fazure-quickstart-templates%2Fmaster%2Fquickstarts%2Fmicrosoft.storage%2Fstorage-account-create%2Fazuredeploy.json)
```

For HTML, use:

```HTML
```

```
<a
href="https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2F
raw.githubusercontent.com%2FAzure%2Fazure-quickstart-
templates%2Fmaster%2Fquickstarts%2Fmicrosoft.storage%2Fstorage-account-
create%2Fazuredeploy.json" target="_blank">
  <img src="https://aka.ms/deploytoazurebutton"/>
</a>
```

For Git with Azure repo, the button is in the format:

```markdown
[![Deploy to Azure](https://aka.ms/deploytoazurebutton)]
(https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2Fdev.a
zure.com%2Forgname%2Fprojectname%2F_apis%2Fgit%2Frepositories%2Freponame%2Fi
tems%3FscopePath%3D%2freponame%2fazuredeploy.json%26api-version%3D6.0)
```

Deploy the template

To test the full solution, select the following button:

Deploy to Azure

The portal displays a pane that allows you to easily provide parameter values. The parameters are pre-filled with the default values from the template. The camel-cased parameter name, *storageAccountType*, defined in the template is turned into a space-separated string when displayed on the portal.

Next steps

- To learn more about templates, see Understand the structure and syntax of ARM templates.

Resource group deployments with ARM templates

Article • 03/09/2023

This article describes how to scope your deployment to a resource group. You use an Azure Resource Manager template (ARM template) for the deployment. The article also shows how to expand the scope beyond the resource group in the deployment operation.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **resource group deployments**.

Supported resources

Most resources can be deployed to a resource group. For a list of available resources, see ARM template reference.

Schema

For templates, use the following schema:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  ...
}
```

For parameter files, use:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  ...
}
```

Deployment commands

To deploy to a resource group, use the resource group deployment commands.

Azure CLI

For Azure CLI, use az deployment group create. The following example deploys a template to create a resource group. The resource group you specify in the `--resource-group` parameter is the **target resource group**.

Azure CLI

```
az deployment group create \
  --name demoRGDeployment \
  --resource-group ExampleGroup \
  --template-uri "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.storage/storage-account-create/azuredeploy.json" \
  --parameters storageAccountType=Standard_GRS
```

For more detailed information about deployment commands and options for deploying ARM templates, see:

- Deploy resources with ARM templates and Azure portal
- Deploy resources with ARM templates and Azure CLI
- Deploy resources with ARM templates and Azure PowerShell
- Deploy resources with ARM templates and Azure Resource Manager REST API
- Use a deployment button to deploy templates from GitHub repository
- Deploy ARM templates from Cloud Shell

Deployment scopes

When deploying to a resource group, you can deploy resources to:

- the target resource group from the operation
- other resource groups in the same subscription or other subscriptions
- any subscription in the tenant
- the tenant for the resource group

An extension resource can be scoped to a target that is different than the deployment target.

The user deploying the template must have access to the specified scope.

This section shows how to specify different scopes. You can combine these different scopes in a single template.

Scope to target resource group

To deploy resources to the target resource, add those resources to the resources section of the template.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
    resource-group-resources
  ],
  "outputs": {}
}
```

For an example template, see Deploy to target resource group.

Scope to resource group in same subscription

To deploy resources to a different resource group in the same subscription, add a nested deployment and include the `resourceGroup` property. If you don't specify the subscription ID or resource group, the subscription and resource group from the parent template are used. All the resource groups must exist before running the deployment.

In the following example, the nested deployment targets a resource group named `demoResourceGroup`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "nestedDeployment",
      "resourceGroup": "demoResourceGroup",
      "properties": {
        "mode": "Incremental",
        "template": {
```

```
            resource-group-resources
          }
        }
      }
    ],
    "outputs": {}
  }
```

For an example template, see Deploy to multiple resource groups.

Scope to resource group in different subscription

To deploy resources to a resource group in a different subscription, add a nested deployment and include the `subscriptionId` and `resourceGroup` properties. In the following example, the nested deployment targets a resource group named `demoResourceGroup`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "nestedDeployment",
      "subscriptionId": "00000000-0000-0000-0000-000000000000",
      "resourceGroup": "demoResourceGroup",
      "properties": {
        "mode": "Incremental",
        "template": {
          resource-group-resources
        }
      }
    }
  ],
  "outputs": {}
}
```

For an example template, see Deploy to multiple resource groups.

Scope to subscription

To deploy resources to a subscription, add a nested deployment and include the `subscriptionId` property. The subscription can be the subscription for the target

resource group, or any other subscription in the tenant. Also, set the `location` property for the nested deployment.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "nestedDeployment",
      "location": "centralus",
      "subscriptionId": "0000000-0000-0000-0000-000000000000",
      "properties": {
        "mode": "Incremental",
        "template": {
          subscription-resources
        }
      }
    }
  ],
  "outputs": {}
}
```

For an example template, see Create resource group.

Scope to tenant

To create resources at the tenant, set the `scope` to `/`. The user deploying the template must have the required access to deploy at the tenant.

To use a nested deployment, set `scope` and `location`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "nestedDeployment",
      "location": "centralus",
      "scope": "/",
      "properties": {
```

```
          "mode": "Incremental",
          "template": {
            tenant-resources
          }
        }
      }
    ],
    "outputs": {}
  }
```

Or, you can set the scope to `/` for some resource types, like management groups.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "mgName": {
      "type": "string",
      "defaultValue": "[concat('mg-', uniqueString(newGuid()))]"
    }
  },
  "resources": [
    {
      "type": "Microsoft.Management/managementGroups",
      "apiVersion": "2021-04-01",
      "name": "[parameters('mgName')]",
      "scope": "/",
      "location": "eastus",
      "properties": {}
    }
  ],
  "outputs": {
    "output": {
      "type": "string",
      "value": "[parameters('mgName')]"
    }
  }
}
```

For more information, see Management group.

Deploy to target resource group

To deploy resources in the target resource group, define those resources in the `resources` section of the template. The following template creates a storage account in the resource group that is specified in the deployment operation.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storagePrefix": {
      "type": "string",
      "minLength": 3,
      "maxLength": 11
    },
    "storageSKU": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_RAGRS",
        "Standard_ZRS",
        "Premium_LRS",
        "Premium_ZRS",
        "Standard_GZRS",
        "Standard_RAGZRS"
      ]
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    }
  },
  "variables": {
    "uniqueStorageName": "[concat(parameters('storagePrefix'),
uniqueString(resourceGroup().id))]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-09-01",
      "name": "[variables('uniqueStorageName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('storageSKU')]"
      },
      "kind": "StorageV2",
      "properties": {
        "supportsHttpsTrafficOnly": true
      }
    }
  ],
  "outputs": {
    "storageEndpoint": {
      "type": "object",
      "value": "
```

```
[reference(variables('uniqueStorageName')).primaryEndpoints]"
    }
  }
}
```

Deploy to multiple resource groups

You can deploy to more than one resource group in a single ARM template. To target a resource group that is different than the one for parent template, use a nested or linked template. Within the deployment resource type, specify values for the subscription ID and resource group that you want the nested template to deploy to. The resource groups can exist in different subscriptions.

> **Note**
>
> You can deploy to **800 resource groups** in a single deployment. Typically, this limitation means you can deploy to one resource group specified for the parent template, and up to 799 resource groups in nested or linked deployments. However, if your parent template contains only nested or linked templates and does not itself deploy any resources, then you can include up to 800 resource groups in nested or linked deployments.

The following example deploys two storage accounts. The first storage account is deployed to the resource group specified in the deployment operation. The second storage account is deployed to the resource group specified in the `secondResourceGroup` and `secondSubscriptionID` parameters:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storagePrefix": {
      "type": "string",
      "maxLength": 11
    },
    "secondResourceGroup": {
      "type": "string"
    },
    "secondSubscriptionID": {
      "type": "string",
      "defaultValue": ""
    },
```

```
    "secondStorageLocation": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    }
  },
  "variables": {
    "firstStorageName": "[concat(parameters('storagePrefix'),
uniqueString(resourceGroup().id))]",
    "secondStorageName": "[concat(parameters('storagePrefix'),
uniqueString(parameters('secondSubscriptionID'),
parameters('secondResourceGroup')))]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-04-01",
      "name": "[variables('firstStorageName')]",
      "location": "[resourceGroup().location]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "Storage",
      "properties": {
      }
    },
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "nestedTemplate",
      "resourceGroup": "[parameters('secondResourceGroup')]",
      "subscriptionId": "[parameters('secondSubscriptionID')]",
      "properties": {
        "mode": "Incremental",
        "template": {
          "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
          "contentVersion": "1.0.0.0",
          "parameters": {},
          "variables": {},
          "resources": [
            {
              "type": "Microsoft.Storage/storageAccounts",
              "apiVersion": "2021-04-01",
              "name": "[variables('secondStorageName')]",
              "location": "[parameters('secondStorageLocation')]",
              "sku": {
                "name": "Standard_LRS"
              },
              "kind": "Storage",
              "properties": {
              }
            }
          ]
        },
        "parameters": {}
```

```
      }
    }
  ]
}
```

If you set `resourceGroup` to the name of a resource group that doesn't exist, the deployment fails.

To test the preceding template and see the results, use PowerShell or Azure CLI.

Azure CLI

To deploy two storage accounts to two resource groups in the **same subscription**, use:

Azure CLI

```
firstRG="primarygroup"
secondRG="secondarygroup"

az group create --name $firstRG --location southcentralus
az group create --name $secondRG --location eastus
az deployment group create \
  --name ExampleDeployment \
  --resource-group $firstRG \
  --template-uri https://raw.githubusercontent.com/Azure/azure-docs-
json-samples/master/azure-resource-manager/crosssubscription.json \
  --parameters storagePrefix=tfstorage secondResourceGroup=$secondRG
secondStorageLocation=eastus
```

To deploy two storage accounts to **two subscriptions**, use:

Azure CLI

```
firstRG="primarygroup"
secondRG="secondarygroup"

firstSub="<first-subscription-id>"
secondSub="<second-subscription-id>"

az account set --subscription $secondSub
az group create --name $secondRG --location eastus

az account set --subscription $firstSub
az group create --name $firstRG --location southcentralus

az deployment group create \
  --name ExampleDeployment \
  --resource-group $firstRG \
```

```
  --template-uri https://raw.githubusercontent.com/Azure/azure-docs-
json-samples/master/azure-resource-manager/crosssubscription.json \
  --parameters storagePrefix=storage secondResourceGroup=$secondRG
secondStorageLocation=eastus secondSubscriptionID=$secondSub
```

Create resource group

From a resource group deployment, you can switch to the level of a subscription and create a resource group. The following template deploys a storage account to the target resource group, and creates a new resource group in the specified subscription.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storagePrefix": {
      "type": "string",
      "maxLength": 11
    },
    "newResourceGroupName": {
      "type": "string"
    },
    "nestedSubscriptionID": {
      "type": "string"
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    }
  },
  "variables": {
    "storageName": "[concat(parameters('storagePrefix'),
uniqueString(resourceGroup().id))]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-04-01",
      "name": "[variables('storageName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "Storage",
      "properties": {
      }
    },
```

```
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "demoSubDeployment",
      "location": "westus",
      "subscriptionId": "[parameters('nestedSubscriptionID')]",
      "properties": {
        "mode": "Incremental",
        "template": {
          "$schema": "https://schema.management.azure.com/schemas/2018-05-
01/subscriptionDeploymentTemplate.json#",
          "contentVersion": "1.0.0.0",
          "parameters": {},
          "variables": {},
          "resources": [
            {
              "type": "Microsoft.Resources/resourceGroups",
              "apiVersion": "2021-04-01",
              "name": "[parameters('newResourceGroupName')]",
              "location": "[parameters('location')]",
              "properties": {}
            }
          ],
          "outputs": {}
        }
      }
    }
  ]
}
```

Next steps

- For an example of deploying workspace settings for Microsoft Defender for Cloud, see deployASCwithWorkspaceSettings.json .

Subscription deployments with ARM templates

Article • 04/09/2023

To simplify the management of resources, you can use an Azure Resource Manager template (ARM template) to deploy resources at the level of your Azure subscription. For example, you can deploy policies and Azure role-based access control (Azure RBAC) to your subscription, which applies them across your subscription. You can also create resource groups within the subscription and deploy resources to resource groups in the subscription.

> **Note**
>
> You can deploy to 800 different resource groups in a subscription level deployment.

To deploy templates at the subscription level, use Azure CLI, PowerShell, REST API, or the portal.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **subscription deployments**.

Supported resources

Not all resource types can be deployed to the subscription level. This section lists which resource types are supported.

For Azure Blueprints, use:

- artifacts
- blueprints
- blueprintAssignments
- versions (Blueprints)

For Azure Policies, use:

- policyAssignments
- policyDefinitions

- policySetDefinitions
- remediations

For access control, use:

- accessReviewScheduleDefinitions
- accessReviewScheduleSettings
- roleAssignments
- roleAssignmentScheduleRequests
- roleDefinitions
- roleEligibilityScheduleRequests
- roleManagementPolicyAssignments

For nested templates that deploy to resource groups, use:

- deployments

For creating new resource groups, use:

- resourceGroups

For managing your subscription, use:

- budgets
- configurations - Advisor
- lineOfCredit
- locks
- profile - Change Analysis
- supportPlanTypes
- tags

For monitoring, use:

- diagnosticSettings
- logprofiles

For security, use:

- advancedThreatProtectionSettings
- alertsSuppressionRules
- assessmentMetadata
- assessments
- autoProvisioningSettings
- connectors
- deviceSecurityGroups

- ingestionSettings
- pricings
- securityContacts
- settings
- workspaceSettings

Other supported types include:

- scopeAssignments
- eventSubscriptions
- peerAsns

Schema

The schema you use for subscription-level deployments is different than the schema for resource group deployments.

For templates, use:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2018-05-01/subscriptionDeploymentTemplate.json#",
  ...
}
```

The schema for a parameter file is the same for all deployment scopes. For parameter files, use:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  ...
}
```

Deployment commands

To deploy to a subscription, use the subscription-level deployment commands.

Azure CLI

For Azure CLI, use az deployment sub create. The following example deploys a template to create a resource group:

Azure CLI

```
az deployment sub create \
  --name demoSubDeployment \
  --location centralus \
  --template-uri "https://raw.githubusercontent.com/Azure/azure-docs-
json-samples/master/azure-resource-manager/emptyrg.json" \
  --parameters rgName=demoResourceGroup rgLocation=centralus
```

For more detailed information about deployment commands and options for deploying ARM templates, see:

- Deploy resources with ARM templates and Azure portal
- Deploy resources with ARM templates and Azure CLI
- Deploy resources with ARM templates and Azure PowerShell
- Deploy resources with ARM templates and Azure Resource Manager REST API
- Use a deployment button to deploy templates from GitHub repository
- Deploy ARM templates from Cloud Shell

Deployment location and name

For subscription level deployments, you must provide a location for the deployment. The location of the deployment is separate from the location of the resources you deploy. The deployment location specifies where to store deployment data. Management group and tenant deployments also require a location. For resource group deployments, the location of the resource group is used to store the deployment data.

You can provide a name for the deployment, or use the default deployment name. The default name is the name of the template file. For example, deploying a template named *azuredeploy.json* creates a default deployment name of **azuredeploy**.

For each deployment name, the location is immutable. You can't create a deployment in one location when there's an existing deployment with the same name in a different location. For example, if you create a subscription deployment with the name **deployment1** in **centralus**, you can't later create another deployment with the name **deployment1** but a location of **westus**. If you get the error code `InvalidDeploymentLocation`, either use a different name or the same location as the previous deployment for that name.

Deployment scopes

When deploying to a subscription, you can deploy resources to:

- the target subscription from the operation
- any subscription in the tenant
- resource groups within the subscription or other subscriptions
- the tenant for the subscription

An extension resource can be scoped to a target that is different than the deployment target.

The user deploying the template must have access to the specified scope.

This section shows how to specify different scopes. You can combine these different scopes in a single template.

Scope to target subscription

To deploy resources to the target subscription, add those resources to the resources section of the template.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2018-05-01/subscriptionDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
    subscription-level-resources
  ],
  "outputs": {}
}
```

For examples of deploying to the subscription, see Create resource groups and Assign policy definition.

Scope to other subscription

To deploy resources to a subscription that is different than the subscription from the operation, add a nested deployment. Set the `subscriptionId` property to the ID of the subscription you want to deploy to. Set the `location` property for the nested deployment.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2018-05-01/subscriptionDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "nestedDeployment",
      "subscriptionId": "00000000-0000-0000-0000-000000000000",
      "location": "westus",
      "properties": {
        "mode": "Incremental",
        "template": {
          subscription-resources
        }
      }
    }
  ],
  "outputs": {}
}
```

Scope to resource group

To deploy resources to a resource group within the subscription, add a nested deployment and include the `resourceGroup` property. In the following example, the nested deployment targets a resource group named `demoResourceGroup`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2018-05-01/subscriptionDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "nestedDeployment",
      "resourceGroup": "demoResourceGroup",
      "properties": {
        "mode": "Incremental",
        "template": {
          resource-group-resources
        }
      }
    }
  ],
```

```
  "outputs": {}
}
```

For an example of deploying to a resource group, see Create resource group and resources.

Scope to tenant

To create resources at the tenant, set the `scope` to `/`. The user deploying the template must have the required access to deploy at the tenant.

To use a nested deployment, set `scope` and `location`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2018-05-01/subscriptionDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "nestedDeployment",
      "location": "centralus",
      "scope": "/",
      "properties": {
        "mode": "Incremental",
        "template": {
          tenant-resources
        }
      }
    }
  ],
  "outputs": {}
}
```

Or, you can set the scope to `/` for some resource types, like management groups.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2018-05-01/subscriptionDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "mgName": {
      "type": "string",
      "defaultValue": "[concat('mg-', uniqueString(newGuid()))]"
```

```
    }
  },
  "resources": [
    {
      "type": "Microsoft.Management/managementGroups",
      "apiVersion": "2021-04-01",
      "name": "[parameters('mgName')]",
      "scope": "/",
      "location": "eastus",
      "properties": {}
    }
  ],
  "outputs": {
    "output": {
      "type": "string",
      "value": "[parameters('mgName')]"
    }
  }
}
```

For more information, see Management group.

Resource groups

Create resource groups

To create a resource group in an ARM template, define a Microsoft.Resources/resourceGroups resource with a name and location for the resource group.

The following template creates an empty resource group.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2018-05-01/subscriptionDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "rgName": {
      "type": "string"
    },
    "rgLocation": {
      "type": "string"
    }
  },
  "variables": {},
  "resources": [
    {
```

```
      "type": "Microsoft.Resources/resourceGroups",
      "apiVersion": "2021-04-01",
      "name": "[parameters('rgName')]",
      "location": "[parameters('rgLocation')]",
      "properties": {}
    }
  ],
  "outputs": {}
}
```

Use the copy element with resource groups to create more than one resource group.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2018-05-
01/subscriptionDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "rgNamePrefix": {
      "type": "string"
    },
    "rgLocation": {
      "type": "string"
    },
    "instanceCount": {
      "type": "int"
    }
  },
  "variables": {},
  "resources": [
    {
      "type": "Microsoft.Resources/resourceGroups",
      "apiVersion": "2021-04-01",
      "location": "[parameters('rgLocation')]",
      "name": "[concat(parameters('rgNamePrefix'), copyIndex())]",
      "copy": {
        "name": "rgCopy",
        "count": "[parameters('instanceCount')]"
      },
      "properties": {}
    }
  ],
  "outputs": {}
}
```

For information about resource iteration, see Resource iteration in ARM templates, and Tutorial: Create multiple resource instances with ARM templates.

Create resource group and resources

To create the resource group and deploy resources to it, use a nested template. The nested template defines the resources to deploy to the resource group. Set the nested template as dependent on the resource group to make sure the resource group exists before deploying the resources. You can deploy to up to 800 resource groups.

The following example creates a resource group, and deploys a storage account to the resource group.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2018-05-
01/subscriptionDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "rgName": {
      "type": "string"
    },
    "rgLocation": {
      "type": "string"
    },
    "storagePrefix": {
      "type": "string",
      "maxLength": 11
    }
  },
  "variables": {
    "storageName": "[concat(parameters('storagePrefix'),
uniqueString(subscription().id, parameters('rgName')))]"
  },
  "resources": [
    {
      "type": "Microsoft.Resources/resourceGroups",
      "apiVersion": "2021-04-01",
      "name": "[parameters('rgName')]",
      "location": "[parameters('rgLocation')]",
      "properties": {}
    },
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "storageDeployment",
      "resourceGroup": "[parameters('rgName')]",
      "dependsOn": [
        "[resourceId('Microsoft.Resources/resourceGroups/',
parameters('rgName'))]"
      ],
      "properties": {
        "mode": "Incremental",
        "template": {
          "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
```

```
          "contentVersion": "1.0.0.0",
          "parameters": {},
          "variables": {},
          "resources": [
            {
              "type": "Microsoft.Storage/storageAccounts",
              "apiVersion": "2021-04-01",
              "name": "[variables('storageName')]",
              "location": "[parameters('rgLocation')]",
              "sku": {
                "name": "Standard_LRS"
              },
              "kind": "StorageV2"
            }
          ],
          "outputs": {}
        }
      }
    }
  ],
  "outputs": {}
}
```

Azure Policy

Assign policy definition

The following example assigns an existing policy definition to the subscription. If the policy definition takes parameters, provide them as an object. If the policy definition doesn't take parameters, use the default empty object.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2018-05-01/subscriptionDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "policyDefinitionID": {
      "type": "string"
    },
    "policyName": {
      "type": "string"
    },
    "policyParameters": {
      "type": "object",
      "defaultValue": {}
    }
  },
  "variables": {},
```

```
  "resources": [
    {
      "type": "Microsoft.Authorization/policyAssignments",
      "apiVersion": "2020-03-01",
      "name": "[parameters('policyName')]",
      "properties": {
        "scope": "[subscription().id]",
        "policyDefinitionId": "[parameters('policyDefinitionID')]",
        "parameters": "[parameters('policyParameters')]"
      }
    }
  ]
}
```

To deploy this template with Azure CLI, use:

Azure CLI

```
# Built-in policy definition that accepts parameters
definition=$(az policy definition list --query "[?displayName=='Allowed
locations'].id" --output tsv)

az deployment sub create \
  --name demoDeployment \
  --location centralus \
  --template-uri "https://raw.githubusercontent.com/Azure/azure-docs-json-
samples/master/azure-resource-manager/policyassign.json" \
  --parameters policyDefinitionID=$definition policyName=setLocation
policyParameters="{'listOfAllowedLocations': {'value': ['westus']} }"
```

To deploy this template with PowerShell, use:

Azure PowerShell

```
$definition = Get-AzPolicyDefinition | Where-Object {
$_.Properties.DisplayName -eq 'Allowed locations' }

$locations = @("westus", "westus2")
$policyParams =@{listOfAllowedLocations = @{ value = $locations}}

New-AzSubscriptionDeployment `
  -Name policyassign `
  -Location centralus `
  -TemplateUri "https://raw.githubusercontent.com/Azure/azure-docs-json-
samples/master/azure-resource-manager/policyassign.json" `
  -policyDefinitionID $definition.PolicyDefinitionId `
  -policyName setLocation `
  -policyParameters $policyParams
```

Create and assign policy definitions

You can define and assign a policy definition in the same template.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2018-05-
01/subscriptionDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {},
  "variables": {},
  "resources": [
    {
      "type": "Microsoft.Authorization/policyDefinitions",
      "apiVersion": "2020-03-01",
      "name": "locationpolicy",
      "properties": {
        "policyType": "Custom",
        "parameters": {},
        "policyRule": {
          "if": {
            "field": "location",
            "equals": "northeurope"
          },
          "then": {
            "effect": "deny"
          }
        }
      }
    },
    {
      "type": "Microsoft.Authorization/policyAssignments",
      "apiVersion": "2020-03-01",
      "name": "location-lock",
      "dependsOn": [
        "locationpolicy"
      ],
      "properties": {
        "scope": "[subscription().id]",
        "policyDefinitionId": "
[subscriptionResourceId('Microsoft.Authorization/policyDefinitions',
'locationpolicy')]"
      }
    }
  ]
}
```

To create the policy definition in your subscription, and assign it to the subscription, use the following CLI command:

Azure CLI

```
az deployment sub create \
  --name demoDeployment \
  --location centralus \
  --template-uri "https://raw.githubusercontent.com/Azure/azure-docs-json-samples/master/azure-resource-manager/policydefineandassign.json"
```

To deploy this template with PowerShell, use:

Azure PowerShell

```
New-AzSubscriptionDeployment `
  -Name definePolicy `
  -Location centralus `
  -TemplateUri "https://raw.githubusercontent.com/Azure/azure-docs-json-samples/master/azure-resource-manager/policydefineandassign.json"
```

Azure Blueprints

Create blueprint definition

You can create a blueprint definition from a template.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2018-05-01/subscriptionDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "blueprintName": {
      "defaultValue": "sample-blueprint",
      "type": "String",
      "metadata": {
        "description": "The name of the blueprint definition."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Blueprint/blueprints",
      "apiVersion": "2018-11-01-preview",
      "name": "[parameters('blueprintName')]",
      "properties": {
        "targetScope": "subscription",
        "description": "Blueprint with a policy assignment artifact.",
        "resourceGroups": {
          "sampleRg": {
            "description": "Resource group to add the assignment to."
```

```
          }
        },
        "parameters": {
          "listOfResourceTypesNotAllowed": {
            "type": "array",
            "metadata": {
              "displayName": "Resource types to pass to the policy
assignment artifact."
            },
            "defaultValue": [
              "Citrix.Cloud/accounts"
            ]
          }
        }
      }
    },
    {
      "type": "Microsoft.Blueprint/blueprints/artifacts",
      "apiVersion": "2018-11-01-preview",
      "name": "[concat(parameters('blueprintName'), '/policyArtifact')]",
      "kind": "policyAssignment",
      "dependsOn": [
        "[parameters('blueprintName')]"
      ],
      "properties": {
        "displayName": "Blocked Resource Types policy definition",
        "description": "Block certain resource types",
        "policyDefinitionId": "
[tenantResourceId('Microsoft.Authorization/policyDefinitions', '6c112d4e-
5bc7-47ae-a041-ea2d9dccd749')]",
        "resourceGroup": "sampleRg",
        "parameters": {
          "listOfResourceTypesNotAllowed": {
            "value": "[[parameters('listOfResourceTypesNotAllowed')]"
          }
        }
      }
    }
  ]
}
```

To create the blueprint definition in your subscription, use the following CLI command:

Azure CLI

```
az deployment sub create \
  --name demoDeployment \
  --location centralus \
  --template-uri "https://raw.githubusercontent.com/Azure/azure-quickstart-
templates/master/subscription-deployments/blueprints-new-
blueprint/azuredeploy.json"
```

To deploy this template with PowerShell, use:

Azure PowerShell

```
New-AzSubscriptionDeployment `
  -Name demoDeployment `
  -Location centralus `
  -TemplateUri "https://raw.githubusercontent.com/Azure/azure-quickstart-
templates/master/subscription-deployments/blueprints-new-
blueprint/azuredeploy.json"
```

Access control

To learn about assigning roles, see Assign Azure roles using Azure Resource Manager templates.

The following example creates a resource group, applies a lock to it, and assigns a role to a principal.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2018-05-
01/subscriptionDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "16815708176905569328"
    }
  },
  "parameters": {
    "rgName": {
      "type": "string",
      "metadata": {
        "description": "Name of the resourceGroup to create"
      }
    },
    "rgLocation": {
      "type": "string",
      "metadata": {
        "description": "Location for the resourceGroup"
      }
    },
    "principalId": {
      "type": "string",
      "metadata": {
        "description": "principalId of the user that will be given
contributor access to the resourceGroup"
```

```
      }
    },
    "roleDefinitionId": {
      "type": "string",
      "defaultValue": "b24988ac-6180-42a0-ab88-20f7382dd24c",
      "metadata": {
        "description": "roleDefinition to apply to the resourceGroup -
default is contributor"
      }
    },
    "roleAssignmentName": {
      "type": "string",
      "defaultValue": "[guid(parameters('principalId'),
parameters('roleDefinitionId'), parameters('rgName'))]",
      "metadata": {
        "description": "Unique name for the roleAssignment in the format of
a guid"
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Resources/resourceGroups",
      "apiVersion": "2019-10-01",
      "name": "[parameters('rgName')]",
      "location": "[parameters('rgLocation')]",
      "tags": {
        "Note": "subscription level deployment"
      },
      "properties": {}
    },
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2020-10-01",
      "name": "applyLock",
      "resourceGroup": "[parameters('rgName')]",
      "properties": {
        "expressionEvaluationOptions": {
          "scope": "inner"
        },
        "mode": "Incremental",
        "parameters": {
          "principalId": {
            "value": "[parameters('principalId')]"
          },
          "roleDefinitionId": {
            "value": "[parameters('roleDefinitionId')]"
          },
          "roleAssignmentName": {
            "value": "[parameters('roleAssignmentName')]"
          }
        },
        "template": {
          "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
```

```
          "contentVersion": "1.0.0.0",
          "metadata": {
            "_generator": {
              "name": "bicep",
              "version": "0.5.6.12127",
              "templateHash": "6034226420560042393"
            }
          },
          "parameters": {
            "principalId": {
              "type": "string",
              "metadata": {
                "description": "principalId of the user that will be given
contributor access to the resourceGroup"
              }
            },
            "roleDefinitionId": {
              "type": "string",
              "metadata": {
                "description": "roleDefinition to apply to the resourceGroup
- default is contributor"
              }
            },
            "roleAssignmentName": {
              "type": "string",
              "metadata": {
                "description": "Unique name for the roleAssignment in the
format of a guid"
              }
            }
          },
          "resources": [
            {
              "type": "Microsoft.Authorization/locks",
              "apiVersion": "2016-09-01",
              "name": "DontDelete",
              "properties": {
                "level": "CanNotDelete",
                "notes": "Prevent deletion of the resourceGroup"
              }
            },
            {
              "type": "Microsoft.Authorization/roleAssignments",
              "apiVersion": "2020-04-01-preview",
              "name": "[guid(parameters('roleAssignmentName'))]",
              "properties": {
                "roleDefinitionId": "
[subscriptionResourceId('Microsoft.Authorization/roleDefinitions',
parameters('roleDefinitionId'))]",
                "principalId": "[parameters('principalId')]"
              }
            }
          ]
        }
      },
```

```
      "dependsOn": [
        "[subscriptionResourceId('Microsoft.Resources/resourceGroups',
parameters('rgName'))]"
      ]
    }
  ]
}
```

Next steps

- For an example of deploying workspace settings for Microsoft Defender for Cloud, see deployASCwithWorkspaceSettings.json.
- Sample templates can be found at GitHub.
- You can also deploy templates at management group level and tenant level.

Management group deployments with ARM templates

Article • 04/09/2023

As your organization matures, you can deploy an Azure Resource Manager template (ARM template) to create resources at the management group level. For example, you may need to define and assign policies or Azure role-based access control (Azure RBAC) for a management group. With management group level templates, you can declaratively apply policies and assign roles at the management group level.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **management group deployments**.

Supported resources

Not all resource types can be deployed to the management group level. This section lists which resource types are supported.

For Azure Blueprints, use:

- artifacts
- blueprints
- blueprintAssignments
- versions

For Azure Policy, use:

- policyAssignments
- policyDefinitions
- policySetDefinitions
- remediations

For access control, use:

- privateLinkAssociations
- roleAssignments
- roleAssignmentScheduleRequests

- roleDefinitions
- roleEligibilityScheduleRequests
- roleManagementPolicyAssignments

For nested templates that deploy to subscriptions or resource groups, use:

- deployments

For managing your resources, use:

- diagnosticSettings
- tags

Management groups are tenant-level resources. However, you can create management groups in a management group deployment by setting the scope of the new management group to the tenant. See Management group.

Schema

The schema you use for management group deployments is different than the schema for resource group deployments.

For templates, use:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-08-01/managementGroupDeploymentTemplate.json#",
  ...
}
```

The schema for a parameter file is the same for all deployment scopes. For parameter files, use:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  ...
}
```

Deployment commands

To deploy to a management group, use the management group deployment commands.

Azure CLI

For Azure CLI, use az deployment mg create:

Azure CLI

```
az deployment mg create \
  --name demoMGDeployment \
  --location WestUS \
  --management-group-id myMG \
  --template-uri "https://raw.githubusercontent.com/Azure/azure-docs-
json-samples/master/management-level-deployment/azuredeploy.json"
```

For more detailed information about deployment commands and options for deploying ARM templates, see:

- Deploy resources with ARM templates and Azure portal
- Deploy resources with ARM templates and Azure CLI
- Deploy resources with ARM templates and Azure PowerShell
- Deploy resources with ARM templates and Azure Resource Manager REST API
- Use a deployment button to deploy templates from GitHub repository
- Deploy ARM templates from Cloud Shell

Deployment location and name

For management group level deployments, you must provide a location for the deployment. The location of the deployment is separate from the location of the resources you deploy. The deployment location specifies where to store deployment data. Subscription and tenant deployments also require a location. For resource group deployments, the location of the resource group is used to store the deployment data.

You can provide a name for the deployment, or use the default deployment name. The default name is the name of the template file. For example, deploying a template named *azuredeploy.json* creates a default deployment name of **azuredeploy**.

For each deployment name, the location is immutable. You can't create a deployment in one location when there's an existing deployment with the same name in a different location. For example, if you create a management group deployment with the name **deployment1** in **centralus**, you can't later create another deployment with the name

deployment1 but a location of **westus**. If you get the error code `InvalidDeploymentLocation`, either use a different name or the same location as the previous deployment for that name.

Deployment scopes

When deploying to a management group, you can deploy resources to:

- the target management group from the operation
- another management group in the tenant
- subscriptions in the management group
- resource groups in the management group
- the tenant for the resource group

An extension resource can be scoped to a target that is different than the deployment target.

The user deploying the template must have access to the specified scope.

This section shows how to specify different scopes. You can combine these different scopes in a single template.

Scope to target management group

Resources defined within the resources section of the template are applied to the management group from the deployment command.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-08-01/managementGroupDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
    management-group-resources-default
  ],
  "outputs": {}
}
```

Scope to another management group

To target another management group, add a nested deployment and specify the `scope` property. Set the `scope` property to a value in the format

`Microsoft.Management/managementGroups/<mg-name>`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-08-01/managementGroupDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "mgName": {
      "type": "string"
    }
  },
  "variables": {
    "mgId": "[concat('Microsoft.Management/managementGroups/', parameters('mgName'))]"
  },
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "nestedDeployment",
      "scope": "[variables('mgId')]",
      "location": "eastus",
      "properties": {
        "mode": "Incremental",
        "template": {
          management-group-resources-non-default
        }
      }
    }
  ],
  "outputs": {}
}
```

Scope to subscription

You can also target subscriptions within a management group. The user deploying the template must have access to the specified scope.

To target a subscription within the management group, use a nested deployment and the `subscriptionId` property.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-08-01/managementGroupDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
```

```
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "nestedSub",
      "location": "westus2",
      "subscriptionId": "00000000-0000-0000-0000-000000000000",
      "properties": {
        "mode": "Incremental",
        "template": {
          "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
          "contentVersion": "1.0.0.0",
          "resources": [
            {
              subscription-resources
            }
          ]
        }
      }
    }
  ]
}
```

Scope to resource group

You can also target resource groups within the management group. The user deploying the template must have access to the specified scope.

To target a resource group within the management group, use a nested deployment. Set the `subscriptionId` and `resourceGroup` properties. Don't set a location for the nested deployment because it's deployed in the location of the resource group.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-08-
01/managementGroupDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "nestedRGDeploy",
      "subscriptionId": "00000000-0000-0000-0000-000000000000",
      "resourceGroup": "demoResourceGroup",
      "properties": {
        "mode": "Incremental",
        "template": {
          "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
          "contentVersion": "1.0.0.0",
```

```
            "resources": [
              {
                resource-group-resources
              }
            ]
          }
        }
      }
    ]
  }
```

To use a management group deployment for creating a resource group within a subscription and deploying a storage account to that resource group, see Deploy to subscription and resource group.

Scope to tenant

To create resources at the tenant, set the `scope` to `/`. The user deploying the template must have the required access to deploy at the tenant.

To use a nested deployment, set `scope` and `location`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-08-01/managementGroupDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "nestedDeployment",
      "location": "centralus",
      "scope": "/",
      "properties": {
        "mode": "Incremental",
        "template": {
          tenant-resources
        }
      }
    }
  ],
  "outputs": {}
}
```

Or, you can set the scope to `/` for some resource types, like management groups. Creating a new management group is described in the next section.

Management group

To create a management group in a management group deployment, you must set the scope to `/` for the management group.

The following example creates a new management group in the root management group.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-08-01/managementGroupDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "mgName": {
      "type": "string",
      "defaultValue": "[concat('mg-', uniqueString(newGuid()))]"
    }
  },
  "resources": [
    {
      "type": "Microsoft.Management/managementGroups",
      "apiVersion": "2021-04-01",
      "name": "[parameters('mgName')]",
      "scope": "/",
      "location": "eastus",
      "properties": {}
    }
  ],
  "outputs": {
    "output": {
      "type": "string",
      "value": "[parameters('mgName')]"
    }
  }
}
```

The next example creates a new management group in the management group specified as the parent. Notice that the scope is set to `/`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-08-01/managementGroupDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "mgName": {
      "type": "string",
```

```
        "defaultValue": "[concat('mg-', uniqueString(newGuid()))]"
      },
      "parentMG": {
        "type": "string"
      }
    },
    "resources": [
      {
        "name": "[parameters('mgName')]",
        "type": "Microsoft.Management/managementGroups",
        "apiVersion": "2021-04-01",
        "scope": "/",
        "location": "eastus",
        "properties": {
          "details": {
            "parent": {
              "id": "
[tenantResourceId('Microsoft.Management/managementGroups',
parameters('parentMG'))]"
            }
          }
        }
      }
    ],
    "outputs": {
      "output": {
        "type": "string",
        "value": "[parameters('mgName')]"
      }
    }
}
```

Subscriptions

To use an ARM template to create a new Azure subscription in a management group, see:

- Programmatically create Azure Enterprise Agreement subscriptions
- Programmatically create Azure subscriptions for a Microsoft Customer Agreement
- Programmatically create Azure subscriptions for a Microsoft Partner Agreement

To deploy a template that moves an existing Azure subscription to a new management group, see Move subscriptions in ARM template

Azure Policy

Custom policy definitions that are deployed to the management group are extensions of the management group. To get the ID of a custom policy definition, use the

extensionResourceId() function. Built-in policy definitions are tenant level resources. To get the ID of a built-in policy definition, use the tenantResourceId() function.

The following example shows how to define a policy at the management group level, and assign it.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-08-
01/managementGroupDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "targetMG": {
      "type": "string",
      "metadata": {
        "description": "Target Management Group"
      }
    },
    "allowedLocations": {
      "type": "array",
      "defaultValue": [
        "australiaeast",
        "australiasoutheast",
        "australiacentral"
      ],
      "metadata": {
        "description": "An array of the allowed locations, all other
locations will be denied by the created policy."
      }
    }
  },
  "variables": {
    "mgScope": "[tenantResourceId('Microsoft.Management/managementGroups',
parameters('targetMG'))]",
    "policyDefinition": "LocationRestriction"
  },
  "resources": [
    {
      "type": "Microsoft.Authorization/policyDefinitions",
      "name": "[variables('policyDefinition')]",
      "apiVersion": "2020-09-01",
      "properties": {
        "policyType": "Custom",
        "mode": "All",
        "parameters": {
        },
        "policyRule": {
          "if": {
            "not": {
              "field": "location",
              "in": "[parameters('allowedLocations')]"
            }
```

```
          },
          "then": {
            "effect": "deny"
          }
        }
      }
    },
    {
      "type": "Microsoft.Authorization/policyAssignments",
      "name": "location-lock",
      "apiVersion": "2020-09-01",
      "dependsOn": [
        "[variables('policyDefinition')]"
      ],
      "properties": {
        "scope": "[variables('mgScope')]",
        "policyDefinitionId": "[extensionResourceId(variables('mgScope'),
'Microsoft.Authorization/policyDefinitions',
variables('policyDefinition'))]"
      }
    }
  ]
}
```

Deploy to subscription and resource group

From a management group level deployment, you can target a subscription within the management group. The following example creates a resource group within a subscription and deploys a storage account to that resource group.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-08-
01/managementGroupDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "nestedsubId": {
      "type": "string"
    },
    "nestedRG": {
      "type": "string"
    },
    "storageAccountName": {
      "type": "string"
    },
    "nestedLocation": {
      "type": "string"
    }
  },
  "resources": [
```

```
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "nestedSub",
      "location": "[parameters('nestedLocation')]",
      "subscriptionId": "[parameters('nestedSubId')]",
      "properties": {
        "mode": "Incremental",
        "template": {
          "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
          "contentVersion": "1.0.0.0",
          "parameters": {
          },
          "variables": {
          },
          "resources": [
            {
              "type": "Microsoft.Resources/resourceGroups",
              "apiVersion": "2021-04-01",
              "name": "[parameters('nestedRG')]",
              "location": "[parameters('nestedLocation')]"
            }
          ]
        }
      }
    },
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "nestedRG",
      "subscriptionId": "[parameters('nestedSubId')]",
      "resourceGroup": "[parameters('nestedRG')]",
      "dependsOn": [
        "nestedSub"
      ],
      "properties": {
        "mode": "Incremental",
        "template": {
          "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
          "contentVersion": "1.0.0.0",
          "resources": [
            {
              "type": "Microsoft.Storage/storageAccounts",
              "apiVersion": "2021-04-01",
              "name": "[parameters('storageAccountName')]",
              "location": "[parameters('nestedLocation')]",
              "kind": "StorageV2",
              "sku": {
                "name": "Standard_LRS"
              }
            }
          ]
        }
```

```
        }
      }
    ]
  }
```

Next steps

- To learn about assigning roles, see Assign Azure roles using Azure Resource Manager templates.
- For an example of deploying workspace settings for Microsoft Defender for Cloud, see deployASCwithWorkspaceSettings.json.
- You can also deploy templates at subscription level and tenant level.

Tenant deployments with ARM templates

Article • 04/09/2023

As your organization matures, you may need to define and assign policies or Azure role-based access control (Azure RBAC) across your Azure AD tenant. With tenant level templates, you can declaratively apply policies and assign roles at a global level.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **tenant deployments**.

Supported resources

Not all resource types can be deployed to the tenant level. This section lists which resource types are supported.

For Azure role-based access control (Azure RBAC), use:

- roleAssignments

For nested templates that deploy to management groups, subscriptions, or resource groups, use:

- deployments

For creating management groups, use:

- managementGroups

For creating subscriptions, use:

- aliases

For managing costs, use:

- billingProfiles
- billingRoleAssignments
- instructions
- invoiceSections
- policies

For configuring the portal, use:

- tenantConfigurations

Built-in policy definitions are tenant-level resources, but you can't deploy custom policy definitions at the tenant. For an example of assigning a built-in policy definition to a resource, see tenantResourceId example.

Schema

The schema you use for tenant deployments is different than the schema for resource group deployments.

For templates, use:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-08-01/tenantDeploymentTemplate.json#",
  ...
}
```

The schema for a parameter file is the same for all deployment scopes. For parameter files, use:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  ...
}
```

Required access

The principal deploying the template must have permissions to create resources at the tenant scope. The principal must have permission to execute the deployment actions (`Microsoft.Resources/deployments/*`) and to create the resources defined in the template. For example, to create a management group, the principal must have Contributor permission at the tenant scope. To create role assignments, the principal must have Owner permission.

The Global Administrator for the Azure Active Directory doesn't automatically have permission to assign roles. To enable template deployments at the tenant scope, the Global Administrator must do the following steps:

1. Elevate account access so the Global Administrator can assign roles. For more information, see Elevate access to manage all Azure subscriptions and management groups.

2. Assign Owner or Contributor to the principal that needs to deploy the templates.

   Azure PowerShell

   ```
   New-AzRoleAssignment -SignInName "[userId]" -Scope "/" -
   RoleDefinitionName "Owner"
   ```

   Azure CLI

   ```
   az role assignment create --assignee "[userId]" --scope "/" --role
   "Owner"
   ```

The principal now has the required permissions to deploy the template.

Deployment commands

The commands for tenant deployments are different than the commands for resource group deployments.

Azure CLI

For Azure CLI, use az deployment tenant create:

Azure CLI

```
az deployment tenant create \
  --name demoTenantDeployment \
  --location WestUS \
  --template-uri "https://raw.githubusercontent.com/Azure/azure-
quickstart-templates/master/tenant-deployments/new-mg/azuredeploy.json"
```

For more detailed information about deployment commands and options for deploying ARM templates, see:

- Deploy resources with ARM templates and Azure portal

- Deploy resources with ARM templates and Azure CLI
- Deploy resources with ARM templates and Azure PowerShell
- Deploy resources with ARM templates and Azure Resource Manager REST API
- Use a deployment button to deploy templates from GitHub repository
- Deploy ARM templates from Cloud Shell

Deployment location and name

For tenant level deployments, you must provide a location for the deployment. The location of the deployment is separate from the location of the resources you deploy. The deployment location specifies where to store deployment data. Subscription and management group deployments also require a location. For resource group deployments, the location of the resource group is used to store the deployment data.

You can provide a name for the deployment, or use the default deployment name. The default name is the name of the template file. For example, deploying a template named *azuredeploy.json* creates a default deployment name of **azuredeploy**.

For each deployment name, the location is immutable. You can't create a deployment in one location when there's an existing deployment with the same name in a different location. For example, if you create a tenant deployment with the name **deployment1** in **centralus**, you can't later create another deployment with the name **deployment1** but a location of **westus**. If you get the error code `InvalidDeploymentLocation`, either use a different name or the same location as the previous deployment for that name.

Deployment scopes

When deploying to a tenant, you can deploy resources to:

- the tenant
- management groups within the tenant
- subscriptions
- resource groups

An extension resource can be scoped to a target that is different than the deployment target.

The user deploying the template must have access to the specified scope.

This section shows how to specify different scopes. You can combine these different scopes in a single template.

Scope to tenant

Resources defined within the resources section of the template are applied to the tenant.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-08-01/tenantDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
    tenant-resources
  ],
  "outputs": {}
}
```

Scope to management group

To target a management group within the tenant, add a nested deployment and specify the `scope` property.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-08-01/tenantDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "mgName": {
      "type": "string"
    }
  },
  "variables": {
    "mgId": "[concat('Microsoft.Management/managementGroups/', parameters('mgName'))]"
  },
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "nestedMG",
      "scope": "[variables('mgId')]",
      "location": "eastus",
      "properties": {
        "mode": "Incremental",
        "template": {
          management-group-resources
        }
      }
```

```
    }
  ],
  "outputs": {}
}
```

Scope to subscription

You can also target subscriptions within the tenant. The user deploying the template must have access to the specified scope.

To target a subscription within the tenant, use a nested deployment and the `subscriptionId` property.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-08-01/tenantDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "nestedSub",
      "location": "westus2",
      "subscriptionId": "00000000-0000-0000-0000-000000000000",
      "properties": {
        "mode": "Incremental",
        "template": {
          "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
          "contentVersion": "1.0.0.0",
          "resources": [
            {
              subscription-resources
            }
          ]
        }
      }
    }
  ]
}
```

Scope to resource group

You can also target resource groups within the tenant. The user deploying the template must have access to the specified scope.

To target a resource group within the tenant, use a nested deployment. Set the `subscriptionId` and `resourceGroup` properties. Don't set a location for the nested deployment because it's deployed in the location of the resource group.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-08-01/tenantDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "nestedRGDeploy",
      "subscriptionId": "00000000-0000-0000-0000-000000000000",
      "resourceGroup": "demoResourceGroup",
      "properties": {
        "mode": "Incremental",
        "template": {
          "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
          "contentVersion": "1.0.0.0",
          "resources": [
            {
              resource-group-resources
            }
          ]
        }
      }
    }
  ]
}
```

Create management group

The following template creates a management group.

JSON

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-08-01/tenantDeploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
      "mgName": {
        "type": "string",
        "defaultValue": "[concat('mg-', uniqueString(newGuid()))]"
      }
    },
```

```
    "resources": [
      {
        "type": "Microsoft.Management/managementGroups",
        "apiVersion": "2020-02-01",
        "name": "[parameters('mgName')]",
        "properties": {
        }
      }
    ]
  }
```

If your account doesn't have permission to deploy to the tenant, you can still create management groups by deploying to another scope. For more information, see Management group.

Assign role

The following template assigns a role at the tenant scope.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-08-
01/tenantDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "17107802581699825924"
    }
  },
  "parameters": {
    "principalId": {
      "type": "string",
      "metadata": {
        "description": "principalId if the user that will be given
contributor access to the tenant"
      }
    },
    "roleDefinitionId": {
      "type": "string",
      "defaultValue": "8e3af657-a8ff-443c-a75c-2fe8c4bcb635",
      "metadata": {
        "description": "roleDefinition for the assignment - default is
owner"
      }
    }
  },
  "variables": {
    "roleAssignmentName": "[guid('/', parameters('principalId'),
```

```
parameters('roleDefinitionId'))]"
  },
  "resources": [
    {
      "type": "Microsoft.Authorization/roleAssignments",
      "apiVersion": "2020-03-01-preview",
      "name": "[variables('roleAssignmentName')]",
      "properties": {
        "roleDefinitionId": "
[tenantResourceId('Microsoft.Authorization/roleDefinitions',
parameters('roleDefinitionId'))]",
        "principalId": "[parameters('principalId')]"
      }
    }
  ]
}
```

Next steps

- To learn about assigning roles, see Assign Azure roles using Azure Resource Manager templates.
- You can also deploy templates at subscription level or management group level.

Setting scope for extension resources in ARM templates

Article • 07/12/2022

An extension resource is a resource that modifies another resource. For example, you can assign a role to a resource. The role assignment is an extension resource type.

For a full list of extension resource types, see Resource types that extend capabilities of other resources.

This article shows how to set the scope for an extension resource type when deployed with an Azure Resource Manager template (ARM template). It describes the scope property that is available for extension resources when applying to a resource.

> **Note**
>
> The scope property is only available to extension resource types. To specify a different scope for a resource type that isn't an extension type, use a nested or linked deployment. For more information, see **resource group deployments**, **subscription deployments**, **management group deployments**, and **tenant deployments**.

Apply at deployment scope

To apply an extension resource type at the target deployment scope, you add the resource to your template, as would with any resource type. The available scopes are resource group, subscription, management group, and tenant. The deployment scope must support the resource type.

The following template deploys a lock.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
  },
  "resources": [
    {
      "type": "Microsoft.Authorization/locks",
```

```
      "apiVersion": "2016-09-01",
      "name": "rgLock",
      "properties": {
        "level": "CanNotDelete",
        "notes": "Resource Group should not be deleted."
      }
    }
  ]
}
```

When deployed to a resource group, it locks the resource group.

Azure CLI

Azure CLI

```
az deployment group create \
  --resource-group ExampleGroup \
  --template-uri "https://raw.githubusercontent.com/Azure/azure-docs-
json-samples/master/azure-resource-manager/scope/locktargetscope.json"
```

The next example assigns a role.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2018-05-
01/subscriptionDeploymentTemplate.json#",
  "contentVersion": "1.0.0.1",
  "parameters": {
    "principalId": {
      "type": "string",
      "metadata": {
        "description": "The principal to assign the role to"
      }
    },
    "builtInRoleType": {
      "type": "string",
      "allowedValues": [
        "Owner",
        "Contributor",
        "Reader"
      ],
      "metadata": {
        "description": "Built-in role to assign"
      }
    },
    "roleNameGuid": {
      "type": "string",
      "metadata": {
```

```
        "description": "The role assignment name"
      }
    }
  },
  "variables": {
    "Owner": "[concat('/subscriptions/', subscription().subscriptionId,
'/providers/Microsoft.Authorization/roleDefinitions/', '8e3af657-a8ff-443c-
a75c-2fe8c4bcb635')]",
    "Contributor": "[concat('/subscriptions/',
subscription().subscriptionId,
'/providers/Microsoft.Authorization/roleDefinitions/', 'b24988ac-6180-42a0-
ab88-20f7382dd24c')]",
    "Reader": "[concat('/subscriptions/', subscription().subscriptionId,
'/providers/Microsoft.Authorization/roleDefinitions/', 'acdd72a7-3385-48ef-
bd42-f606fba81ae7')]"
  },
  "resources": [
    {
      "type": "Microsoft.Authorization/roleAssignments",
      "apiVersion": "2020-10-01-preview",
      "name": "[parameters('roleNameGuid')]",
      "properties": {
        "roleDefinitionId": "[variables(parameters('builtInRoleType'))]",
        "principalId": "[parameters('principalId')]"
      }
    }
  ],
  "outputs": {}
}
```

When deployed to a subscription, it assigns the role to the subscription.

Azure CLI

Azure CLI

```
az deployment sub create \
  --name demoSubDeployment \
  --location centralus \
  --template-uri "https://raw.githubusercontent.com/Azure/azure-docs-
json-samples/master/azure-resource-manager/scope/roletargetscope.json"
```

Apply to resource

To apply an extension resource to a resource, use the `scope` property. Set the scope property to the name of the resource you're adding the extension to. The scope property is a root property for the extension resource type.

The following example creates a storage account and applies a role to it.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "principalId": {
      "type": "string",
      "metadata": {
        "description": "The principal to assign the role to"
      }
    },
    "builtInRoleType": {
      "type": "string",
      "allowedValues": [
        "Owner",
        "Contributor",
        "Reader"
      ],
      "metadata": {
        "description": "Built-in role to assign"
      }
    },
    "roleNameGuid": {
      "type": "string",
      "defaultValue": "[newGuid()]",
      "metadata": {
        "description": "A new GUID used to identify the role assignment"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    }
  },
  "variables": {
    "Owner": "[concat('/subscriptions/', subscription().subscriptionId, '/providers/Microsoft.Authorization/roleDefinitions/', '8e3af657-a8ff-443c-a75c-2fe8c4bcb635')]",
    "Contributor": "[concat('/subscriptions/', subscription().subscriptionId, '/providers/Microsoft.Authorization/roleDefinitions/', 'b24988ac-6180-42a0-ab88-20f7382dd24c')]",
    "Reader": "[concat('/subscriptions/', subscription().subscriptionId, '/providers/Microsoft.Authorization/roleDefinitions/', 'acdd72a7-3385-48ef-bd42-f606fba81ae7')]",
    "storageName": "[concat('storage', uniqueString(resourceGroup().id))]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
```

```
      "apiVersion": "2021-04-01",
      "name": "[variables('storageName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "Storage",
      "properties": {}
    },
    {
      "type": "Microsoft.Authorization/roleAssignments",
      "apiVersion": "2020-10-01-preview",
      "name": "[parameters('roleNameGuid')]",
      "scope": "[concat('Microsoft.Storage/storageAccounts', '/',
variables('storageName'))]",
      "dependsOn": [
        "[variables('storageName')]"
      ],
      "properties": {
        "roleDefinitionId": "[variables(parameters('builtInRoleType'))]",
        "principalId": "[parameters('principalId')]"
      }
    }
  ]
}
```

The resourceGroup and subscription properties are only allowed on nested or linked deployments. These properties are not allowed on individual resources. Use nested or linked deployments if you want to deploy an extension resource with the scope set to a resource in a different resource group.

Next steps

- To understand how to define parameters in your template, see Understand the structure and syntax of ARM 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 ARM template with SAS token.

ARM template functions in deployment scopes

Article • 03/09/2023

With Azure Resource Manager templates (ARM templates), you can deploy to resource groups, subscriptions, management groups, or tenants. Generally, ARM template functions work the same for all scopes. This article describes the differences that exist for some functions depending on the scope.

Supported functions

When deploying to different scopes, there are some important considerations:

- The resourceGroup() function is **supported** for resource group deployments.
- The subscription() function is **supported** for resource group and subscription deployments.
- The reference() and list() functions are **supported** for all scopes.
- Use resourceId() to get the ID for a resource deployed at the resources group.

  JSON

  ```
  "subnet": {
    "id": "[resourceId(parameters('virtualNetworkResourceGroup'),
  'Microsoft.Network/virtualNetworks/subnets',
  parameters('virtualNetworkName'), parameters('subnet1Name'))]"
  }
  ```

- Use the subscriptionResourceId() function to get the ID for a resource deployed at the subscription.

  For example, to get the resource ID for a policy definition that is deployed to a subscription, use:

  JSON

  ```
  "roleDefinitionId": "
  [subscriptionResourceId('Microsoft.Authorization/roleDefinitions',
  'acdd72a7-3385-48ef-bd42-f606fba81ae7')]"
  ```

- Use the extensionResourceId() function for resources that are implemented as extensions of the management group. Custom policy definitions that are deployed to the management group are extensions of the management group.

  To get the resource ID for a custom policy definition at the management group level, use:

  JSON

  ```
  "policyDefinitionId": "[extensionResourceId(variables('mgScope'),
  'Microsoft.Authorization/policyDefinitions',
  parameters('policyDefinitionID'))]"
  ```

- Use the tenantResourceId() function to get the ID for a resource deployed at the tenant. Built-in policy definitions are tenant level resources. When assigning a built-in policy at the management group level, use the tenantResourceId function.

  To get the resource ID for a built-in policy definition, use:

  JSON

  ```
  "policyDefinitionId": "
  [tenantResourceId('Microsoft.Authorization/policyDefinitions',
  parameters('policyDefinitionID'))]"
  ```

Function resolution in scopes

When you deploy to more than one scope, the resourceGroup() and subscription() functions resolve differently based on how you specify the template. When you link to an external template, the functions always resolve to the scope for that template. When you nest a template within a parent template, use the `expressionEvaluationOptions` property to specify whether the functions resolve to the resource group and subscription for the parent template or the nested template. Set the property to `inner` to resolve to the scope for the nested template. Set the property to `outer` to resolve to the scope of the parent template.

The following table shows whether the functions resolve to the parent or embedded resource group and subscription.

Template type	Scope	Resolution
nested	outer (default)	Parent resource group

Template type	Scope	Resolution
nested	inner	Sub resource group
linked	N/A	Sub resource group

The following example template shows:

- nested template with default (outer) scope
- nested template with inner scope
- linked template

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {},
  "variables": {},
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "defaultScopeTemplate",
      "resourceGroup": "inlineGroup",
      "properties": {
        "mode": "Incremental",
        "template": {
          "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
          "contentVersion": "1.0.0.0",
          "parameters": {},
          "variables": {},
          "resources": [
          ],
          "outputs": {
            "resourceGroupOutput": {
              "type": "string",
              "value": "[resourceGroup().name]"
            }
          }
        },
        "parameters": {}
      }
    },
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "innerScopeTemplate",
      "resourceGroup": "inlineGroup",
      "properties": {
```

```
        "expressionEvaluationOptions": {
          "scope": "inner"
        },
        "mode": "Incremental",
        "template": {
          "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
          "contentVersion": "1.0.0.0",
          "parameters": {},
          "variables": {},
          "resources": [
          ],
          "outputs": {
            "resourceGroupOutput": {
              "type": "string",
              "value": "[resourceGroup().name]"
            }
          }
        },
        "parameters": {}
      }
    },
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2021-04-01",
      "name": "linkedTemplate",
      "resourceGroup": "linkedGroup",
      "properties": {
        "mode": "Incremental",
        "templateLink": {
          "contentVersion": "1.0.0.0",
          "uri": "https://raw.githubusercontent.com/Azure/azure-docs-json-
samples/master/azure-resource-manager/resourcegroupname.json"
        },
        "parameters": {}
      }
    }
  ],
  "outputs": {
    "parentRG": {
      "type": "string",
      "value": "[concat('Parent resource group is ', resourceGroup().name)]"
    },
    "defaultScopeRG": {
      "type": "string",
      "value": "[concat('Default scope resource group is ',
reference('defaultScopeTemplate').outputs.resourceGroupOutput.value)]"
    },
    "innerScopeRG": {
      "type": "string",
      "value": "[concat('Inner scope resource group is ',
reference('innerScopeTemplate').outputs.resourceGroupOutput.value)]"
    },
    "linkedRG": {
      "type": "string",
```

```
      "value": "[concat('Linked resource group is ',
reference('linkedTemplate').outputs.resourceGroupOutput.value)]"
    }
  }
}
```

To test the preceding template and see the results, use PowerShell or Azure CLI.

PowerShell

Azure PowerShell

```
New-AzResourceGroup -Name parentGroup -Location southcentralus
New-AzResourceGroup -Name inlineGroup -Location southcentralus
New-AzResourceGroup -Name linkedGroup -Location southcentralus

New-AzResourceGroupDeployment `
  -ResourceGroupName parentGroup `
  -TemplateUri https://raw.githubusercontent.com/Azure/azure-docs-json-
samples/master/azure-resource-manager/crossresourcegroupproperties.json
```

The output from the preceding example is:

Output

```
 Name              Type                        Value
 ===============   ==========================  ==========
 parentRG          String                      Parent resource group is
parentGroup
 defaultScopeRG    String                      Default scope resource
group is parentGroup
 innerScopeRG      String                      Inner scope resource group
is inlineGroup
 linkedRG          String                      Linked resource group is
linkedgroup
```

Next steps

- To understand how to define parameters in your template, see Understand the structure and syntax of ARM 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 ARM template with SAS token.

ARM template deployment what-if operation

Article • 04/09/2023

Before deploying an Azure Resource Manager template (ARM template), you can preview the changes that will happen. Azure Resource Manager provides the what-if operation to let you see how resources will change if you deploy the template. The what-if operation doesn't make any changes to existing resources. Instead, it predicts the changes if the specified template is deployed.

You can use the what-if operation with Azure PowerShell, Azure CLI, or REST API operations. What-if is supported for resource group, subscription, management group, and tenant level deployments.

Training resources

To learn more about what-if, and for hands-on guidance, see Preview Azure deployment changes by using what-if.

Required permissions

To deploy a Bicep file or ARM template, you need write access on the resources you're deploying and access to all operations on the Microsoft.Resources/deployments resource type. For example, to deploy a virtual machine, you need `Microsoft.Compute/virtualMachines/write` and `Microsoft.Resources/deployments/*` permissions. The what-if operation has the same permission requirements.

For a list of roles and permissions, see Azure built-in roles.

What-if limits

What-if expands nested templates until these limits are reached:

- 500 nested templates.
- 800 resource groups in a cross resource-group deployment.
- 5 minutes taken for expanding the nested templates.

When one of the limits is reached, the remaining resources' change type is set to **Ignore**.

Install Azure PowerShell module

To use what-if in PowerShell, you must have version **4.2 or later of the Az module**.

To install the module, use:

PowerShell

```
Install-Module -Name Az -Force
```

For more information about installing modules, see Install Azure PowerShell.

Install Azure CLI module

To use what-if in Azure CLI, you must have Azure CLI 2.14.0 or later. If needed, install the latest version of Azure CLI.

See results

When you use what-if in PowerShell or Azure CLI, the output includes color-coded results that help you see the different types of changes.

```
Resource and property changes are indicated with these symbols:
  - Delete
  + Create
  ~ Modify

The deployment will update the following scope:

Scope: /subscriptions/./resourceGroups/ExampleGroup

  ~ Microsoft.Network/virtualNetworks/vnet-001 [2018-10-01]
    - tags.Owner: "Team A"
    ~ properties.addressSpace.addressPrefixes: [
      - 0: "10.0.0.0/16"
      + 0: "10.0.0.0/15"
      ]
    ~ properties.subnets: [
      - 0:

          name:                      "subnet001"
          properties.addressPrefix: "10.0.0.0/24"

      ]

Resource changes: 1 to modify.
```

The text output is:

PowerShell

```
Resource and property changes are indicated with these symbols:
  - Delete
  + Create
  ~ Modify

The deployment will update the following scope:

Scope: /subscriptions/./resourceGroups/ExampleGroup

  ~ Microsoft.Network/virtualNetworks/vnet-001 [2018-10-01]
    - tags.Owner: "Team A"
    ~ properties.addressSpace.addressPrefixes: [
      - 0: "10.0.0.0/16"
      + 0: "10.0.0.0/15"
      ]
    ~ properties.subnets: [
      - 0:

          name:                      "subnet001"
          properties.addressPrefix: "10.0.0.0/24"

      ]

Resource changes: 1 to modify.
```

Note

The what-if operation can't resolve the **reference function**. Every time you set a property to a template expression that includes the reference function, what-if reports the property will change. This behavior happens because what-if compares the current value of the property (such as `true` or `false` for a boolean value) with the unresolved template expression. Obviously, these values will not match. When you deploy the template, the property will only change when the template expression resolves to a different value.

What-if commands

Azure PowerShell

To preview changes before deploying a template, use New-AzResourceGroupDeployment or New-AzSubscriptionDeployment. Add the `-Whatif` switch parameter to the deployment command.

- `New-AzResourceGroupDeployment -Whatif` for resource group deployments
- `New-AzSubscriptionDeployment -Whatif` and `New-AzDeployment -Whatif` for subscription level deployments

You can use the `-Confirm` switch parameter to preview the changes and get prompted to continue with the deployment.

- `New-AzResourceGroupDeployment -Confirm` for resource group deployments
- `New-AzSubscriptionDeployment -Confirm` and `New-AzDeployment -Confirm` for subscription level deployments

The preceding commands return a text summary that you can manually inspect. To get an object that you can programmatically inspect for changes, use Get-AzResourceGroupDeploymentWhatIfResult or Get-AzSubscriptionDeploymentWhatIfResult.

- `$results = Get-AzResourceGroupDeploymentWhatIfResult` for resource group deployments
- `$results = Get-AzSubscriptionDeploymentWhatIfResult` or `$results = Get-AzDeploymentWhatIfResult` for subscription level deployments

Azure CLI

To preview changes before deploying a template, use:

- az deployment group what-if for resource group deployments
- az deployment sub what-if for subscription level deployments
- az deployment mg what-if for management group deployments
- az deployment tenant what-if for tenant deployments

You can use the `--confirm-with-what-if` switch (or its short form `-c`) to preview the changes and get prompted to continue with the deployment. Add this switch to:

- az deployment group create
- az deployment sub create.
- az deployment mg create
- az deployment tenant create

For example, use `az deployment group create --confirm-with-what-if` or `-c` for resource group deployments.

The preceding commands return a text summary that you can manually inspect. To get a JSON object that you can programmatically inspect for changes, use the `--no-pretty-print` switch. For example, use `az deployment group what-if --no-pretty-print` for resource group deployments.

If you want to return the results without colors, open your Azure CLI configuration file. Set **no_color** to **yes**.

Azure REST API

For REST API, use:

- Deployments - What If for resource group deployments
- Deployments - What If At Subscription Scope for subscription deployments
- Deployments - What If At Management Group Scope for management group deployments
- Deployments - What If At Tenant Scope for tenant deployments.

Change types

The what-if operation lists six different types of changes:

- **Create**: The resource doesn't currently exist but is defined in the template. The resource will be created.

- **Delete**: This change type only applies when using complete mode for deployment. The resource exists, but isn't defined in the template. With complete mode, the resource will be deleted. Only resources that support complete mode deletion are included in this change type.

- **Ignore**: The resource exists, but isn't defined in the template. The resource won't be deployed or modified. When you reach the limits for expanding nested templates, you will encounter this change type. See What-if limits.

- **NoChange**: The resource exists, and is defined in the template. The resource will be redeployed, but the properties of the resource won't change. This change type is returned when ResultFormat is set to `FullResourcePayloads`, which is the default value.

- **Modify**: The resource exists, and is defined in the template. The resource will be redeployed, and the properties of the resource will change. This change type is

returned when ResultFormat is set to `FullResourcePayloads`, which is the default value.

- **Deploy**: The resource exists, and is defined in the template. The resource will be redeployed. The properties of the resource may or may not change. The operation returns this change type when it doesn't have enough information to determine if any properties will change. You only see this condition when ResultFormat is set to `ResourceIdOnly`.

Result format

You control the level of detail that is returned about the predicted changes. You have two options:

- **FullResourcePayloads** - returns a list of resources that will change and details about the properties that will change
- **ResourceIdOnly** - returns a list of resources that will change

The default value is **FullResourcePayloads**.

For PowerShell deployment commands, use the `-WhatIfResultFormat` parameter. In the programmatic object commands, use the `ResultFormat` parameter.

For Azure CLI, use the `--result-format` parameter.

The following results show the two different output formats:

- Full resource payloads

PowerShell

```
Resource and property changes are indicated with these symbols:
  - Delete
  + Create
  ~ Modify

The deployment will update the following scope:

Scope: /subscriptions/./resourceGroups/ExampleGroup

  ~ Microsoft.Network/virtualNetworks/vnet-001 [2018-10-01]
    - tags.Owner: "Team A"
    ~ properties.addressSpace.addressPrefixes: [
      - 0: "10.0.0.0/16"
      + 0: "10.0.0.0/15"
      ]
    ~ properties.subnets: [
```

```
      - 0:

        name:                       "subnet001"
        properties.addressPrefix: "10.0.0.0/24"

      ]

Resource changes: 1 to modify.
```

- Resource ID only

PowerShell

```
Resource and property changes are indicated with this symbol:
  ! Deploy

The deployment will update the following scope:

Scope: /subscriptions/./resourceGroups/ExampleGroup

  ! Microsoft.Network/virtualNetworks/vnet-001

Resource changes: 1 to deploy.
```

Run what-if operation

Set up environment

To see how what-if works, let's runs some tests. First, deploy a template that creates a virtual network. You'll use this virtual network to test how changes are reported by what-if.

PowerShell

Azure PowerShell

```
New-AzResourceGroup `
  -Name ExampleGroup `
  -Location centralus
New-AzResourceGroupDeployment `
  -ResourceGroupName ExampleGroup `
  -TemplateUri "https://raw.githubusercontent.com/Azure/azure-docs-json-samples/master/azure-resource-manager/what-if/what-if-before.json"
```

Test modification

After the deployment completes, you're ready to test the what-if operation. This time you deploy a template that changes the virtual network. It's missing one the original tags, a subnet has been removed, and the address prefix has changed.

PowerShell

Azure PowerShell

```
New-AzResourceGroupDeployment `
  -Whatif `
  -ResourceGroupName ExampleGroup `
  -TemplateUri "https://raw.githubusercontent.com/Azure/azure-docs-json-samples/master/azure-resource-manager/what-if/what-if-after.json"
```

The what-if output appears similar to:

```
Resource and property changes are indicated with these symbols:
  - Delete
  + Create
  ~ Modify

The deployment will update the following scope:

Scope: /subscriptions/./resourceGroups/ExampleGroup

  ~ Microsoft.Network/virtualNetworks/vnet-001 [2018-10-01]
    - tags.Owner: "Team A"
    ~ properties.addressSpace.addressPrefixes: [
      - 0: "10.0.0.0/16"
      + 0: "10.0.0.0/15"
      ]
    ~ properties.subnets: [
      - 0:

          name:                      "subnet001"
          properties.addressPrefix: "10.0.0.0/24"

      ]

Resource changes: 1 to modify.
```

The text output is:

PowerShell

```
Resource and property changes are indicated with these symbols:
  - Delete
  + Create
```

```
  ~ Modify

The deployment will update the following scope:

Scope: /subscriptions/./resourceGroups/ExampleGroup

  ~ Microsoft.Network/virtualNetworks/vnet-001 [2018-10-01]
    - tags.Owner: "Team A"
    ~ properties.addressSpace.addressPrefixes: [
      - 0: "10.0.0.0/16"
      + 0: "10.0.0.0/15"
      ]
    ~ properties.subnets: [
      - 0:

        name:                      "subnet001"
        properties.addressPrefix: "10.0.0.0/24"

      ]

Resource changes: 1 to modify.
```

Notice at the top of the output that colors are defined to indicate the type of changes.

At the bottom of the output, it shows the tag Owner was deleted. The address prefix changed from 10.0.0.0/16 to 10.0.0.0/15. The subnet named subnet001 was deleted. Remember these changes weren't deployed. You see a preview of the changes that will happen if you deploy the template.

Some of the properties that are listed as deleted won't actually change. Properties can be incorrectly reported as deleted when they aren't in the template, but are automatically set during deployment as default values. This result is considered "noise" in the what-if response. The final deployed resource will have the values set for the properties. As the what-if operation matures, these properties will be filtered out of the result.

Programmatically evaluate what-if results

Now, let's programmatically evaluate the what-if results by setting the command to a variable.

PowerShell

Azure PowerShell

```
$results = Get-AzResourceGroupDeploymentWhatIfResult `
  -ResourceGroupName ExampleGroup `
```

```
  -TemplateUri "https://raw.githubusercontent.com/Azure/azure-docs-json-samples/master/azure-resource-manager/what-if/what-if-after.json"
```

You can see a summary of each change.

Azure PowerShell

```
foreach ($change in $results.Changes)
{
  $change.Delta
}
```

Confirm deletion

The what-if operation supports using deployment mode. When set to complete mode, resources not in the template are deleted. The following example deploys a template that has no resources defined in complete mode.

To preview changes before deploying a template, use the confirm switch parameter with the deployment command. If the changes are as you expected, respond that you want the deployment to complete.

PowerShell

Azure PowerShell

```
New-AzResourceGroupDeployment `
  -ResourceGroupName ExampleGroup `
  -Mode Complete `
  -Confirm `
  -TemplateUri "https://raw.githubusercontent.com/Azure/azure-docs-json-samples/master/azure-resource-manager/what-if/azuredeploy.json"
```

Because no resources are defined in the template and the deployment mode is set to complete, the virtual network will be deleted.

```
Resource and property changes are indicated with this symbol:
  - Delete

The deployment will update the following scope:

Scope: /subscriptions/. /resourceGroups/ExampleGroup

  - Microsoft.Network/virtualNetworks/vnet-001

      id:
"/subscriptions/. /resourceGroups/ExampleGroup/providers/Microsoft.Network/virtualNet
works/vnet-001"
      location:        "centralus"
      name:            "vnet-001"
      tags.CostCenter: "12345"
      tags.Owner:      "Team A"
      type:            "Microsoft.Network/virtualNetworks"

Resource changes: 1 to delete.

Are you sure you want to execute the deployment?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "Y"):
```

The text output is:

PowerShell

```
Resource and property changes are indicated with this symbol:
  - Delete

The deployment will update the following scope:

Scope: /subscriptions/./resourceGroups/ExampleGroup

  - Microsoft.Network/virtualNetworks/vnet-001

      id:
"/subscriptions/./resourceGroups/ExampleGroup/providers/Microsoft.Network/vi
rtualNet
works/vnet-001"
      location:        "centralus"
      name:            "vnet-001"
      tags.CostCenter: "12345"
      tags.Owner:      "Team A"
      type:            "Microsoft.Network/virtualNetworks"

Resource changes: 1 to delete.

Are you sure you want to execute the deployment?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help
(default is "Y"):
```

You see the expected changes and can confirm that you want the deployment to run.

SDKs

You can use the what-if operation through the Azure SDKs.

- For Python, use what-if.
- For Java, use DeploymentWhatIf Class.
- For .NET, use DeploymentWhatIf Class.

Next steps

- ARM Deployment Insights extension provides an easy way to integrate the what-if operation in your Azure DevOps pipeline.
- To use the what-if operation in a pipeline, see Test ARM templates with What-If in a pipeline.
- If you notice incorrect results from the what-if operation, please report the issues at https://aka.ms/whatifissues.
- For a Learn module that covers using what if, see Preview changes and validate Azure resources by using what-if and the ARM template test toolkit.

Rollback on error to successful deployment

Article • 03/09/2023

When a deployment fails, you can automatically redeploy an earlier, successful deployment from your deployment history. This functionality is useful if you've got a known good state for your infrastructure deployment and want to revert to this state. You can specify either a particular earlier deployment or the last successful deployment.

> **Important**
>
> This feature rollbacks a failed deployment by redeploying an earlier deployment. This result may be different than what you would expect from undoing the failed deployment. Make sure you understand how the earlier deployment is redeployed.

Considerations for redeploying

Before using this feature, consider these details about how the redeployment is handled:

- The previous deployment is run using the complete mode, even if you used incremental mode during the earlier deployment. Redeploying in complete mode could produce unexpected results when the earlier deployment used incremental. The complete mode means that any resources not included in the previous deployment are deleted. Specify an earlier deployment that represents all of the resources and their states that you want to exist in the resource group. For more information, see deployment modes.
- The redeployment is run exactly as it was run previously with the same parameters. You can't change the parameters.
- The redeployment only affects the resources, any data changes aren't affected.
- You can use this feature only with resource group deployments. It doesn't support subscription, management group, or tenant level deployments. For more information about subscription level deployment, see Create resource groups and resources at the subscription level.
- You can only use this option with root level deployments. Deployments from a nested template aren't available for redeployment.

To use this option, your deployments must have unique names in the deployment history. It's only with unique names that a specific deployment can be identified. If you

don't have unique names, a failed deployment might overwrite a successful deployment in the history.

If you specify an earlier deployment that doesn't exist in the deployment history, the rollback returns an error.

PowerShell

To redeploy the last successful deployment, add the `-RollbackToLastDeployment` parameter as a flag.

Azure PowerShell

```
New-AzResourceGroupDeployment -Name ExampleDeployment02 `
  -ResourceGroupName $resourceGroupName `
  -TemplateFile c:\MyTemplates\azuredeploy.json `
  -RollbackToLastDeployment
```

To redeploy a specific deployment, use the `-RollBackDeploymentName` parameter and provide the name of the deployment. The specified deployment must have succeeded.

Azure PowerShell

```
New-AzResourceGroupDeployment -Name ExampleDeployment02 `
  -ResourceGroupName $resourceGroupName `
  -TemplateFile c:\MyTemplates\azuredeploy.json `
  -RollBackDeploymentName ExampleDeployment01
```

Azure CLI

To redeploy the last successful deployment, add the `--rollback-on-error` parameter as a flag.

Azure CLI

```
az deployment group create \
  --name ExampleDeployment \
  --resource-group ExampleGroup \
  --template-file storage.json \
  --parameters storageAccountType=Standard_GRS \
  --rollback-on-error
```

To redeploy a specific deployment, use the `--rollback-on-error` parameter and provide the name of the deployment. The specified deployment must have succeeded.

Azure CLI

```
az deployment group create \
  --name ExampleDeployment02 \
  --resource-group ExampleGroup \
  --template-file storage.json \
  --parameters storageAccountType=Standard_GRS \
  --rollback-on-error ExampleDeployment01
```

REST API

To redeploy the last successful deployment if the current deployment fails, use:

JSON

```
{
  "properties": {
    "templateLink": {
      "uri":
"http://mystorageaccount.blob.core.windows.net/templates/template.json",
      "contentVersion": "1.0.0.0"
    },
    "mode": "Incremental",
    "parametersLink": {
      "uri":
"http://mystorageaccount.blob.core.windows.net/templates/parameters.json",
      "contentVersion": "1.0.0.0"
    },
    "onErrorDeployment": {
      "type": "LastSuccessful",
    }
  }
}
```

To redeploy a specific deployment if the current deployment fails, use:

JSON

```
{
  "properties": {
    "templateLink": {
      "uri":
"http://mystorageaccount.blob.core.windows.net/templates/template.json",
      "contentVersion": "1.0.0.0"
    },
    "mode": "Incremental",
    "parametersLink": {
      "uri":
"http://mystorageaccount.blob.core.windows.net/templates/parameters.json",
```

```
        "contentVersion": "1.0.0.0"
      },
      "onErrorDeployment": {
        "type": "SpecificDeployment",
        "deploymentName": "<deploymentname>"
      }
    }
  }
```

The specified deployment must have succeeded.

Next steps

- To understand complete and incremental modes, 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.

How to deploy private ARM template with SAS token

Article • 04/09/2023

When your Azure Resource Manager template (ARM template) is located in a storage account, you can restrict access to the template to avoid exposing it publicly. You access a secured template by creating a shared access signature (SAS) token for the template, and providing that token during deployment. This article explains how to use Azure PowerShell or Azure CLI to securely deploy an ARM template with a SAS token.

You will find information on how to protect and manage access to your private ARM templates with directions on how to do the following:

- Create storage account with secured container
- Upload template to storage account
- Provide SAS token during deployment

> **Important**
>
> Instead of securing your private template with a SAS token, consider using **template specs**. With template specs, you can share your templates with other users in your organization and manage access to the templates through Azure RBAC.

Create storage account with secured container

The following script creates a storage account and container with public access turned off for template security.

PowerShell

Azure PowerShell

```
New-AzResourceGroup `
  -Name ExampleGroup `
  -Location "Central US"
New-AzStorageAccount `
  -ResourceGroupName ExampleGroup `
  -Name {your-unique-name} `
  -Type Standard_LRS `
  -Location "Central US"
```

```
Set-AzCurrentStorageAccount `
  -ResourceGroupName ExampleGroup `
  -Name {your-unique-name}
New-AzStorageContainer `
  -Name templates `
  -Permission Off
```

Upload private template to storage account

Now, you're ready to upload your template to the storage account. Provide the path to the template you want to use.

PowerShell

Azure PowerShell

```
Set-AzStorageBlobContent `
  -Container templates `
  -File c:\Templates\azuredeploy.json
```

Provide SAS token during deployment

To deploy a private template in a storage account, generate a SAS token and include it in the URI for the template. Set the expiry time to allow enough time to complete the deployment.

> **Important**
>
> The blob containing the private template is accessible to only the account owner. However, when you create a SAS token for the blob, the blob is accessible to anyone with that URI. If another user intercepts the URI, that user is able to access the template. A SAS token is a good way of limiting access to your templates, but you should not include sensitive data like passwords directly in the template.

PowerShell

Azure PowerShell

```
# get the URI with the SAS token
$templateuri = New-AzStorageBlobSASToken `
```

```
  -Container templates `
  -Blob azuredeploy.json `
  -Permission r `
  -ExpiryTime (Get-Date).AddHours(2.0) -FullUri

# provide URI with SAS token during deployment
New-AzResourceGroupDeployment `
  -ResourceGroupName ExampleGroup `
  -TemplateUri $templateuri
```

For an example of using a SAS token with linked templates, see Using linked templates with Azure Resource Manager.

Next steps

- For an introduction to deploying templates, see Deploy resources with ARM templates and Azure PowerShell.
- To define parameters in template, see Authoring templates.

View deployment history with Azure Resource Manager

Article • 04/09/2023

Azure Resource Manager enables you to view your deployment history. You can examine specific operations in past deployments and see which resources were deployed. This history contains information about any errors.

The deployment history for a resource group is limited to 800 deployments. As you near the limit, deployments are automatically deleted from the history. For more information, see Automatic deletions from deployment history.

For help with resolving particular deployment errors, see Troubleshoot common Azure deployment errors.

Correlation ID and support

Each deployment has a correlation ID, which is used to track related events. If you create an Azure support request, support may ask you for the correlation ID. Support uses the correlation ID to identify the operations for the failed deployment.

The examples in this article show how to retrieve the correlation ID.

Resource group deployments

You can view details about a resource group deployment through the Azure portal, PowerShell, Azure CLI, or REST API.

Portal

1. Select the **resource group** you want to examine.

2. Select the link under **Deployments**.

3. Select one of the deployments from the deployment history.

4. A summary of the deployment is displayed, including the correlation ID.

Subscription deployments

You can view the history of deployments to a subscription.

Portal

1. Select the **subscription** you want to examine.

2. In the left pane, select **Deployments**.

3. Select one of the deployments from the deployment history.

4. A summary of the deployment is displayed, including the correlation ID.

Management group deployments

You can view the history of deployments to a management group.

Portal

1. Select the **management group** you want to examine. If you don't have sufficient permissions to view details about the management group, you won't be able to select it.

2. In the left pane, select **Deployments**.

3. Select one of the deployments from the deployment history.

4. A summary of the deployment is displayed, including the correlation ID.

Tenant deployments

You can view the history of deployments to a tenant.

Portal

The portal doesn't currently show tenant deployments.

Deployment operations and error message

Each deployment can include multiple operations. To see more details about a deployment, view the deployment operations. When a deployment fails, the deployment operations include an error message.

Portal

1. On the summary for a deployment, select **Operation details**.

2. You see the details for that step of the deployment. When an error occurs, the details include the error message.

Next steps

- For help resolve specific deployment errors, see Troubleshoot common Azure deployment errors.
- To learn about how deployments are managed in the history, see Automatic deletions from deployment history.

- To preview changes a template will make before you deploy, see ARM template deployment what-if operation.

Create Resource Manager parameter file

Article • 03/09/2023

Rather than passing parameters as inline values in your script, you can use a JSON file that contains the parameter values. This article shows how to create a parameter file that you use with a JSON template.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **parameter files**.

Parameter file

A parameter file uses the following format:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "<first-parameter-name>": {
      "value": "<first-value>"
    },
    "<second-parameter-name>": {
      "value": "<second-value>"
    }
  }
}
```

It's worth noting that the parameter file saves parameter values as plain text. For security reasons, this approach is not recommended for sensitive values such as passwords. If you must pass a parameter with a sensitive value, keep the value in a key vault. Then, in your parameter file, include a reference to the key vault. During deployment, the sensitive value is securely retrieved. For more information, see Use Azure Key Vault to pass secure parameter value during deployment.

The following parameter file includes a plain text value and a sensitive value that's stored in a key vault.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "<first-parameter-name>": {
      "value": "<first-value>"
    },
    "<second-parameter-name>": {
      "reference": {
        "keyVault": {
          "id": "<resource-id-key-vault>"
        },
        "secretName": "<secret-name>"
      }
    }
  }
}
```

For more information about using values from a key vault, see Use Azure Key Vault to pass secure parameter value during deployment.

Define parameter values

To determine how to define the parameter names and values, open your JSON template and review the `parameters` section. The following example shows the JSON template's parameters.

JSON

```
"parameters": {
  "storagePrefix": {
    "type": "string",
    "maxLength": 11
  },
  "storageAccountType": {
    "type": "string",
    "defaultValue": "Standard_LRS",
    "allowedValues": [
    "Standard_LRS",
    "Standard_GRS",
    "Standard_ZRS",
    "Premium_LRS"
    ]
  }
}
```

In the parameter file, the first detail to notice is the name of each parameter. The parameter names in your parameter file must match the parameter names in your template.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storagePrefix": {
    },
    "storageAccountType": {
    }
  }
}
```

Notice the parameter type. The parameter types in your parameter file must use the same types as your template. In this example, both parameter types are strings.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storagePrefix": {
      "value": ""
    },
    "storageAccountType": {
      "value": ""
    }
  }
}
```

Check the template for parameters with a default value. If a parameter has a default value, you can provide a value in the parameter file but it's not required. The parameter file value overrides the template's default value.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storagePrefix": {
```

```
      "value": "" // This value must be provided.
    },
    "storageAccountType": {
      "value": "" // This value is optional. Template will use default value
if not provided.
    }
  }
}
```

Check the template's allowed values and any restrictions such as maximum length. Those values specify the range of values you can provide for a parameter. In this example, `storagePrefix` can have a maximum of 11 characters and `storageAccountType` must specify an allowed value.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storagePrefix": {
      "value": "storage"
    },
    "storageAccountType": {
      "value": "Standard_ZRS"
    }
  }
}
```

> **Note**
>
> Your parameter file can only contain values for parameters that are defined in the template. If your parameter file contains extra parameters that don't match the template's parameters, you receive an error.

Parameter type formats

The following example shows the formats of different parameter types: string, integer, boolean, array, and object.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentParameters.json#",
```

```
  "contentVersion": "1.0.0.0",
  "parameters": {
    "exampleString": {
      "value": "test string"
    },
    "exampleInt": {
      "value": 4
    },
    "exampleBool": {
      "value": true
    },
    "exampleArray": {
      "value": [
        "value 1",
        "value 2"
      ]
    },
    "exampleObject": {
      "value": {
        "property1": "value1",
        "property2": "value2"
      }
    }
  }
}
```

Deploy template with parameter file

From Azure CLI you pass a local parameter file using `@` and the parameter file name. For example, `@storage.parameters.json`.

Azure CLI

```
az deployment group create \
  --name ExampleDeployment \
  --resource-group ExampleGroup \
  --template-file storage.json \
  --parameters @storage.parameters.json
```

For more information, see Deploy resources with ARM templates and Azure CLI.

From Azure PowerShell you pass a local parameter file using the `TemplateParameterFile` parameter.

Azure PowerShell

```
New-AzResourceGroupDeployment -Name ExampleDeployment -ResourceGroupName
ExampleResourceGroup `
```

```
  -TemplateFile C:\MyTemplates\storage.json `
  -TemplateParameterFile C:\MyTemplates\storage.parameters.json
```

For more information, see Deploy resources with ARM templates and Azure PowerShell.

> **Note**
>
> It's not possible to use a parameter file with the custom template blade in the portal.

> **Tip**
>
> If you're using the **Azure Resource Group project in Visual Studio**, make sure the parameter file has its **Build Action** set to **Content**.

File name

The general naming convention for the parameter file is to include *parameters* in the template name. For example, if your template is named *azuredeploy.json*, your parameter file is named *azuredeploy.parameters.json*. This naming convention helps you see the connection between the template and the parameters.

To deploy to different environments, you create more than one parameter file. When you name the parameter files, identify their use such as development and production. For example, use *azuredeploy.parameters-dev.json* and *azuredeploy.parameters-prod.json* to deploy resources.

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.

It's possible to use an external parameter file, by providing the URI to the file. When you use an external parameter file, you can't pass other values either inline or from a local file. All inline parameters are ignored. Provide all parameter values in the external file.

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-AzResourceGroupDeployment cmdlet. You're prompted to provide a value for `ResourceGroupNameFromTemplate`. To avoid this confusion, use parameter names that aren't used for deployment commands.

Next steps

- For more information about how to define parameters in a template, see Parameters in ARM templates.
- For more information about using values from a key vault, see Use Azure Key Vault to pass secure parameter value during deployment.

Use Azure Key Vault to pass secure parameter value during deployment

Article • 03/21/2023

Instead of putting a secure value (like a password) directly in your template or parameter file, you can retrieve the value from an Azure Key Vault during a deployment. You retrieve the value by referencing the key vault and secret in your parameter file. The value is never exposed because you only reference its key vault ID.

> **Important**
>
> This article focuses on how to pass a sensitive value as a template parameter. When the secret is passed as a parameter, the key vault can exist in a different subscription than the resource group you're deploying to.
>
> This article doesn't cover how to set a virtual machine property to a certificate's URL in a key vault. For a quickstart template of that scenario, see **Install a certificate from Azure Key Vault on a Virtual Machine**.

Deploy key vaults and secrets

To access a key vault during template deployment, set `enabledForTemplateDeployment` on the key vault to `true`.

If you already have a key vault, make sure it allows template deployments.

Azure CLI

Azure CLI

```
az keyvault update  --name ExampleVault --enabled-for-template-deployment true
```

To create a new key vault and add a secret, use:

Azure CLI

Azure CLI

```
az group create --name ExampleGroup --location centralus
az keyvault create \
  --name ExampleVault \
  --resource-group ExampleGroup \
  --location centralus \
  --enabled-for-template-deployment true
az keyvault secret set --vault-name ExampleVault --name
"ExamplePassword" --value "hVFkk965BuUv"
```

As the owner of the key vault, you automatically have access to create secrets. If you need to let another user create secrets, use:

Azure CLI

Azure CLI

```
az keyvault set-policy \
  --upn <user-principal-name> \
  --name ExampleVault \
  --secret-permissions set delete get list
```

The access policies aren't needed if the user is deploying a template that retrieves a secret. Add a user to the access policies only if the user needs to work directly with the secrets. The deployment permissions are defined in the next section.

For more information about creating key vaults and adding secrets, see:

- Set and retrieve a secret by using CLI
- Set and retrieve a secret by using PowerShell
- Set and retrieve a secret by using the portal
- Set and retrieve a secret by using .NET
- Set and retrieve a secret by using Node.js

Grant deployment access to the secrets

The user who deploys the template must have the `Microsoft.KeyVault/vaults/deploy/action` permission for the scope of the resource group and key vault. By checking this access, Azure Resource Manager prevents an unapproved user from accessing the secret by passing in the resource ID for the key vault. You can grant deployment access to users without granting write access to the secrets.

The Owner and Contributor roles both grant this access. If you created the key vault, you're the owner and have the permission.

For other users, grant the `Microsoft.KeyVault/vaults/deploy/action` permission. The following procedure shows how to create a role with the minimum permission, and assign it to a user.

1. Create a custom role definition JSON file:

JSON

```
{
  "Name": "Key Vault resource manager template deployment operator",
  "IsCustom": true,
  "Description": "Lets you deploy a resource manager template with the
access to the secrets in the Key Vault.",
  "Actions": [
    "Microsoft.KeyVault/vaults/deploy/action"
  ],
  "NotActions": [],
  "DataActions": [],
  "NotDataActions": [],
  "AssignableScopes": [
    "/subscriptions/00000000-0000-0000-0000-000000000000"
  ]
}
```

Replace "00000000-0000-0000-0000-000000000000" with the subscription ID.

2. Create the new role using the JSON file:

Azure CLI

Azure CLI

```
az role definition create --role-definition "<path-to-role-file>"
az role assignment create \
  --role "Key Vault resource manager template deployment operator"
\
  --scope /subscriptions/<Subscription-
id>/resourceGroups/<resource-group-name> \
  --assignee <user-principal-name> \
  --resource-group ExampleGroup
```

The samples assign the custom role to the user on the resource group level.

When using a key vault with the template for a Managed Application, you must grant access to the **Appliance Resource Provider** service principal. For more information, see Access Key Vault secret when deploying Azure Managed Applications.

Reference secrets with static ID

With this approach, you reference the key vault in the parameter file, not the template. The following image shows how the parameter file references the secret and passes that value to the template.

Tutorial: Integrate Azure Key Vault in Resource Manager Template deployment uses this method.

The following template deploys a SQL server that includes an administrator password. The password parameter is set to a secure string. But the template doesn't specify where that value comes from.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "adminLogin": {
      "type": "string"
    },
    "adminPassword": {
      "type": "securestring"
    },
    "sqlServerName": {
      "type": "string"
    }
  },
  "resources": [
    {
      "type": "Microsoft.Sql/servers",
      "apiVersion": "2015-05-01-preview",
      "name": "[parameters('sqlServerName')]",
      "location": "[resourceGroup().location]",
      "tags": {},
      "properties": {
        "administratorLogin": "[parameters('adminLogin')]",
        "administratorLoginPassword": "[parameters('adminPassword')]",
```

```
        "version": "12.0"
      }
    }
  ],
  "outputs": {
  }
}
```

Now, create a parameter file for the preceding template. In the parameter file, specify a parameter that matches the name of the parameter in the template. For the parameter value, reference the secret from the key vault. You reference the secret by passing the resource identifier of the key vault and the name of the secret:

In the following parameter file, the key vault secret must already exist, and you provide a static value for its resource ID.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "adminLogin": {
      "value": "exampleadmin"
    },
    "adminPassword": {
      "reference": {
        "keyVault": {
          "id": "/subscriptions/<subscription-id>/resourceGroups/<rg-
name>/providers/Microsoft.KeyVault/vaults/<vault-name>"
        },
        "secretName": "ExamplePassword"
      }
    },
    "sqlServerName": {
      "value": "<your-server-name>"
    }
  }
}
```

If you need to use a version of the secret other than the current version, include the `secretVersion` property.

JSON

```
"secretName": "ExamplePassword",
"secretVersion": "cd91b2b7e10e492ebb870a6ee0591b68"
```

Deploy the template and pass in the parameter file:

Azure CLI

Azure CLI

```
az group create --name SqlGroup --location westus2
az deployment group create \
  --resource-group SqlGroup \
  --template-uri <template-file-URI> \
  --parameters <parameter-file>
```

Reference secrets with dynamic ID

The previous section showed how to pass a static resource ID for the key vault secret from the parameter. In some scenarios, you need to reference a key vault secret that varies based on the current deployment. Or you may want to pass parameter values to the template rather than create a reference parameter in the parameter file. The solution is to dynamically generate the resource ID for a key vault secret by using a linked template.

You can't dynamically generate the resource ID in the parameters file because template expressions aren't allowed in the parameters file.

In your parent template, you add the nested template and pass in a parameter that contains the dynamically generated resource ID. The following image shows how a parameter in the linked template references the secret.

The following template dynamically creates the key vault ID and passes it as a parameter.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
      "location": {
        "type": "string",
        "defaultValue": "[resourceGroup().location]",
        "metadata": {
          "description": "The location where the resources will be
deployed."
        }
      },
      "vaultName": {
        "type": "string",
        "metadata": {
          "description": "The name of the keyvault that contains the
secret."
        }
      },
      "secretName": {
        "type": "string",
        "metadata": {
          "description": "The name of the secret."
        }
      },
      "vaultResourceGroupName": {
        "type": "string",
        "metadata": {
          "description": "The name of the resource group that contains the
keyvault."
        }
      },
      "vaultSubscription": {
        "type": "string",
        "defaultValue": "[subscription().subscriptionId]",
        "metadata": {
          "description": "The name of the subscription that contains the
keyvault."
        }
      }
  },
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2020-10-01",
      "name": "dynamicSecret",
      "properties": {
        "mode": "Incremental",
        "expressionEvaluationOptions": {
          "scope": "inner"
        },
        "template": {
          "$schema": "https://schema.management.azure.com/schemas/2019-04-
```

```
01/deploymentTemplate.json#",
          "contentVersion": "1.0.0.0",
          "parameters": {
            "adminLogin": {
              "type": "string"
            },
            "adminPassword": {
              "type": "securestring"
            },
            "location": {
              "type": "string"
            }
          },
          "variables": {
            "sqlServerName": "[concat('sql-',
uniqueString(resourceGroup().id, 'sql'))]"
          },
          "resources": [
            {
              "type": "Microsoft.Sql/servers",
              "apiVersion": "2018-06-01-preview",
              "name": "[variables('sqlServerName')]",
              "location": "[parameters('location')]",
              "properties": {
                "administratorLogin": "[parameters('adminLogin')]",
                "administratorLoginPassword": "
[parameters('adminPassword')]"
              }
            }
          ],
          "outputs": {
            "sqlFQDN": {
              "type": "string",
              "value": "
[reference(variables('sqlServerName')).fullyQualifiedDomainName]"
            }
          }
        },
        "parameters": {
          "location": {
            "value": "[parameters('location')]"
          },
          "adminLogin": {
            "value": "ghuser"
          },
          "adminPassword": {
            "reference": {
              "keyVault": {
                "id": "[resourceId(parameters('vaultSubscription'),
parameters('vaultResourceGroupName'), 'Microsoft.KeyVault/vaults',
parameters('vaultName'))]"
              },
              "secretName": "[parameters('secretName')]"
            }
          }
```

```
        }
      }
    }
  ],
  "outputs": {
  }
}
```

Next steps

- For general information about key vaults, see What is Azure Key Vault?
- For complete examples of referencing key secrets, see key vault examples on GitHub.
- For a Learn module that covers passing a secure value from a key vault, see Manage complex cloud deployments by using advanced ARM template features.

Use Azure portal to export a template

Article • 09/01/2021

To assist with creating Azure Resource Manager templates, you can export a template from existing resources. The exported template helps you understand the JSON syntax and properties that deploy your resources. To automate future deployments, start with the exported template and modify it for your scenario. The export template process attempts to create a usable template. However, most exported templates require some modifications before they can be used to deploy Azure resources.

Resource Manager enables you to pick one or more resources for exporting to a template. You can focus on exactly the resources you need in the template.

This article shows how to export templates through the **portal**. For other options, see:

- Export template with Azure CLI
- Export template with Azure PowerShell
- REST API export from resource group and REST API export from deployment history.

Choose the right export option

There are two ways to export a template:

- **Export from resource group or resource**. This option generates a new template from existing resources. The exported template is a "snapshot" of the current state of the resource group. You can export an entire resource group or specific resources within that resource group.

- **Save from history**. This option retrieves an exact copy of a template used for deployment. You specify the deployment from the deployment history.

Depending on the option you choose, the exported templates have different qualities.

From resource group or resource	From history
Template is autogenerated. You'll probably want to improve the code before deploying it.	Template is an exact copy of the one created by the template author. It's ready to be redeployed without any changes.

From resource group or resource	From history
Template is snapshot of the resources' current state. It includes any manual changes you made after deployment.	Template only shows state of resources at the time of deployment. Any manual changes you made after deployment aren't included.
You can select which resources from a resource group to export.	All resources for a specific deployment are included. You can't pick a subset of those resources or add resources that were added at a different time.
Template includes all properties for the resources, including some properties you wouldn't normally set during deployment. You might want to remove or clean up these properties before reusing the template.	Template includes only the properties needed for the deployment. The template is more sparse and easier to read.
Template probably doesn't include all of the parameters you need for reuse. Most property values are hard-coded in the template. To redeploy the template in other environments, you need to add parameters that increase the ability to configure the resources. You can unselect **Include parameters** so that you can author your own parameters.	Template includes parameters that make it easy to redeploy in different environments.

Export the template from a resource group or resource, when:

- You need to capture changes to the resources that were made after the original deployment.
- You want to select which resources are exported.
- The resources weren't created with a template.

Export the template from the history, when:

- You want an easy-to-reuse template.
- You don't need to include changes you made after the original deployment.

Limitations

Export is not guaranteed to succeed. Export is not a reliable way to turn pre-existing resources into templates that are usable in production. It is better to create resources from scratch using hand-written Bicep file, ARM template or terraform.

When exporting from a resource group or resource, the exported template is generated from the published schemas for each resource type. Occasionally, the schema doesn't have the latest version for a resource type. Check your exported template to make sure it includes the properties you need. If necessary, edit the exported template to use the API version you need.

Some password parameters might be missing from the exported templates. You need to check template reference, and manually add these parameters before you can use the templates to deploy resources.

The export template feature doesn't support exporting Azure Data Factory resources. To learn about how you can export Data Factory resources, see Copy or clone a data factory in Azure Data Factory.

To export resources created through classic deployment model, you must migrate them to the Resource Manager deployment model.

If you get a warning when exporting a template that indicates a resource type wasn't exported, you can still discover the properties for that resource. For resource properties, see template reference. You can also look at the Azure REST API for the resource type.

There's a limit of 200 resources in the resource group you create the exported template for. If you attempt to export a resource group that has more than 200 resources, the error message `Export template is not supported for resource groups more than 200 resources` is shown.

Export template from a resource group

To export one or more resources from a resource group:

1. Select the resource group that contains the resources you want to export.

2. Select one or more resources by selecting the checkboxes. To select all, select the checkbox on the left of **Name**. The **Export template** menu item only becomes enabled after you've selected at least one resource.

On the screenshot, only the storage account is selected.

3. Select **Export template**.

4. The exported template is displayed, and is available to download and deploy.

Include parameters is selected by default. When selected, all template parameters will be included when the template is generated. If you'd like to author your own parameters, toggle this checkbox to not include them.

Export template from a resource

To export one resource:

1. Select the resource group containing the resource you want to export.

2. Select the resource that you want to export to open the resource.

3. For that resource, select **Export template** in the left pane.

4. The exported template is displayed, and is available to download and deploy. The template only contains the single resource. **Include parameters** is selected by default. When selected, all template parameters will be included when the template is generated. If you'd like to author your own parameters, toggle this checkbox to not include them.

Download template before deployment

The portal has the option of downloading a template before deploying it. This option isn't available through PowerShell or Azure CLI.

1. Select the Azure service you want to deploy.

2. Fill in the values for the new service.

3. After passing validation, but before starting the deployment, select **Download a template for automation**.

4. The template is displayed and is available for download and deploy.

Export template after deployment

You can export the template that was used to deploy existing resources. The template you get is exactly the one that was used for deployment.

1. Select the resource group you want to export.

2. Select the link under **Deployments**.

3. Select one of the deployments from the deployment history.

4. Select **Template**. The template used for this deployment is displayed, and is available for download.

Next steps

- Learn how to export templates with Azure CLI, Azure PowerShell, or REST API.
- To learn the Resource Manager template syntax, see Understand the structure and syntax of Azure Resource Manager templates.

- To learn how to develop templates, see the step-by-step tutorials.

Use Azure CLI to export a template

Article • 03/24/2022

To assist with creating Azure Resource Manager templates, you can export a template from existing resources. The exported template helps you understand the JSON syntax and properties that deploy your resources. To automate future deployments, start with the exported template and modify it for your scenario. The export template process attempts to create a usable template. However, most exported templates require some modifications before they can be used to deploy Azure resources.

Resource Manager enables you to pick one or more resources for exporting to a template. You can focus on exactly the resources you need in the template.

This article shows how to export templates through **Azure CLI**. For other options, see:

- Export template with Azure portal
- Export template with Azure PowerShell
- REST API export from resource group and REST API export from deployment history.

Choose the right export option

There are two ways to export a template:

- **Export from resource group or resource**. This option generates a new template from existing resources. The exported template is a "snapshot" of the current state of the resource group. You can export an entire resource group or specific resources within that resource group.

- **Save from history**. This option retrieves an exact copy of a template used for deployment. You specify the deployment from the deployment history.

Depending on the option you choose, the exported templates have different qualities.

From resource group or resource	From history
Template is autogenerated. You'll probably want to improve the code before deploying it.	Template is an exact copy of the one created by the template author. It's ready to be redeployed without any changes.

From resource group or resource	From history
Template is snapshot of the resources' current state. It includes any manual changes you made after deployment.	Template only shows state of resources at the time of deployment. Any manual changes you made after deployment aren't included.
You can select which resources from a resource group to export.	All resources for a specific deployment are included. You can't pick a subset of those resources or add resources that were added at a different time.
Template includes all properties for the resources, including some properties you wouldn't normally set during deployment. You might want to remove or clean up these properties before reusing the template.	Template includes only the properties needed for the deployment. The template is more sparse and easier to read.
Template probably doesn't include all of the parameters you need for reuse. Most property values are hard-coded in the template. To redeploy the template in other environments, you need to add parameters that increase the ability to configure the resources. You can unselect **Include parameters** so that you can author your own parameters.	Template includes parameters that make it easy to redeploy in different environments.

Export the template from a resource group or resource, when:

- You need to capture changes to the resources that were made after the original deployment.
- You want to select which resources are exported.
- The resources weren't created with a template.

Export the template from the history, when:

- You want an easy-to-reuse template.
- You don't need to include changes you made after the original deployment.

Limitations

Export is not guaranteed to succeed. Export is not a reliable way to turn pre-existing resources into templates that are usable in production. It is better to create resources from scratch using hand-written Bicep file, ARM template or terraform.

When exporting from a resource group or resource, the exported template is generated from the published schemas for each resource type. Occasionally, the schema doesn't have the latest version for a resource type. Check your exported template to make sure it includes the properties you need. If necessary, edit the exported template to use the API version you need.

Some password parameters might be missing from the exported templates. You need to check template reference, and manually add these parameters before you can use the templates to deploy resources.

The export template feature doesn't support exporting Azure Data Factory resources. To learn about how you can export Data Factory resources, see Copy or clone a data factory in Azure Data Factory.

To export resources created through classic deployment model, you must migrate them to the Resource Manager deployment model.

If you get a warning when exporting a template that indicates a resource type wasn't exported, you can still discover the properties for that resource. For resource properties, see template reference. You can also look at the Azure REST API for the resource type.

There's a limit of 200 resources in the resource group you create the exported template for. If you attempt to export a resource group that has more than 200 resources, the error message `Export template is not supported for resource groups more than 200 resources` is shown.

Export template from a resource group

After setting up your resource group successfully, you can export an Azure Resource Manager template for the resource group.

To export all resources in a resource group, use az group export and provide the resource group name.

Azure CLI

```
az group export --name demoGroup
```

The script displays the template on the console. To save to a file, use:

Azure CLI

```
az group export --name demoGroup > exportedtemplate.json
```

Instead of exporting all resources in the resource group, you can select which resources to export.

To export one resource, pass that resource ID.

Azure CLI

```
storageAccountID=$(az resource show --resource-group demoGroup --name
demostg --resource-type Microsoft.Storage/storageAccounts --query id --
output tsv)
az group export --resource-group demoGroup --resource-ids $storageAccountID
```

To export more than one resource, pass the space-separated resource IDs. To export all resources, don't specify this argument or supply "*".

Azure CLI

```
az group export --resource-group <resource-group-name> --resource-ids
$storageAccountID1 $storageAccountID2
```

When exporting the template, you can specify whether parameters are used in the template. By default, parameters for resource names are included but they don't have a default value.

JSON

```
"parameters": {
  "serverfarms_demoHostPlan_name": {
    "type": "String"
  },
  "sites_webSite3bwt23ktvdo36_name": {
    "type": "String"
  }
}
```

If you use the `--skip-resource-name-params` parameter when exporting the template, parameters for resource names aren't included in the template. Instead, the resource name is set directly on the resource to its current value. You can't customize the name during deployment.

JSON

```
"resources": [
  {
    "type": "Microsoft.Web/serverfarms",
    "apiVersion": "2016-09-01",
    "name": "demoHostPlan",
```

```
    ...
  }
]
```

If you use the `--include-parameter-default-value` parameter when exporting the template, the template parameter includes a default value that is set to the current value. You can either use that default value or overwrite the default value by passing in a different value.

JSON

```
"parameters": {
  "serverfarms_demoHostPlan_name": {
    "defaultValue": "demoHostPlan",
    "type": "String"
  },
  "sites_webSite3bwt23ktvdo36_name": {
    "defaultValue": "webSite3bwt23ktvdo36",
    "type": "String"
  }
}
```

Save template from deployment history

You can save a template from a deployment in the deployment history. The template you get is exactly the one that was used for deployment.

To get a template from a resource group deployment, use the az deployment group export command. You specify the name of the deployment to retrieve. For help with getting the name of a deployment, see View deployment history with Azure Resource Manager.

Azure CLI

```
az deployment group export --resource-group demoGroup --name demoDeployment
```

The template is displayed in the console. To save the file, use:

Azure CLI

```
az deployment group export --resource-group demoGroup --name demoDeployment
> demoDeployment.json
```

To get templates deployed at other levels, use:

- az deployment sub export for deployments to subscriptions
- az deployment mg export for deployments to management groups
- az deployment tenant export for deployments to tenants

Next steps

- Learn how to export templates with Azure portal, Azure PowerShell, or REST API.
- To learn the Resource Manager template syntax, see Understand the structure and syntax of Azure Resource Manager templates.
- To learn how to develop templates, see the step-by-step tutorials.

Use Azure PowerShell to export a template

Article • 03/09/2023

To assist with creating Azure Resource Manager templates, you can export a template from existing resources. The exported template helps you understand the JSON syntax and properties that deploy your resources. To automate future deployments, start with the exported template and modify it for your scenario. The export template process attempts to create a usable template. However, most exported templates require some modifications before they can be used to deploy Azure resources.

Resource Manager enables you to pick one or more resources for exporting to a template. You can focus on exactly the resources you need in the template.

This article shows how to export templates through **Azure PowerShell**. For other options, see:

- Export template with Azure CLI
- Export template with Azure portal
- REST API export from resource group and REST API export from deployment history.

Choose the right export option

There are two ways to export a template:

- **Export from resource group or resource**. This option generates a new template from existing resources. The exported template is a "snapshot" of the current state of the resource group. You can export an entire resource group or specific resources within that resource group.

- **Save from history**. This option retrieves an exact copy of a template used for deployment. You specify the deployment from the deployment history.

Depending on the option you choose, the exported templates have different qualities.

From resource group or resource	From history

From resource group or resource	From history
Template is autogenerated. You'll probably want to improve the code before deploying it.	Template is an exact copy of the one created by the template author. It's ready to be redeployed without any changes.
Template is snapshot of the resources' current state. It includes any manual changes you made after deployment.	Template only shows state of resources at the time of deployment. Any manual changes you made after deployment aren't included.
You can select which resources from a resource group to export.	All resources for a specific deployment are included. You can't pick a subset of those resources or add resources that were added at a different time.
Template includes all properties for the resources, including some properties you wouldn't normally set during deployment. You might want to remove or clean up these properties before reusing the template.	Template includes only the properties needed for the deployment. The template is more sparse and easier to read.
Template probably doesn't include all of the parameters you need for reuse. Most property values are hard-coded in the template. To redeploy the template in other environments, you need to add parameters that increase the ability to configure the resources. You can unselect **Include parameters** so that you can author your own parameters.	Template includes parameters that make it easy to redeploy in different environments.

Export the template from a resource group or resource, when:

- You need to capture changes to the resources that were made after the original deployment.
- You want to select which resources are exported.
- The resources weren't created with a template.

Export the template from the history, when:

- You want an easy-to-reuse template.
- You don't need to include changes you made after the original deployment.

Limitations

Export is not guaranteed to succeed. Export is not a reliable way to turn pre-existing resources into templates that are usable in production. It is better to create resources from scratch using hand-written Bicep file, ARM template or terraform.

When exporting from a resource group or resource, the exported template is generated from the published schemas for each resource type. Occasionally, the schema doesn't have the latest version for a resource type. Check your exported template to make sure it includes the properties you need. If necessary, edit the exported template to use the API version you need.

Some password parameters might be missing from the exported templates. You need to check template reference, and manually add these parameters before you can use the templates to deploy resources.

The export template feature doesn't support exporting Azure Data Factory resources. To learn about how you can export Data Factory resources, see Copy or clone a data factory in Azure Data Factory.

To export resources created through classic deployment model, you must migrate them to the Resource Manager deployment model.

If you get a warning when exporting a template that indicates a resource type wasn't exported, you can still discover the properties for that resource. For resource properties, see template reference. You can also look at the Azure REST API for the resource type.

There's a limit of 200 resources in the resource group you create the exported template for. If you attempt to export a resource group that has more than 200 resources, the error message `Export template is not supported for resource groups more than 200 resources` is shown.

Export template from a resource group

After setting up your resource group, you can export an Azure Resource Manager template for the resource group.

To export all resources in a resource group, use the Export-AzResourceGroup cmdlet and provide the resource group name.

Azure PowerShell

```
Export-AzResourceGroup -ResourceGroupName demoGroup
```

It saves the template as a local file.

Instead of exporting all resources in the resource group, you can select which resources to export.

To export one resource, pass that resource ID.

Azure PowerShell

```
$resource = Get-AzResource `
  -ResourceGroupName <resource-group-name> `
  -ResourceName <resource-name> `
  -ResourceType <resource-type>
Export-AzResourceGroup `
  -ResourceGroupName <resource-group-name> `
  -Resource $resource.ResourceId
```

To export more than one resource, pass the resource IDs in an array.

Azure PowerShell

```
Export-AzResourceGroup `
  -ResourceGroupName <resource-group-name> `
  -Resource @($resource1.ResourceId, $resource2.ResourceId)
```

When exporting the template, you can specify whether parameters are used in the template. By default, parameters for resource names are included but they don't have a default value.

JSON

```
"parameters": {
  "serverfarms_demoHostPlan_name": {
    "type": "String"
  },
  "sites_webSite3bwt23ktvdo36_name": {
    "type": "String"
  }
}
```

If you use the `-SkipResourceNameParameterization` parameter when exporting the template, parameters for resource names aren't included in the template. Instead, the resource name is set directly on the resource to its current value. You can't customize the name during deployment.

JSON

```
"resources": [
  {
```

```
    "type": "Microsoft.Web/serverfarms",
    "apiVersion": "2016-09-01",
    "name": "demoHostPlan",
    ...
  }
]
```

If you use the `-IncludeParameterDefaultValue` parameter when exporting the template, the template parameter includes a default value that is set to the current value. You can either use that default value or overwrite the default value by passing in a different value.

JSON

```
"parameters": {
  "serverfarms_demoHostPlan_name": {
    "defaultValue": "demoHostPlan",
    "type": "String"
  },
  "sites_webSite3bwt23ktvdo36_name": {
    "defaultValue": "webSite3bwt23ktvdo36",
    "type": "String"
  }
}
```

Save template from deployment history

You can save a template from a deployment in the deployment history. The template you get is exactly the one that was used for deployment.

To get a template from a resource group deployment, use the Save-AzResourceGroupDeploymentTemplate cmdlet. You specify the name of the deployment to retrieve. For help with getting the name of a deployment, see View deployment history with Azure Resource Manager.

Azure PowerShell

```
Save-AzResourceGroupDeploymentTemplate -ResourceGroupName demoGroup -
DeploymentName demoDeployment
```

The template is saved as a local file with the name of the deployment.

To get templates deployed at other levels, use:

- Save-AzDeploymentTemplate for deployments to subscriptions

- Save-AzManagementGroupDeploymentTemplate for deployments to management groups
- Save-AzTenantDeploymentTemplate for deployments to tenants

Next steps

- Learn how to export templates with Azure CLI, Azure portal, or REST API.
- To learn the Resource Manager template syntax, see Understand the structure and syntax of Azure Resource Manager templates.
- To learn how to develop templates, see the step-by-step tutorials.

Use ARM template test toolkit

Article • 03/21/2023

The Azure Resource Manager template (ARM template) test toolkit checks whether your template uses recommended practices. When your template isn't compliant with recommended practices, it returns a list of warnings with the suggested changes. By using the test toolkit, you can learn how to avoid common problems in template development. This article describes how to run the test toolkit and how to add or remove tests. For more information about how to run tests or how to run a specific test, see Test parameters.

The toolkit is a set of PowerShell scripts that can be run from a command in PowerShell or CLI. These tests are recommendations but not requirements. You can decide which tests are relevant to your goals and customize which tests are run.

The toolkit contains four sets of tests:

- Test cases for ARM templates
- Test cases for parameter files
- Test cases for createUiDefinition.json
- Test cases for all files

> **Note**
>
> The test toolkit is only available for ARM templates. To validate Bicep files, use the **Bicep linter**.

Training resources

To learn more about the ARM template test toolkit, and for hands-on guidance, see Validate Azure resources by using the ARM Template Test Toolkit.

Install on Windows

1. If you don't already have PowerShell, install PowerShell on Windows.
2. Download the latest .zip file for the test toolkit and extract it.
3. Start PowerShell.

4. Navigate to the folder where you extracted the test toolkit. Within that folder, navigate to *arm-ttk* folder.

5. If your execution policy blocks scripts from the Internet, you need to unblock the script files. Make sure you're in the *arm-ttk* folder.

```
PowerShell
Get-ChildItem *.ps1, *.psd1, *.ps1xml, *.psm1 -Recurse | Unblock-File
```

6. Import the module.

```
PowerShell
Import-Module .\arm-ttk.psd1
```

7. To run the tests, use the following command:

```
PowerShell
Test-AzTemplate -TemplatePath \path\to\template
```

Install on Linux

1. If you don't already have PowerShell, install PowerShell on Linux.

2. Download the latest .zip file for the test toolkit and extract it.

3. Start PowerShell.

```
Bash
pwsh
```

4. Navigate to the folder where you extracted the test toolkit. Within that folder, navigate to *arm-ttk* folder.

5. If your execution policy blocks scripts from the Internet, you need to unblock the script files. Make sure you're in the *arm-ttk* folder.

```
PowerShell
Get-ChildItem *.ps1, *.psd1, *.ps1xml, *.psm1 -Recurse | Unblock-File
```

6. Import the module.

```
PowerShell
Import-Module ./arm-ttk.psd1
```

7. To run the tests, use the following command:

```
PowerShell
Test-AzTemplate -TemplatePath /path/to/template
```

Install on macOS

1. If you don't already have PowerShell, install PowerShell on macOS.

2. Install `coreutils`:

```
Bash
brew install coreutils
```

3. Download the latest .zip file for the test toolkit and extract it.

4. Start PowerShell.

```
Bash
pwsh
```

5. Navigate to the folder where you extracted the test toolkit. Within that folder, navigate to *arm-ttk* folder.

6. If your execution policy blocks scripts from the Internet, you need to unblock the script files. Make sure you're in the *arm-ttk* folder.

```
PowerShell
Get-ChildItem *.ps1, *.psd1, *.ps1xml, *.psm1 -Recurse | Unblock-File
```

7. Import the module.

```
PowerShell
```

```
Import-Module ./arm-ttk.psd1
```

8. To run the tests, use the following command:

PowerShell

```
Test-AzTemplate -TemplatePath /path/to/template
```

Result format

Tests that pass are displayed in **green** and prefaced with `[+]`.

Tests that fail are displayed in **red** and prefaced with `[-]`.

Tests with a warning are displayed in **yellow** and prefaced with `[?]`.

```
  deploymentTemplate
    [+] adminUsername Should Not Be A Literal (6 ms)
    [+] apiVersions Should Be Recent In Reference Functions (9 ms)
    [-] apiVersions Should Be Recent (6 ms)
        Api versions must be the latest or under 2 years old (730 days) - API version 2019-06-01 of
Microsoft.Storage/storageAccounts is 760 days old
        Valid Api Versions:
        2021-04-01
        2021-02-01
        2021-01-01
        2020-08-01-preview

    [+] artifacts parameter (4 ms)
    [+] CommandToExecute Must Use ProtectedSettings For Secrets (9 ms)
    [+] DependsOn Best Practices (5 ms)
    [+] Deployment Resources Must Not Be Debug (6 ms)
    [+] DeploymentTemplate Must Not Contain Hardcoded Uri (4 ms)
    [?] DeploymentTemplate Schema Is Correct (6 ms)
        Template is using schema version '2015-01-01' which has been deprecated and is no longer
maintained.
```

The text results are:

PowerShell

```
deploymentTemplate
[+] adminUsername Should Not Be A Literal (6 ms)
[+] apiVersions Should Be Recent In Reference Functions (9 ms)
[-] apiVersions Should Be Recent (6 ms)
    Api versions must be the latest or under 2 years old (730 days) - API
version 2019-06-01 of
    Microsoft.Storage/storageAccounts is 760 days old
    Valid Api Versions:
    2021-04-01
    2021-02-01
    2021-01-01
    2020-08-01-preview
```

```
[+] artifacts parameter (4 ms)
[+] CommandToExecute Must Use ProtectedSettings For Secrets (9 ms)
[+] DependsOn Best Practices (5 ms)
[+] Deployment Resources Must Not Be Debug (6 ms)
[+] DeploymentTemplate Must Not Contain Hardcoded Uri (4 ms)
[?] DeploymentTemplate Schema Is Correct (6 ms)
    Template is using schema version '2015-01-01' which has been deprecated
and is no longer
    maintained.
```

Test parameters

When you provide the `-TemplatePath` parameter, the toolkit looks in that folder for a template named *azuredeploy.json* or *maintemplate.json*. It tests this template first and then tests all other templates in the folder and its subfolders. The other templates are tested as linked templates. If your path includes a file named createUiDefinition.json, it runs tests that are relevant to UI definition. Tests are also run for parameter files and all JSON files in the folder.

PowerShell

```
Test-AzTemplate -TemplatePath $TemplateFolder
```

To test one file in that folder, add the `-File` parameter. However, the folder must still have a main template named *azuredeploy.json* or *maintemplate.json*.

PowerShell

```
Test-AzTemplate -TemplatePath $TemplateFolder -File cdn.json
```

By default, all tests are run. To specify individual tests to run, use the `-Test` parameter and provide the test name. For the test names, see ARM templates, parameter files, createUiDefinition.json, and all files.

PowerShell

```
Test-AzTemplate -TemplatePath $TemplateFolder -Test "Resources Should Have
Location"
```

Customize tests

You can customize the default tests or create your own tests. If you want to permanently remove a test, delete the *.test.ps1* file from the folder.

The toolkit has four folders that contain the default tests that are run for specific file types:

- ARM templates: *\arm-ttk\testcases\deploymentTemplate*
- Parameter files: *\arm-ttk\testcases\deploymentParameters*
- createUiDefinition.json: *\arm-ttk\testcases\CreateUIDefinition*
- All files: *\arm-ttk\testcases\AllFiles*

Add a custom test

To add your own test, create a file with the naming convention: *Your-Custom-Test-Name.test.ps1*.

The test can get the template as an object parameter or a string parameter. Typically, you use one or the other, but you can use both.

Use the object parameter when you need to get a section of the template and iterate through its properties. A test that uses the object parameter has the following format:

PowerShell

```
param(
  [Parameter(Mandatory=$true,Position=0)]
  [PSObject]
  $TemplateObject
)

# Implement test logic that evaluates parts of the template.
# Output error with: Write-Error -Message
```

The template object has the following properties:

- `$schema`
- `contentVersion`
- `parameters`
- `variables`
- `resources`
- `outputs`

For example, you can get the collection of parameters with `$TemplateObject.parameters`.

Use the string parameter when you need to do a string operation on the whole template. A test that uses the string parameter has the following format:

PowerShell

```
param(
  [Parameter(Mandatory)]
  [string]
  $TemplateText
)

# Implement test logic that performs string operations.
# Output error with: Write-Error -Message
```

For example, you can run a regular expression of the string parameter to see if a specific syntax is used.

To learn more about implementing the test, look at the other tests in that folder.

Validate templates for Azure Marketplace

To publish an offering to Azure Marketplace, use the test toolkit to validate the templates. When your templates pass the validation tests, Azure Marketplace will approve your offering more quickly. If they fail the tests, the offering will fail certification.

> **Important**
>
> The Marketplace tests were added in July 2022. Update your module if you have an earlier version.

Run the tests in your environment

After installing the toolkit and importing the module, run the following cmdlet to test your package:

PowerShell

```
Test-AzMarketplacePackage -TemplatePath "Path to the unzipped package
folder"
```

Interpret the results

The tests return results in two sections. The first section includes the tests that are **mandatory**. The results of these tests are displayed in the summary section.

> **Important**
>
> You must fix any results in red before the Marketplace offer is accepted. We recommend fixing any results that return in yellow.

The text results are:

PowerShell

```
Validating nestedtemplates\AzDashboard.json
    [+] adminUsername Should Not Be A Literal (210 ms)
    [+] artifacts parameter (3 ms)
    [+] CommandToExecute Must Use ProtectedSettings For Secrets (201 ms)
    [+] Deployment Resources Must Not Be Debug (160 ms)
    [+] DeploymentTemplate Must Not Contain Hardcoded Url (13 ms)
    [+] Location Should Not Be Hardcoded (31 ms)
    [+] Min and Max Value Are Numbers (4 ms)
    [+] Outputs Must Not Contain Secrets (9 ms)
    [+] Password params must be secure (3 ms)
    [+] Resources Should Have Location (2 ms)
    [+] Resources Should Not Be Ambiguous (2 ms)
    [+] Secure Params In Nested Deployments (205 ms)
    [+] Secure String Parameters Cannot Have Default (3 ms)
    [+] URIs Should Be Properly Constructed (190 ms)
    [+] Variables Must Be Referenced (9 ms)
    [+] Virtual Machines Should Not Be Preview (173 ms)
    [+] VM Size Should Be A Parameter (165 ms)
Pass : 99
Fail : 3
Total: 102
Validating StartStopV2mkpl_1.0.09302021\anothertemplate.json
    [?] Parameters Must Be Referenced (86 ms)
        Unreferenced parameter: resourceGroupName
        Unreferenced parameter: location
        Unreferenced parameter: azureFunctionAppName
        Unreferenced parameter: applicationInsightsName
        Unreferenced parameter: applicationInsightsRegion
```

The section below the summary section includes test failures that can be interpreted as warnings. Fixing the failures is optional but highly recommended. The failures often point to common issues that cause failures when a customer installs your offer.

To fix your tests, follow the test case applicable to you:

- ARM template test case

- All files test case
- CreateUiDefinition test case

Submit the offer

After making any necessary fixes, rerun the test toolkit. Before submitting your offer to Azure Marketplace, make sure you have zero failures.

Integrate with Azure Pipelines

You can add the test toolkit to your Azure Pipeline. With a pipeline, you can run the test every time the template is updated, or run it as part of your deployment process.

The easiest way to add the test toolkit to your pipeline is with third-party extensions. The following two extensions are available:

- Run ARM template TTK Tests
- ARM Template Tester

Or, you can implement your own tasks. The following example shows how to download the test toolkit.

For Release Pipeline:

JSON

```
{
  "environment": {},
  "enabled": true,
  "continueOnError": false,
  "alwaysRun": false,
  "displayName": "Download TTK",
  "timeoutInMinutes": 0,
  "condition": "succeeded()",
  "task": {
    "id": "e213ff0f-5d5c-4791-802d-52ea3e7be1f1",
    "versionSpec": "2.*",
    "definitionType": "task"
  },
  "inputs": {
    "targetType": "inline",
    "filePath": "",
    "arguments": "",
    "script": "New-Item '$(ttk.folder)' -ItemType Directory\nInvoke-
WebRequest -uri '$(ttk.uri)' -OutFile
\"$(ttk.folder)/$(ttk.asset.filename)\" -Verbose\nGet-ChildItem
'$(ttk.folder)' -Recurse\n\nWrite-Host \"Expanding files...\"\nExpand-
Archive -Path '$(ttk.folder)/*.zip' -DestinationPath '$(ttk.folder)' -
```

```
Verbose\n\nWrite-Host \"Expanded files found:\"\nGet-ChildItem
'$(ttk.folder)' -Recurse",
    "errorActionPreference": "stop",
    "failOnStderr": "false",
    "ignoreLASTEXITCODE": "false",
    "pwsh": "true",
    "workingDirectory": ""
  }
}
```

For Pipeline YAML definition:

YAML

```
- pwsh: |
   New-Item '$(ttk.folder)' -ItemType Directory
   Invoke-WebRequest -uri '$(ttk.uri)' -OutFile
"$(ttk.folder)/$(ttk.asset.filename)" -Verbose
   Get-ChildItem '$(ttk.folder)' -Recurse

   Write-Host "Expanding files..."
   Expand-Archive -Path '$(ttk.folder)/*.zip' -DestinationPath
'$(ttk.folder)' -Verbose

   Write-Host "Expanded files found:"
   Get-ChildItem '$(ttk.folder)' -Recurse
  displayName: 'Download TTK'
```

The next example shows how to run the tests.

For Release Pipeline:

JSON

```
{
  "environment": {},
  "enabled": true,
  "continueOnError": true,
  "alwaysRun": false,
  "displayName": "Run Best Practices Tests",
  "timeoutInMinutes": 0,
  "condition": "succeeded()",
  "task": {
    "id": "e213ff0f-5d5c-4791-802d-52ea3e7be1f1",
    "versionSpec": "2.*",
    "definitionType": "task"
  },
  "inputs": {
    "targetType": "inline",
    "filePath": "",
    "arguments": "",
```

```
    "script": "Import-Module $(ttk.folder)/arm-ttk/arm-ttk.psd1 -
Verbose\n$testOutput = @(Test-AzTemplate -TemplatePath
\"$(sample.folder)\")\n$testOutput\n\nif ($testOutput | ? {$_.Errors }) {\n
exit 1 \n} else {\n    Write-Host \"##vso[task.setvariable
variable=result.best.practice]$true\"\n    exit 0\n} \n",
    "errorActionPreference": "continue",
    "failOnStderr": "true",
    "ignoreLASTEXITCODE": "false",
    "pwsh": "true",
    "workingDirectory": ""
  }
}
```

For Pipeline YAML definition:

YAML

```
- pwsh: |
   Import-Module $(ttk.folder)/arm-ttk/arm-ttk.psd1 -Verbose
   $testOutput = @(Test-AzTemplate -TemplatePath "$(sample.folder)")
   $testOutput

   if ($testOutput | ? {$_.Errors }) {
      exit 1
   } else {
       Write-Host "##vso[task.setvariable
variable=result.best.practice]$true"
       exit 0
   }
  errorActionPreference: continue
  failOnStderr: true
  displayName: 'Run Best Practices Tests'
  continueOnError: true
```

Next steps

- To learn about the template tests, see Test cases for ARM templates.
- To test parameter files, see Test cases for parameter files.
- For createUiDefinition tests, see Test cases for createUiDefinition.json.
- To learn about tests for all files, see Test cases for all files.
- For a Learn module that covers using the test toolkit, see Validate Azure resources by using the ARM Template Test Toolkit.

Test cases for ARM templates

Article • 04/09/2023

This article describes tests that are run with the template test toolkit for Azure Resource Manager templates (ARM templates). It provides examples that **pass** or **fail** the test and includes the name of each test. For more information about how to run tests or how to run a specific test, see Test parameters.

Use correct schema

Test name: **DeploymentTemplate Schema Is Correct**

In your template, you must specify a valid schema value.

The following example **fails** because the schema is invalid.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-01-01/deploymentTemplate.json#",
}
```

The following example displays a **warning** because schema version `2015-01-01` is deprecated and isn't maintained.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
}
```

The following example **passes** using a valid schema.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
}
```

The template's `schema` property must be set to one of the following schemas:

- `https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#`
- `https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#`
- `https://schema.management.azure.com/schemas/2018-05-01/subscriptionDeploymentTemplate.json#`
- `https://schema.management.azure.com/schemas/2019-08-01/tenantDeploymentTemplate.json#`
- `https://schema.management.azure.com/schemas/2019-08-01/managementGroupDeploymentTemplate.json`

Declared parameters must be used

Test name: **Parameters Must Be Referenced**

This test finds parameters that aren't used in the template or parameters that aren't used in a valid expression.

To reduce confusion in your template, delete any parameters that are defined but not used. Eliminating unused parameters simplifies template deployments because you don't have to provide unnecessary values.

In Bicep, use Linter rule - no unused parameters.

The following example **fails** because the expression that references a parameter is missing the leading square bracket (`[`).

JSON

```
"resources": [
  {
    "location": " parameters('location')]"
  }
]
```

The following example **passes** because the expression is valid.

JSON

```
"resources": [
  {
    "location": "[parameters('location')]"
  }
]
```

Secure parameters can't have hard-coded default

Test name: **Secure String Parameters Cannot Have Default**

Don't provide a hard-coded default value for a secure parameter in your template. A secure parameter can have an empty string as a default value or use the newGuid function in an expression.

You use the types `secureString` or `secureObject` on parameters that contain sensitive values, like passwords. When a parameter uses a secure type, the value of the parameter isn't logged or stored in the deployment history. This action prevents a malicious user from discovering the sensitive value.

When you provide a default value, that value is discoverable by anyone who can access the template or the deployment history.

In Bicep, use Linter rule - secure parameter default.

The following example **fails**.

JSON

```
"parameters": {
  "adminPassword": {
    "defaultValue": "HardcodedPassword",
    "type": "secureString"
  }
}
```

The next example **passes**.

JSON

```
"parameters": {
  "adminPassword": {
    "type": "secureString"
  }
}
```

The following example **passes** because the `newGuid` function is used.

JSON

```
"parameters": {
  "secureParameter": {
    "type": "secureString",
    "defaultValue": "[newGuid()]"
  }
}
```

Environment URLs can't be hard-coded

Test name: **DeploymentTemplate Must Not Contain Hardcoded Uri**

Don't hard-code environment URLs in your template. Instead, use the environment function to dynamically get these URLs during deployment. For a list of the URL hosts that are blocked, see the test case.

In Bicep, use Linter rule - no hardcoded environment URL.

The following example **fails** because the URL is hard-coded.

JSON

```
"variables":{
  "AzureURL":"https://management.azure.com"
}
```

The test also **fails** when used with concat or uri.

JSON

```
"variables":{
  "AzureSchemaURL1": "[concat('https://','gallery.azure.com')]",
  "AzureSchemaURL2": "[uri('gallery.azure.com','test')]"
}
```

The following example **passes**.

JSON

```
"variables": {
  "AzureSchemaURL": "[environment().gallery]"
}
```

Location uses parameter

Test name: **Location Should Not Be Hardcoded**

To set a resource's location, your templates should have a parameter named `location` with the type set to `string`. In the main template, *azuredeploy.json* or *mainTemplate.json*, this parameter can default to the resource group location. In linked or nested templates, the location parameter shouldn't have a default location.

Template users may have limited access to regions where they can create resources. A hard-coded resource location might block users from creating a resource. The `"[resourceGroup().location]"` expression could block users if the resource group was created in a region the user can't access. Users who are blocked are unable to use the template.

By providing a `location` parameter that defaults to the resource group location, users can use the default value when convenient but also specify a different location.

In Bicep, use Linter rule - no location expressions outside of parameter default values.

The following example **fails** because the resource's `location` is set to `resourceGroup().location`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {},
  "variables": {},
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-02-01",
      "name": "storageaccount1",
      "location": "[resourceGroup().location]",
      "kind": "StorageV2",
      "sku": {
        "name": "Premium_LRS",
        "tier": "Premium"
      }
    }
  ]
}
```

The next example uses a `location` parameter but **fails** because the parameter defaults to a hard-coded location.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "location": {
      "type": "string",
      "defaultValue": "westus"
    }
  },
  "variables": {},
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-02-01",
      "name": "storageaccount1",
      "location": "[parameters('location')]",
      "kind": "StorageV2",
      "sku": {
        "name": "Premium_LRS",
        "tier": "Premium"
      }
    }
  ],
  "outputs": {}
}
```

The following example **passes** when the template is used as the main template. Create a parameter that defaults to the resource group location but allows users to provide a different value.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for the resources."
      }
    }
  },
  "variables": {},
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
```

```
      "apiVersion": "2021-02-01",
      "name": "storageaccount1",
      "location": "[parameters('location')]",
      "kind": "StorageV2",
      "sku": {
        "name": "Premium_LRS",
        "tier": "Premium"
      }
    }
  ],
  "outputs": {}
}
```

> **Note**
>
> If the preceding example is used as a linked template, the test **fails**. When used as a linked template, remove the default value.

Resources should have location

Test name: **Resources Should Have Location**

The location for a resource should be set to a template expression or `global`. The template expression would typically use the `location` parameter described in Location uses parameter.

In Bicep, use Linter rule - no hardcoded locations.

The following example **fails** because the `location` isn't an expression or `global`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {},
  "functions": [],
  "variables": {},
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-02-01",
      "name": "storageaccount1",
      "location": "westus",
      "kind": "StorageV2",
      "sku": {
```

```
        "name": "Premium_LRS",
        "tier": "Premium"
      }
    }
  ],
  "outputs": {}
}
```

The following example **passes** because the resource `location` is set to `global`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {},
  "functions": [],
  "variables": {},
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-02-01",
      "name": "storageaccount1",
      "location": "global",
      "kind": "StorageV2",
      "sku": {
        "name": "Premium_LRS",
        "tier": "Premium"
      }
    }
  ],
  "outputs": {}
}
```

The next example also **passes** because the `location` parameter uses an expression. The resource `location` uses the expression's value.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for the resources."
```

```
      }
    }
  },
  "variables": {},
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-02-01",
      "name": "storageaccount1",
      "location": "[parameters('location')]",
      "kind": "StorageV2",
      "sku": {
        "name": "Premium_LRS",
        "tier": "Premium"
      }
    }
  ],
  "outputs": {}
}
```

VM size uses parameter

Test name: **VM Size Should Be A Parameter**

Don't hard-code the `hardwareProfile` object's `vmSize`. The test fails when the `hardwareProfile` is omitted or contains a hard-coded value. Provide a parameter so users of your template can modify the size of the deployed virtual machine. For more information, see Microsoft.Compute virtualMachines.

The following example **fails** because the `hardwareProfile` object's `vmSize` is a hard-coded value.

JSON

```
"resources": [
  {
    "type": "Microsoft.Compute/virtualMachines",
    "apiVersion": "2020-12-01",
    "name": "demoVM",
    "location": "[parameters('location')]",
    "properties": {
      "hardwareProfile": {
        "vmSize": "Standard_D2_v3"
      }
    }
  }
]
```

The example **passes** when a parameter specifies a value for `vmSize`:

JSON

```
"parameters": {
  "vmSizeParameter": {
    "type": "string",
    "defaultValue": "Standard_D2_v3",
    "metadata": {
      "description": "Size for the virtual machine."
    }
  }
}
```

Then, `hardwareProfile` uses an expression for `vmSize` to reference the parameter's value:

JSON

```
"resources": [
  {
    "type": "Microsoft.Compute/virtualMachines",
    "apiVersion": "2020-12-01",
    "name": "demoVM",
    "location": "[parameters('location')]",
    "properties": {
      "hardwareProfile": {
        "vmSize": "[parameters('vmSizeParameter')]"
      }
    }
  }
]
```

Min and max values are numbers

Test name: **Min And Max Value Are Numbers**

When you define a parameter with `minValue` and `maxValue`, specify them as numbers. You must use `minValue` and `maxValue` as a pair or the test fails.

The following example **fails** because `minValue` and `maxValue` are strings.

JSON

```
"exampleParameter": {
  "type": "int",
  "minValue": "0",
```

```
  "maxValue": "10"
}
```

The following example **fails** because only `minValue` is used.

JSON

```
"exampleParameter": {
  "type": "int",
  "minValue": 0
}
```

The following example **passes** because `minValue` and `maxValue` are numbers.

JSON

```
"exampleParameter": {
  "type": "int",
  "minValue": 0,
  "maxValue": 10
}
```

Artifacts parameter defined correctly

Test name: **artifacts parameter**

When you include parameters for `_artifactsLocation` and `_artifactsLocationSasToken`, use the correct defaults and types. The following conditions must be met to pass this test:

- If you provide one parameter, you must provide the other.
- `_artifactsLocation` must be a `string`.
- `_artifactsLocation` must have a default value in the main template.
- `_artifactsLocation` can't have a default value in a nested template.
- `_artifactsLocation` must have either `"[deployment().properties.templateLink.uri]"` or the raw repo URL for its default value.
- `_artifactsLocationSasToken` must be a `secureString`.
- `_artifactsLocationSasToken` can only have an empty string for its default value.
- `_artifactsLocationSasToken` can't have a default value in a nested template.

In Bicep, use Linter rule - artifacts parameters.

Declared variables must be used

Test name: **Variables Must Be Referenced**

This test finds variables that aren't used in the template or aren't used in a valid expression. To reduce confusion in your template, delete any variables that are defined but not used.

Variables that use the `copy` element to iterate values must be referenced. For more information, see Variable iteration in ARM templates.

In Bicep, use Linter rule - no unused variables.

The following example **fails** because the variable that uses the `copy` element isn't referenced.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "itemCount": {
      "type": "int",
      "defaultValue": 5
    }
  },
  "variables": {
    "copy": [
      {
        "name": "stringArray",
        "count": "[parameters('itemCount')]",
        "input": "[concat('item', copyIndex('stringArray', 1))]"
      }
    ]
  },
  "resources": [],
  "outputs": {}
}
```

The following example **fails** because the expression that references a variable is missing the leading square bracket (`[`).

JSON

```
"outputs": {
  "outputVariable": {
    "type": "string",
```

```
    "value": " variables('varExample')]"
  }
}
```

The following example **passes** because the variable is referenced in `outputs`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "itemCount": {
      "type": "int",
      "defaultValue": 5
    }
  },
  "variables": {
    "copy": [
      {
        "name": "stringArray",
        "count": "[parameters('itemCount')]",
        "input": "[concat('item', copyIndex('stringArray', 1))]"
      }
    ]
  },
  "resources": [],
  "outputs": {
    "arrayResult": {
      "type": "array",
      "value": "[variables('stringArray')]"
    }
  }
}
```

The following example **passes** because the expression is valid.

JSON

```
"outputs": {
  "outputVariable": {
    "type": "string",
    "value": "[variables('varExample')]"
  }
}
```

Dynamic variable should not use concat

Test name: **Dynamic Variable References Should Not Use Concat**

Sometimes you need to dynamically construct a variable based on the value of another variable or parameter. Don't use the concat function when setting the value. Instead, use an object that includes the available options and dynamically get one of the properties from the object during deployment.

The following example **passes**. The `currentImage` variable is dynamically set during deployment.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "osType": {
      "type": "string",
      "allowedValues": [
        "Windows",
        "Linux"
      ]
    }
  },
  "variables": {
    "imageOS": {
      "Windows": {
        "image": "Windows Image"
      },
      "Linux": {
        "image": "Linux Image"
      }
    },
    "currentImage": "[variables('imageOS')[parameters('osType')].image]"
  },
  "resources": [],
  "outputs": {
    "result": {
      "type": "string",
      "value": "[variables('currentImage')]"
    }
  }
}
```

Use recent API version

Test name: **apiVersions Should Be Recent**

The API version for each resource should use a recent version that's hard-coded as a string. The test evaluates the API version in your template against the resource provider's versions in the toolkit's cache. An API version that's less than two years old from the date the test was run is considered recent. Don't use a preview version when a more recent version is available.

A warning that an API version wasn't found only indicates the version isn't included in the toolkit's cache. Using the latest version of an API, which is recommended, can generate the warning.

Learn more about the toolkit cache.

In Bicep, use Linter rule - use recent API versions.

The following example **fails** because the API version is more than two years old.

JSON

```
"resources": [
  {
    "type": "Microsoft.Storage/storageAccounts",
    "apiVersion": "2019-06-01",
    "name": "storageaccount1",
    "location": "[parameters('location')]"
  }
]
```

The following example **fails** because a preview version is used when a newer version is available.

JSON

```
"resources": [
  {
    "type": "Microsoft.Storage/storageAccounts",
    "apiVersion": "2020-08-01-preview",
    "name": "storageaccount1",
    "location": "[parameters('location')]"
  }
]
```

The following example **passes** because it's a recent version that's not a preview version.

JSON

```
"resources": [
  {
    "type": "Microsoft.Storage/storageAccounts",
```

```
    "apiVersion": "2021-02-01",
    "name": "storageaccount1",
    "location": "[parameters('location')]"
  }
]
```

Use hard-coded API version

Test name: **Providers apiVersions Is Not Permitted**

The API version for a resource type determines which properties are available. Provide a hard-coded API version in your template. Don't retrieve an API version that's determined during deployment because you won't know which properties are available.

The following example **fails**.

JSON

```
"resources": [
  {
    "type": "Microsoft.Compute/virtualMachines",
    "apiVersion": "[providers('Microsoft.Compute',
'virtualMachines').apiVersions[0]]",
    ...
  }
]
```

The following example **passes**.

JSON

```
"resources": [
  {
    "type": "Microsoft.Compute/virtualMachines",
    "apiVersion": "2020-12-01",
    ...
  }
]
```

Properties can't be empty

Test name: **Template Should Not Contain Blanks**

Don't hard-code properties to an empty value. Empty values include null and empty strings, objects, or arrays. If a property is set to an empty value, remove that property

from your template. You can set a property to an empty value during deployment, such as through a parameter.

The `template` property in a nested template can include empty properties. For more information about nested templates, see Microsoft.Resources deployments.

The following example **fails** because there are empty properties.

JSON

```
"resources": [
  {
    "type": "Microsoft.Storage/storageAccounts",
    "apiVersion": "2021-01-01",
    "name": "storageaccount1",
    "location": "[parameters('location')]",
    "sku": {},
    "kind": ""
  }
]
```

The following example **passes** because the properties include values.

JSON

```
"resources": [
  {
    "type": "Microsoft.Storage/storageAccounts",
    "apiVersion": "2021-01-01",
    "name": "storageaccount1",
    "location": "[parameters('location')]",
    "sku": {
      "name": "Standard_LRS",
      "tier": "Standard"
    },
    "kind": "Storage"
  }
]
```

Use Resource ID functions

Test name: **IDs Should Be Derived From ResourceIDs**

When specifying a resource ID, use one of the resource ID functions. The allowed functions are:

- resourceId

- subscriptionResourceId
- tenantResourceId
- extensionResourceId

Don't use the concat function to create a resource ID.

In Bicep, use Linter rule - use resource ID functions.

The following example **fails**.

JSON

```
"networkSecurityGroup": {
    "id": "[concat('/subscriptions/', subscription().subscriptionId,
'/resourceGroups/', resourceGroup().name,
'/providers/Microsoft.Network/networkSecurityGroups/',
variables('networkSecurityGroupName'))]"
}
```

The next example **passes**.

JSON

```
"networkSecurityGroup": {
    "id": "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('networkSecurityGroupName'))]"
}
```

ResourceId function has correct parameters

Test name: **ResourceIds should not contain**

When generating resource IDs, don't use unnecessary functions for optional parameters. By default, the resourceId function uses the current subscription and resource group. You don't need to provide those values.

The following example **fails** because you don't need to provide the current subscription ID and resource group name.

JSON

```
"networkSecurityGroup": {
    "id": "[resourceId(subscription().subscriptionId, resourceGroup().name,
'Microsoft.Network/networkSecurityGroups',
variables('networkSecurityGroupName'))]"
}
```

The next example **passes**.

```JSON
"networkSecurityGroup": {
    "id": "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('networkSecurityGroupName'))]"
}
```

This test applies to:

- resourceId
- subscriptionResourceId
- tenantResourceId
- extensionResourceId
- reference
- list*

For `reference` and `list*`, the test **fails** when you use `concat` to construct the resource ID.

dependsOn best practices

Test name: **DependsOn Best Practices**

When setting the deployment dependencies, don't use the if function to test a condition. If one resource depends on a resource that's conditionally deployed, set the dependency as you would with any resource. When a conditional resource isn't deployed, Azure Resource Manager automatically removes it from the required dependencies.

The `dependsOn` element can't begin with a concat function.

In Bicep, use Linter rule - no unnecessary dependsOn entries.

The following example **fails** because it contains an `if` function.

```JSON
"dependsOn": [
  "[if(equals(parameters('newOrExisting'),'new'),
variables('storageAccountName'), '')]"
]
```

The following example **fails** because it begins with `concat`.

JSON

```
"dependsOn": [
  "[concat(variables('storageAccountName'))]"
]
```

The following example **passes**.

JSON

```
"dependsOn": [
  "[variables('storageAccountName')]"
]
```

Nested or linked deployments can't use debug

Test name: **Deployment Resources Must Not Be Debug**

When you define a nested or linked template with the `Microsoft.Resources/deployments` resource type, you can enable debugging. Debugging is used when you need to test a template but can expose sensitive information. Before the template is used in production, turn off debugging. You can remove the `debugSetting` object or change the `detailLevel` property to `none`.

The following example **fails**.

JSON

```
"debugSetting": {
  "detailLevel": "requestContent"
}
```

The following example **passes**.

JSON

```
"debugSetting": {
  "detailLevel": "none"
}
```

Admin user names can't be literal value

Test name: **adminUsername Should Not Be A Literal**

When setting an `adminUserName`, don't use a literal value. Create a parameter for the user name and use an expression to reference the parameter's value.

In Bicep, use Linter rule - admin user name should not be literal.

The following example **fails** with a literal value.

JSON

```
"osProfile":  {
  "adminUserName": "myAdmin"
}
```

The following example **passes** with an expression.

JSON

```
"osProfile": {
  "adminUsername": "[parameters('adminUsername')]"
}
```

Use latest VM image

Test name: **VM Images Should Use Latest Version**

This test is disabled, but the output shows that it passed. The best practice is to check your template for the following criteria:

If your template includes a virtual machine with an image, make sure it's using the latest version of the image.

In Bicep, use Linter rule - use stable VM image.

Use stable VM images

Test name: **Virtual Machines Should Not Be Preview**

Virtual machines shouldn't use preview images. The test checks the `storageProfile` to verify that the `imageReference` doesn't use a string that contains *preview*. And that *preview* isn't used in the `imageReference` properties `offer`, `sku`, or `version`.

For more information about the `imageReference` property, see Microsoft.Compute virtualMachines and Microsoft.Compute virtualMachineScaleSets.

In Bicep, use Linter rule - use stable VM image.

The following example **fails** because `imageReference` is a string that contains *preview*.

JSON

```
"properties": {
  "storageProfile": {
    "imageReference": "latest-preview"
  }
}
```

The following example **fails** when *preview* is used in `offer`, `sku`, or `version`.

JSON

```
"properties": {
  "storageProfile": {
    "imageReference": {
      "publisher": "Canonical",
      "offer": "UbuntuServer_preview",
      "sku": "16.04-LTS-preview",
      "version": "preview"
    }
  }
}
```

The following example **passes**.

JSON

```
"storageProfile": {
  "imageReference": {
    "publisher": "Canonical",
    "offer": "UbuntuServer",
    "sku": "16.04-LTS",
    "version": "latest"
  }
}
```

Don't use ManagedIdentity extension

Test name: **ManagedIdentityExtension must not be used**

Don't apply the `ManagedIdentity` extension to a virtual machine. The extension was deprecated in 2019 and should no longer be used.

Outputs can't include secrets

Test name: **Outputs Must Not Contain Secrets**

Don't include any values in the `outputs` section that potentially exposes secrets. For example, secure parameters of type `secureString` or `secureObject`, or list* functions such as `listKeys`.

The output from a template is stored in the deployment history, so a malicious user could find that information.

In Bicep, use Linter rule - outputs should not contain secrets.

The following example **fails** because it includes a secure parameter in an output value.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "secureParam": {
      "type": "secureString"
    }
  },
  "functions": [],
  "variables": {},
  "resources": [],
  "outputs": {
    "badResult": {
      "type": "string",
      "value": "[concat('this is the value ', parameters('secureParam'))]"
    }
  }
}
```

The following example **fails** because it uses a list* function in the outputs.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
```

```
  "parameters": {
    "storageName": {
      "type": "string"
    }
  },
  "functions": [],
  "variables": {},
  "resources": [],
  "outputs": {
    "badResult": {
      "type": "object",
      "value": "[listKeys(resourceId('Microsoft.Storage/storageAccounts',
parameters('storageName')), '2021-02-01')]"
    }
  }
}
```

Use protectedSettings for commandToExecute secrets

Test name: **CommandToExecute Must Use ProtectedSettings For Secrets**

For resources with type `CustomScript`, use the encrypted `protectedSettings` when `commandToExecute` includes secret data such as a password. For example, secret data can be used in secure parameters of type `secureString` or `secureObject`, list* functions such as `listKeys`, or custom scripts.

Don't use secret data in the `settings` object because it uses clear text. For more information, see Microsoft.Compute virtualMachines/extensions, Windows, or Linux.

In Bicep, use Linter rule - use protectedSettings for commandToExecute secrets.

The following example **fails** because `settings` uses `commandToExecute` with a secure parameter.

JSON

```
"parameters": {
  "adminPassword": {
    "type": "secureString"
  }
}
...
"properties": {
  "type": "CustomScript",
  "settings": {
    "commandToExecute": "[parameters('adminPassword')]"
```

```
  }
}
```

The following example **fails** because `settings` uses `commandToExecute` with a `listKeys` function.

JSON

```
"properties": {
  "type": "CustomScript",
  "settings": {
    "commandToExecute": "
[listKeys(resourceId('Microsoft.Storage/storageAccounts',
parameters('storageName')), '2021-02-01')]"
  }
}
```

The following example **passes** because `protectedSettings` uses `commandToExecute` with a secure parameter.

JSON

```
"parameters": {
  "adminPassword": {
    "type": "secureString"
  }
}
...
"properties": {
  "type": "CustomScript",
  "protectedSettings": {
    "commandToExecute": "[parameters('adminPassword')]"
  }
}
```

The following example **passes** because `protectedSettings` uses `commandToExecute` with a `listKeys` function.

JSON

```
"properties": {
  "type": "CustomScript",
  "protectedSettings": {
    "commandToExecute": "
[listKeys(resourceId('Microsoft.Storage/storageAccounts',
parameters('storageName')), '2021-02-01')]"
  }
}
```

Use recent API versions in reference functions

Test name: **apiVersions Should Be Recent In Reference Functions**

The API version used in a reference function must be recent and not a preview version. The test evaluates the API version in your template against the resource provider's versions in the toolkit's cache. An API version that's less than two years old from the date the test was run is considered recent.

A warning that an API version wasn't found only indicates the version isn't included in the toolkit's cache. Using the latest version of an API, which is recommended, can generate the warning.

Learn more about the toolkit cache.

The following example **fails** because the API version is more than two years old.

JSON

```
"outputs": {
  "stgAcct": {
    "type": "string",
    "value": "[reference(resourceId(parameters('storageResourceGroup'),
'Microsoft.Storage/storageAccounts', parameters('storageAccountName')),
'2019-06-01')]"
  }
}
```

The following example **fails** because the API version is a preview version.

JSON

```
"outputs": {
  "stgAcct": {
    "type": "string",
    "value": "[reference(resourceId(parameters('storageResourceGroup'),
'Microsoft.Storage/storageAccounts', parameters('storageAccountName')),
'2020-08-01-preview')]"
  }
}
```

The following example **passes** because the API version less than two years old and isn't a preview version.

JSON

```
"outputs": {
  "stgAcct": {
    "type": "string",
    "value": "[reference(resourceId(parameters('storageResourceGroup'),
'Microsoft.Storage/storageAccounts', parameters('storageAccountName')),
'2021-02-01')]"
  }
}
```

Use type and name in resourceId functions

Test name: **Resources Should Not Be Ambiguous**

This test is disabled, but the output shows that it passed. The best practice is to check your template for the following criteria:

A resourceId must include a resource type and resource name. This test finds all the template's `resourceId` functions and verifies that the resource is used in the template with the correct syntax. Otherwise the function is considered ambiguous.

For example, a `resourceId` function is considered ambiguous:

- When a resource isn't found in the template and a resource group isn't specified.
- If a resource includes a condition and a resource group isn't specified.
- If a related resource contains some but not all of the name segments. For example, a child resource contains more than one name segment. For more information, see resourceId remarks.

Use inner scope for nested deployment secure parameters

Test name: **Secure Params In Nested Deployments**

Use the nested template's `expressionEvaluationOptions` object with `inner` scope to evaluate expressions that contain secure parameters of type `secureString` or `secureObject` or list* functions such as `listKeys`. If the `outer` scope is used, expressions are evaluated in clear text within the parent template's scope. The secure value is then visible to anyone with access to the deployment history. The default value of `expressionEvaluationOptions` is `outer`.

For more information about nested templates, see Microsoft.Resources deployments and Expression evaluation scope in nested templates.

In Bicep, use Linter rule - secure params in nested deploy.

The following example **fails** because `expressionEvaluationOptions` uses `outer` scope to evaluate secure parameters or `list*` functions.

JSON

```
"resources": [
  {
    "type": "Microsoft.Resources/deployments",
    "apiVersion": "2021-04-01",
    "name": "nestedTemplate",
    "properties": {
      "expressionEvaluationOptions": {
        "scope": "outer"
      }
    }
  }
]
```

The following example **passes** because `expressionEvaluationOptions` uses `inner` scope to evaluate secure parameters or `list*` functions.

JSON

```
"resources": [
  {
    "type": "Microsoft.Resources/deployments",
    "apiVersion": "2021-04-01",
    "name": "nestedTemplate",
    "properties": {
      "expressionEvaluationOptions": {
        "scope": "inner"
      }
    }
  }
]
```

Next steps

- To learn about running the test toolkit, see Use ARM template test toolkit.
- For a Learn module that covers using the test toolkit, see Preview changes and validate Azure resources by using what-if and the ARM template test toolkit.
- To test parameter files, see Test cases for parameter files.
- For createUiDefinition tests, see Test cases for createUiDefinition.json.
- To learn about tests for all files, see Test cases for all files.

Test cases for parameter files

Article • 04/09/2023

This article describes the tests that are run with the template test toolkit for parameter files. For example, a file named *azuredeploy.parameters.json*. The examples include the test names and code samples that **pass** or **fail** the tests. For more information about how to run tests or how to run a specific test, see Test parameters.

The toolkit includes test cases for Azure Resource Manager templates (ARM templates) and the main template files named *azuredeploy.json* or *maintemplate.json*.

Use valid contentVersion

Test name: DeploymentParameters Should Have ContentVersion

The `contentVersion` must contain a string in the format `1.0.0.0` and only use numbers.

The following example **fails** because the `contentVersion` is missing.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  "parameters": {
    "stgAcctName": {
      "value": "demostorage01"
    }
  }
}
```

The following example **fails** because `contentVersion` isn't a string.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  "contentVersion": {},
  "parameters": {
    "stgAcctName": {
      "value": "demostorage01"
    }
  }
}
```

The following example **passes**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "stgAcctName": {
      "value": "demostorage01"
    }
  }
}
```

File must include parameters

Test name: DeploymentParameters Should Have Parameters

A parameter file must include the `parameters` section.

The following example **fails**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
}
```

The following example **passes**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "stgAcctName": {
      "value": "demostorage01"
    }
  }
}
```

Use valid schema version

Test name: DeploymentParameters Should Have Schema

The parameter file must include a valid schema version.

There are two valid schema versions for parameter files:

- `https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#`
- `https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#`

The following example **fails**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2021-01-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "stgAcctName": {
      "value": "demostorage01"
    }
  }
}
```

The following example **passes**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "stgAcctName": {
      "value": "demostorage01"
    }
  }
}
```

Parameters must contain values

Test name: DeploymentParameters Should Have Value

A parameter must contain a `value` or a `reference`. For secrets such as a password, a key vault uses a `reference` in the parameter file. For more information, see Use Azure Key Vault to pass secure parameter value during deployment.

The following example **fails** because `stgAcctName` doesn't have a `value`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "stgAcctName": {}
  }
}
```

The following example **passes**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "stgAcctName": {
      "value": "demostorage01"
    }
  }
}
```

Next steps

- To learn about the test toolkit, see Use ARM template test toolkit.
- For ARM template tests, see Test cases for ARM templates.
- For createUiDefinition tests, see Test cases for createUiDefinition.json.
- To learn about tests for all files, see Test cases for all files.

Test cases for createUiDefinition.json

Article • 04/09/2023

This article describes the tests that are run with the template test toolkit for createUiDefinition.json files. The examples include the test names and code samples that **pass** or **fail** the tests.

The toolkit includes test cases for Azure Resource Manager templates (ARM templates) and the main template files named *azuredeploy.json* or *maintemplate.json*. When the directory contains a *createUiDefinition.json* file, specific tests are run for UI controls. For more information about how to run tests or how to run a specific test, see Test parameters.

The *createUiDefinition.json* file creates custom user-interface (UI) controls using elements and functions.

Verify template parameter allows values

Test name: **Allowed Values Should Actually Be Allowed**

This test checks that values for each control in *createUiDefinition.json* are allowed in the main template's parameters. The parameters are mapped by name between the main template and the *createUiDefinition.json* file.

The main template's parameter must accept the values from the control's `allowedValues`. The test also checks that the control is referenced in the *createUiDefinition.json* `outputs` section.

This test checks the main template and *createUiDefinition.json* file. An example of the *createUiDefinition.json* file is shown after the main template examples.

The following example **fails** because the main template's parameter name `combo` doesn't match the control's parameter name `comboBox`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "combo": {
      "type": "string",
      "defaultValue": "two"
```

```
    }
  },
  "resources": [],
  "outputs": {
    "comboBoxOutput": {
      "type": "string",
      "value": "[parameters('combo')]"
    }
  }
}
```

The following example **fails** because the main template's parameter type `int` doesn't accept the control's `string` value. And if a main template's parameter defines a `defaultValue` it must be a valid `value` in the control's `allowedValues`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "comboBox": {
      "type": "int",
      "defaultValue": 4
    }
  },
  "resources": [],
  "outputs": {
    "comboBoxOutput": {
      "type": "string",
      "value": "[parameters('combo')]"
    }
  }
}
```

The following example **passes** because the main template's parameter name matches the control's parameter name. And the template's parameter type is a `string` with a `defaultValue` that's specified in the control's `allowedValues`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "comboBox": {
      "type": "string",
      "defaultValue": "two"
```

```
    }
  },
  "resources": [],
  "outputs": {
    "comboBoxOutput": {
      "type": "string",
      "value": "[parameters('comboBox')]"
    }
  }
}
```

The *createUiDefinition.json* file for this example:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-
preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [],
    "steps": [
      {
        "name": "demoComboBox",
        "label": "demoComboBoxLabel",
        "elements": [
          {
            "name": "comboBox",
            "type": "Microsoft.Common.DropDown",
            "label": "Example drop down",
            "defaultValue": "Value two",
            "toolTip": "This is a tool tip",
            "constraints": {
              "allowedValues": [
                {
                  "label": "Value one",
                  "description": "The value to select for option 1.",
                  "value": "one"
                },
                {
                  "label": "Value two",
                  "description": "The value to select for option 2.",
                  "value": "two"
                }
              ],
              "required": true
            },
            "visible": true
          }
        ]
      }
    ],
```

```
    "outputs": {
      "comboBox": "[steps('demoComboBox').comboBox]"
    }
  }
}
```

Output controls must exist

Test name: **Controls In Outputs Must Exist**

Controls that are used in the `outputs` section must exist in an element elsewhere in *createUiDefinition.json*. The name referenced in `outputs` must match a name used in `basics[]` or `steps[]`.

The following example **fails**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "nameDoesNotMatchOutput",
        "type": "Microsoft.Common.DropDown",
        "label": "Example drop down",
        "toolTip": "This is a tool tip"
      }
    ],
    "steps": [],
    "outputs": {
      "comboBox": "[basics('comboBox')]"
    }
  }
}
```

The following example **passes**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
```

```
    "basics": [
      {
        "name": "comboBox",
        "type": "Microsoft.Common.DropDown",
        "label": "Example drop down",
        "toolTip": "This is a tool tip"
      }
    ],
    "steps": [],
    "outputs": {
      "comboBox": "[basics('comboBox')]"
    }
  }
}
```

Properties must include values

Test name: **CreateUIDefinition Must Not Have Blanks**

Properties must include values. Required properties must use valid values. Optional properties that are blank should be removed. The test allows blank `"basics": []`, `"steps": []`, or `defaultValue`.

The following example **fails** because `label`, `placeholder`, and `toolTip` are blank.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "comboBox",
        "type": "Microsoft.Common.DropDown",
        "label": "",
        "placeholder": "",
        "defaultValue": "",
        "toolTip": ""
      }
    ],
    "steps": [],
    "outputs": {
      "comboBox": "[basics('comboBox')]"
    }
  }
}
```

The following example **passes** because `label` and `toolTip` have values, and `placeholder` was removed.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "comboBox",
        "type": "Microsoft.Common.DropDown",
        "label": "Example drop down",
        "defaultValue": "",
        "toolTip": "This is a tool tip"
      }
    ],
    "steps": [],
    "outputs": {
      "comboBox": "[basics('comboBox')]"
    }
  }
}
```

Use valid schema and version

Test name: **CreateUIDefinition Should Have Schema**

The *createUiDefinition.json* file must include a `$schema` property and use a valid `$schema` and `version`. The version numbers in `$schema` and `version` must match.

The following example **fails**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.9.9-preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.9.9-preview"
}
```

The following example **passes** because it uses the latest `$schema` and `version`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-
preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview"
}
```

Don't hide credential confirmation

Test name: **Credential Confirmation Should Not Be Hidden**

This test checks that credentials are confirmed for Microsoft.Common.PasswordBox or Microsoft.Compute.CredentialsCombo. The `hideConfirmation` property should be set to `false` so that the confirmation is visible.

The following example **fails** because `hideConfirmation` is `true`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-
preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "credentials",
        "type": "Microsoft.Compute.CredentialsCombo",
        "label": {
          "password": "Password",
          "confirmPassword": "Confirm password"
        },
        "toolTip": {
          "password": "Type your credentials"
        },
        "constraints": {
          "required": true,
          "customPasswordRegex": "^(?=.*[A-Za-z])(?=.*\\d)[A-Za-z\\d]
{12,}$",
          "customValidationMessage": "The password must be alphanumeric,
contain at least 12 characters, and have at least 1 letter and 1 number."
        },
        "options": {
          "hideConfirmation": true
        },
        "osPlatform": "Windows",
        "visible": true
      }
```

```
    ],
    "steps": [],
    "outputs": {
      "location": "[location()]",
      "credentials": "[basics('credentials')]"
    }
  }
}
```

The following example **passes** because `hideConfirmation` is `false`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-
preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "credentials",
        "type": "Microsoft.Compute.CredentialsCombo",
        "label": {
          "password": "Password",
          "confirmPassword": "Confirm password"
        },
        "toolTip": {
          "password": "Type your credentials"
        },
        "constraints": {
          "required": true,
          "customPasswordRegex": "^(?=.*[A-Za-z])(?=.*\\d)[A-Za-z\\d]
{12,}$",
          "customValidationMessage": "The password must be alphanumeric,
contain at least 12 characters, and have at least 1 letter and 1 number."
        },
        "options": {
          "hideConfirmation": false
        },
        "osPlatform": "Windows",
        "visible": true
      }
    ],
    "steps": [],
    "outputs": {
      "location": "[location()]",
      "credentials": "[basics('credentials')]"
    }
  }
}
```

Use correct handler

Test name: **Handler Must Be Correct**

Use `Microsoft.Azure.CreateUIDef` or `Microsoft.Compute.MultiVm` in the *createUiDefinition.json* file.

The following example **fails**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.",
  "version": "0.1.2-preview"
}
```

The following example **passes**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview"
}
```

Don't hide existing resources

Test name: **HideExisting Must Be Correctly Handled**

If `hideExisting` is set to `false` or omitted, `outputs` must contain `resourceGroup` and `newOrExisting`. The default for `hideExisting` is `false`.

Examples of control types that include `hideExisting` are Microsoft.Storage.StorageAccountSelector, Microsoft.Network.PublicIpAddressCombo, or Microsoft.Network.VirtualNetworkCombo.

The following example **fails**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-
```

```
preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "storage",
        "type": "Microsoft.Storage.StorageAccountSelector",
        "label": "Storage account",
        "toolTip": "This is a demo storage account",
        "defaultValue": {
          "name": "storageaccount01",
          "type": "Premium_LRS"
        },
        "options": {
          "hideExisting": false
        },
        "visible": true
      }
    ],
    "steps": [],
    "outputs": {
      "location": "[location()]"
    }
  }
}
```

The following example **passes**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-
preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "storage",
        "type": "Microsoft.Storage.StorageAccountSelector",
        "label": "Storage account",
        "toolTip": "This is a demo storage account",
        "defaultValue": {
          "name": "storageaccount01",
          "type": "Premium_LRS"
        },
        "options": {
          "hideExisting": false
        },
        "visible": false
      }
    ],
```

```
    "steps": [],
    "outputs": {
      "location": "[location()]",
      "resourceGroup": "[basics('storage').resourceGroup]",
      "newOrExisting": "[basics('storage').newOrExisting]"
    }
  }
}
```

Use location in outputs

Test name: **Location Should Be In Outputs**

The `outputs` section should contain a location using the location function.

The following example **fails** because `outputs` doesn't include a location.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-
preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "comboBox",
        "type": "Microsoft.Common.DropDown",
        "label": "Example drop down",
        "toolTip": "This is a tool tip"
      }
    ],
    "steps": [],
    "outputs": {
      "comboBox": "[basics('comboBox')]"
    }
  }
}
```

The following example **passes**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-
preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
```

```
  "parameters": {
    "basics": [
      {
        "name": "comboBox",
        "type": "Microsoft.Common.DropDown",
        "label": "Example drop down",
        "toolTip": "This is a tool tip"
      }
    ],
    "steps": [],
    "outputs": {
      "comboBox": "[basics('comboBox')]",
      "location": "[location()]"
    }
  }
}
```

Include control outputs in template parameters

Test name: **Outputs Must Be Present In Template Parameters**

The test checks that *createUiDefinition.json* includes an `outputs` section. The test also checks that those `outputs` are defined in the main template's `parameters` section. The names must match because parameters are mapped by name between the *createUiDefinition.json* and the main template.

This test checks the main template and *createUiDefinition.json* file. An example of the *createUiDefinition.json* file is shown after the main template examples.

The following example **fails** because the main template doesn't include the `comboBox` parameter from the *createUiDefinition.json* file's `outputs` section.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    }
  },
  "resources": [],
  "outputs": {
    "location": {
      "type": "string",
      "value": "[parameters('location')]"
```

```
    }
  }
}
```

The following example **passes** because the main template includes the `comboBox` parameter.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "comboBox": {
      "type": "string",
      "defaultValue": "two"
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    }
  },
  "resources": [],
  "outputs": {
    "comboBox": {
      "type": "string",
      "value": "[parameters('comboBox')]"
    },
    "location": {
      "type": "string",
      "value": "[parameters('location')]"
    }
  }
}
```

The *createUiDefinition.json* file for this example:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "comboBox",
        "type": "Microsoft.Common.DropDown",
        "label": "Example drop down",
```

```
        "toolTip": "This is a tool tip"
      }
    ],
    "steps": [],
    "outputs": {
      "comboBox": "[basics('comboBox')]",
      "location": "[location()]"
    }
  }
}
```

Parameters without default must exist in outputs

Test name: **Parameters Without Default Must Exist In CreateUIDefinition**

Parameters in the main template without a default value must exist in the *createUiDefinition.json* file's `outputs` section.

This test checks the main template and *createUiDefinition.json* file. An example of the *azuredeploy.json* file is shown after the control's examples.

The following example **fails** because the *createUiDefinition.json* file's `outputs` doesn't include the main template's parameter `comboBox`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-
preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "comboBox",
        "type": "Microsoft.Common.DropDown",
        "label": "Example drop down",
        "toolTip": "This is a tool tip"
      }
    ],
    "steps": [],
    "outputs": {
      "location": "[location()]"
    }
  }
}
```

The following example **passes** because *createUiDefinition.json* includes the `comboBox` in `outputs`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "comboBox",
        "type": "Microsoft.Common.DropDown",
        "label": "Example drop down",
        "toolTip": "This is a tool tip"
      }
    ],
    "steps": [],
    "outputs": {
      "comboBox": "[basics('comboBox')]",
      "location": "[location()]"
    }
  }
}
```

The *azuredeploy.json* file for this example. The `comboBox` parameter doesn't have a default value.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "comboBox": {
      "type": "string"
    },
    "location": {
      "type": "string"
    }
  },
  "resources": [],
  "outputs": {
    "comboBox": {
      "type": "string",
      "value": "[parameters('comboBox')]"
    },
    "location": {
```

```
      "type": "string",
      "value": "[parameters('location')]"
    }
  }
}
```

Use secure parameter with password box

Test name: **Password Textboxes Must Be Used For Password Parameters**

This test checks that a Microsoft.Common.PasswordBox element is defined in the main template's `parameters` and the *createUiDefinition.json* `outputs`. The main template's parameter type for a password box must be `secureString` or `secureObject`.

This test checks the main template and *createUiDefinition.json* file. An example of the *createUiDefinition.json* file is shown after the main template examples.

The following example **fails** because the main template's `passwordBox` parameter is a `string`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "passwordBox": {
      "type": "string"
    },
    "location": {
      "type": "string"
    }
  },
  "resources": [],
  "outputs": {
    "location": {
      "type": "string",
      "value": "[parameters('location')]"
    }
  }
}
```

The following example **passes** because the main template's `passwordBox` parameter is a `secureString`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "passwordBox": {
      "type": "secureString"
    },
    "location": {
      "type": "string"
    }
  },
  "resources": [],
  "outputs": {
    "location": {
      "type": "string",
      "value": "[parameters('location')]"
    }
  }
}
```

The *createUiDefinition.json* file for this example:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "passwordBox",
        "type": "Microsoft.Common.PasswordBox",
        "label": {
          "password": "Password",
          "confirmPassword": "Confirm password"
        },
        "toolTip": "Type a password"
      }
    ],
    "steps": [],
    "outputs": {
      "location": "[location()]",
      "passwordBox": "[basics('passwordBox')]"
    }
  }
}
```

Password box requires minimum length

Test name: **PasswordBoxes Must Have Min Length**

The test checks that the Microsoft.Common.PasswordBox element uses `constraints` with a `regex` that requires at least 12 characters.

The following example **fails** because there are no `constraints`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "passwordBox",
        "type": "Microsoft.Common.PasswordBox",
        "label": {
          "password": "Password",
          "confirmPassword": "Confirm password"
        },
        "toolTip": "Type a password"
      }
    ],
    "steps": [],
    "outputs": {
      "location": "[location()]",
      "passwordBox": "[basics('passwordBox')]"
    }
  }
}
```

The following example **passes** because the `regex` requires at least 12 characters.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "passwordBox",
        "type": "Microsoft.Common.PasswordBox",
```

```
        "label": {
          "password": "Password",
          "confirmPassword": "Confirm password"
        },
        "toolTip": "Type a password",
        "constraints": {
          "required": true,
          "regex": "^[a-zA-Z0-9]{12,}$",
          "validationMessage": "Password must be at least 12 characters
long, contain only numbers and letters"
        }
      }
    ],
    "steps": [],
    "outputs": {
      "location": "[location()]",
      "passwordBox": "[basics('passwordBox')]"
    }
  }
}
```

Text box must use validation

Test name: **Textboxes Are Well Formed**

Use validation with text boxes to check for `constraints` that contain a `regex` and `message`.

The following example **fails**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-
preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "textBox",
        "type": "Microsoft.Common.TextBox",
        "label": "Text box",
        "toolTip": "Type 1-30 alphanumeric characters",
        "placeholder": "Type your text here",
        "visible": true
      }
    ],
    "steps": [],
    "outputs": {
      "location": "[location()]",
```

```
      "textBox": "[basics('textBox')]"
    }
  }
}
```

The following example **passes**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-
preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "textBox",
        "type": "Microsoft.Common.TextBox",
        "label": "Text box",
        "toolTip": "Type 1-30 alphanumeric characters",
        "placeholder": "Type your text here",
        "constraints": {
          "required": true,
          "validations": [
            {
              "regex": "^[a-z0-9A-Z]{1,30}$",
              "message": "Only 1-30 characters alphanumeric characters are
allowed."
            }
          ]
        },
        "visible": true
      }
    ],
    "steps": [],
    "outputs": {
      "location": "[location()]",
      "textBox": "[basics('textBox')]"
    }
  }
}
```

toolTip must exist with a value

Test name: **Tooltips Should Be Present**

This test checks that the `toolTip` property exists and contains a value.

The following example **fails**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "comboBox",
        "type": "Microsoft.Common.DropDown",
        "label": "Example drop down",
        "toolTip": ""
      }
    ],
    "steps": [],
    "outputs": {
      "location": "[location()]",
      "comboBox": "[basics('comboBox')]"
    }
  }
}
```

The following example **passes**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "comboBox",
        "type": "Microsoft.Common.DropDown",
        "label": "Example drop down",
        "toolTip": "This is a tool tip"
      }
    ],
    "steps": [],
    "outputs": {
      "location": "[location()]",
      "comboBox": "[basics('comboBox')]"
    }
  }
}
```

Don't set a default user name

Test name: **Usernames Should Not Have A Default**

The test checks if there's a `defaultValue` set for Microsoft.Compute.UserNameTextBox.

The following example **fails** because a `defaultValue` is provided.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "userNameBox",
        "type": "Microsoft.Compute.UserNameTextBox",
        "label": "User name",
        "defaultValue": "admin",
        "toolTip": "Enter your user name",
        "osPlatform": "Windows"
      }
    ],
    "steps": [],
    "outputs": {
      "location": "[location()]",
      "userNameBox": "[basics('userNameBox')]"
    }
  }
}
```

The following example **passes**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "userNameBox",
        "type": "Microsoft.Compute.UserNameTextBox",
        "label": "User name",
        "toolTip": "Enter your user name",
        "osPlatform": "Windows"
```

```
      }
    ],
    "steps": [],
    "outputs": {
      "location": "[location()]",
      "userNameBox": "[basics('userNameBox')]"
    }
  }
}
```

Use message with validations

Test name: **Validations Must Have Message**

This test checks that any `validations` in *createUiDefinition.json* include a `message`.

The following example **fails** because the `regex` validation doesn't have a `message`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "textBox",
        "type": "Microsoft.Common.TextBox",
        "label": "Text box",
        "toolTip": "Type 1-30 alphanumeric characters",
        "placeholder": "Type your text here",
        "constraints": {
          "required": true,
          "validations": [
            {
              "regex": "^[a-z0-9A-Z]{1,30}$"
            }
          ]
        },
        "visible": true
      }
    ],
    "steps": [],
    "outputs": {
      "location": "[location()]",
      "textBox": "[basics('textBox')]"
    }
  }
}
```

The following example **passes**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "textBox",
        "type": "Microsoft.Common.TextBox",
        "label": "Text box",
        "toolTip": "Type 1-30 alphanumeric characters",
        "placeholder": "Type your text here",
        "constraints": {
          "required": true,
          "validations": [
            {
              "regex": "^[a-z0-9A-Z]{1,30}$",
              "message": "Only 1-30 characters alphanumeric characters are allowed."
            }
          ]
        },
        "visible": true
      }
    ],
    "steps": [],
    "outputs": {
      "location": "[location()]",
      "textBox": "[basics('textBox')]"
    }
  }
}
```

Virtual machine sizes must match

Test name: **VM Sizes Must Match Template**

This test checks that the Microsoft.Compute.SizeSelector is in the *createUiDefinition.json* `outputs` and the main template's `parameters` section. Main template parameters that specify a `defaultValue` must match a value in the control's `allowedSizes`.

This test checks the main template and *createUiDefinition.json* file. An example of the *createUiDefinition.json* file is shown after the main template examples.

The following example **fails** because the main template's `defaultValue` doesn't match a value in `allowedSizes`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "location": {
      "type": "string"
    },
    "vmSize": {
      "type": "string",
      "defaultValue": "Standard_D9"
    }
  },
  "resources": [],
  "outputs": {
    "location": {
      "type": "string",
      "value": "[parameters('location')]"
    },
    "vmSize": {
      "type": "string",
      "value": "[parameters('vmSize')]"
    }
  }
}
```

The following example **passes** because the main template's `defaultValue` matches a value in `allowedSizes`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "location": {
      "type": "string"
    },
    "vmSize": {
      "type": "string",
      "defaultValue": "Standard_D3"
    }
  },
  "resources": [],
  "outputs": {
```

```
    "location": {
      "type": "string",
      "value": "[parameters('location')]"
    },
    "vmSize": {
      "type": "string",
      "value": "[parameters('vmSize')]"
    }
  }
}
```

The *createUiDefinition.json* file for this example:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-
preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "vmSize",
        "type": "Microsoft.Compute.SizeSelector",
        "label": "VM Size",
        "toolTip": "Select a virtual machine size",
        "recommendedSizes": [
          "Standard_D1"
        ],
        "constraints": {
          "allowedSizes": [
            "Standard_D1",
            "Standard_D2",
            "Standard_D3"
          ]
        },
        "osPlatform": "Windows",
        "visible": true
      }
    ],
    "steps": [],
    "outputs": {
      "location": "[location()]",
      "vmSize": "[basics('vmSize')]"
    }
  }
}
```

Next steps

- To create an Azure portal user interface, see CreateUiDefinition.json for Azure managed application's create experience.
- To use the Create UI Definition Sandbox, see Test your portal interface for Azure Managed Applications.
- For more information about UI controls, see CreateUiDefinition elements and CreateUiDefinition functions.
- To learn more about ARM template tests, see Test cases for ARM templates.

Test cases for all files

Article • 04/09/2023

This article describes the tests that are run with the template test toolkit for all JavaScript Object Notation (JSON) files. The examples include the test names and code samples that **pass** or **fail** the tests. For more information about how to run tests or how to run a specific test, see Test parameters.

Use valid JSON syntax

Test name: JSONFiles Should Be Valid

This test checks that all JSON files contain valid syntax. For example, *azuredeploy.json*, *azuredeploy.parameters.json*, or *createUiDefinition.json* files. If the test **fails**, you'll see failures or warnings for other tests, or JSON parsing.

Template file example

The following example fails because in *azuredeploy.json* the leading curly brace (`{`) is missing from `parameters`, `comboBox`, and `location`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters":
    "comboBox":
      "type": "string"
    },
    "location":
      "type": "string"
    }
  },
  "resources": [],
  "outputs": {
    "comboBox": {
      "type": "string",
      "value": "[parameters('comboBox')]"
    },
    "location": {
      "type": "string",
      "value": "[parameters('location')]"
    }
```

```
  }
}
```

The following example **passes**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "comboBox": {
      "type": "string"
    },
    "location": {
      "type": "string"
    }
  },
  "resources": [],
  "outputs": {
    "comboBox": {
      "type": "string",
      "value": "[parameters('comboBox')]"
    },
    "location": {
      "type": "string",
      "value": "[parameters('location')]"
    }
  }
}
```

Parameter file example

The following example **fails** because *azuredeploy.parameters.json* uses a parameter without a `value`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "location": {
      "value":
    }
  }
}
```

The following example **passes**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "location": {
      "value": "westus"
    }
  }
}
```

CreateUiDefintion example

The following example **fails** because in *createUiDefinition.json* the leading curly brace (`{`) is missing from the `outputs` section.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "comboBox",
        "type": "Microsoft.Common.DropDown",
        "label": "Example drop down",
        "toolTip": "This is a tool tip"
      }
    ],
    "steps": [],
    "outputs":
      "comboBox": "[basics('comboBox')]",
      "location": "[location()]"
    }
  }
}
```

The following example **passes**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/0.1.2-preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Azure.CreateUIDef",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [
      {
        "name": "comboBox",
        "type": "Microsoft.Common.DropDown",
        "label": "Example drop down",
        "toolTip": "This is a tool tip"
      }
    ],
    "steps": [],
    "outputs": {
      "comboBox": "[basics('comboBox')]",
      "location": "[location()]"
    }
  }
}
```

Next steps

- To learn about the test toolkit, see Use ARM template test toolkit.
- For ARM template tests, see Test cases for ARM templates
- To test parameter files, see Test cases for parameter files.
- For createUiDefinition tests, see Test cases for createUiDefinition.json

Azure Resource Manager template specs

Article • 04/09/2023

A template spec is a resource type for storing an Azure Resource Manager template (ARM template) in Azure for later deployment. This resource type enables you to share ARM templates with other users in your organization. Just like any other Azure resource, you can use Azure role-based access control (Azure RBAC) to share the template spec.

Microsoft.Resources/templateSpecs is the resource type for template specs. It consists of a main template and any number of linked templates. Azure securely stores template specs in resource groups. Template Specs support versioning.

To deploy the template spec, you use standard Azure tools like PowerShell, Azure CLI, Azure portal, REST, and other supported SDKs and clients. You use the same commands as you would for the template.

> **Note**
>
> To use template spec with Azure PowerShell, you must install **version 5.0.0 or later**. To use it with Azure CLI, use **version 2.14.2 or later**.

When designing your deployment, always consider the lifecycle of the resources and group the resources that share similar lifecycle into a single template spec. For example, your deployments include multiple instances of Azure Cosmos DB, with each instance containing its own databases and containers. Given the databases and the containers don't change much, you want to create one template spec to include a Cosmo DB instance and its underlying databases and containers. You can then use conditional statements in your templates along with copy loops to create multiple instances of these resources.

Training resources

To learn more about template specs, and for hands-on guidance, see Publish libraries of reusable infrastructure code by using template specs.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **Azure Resource Manager template specs in Bicep**.

Why use template specs?

Template specs provide the following benefits:

- You use standard ARM templates for your template spec.
- You manage access through Azure RBAC, rather than SAS tokens.
- Users can deploy the template spec without having write access to the template.
- You can integrate the template spec into existing deployment process, such as PowerShell script or DevOps pipeline.

Template specs enable you to create canonical templates and share them with teams in your organization. The template specs are secure because they're available to Azure Resource Manager for deployment, but not accessible to users without the correct permission. Users only need read access to the template spec to deploy its template, so you can share the template without allowing others to modify it.

If you currently have your templates in a GitHub repo or storage account, you run into several challenges when trying to share and use the templates. To deploy the template, you need to either make the template publicly accessible or manage access with SAS tokens. To get around this limitation, users might create local copies, which eventually diverge from your original template. Template specs simplify sharing templates.

The templates you include in a template spec should be verified by administrators in your organization to follow the organization's requirements and guidance.

Required permissions

There are two Azure build-in roles defined for template spec:

- Template Spec Reader
- Template Spec Contributor

In addition, you also need the permissions for deploying a Bicep file. See Deploy - CLI or Deploy - PowerShell.

Create template spec

The following example shows a simple template for creating a storage account in Azure.

```
JSON
```

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageAccountType": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_ZRS",
        "Premium_LRS"
      ]
    }
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2019-06-01",
      "name": "[concat('store', uniquestring(resourceGroup().id))]",
      "location": "[resourceGroup().location]",
      "kind": "StorageV2",
      "sku": {
        "name": "[parameters('storageAccountType')]"
      }
    }
  ]
}
```

When you create the template spec, the PowerShell or CLI commands are passed the main template file. If the main template references linked templates, the commands will find and package them to create the template spec. To learn more, see Create a template spec with linked templates.

Create a template spec by using:

PowerShell

Azure PowerShell

```
New-AzTemplateSpec -Name storageSpec -Version 1.0a -ResourceGroupName templateSpecsRg -Location westus2 -TemplateFile ./mainTemplate.json
```

You can also create template specs by using ARM templates. The following template creates a template spec to deploy a storage account:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "templateSpecName": {
      "type": "string",
      "defaultValue": "CreateStorageAccount"
    },
    "templateSpecVersionName": {
      "type": "string",
      "defaultValue": "0.1"
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    }
  },
  "resources": [
    {
      "type": "Microsoft.Resources/templateSpecs",
      "apiVersion": "2021-05-01",
      "name": "[parameters('templateSpecName')]",
      "location": "[parameters('location')]",
      "properties": {
        "description": "A basic templateSpec - creates a storage account.",
        "displayName": "Storage account (Standard_LRS)"
      }
    },
    {
      "type": "Microsoft.Resources/templateSpecs/versions",
      "apiVersion": "2021-05-01",
      "name": "[format('{0}/{1}', parameters('templateSpecName'),
parameters('templateSpecVersionName'))]",
      "location": "[parameters('location')]",
      "dependsOn": [
        "[resourceId('Microsoft.Resources/templateSpecs',
parameters('templateSpecName'))]"
      ],
      "properties": {
        "mainTemplate": {
          "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
          "contentVersion": "1.0.0.0",
          "parameters": {
            "storageAccountType": {
              "type": "string",
              "defaultValue": "Standard_LRS",
              "allowedValues": [
                "Standard_LRS",
                "Standard_GRS",
                "Standard_ZRS",
```

```
                "Premium_LRS"
              ]
            }
          },
          "resources": [
            {
              "type": "Microsoft.Storage/storageAccounts",
              "apiVersion": "2019-06-01",
              "name": "[concat('store', uniquestring(resourceGroup().id))]",
              "location": "[resourceGroup().location]",
              "kind": "StorageV2",
              "sku": {
                "name": "[[parameters('storageAccountType')]"
              }
            }
          ]
        }
      }
    }
  ]
}
```

The size of a template spec is limited to approximated 2 MB. If a template spec size exceeds the limit, you will get the **TemplateSpecTooLarge** error code. The error message says:

error

```
The size of the template spec content exceeds the maximum limit. For large
template specs with many artifacts, the recommended course of action is to
split it into multiple template specs and reference them modularly via
TemplateLinks.
```

You can view all template specs in your subscription by using:

PowerShell

Azure PowerShell

```
Get-AzTemplateSpec
```

You can view details of a template spec, including its versions with:

PowerShell

Azure PowerShell

```
Get-AzTemplateSpec -ResourceGroupName templateSpecsRG -Name storageSpec
```

Deploy template spec

After you've created the template spec, users with the template spec reader role can deploy it. In addition, you also need the permissions for deploying an ARM template. See Deploy - CLI or Deploy - PowerShell.

Template specs can be deployed through the portal, PowerShell, Azure CLI, or as a linked template in a larger template deployment. Users in an organization can deploy a template spec to any scope in Azure (resource group, subscription, management group, or tenant).

Instead of passing in a path or URI for a template, you deploy a template spec by providing its resource ID. The resource ID has the following format:

/subscriptions/{subscription-id}/resourceGroups/{resource-group}/providers/Microsoft.Resources/templateSpecs/{template-spec-name}/versions/{template-spec-version}

Notice that the resource ID includes a version name for the template spec.

For example, you deploy a template spec with the following command.

PowerShell

Azure PowerShell

```
$id = "/subscriptions/11111111-1111-1111-1111-111111111111/resourceGroups/templateSpecsRG/providers/Microsoft.Resources/templateSpecs/storageSpec/versions/1.0a"

New-AzResourceGroupDeployment `
  -TemplateSpecId $id `
  -ResourceGroupName demoRG
```

In practice, you'll typically run `Get-AzTemplateSpec` or `az ts show` to get the ID of the template spec you want to deploy.

PowerShell

Azure PowerShell

```
$id = (Get-AzTemplateSpec -Name storageSpec -ResourceGroupName
templateSpecsRg -Version 1.0a).Versions.Id

New-AzResourceGroupDeployment `
  -ResourceGroupName demoRG `
  -TemplateSpecId $id
```

You can also open a URL in the following format to deploy a template spec:

```
https://portal.azure.com/#create/Microsoft.Template/templateSpecVersionId/%2
fsubscriptions%2f{subscription-id}%2fresourceGroups%2f{resource-group-
name}%2fproviders%2fMicrosoft.Resources%2ftemplateSpecs%2f{template-spec-
name}%2fversions%2f{template-spec-version}
```

Parameters

Passing in parameters to template spec is exactly like passing parameters to an ARM template. Add the parameter values either inline or in a parameter file.

To pass a parameter inline, use:

PowerShell

```
New-AzResourceGroupDeployment `
  -TemplateSpecId $id `
  -ResourceGroupName demoRG `
  -StorageAccountType Standard_GRS
```

To create a local parameter file, use:

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "StorageAccountType": {
      "value": "Standard_GRS"
    }
```

```
    }
  }
```

And, pass that parameter file with:

PowerShell

Azure PowerShell

```
New-AzResourceGroupDeployment `
  -TemplateSpecId $id `
  -ResourceGroupName demoRG `
  -TemplateParameterFile ./mainTemplate.parameters.json
```

Versioning

When you create a template spec, you provide a version name for it. As you iterate on the template code, you can either update an existing version (for hotfixes) or publish a new version. The version is a text string. You can choose to follow any versioning system, including semantic versioning. Users of the template spec can provide the version name they want to use when deploying it.

Use tags

Tags help you logically organize your resources. You can add tags to template specs by using Azure PowerShell and Azure CLI:

PowerShell

Azure PowerShell

```
New-AzTemplateSpec `
  -Name storageSpec `
  -Version 1.0a `
  -ResourceGroupName templateSpecsRg `
  -Location westus2 `
  -TemplateFile ./mainTemplate.json `
  -Tag @{Dept="Finance";Environment="Production"}
```

PowerShell

Azure PowerShell

```
Set-AzTemplateSpec `
  -Name storageSpec `
  -Version 1.0a `
  -ResourceGroupName templateSpecsRg `
  -Location westus2 `
  -TemplateFile ./mainTemplate.json `
  -Tag @{Dept="Finance";Environment="Production"}
```

When creating or modifying a template spec with the version parameter specified, but without the tag/tags parameter:

- If the template spec exists and has tags, but the version doesn't exist, the new version inherits the same tags as the existing template spec.

When creating or modifying a template spec with both the tag/tags parameter and the version parameter specified:

- If both the template spec and the version don't exist, the tags are added to both the new template spec and the new version.
- If the template spec exists, but the version doesn't exist, the tags are only added to the new version.
- If both the template spec and the version exist, the tags only apply to the version.

When modifying a template with the tag/tags parameter specified but without the version parameter specified, the tags is only added to the template spec.

Create a template spec with linked templates

If the main template for your template spec references linked templates, the PowerShell and CLI commands can automatically find and package the linked templates from your local drive. You don't need to manually configure storage accounts or repositories to host the template specs - everything is self-contained in the template spec resource.

The following example consists of a main template with two linked templates. The example is only an excerpt of the template. Notice that it uses a property named `relativePath` to link to the other templates. You must use `apiVersion` of `2020-06-01` or later for the deployments resource.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
```

```
01/deploymentTemplate.json#",
  ...
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2020-06-01",
      ...
      "properties": {
        "mode": "Incremental",
        "templateLink": {
          "relativePath": "artifacts/webapp.json"
        }
      }
    },
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2020-06-01",
      ...
      "properties": {
        "mode": "Incremental",
        "templateLink": {
          "relativePath": "artifacts/database.json"
        }
      }
    }
  ],
  "outputs": {}
}
```

When the PowerShell or CLI command to create the template spec is executed for the preceding example, the command finds three files - the main template, the web app template (`webapp.json`), and the database template (`database.json`) - and packages them into the template spec.

For more information, see Tutorial: Create a template spec with linked templates.

Deploy template spec as a linked template

Once you've created a template spec, it's easy to reuse it from an ARM template or another template spec. You link to a template spec by adding its resource ID to your template. The linked template spec is automatically deployed when you deploy the main template. This behavior lets you develop modular template specs, and reuse them as needed.

For example, you can create a template spec that deploys networking resources, and another template spec that deploys storage resources. In ARM templates, you link to these two template specs anytime you need to configure networking or storage resources.

The following example is similar to the earlier example, but you use the `id` property to link to a template spec rather than the `relativePath` property to link to a local template. Use `2020-06-01` for API version for the deployments resource. In the example, the template specs are in a resource group named **templateSpecsRG**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  ...
  "resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2020-06-01",
      "name": "networkingDeployment",
      ...
      "properties": {
        "mode": "Incremental",
        "templateLink": {
          "id": "[resourceId('templateSpecsRG', 'Microsoft.Resources/templateSpecs/versions', 'networkingSpec', '1.0a')]"
        }
      }
    },
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2020-06-01",
      "name": "storageDeployment",
      ...
      "properties": {
        "mode": "Incremental",
        "templateLink": {
          "id": "[resourceId('templateSpecsRG', 'Microsoft.Resources/templateSpecs/versions', 'storageSpec', '1.0a')]"
        }
      }
    }
  ],
  "outputs": {}
}
```

For more information about linking template specs, see Tutorial: Deploy a template spec as a linked template.

Next steps

- To create and deploy a template spec, see Quickstart: Create and deploy template spec.

- For more information about linking templates in template specs, see Tutorial: Create a template spec with linked templates.
- For more information about deploying a template spec as a linked template, see Tutorial: Deploy a template spec as a linked template.

Convert template gallery in portal to template specs

Article • 05/25/2022

The Azure portal provides a way to store Azure Resource Manager templates (ARM templates) in your account. However, template specs offers an easier way to share your templates with users in your organization, and link with other templates. This article shows how to convert existing templates in the template gallery to template specs.

To see if you have any templates to convert, view the template gallery in the portal. These templates have the resource type `Microsoft.Gallery/myareas/galleryitems`.

Deprecation of portal feature

The template gallery in the portal is being deprecated on March 31, 2025. To continue using a template in the template gallery, you need to migrate it to a template spec. Use one of the methods shown in this article to migrate the template.

Convert with PowerShell script

To simplify converting templates in the template gallery, use a PowerShell script from the Azure Quickstart Templates repo. When you run the script, you can either create a new template spec for each template or download a template that creates the template spec. The script doesn't delete the template from the template gallery.

1. Copy the migration script. Save a local copy with the name *Migrate-GalleryItems.ps1*.

2. To create new template specs, provide values for the `-ResourceGroupName` and `-Location` parameters.

   Set `ItemsToExport` to `MyGalleryItems` to export your templates. Set it to `AllGalleryItems` to export all templates you have access to.

   The following example creates new template specs for each template in a resource group named **migratedRG**. The script creates the resource group if it doesn't exist.

   Azure PowerShell

   ```
   .\Migrate-GalleryItems.ps1 -ResourceGroupName migratedRG -Location
   westus2 -ItemsToExport MyGalleryItems
   ```

3. To download templates that you can use to create the template specs, don't provide values for the resource group or location. Instead, specify `-ExportToFile`. The template isn't the same as your template in the gallery. Instead, it contains a template spec resource that creates the template spec for your template.

   The following example downloads the templates without creating template specs.

   Azure PowerShell

   ```
   .\Migrate-GalleryItems.ps1 -ItemsToExport MyGalleryItems -ExportToFile
   ```

   To learn how to deploy the template that creates the template spec, see Quickstart: Create and deploy template spec.

For more information about the script and its parameters, see Create TemplateSpecs from Template Gallery Templates.

Manually convert through portal

You can manually copy templates from the gallery to a new template specs.

1. Open the Templates (preview) in the portal.
2. Select the template to migrate.
3. Select **View Template**.
4. Copy the template content.
5. In the portal search bar, search for **Template specs**. Select that option.
6. Select **Create template spec**.
7. Provide values for the name, subscription, resource group, location, and version.
8. Select **Next: Edit template**.
9. For the contents of the template, paste the template you copied from the template gallery.
10. Select **Review + Create**.
11. After validation successfully completes, select **Create**.

If you need to share the template spec with other users in your organization, set role-based access control to the group or users that need access.

Next steps

To learn more about template specs, see Create and deploy template specs.

Integrate ARM templates with Azure Pipelines

Article • 04/09/2023

You can integrate Azure Resource Manager templates (ARM templates) with Azure Pipelines for continuous integration and continuous deployment (CI/CD). In this article, you learn two more advanced ways to deploy templates with Azure Pipelines.

Select your option

Before proceeding with this article, let's consider the different options for deploying an ARM template from a pipeline.

- **Use ARM template deployment task**. This option is the easiest option. This approach works when you want to deploy a template directly from a repository. This option isn't covered in this article but instead is covered in the tutorial Continuous integration of ARM templates with Azure Pipelines. It shows how to use the ARM template deployment task to deploy a template from your GitHub repo.

- **Add task that runs an Azure PowerShell script**. This option has the advantage of providing consistency throughout the development life cycle because you can use the same script that you used when running local tests. Your script deploys the template but can also perform other operations such as getting values to use as parameters. This option is shown in this article. See Azure PowerShell task.

  Visual Studio provides the Azure Resource Group project that includes a PowerShell script. The script stages artifacts from your project to a storage account that Resource Manager can access. Artifacts are items in your project such as linked templates, scripts, and application binaries. If you want to continue using the script from the project, use the PowerShell script task shown in this article.

- **Add tasks to copy and deploy tasks**. This option offers a convenient alternative to the project script. You configure two tasks in the pipeline. One task stages the artifacts to an accessible location. The other task deploys the template from that location. This option is shown in this article. See Copy and deploy tasks.

Prepare your project

This article assumes your ARM template and Azure DevOps organization are ready for creating the pipeline. The following steps show how to make sure you're ready:

- You have an Azure DevOps organization. If you don't have one, create one for free. If your team already has an Azure DevOps organization, make sure you're an administrator of the Azure DevOps project that you want to use.
- You've configured a service connection to your Azure subscription. The tasks in the pipeline execute under the identity of the service principal. For steps to create the connection, see Create a DevOps project.
- You have an ARM template that defines the infrastructure for your project.

Create pipeline

1. If you haven't added a pipeline previously, you need to create a new pipeline. From your Azure DevOps organization, select **Pipelines** and **New pipeline**.

2. Specify where your code is stored. The following image shows selecting **Azure Repos Git**.

3. From that source, select the repository that has the code for your project.

4. Select the type of pipeline to create. You can select **Starter pipeline**.

You're ready to either add an Azure PowerShell task or the copy file and deploy tasks.

Azure PowerShell task

This section shows how to configure continuous deployment by using a single task that runs the PowerShell script in your project. If you need a PowerShell script that deploys a template, see Deploy-AzTemplate.ps1 or Deploy-AzureResourceGroup.ps1.

The following YAML file creates an Azure PowerShell task:

yml

```yml
trigger:
- master

pool:
  vmImage: 'ubuntu-latest'

steps:
- task: AzurePowerShell@5
  inputs:
    azureSubscription: 'script-connection'
    ScriptType: 'FilePath'
    ScriptPath: './Deploy-AzTemplate.ps1'
    ScriptArguments: -Location 'centralus' -ResourceGroupName 'demogroup' -
```

```
TemplateFile templates\mainTemplate.json
    azurePowerShellVersion: 'LatestVersion'
```

When you set the task to `AzurePowerShell@5`, the pipeline uses the Az module. If you're using the AzureRM module in your script, set the task to `AzurePowerShell@3`.

YAML

```yaml
steps:
- task: AzurePowerShell@3
```

For `azureSubscription`, provide the name of the service connection you created.

YAML

```yaml
inputs:
    azureSubscription: '<your-connection-name>'
```

For `scriptPath`, provide the relative path from the pipeline file to your script. You can look in your repository to see the path.

YAML

```yaml
ScriptPath: '<your-relative-path>/<script-file-name>.ps1'
```

In `ScriptArguments`, provide any parameters needed by your script. The following example shows some parameters for a script, but you'll need to customize the parameters for your script.

YAML

```yaml
ScriptArguments: -Location 'centralus' -ResourceGroupName 'demogroup' -
TemplateFile templates\mainTemplate.json
```

When you select **Save**, the build pipeline is automatically run. Go back to the summary for your build pipeline, and watch the status.

You can select the currently running pipeline to see details about the tasks. When it finishes, you see the results for each step.

Copy and deploy tasks

This section shows how to configure continuous deployment by using two tasks. The first task stages the artifacts to a storage account and the second task deploys the template.

To copy files to a storage account, the service principal for the service connection must be assigned the Storage Blob Data Contributor or Storage Blob Data Owner role. For more information, see Get started with AzCopy.

The following YAML shows the Azure file copy task.

yml

```yml
trigger:
- master

pool:
  vmImage: 'windows-latest'

steps:
- task: AzureFileCopy@4
  inputs:
    SourcePath: 'templates'
    azureSubscription: 'copy-connection'
    Destination: 'AzureBlob'
    storage: 'demostorage'
    ContainerName: 'projecttemplates'
  name: AzureFileCopy
```

There are several parts of this task to revise for your environment. The `SourcePath` indicates the location of the artifacts relative to the pipeline file.

YAML

```
SourcePath: '<path-to-artifacts>'
```

For `azureSubscription`, provide the name of the service connection you created.

YAML

```
azureSubscription: '<your-connection-name>'
```

For storage and container name, provide the names of the storage account and container you want to use for storing the artifacts. The storage account must exist.

YAML

```
storage: '<your-storage-account-name>'
ContainerName: '<container-name>'
```

After creating the copy file task, you're ready to add the task to deploy the staged template.

The following YAML shows the Azure Resource Manager template deployment task:

YAML

```
- task: AzureResourceManagerTemplateDeployment@3
  inputs:
    deploymentScope: 'Resource Group'
    azureResourceManagerConnection: 'copy-connection'
    subscriptionId: '00000000-0000-0000-0000-000000000000'
    action: 'Create Or Update Resource Group'
    resourceGroupName: 'demogroup'
    location: 'West US'
    templateLocation: 'URL of the file'
    csmFileLink:
'$(AzureFileCopy.StorageContainerUri)templates/mainTemplate.json$(AzureFileC
opy.StorageContainerSasToken)'
    csmParametersFileLink:
'$(AzureFileCopy.StorageContainerUri)templates/mainTemplate.parameters.json$
(AzureFileCopy.StorageContainerSasToken)'
    deploymentMode: 'Incremental'
    deploymentName: 'deploy1'
```

There are several parts of this task to review in greater detail.

- `deploymentScope`: Select the scope of deployment from the options: `Management Group`, `Subscription`, and `Resource Group`. To learn more about the scopes, see Deployment scopes.

- `azureResourceManagerConnection`: Provide the name of the service connection you created.
- `subscriptionId`: Provide the target subscription ID. This property only applies to the Resource Group deployment scope and the subscription deployment scope.
- `resourceGroupName` and `location`: provide the name and location of the resource group you want to deploy to. The task creates the resource group if it doesn't exist.

  ```yml
  resourceGroupName: '<resource-group-name>'
  location: '<location>'
  ```

- `csmFileLink`: Provide the link for the staged template. When setting the value, use variables returned from the file copy task. The following example links to a template named mainTemplate.json. The folder named **templates** is included because that where the file copy task copied the file to. In your pipeline, provide the path to your template and the name of your template.

  ```yml
  csmFileLink:
  '$(AzureFileCopy.StorageContainerUri)templates/mainTemplate.json$(Azure
  FileCopy.StorageContainerSasToken)'
  ```

Your pipeline look like:

```yml
trigger:
- master

pool:
  vmImage: 'windows-latest'

steps:
- task: AzureFileCopy@4
  inputs:
    SourcePath: 'templates'
    azureSubscription: 'copy-connection'
    Destination: 'AzureBlob'
    storage: 'demostorage'
    ContainerName: 'projecttemplates'
  name: AzureFileCopy
- task: AzureResourceManagerTemplateDeployment@3
  inputs:
    deploymentScope: 'Resource Group'
```

```
    azureResourceManagerConnection: 'copy-connection'
    subscriptionId: '00000000-0000-0000-0000-000000000000'
    action: 'Create Or Update Resource Group'
    resourceGroupName: 'demogroup'
    location: 'West US'
    templateLocation: 'URL of the file'
    csmFileLink:
'$(AzureFileCopy.StorageContainerUri)templates/mainTemplate.json$(AzureFileC
opy.StorageContainerSasToken)'
    csmParametersFileLink:
'$(AzureFileCopy.StorageContainerUri)templates/mainTemplate.parameters.json$
(AzureFileCopy.StorageContainerSasToken)'
    deploymentMode: 'Incremental'
    deploymentName: 'deploy1'
```

When you select **Save**, the build pipeline is automatically run. Under the **Jobs** frame, select **Job** to see the job status.

Next steps

- To use the what-if operation in a pipeline, see Test ARM templates with What-If in a pipeline.
- To learn about using ARM templates with GitHub Actions, see Deploy Azure Resource Manager templates by using GitHub Actions.

Deploy ARM templates by using GitHub Actions

Article • 04/09/2023

GitHub Actions is a suite of features in GitHub to automate your software development workflows in the same place you store code and collaborate on pull requests and issues.

Use the Deploy Azure Resource Manager Template Action to automate deploying an Azure Resource Manager template (ARM template) to Azure.

Prerequisites

- An Azure account with an active subscription. Create an account for free.
- A GitHub account. If you don't have one, sign up for free.
  - A GitHub repository to store your Resource Manager templates and your workflow files. To create one, see Creating a new repository.

Workflow file overview

A workflow is defined by a YAML (.yml) file in the `/.github/workflows/` path in your repository. This definition contains the various steps and parameters that make up the workflow.

The file has two sections:

Section	Tasks
Authentication	1. Generate deployment credentials.
Deploy	1. Deploy the Resource Manager template.

Generate deployment credentials

Service principal

Create a service principal with the az ad sp create-for-rbac command in the Azure CLI. Run this command with Azure Cloud Shell in the Azure portal or by selecting

the **Try it** button.

Azure CLI

```
az ad sp create-for-rbac --name "myML" --role contributor \
                            --scopes /subscriptions/<subscription-
id>/resourceGroups/<group-name> \
                            --sdk-auth
```

In the example above, replace the placeholders with your subscription ID, resource group name, and app name. The output is a JSON object with the role assignment credentials that provide access to your App Service app similar to below. Copy this JSON object for later.

Output

```
{
  "clientId": "<GUID>",
  "clientSecret": "<GUID>",
  "subscriptionId": "<GUID>",
  "tenantId": "<GUID>",
  (...)
}
```

Configure the GitHub secrets

Service principal

1. In GitHub, go to your repository.
2. Select **Security** > **Secrets and variables** > **Actions**.

3. Select **New repository secret**.
4. Paste the entire JSON output from the Azure CLI command into the secret's value field. Give the secret the name `AZURE_CREDENTIALS`.

5. Select **Add secret**.

Add Resource Manager template

Add a Resource Manager template to your GitHub repository. This template creates a storage account.

```
https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.storage/storage-account-create/azuredeploy.json
```

You can put the file anywhere in the repository. The workflow sample in the next section assumes the template file is named **azuredeploy.json**, and it is stored at the root of your repository.

Create workflow

The workflow file must be stored in the **.github/workflows** folder at the root of your repository. The workflow file extension can be either **.yml** or **.yaml**.

1. From your GitHub repository, select **Actions** from the top menu.
2. Select **New workflow**.
3. Select **set up a workflow yourself**.
4. Rename the workflow file if you prefer a different name other than **main.yml**. For example: **deployStorageAccount.yml**.
5. Replace the content of the yml file with the following:

Service principal

```yml
on: [push]
name: Azure ARM
jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:

      # Checkout code
    - uses: actions/checkout@main
```

```
        # Log into Azure
      - uses: azure/login@v1
        with:
          creds: ${{ secrets.AZURE_CREDENTIALS }}

        # Deploy ARM template
      - name: Run ARM deploy
        uses: azure/arm-deploy@v1
        with:
          subscriptionId: ${{ secrets.AZURE_SUBSCRIPTION }}
          resourceGroupName: ${{ secrets.AZURE_RG }}
          template: ./azuredeploy.json
          parameters: storageAccountType=Standard_LRS

        # output containerName variable from template
      - run: echo ${{ steps.deploy.outputs.containerName }}
```

> ⓘ **Note**
>
> You can specify a JSON format parameters file instead in the ARM Deploy action (example: `.azuredeploy.parameters.json`).

The first section of the workflow file includes:

- **name**: The name of the workflow.
- **on**: The name of the GitHub events that triggers the workflow. The workflow is trigger when there is a push event on the main branch, which modifies at least one of the two files specified. The two files are the workflow file and the template file.

1. Select **Start commit**.
2. Select **Commit directly to the main branch**.
3. Select **Commit new file** (or **Commit changes**).

Because the workflow is configured to be triggered by either the workflow file or the template file being updated, the workflow starts right after you commit the changes.

Check workflow status

1. Select the **Actions** tab. You will see a **Create deployStorageAccount.yml** workflow listed. It takes 1-2 minutes to run the workflow.
2. Select the workflow to open it.
3. Select **Run ARM deploy** from the menu to verify the deployment.

Clean up resources

When your resource group and repository are no longer needed, clean up the resources you deployed by deleting the resource group and your GitHub repository.

Next steps

Create your first ARM template

Learn module: Automate the deployment of ARM templates by using GitHub Actions

Creating and deploying Azure resource groups through Visual Studio

Article • 06/10/2021

With Visual Studio, you can create a project that deploys your infrastructure and code to Azure. For example, you can deploy the web host, website, and code for the website. Visual Studio provides many different starter templates for deploying common scenarios. In this article, you deploy a web app.

This article shows how to use Visual Studio 2019 or later with the Azure development and ASP.NET workloads installed. If you use Visual Studio 2017, your experience is largely the same.

Create Azure Resource Group project

In this section, you create an Azure Resource Group project with a **Web app** template.

1. In Visual Studio, choose **File**>**New**>**Project**.

2. Select the **Azure Resource Group** project template and **Next**.

3. Give your project a name. The other default settings are probably fine, but review them to make they work for your environment. When done, select **Create**.

4. 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 article, choose the **Web app** template and **OK**.

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

5. Visual Studio creates a resource group deployment project for the web app. To see the files for your project, look at the node in the deployment project.

Since you chose the Web app template, you see the following files:

File name	Description
Deploy-AzureResourceGroup.ps1	A PowerShell script that runs PowerShell commands to deploy to Azure Resource Manager. Visual Studio uses this PowerShell script to deploy your template.
WebSite.json	The Resource Manager template that defines the infrastructure you want deploy to Azure, and the parameters you can provide during deployment. It also defines the dependencies between the resources so Resource Manager deploys the resources in the correct order.
WebSite.parameters.json	A parameters file that has values needed by the template. You pass in parameter values to customize each deployment. Notice that **Build Action** is set to **Content**. If you add more parameter files, make sure the build action is set to **Content**.

All resource group deployment projects have these basic files. Other projects may have more files to support other functionality.

Customize Resource Manager template

You can customize a deployment project by modifying the Resource Manager template that describes the resources you want to deploy. To learn about the elements of the Resource Manager template, see Authoring Azure Resource Manager templates.

1. To work on your template, open **WebSite.json**.

2. 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.

3. Select an element in the outline to go to that part of the template.

4. 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**.

5. 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.

6. 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.

7. The parameter for the type of storage account is pre-defined with allowed types and a default type. You can leave these values or edit them for your scenario. If you don't want anyone to deploy a **Premium_LRS** storage account through this template, remove it from the allowed types.

JSON

```
"demoaccountType": {
  "type": "string",
  "defaultValue": "Standard_LRS",
  "allowedValues": [
    "Standard_LRS",
    "Standard_ZRS",
    "Standard_GRS",
    "Standard_RAGRS"
```

```
    ]
  }
```

8. Visual Studio also provides intellisense to help you understand the properties that 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, and save the file.

JSON

```
"properties": {
  "name": "[parameters('hostingPlanName')]",
  "numberOfWorkers": 1
}
```

9. Open the **WebSite.parameters.json** file. You use the parameters file to pass in values during deployment that customize the resource being deployed. Give the hosting plan a name, and save the file.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
```

```
  "parameters": {
    "hostingPlanName": {
      "value": "demoHostPlan"
    }
  }
}
```

Deploy project to Azure

You're now ready to deploy your project to a resource group.

By default, the PowerShell script (Deploy-AzureResourceGroup.ps1) in the project uses the AzureRM module. If you still have the AzureRM module installed and want to continue using it, you can use this default script. With this script, you can use the Visual Studio interface to deploy your solution.

However, if you've migrated to the new Az module, you need to add a new script to your project. To add a script that uses the Az module, copy the Deploy-AzTemplate.ps1 script and add it to your project. To use this script for deployment, you must run it from a PowerShell console, rather than using Visual Studio's deployment interface.

Both approaches are shown in this article. This article refers to the default script as the AzureRM module script, and the new script as the Az module script.

Az module script

For the Az module script, open a PowerShell console and run:

PowerShell

```
.\Deploy-AzTemplate.ps1 -ArtifactStagingDirectory . -Location centralus -
TemplateFile WebSite.json -TemplateParametersFile WebSite.parameters.json
```

AzureRM module script

For the AzureRM module script, use Visual Studio:

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

2. The **Deploy to Resource Group** dialog box appears. In the **Resource group** dropdown box, choose an existing resource group or create a new one. Select **Deploy**.

3. 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:

Output

```
18:00:58 - Successfully deployed template 'website.json' to resource
group 'ExampleAppDeploy'.
```

View deployed resources

Let's check the results.

1. In a browser, open the Azure portal and sign in to your account. To see the resource group, select **Resource groups** and the resource group you deployed to.

2. You see all the deployed resources. Notice that the name of the storage account isn't 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 create a unique name.

Add code to project

At this point, you've deployed the infrastructure for your app, but there's no actual code deployed with the project.

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

2. Add an **ASP.NET Core Web Application**.

3. Give your web app a name, and select **Create**.

4. Select **Web Application** and **Create**.

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

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

7. 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 sets some properties. You see these properties in the **Properties** window for the reference. The **Include File Path** has the path where the package is created. Note the folder (ExampleApp) and file (package.zip). You need to know these values because you provide them as parameters when deploying the app.

8. Go back to your template (WebSite.json) and add a resource to the template.

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

Save your template.

10. There are some new parameters in your template. They were added in the previous step. You don't need to provide values for **_artifactsLocation** or **_artifactsLocationSasToken** because those values are automatically generated. However, you have to set the folder and file name to the path that contains the deployment package. The names of these parameters end with **PackageFolder** and **PackageFileName**. The first part of the name is the name of the Web Deploy resource you added. In this article, they're named **ExampleAppPackageFolder** and **ExampleAppPackageFileName**.

    Open **Website.parameters.json** and set those parameters to the values you saw in the reference properties. Set **ExampleAppPackageFolder** to the name of the folder. Set **ExampleAppPackageFileName** to the name of the zip file.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "hostingPlanName": {
      "value": "demoHostPlan"
    },
    "ExampleAppPackageFolder": {
```

```
      "value": "ExampleApp"
    },
    "ExampleAppPackageFileName": {
      "value": "package.zip"
    }
  }
}
```

Deploy code with infrastructure

Because you added code to the project, your deployment is a little different this time. During deployment, you stage artifacts for your project to a place that Resource Manager can access. The artifacts are staged to a storage account.

Az module script

There's one small change you need to make to your template if you're using the Az module script. This script adds a slash to the artifacts location but your template doesn't expect that slash. Open WebSite.json and find the properties for the MSDeploy extension. It has a property named **packageUri**. Remove the slash between the artifacts location and the package folder.

It should look like:

JSON

```
"packageUri": "[concat(parameters('_artifactsLocation'),
parameters('ExampleAppPackageFolder'), '/',
parameters('ExampleAppPackageFileName'),
parameters('_artifactsLocationSasToken'))]",
```

Notice in the preceding example there's no `'/',` between **parameters('_artifactsLocation')** and **parameters('ExampleAppPackageFolder')**.

Rebuild the project. Building the project makes sure the files you need to deploy are added to the staging folder.

Now, open a PowerShell console and run:

PowerShell

```
.\Deploy-AzTemplate.ps1 -ArtifactStagingDirectory
.\bin\Debug\staging\ExampleAppDeploy -Location centralus -TemplateFile
WebSite.json -TemplateParametersFile WebSite.parameters.json -
UploadArtifacts -StorageAccountName <storage-account-name>
```

AzureRM module script

For the AzureRM module script, use Visual Studio:

1. To redeploy, choose **Deploy**, and the resource group you deployed earlier.

2. Select the storage account you deployed with this resource group for the **Artifact storage account**.

View web app

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

2. Notice that you've successfully deployed the default ASP.NET app.

Add operations dashboard

You aren't limited to only the resources that are available through the Visual Studio interface. You can customize your deployment by adding a custom resource to your template. To show adding a resource, you add an operational dashboard to manage the resource you deployed.

1. Open the WebSite.json file and add the following JSON after the storage account resource but before the closing `]` of the resources section.

```
JSON
```

```
,{
  "properties": {
    "lenses": {
      "0": {
        "order": 0,
        "parts": {
          "0": {
            "position": {
              "x": 0,
              "y": 0,
              "colSpan": 4,
              "rowSpan": 6
            },
            "metadata": {
              "inputs": [
                {
                  "name": "resourceGroup",
                  "isOptional": true
                },
                {
                  "name": "id",
                  "value": "[resourceGroup().id]",
                  "isOptional": true
                }
              ],
              "type":
"Extension/HubsExtension/PartType/ResourceGroupMapPinnedPart"
            }
          },
          "1": {
            "position": {
              "x": 4,
              "y": 0,
              "rowSpan": 3,
              "colSpan": 4
            },
            "metadata": {
              "inputs": [],
              "type":
"Extension[azure]/HubsExtension/PartType/MarkdownPart",
              "settings": {
                "content": {
                  "settings": {
                    "content": "__Customizations__\n\nUse this
dashboard to create and share the operational views of services
critical to the application performing. To customize simply pin
components to the dashboard and then publish when you're done. Others
will see your changes when you publish and share the dashboard.\n\nYou
can customize this text too. It supports plain text, __Markdown__, and
even limited HTML like images <img width='10'
src='https://portal.azure.com/favicon.ico'/> and <a
href='https://azure.microsoft.com' target='_blank'>links</a> that open
in a new tab.\n",
                    "title": "Operations",
```

```
                    "subtitle": "[resourceGroup().name]"
                  }
                }
              }
            }
          }
        }
      }
    },
    "metadata": {
      "model": {
        "timeRange": {
          "value": {
            "relative": {
              "duration": 24,
              "timeUnit": 1
            }
          },
          "type":
"MsPortalFx.Composition.Configuration.ValueTypes.TimeRange"
        }
      }
    }
  },
  "type": "Microsoft.Portal/dashboards",
  "apiVersion": "2015-08-01-preview",
  "name": "[concat('ARM-',resourceGroup().name)]",
  "location": "[resourceGroup().location]",
  "tags": {
    "hidden-title": "[concat('OPS-',resourceGroup().name)]"
  }
}
```

2. Redeploy your project.

3. After deployment has finished, view your dashboard in the portal. Select **Dashboard** and pick the one you deployed.

4. You see the customized dashboard.

You can manage access to the dashboard by using Azure role-based access control (Azure RBAC). You can also customize the dashboard's appearance after it's deployed. However, if you redeploy the resource group, the dashboard is reset to its default state in your template. For more information about creating dashboards, see Programmatically create Azure Dashboards.

Clean up resources

When the Azure resources are no longer needed, clean up the resources you deployed by deleting the resource group.

1. From the Azure portal, select **Resource groups** from the left menu.

2. Select the resource group name.

3. Select **Delete resource group** from the top menu.

Next steps

In this article, you learned how to create and deploy templates using Visual Studio. To learn more about template development, see our new beginner tutorial series:

Beginner tutorials

Update Visual Studio template deployment script to use Az PowerShell module

Article • 05/11/2023

Visual Studio 16.4 supports using the Az PowerShell module in the template deployment script. However, Visual Studio doesn't automatically install that module. To use the Az module, you need to take four steps:

1. Uninstall AzureRM module
2. Install Az module
3. Update Visual Studio to 16.4
4. Update the deployment script in your project.

Update Visual Studio to 16.4

Update your installation of Visual Studio to version 16.4 or later. During the upgrade, make sure the Azure PowerShell component isn't checked. Since you installed the Az module through the gallery, you don't want to reinstall the AzureRM module.

If you already upgraded to 16.4 and the Azure PowerShell component was checked, you can uninstall it by running the Visual Studio Installer. Don't select the Azure PowerShell component in the Azure Workload or on the individual components page.

Update the deployment script in your project

Replace all occurrences of the string 'AzureRm' with 'Az' in your deployment script. You can refer to revisions in this gist to see the changes. For more information about upgrading scripts to the Az module, see Migrate Azure PowerShell from AzureRM to Az.

Next steps

To learn about using the Visual Studio project, see Creating and deploying Azure resource group projects through Visual Studio.

Create and deploy ARM templates by using the IntelliJ IDEA

Article • 04/09/2023

Learn how to deploy an Azure Resource Manager template (ARM template) to Azure using the IntelliJ IDEA, and the process of editing and update the template directly from the IDE. ARM templates are JSON files that define the resources you need to deploy for your solution. To understand the concepts associated with deploying and managing your Azure solutions, see the template deployment overview.

After completing the tutorial, you deploy an Azure Storage account. The same process can be used to deploy other Azure resources.

If you don't have an Azure subscription, create a free account before you begin.

Prerequisites

To complete this article, you need:

- An IntelliJ IDEA Ultimate Edition or Community Edition installed
- The Azure Toolkit for IntelliJ installed, check IntelliJ's plugins management guide for more information
- Be signed in to your Azure account for the Azure Toolkit for IntelliJ

Deploy a Quickstart template

Instead of creating a template from scratch, you open a template from Azure Quickstart Templates. Azure Quickstart Templates is a repository for ARM templates. The template used in this article is called Create a standard storage account. It defines an Azure Storage account resource.

1. Right-click and save the azuredeploy.json and azuredeploy.parameters.json to your local computer.

2. If your Azure Toolkit are properly installed and signed-in, you should see Azure Explorer in your IntelliJ IDEA's sidebar. Right-click on the **Resource Management** and select **Create Deployment**.

3. Config your **Deployment Name**, **Subscription**, **Resource Group**, and **Region**. Here we deploy the template into a new resource group `testRG`. Then, select path for **Resource Template** as `azuredeploy.json` and **Resource Parameters** as `azuredeploy.parameters.json` you downloaded.

4. After you click OK, the deployment is started. Until the deployment complete, you can find the progress from IntelliJ IDEA's **status bar** on the bottom.

Browse an existing deployment

1. After the deployment is done, you can see the new resource group `testRG` and a new deployment created. Right-click on the deployment and you can see a list of possible actions. Now select **Show Properties**.

2. A tab view will be open to show some useful properties like deployment status and template structure.

Edit and update an existing deployment

1. Select **Edit Deployment** from right-click menu or the show properties view before. Another tab view will be open, showing the template and parameter files for the

deployment on Azure. To save those files to local, you could click **Export Template File** or **Export Parameter Files**.

2. You can edit the two files on this page and deploy the changes to Azure. Here we modify the value of **storageAccountType** in parameter files, from `Standard_LRS` to `Standard_GRS`. Then, click **Update Deployment** on the bottom and confirm the update.

3. After update deployment completed, you can verify on the portal that the created storage account is changed to `Standard_GRS`.

Clean up resources

1. When the Azure resources are no longer needed, clean up the resources you deployed by deleting the resource group. You can do it from Azure portal or Azure CLI. In Azure Explorer from IntelliJ IDEA, right click on your created **resource group** and select delete.

> **Note**
>
> Notice that delete a deployment will not delete resources created by the deployment. Please delete corresponding resource group or specific resources if you no longer need them.

Next steps

The main focus of this article is to use IntelliJ IDEA to deploy an existing template from Azure Quickstart templates. You also learned how to view and update an existing deployment on Azure. The templates from Azure Quickstart templates might not give you everything you need. To learn more about template development, see our new beginner tutorial series:

Beginner tutorials

Visit Java on Azure Dev center

Define resources with Bicep, ARM templates, and Terraform AzAPI provider

Article • 03/22/2023

When deploying Azure resources with an Infrastructure as Code tool, you need to understand what resource types are available, and what values to use in your files. The Azure resource reference documentation provides these values. The syntax is shown for Bicep, ARM template JSON, and Terraform AzAPI provider.

Choose language

Select the deployment language you wish to use for viewing the resource reference. The options are available at the top of each article.

Bicep

For an introduction to working with Bicep files, see Quickstart: Create Bicep files with Visual Studio Code. To learn about the sections of a Bicep file, see Understand the structure and syntax of Bicep files.

To learn about Bicep files through a guided set of Learn modules, see Deploy and manage resources in Azure by using Bicep.

Microsoft recommends that you use VS Code to create Bicep files. For more information, see Install Bicep tools.

ARM templates

> **Tip**
>
> Bicep is a new language that offers the same capabilities as ARM templates but with a syntax that's easier to use. If you're deciding between the two languages, we recommend Bicep.

To learn about the sections of an ARM template, see Understand the structure and syntax of ARM templates. For an introduction to working with templates, see Tutorial: Create and deploy your first ARM template.

Microsoft recommends that you use VS Code to create ARM templates. When you add the Azure Resource Managed tools extension, you get intellisense for the template properties. For more information, see Quickstart: Create ARM templates with Visual Studio Code.

Terraform AzAPI provider

To learn about the Terraform AzAPI provider, see Overview of the Terraform AzAPI provider.

For an introduction to creating a configuration file for the Terraform AzAPI provider, see Quickstart: Deploy your first Azure resource with the AzAPI Terraform provider.

Find resources

If you know the resource type, you can go directly to it with the following URL format: `https://learn.microsoft.com/azure/templates/{provider-namespace}/{resource-type}`. For example, the SQL database reference content is available at: https://learn.microsoft.com/azure/templates/microsoft.sql/servers/databases.

The resource types are located under the Reference node. Expand the resource provider that contains the type you are looking for. The following image shows the types for Storage.

Or, you can filter the resource types in navigation pane:

See changes in versions

Each resource provider contains a list of changes for each API version. You can locate the change log in the left navigation pane.

- Storage
  - Storage Accounts
  - Storage Accounts/
  - (API versions)
  - (Change log)
    - Summary
    - Resource types
      - Storage Accounts
      - Storage Accounts/

ARM template functions

Article • 04/09/2023

This article describes all the functions you can use in an Azure Resource Manager template (ARM template). For information about using functions in your template, see template syntax.

To create your own functions, see User-defined functions.

Most functions work the same when deployed to a resource group, subscription, management group, or tenant. A few functions can't be used in all scopes. They're noted in the lists below.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **Bicep functions** and **Bicep operators**.

Any function

The any function is available in Bicep to help resolve issues around data type warnings.

Array functions

Resource Manager provides several functions for working with arrays.

- array
- concat
- contains
- createArray
- empty
- first
- indexOf
- intersection
- last
- lastIndexOf
- length
- min

- max
- range
- skip
- take
- union

For Bicep files, use the array functions.

Comparison functions

Resource Manager provides several functions for making comparisons in your templates.

- coalesce
- equals
- less
- lessOrEquals
- greater
- greaterOrEquals

For Bicep files, use the coalesce logical operator. For comparisons, use the comparison operators.

Date functions

Resource Manager provides the following functions for working with dates.

- dateTimeAdd
- dateTimeFromEpoch
- dateTimeToEpoch
- utcNow

For Bicep files, use the date functions.

Deployment value functions

Resource Manager provides the following functions for getting values from sections of the template and values related to the deployment:

- deployment
- environment
- parameters

- variables

For Bicep files, use the deployment functions.

Logical functions

Resource Manager provides the following functions for working with logical conditions:

- and
- bool
- false
- if
- not
- or
- true

For Bicep files, use the bool logical function. For other logical values, use logical operators.

Numeric functions

Resource Manager provides the following functions for working with integers:

- add
- copyIndex
- div
- float
- int
- min
- max
- mod
- mul
- sub

For Bicep files that use `int`, `min`, and `max` use numeric functions. For other numeric values, use numeric operators.

Object functions

Resource Manager provides several functions for working with objects.

- contains

- createObject
- empty
- intersection
- items
- json
- length
- null
- union

For Bicep files, use the object functions.

Resource functions

Resource Manager provides the following functions for getting resource values:

- extensionResourceId
- listAccountSas
- listKeys
- listSecrets
- list*
- pickZones
- providers (deprecated)
- reference
- resourceId - can be used at any scope, but the valid parameters change depending on the scope.
- subscriptionResourceId
- tenantResourceId

For Bicep files, use the resource functions.

Scope functions

Resource Manager provides the following functions for getting deployment scope values:

- managementGroup - can only be used in deployments to a management group.
- resourceGroup - can only be used in deployments to a resource group.
- subscription - can only be used in deployments to a resource group or subscription.
- tenant - can be used for deployments at any scope.

For Bicep files, use the scope functions.

String functions

Resource Manager provides the following functions for working with strings:

- base64
- base64ToJson
- base64ToString
- concat
- contains
- dataUri
- dataUriToString
- empty
- endsWith
- first
- format
- guid
- indexOf
- join
- last
- lastIndexOf
- length
- newGuid
- padLeft
- replace
- skip
- split
- startsWith
- string
- substring
- take
- toLower
- toUpper
- trim
- uniqueString
- uri
- uriComponent
- uriComponentToString

For Bicep files, use the string functions.

Next steps

- For a description of the sections in an ARM template, see Understand the structure and syntax of ARM templates.
- To merge multiple templates, see Using linked and nested templates when deploying Azure resources.
- To iterate a specified number of times when creating a type of resource, see Resource iteration in ARM templates.
- To see how to deploy the template you've created, see Deploy resources with ARM templates and Azure PowerShell.

Array functions for ARM templates

Article • 04/09/2023

This article describes the template functions for working with arrays.

To get an array of string values delimited by a value, see split.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **array** functions.

array

`array(convertToArray)`

Converts the value to an array.

In Bicep, use the array function.

Parameters

Parameter	Required	Type	Description
convertToArray	Yes	int, string, array, or object	The value to convert to an array.

Return value

An array.

Example

The following example shows how to use the array function with different types.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
```

```
    "intToConvert": {
      "type": "int",
      "defaultValue": 1
    },
    "stringToConvert": {
      "type": "string",
      "defaultValue": "efgh"
    },
    "objectToConvert": {
      "type": "object",
      "defaultValue": {
        "a": "b",
        "c": "d"
      }
    }
  },
  "resources": [
  ],
  "outputs": {
    "intOutput": {
      "type": "array",
      "value": "[array(parameters('intToConvert'))]"
    },
    "stringOutput": {
      "type": "array",
      "value": "[array(parameters('stringToConvert'))]"
    },
    "objectOutput": {
      "type": "array",
      "value": "[array(parameters('objectToConvert'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
intOutput	Array	[1]
stringOutput	Array	["efgh"]
objectOutput	Array	[{"a": "b", "c": "d"}]

concat

`concat(arg1, arg2, arg3, ...)`

Combines multiple arrays and returns the concatenated array, or combines multiple string values and returns the concatenated string.

In Bicep, use the concat function.

Parameters

Parameter	Required	Type	Description
arg1	Yes	array or string	The first array or string for concatenation.
more arguments	No	array or string	More arrays or strings in sequential order for concatenation.

This function can take any number of arguments, and can accept either strings or arrays for the parameters. However, you can't provide both arrays and strings for parameters. Arrays are only concatenated with other arrays.

Return value

A string or array of concatenated values.

Example

The following example shows how to combine two arrays.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "firstArray": {
      "type": "array",
      "defaultValue": [
        "1-1",
        "1-2",
        "1-3"
      ]
    },
    "secondArray": {
      "type": "array",
      "defaultValue": [
        "2-1",
        "2-2",
        "2-3"
      ]
    }
  },
```

```
  "resources": [
  ],
  "outputs": {
    "return": {
      "type": "array",
      "value": "[concat(parameters('firstArray'),
parameters('secondArray'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
return	Array	["1-1", "1-2", "1-3", "2-1", "2-2", "2-3"]

The following example shows how to combine two string values and return a concatenated string.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "prefix": {
      "type": "string",
      "defaultValue": "prefix"
    }
  },
  "resources": [],
  "outputs": {
    "concatOutput": {
      "type": "string",
      "value": "[concat(parameters('prefix'), '-',
uniqueString(resourceGroup().id))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
concatOutput	String	prefix-5yj4yjf5mbg72

contains

`contains(container, itemToFind)`

Checks whether an array contains a value, an object contains a key, or a string contains a substring. The string comparison is case-sensitive. However, when testing if an object contains a key, the comparison is case-insensitive.

In Bicep, use the contains function.

Parameters

Parameter	Required	Type	Description
container	Yes	array, object, or string	The value that contains the value to find.
itemToFind	Yes	string or int	The value to find.

Return value

True if the item is found; otherwise, **False**.

Example

The following example shows how to use contains with different types:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "stringToTest": {
      "type": "string",
      "defaultValue": "OneTwoThree"
    },
    "objectToTest": {
      "type": "object",
      "defaultValue": {
        "one": "a",
        "two": "b",
        "three": "c"
      }
    },
    "arrayToTest": {
      "type": "array",
```

```
      "defaultValue": [ "one", "two", "three" ]
    }
  },
  "resources": [
  ],
  "outputs": {
    "stringTrue": {
      "type": "bool",
      "value": "[contains(parameters('stringToTest'), 'e')]"
    },
    "stringFalse": {
      "type": "bool",
      "value": "[contains(parameters('stringToTest'), 'z')]"
    },
    "objectTrue": {
      "type": "bool",
      "value": "[contains(parameters('objectToTest'), 'one')]"
    },
    "objectFalse": {
      "type": "bool",
      "value": "[contains(parameters('objectToTest'), 'a')]"
    },
    "arrayTrue": {
      "type": "bool",
      "value": "[contains(parameters('arrayToTest'), 'three')]"
    },
    "arrayFalse": {
      "type": "bool",
      "value": "[contains(parameters('arrayToTest'), 'four')]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
stringTrue	Bool	True
stringFalse	Bool	False
objectTrue	Bool	True
objectFalse	Bool	False
arrayTrue	Bool	True
arrayFalse	Bool	False

createArray

`createArray(arg1, arg2, arg3, ...)`

Creates an array from the parameters.

In Bicep, the `createArray` function isn't supported. To construct an array, see the Bicep array data type.

Parameters

Parameter	Required	Type	Description
args	No	String, Integer, Array, or Object	The values in the array.

Return value

An array. When no parameters are provided, it returns an empty array.

Example

The following example shows how to use createArray with different types:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "objectToTest": {
      "type": "object",
      "defaultValue": {
        "one": "a",
        "two": "b",
        "three": "c"
      }
    },
    "arrayToTest": {
      "type": "array",
      "defaultValue": [ "one", "two", "three" ]
    }
  },
  "resources": [
  ],
  "outputs": {
    "stringArray": {
      "type": "array",
      "value": "[createArray('a', 'b', 'c')]"
    },
```

```
    "intArray": {
      "type": "array",
      "value": "[createArray(1, 2, 3)]"
    },
    "objectArray": {
      "type": "array",
      "value": "[createArray(parameters('objectToTest'))]"
    },
    "arrayArray": {
      "type": "array",
      "value": "[createArray(parameters('arrayToTest'))]"
    },
    "emptyArray": {
      "type": "array",
      "value": "[createArray()]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
stringArray	Array	["a", "b", "c"]
intArray	Array	[1, 2, 3]
objectArray	Array	[{"one": "a", "two": "b", "three": "c"}]
arrayArray	Array	[["one", "two", "three"]]
emptyArray	Array	[]

empty

`empty(itemToTest)`

Determines if an array, object, or string is empty.

In Bicep, use the empty function.

Parameters

Parameter	Required	Type	Description
itemToTest	Yes	array, object, or string	The value to check if it's empty.

Return value

Returns **True** if the value is empty; otherwise, **False**.

Example

The following example checks whether an array, object, and string are empty.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "testArray": {
      "type": "array",
      "defaultValue": []
    },
    "testObject": {
      "type": "object",
      "defaultValue": {}
    },
    "testString": {
      "type": "string",
      "defaultValue": ""
    }
  },
  "resources": [
  ],
  "outputs": {
    "arrayEmpty": {
      "type": "bool",
      "value": "[empty(parameters('testArray'))]"
    },
    "objectEmpty": {
      "type": "bool",
      "value": "[empty(parameters('testObject'))]"
    },
    "stringEmpty": {
      "type": "bool",
      "value": "[empty(parameters('testString'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value

Name	Type	Value
arrayEmpty	Bool	True
objectEmpty	Bool	True
stringEmpty	Bool	True

first

`first(arg1)`

Returns the first element of the array, or first character of the string.

In Bicep, use the first function.

Parameters

Parameter	Required	Type	Description
arg1	Yes	array or string	The value to retrieve the first element or character.

Return value

The type (string, int, array, or object) of the first element in an array, or the first character of a string.

Example

The following example shows how to use the first function with an array and string.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "arrayToTest": {
      "type": "array",
      "defaultValue": [ "one", "two", "three" ]
    }
  },
  "resources": [
  ],
```

```
  "outputs": {
    "arrayOutput": {
      "type": "string",
      "value": "[first(parameters('arrayToTest'))]"
    },
    "stringOutput": {
      "type": "string",
      "value": "[first('One Two Three')]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
arrayOutput	String	one
stringOutput	String	O

indexOf

`indexOf(arrayToSearch, itemToFind)`

Returns an integer for the index of the first occurrence of an item in an array. The comparison is **case-sensitive** for strings.

Parameters

Parameter	Required	Type	Description
arrayToSearch	Yes	array	The array to use for finding the index of the searched item.
itemToFind	Yes	int, string, array, or object	The item to find in the array.

Return value

An integer representing the first index of the item in the array. The index is zero-based. If the item isn't found, -1 is returned.

Examples

The following example shows how to use the indexOf and lastIndexOf functions:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "variables": {
    "names": [
      "one",
      "two",
      "three"
    ],
    "numbers": [
      4,
      5,
      6
    ],
    "collection": [
      "[variables('names')]",
      "[variables('numbers')]"
    ],
    "duplicates": [
      1,
      2,
      3,
      1
    ]
  },
  "resources": [],
  "outputs": {
    "index1": {
      "type": "int",
      "value": "[lastIndexOf(variables('names'), 'two')]"
    },
    "index2": {
      "type": "int",
      "value": "[indexOf(variables('names'), 'one')]"
    },
    "notFoundIndex1": {
      "type": "int",
      "value": "[lastIndexOf(variables('names'), 'Three')]"
    },
    "index3": {
      "type": "int",
      "value": "[lastIndexOf(variables('numbers'), 4)]"
    },
    "index4": {
      "type": "int",
      "value": "[indexOf(variables('numbers'), 6)]"
    },
    "notFoundIndex2": {
      "type": "int",
```

```
        "value": "[lastIndexOf(variables('numbers'), '5')]"
      },
      "index5": {
        "type": "int",
        "value": "[indexOf(variables('collection'), variables('numbers'))]"
      },
      "index6": {
        "type": "int",
        "value": "[indexOf(variables('duplicates'), 1)]"
      },
      "index7": {
        "type": "int",
        "value": "[lastIndexOf(variables('duplicates'), 1)]"
      }
    }
  }
```

The output from the preceding example is:

Name	Type	Value
index1	int	1
index2	int	0
index3	int	0
index4	int	2
index5	int	1
index6	int	0
index7	int	3
notFoundIndex1	int	-1
notFoundIndex2	int	-1

intersection

`intersection(arg1, arg2, arg3, ...)`

Returns a single array or object with the common elements from the parameters.

In Bicep, use the intersection function.

Parameters

Parameter	Required	Type	Description
arg1	Yes	array or object	The first value to use for finding common elements.
arg2	Yes	array or object	The second value to use for finding common elements.
more arguments	No	array or object	More values to use for finding common elements.

Return value

An array or object with the common elements.

Example

The following example shows how to use intersection with arrays and objects.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "firstObject": {
      "type": "object",
      "defaultValue": {
        "one": "a",
        "two": "b",
        "three": "c"
      }
    },
    "secondObject": {
      "type": "object",
      "defaultValue": {
        "one": "a",
        "two": "z",
        "three": "c"
      }
    },
    "firstArray": {
      "type": "array",
      "defaultValue": [ "one", "two", "three" ]
    },
    "secondArray": {
      "type": "array",
      "defaultValue": [ "two", "three" ]
    }
```

```
  },
  "resources": [
  ],
  "outputs": {
    "objectOutput": {
      "type": "object",
      "value": "[intersection(parameters('firstObject'),
parameters('secondObject'))]"
    },
    "arrayOutput": {
      "type": "array",
      "value": "[intersection(parameters('firstArray'),
parameters('secondArray'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
objectOutput	Object	{"one": "a", "three": "c"}
arrayOutput	Array	["two", "three"]

last

`last(arg1)`

Returns the last element of the array, or last character of the string.

In Bicep, use the last function.

Parameters

Parameter	Required	Type	Description
arg1	Yes	array or string	The value to retrieve the last element or character.

Return value

The type (string, int, array, or object) of the last element in an array, or the last character of a string.

Example

The following example shows how to use the last function with an array and string.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "arrayToTest": {
      "type": "array",
      "defaultValue": [ "one", "two", "three" ]
    }
  },
  "resources": [
  ],
  "outputs": {
    "arrayOutput": {
      "type": "string",
      "value": "[last(parameters('arrayToTest'))]"
    },
    "stringOutput": {
      "type": "string",
      "value": "[last('One Two Three')]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
arrayOutput	String	three
stringOutput	String	e

lastIndexOf

`lastIndexOf(arrayToSearch, itemToFind)`

Returns an integer for the index of the last occurrence of an item in an array. The comparison is **case-sensitive** for strings.

Parameters

Parameter	Required	Type	Description

Parameter	Required	Type	Description
arrayToSearch	Yes	array	The array to use for finding the index of the searched item.
itemToFind	Yes	int, string, array, or object	The item to find in the array.

Return value

An integer representing the last index of the item in the array. The index is zero-based. If the item isn't found, -1 is returned.

Examples

The following example shows how to use the indexOf and lastIndexOf functions:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "variables": {
    "names": [
      "one",
      "two",
      "three"
    ],
    "numbers": [
      4,
      5,
      6
    ],
    "collection": [
      "[variables('names')]",
      "[variables('numbers')]"
    ],
    "duplicates": [
      1,
      2,
      3,
      1
    ]
  },
  "resources": [],
  "outputs": {
    "index1": {
      "type": "int",
      "value": "[lastIndexOf(variables('names'), 'two')]"
```

```
    },
    "index2": {
      "type": "int",
      "value": "[indexOf(variables('names'), 'one')]"
    },
    "notFoundIndex1": {
      "type": "int",
      "value": "[lastIndexOf(variables('names'), 'Three')]"
    },
    "index3": {
      "type": "int",
      "value": "[lastIndexOf(variables('numbers'), 4)]"
    },
    "index4": {
      "type": "int",
      "value": "[indexOf(variables('numbers'), 6)]"
    },
    "notFoundIndex2": {
      "type": "int",
      "value": "[lastIndexOf(variables('numbers'), '5')]"
    },
    "index5": {
      "type": "int",
      "value": "[indexOf(variables('collection'), variables('numbers'))]"
    },
    "index6": {
      "type": "int",
      "value": "[indexOf(variables('duplicates'), 1)]"
    },
    "index7": {
      "type": "int",
      "value": "[lastIndexOf(variables('duplicates'), 1)]"
    }
  }
}
```

The output from the preceding example is:

Name	Type	Value
index1	int	1
index2	int	0
index3	int	0
index4	int	2
index5	int	1
index6	int	0
index7	int	3

Name	Type	Value
notFoundIndex1	int	-1
notFoundIndex2	int	-1

length

`length(arg1)`

Returns the number of elements in an array, characters in a string, or root-level properties in an object.

In Bicep, use the length function.

Parameters

Parameter	Required	Type	Description
arg1	Yes	array, string, or object	The array to use for getting the number of elements, the string to use for getting the number of characters, or the object to use for getting the number of root-level properties.

Return value

An int.

Example

The following example shows how to use length with an array and string.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "arrayToTest": {
      "type": "array",
      "defaultValue": [
        "one",
        "two",
        "three"
```

```
      ]
    },
    "stringToTest": {
      "type": "string",
      "defaultValue": "One Two Three"
    },
    "objectToTest": {
      "type": "object",
      "defaultValue": {
        "propA": "one",
        "propB": "two",
        "propC": "three",
        "propD": {
          "propD-1": "sub",
          "propD-2": "sub"
        }
      }
    }
  },
  "resources": [],
  "outputs": {
    "arrayLength": {
      "type": "int",
      "value": "[length(parameters('arrayToTest'))]"
    },
    "stringLength": {
      "type": "int",
      "value": "[length(parameters('stringToTest'))]"
    },
    "objectLength": {
      "type": "int",
      "value": "[length(parameters('objectToTest'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
arrayLength	Int	3
stringLength	Int	13
objectLength	Int	4

You can use this function with an array to specify the number of iterations when creating resources. In the following example, the parameter **siteNames** would refer to an array of names to use when creating the web sites.

JSON

```
"copy": {
  "name": "websitescopy",
  "count": "[length(parameters('siteNames'))]"
}
```

For more information about using this function with an array, see Resource iteration in ARM templates.

max

`max(arg1)`

Returns the maximum value from an array of integers or a comma-separated list of integers.

In Bicep, use the max function.

Parameters

Parameter	Required	Type	Description
arg1	Yes	array of integers, or comma-separated list of integers	The collection to get the maximum value.

Return value

An int representing the maximum value.

Example

The following example shows how to use max with an array and a list of integers.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "arrayToTest": {
      "type": "array",
      "defaultValue": [ 0, 3, 2, 5, 4 ]
    }
  },
  "resources": [],
```

```
  "outputs": {
    "arrayOutput": {
      "type": "int",
      "value": "[max(parameters('arrayToTest'))]"
    },
    "intOutput": {
      "type": "int",
      "value": "[max(0,3,2,5,4)]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
arrayOutput	Int	5
intOutput	Int	5

min

`min(arg1)`

Returns the minimum value from an array of integers or a comma-separated list of integers.

In Bicep, use the min function.

Parameters

Parameter	Required	Type	Description
arg1	Yes	array of integers, or comma-separated list of integers	The collection to get the minimum value.

Return value

An int representing the minimum value.

Example

The following example shows how to use min with an array and a list of integers.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "arrayToTest": {
      "type": "array",
      "defaultValue": [ 0, 3, 2, 5, 4 ]
    }
  },
  "resources": [],
  "outputs": {
    "arrayOutput": {
      "type": "int",
      "value": "[min(parameters('arrayToTest'))]"
    },
    "intOutput": {
      "type": "int",
      "value": "[min(0,3,2,5,4)]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
arrayOutput	Int	0
intOutput	Int	0

range

`range(startIndex, count)`

Creates an array of integers from a starting integer and containing a number of items.

In Bicep, use the range function.

Parameters

Parameter	Required	Type	Description
startIndex	Yes	int	The first integer in the array. The sum of startIndex and count must be no greater than 2147483647.

Parameter	Required	Type	Description
count	Yes	int	The number of integers in the array. Must be non-negative integer up to 10000.

Return value

An array of integers.

Example

The following example shows how to use the range function.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "startingInt": {
      "type": "int",
      "defaultValue": 5
    },
    "numberOfElements": {
      "type": "int",
      "defaultValue": 3
    }
  },
  "resources": [],
  "outputs": {
    "rangeOutput": {
      "type": "array",
      "value": "[range(parameters('startingInt'),parameters('numberOfElements'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
rangeOutput	Array	[5, 6, 7]

skip

`skip(originalValue, numberToSkip)`

Returns an array with all the elements after the specified number in the array, or returns a string with all the characters after the specified number in the string.

In Bicep, use the skip function.

Parameters

Parameter	Required	Type	Description
originalValue	Yes	array or string	The array or string to use for skipping.
numberToSkip	Yes	int	The number of elements or characters to skip. If this value is 0 or less, all the elements or characters in the value are returned. If it's larger than the length of the array or string, an empty array or string is returned.

Return value

An array or string.

Example

The following example skips the specified number of elements in the array, and the specified number of characters in a string.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "testArray": {
      "type": "array",
      "defaultValue": [
        "one",
        "two",
        "three"
      ]
    },
    "elementsToSkip": {
      "type": "int",
      "defaultValue": 2
```

```
    },
    "testString": {
      "type": "string",
      "defaultValue": "one two three"
    },
    "charactersToSkip": {
      "type": "int",
      "defaultValue": 4
    }
  },
  "resources": [],
  "outputs": {
    "arrayOutput": {
      "type": "array",
      "value": "
[skip(parameters('testArray'),parameters('elementsToSkip'))]"
    },
    "stringOutput": {
      "type": "string",
      "value": "
[skip(parameters('testString'),parameters('charactersToSkip'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
arrayOutput	Array	["three"]
stringOutput	String	two three

take

`take(originalValue, numberToTake)`

Returns an array or string. An array has the specified number of elements from the start of the array. A string has the specified number of characters from the start of the string.

In Bicep, use the take function.

Parameters

Parameter	Required	Type	Description

Parameter	Required	Type	Description
originalValue	Yes	array or string	The array or string to take the elements from.
numberToTake	Yes	int	The number of elements or characters to take. If this value is 0 or less, an empty array or string is returned. If it's larger than the length of the given array or string, all the elements in the array or string are returned.

Return value

An array or string.

Example

The following example takes the specified number of elements from the array, and characters from a string.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "testArray": {
      "type": "array",
      "defaultValue": [
        "one",
        "two",
        "three"
      ]
    },
    "elementsToTake": {
      "type": "int",
      "defaultValue": 2
    },
    "testString": {
      "type": "string",
      "defaultValue": "one two three"
    },
    "charactersToTake": {
      "type": "int",
      "defaultValue": 2
    }
  },
  "resources": [],
```

```
  "outputs": {
    "arrayOutput": {
      "type": "array",
      "value": "
[take(parameters('testArray'),parameters('elementsToTake'))]"
    },
    "stringOutput": {
      "type": "string",
      "value": "
[take(parameters('testString'),parameters('charactersToTake'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
arrayOutput	Array	["one", "two"]
stringOutput	String	on

union

`union(arg1, arg2, arg3, ...)`

Returns a single array or object with all elements from the parameters. For arrays, duplicate values are included once. For objects, duplicate property names are only included once.

In Bicep, use the union function.

Parameters

Parameter	Required	Type	Description
arg1	Yes	array or object	The first value to use for joining elements.
arg2	Yes	array or object	The second value to use for joining elements.
more arguments	No	array or object	More values to use for joining elements.

Return value

An array or object.

Remarks

The union function uses the sequence of the parameters to determine the order and values of the result.

For arrays, the function iterates through each element in the first parameter and adds it to the result if it isn't already present. Then, it repeats the process for the second parameter and any more parameters. If a value is already present, its earlier placement in the array is preserved.

For objects, property names and values from the first parameter are added to the result. For later parameters, any new names are added to the result. If a later parameter has a property with the same name, that value overwrites the existing value. The order of the properties isn't guaranteed.

Example

The following example shows how to use union with arrays and objects.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "firstObject": {
      "type": "object",
      "defaultValue": {
        "one": "a",
        "two": "b",
        "three": "c1"
      }
    },
    "secondObject": {
      "type": "object",
      "defaultValue": {
        "three": "c2",
        "four": "d",
        "five": "e"
      }
    },
    "firstArray": {
      "type": "array",
      "defaultValue": [ "one", "two", "three" ]
    },
    "secondArray": {
      "type": "array",
      "defaultValue": [ "three", "four" ]
```

```
    }
  },
  "resources": [
  ],
  "outputs": {
    "objectOutput": {
      "type": "object",
      "value": "[union(parameters('firstObject'),
parameters('secondObject'))]"
    },
    "arrayOutput": {
      "type": "array",
      "value": "[union(parameters('firstArray'),
parameters('secondArray'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
objectOutput	Object	{"one": "a", "two": "b", "three": "c2", "four": "d", "five": "e"}
arrayOutput	Array	["one", "two", "three", "four"]

Next steps

- For a description of the sections in an ARM template, see Understand the structure and syntax of ARM templates.

Comparison functions for ARM templates

Article • 04/09/2023

Resource Manager provides several functions for making comparisons in your Azure Resource Manager template (ARM template):

- coalesce
- equals
- greater
- greaterOrEquals
- less
- lessOrEquals

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see the **coalesce** logical operator and **comparison** operators.

coalesce

`coalesce(arg1, arg2, arg3, ...)`

Returns first non-null value from the parameters. Empty strings, empty arrays, and empty objects are not null.

In Bicep, use the `??` operator instead. See Coalesce ??.

Parameters

Parameter	Required	Type	Description
arg1	Yes	int, string, array, or object	The first value to test for null.
more args	No	int, string, array, or object	More values to test for null.

Return value

The value of the first non-null parameters, which can be a string, int, array, or object. Null if all parameters are null.

Example

The following example template shows the output from different uses of coalesce.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "objectToTest": {
      "type": "object",
      "defaultValue": {
        "null1": null,
        "null2": null,
        "string": "default",
        "int": 1,
        "object": { "first": "default" },
        "array": [ 1 ]
      }
    }
  },
  "resources": [
  ],
  "outputs": {
    "stringOutput": {
      "type": "string",
      "value": "[coalesce(parameters('objectToTest').null1, parameters('objectToTest').null2, parameters('objectToTest').string)]"
    },
    "intOutput": {
      "type": "int",
      "value": "[coalesce(parameters('objectToTest').null1, parameters('objectToTest').null2, parameters('objectToTest').int)]"
    },
    "objectOutput": {
      "type": "object",
      "value": "[coalesce(parameters('objectToTest').null1, parameters('objectToTest').null2, parameters('objectToTest').object)]"
    },
    "arrayOutput": {
      "type": "array",
      "value": "[coalesce(parameters('objectToTest').null1, parameters('objectToTest').null2, parameters('objectToTest').array)]"
    },
    "emptyOutput": {
      "type": "bool",
      "value": "[empty(coalesce(parameters('objectToTest').null1,
```

```
parameters('objectToTest').null2))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
stringOutput	String	default
intOutput	Int	1
objectOutput	Object	{"first": "default"}
arrayOutput	Array	[1]
emptyOutput	Bool	True

equals

`equals(arg1, arg2)`

Checks whether two values equal each other.

In Bicep, use the `==` operator instead. See Equals `==`.

Parameters

Parameter	Required	Type	Description
arg1	Yes	int, string, array, or object	The first value to check for equality.
arg2	Yes	int, string, array, or object	The second value to check for equality.

Return value

Returns **True** if the values are equal; otherwise, **False**.

Remarks

The equals function is often used with the `condition` element to test whether a resource is deployed.

JSON

```
{
  "condition": "[equals(parameters('newOrExisting'),'new')]",
  "type": "Microsoft.Storage/storageAccounts",
  "name": "[variables('storageAccountName')]",
  "apiVersion": "2017-06-01",
  "location": "[resourceGroup().location]",
  "sku": {
    "name": "[variables('storageAccountType')]"
  },
  "kind": "Storage",
  "properties": {}
}
```

Example

The following example checks different types of values for equality. All the default values return True.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "firstInt": {
      "type": "int",
      "defaultValue": 1
    },
    "secondInt": {
      "type": "int",
      "defaultValue": 1
    },
    "firstString": {
      "type": "string",
      "defaultValue": "a"
    },
    "secondString": {
      "type": "string",
      "defaultValue": "a"
    },
    "firstArray": {
      "type": "array",
      "defaultValue": [ "a", "b" ]
    },
    "secondArray": {
      "type": "array",
      "defaultValue": [ "a", "b" ]
    },
```

```
    "firstObject": {
      "type": "object",
      "defaultValue": { "a": "b" }
    },
    "secondObject": {
      "type": "object",
      "defaultValue": { "a": "b" }
    }
  },
  "resources": [
  ],
  "outputs": {
    "checkInts": {
      "type": "bool",
      "value": "[equals(parameters('firstInt'), parameters('secondInt') )]"
    },
    "checkStrings": {
      "type": "bool",
      "value": "[equals(parameters('firstString'),
parameters('secondString'))]"
    },
    "checkArrays": {
      "type": "bool",
      "value": "[equals(parameters('firstArray'),
parameters('secondArray'))]"
    },
    "checkObjects": {
      "type": "bool",
      "value": "[equals(parameters('firstObject'),
parameters('secondObject'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
checkInts	Bool	True
checkStrings	Bool	True
checkArrays	Bool	True
checkObjects	Bool	True

The following example template uses not with **equals**.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
```

```
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
  ],
  "outputs": {
    "checkNotEquals": {
      "type": "bool",
      "value": "[not(equals(1, 2))]"
    }
  }
}
```

The output from the preceding example is:

Name	Type	Value
checkNotEquals	Bool	True

greater

`greater(arg1, arg2)`

Checks whether the first value is greater than the second value.

In Bicep, use the `>` operator instead. See Greater than >.

Parameters

Parameter	Required	Type	Description
arg1	Yes	int or string	The first value for the greater comparison.
arg2	Yes	int or string	The second value for the greater comparison.

Return value

Returns **True** if the first value is greater than the second value; otherwise, **False**.

Example

The following example checks whether the one value is greater than the other.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "firstInt": {
      "type": "int",
      "defaultValue": 1
    },
    "secondInt": {
      "type": "int",
      "defaultValue": 2
    },
    "firstString": {
      "type": "string",
      "defaultValue": "A"
    },
    "secondString": {
      "type": "string",
      "defaultValue": "a"
    }
  },
  "resources": [
  ],
  "outputs": {
    "checkInts": {
      "type": "bool",
      "value": "[greater(parameters('firstInt'), parameters('secondInt') )]"
    },
    "checkStrings": {
      "type": "bool",
      "value": "[greater(parameters('firstString'), parameters('secondString'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
checkInts	Bool	False
checkStrings	Bool	True

greaterOrEquals

`greaterOrEquals(arg1, arg2)`

Checks whether the first value is greater than or equal to the second value.

In Bicep, use the `>=` operator instead. See Greater than or equal `>=`.

Parameters

Parameter	Required	Type	Description
arg1	Yes	int or string	The first value for the greater or equal comparison.
arg2	Yes	int or string	The second value for the greater or equal comparison.

Return value

Returns **True** if the first value is greater than or equal to the second value; otherwise, **False**.

Example

The following example checks whether the one value is greater than or equal to the other.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "firstInt": {
      "type": "int",
      "defaultValue": 1
    },
    "secondInt": {
      "type": "int",
      "defaultValue": 2
    },
    "firstString": {
      "type": "string",
      "defaultValue": "A"
    },
    "secondString": {
      "type": "string",
      "defaultValue": "a"
    }
  },
  "resources": [
  ],
  "outputs": {
    "checkInts": {
```

```
      "type": "bool",
      "value": "[greaterOrEquals(parameters('firstInt'),
parameters('secondInt') )]"
    },
    "checkStrings": {
      "type": "bool",
      "value": "[greaterOrEquals(parameters('firstString'),
parameters('secondString'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
checkInts	Bool	False
checkStrings	Bool	True

less

`less(arg1, arg2)`

Checks whether the first value is less than the second value.

In Bicep, use the `<` operator instead. See Less than <.

Parameters

Parameter	Required	Type	Description
arg1	Yes	int or string	The first value for the less comparison.
arg2	Yes	int or string	The second value for the less comparison.

Return value

Returns **True** if the first value is less than the second value; otherwise, **False**.

Example

The following example checks whether the one value is less than the other.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "firstInt": {
      "type": "int",
      "defaultValue": 1
    },
    "secondInt": {
      "type": "int",
      "defaultValue": 2
    },
    "firstString": {
      "type": "string",
      "defaultValue": "A"
    },
    "secondString": {
      "type": "string",
      "defaultValue": "a"
    }
  },
  "resources": [
  ],
  "outputs": {
    "checkInts": {
      "type": "bool",
      "value": "[less(parameters('firstInt'), parameters('secondInt') )]"
    },
    "checkStrings": {
      "type": "bool",
      "value": "[less(parameters('firstString'), parameters('secondString'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
checkInts	Bool	True
checkStrings	Bool	False

lessOrEquals

`lessOrEquals(arg1, arg2)`

Checks whether the first value is less than or equal to the second value.

In Bicep, use the `<=` operator instead. See Less than or equal <=.

Parameters

Parameter	Required	Type	Description
arg1	Yes	int or string	The first value for the less or equals comparison.
arg2	Yes	int or string	The second value for the less or equals comparison.

Return value

Returns **True** if the first value is less than or equal to the second value; otherwise, **False**.

Example

The following example checks whether the one value is less than or equal to the other.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "firstInt": {
      "type": "int",
      "defaultValue": 1
    },
    "secondInt": {
      "type": "int",
      "defaultValue": 2
    },
    "firstString": {
      "type": "string",
      "defaultValue": "A"
    },
    "secondString": {
      "type": "string",
      "defaultValue": "a"
    }
  },
  "resources": [
  ],
  "outputs": {
    "checkInts": {
      "type": "bool",
      "value": "[lessOrEquals(parameters('firstInt'),
parameters('secondInt') )]"
```

```
    },
    "checkStrings": {
      "type": "bool",
      "value": "[lessOrEquals(parameters('firstString'),
parameters('secondString'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
checkInts	Bool	True
checkStrings	Bool	False

Next steps

- For a description of the sections in an ARM template, see Understand the structure and syntax of ARM templates.

Date functions for ARM templates

Article • 04/09/2023

This article describes the functions for working with dates in your Azure Resource Manager template (ARM template).

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **date** functions.

dateTimeAdd

`dateTimeAdd(base, duration, [format])`

Adds a time duration to a base value. ISO 8601 format is expected.

In Bicep, use the dateTimeAdd function.

Parameters

Parameter	Required	Type	Description
base	Yes	string	The starting datetime value for the addition. Use ISO 8601 timestamp format.
duration	Yes	string	The time value to add to the base. It can be a negative value. Use ISO 8601 duration format.
format	No	string	The output format for the date time result. If not provided, the format of the base value is used. Use either standard format strings or custom format strings.

Return value

The datetime value that results from adding the duration value to the base value.

Examples

The following example template shows different ways of adding time values.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "baseTime": {
      "type": "string",
      "defaultValue": "[utcNow('u')]"
    }
  },
  "variables": {
    "add3Years": "[dateTimeAdd(parameters('baseTime'), 'P3Y')]",
    "subtract9Days": "[dateTimeAdd(parameters('baseTime'), '-P9D')]",
    "add1Hour": "[dateTimeAdd(parameters('baseTime'), 'PT1H')]"
  },
  "resources": [],
  "outputs": {
    "add3YearsOutput": {
      "value": "[variables('add3Years')]",
      "type": "string"
    },
    "subtract9DaysOutput": {
      "value": "[variables('subtract9Days')]",
      "type": "string"
    },
    "add1HourOutput": {
      "value": "[variables('add1Hour')]",
      "type": "string"
    }
  }
}
```

When the preceding template is deployed with a base time of `2020-04-07 14:53:14Z`, the output is:

Name	Type	Value
add3YearsOutput	String	4/7/2023 2:53:14 PM
subtract9DaysOutput	String	3/29/2020 2:53:14 PM
add1HourOutput	String	4/7/2020 3:53:14 PM

The next example template shows how to set the start time for an Automation schedule.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
```

```
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "omsAutomationAccountName": {
      "type": "string",
      "defaultValue": "demoAutomation",
      "metadata": {
        "description": "Use an existing Automation account."
      }
    },
    "scheduleName": {
      "type": "string",
      "defaultValue": "demoSchedule1",
      "metadata": {
        "description": "Name of the new schedule."
      }
    },
    "baseTime": {
      "type": "string",
      "defaultValue": "[utcNow('u')]",
      "metadata": {
        "description": "Schedule will start one hour from this time."
      }
    }
  },
  "variables": {
    "startTime": "[dateTimeAdd(parameters('baseTime'), 'PT1H')]"
  },
  "resources": [
    ...
    {
      "type": "Microsoft.Automation/automationAccounts/schedules",
      "apiVersion": "2020-01-13-preview",
      "name": "[concat(parameters('omsAutomationAccountName'), '/',
parameters('scheduleName'))]",

      "properties": {
        "description": "Demo Scheduler",
        "startTime": "[variables('startTime')]",
        "interval": 1,
        "frequency": "Hour"
      }
    }
  ],
  "outputs": {
  }
}
```

dateTimeFromEpoch

`dateTimeFromEpoch(epochTime)`

Converts an epoch time integer value to an ISO 8601 datetime.

In Bicep, use the dateTimeFromEpoch function.

Parameters

Parameter	Required	Type	Description
epochTime	Yes	int	The epoch time to convert to a datetime string.

Return value

An ISO 8601 datetime string.

Example

The following example shows output values for the epoch time functions.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "convertedEpoch": {
      "type": "int",
      "defaultValue": "[dateTimeToEpoch(dateTimeAdd(utcNow(), 'P1Y'))]"
    }
  },
  "variables": {
    "convertedDatetime": "[dateTimeFromEpoch(parameters('convertedEpoch'))]"
  },
  "resources": [],
  "outputs": {
    "epochValue": {
      "type": "int",
      "value": "[parameters('convertedEpoch')]"
    },
    "datetimeValue": {
      "type": "string",
      "value": "[variables('convertedDatetime')]"
    }
  }
}
```

The output is:

Name	Type	Value
datetimeValue	String	2023-05-02T15:16:13Z
epochValue	Int	1683040573

dateTimeToEpoch

`dateTimeToEpoch(dateTime)`

Converts an ISO 8601 datetime string to an epoch time integer value.

In Bicep, use the dateTimeToEpoch function.

Parameters

Parameter	Required	Type	Description
dateTime	Yes	string	The datetime string to convert to an epoch time.

Return value

An integer that represents the number of seconds from midnight on January 1, 1970.

Examples

The following example shows output values for the epoch time functions.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "convertedEpoch": {
      "type": "int",
      "defaultValue": "[dateTimeToEpoch(dateTimeAdd(utcNow(), 'P1Y'))]"
    }
  },
  "variables": {
    "convertedDatetime": "[dateTimeFromEpoch(parameters('convertedEpoch'))]"
  },
  "resources": [],
  "outputs": {
    "epochValue": {
```

```
      "type": "int",
      "value": "[parameters('convertedEpoch')]"
    },
    "datetimeValue": {
      "type": "string",
      "value": "[variables('convertedDatetime')]"
    }
  }
}
```

The output is:

Name	Type	Value
datetimeValue	String	2023-05-02T15:16:13Z
epochValue	Int	1683040573

The next example uses the epoch time value to set the expiration for a key in a key vault.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "16023511331197397029"
    }
  },
  "parameters": {
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "The location into which the resources should be
deployed."
      }
    },
    "tenantId": {
      "type": "string",
      "defaultValue": "[subscription().tenantId]",
      "metadata": {
        "description": "The Tenant Id that should be used throughout the
deployment."
      }
    },
    "userAssignedIdentityName": {
      "type": "string",
```

```
      "metadata": {
        "description": "The name of the existing User Assigned Identity."
      }
    },
    "userAssignedIdentityResourceGroupName": {
      "type": "string",
      "metadata": {
        "description": "The name of the resource group for the User Assigned
Identity."
      }
    },
    "keyVaultName": {
      "type": "string",
      "defaultValue": "[format('vault-{0}',
uniqueString(resourceGroup().id))]",
      "metadata": {
        "description": "The name of the Key Vault."
      }
    },
    "keyVaultKeyName": {
      "type": "string",
      "defaultValue": "cmkey",
      "metadata": {
        "description": "Name of the key in the Key Vault"
      }
    },
    "keyExpiration": {
      "type": "int",
      "defaultValue": "[dateTimeToEpoch(dateTimeAdd(utcNow(), 'P1Y'))]",
      "metadata": {
        "description": "Expiration time of the key"
      }
    },
    "storageAccountName": {
      "type": "string",
      "defaultValue": "[format('storage{0}',
uniqueString(resourceGroup().id))]",
      "metadata": {
        "description": "The name of the Storage Account"
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.KeyVault/vaults",
      "apiVersion": "2021-10-01",
      "name": "[parameters('keyVaultName')]",
      "location": "[parameters('location')]",
      "properties": {
        "sku": {
          "name": "standard",
          "family": "A"
        },
        "enableSoftDelete": true,
        "enablePurgeProtection": true,
```

```
        "enabledForDiskEncryption": true,
        "tenantId": "[parameters('tenantId')]",
        "accessPolicies": [
          {
            "tenantId": "[parameters('tenantId')]",
            "permissions": {
              "keys": [
                "unwrapKey",
                "wrapKey",
                "get"
              ]
            },
            "objectId": "
[reference(extensionResourceId(format('/subscriptions/{0}/resourceGroups/{1}
', subscription().subscriptionId,
parameters('userAssignedIdentityResourceGroupName')),
'Microsoft.ManagedIdentity/userAssignedIdentities',
parameters('userAssignedIdentityName')), '2018-11-30').principalId]"
          }
        ]
      }
    },
    {
      "type": "Microsoft.KeyVault/vaults/keys",
      "apiVersion": "2021-10-01",
      "name": "[format('{0}/{1}', parameters('keyVaultName'),
parameters('keyVaultKeyName'))]",
      "properties": {
        "attributes": {
          "enabled": true,
          "exp": "[parameters('keyExpiration')]"
        },
        "keySize": 4096,
        "kty": "RSA"
      },
      "dependsOn": [
        "[resourceId('Microsoft.KeyVault/vaults',
parameters('keyVaultName'))]"
      ]
    },
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-04-01",
      "name": "[parameters('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "StorageV2",
      "identity": {
        "type": "UserAssigned",
        "userAssignedIdentities": {
          "[format('{0}',
extensionResourceId(format('/subscriptions/{0}/resourceGroups/{1}',
subscription().subscriptionId,
```

```
parameters('userAssignedIdentityResourceGroupName')),
'Microsoft.ManagedIdentity/userAssignedIdentities',
parameters('userAssignedIdentityName')))]": {}
        }
      },
      "properties": {
        "accessTier": "Hot",
        "supportsHttpsTrafficOnly": true,
        "minimumTlsVersion": "TLS1_2",
        "encryption": {
          "identity": {
            "userAssignedIdentity": "
[extensionResourceId(format('/subscriptions/{0}/resourceGroups/{1}',
subscription().subscriptionId,
parameters('userAssignedIdentityResourceGroupName')),
'Microsoft.ManagedIdentity/userAssignedIdentities',
parameters('userAssignedIdentityName'))]"
          },
          "services": {
            "blob": {
              "enabled": true
            }
          },
          "keySource": "Microsoft.Keyvault",
          "keyvaultproperties": {
            "keyname": "[parameters('keyVaultKeyName')]",
            "keyvaulturi": "
[if(endsWith(reference(resourceId('Microsoft.KeyVault/vaults',
parameters('keyVaultName'))).vaultUri, '/'),
substring(reference(resourceId('Microsoft.KeyVault/vaults',
parameters('keyVaultName'))).vaultUri, 0,
sub(length(reference(resourceId('Microsoft.KeyVault/vaults',
parameters('keyVaultName'))).vaultUri), 1)),
reference(resourceId('Microsoft.KeyVault/vaults',
parameters('keyVaultName'))).vaultUri)]"
          }
        }
      },
      "dependsOn": [
        "[resourceId('Microsoft.KeyVault/vaults',
parameters('keyVaultName'))]",
        "[resourceId('Microsoft.KeyVault/vaults/keys',
parameters('keyVaultName'), parameters('keyVaultKeyName'))]"
      ]
    }
  ]
}
```

utcNow

`utcNow(format)`

Returns the current (UTC) datetime value in the specified format. If no format is provided, the ISO 8601 (`yyyyMMddTHHmmssZ`) format is used. **This function can only be used in the default value for a parameter.**

In Bicep, use the utcNow function.

Parameters

Parameter	Required	Type	Description
format	No	string	The URI encoded value to convert to a string. Use either standard format strings or custom format strings.

Remarks

You can only use this function within an expression for the default value of a parameter. Using this function anywhere else in a template returns an error. The function isn't allowed in other parts of the template because it returns a different value each time it's called. Deploying the same template with the same parameters wouldn't reliably produce the same results.

If you use the option to rollback on error to an earlier successful deployment, and the earlier deployment includes a parameter that uses `utcNow`, the parameter isn't reevaluated. Instead, the parameter value from the earlier deployment is automatically reused in the rollback deployment.

Be careful redeploying a template that relies on the `utcNow` function for a default value. When you redeploy and don't provide a value for the parameter, the function is reevaluated. If you want to update an existing resource rather than create a new one, pass in the parameter value from the earlier deployment.

Return value

The current UTC datetime value.

Examples

The following example template shows different formats for the datetime value.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "utcValue": {
      "type": "string",
      "defaultValue": "[utcNow()]"
    },
    "utcShortValue": {
      "type": "string",
      "defaultValue": "[utcNow('d')]"
    },
    "utcCustomValue": {
      "type": "string",
      "defaultValue": "[utcNow('M d')]"
    }
  },
  "resources": [
  ],
  "outputs": {
    "utcOutput": {
      "type": "string",
      "value": "[parameters('utcValue')]"
    },
    "utcShortOutput": {
      "type": "string",
      "value": "[parameters('utcShortValue')]"
    },
    "utcCustomOutput": {
      "type": "string",
      "value": "[parameters('utcCustomValue')]"
    }
  }
}
```

The output from the preceding example varies for each deployment but will be similar to:

Name	Type	Value
utcOutput	string	20190305T175318Z
utcShortOutput	string	03/05/2019
utcCustomOutput	string	3 5

The next example shows how to use a value from the function when setting a tag value.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2018-05-01/subscriptionDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "utcShort": {
      "type": "string",
      "defaultValue": "[utcNow('d')]"
    },
    "rgName": {
      "type": "string"
    }
  },
  "resources": [
    {
      "type": "Microsoft.Resources/resourceGroups",
      "apiVersion": "2021-04-01",
      "name": "[parameters('rgName')]",
      "location": "westeurope",
      "tags": {
        "createdDate": "[parameters('utcShort')]"
      },
      "properties": {}
    }
  ],
  "outputs": {
    "utcShortOutput": {
      "type": "string",
      "value": "[parameters('utcShort')]"
    }
  }
}
```

Next steps

- For a description of the sections in an ARM template, see Understand the structure and syntax of ARM templates.

Deployment functions for ARM templates

Article • 04/09/2023

Resource Manager provides the following functions for getting values related to the current deployment of your Azure Resource Manager template (ARM template):

- deployment
- environment
- parameters
- variables

To get values from resources, resource groups, or subscriptions, see Resource functions.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **deployment** functions.

deployment

`deployment()`

Returns information about the current deployment operation.

In Bicep, use the deployment function.

Return value

This function returns the object that is passed during deployment. The properties in the returned object differ based on whether you are:

- deploying a template or a template spec.
- deploying a template that is a local file or deploying a template that is a remote file accessed through a URI.
- deploying to a resource group or deploying to one of the other scopes (Azure subscription, management group, or tenant).

When deploying a local template to a resource group: the function returns the following format:

JSON

```
{
  "name": "",
  "properties": {
    "template": {
      "$schema": "",
      "contentVersion": "",
      "parameters": {},
      "variables": {},
      "resources": [],
      "outputs": {}
    },
    "templateHash": "",
    "parameters": {},
    "mode": "",
    "provisioningState": ""
  }
}
```

When deploying a remote template to a resource group: the function returns the following format:

JSON

```
{
  "name": "",
  "properties": {
    "templateLink": {
      "uri": ""
    },
    "template": {
      "$schema": "",
      "contentVersion": "",
      "parameters": {},
      "variables": {},
      "resources": [],
      "outputs": {}
    },
    "templateHash": "",
    "parameters": {},
    "mode": "",
    "provisioningState": ""
  }
}
```

When deploying a template spec to a resource group: the function returns the following format:

JSON

```
{
  "name": "",
  "properties": {
    "templateLink": {
      "id": ""
    },
    "template": {
      "$schema": "",
      "contentVersion": "",
      "parameters": {},
      "variables": {},
      "resources": [],
      "outputs": {}
    },
    "templateHash": "",
    "parameters": {},
    "mode": "",
    "provisioningState": ""
  }
}
```

When you deploy to an Azure subscription, management group, or tenant, the return object includes a `location` property. The location property is included when deploying either a local template or an external template. The format is:

JSON

```
{
  "name": "",
  "location": "",
  "properties": {
    "template": {
      "$schema": "",
      "contentVersion": "",
      "resources": [],
      "outputs": {}
    },
    "templateHash": "",
    "parameters": {},
    "mode": "",
    "provisioningState": ""
  }
}
```

Remarks

You can use `deployment()` to link to another template based on the URI of the parent template.

JSON

```
"variables": {
  "sharedTemplateUrl": "[uri(deployment().properties.templateLink.uri,
'shared-resources.json')]"
}
```

If you redeploy a template from the deployment history in the portal, the template is deployed as a local file. The `templateLink` property isn't returned in the deployment function. If your template relies on `templateLink` to construct a link to another template, don't use the portal to redeploy. Instead, use the commands you used to originally deploy the template.

Example

The following example returns a deployment object.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [],
  "outputs": {
    "deploymentOutput": {
      "type": "object",
      "value": "[deployment()]"
    }
  }
}
```

The preceding example returns the following object:

JSON

```
{
  "name": "deployment",
  "properties": {
    "template": {
      "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
      "contentVersion": "1.0.0.0",
      "resources": [],
      "outputs": {
        "deploymentOutput": {
          "type": "Object",
          "value": "[deployment()]"
```

```
        }
      }
    },
    "templateHash": "13135986259522608210",
    "parameters": {},
    "mode": "Incremental",
    "provisioningState": "Accepted"
  }
}
```

For a subscription deployment, the following example returns a deployment object.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2018-05-01/subscriptionDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {},
  "variables": {},
  "resources": [],
  "outputs": {
    "exampleOutput": {
      "type": "object",
      "value": "[deployment()]"
    }
  }
}
```

environment

`environment()`

Returns information about the Azure environment used for deployment.

In Bicep, use the environment function.

Remarks

To see a list of registered environments for your account, use az cloud list or Get-AzEnvironment.

Return value

This function returns properties for the current Azure environment. The following example shows the properties for global Azure. Sovereign clouds may return slightly

different properties.

JSON

```
{
  "name": "",
  "gallery": "",
  "graph": "",
  "portal": "",
  "graphAudience": "",
  "activeDirectoryDataLake": "",
  "batch": "",
  "media": "",
  "sqlManagement": "",
  "vmImageAliasDoc": "",
  "resourceManager": "",
  "authentication": {
    "loginEndpoint": "",
    "audiences": [
      "",
      ""
    ],
    "tenant": "",
    "identityProvider": ""
  },
  "suffixes": {
    "acrLoginServer": "",
    "azureDatalakeAnalyticsCatalogAndJob": "",
    "azureDatalakeStoreFileSystem": "",
    "azureFrontDoorEndpointSuffix": "",
    "keyvaultDns": "",
    "sqlServerHostname": "",
    "storage": ""
  }
}
```

Example

The following example template returns the environment object.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [],
  "outputs": {
    "environmentOutput": {
      "type": "object",
      "value": "[environment()]"
```

```
    }
  }
}
```

The preceding example returns the following object when deployed to global Azure:

JSON

```
{
  "name": "AzureCloud",
  "gallery": "https://gallery.azure.com/",
  "graph": "https://graph.windows.net/",
  "portal": "https://portal.azure.com",
  "graphAudience": "https://graph.windows.net/",
  "activeDirectoryDataLake": "https://datalake.azure.net/",
  "batch": "https://batch.core.windows.net/",
  "media": "https://rest.media.azure.net",
  "sqlManagement": "https://management.core.windows.net:8443/",
  "vmImageAliasDoc": "https://raw.githubusercontent.com/Azure/azure-rest-
api-specs/master/arm-compute/quickstart-templates/aliases.json",
  "resourceManager": "https://management.azure.com/",
  "authentication": {
    "loginEndpoint": "https://login.microsoftonline.com/",
    "audiences": [
      "https://management.core.windows.net/",
      "https://management.azure.com/"
    ],
    "tenant": "common",
    "identityProvider": "AAD"
  },
  "suffixes": {
    "acrLoginServer": ".azurecr.io",
    "azureDatalakeAnalyticsCatalogAndJob": "azuredatalakeanalytics.net",
    "azureDatalakeStoreFileSystem": "azuredatalakestore.net",
    "azureFrontDoorEndpointSuffix": "azurefd.net",
    "keyvaultDns": ".vault.azure.net",
    "sqlServerHostname": ".database.windows.net",
    "storage": "core.windows.net"
  }
}
```

parameters

`parameters(parameterName)`

Returns a parameter value. The specified parameter name must be defined in the parameters section of the template.

In Bicep, directly reference parameters by using their symbolic names.

Parameters

Parameter	Required	Type	Description
parameterName	Yes	string	The name of the parameter to return.

Return value

The value of the specified parameter.

Remarks

Typically, you use parameters to set resource values. The following example sets the name of web site to the parameter value passed in during deployment.

JSON

```
"parameters": {
  "siteName": {
    "type": "string"
  }
}, "resources": [
  {
    "type": "Microsoft.Web/Sites",
    "apiVersion": "2016-08-01",
    "name": "[parameters('siteName')]",
    ...
  }
]
```

Example

The following example shows a simplified use of the parameters function.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "stringParameter": {
      "type": "string",
      "defaultValue": "option 1"
    },
    "intParameter": {
```

```
        "type": "int",
        "defaultValue": 1
      },
      "objectParameter": {
        "type": "object",
        "defaultValue": {
          "one": "a",
          "two": "b"
        }
      },
      "arrayParameter": {
        "type": "array",
        "defaultValue": [ 1, 2, 3 ]
      },
      "crossParameter": {
        "type": "string",
        "defaultValue": "[parameters('stringParameter')]"
      }
    },
    "variables": {},
    "resources": [],
    "outputs": {
      "stringOutput": {
        "type": "string",
        "value": "[parameters('stringParameter')]"
      },
      "intOutput": {
        "type": "int",
        "value": "[parameters('intParameter')]"
      },
      "objectOutput": {
        "type": "object",
        "value": "[parameters('objectParameter')]"
      },
      "arrayOutput": {
        "type": "array",
        "value": "[parameters('arrayParameter')]"
      },
      "crossOutput": {
        "type": "string",
        "value": "[parameters('crossParameter')]"
      }
    }
  }
```

The output from the preceding example with the default values is:

Name	Type	Value
stringOutput	String	option 1
intOutput	Int	1

Name	Type	Value
objectOutput	Object	{"one": "a", "two": "b"}
arrayOutput	Array	[1, 2, 3]
crossOutput	String	option 1

For more information about using parameters, see Parameters in ARM templates.

variables

`variables(variableName)`

Returns the value of variable. The specified variable name must be defined in the variables section of the template.

In Bicep, directly reference variables by using their symbolic names.

Parameters

Parameter	Required	Type	Description
variableName	Yes	String	The name of the variable to return.

Return value

The value of the specified variable.

Remarks

Typically, you use variables to simplify your template by constructing complex values only once. The following example constructs a unique name for a storage account.

JSON

```
"variables": {
  "storageName": "[concat('storage', uniqueString(resourceGroup().id))]"
},
"resources": [
  {
    "type": "Microsoft.Storage/storageAccounts",
    "name": "[variables('storageName')]",
    ...
  },
```

```
  {
    "type": "Microsoft.Compute/virtualMachines",
    "dependsOn": [
      "[variables('storageName')]"
    ],
    ...
  }
],
```

Example

The following example returns different variable values.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {},
  "variables": {
    "var1": "myVariable",
    "var2": [ 1, 2, 3, 4 ],
    "var3": "[ variables('var1') ]",
    "var4": {
      "property1": "value1",
      "property2": "value2"
    }
  },
  "resources": [],
  "outputs": {
    "exampleOutput1": {
      "type": "string",
      "value": "[variables('var1')]"
    },
    "exampleOutput2": {
      "type": "array",
      "value": "[variables('var2')]"
    },
    "exampleOutput3": {
      "type": "string",
      "value": "[variables('var3')]"
    },
    "exampleOutput4": {
      "type": "object",
      "value": "[variables('var4')]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
exampleOutput1	String	myVariable
exampleOutput2	Array	[1, 2, 3, 4]
exampleOutput3	String	myVariable
exampleOutput4	Object	{"property1": "value1", "property2": "value2"}

For more information about using variables, see Variables in ARM template.

Next steps

- For a description of the sections in an ARM template, see Understand the structure and syntax of ARM templates.

Lambda functions for ARM templates

Article • 03/15/2023

This article describes the lambda functions to use in ARM templates. Lambda functions are essentially blocks of code that can be passed as an argument. They can take multiple parameters, but are restricted to a single line of code. In Bicep, lambda expression is in this format:

Bicep

```
lambda(<lambda variable>, [<lambda variable>, ...], <expression>)
```

Tip

We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **deployment** functions.

Limitations

ARM template lambda function has these limitations:

- Lambda function can only be specified directly as function arguments in these functions: filter(), map(), reduce(), sort(), and toObject().
- Using lambda variables (the temporary variables used in the lambda functions) inside resource or module array access isn't currently supported.
- Using lambda variables inside the listKeys function isn't currently supported.
- Using lambda variables inside the reference function isn't currently supported.

filter

`filter(inputArray, lambda function)`

Filters an array with a custom filtering function.

In Bicep, use the filter function.

Parameters

Parameter	Required	Type	Description

Parameter	Required	Type	Description
inputArray	Yes	array	The array to filter.
lambda function	Yes	expression	The lambda function applied to each input array element. If false, the item will be filtered out of the output array.

Return value

An array.

Examples

The following examples show how to use the `filter` function.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "variables": {
    "dogs": [
      {
        "name": "Evie",
        "age": 5,
        "interests": [
          "Ball",
          "Frisbee"
        ]
      },
      {
        "name": "Casper",
        "age": 3,
        "interests": [
          "Other dogs"
        ]
      },
      {
        "name": "Indy",
        "age": 2,
        "interests": [
          "Butter"
        ]
      },
      {
        "name": "Kira",
        "age": 8,
        "interests": [
          "Rubs"
```

```
        ]
      }
    ]
  },
  "resources": {},
  "outputs": {
    "oldDogs": {
      "type": "array",
      "value": "[filter(variables('dogs'), lambda('dog',
greaterOrEquals(lambdaVariables('dog').age, 5)))]"
    }
  }
}
```

The output from the preceding example shows the dogs that are five or older:

Name	Type	Value
oldDogs	Array	[{"name":"Evie","age":5,"interests":["Ball","Frisbee"]}, {"name":"Kira","age":8,"interests":["Rubs"]}]

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "variables": {
    "copy": [
      {
        "name": "itemForLoop",
        "count": "[length(range(0, 10))]",
        "input": "[range(0, 10)[copyIndex('itemForLoop')]]"
      }
    ]
  },
  "resources": {},
  "outputs": {
    "filteredLoop": {
      "type": "array",
      "value": "[filter(variables('itemForLoop'), lambda('i',
greater(lambdaVariables('i'), 5)))]"
    },
    "isEven": {
      "type": "array",
      "value": "[filter(range(0, 10), lambda('i', equals(0,
mod(lambdaVariables('i'), 2))))]"
    }
  }
}
```

The output from the preceding example:

Name	Type	Value
filteredLoop	Array	[6, 7, 8, 9]
isEven	Array	[0, 2, 4, 6, 8]

filterdLoop shows the numbers in an array that are greater than 5; and **isEven** shows the even numbers in the array.

map

`map(inputArray, lambda function)`

Applies a custom mapping function to each element of an array.

In Bicep, use the map function.

Parameters

Parameter	Required	Type	Description
inputArray	Yes	array	The array to map.
lambda function	Yes	expression	The lambda function applied to each input array element, in order to generate the output array.

Return value

An array.

Example

The following example shows how to use the `map` function.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "variables": {
    "dogs": [
      {
```

```
        "name": "Evie",
        "age": 5,
        "interests": [
          "Ball",
          "Frisbee"
        ]
      },
      {
        "name": "Casper",
        "age": 3,
        "interests": [
          "Other dogs"
        ]
      },
      {
        "name": "Indy",
        "age": 2,
        "interests": [
          "Butter"
        ]
      },
      {
        "name": "Kira",
        "age": 8,
        "interests": [
          "Rubs"
        ]
      }
    ]
  },
  "resources": {},
  "outputs": {
    "dogNames": {
      "type": "array",
      "value": "[map(variables('dogs'), lambda('dog',
lambdaVariables('dog').name))]"
    },
    "sayHi": {
      "type": "array",
      "value": "[map(variables('dogs'), lambda('dog', format('Hello {0}!',
lambdaVariables('dog').name)))]"
    },
    "mapObject": {
      "type": "array",
      "value": "[map(range(0, length(variables('dogs'))), lambda('i',
createObject('i', lambdaVariables('i'), 'dog', variables('dogs')
[lambdaVariables('i')].name, 'greeting', format('Ahoy, {0}!',
variables('dogs')[lambdaVariables('i')].name))))]"
    }
  }
}
```

The output from the preceding example is:

Name	Type	Value
dogNames	Array	["Evie","Casper","Indy","Kira"]
sayHi	Array	["Hello Evie!","Hello Casper!","Hello Indy!","Hello Kira!"]
mapObject	Array	[{"i":0,"dog":"Evie","greeting":"Ahoy, Evie!"}, {"i":1,"dog":"Casper","greeting":"Ahoy, Casper!"}, {"i":2,"dog":"Indy","greeting":"Ahoy, Indy!"}, {"i":3,"dog":"Kira","greeting":"Ahoy, Kira!"}]

dogNames shows the dog names from the array of objects; **sayHi** concatenates "Hello" and each of the dog names; and **mapObject** creates another array of objects.

reduce

`reduce(inputArray, initialValue, lambda function)`

Reduces an array with a custom reduce function.

In Bicep, use the reduce function.

Parameters

Parameter	Required	Type	Description
inputArray	Yes	array	The array to reduce.
initialValue	No	any	Initial value.
lambda function	Yes	expression	The lambda function used to aggregate the current value and the next value.

Return value

Any.

Example

The following examples show how to use the `reduce` function.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
```

```
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "variables": {
    "dogs": [
      {
        "name": "Evie",
        "age": 5,
        "interests": [
          "Ball",
          "Frisbee"
        ]
      },
      {
        "name": "Casper",
        "age": 3,
        "interests": [
          "Other dogs"
        ]
      },
      {
        "name": "Indy",
        "age": 2,
        "interests": [
          "Butter"
        ]
      },
      {
        "name": "Kira",
        "age": 8,
        "interests": [
          "Rubs"
        ]
      }
    ],
    "ages": "[map(variables('dogs'), lambda('dog',
lambdaVariables('dog').age))]"
  },
  "resources": {},
  "outputs": {
    "totalAge": {
      "type": "int",
      "value": "[reduce(variables('ages'), 0, lambda('cur', 'next',
add(lambdaVariables('cur'), lambdaVariables('next'))))]"
    },
    "totalAgeAdd1": {
      "type": "int",
      "value": "[reduce(variables('ages'), 1, lambda('cur', 'next',
add(lambdaVariables('cur'), lambdaVariables('next'))))]"
    }
  }
}
```

The output from the preceding example is:

Name	Type	Value
totalAge	int	18
totalAgeAdd1	int	19

totalAge sums the ages of the dogs; **totalAgeAdd1** has an initial value of 1, and adds all the dog ages to the initial values.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": {},
  "outputs": {
    "reduceObjectUnion": {
      "type": "object",
      "value": "[reduce(createArray(createObject('foo', 123), createObject('bar', 456), createObject('baz', 789)), createObject(), lambda('cur', 'next', union(lambdaVariables('cur'), lambdaVariables('next'))))]"
    }
  }
}
```

The output from the preceding example is:

Name	Type	Value
reduceObjectUnion	object	{"foo":123,"bar":456,"baz":789}

The union function returns a single object with all elements from the parameters. The function call unionizes the key value pairs of the objects into a new object.

sort

`sort(inputArray, lambda function)`

Sorts an array with a custom sort function.

In Bicep, use the sort function.

Parameters

Parameter	Required	Type	Description
inputArray	Yes	array	The array to sort.
lambda function	Yes	expression	The lambda function used to compare two array elements for ordering. If true, the second element will be ordered after the first in the output array.

Return value

An array.

Example

The following example shows how to use the `sort` function.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "variables": {
    "dogs": [
      {
        "name": "Evie",
        "age": 5,
        "interests": [
          "Ball",
          "Frisbee"
        ]
      },
      {
        "name": "Casper",
        "age": 3,
        "interests": [
          "Other dogs"
        ]
      },
      {
        "name": "Indy",
        "age": 2,
        "interests": [
          "Butter"
        ]
      },
      {
        "name": "Kira",
        "age": 8,
        "interests": [
```

```
            "Rubs"
          ]
        }
      ]
    },
    "resources": {},
    "outputs": {
      "dogsByAge": {
        "type": "array",
        "value": "[sort(variables('dogs'), lambda('a', 'b',
less(lambdaVariables('a').age, lambdaVariables('b').age)))]"
      }
    }
  }
```

The output from the preceding example sorts the dog objects from the youngest to the oldest:

Name	Type	Value
dogsByAge	Array	[{"name":"Indy","age":2,"interests":["Butter"]}, {"name":"Casper","age":3,"interests":["Other dogs"]}, {"name":"Evie","age":5,"interests":["Ball","Frisbee"]}, {"name":"Kira","age":8,"interests":["Rubs"]}]

toObject

`toObject(inputArray, lambda function, [lambda function])`

Converts an array to an object with a custom key function and optional custom value function. See items about converting an object to an array.

In Bicep, use the toObject function.

Parameters

Parameter	Required	Type	Description
inputArray	Yes	array	The array used for creating an object.
lambda function	Yes	expression	The lambda function used to provide the key predicate.
lambda function	No	expression	The lambda function used to provide the value predicate.

Return value

An object.

Example

The following example shows how to use the `toObject` function with the two required parameters:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "variables": {
    "dogs": [
      {
        "name": "Evie",
        "age": 5,
        "interests": [
          "Ball",
          "Frisbee"
        ]
      },
      {
        "name": "Casper",
        "age": 3,
        "interests": [
          "Other dogs"
        ]
      },
      {
        "name": "Indy",
        "age": 2,
        "interests": [
          "Butter"
        ]
      },
      {
        "name": "Kira",
        "age": 8,
        "interests": [
          "Rubs"
        ]
      }
    ]
  },
  "resources": {},
  "outputs": {
    "dogsObject": {
```

```
      "type": "object",
      "value": "[toObject(variables('dogs'), lambda('entry',
lambdaVariables('entry').name))]"
    }
  }
}
```

The preceding example generates an object based on an array.

Name	Type	Value
dogsObject	Object	{"Evie":{"name":"Evie","age":5,"interests":["Ball","Frisbee"]},"Casper": {"name":"Casper","age":3,"interests":["Other dogs"]},"Indy": {"name":"Indy","age":2,"interests":["Butter"]},"Kira": {"name":"Kira","age":8,"interests":["Rubs"]}}

The following `toObject` function with the third parameter provides the same output.

JSON

```
"outputs": {
  "dogsObject": {
    "type": "object",
    "value": "[toObject(variables('dogs'), lambda('entry',
lambdaVariables('entry').name), lambda('entry', lambdaVariables('entry')))]"
  }
}
```

The following example shows how to use the `toObject` function with three parameters.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "variables": {
    "dogs": [
      {
        "name": "Evie",
        "properties": {
          "age": 5,
          "interests": [
            "Ball",
            "Frisbee"
          ]
        }
      },
      {
        "name": "Casper",
```

```
        "properties": {
          "age": 3,
          "interests": [
            "Other dogs"
          ]
        }
      },
      {
        "name": "Indy",
        "properties": {
          "age": 2,
          "interests": [
            "Butter"
          ]
        }
      },
      {
        "name": "Kira",
        "properties": {
          "age": 8,
          "interests": [
            "Rubs"
          ]
        }
      }
    ]
  },
  "resources": {},
  "outputs": {
    "dogsObject": {
      "type": "object",
      "value": "[toObject(variables('dogs'), lambda('entry',
lambdaVariables('entry').name), lambda('entry',
lambdaVariables('entry').properties))]"
    }
  }
}
```

The preceding example generates an object based on an array.

Name	Type	Value
dogsObject	Object	{"Evie":{"age":5,"interests":["Ball","Frisbee"]},"Casper":{"age":3,"interests":["Other dogs"]},"Indy":{"age":2,"interests":["Butter"]},"Kira":{"age":8,"interests":["Rubs"]}}

Next steps

- See Template functions - arrays for additional array related template functions.

Logical functions for ARM templates

Article • 04/09/2023

Resource Manager provides several functions for making comparisons in your Azure Resource Manager template (ARM template):

- and
- bool
- false
- if
- not
- or
- true

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see the **bool** logical function and **logical** operators.

and

`and(arg1, arg2, ...)`

Checks whether all parameter values are true.

The `and` function isn't supported in Bicep. Use the && operator instead.

Parameters

Parameter	Required	Type	Description
arg1	Yes	boolean	The first value to check whether is true.
arg2	Yes	boolean	The second value to check whether is true.
more arguments	No	boolean	More arguments to check whether are true.

Return value

Returns **True** if all values are true; otherwise, **False**.

Examples

The following example shows how to use logical functions.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [],
  "outputs": {
    "andExampleOutput": {
      "type": "bool",
      "value": "[and(bool('true'), bool('false'))]"
    },
    "orExampleOutput": {
      "type": "bool",
      "value": "[or(bool('true'), bool('false'))]"
    },
    "notExampleOutput": {
      "type": "bool",
      "value": "[not(bool('true'))]"
    }
  }
}
```

The output from the preceding example is:

Name	Type	Value
andExampleOutput	Bool	False
orExampleOutput	Bool	True
notExampleOutput	Bool	False

bool

`bool(arg1)`

Converts the parameter to a boolean.

In Bicep, use the bool logical function.

Parameters

Parameter	Required	Type	Description
arg1	Yes	string or int	The value to convert to a boolean.

Return value

A boolean of the converted value.

Remarks

You can also use true() and false() to get boolean values.

Examples

The following example shows how to use bool with a string or integer.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [],
  "outputs": {
    "trueString": {
      "type": "bool",
      "value": "[bool('true')]"
    },
    "falseString": {
      "type": "bool",
      "value": "[bool('false')]"
    },
    "trueInt": {
      "type": "bool",
      "value": "[bool(1)]"
    },
    "falseInt": {
      "type": "bool",
      "value": "[bool(0)]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
trueString	Bool	True
falseString	Bool	False
trueInt	Bool	True
falseInt	Bool	False

false

`false()`

Returns false.

The `false` function isn't available in Bicep. Use the `false` keyword instead.

Parameters

The false function doesn't accept any parameters.

Return value

A boolean that is always false.

Example

The following example returns a false output value.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [],
  "outputs": {
    "falseOutput": {
      "type": "bool",
      "value": "[false()]"
    }
  }
}
```

The output from the preceding example is:

Name	Type	Value
falseOutput	Bool	False

if

`if(condition, trueValue, falseValue)`

Returns a value based on whether a condition is true or false.

The `if` function isn't supported in Bicep. Use the ?: operator instead.

Parameters

Parameter	Required	Type	Description
condition	Yes	boolean	The value to check whether it's true or false.
trueValue	Yes	string, int, object, or array	The value to return when the condition is true.
falseValue	Yes	string, int, object, or array	The value to return when the condition is false.

Return value

Returns second parameter when first parameter is **True**; otherwise, returns third parameter.

Remarks

When the condition is **True**, only the true value is evaluated. When the condition is **False**, only the false value is evaluated. With the `if` function, you can include expressions that are only conditionally valid. For example, you can reference a resource that exists under one condition but not under the other condition. An example of conditionally evaluating expressions is shown in the following section.

Examples

The following example shows how to use the `if` function.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
  ],
  "outputs": {
    "yesOutput": {
      "type": "string",
      "value": "[if(equals('a', 'a'), 'yes', 'no')]"
    },
    "noOutput": {
      "type": "string",
      "value": "[if(equals('a', 'b'), 'yes', 'no')]"
    },
    "objectOutput": {
      "type": "object",
      "value": "[if(equals('a', 'a'), json('{\"test\": \"value1\"}'), json('null'))]"
    }
  }
}
```

The output from the preceding example is:

Name	Type	Value
yesOutput	String	yes
noOutput	String	no
objectOutput	Object	{ "test": "value1" }

The following example template shows how to use this function with expressions that are only conditionally valid.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "vmName": {
      "type": "string"
    },
    "location": {
      "type": "string"
    },
```

```
    "logAnalytics": {
      "type": "string",
      "defaultValue": ""
    }
  },
  "resources": [
    {
      "condition": "[not(empty(parameters('logAnalytics')))]",
      "type": "Microsoft.Compute/virtualMachines/extensions",
      "apiVersion": "2017-03-30",
      "name": "[concat(parameters('vmName'),'/omsOnboarding')]",
      "location": "[parameters('location')]",
      "properties": {
        "publisher": "Microsoft.EnterpriseCloud.Monitoring",
        "type": "MicrosoftMonitoringAgent",
        "typeHandlerVersion": "1.0",
        "autoUpgradeMinorVersion": true,
        "settings": {
          "workspaceId": "[if(not(empty(parameters('logAnalytics'))),
reference(parameters('logAnalytics'), '2015-11-01-preview').customerId,
json('null'))]"
        },
        "protectedSettings": {
          "workspaceKey": "[if(not(empty(parameters('logAnalytics'))),
listKeys(parameters('logAnalytics'), '2015-11-01-preview').primarySharedKey,
json('null'))]"
        }
      }
    }
  ],
  "outputs": {
    "mgmtStatus": {
      "type": "string",
      "value": "[if(not(empty(parameters('logAnalytics'))), 'Enabled
monitoring for VM!', 'Nothing to enable')]"
    }
  }
}
```

not

`not(arg1)`

Converts boolean value to its opposite value.

The `not` function isn't supported in Bicep. Use the ! operator instead.

Parameters

Parameter	Required	Type	Description
arg1	Yes	boolean	The value to convert.

Return value

Returns **True** when parameter is **False**. Returns **False** when parameter is **True**.

Examples

The following example shows how to use logical functions.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [],
  "outputs": {
    "andExampleOutput": {
      "type": "bool",
      "value": "[and(bool('true'), bool('false'))]"
    },
    "orExampleOutput": {
      "type": "bool",
      "value": "[or(bool('true'), bool('false'))]"
    },
    "notExampleOutput": {
      "type": "bool",
      "value": "[not(bool('true'))]"
    }
  }
}
```

The output from the preceding example is:

Name	Type	Value
andExampleOutput	Bool	False
orExampleOutput	Bool	True
notExampleOutput	Bool	False

The following example uses `not` with equals.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
  ],
  "outputs": {
    "checkNotEquals": {
      "type": "bool",
      "value": "[not(equals(1, 2))]"
    }
  }
}
```

The output from the preceding example is:

Name	Type	Value
checkNotEquals	Bool	True

or

`or(arg1, arg2, ...)`

Checks whether any parameter value is true.

The `or` function isn't supported in Bicep. Use the || operator instead.

Parameters

Parameter	Required	Type	Description
arg1	Yes	boolean	The first value to check whether is true.
arg2	Yes	boolean	The second value to check whether is true.
more arguments	No	boolean	More arguments to check whether are true.

Return value

Returns **True** if any value is true; otherwise, **False**.

Examples

The following example shows how to use logical functions.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [],
  "outputs": {
    "andExampleOutput": {
      "type": "bool",
      "value": "[and(bool('true'), bool('false'))]"
    },
    "orExampleOutput": {
      "type": "bool",
      "value": "[or(bool('true'), bool('false'))]"
    },
    "notExampleOutput": {
      "type": "bool",
      "value": "[not(bool('true'))]"
    }
  }
}
```

The output from the preceding example is:

Name	Type	Value
andExampleOutput	Bool	False
orExampleOutput	Bool	True
notExampleOutput	Bool	False

true

`true()`

Returns true.

The `true` function isn't available in Bicep. Use the `true` keyword instead.

Parameters

The true function doesn't accept any parameters.

Return value

A boolean that is always true.

Example

The following example returns a true output value.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [],
  "outputs": {
    "trueOutput": {
      "type": "bool",
      "value": "[true()]"
    }
  }
}
```

The output from the preceding example is:

Name	Type	Value
trueOutput	Bool	True

Next steps

- For a description of the sections in an ARM template, see Understand the structure and syntax of ARM templates.

Numeric functions for ARM templates

Article • 04/09/2023

Resource Manager provides the following functions for working with integers in your Azure Resource Manager template (ARM template):

- add
- copyIndex
- div
- float
- int
- max
- min
- mod
- mul
- sub

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more about using `int`, `min`, and `max` in Bicep, see **numeric** functions. For other numeric values, see **numeric** operators.

add

`add(operand1, operand2)`

Returns the sum of the two provided integers.

The `add` function isn't supported in Bicep. Use the + operator instead.

Parameters

Parameter	Required	Type	Description
operand1	Yes	int	First number to add.
operand2	Yes	int	Second number to add.

Return value

An integer that contains the sum of the parameters.

Example

The following example adds two parameters.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "first": {
      "type": "int",
      "defaultValue": 5,
      "metadata": {
        "description": "First integer to add"
      }
    },
    "second": {
      "type": "int",
      "defaultValue": 3,
      "metadata": {
        "description": "Second integer to add"
      }
    }
  },
  "resources": [
  ],
  "outputs": {
    "addResult": {
      "type": "int",
      "value": "[add(parameters('first'), parameters('second'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
addResult	Int	8

copyIndex

`copyIndex(loopName, offset)`

Returns the index of an iteration loop.

In Bicep, use iterative loops.

Parameters

Parameter	Required	Type	Description
loopName	No	string	The name of the loop for getting the iteration.
offset	No	int	The number to add to the zero-based iteration value.

Remarks

This function is always used with a **copy** object. If no value is provided for **offset**, the current iteration value is returned. The iteration value starts at zero.

The **loopName** property enables you to specify whether copyIndex is referring to a resource iteration or property iteration. If no value is provided for **loopName**, the current resource type iteration is used. Provide a value for **loopName** when iterating on a property.

For more information about using copy, see:

- Resource iteration in ARM templates
- Property iteration in ARM templates
- Variable iteration in ARM templates
- Output iteration in ARM templates

Example

The following example shows a copy loop and the index value included in the name.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageCount": {
      "type": "int",
      "defaultValue": 2
    }
  },
  "resources": [
```

```
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-04-01",
      "name": "[concat(copyIndex(),'storage',
uniqueString(resourceGroup().id))]",
      "location": "[resourceGroup().location]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "Storage",
      "properties": {},
      "copy": {
        "name": "storagecopy",
        "count": "[parameters('storageCount')]"
      }
    }
  ],
  "outputs": {}
}
```

Return value

An integer representing the current index of the iteration.

div

`div(operand1, operand2)`

Returns the integer division of the two provided integers.

The `div` function isn't supported in Bicep. Use the / operator instead.

Parameters

Parameter	Required	Type	Description
operand1	Yes	int	The number being divided.
operand2	Yes	int	The number that is used to divide. Can't be 0.

Return value

An integer representing the division.

Example

The following example divides one parameter by another parameter.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "first": {
      "type": "int",
      "defaultValue": 8,
      "metadata": {
        "description": "Integer being divided"
      }
    },
    "second": {
      "type": "int",
      "defaultValue": 3,
      "metadata": {
        "description": "Integer used to divide"
      }
    }
  },
  "resources": [
  ],
  "outputs": {
    "divResult": {
      "type": "int",
      "value": "[div(parameters('first'), parameters('second'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
divResult	Int	2

float

`float(arg1)`

Converts the value to a floating point number. You only use this function when passing custom parameters to an application, such as a Logic App.

The `float` function isn't supported in Bicep.

Parameters

Parameter	Required	Type	Description
arg1	Yes	string or int	The value to convert to a floating point number.

Return value

A floating point number.

Example

The following example shows how to use float to pass parameters to a Logic App:

JSON

```
{
  "type": "Microsoft.Logic/workflows",
  "properties": {
    ...
    "parameters": {
      "custom1": {
        "value": "[float('3.0')]"
      },
      "custom2": {
        "value": "[float(3)]"
      },
```

int

`int(valueToConvert)`

Converts the specified value to an integer.

In Bicep, use the int function.

Parameters

Parameter	Required	Type	Description
valueToConvert	Yes	string or int	The value to convert to an integer.

Return value

An integer of the converted value.

Example

The following example template converts the user-provided parameter value to integer.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "stringToConvert": {
      "type": "string",
      "defaultValue": "4"
    }
  },
  "resources": [
  ],
  "outputs": {
    "intResult": {
      "type": "int",
      "value": "[int(parameters('stringToConvert'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
intResult	Int	4

max

`max(arg1)`

Returns the maximum value from an array of integers or a comma-separated list of integers.

In Bicep, use the max function.

Parameters

Parameter	Required	Type	Description

Parameter	Required	Type	Description
arg1	Yes	array of integers, or comma-separated list of integers	The collection to get the maximum value.

Return value

An integer representing the maximum value from the collection.

Example

The following example shows how to use max with an array and a list of integers.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "arrayToTest": {
      "type": "array",
      "defaultValue": [ 0, 3, 2, 5, 4 ]
    }
  },
  "resources": [],
  "outputs": {
    "arrayOutput": {
      "type": "int",
      "value": "[max(parameters('arrayToTest'))]"
    },
    "intOutput": {
      "type": "int",
      "value": "[max(0,3,2,5,4)]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
arrayOutput	Int	5
intOutput	Int	5

min

`min(arg1)`

Returns the minimum value from an array of integers or a comma-separated list of integers.

In Bicep, use the min function.

Parameters

Parameter	Required	Type	Description
arg1	Yes	array of integers, or comma-separated list of integers	The collection to get the minimum value.

Return value

An integer representing minimum value from the collection.

Example

The following example shows how to use min with an array and a list of integers.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "arrayToTest": {
      "type": "array",
      "defaultValue": [ 0, 3, 2, 5, 4 ]
    }
  },
  "resources": [],
  "outputs": {
    "arrayOutput": {
      "type": "int",
      "value": "[min(parameters('arrayToTest'))]"
    },
    "intOutput": {
      "type": "int",
      "value": "[min(0,3,2,5,4)]"
    }
```

```
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
arrayOutput	Int	0
intOutput	Int	0

mod

`mod(operand1, operand2)`

Returns the remainder of the integer division using the two provided integers.

The `mod` function isn't supported in Bicep. Use the % operator instead.

Parameters

Parameter	Required	Type	Description
operand1	Yes	int	The number being divided.
operand2	Yes	int	The number that is used to divide, Can't be 0.

Return value

An integer representing the remainder.

Example

The following example returns the remainder of dividing one parameter by another parameter.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "first": {
```

```
        "type": "int",
        "defaultValue": 7,
        "metadata": {
          "description": "Integer being divided"
        }
      },
      "second": {
        "type": "int",
        "defaultValue": 3,
        "metadata": {
          "description": "Integer used to divide"
        }
      }
    },
    "resources": [
    ],
    "outputs": {
      "modResult": {
        "type": "int",
        "value": "[mod(parameters('first'), parameters('second'))]"
      }
    }
  }
```

The output from the preceding example with the default values is:

Name	Type	Value
modResult	Int	1

mul

`mul(operand1, operand2)`

Returns the multiplication of the two provided integers.

The `mul` function isn't supported in Bicep. Use the * operator instead.

Parameters

Parameter	Required	Type	Description
operand1	Yes	int	First number to multiply.
operand2	Yes	int	Second number to multiply.

Return value

An integer representing the multiplication.

Example

The following example multiplies one parameter by another parameter.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "first": {
      "type": "int",
      "defaultValue": 5,
      "metadata": {
        "description": "First integer to multiply"
      }
    },
    "second": {
      "type": "int",
      "defaultValue": 3,
      "metadata": {
        "description": "Second integer to multiply"
      }
    }
  },
  "resources": [
  ],
  "outputs": {
    "mulResult": {
      "type": "int",
      "value": "[mul(parameters('first'), parameters('second'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
mulResult	Int	15

sub

`sub(operand1, operand2)`

Returns the subtraction of the two provided integers.

The `sub` function isn't supported in Bicep. Use the - operator instead.

Parameters

Parameter	Required	Type	Description
operand1	Yes	int	The number that is subtracted from.
operand2	Yes	int	The number that is subtracted.

Return value

An integer representing the subtraction.

Example

The following example subtracts one parameter from another parameter.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "first": {
      "type": "int",
      "defaultValue": 7,
      "metadata": {
        "description": "Integer subtracted from"
      }
    },
    "second": {
      "type": "int",
      "defaultValue": 3,
      "metadata": {
        "description": "Integer to subtract"
      }
    }
  },
  "resources": [
  ],
  "outputs": {
    "subResult": {
      "type": "int",
      "value": "[sub(parameters('first'), parameters('second'))]"
```

```
      }
    }
  }
```

The output from the preceding example with the default values is:

Name	Type	Value
subResult	Int	4

Next steps

- For a description of the sections in an ARM template, see Understand the structure and syntax of ARM templates.
- To iterate a specified number of times when creating a type of resource, see Resource iteration in ARM templates.

Object functions for ARM templates

Article • 03/15/2023

Resource Manager provides several functions for working with objects in your Azure Resource Manager template (ARM template):

- contains
- createObject
- empty
- intersection
- items
- json
- length
- null
- union

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **object** functions.

contains

`contains(container, itemToFind)`

Checks whether an array contains a value, an object contains a key, or a string contains a substring. The string comparison is case-sensitive. However, when testing if an object contains a key, the comparison is case-insensitive.

In Bicep, use the contains function.

Parameters

Parameter	Required	Type	Description
container	Yes	array, object, or string	The value that contains the value to find.
itemToFind	Yes	string or int	The value to find.

Return value

True if the item is found; otherwise, **False**.

Example

The following example shows how to use contains with different types:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "stringToTest": {
      "type": "string",
      "defaultValue": "OneTwoThree"
    },
    "objectToTest": {
      "type": "object",
      "defaultValue": {
        "one": "a",
        "two": "b",
        "three": "c"
      }
    },
    "arrayToTest": {
      "type": "array",
      "defaultValue": [ "one", "two", "three" ]
    }
  },
  "resources": [
  ],
  "outputs": {
    "stringTrue": {
      "type": "bool",
      "value": "[contains(parameters('stringToTest'), 'e')]"
    },
    "stringFalse": {
      "type": "bool",
      "value": "[contains(parameters('stringToTest'), 'z')]"
    },
    "objectTrue": {
      "type": "bool",
      "value": "[contains(parameters('objectToTest'), 'one')]"
    },
    "objectFalse": {
      "type": "bool",
      "value": "[contains(parameters('objectToTest'), 'a')]"
    },
    "arrayTrue": {
      "type": "bool",
      "value": "[contains(parameters('arrayToTest'), 'three')]"
```

```
    },
    "arrayFalse": {
      "type": "bool",
      "value": "[contains(parameters('arrayToTest'), 'four')]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
stringTrue	Bool	True
stringFalse	Bool	False
objectTrue	Bool	True
objectFalse	Bool	False
arrayTrue	Bool	True
arrayFalse	Bool	False

createObject

`createObject(key1, value1, key2, value2, ...)`

Creates an object from the keys and values.

The `createObject` function isn't supported by Bicep. Construct an object by using `{}`. See Objects.

Parameters

Parameter	Required	Type	Description
key1	No	string	The name of the key.
value1	No	int, boolean, string, object, or array	The value for the key.
more keys	No	string	More names of the keys.
more values	No	int, boolean, string, object, or array	More values for the keys.

The function only accepts an even number of parameters. Each key must have a matching value.

Return value

An object with each key and value pair.

Example

The following example creates an object from different types of values.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
  ],
  "outputs": {
    "newObject": {
      "type": "object",
      "value": "[createObject('intProp', 1, 'stringProp', 'abc', 'boolProp', true(), 'arrayProp', createArray('a', 'b', 'c'), 'objectProp', createObject('key1', 'value1'))]"
    }
  }
}
```

The output from the preceding example with the default values is an object named `newObject` with the following value:

JSON

```
{
  "intProp": 1,
  "stringProp": "abc",
  "boolProp": true,
  "arrayProp": ["a", "b", "c"],
  "objectProp": {"key1": "value1"}
}
```

empty

`empty(itemToTest)`

Determines if an array, object, or string is empty.

In Bicep, use the empty function.

Parameters

Parameter	Required	Type	Description
itemToTest	Yes	array, object, or string	The value to check if it's empty.

Return value

Returns **True** if the value is empty; otherwise, **False**.

Example

The following example checks whether an array, object, and string are empty.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "testArray": {
      "type": "array",
      "defaultValue": []
    },
    "testObject": {
      "type": "object",
      "defaultValue": {}
    },
    "testString": {
      "type": "string",
      "defaultValue": ""
    }
  },
  "resources": [
  ],
  "outputs": {
    "arrayEmpty": {
      "type": "bool",
      "value": "[empty(parameters('testArray'))]"
    },
    "objectEmpty": {
      "type": "bool",
      "value": "[empty(parameters('testObject'))]"
    },
    "stringEmpty": {
      "type": "bool",
      "value": "[empty(parameters('testString'))]"
    }
```

```
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
arrayEmpty	Bool	True
objectEmpty	Bool	True
stringEmpty	Bool	True

intersection

`intersection(arg1, arg2, arg3, ...)`

Returns a single array or object with the common elements from the parameters.

In Bicep, use the intersection function.

Parameters

Parameter	Required	Type	Description
arg1	Yes	array or object	The first value to use for finding common elements.
arg2	Yes	array or object	The second value to use for finding common elements.
more arguments	No	array or object	More values to use for finding common elements.

Return value

An array or object with the common elements.

Example

The following example shows how to use intersection with arrays and objects.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "firstObject": {
      "type": "object",
      "defaultValue": {
        "one": "a",
        "two": "b",
        "three": "c"
      }
    },
    "secondObject": {
      "type": "object",
      "defaultValue": {
        "one": "a",
        "two": "z",
        "three": "c"
      }
    },
    "firstArray": {
      "type": "array",
      "defaultValue": [ "one", "two", "three" ]
    },
    "secondArray": {
      "type": "array",
      "defaultValue": [ "two", "three" ]
    }
  },
  "resources": [
  ],
  "outputs": {
    "objectOutput": {
      "type": "object",
      "value": "[intersection(parameters('firstObject'), parameters('secondObject'))]"
    },
    "arrayOutput": {
      "type": "array",
      "value": "[intersection(parameters('firstArray'), parameters('secondArray'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
objectOutput	Object	{"one": "a", "three": "c"}

Name	Type	Value
arrayOutput	Array	["two", "three"]

items

`items(object)`

Converts a dictionary object to an array. See toObject about converting an array to an object.

In Bicep, use the items.

Parameters

Parameter	Required	Type	Description
object	Yes	object	The dictionary object to convert to an array.

Return value

An array of objects for the converted dictionary. Each object in the array has a `key` property that contains the key value for the dictionary. Each object also has a `value` property that contains the properties for the object.

Example

The following example converts a dictionary object to an array. For each object in the array, it creates a new object with modified values.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "variables": {
    "copy": [
      {
        "name": "modifiedListOfEntities",
        "count": "[length(items(variables('entities')))]",
        "input": {
          "key": "[items(variables('entities'))[copyIndex('modifiedListOfEntities')].key]",
```

```
          "fullName": "[items(variables('entities'))
[copyIndex('modifiedListOfEntities')].value.displayName]",
          "itemEnabled": "[items(variables('entities'))
[copyIndex('modifiedListOfEntities')].value.enabled]"
        }
      }
    ],
    "entities": {
      "item002": {
        "enabled": false,
        "displayName": "Example item 2",
        "number": 200
      },
      "item001": {
        "enabled": true,
        "displayName": "Example item 1",
        "number": 300
      }
    }
  },
  "resources": [],
  "outputs": {
    "modifiedResult": {
      "type": "array",
      "value": "[variables('modifiedListOfEntities')]"
    }
  }
}
```

The preceding example returns:

JSON

```
"modifiedResult": {
  "type": "Array",
  "value": [
    {
      "fullName": "Example item 1",
      "itemEnabled": true,
      "key": "item001"
    },
    {
      "fullName": "Example item 2",
      "itemEnabled": false,
      "key": "item002"
    }
  ]
}
```

The following example shows the array that is returned from the items function.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "variables": {
    "entities": {
      "item002": {
        "enabled": false,
        "displayName": "Example item 2",
        "number": 200
      },
      "item001": {
        "enabled": true,
        "displayName": "Example item 1",
        "number": 300
      }
    },
    "entitiesArray": "[items(variables('entities'))]"
  },
  "resources": [],
  "outputs": {
    "itemsResult": {
      "type": "array",
      "value": "[variables('entitiesArray')]"
    }
  }
}
```

The example returns:

JSON

```
"itemsResult": {
  "type": "Array",
  "value": [
    {
      "key": "item001",
      "value": {
        "displayName": "Example item 1",
        "enabled": true,
        "number": 300
      }
    },
    {
      "key": "item002",
      "value": {
        "displayName": "Example item 2",
        "enabled": false,
        "number": 200
      }
    }
```

```
  ]
}
```

In JSON, an object is an unordered collection of zero or more key/value pairs. The ordering can be different depending on the implementations. For example, the Bicep items() function sorts the objects in the alphabetical order. In other places, the original ordering can be preserved. Because of this non-determinism, avoid making any assumptions about the ordering of object keys when writing code, which interacts with deployments parameters & outputs.

json

`json(arg1)`

Converts a valid JSON string into a JSON data type.

In Bicep, use the json function.

Parameters

Parameter	Required	Type	Description
arg1	Yes	string	The value to convert to JSON. The string must be a properly formatted JSON string.

Return value

The JSON data type from the specified string, or an empty value when **null** is specified.

Remarks

If you need to include a parameter value or variable in the JSON object, use the concat function to create the string that you pass to the function.

You can also use null() to get a null value.

Example

The following example shows how to use the `json` function. Notice that you can pass in `null` for an empty object.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "jsonEmptyObject": {
      "type": "string",
      "defaultValue": "null"
    },
    "jsonObject": {
      "type": "string",
      "defaultValue": "{\"a\": \"b\"}"
    },
    "jsonString": {
      "type": "string",
      "defaultValue": "\"test\""
    },
    "jsonBoolean": {
      "type": "string",
      "defaultValue": "true"
    },
    "jsonInt": {
      "type": "string",
      "defaultValue": "3"
    },
    "jsonArray": {
      "type": "string",
      "defaultValue": "[[1,2,3 ]"
    },
    "concatValue": {
      "type": "string",
      "defaultValue": "demo value"
    }
  },
  "resources": [
  ],
  "outputs": {
    "emptyObjectOutput": {
      "type": "bool",
      "value": "[empty(json(parameters('jsonEmptyObject')))]"
    },
    "objectOutput": {
      "type": "object",
      "value": "[json(parameters('jsonObject'))]"
    },
    "stringOutput": {
      "type": "string",
      "value": "[json(parameters('jsonString'))]"
    },
    "booleanOutput": {
      "type": "bool",
      "value": "[json(parameters('jsonBoolean'))]"
```

```
    },
    "intOutput": {
      "type": "int",
      "value": "[json(parameters('jsonInt'))]"
    },
    "arrayOutput": {
      "type": "array",
      "value": "[json(parameters('jsonArray'))]"
    },
    "concatObjectOutput": {
      "type": "object",
      "value": "[json(concat('{\"a\": \"', parameters('concatValue'),
'\"}'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
emptyObjectOutput	Boolean	True
objectOutput	Object	{"a": "b"}
stringOutput	String	test
booleanOutput	Boolean	True
intOutput	Integer	3
arrayOutput	Array	[1, 2, 3]
concatObjectOutput	Object	{ "a": "demo value" }

length

`length(arg1)`

Returns the number of elements in an array, characters in a string, or root-level properties in an object.

In Bicep, use the length function.

Parameters

Parameter	Required	Type	Description

Parameter	Required	Type	Description
arg1	Yes	array, string, or object	The array to use for getting the number of elements, the string to use for getting the number of characters, or the object to use for getting the number of root-level properties.

Return value

An int.

Example

The following example shows how to use length with an array and string:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "arrayToTest": {
      "type": "array",
      "defaultValue": [
        "one",
        "two",
        "three"
      ]
    },
    "stringToTest": {
      "type": "string",
      "defaultValue": "One Two Three"
    },
    "objectToTest": {
      "type": "object",
      "defaultValue": {
        "propA": "one",
        "propB": "two",
        "propC": "three",
        "propD": {
          "propD-1": "sub",
          "propD-2": "sub"
        }
      }
    }
  },
  "resources": [],
  "outputs": {
    "arrayLength": {
```

```
        "type": "int",
        "value": "[length(parameters('arrayToTest'))]"
      },
      "stringLength": {
        "type": "int",
        "value": "[length(parameters('stringToTest'))]"
      },
      "objectLength": {
        "type": "int",
        "value": "[length(parameters('objectToTest'))]"
      }
    }
  }
```

The output from the preceding example with the default values is:

Name	Type	Value
arrayLength	Int	3
stringLength	Int	13
objectLength	Int	4

null

`null()`

Returns null.

The `null` function isn't available in Bicep. Use the `null` keyword instead.

Parameters

The null function doesn't accept any parameters.

Return value

A value that is always null.

Example

The following example uses the null function.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [],
  "outputs": {
    "emptyOutput": {
      "type": "bool",
      "value": "[empty(null())]"
    }
  }
}
```

The output from the preceding example is:

Name	Type	Value
emptyOutput	Bool	True

union

`union(arg1, arg2, arg3, ...)`

Returns a single array or object with all elements from the parameters. For arrays, duplicate values are included once. For objects, duplicate property names are only included once.

In Bicep, use the union function.

Parameters

Parameter	Required	Type	Description
arg1	Yes	array or object	The first value to use for joining elements.
arg2	Yes	array or object	The second value to use for joining elements.
more arguments	No	array or object	More values to use for joining elements.

Return value

An array or object.

Remarks

The union function uses the sequence of the parameters to determine the order and values of the result.

For arrays, the function iterates through each element in the first parameter and adds it to the result if it isn't already present. Then, it repeats the process for the second parameter and any additional parameters. If a value is already present, it's earlier placement in the array is preserved.

For objects, property names and values from the first parameter are added to the result. For later parameters, any new names are added to the result. If a later parameter has a property with the same name, that value overwrites the existing value. The order of the properties isn't guaranteed.

Example

The following example shows how to use union with arrays and objects:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "firstObject": {
      "type": "object",
      "defaultValue": {
        "one": "a",
        "two": "b",
        "three": "c1"
      }
    },
    "secondObject": {
      "type": "object",
      "defaultValue": {
        "three": "c2",
        "four": "d",
        "five": "e"
      }
    },
    "firstArray": {
      "type": "array",
      "defaultValue": [ "one", "two", "three" ]
    },
    "secondArray": {
      "type": "array",
      "defaultValue": [ "three", "four" ]
    }
  },
  "resources": [
```

```
  ],
  "outputs": {
    "objectOutput": {
      "type": "object",
      "value": "[union(parameters('firstObject'),
parameters('secondObject'))]"
    },
    "arrayOutput": {
      "type": "array",
      "value": "[union(parameters('firstArray'),
parameters('secondArray'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
objectOutput	Object	{"one": "a", "two": "b", "three": "c2", "four": "d", "five": "e"}
arrayOutput	Array	["one", "two", "three", "four"]

Next steps

- For a description of the sections in an ARM template, see Understand the structure and syntax of ARM templates.

Resource functions for ARM templates

Article • 03/09/2023

Resource Manager provides the following functions for getting resource values in your Azure Resource Manager template (ARM template):

- extensionResourceId
- list*
- pickZones
- providers (deprecated)
- reference
- resourceId
- subscriptionResourceId
- managementGroupResourceId
- tenantResourceId

To get values from parameters, variables, or the current deployment, see Deployment value functions.

To get deployment scope values, see Scope functions.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **resource** functions.

extensionResourceId

`extensionResourceId(baseResourceId, resourceType, resourceName1, [resourceName2], ...)`

Returns the resource ID for an extension resource. An extension resource is a resource type that's applied to another resource to add to its capabilities.

In Bicep, use the extensionResourceId function.

Parameters

Parameter	Required	Type	Description
baseResourceId	Yes	string	The resource ID for the resource that the extension resource is applied to.
resourceType	Yes	string	Type of the extension resource including resource provider namespace.
resourceName1	Yes	string	Name of the extension resource.
resourceName2	No	string	Next resource name segment, if needed.

Continue adding resource names as parameters when the resource type includes more segments.

Return value

The basic format of the resource ID returned by this function is:

```JSON
{scope}/providers/{extensionResourceProviderNamespace}/{extensionResourceType}/{extensionResourceName}
```

The scope segment varies by the base resource being extended. For example, the ID for a subscription has different segments than the ID for a resource group.

When the extension resource is applied to a **resource**, the resource ID is returned in the following format:

JSON

```
/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/{baseResourceProviderNamespace}/{baseResourceType}/{baseResourceName}/providers/{extensionResourceProviderNamespace}/{extensionResourceType}/{extensionResourceName}
```

When the extension resource is applied to a **resource group**, the returned format is:

JSON

```
/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/{extensionResourceProviderNamespace}/{extensionResourceType}/{extensionResourceName}
```

An example of using this function with a resource group is shown in the next section.

When the extension resource is applied to a **subscription**, the returned format is:

JSON

```
/subscriptions/{subscriptionId}/providers/{extensionResourceProviderNamespace}/{extensionResourceType}/{extensionResourceName}
```

When the extension resource is applied to a **management group**, the returned format is:

JSON

```
/providers/Microsoft.Management/managementGroups/{managementGroupName}/providers/{extensionResourceProviderNamespace}/{extensionResourceType}/{extensionResourceName}
```

An example of using this function with a management group is shown in the next section.

extensionResourceId example

The following example returns the resource ID for a resource group lock.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "lockName": {
      "type": "string"
    }
  },
  "variables": {},
  "resources": [],
  "outputs": {
    "lockResourceId": {
      "type": "string",
      "value": "[extensionResourceId(resourceGroup().Id , 'Microsoft.Authorization/locks',
parameters('lockName'))]"
    }
  }
}
```

A custom policy definition deployed to a management group is implemented as an extension resource. To create and assign a policy, deploy the following template to a management group.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-08-01/managementGroupDeploymentTemplate.json#",
```

```
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.5.6.12127",
      "templateHash": "1532257987028557958"
    }
  },
  "parameters": {
    "targetMG": {
      "type": "string",
      "metadata": {
        "description": "Target Management Group"
      }
    },
    "allowedLocations": {
      "type": "array",
      "defaultValue": [
        "australiaeast",
        "australiasoutheast",
        "australiacentral"
      ],
      "metadata": {
        "description": "An array of the allowed locations, all other locations will be denied by the created
policy."
      }
    }
  },
  "variables": {
    "mgScope": "[tenantResourceId('Microsoft.Management/managementGroups', parameters('targetMG'))]",
    "policyDefinitionName": "LocationRestriction"
  },
  "resources": [
    {
      "type": "Microsoft.Authorization/policyDefinitions",
      "apiVersion": "2020-03-01",
      "name": "[variables('policyDefinitionName')]",
      "properties": {
        "policyType": "Custom",
        "mode": "All",
        "parameters": {},
        "policyRule": {
          "if": {
            "not": {
              "field": "location",
              "in": "[parameters('allowedLocations')]"
            }
          },
          "then": {
            "effect": "deny"
          }
        }
      }
    },
    {
      "type": "Microsoft.Authorization/policyAssignments",
      "apiVersion": "2020-03-01",
      "name": "location-lock",
      "properties": {
        "scope": "[variables('mgScope')]",
        "policyDefinitionId": "[extensionResourceId(variables('mgScope'),
'Microsoft.Authorization/policyDefinitions', variables('policyDefinitionName'))]"
      },
      "dependsOn": [
        "[extensionResourceId(managementGroup().id, 'Microsoft.Authorization/policyDefinitions',
variables('policyDefinitionName'))]"
      ]
    }
  ]
}
```

Built-in policy definitions are tenant level resources. For an example of deploying a built-in policy definition, see tenantResourceId.

list*

`list{Value}(resourceName or resourceIdentifier, apiVersion, functionValues)`

The syntax for this function varies by name of the list operations. Each implementation returns values for the resource type that supports a list operation. The operation name must start with `list` and may have a suffix. Some common usages are `list`, `listKeys`, `listKeyValue`, and `listSecrets`.

In Bicep, use the list* function.

Parameters

Parameter	Required	Type	Description
resourceName or resourceIdentifier	Yes	string	Unique identifier for the resource.
apiVersion	Yes	string	API version of resource runtime state. Typically, in the format, **yyyy-mm-dd**.
functionValues	No	object	An object that has values for the function. Only provide this object for functions that support receiving an object with parameter values, such as `listAccountSas` on a storage account. An example of passing function values is shown in this article.

Valid uses

The list functions can be used in the properties of a resource definition. Don't use a list function that exposes sensitive information in the outputs section of a template. Output values are stored in the deployment history and could be retrieved by a malicious user.

When used with property iteration, you can use the list functions for `input` because the expression is assigned to the resource property. You can't use them with `count` because the count must be determined before the list function is resolved.

Implementations

The possible uses of `list*` are shown in the following table.

Resource type	Function name
Microsoft.Addons/supportProviders	listsupportplaninfo
Microsoft.AnalysisServices/servers	listGatewayStatus
Microsoft.ApiManagement/service/authorizationServers	listSecrets
Microsoft.ApiManagement/service/gateways	listKeys
Microsoft.ApiManagement/service/identityProviders	listSecrets
Microsoft.ApiManagement/service/namedValues	listValue
Microsoft.ApiManagement/service/openidConnectProviders	listSecrets
Microsoft.ApiManagement/service/subscriptions	listSecrets
Microsoft.AppConfiguration/configurationStores	ListKeys
Microsoft.AppPlatform/Spring	listTestKeys
Microsoft.Automation/automationAccounts	listKeys
Microsoft.Batch/batchAccounts	listKeys
Microsoft.BatchAI/workspaces/experiments/jobs	listoutputfiles
Microsoft.BotService/botServices/channels	listChannelWithKeys

Resource type	Function name
Microsoft.Cache/redis	listKeys
Microsoft.CognitiveServices/accounts	listKeys
Microsoft.ContainerRegistry/registries	listBuildSourceUploadUrl
Microsoft.ContainerRegistry/registries	listCredentials
Microsoft.ContainerRegistry/registries	listUsages
Microsoft.ContainerRegistry/registries/agentpools	listQueueStatus
Microsoft.ContainerRegistry/registries/buildTasks	listSourceRepositoryProperties
Microsoft.ContainerRegistry/registries/buildTasks/steps	listBuildArguments
Microsoft.ContainerRegistry/registries/taskruns	listDetails
Microsoft.ContainerRegistry/registries/webhooks	listEvents
Microsoft.ContainerRegistry/registries/runs	listLogSasUrl
Microsoft.ContainerRegistry/registries/tasks	listDetails
Microsoft.ContainerService/managedClusters	listClusterAdminCredential
Microsoft.ContainerService/managedClusters	listClusterMonitoringUserCredential
Microsoft.ContainerService/managedClusters	listClusterUserCredential
Microsoft.ContainerService/managedClusters/accessProfiles	listCredential
Microsoft.DataBox/jobs	listCredentials
Microsoft.DataFactory/datafactories/gateways	listauthkeys
Microsoft.DataFactory/factories/integrationruntimes	listauthkeys
Microsoft.DataLakeAnalytics/accounts/storageAccounts/Containers	listSasTokens
Microsoft.DataShare/accounts/shares	listSynchronizations
Microsoft.DataShare/accounts/shareSubscriptions	listSourceShareSynchronizationSettings
Microsoft.DataShare/accounts/shareSubscriptions	listSynchronizationDetails
Microsoft.DataShare/accounts/shareSubscriptions	listSynchronizations
Microsoft.Devices/iotHubs	listKeys
Microsoft.Devices/iotHubs/iotHubKeys	listKeys
Microsoft.Devices/provisioningServices/keys	listKeys
Microsoft.Devices/provisioningServices	listKeys
Microsoft.DevTestLab/labs	ListVhds
Microsoft.DevTestLab/labs/schedules	ListApplicable
Microsoft.DevTestLab/labs/users/serviceFabrics	ListApplicableSchedules
Microsoft.DevTestLab/labs/virtualMachines	ListApplicableSchedules
Microsoft.DocumentDB/databaseAccounts	listConnectionStrings
Microsoft.DocumentDB/databaseAccounts	listKeys
Microsoft.DocumentDB/databaseAccounts/notebookWorkspaces	listConnectionInfo
Microsoft.DomainRegistration/topLevelDomains	listAgreements
Microsoft.EventGrid/domains	listKeys

Resource type	Function name
Microsoft.EventGrid/topics	listKeys
Microsoft.EventHub/namespaces/authorizationRules	listKeys
Microsoft.EventHub/namespaces/disasterRecoveryConfigs/authorizationRules	listKeys
Microsoft.EventHub/namespaces/eventhubs/authorizationRules	listKeys
Microsoft.ImportExport/jobs	listBitLockerKeys
Microsoft.Kusto/Clusters/Databases	ListPrincipals
Microsoft.LabServices/labs/users	list
Microsoft.LabServices/labs/virtualMachines	list
Microsoft.Logic/integrationAccounts/agreements	listContentCallbackUrl
Microsoft.Logic/integrationAccounts/assemblies	listContentCallbackUrl
Microsoft.Logic/integrationAccounts	listCallbackUrl
Microsoft.Logic/integrationAccounts	listKeyVaultKeys
Microsoft.Logic/integrationAccounts/maps	listContentCallbackUrl
Microsoft.Logic/integrationAccounts/partners	listContentCallbackUrl
Microsoft.Logic/integrationAccounts/schemas	listContentCallbackUrl
Microsoft.Logic/workflows	listCallbackUrl
Microsoft.Logic/workflows	listSwagger
Microsoft.Logic/workflows/runs/actions	listExpressionTraces
Microsoft.Logic/workflows/runs/actions/repetitions	listExpressionTraces
Microsoft.Logic/workflows/triggers	listCallbackUrl
Microsoft.Logic/workflows/versions/triggers	listCallbackUrl
Microsoft.MachineLearning/webServices	listkeys
Microsoft.MachineLearning/Workspaces	listworkspacekeys
Microsoft.MachineLearningServices/workspaces/computes	listKeys
Microsoft.MachineLearningServices/workspaces/computes	listNodes
Microsoft.MachineLearningServices/workspaces	listKeys
Microsoft.Maps/accounts	listKeys
Microsoft.Media/mediaservices/assets	listContainerSas
Microsoft.Media/mediaservices/assets	listStreamingLocators
Microsoft.Media/mediaservices/streamingLocators	listContentKeys
Microsoft.Media/mediaservices/streamingLocators	listPaths
Microsoft.Network/applicationSecurityGroups	listIpConfigurations
Microsoft.NotificationHubs/Namespaces/authorizationRules	listkeys
Microsoft.NotificationHubs/Namespaces/NotificationHubs/authorizationRules	listkeys
Microsoft.OperationalInsights/workspaces	list
Microsoft.OperationalInsights/workspaces	listKeys
Microsoft.PolicyInsights/remediations	listDeployments

Resource type	Function name
Microsoft.RedHatOpenShift/openShiftClusters	listCredentials
Microsoft.Relay/namespaces/authorizationRules	listKeys
Microsoft.Relay/namespaces/disasterRecoveryConfigs/authorizationRules	listKeys
Microsoft.Relay/namespaces/HybridConnections/authorizationRules	listKeys
Microsoft.Relay/namespaces/WcfRelays/authorizationRules	listkeys
Microsoft.Search/searchServices	listAdminKeys
Microsoft.Search/searchServices	listQueryKeys
Microsoft.ServiceBus/namespaces/authorizationRules	listKeys
Microsoft.ServiceBus/namespaces/disasterRecoveryConfigs/authorizationRules	listKeys
Microsoft.ServiceBus/namespaces/queues/authorizationRules	listKeys
Microsoft.ServiceBus/namespaces/topics/authorizationRules	listKeys
Microsoft.SignalRService/SignalR	listKeys
Microsoft.Storage/storageAccounts	listAccountSas
Microsoft.Storage/storageAccounts	listKeys
Microsoft.Storage/storageAccounts	listServiceSas
Microsoft.StorSimple/managers/devices	listFailoverSets
Microsoft.StorSimple/managers/devices	listFailoverTargets
Microsoft.StorSimple/managers	listActivationKey
Microsoft.StorSimple/managers	listPublicEncryptionKey
Microsoft.Synapse/workspaces/integrationRuntimes	listAuthKeys
Microsoft.Web/connectionGateways	ListStatus
microsoft.web/connections	listconsentlinks
Microsoft.Web/customApis	listWsdlInterfaces
microsoft.web/locations	listwsdlinterfaces
microsoft.web/apimanagementaccounts/apis/connections	listconnectionkeys
microsoft.web/apimanagementaccounts/apis/connections	`listSecrets`
microsoft.web/sites/backups	list
Microsoft.Web/sites/config	list
microsoft.web/sites/functions	listKeys
microsoft.web/sites/functions	listSecrets
microsoft.web/sites/hybridconnectionnamespaces/relays	listKeys
microsoft.web/sites	listsyncfunctiontriggerstatus
microsoft.web/sites/slots/functions	listSecrets
microsoft.web/sites/slots/backups	list
Microsoft.Web/sites/slots/config	list
microsoft.web/sites/slots/functions	listSecrets

To determine which resource types have a list operation, you have the following options:

- View the REST API operations for a resource provider, and look for list operations. For example, storage accounts have the listKeys operation.
- Use the Get-AzProviderOperation PowerShell cmdlet. The following example gets all list operations for storage accounts:

  PowerShell

  ```
  Get-AzProviderOperation -OperationSearchString "Microsoft.Storage/*" | where {$_.Operation -like "*list*"}
  | FT Operation
  ```

- Use the following Azure CLI command to filter only the list operations:

  Azure CLI

  ```
  az provider operation show --namespace Microsoft.Storage --query "resourceTypes[?
  name=='storageAccounts'].operations[].name | [?contains(@, 'list')]"
  ```

Return value

The returned object varies by the `list` function you use. For example, the `listKeys` for a storage account returns the following format:

JSON

```
{
  "keys": [
    {
      "keyName": "key1",
      "permissions": "Full",
      "value": "{value}"
    },
    {
      "keyName": "key2",
      "permissions": "Full",
      "value": "{value}"
    }
  ]
}
```

Other `list` functions have different return formats. To see the format of a function, include it in the outputs section as shown in the example template.

Remarks

Specify the resource by using either the resource name or the resourceId function. When using a `list` function in the same template that deploys the referenced resource, use the resource name.

If you use a `list` function in a resource that is conditionally deployed, the function is evaluated even if the resource isn't deployed. You get an error if the `list` function refers to a resource that doesn't exist. Use the `if` function to make sure the function is only evaluated when the resource is being deployed. See the if function for a sample template that uses `if` and `list` with a conditionally deployed resource.

List example

The following example uses `listKeys` when setting a value for deployment scripts.

JSON

```
"storageAccountSettings": {
  "storageAccountName": "[variables('storageAccountName')]",
```

```
  "storageAccountKey": "[listKeys(resourceId('Microsoft.Storage/storageAccounts',
variables('storageAccountName')), '2019-06-01').keys[0].value]"
}
```

The next example shows a `list` function that takes a parameter. In this case, the function is `listAccountSas`. Pass an object for the expiry time. The expiry time must be in the future.

JSON

```
"parameters": {
  "accountSasProperties": {
    "type": "object",
    "defaultValue": {
      "signedServices": "b",
      "signedPermission": "r",
      "signedExpiry": "2020-08-20T11:00:00Z",
      "signedResourceTypes": "s"
    }
  }
},
...
"sasToken": "[listAccountSas(parameters('storagename'), '2018-02-01',
parameters('accountSasProperties')).accountSasToken]"
```

pickZones

`pickZones(providerNamespace, resourceType, location, [numberOfZones], [offset])`

Determines whether a resource type supports zones for the specified location or region. This function **only supports zonal resources**. Zone redundant services return an empty array. For more information, see Azure Services that support Availability Zones.

In Bicep, use the pickZones function.

Parameters

Parameter	Required	Type	Description
providerNamespace	Yes	string	The resource provider namespace for the resource type to check for zone support.
resourceType	Yes	string	The resource type to check for zone support.
location	Yes	string	The region to check for zone support.
numberOfZones	No	integer	The number of logical zones to return. The default is 1. The number must be a positive integer from 1 to 3. Use 1 for single-zoned resources. For multi-zoned resources, the value must be less than or equal to the number of supported zones.
offset	No	integer	The offset from the starting logical zone. The function returns an error if offset plus numberOfZones exceeds the number of supported zones.

Return value

An array with the supported zones. When using the default values for offset and `numberOfZones`, a resource type and region that supports zones returns the following array:

JSON

```
[
    "1"
]
```

When the `numberOfZones` parameter is set to 3, it returns:

```JSON
[
    "1",
    "2",
    "3"
]
```

When the resource type or region doesn't support zones, an empty array is returned. An empty array is also returned for zone redundant services.

```JSON
[
]
```

Remarks

There are different categories for Azure Availability Zones - zonal and zone-redundant. The `pickZones` function can be used to return an availability zone for a zonal resource. For zone redundant services (ZRS), the function returns an empty array. Zonal resources typically have a `zones` property at the top level of the resource definition. To determine the category of support for availability zones, see Azure Services that support Availability Zones.

To determine if a given Azure region or location supports availability zones, call the `pickZones` function with a zonal resource type, such as `Microsoft.Network/publicIPAddresses`. If the response isn't empty, the region supports availability zones.

pickZones example

The following template shows three results for using the `pickZones` function.

```JSON
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {},
  "functions": [],
  "variables": {},
  "resources": [],
  "outputs": {
    "supported": {
      "type": "array",
      "value": "[pickZones('Microsoft.Compute', 'virtualMachines', 'westus2')]"
    },
    "notSupportedRegion": {
      "type": "array",
      "value": "[pickZones('Microsoft.Compute', 'virtualMachines', 'westus')]"
    },
    "notSupportedType": {
      "type": "array",
      "value": "[pickZones('Microsoft.Cdn', 'profiles', 'westus2')]"
    }
  }
}
```

The output from the preceding examples returns three arrays.

Name	Type	Value
supported	array	["1"]
notSupportedRegion	array	[]
notSupportedType	array	[]

You can use the response from `pickZones` to determine whether to provide null for zones or assign virtual machines to different zones. The following example sets a value for the zone based on the availability of zones.

JSON

```
"zones": {
  "value": "[if(not(empty(pickZones('Microsoft.Compute', 'virtualMachines', 'westus2'))),
string(add(mod(copyIndex(),3),1)), json('null'))]"
},
```

Azure Cosmos DB isn't a zonal resource, but you can use the `pickZones` function to determine whether to enable zone redundancy for georeplication. Pass the **Microsoft.Storage/storageAccounts** resource type to determine whether to enable zone redundancy.

JSON

```
"resources": [
  {
    "type": "Microsoft.DocumentDB/databaseAccounts",
    "apiVersion": "2021-04-15",
    "name": "[variables('accountName_var')]",
    "location": "[parameters('location')]",
    "kind": "GlobalDocumentDB",
    "properties": {
      "consistencyPolicy": "[variables('consistencyPolicy')[parameters('defaultConsistencyLevel')]]",
      "locations": [
        {
          "locationName": "[parameters('primaryRegion')]",
          "failoverPriority": 0,
          "isZoneRedundant": "[if(empty(pickZones('Microsoft.Storage', 'storageAccounts',
parameters('primaryRegion'))), bool('false'), bool('true'))]",
        },
        {
          "locationName": "[parameters('secondaryRegion')]",
          "failoverPriority": 1,
          "isZoneRedundant": "[if(empty(pickZones('Microsoft.Storage', 'storageAccounts',
parameters('secondaryRegion'))), bool('false'), bool('true'))]",
        }
      ],
      "databaseAccountOfferType": "Standard",
      "enableAutomaticFailover": "[parameters('automaticFailover')]"
    }
  }
]
```

providers

The providers function has been deprecated in ARM templates. We no longer recommend using it. If you used this function to get an API version for the resource provider, we recommend that you provide a specific API version in your template. Using a dynamically returned API version can break your template if the properties change between versions.

In Bicep, the providers function is deprecated.

The providers operation is still available through the REST API. It can be used outside of an ARM template to get information about a resource provider.

reference

`reference(resourceName or resourceIdentifier, [apiVersion], ['Full'])`

Returns an object representing a resource's runtime state.

In Bicep, use the reference function.

Parameters

Parameter	Required	Type	Description
resourceName or resourceIdentifier	Yes	string	Name or unique identifier of a resource. When referencing a resource in the current template, provide only the resource name as a parameter. When referencing a previously deployed resource or when the name of the resource is ambiguous, provide the resource ID.
apiVersion	No	string	API version of the specified resource. **This parameter is required when the resource isn't provisioned within same template.** Typically, in the format, **yyyy-mm-dd**. For valid API versions for your resource, see template reference.
'Full'	No	string	Value that specifies whether to return the full resource object. If you don't specify `'Full'`, only the properties object of the resource is returned. The full object includes values such as the resource ID and location.

Return value

Every resource type returns different properties for the reference function. The function doesn't return a single, predefined format. Also, the returned value differs based on the value of the `'Full'` argument. To see the properties for a resource type, return the object in the outputs section as shown in the example.

Remarks

The reference function retrieves the runtime state of either a previously deployed resource or a resource deployed in the current template. This article shows examples for both scenarios.

Typically, you use the `reference` function to return a particular value from an object, such as the blob endpoint URI or fully qualified domain name.

JSON

```
"outputs": {
  "BlobUri": {
    "type": "string",
    "value": "[reference(resourceId('Microsoft.Storage/storageAccounts',
parameters('storageAccountName'))).primaryEndpoints.blob]"
  },
  "FQDN": {
    "type": "string",
    "value": "[reference(resourceId('Microsoft.Network/publicIPAddresses',
parameters('ipAddressName'))).dnsSettings.fqdn]"
  }
}
```

Use `'Full'` when you need resource values that aren't part of the properties schema. For example, to set key vault access policies, get the identity properties for a virtual machine.

JSON

```
{
  "type": "Microsoft.KeyVault/vaults",
  "apiVersion": "2019-09-01",
  "name": "vaultName",
  "properties": {
    "tenantId": "[subscription().tenantId]",
    "accessPolicies": [
      {
        "tenantId": "[reference(resourceId('Microsoft.Compute/virtualMachines', variables('vmName')), '2019-03-
01', 'Full').identity.tenantId]",
        "objectId": "[reference(resourceId('Microsoft.Compute/virtualMachines', variables('vmName')), '2019-03-
01', 'Full').identity.principalId]",
        "permissions": {
          "keys": [
            "all"
          ],
          "secrets": [
            "all"
          ]
```

```
        }
      }
    ],
    ...
```

Valid uses

The `reference` function can only be used in the outputs section of a template or deployment and properties object of a resource definition. It cannot be used for resource properties such as `type`, `name`, `location` and other top level properties of the resource definition. When used with property iteration, you can use the `reference` function for `input` because the expression is assigned to the resource property.

You can't use the `reference` function to set the value of the `count` property in a copy loop. You can use to set other properties in the loop. Reference is blocked for the count property because that property must be determined before the `reference` function is resolved.

To use the `reference` function or any `list*` function in the outputs section of a nested template, you must set the `expressionEvaluationOptions` to use inner scope evaluation or use a linked instead of a nested template.

If you use the `reference` function in a resource that is conditionally deployed, the function is evaluated even if the resource isn't deployed. You get an error if the `reference` function refers to a resource that doesn't exist. Use the `if` function to make sure the function is only evaluated when the resource is being deployed. See the if function for a sample template that uses `if` and `reference` with a conditionally deployed resource.

Implicit dependency

By using the `reference` function, you implicitly declare that one resource depends on another resource if the referenced resource is provisioned within same template and you refer to the resource by its name (not resource ID). You don't need to also use the `dependsOn` property. The function isn't evaluated until the referenced resource has completed deployment.

Resource name or identifier

When referencing a resource that is deployed in the same template, provide the name of the resource.

JSON

```
"value": "[reference(parameters('storageAccountName'))]"
```

When referencing a resource that isn't deployed in the same template, provide the resource ID and `apiVersion`.

JSON

```
"value": "[reference(resourceId(parameters('storageResourceGroup'), 'Microsoft.Storage/storageAccounts',
parameters('storageAccountName')), '2018-07-01')]"
```

To avoid ambiguity about which resource you're referencing, you can provide a fully qualified resource identifier.

JSON

```
"value": "[reference(resourceId('Microsoft.Network/publicIPAddresses', parameters('ipAddressName')))]"
```

When constructing a fully qualified reference to a resource, the order to combine segments from the type and name isn't simply a concatenation of the two. Instead, after the namespace, use a sequence of *type/name* pairs from least specific to most specific:

`{resource-provider-namespace}/{parent-resource-type}/{parent-resource-name}[/{child-resource-type}/{child-resource-name}]`

For example:

`Microsoft.Compute/virtualMachines/myVM/extensions/myExt` is correct
`Microsoft.Compute/virtualMachines/extensions/myVM/myExt` is not correct

To simplify the creation of any resource ID, use the `resourceId()` functions described in this document instead of the `concat()` function.

Get managed identity

Managed identities for Azure resources are extension resource types that are created implicitly for some resources. Because the managed identity isn't explicitly defined in the template, you must reference the resource that the identity is applied to. Use `Full` to get all of the properties, including the implicitly created identity.

The pattern is:

```
"[reference(resourceId(<resource-provider-namespace>, <resource-name>), <API-version>, 'Full').Identity.propertyName]"
```

For example, to get the principal ID for a managed identity that is applied to a virtual machine, use:

JSON

```
"[reference(resourceId('Microsoft.Compute/virtualMachines', variables('vmName')),'2019-12-01',
'Full').identity.principalId]",
```

Or, to get the tenant ID for a managed identity that is applied to a virtual machine scale set, use:

JSON

```
"[reference(resourceId('Microsoft.Compute/virtualMachineScaleSets',  variables('vmNodeType0Name')), 2019-12-01,
'Full').Identity.tenantId]"
```

Reference example

The following example deploys a resource, and references that resource.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageAccountName": {
      "type": "string"
    }
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-04-01",
      "name": "[parameters('storageAccountName')]",
      "location": "[resourceGroup().location]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "Storage",
      "tags": {},
      "properties": {
      }
    }
  ],
  "outputs": {
    "referenceOutput": {
      "type": "object",
      "value": "[reference(parameters('storageAccountName'))]"
    },
```

```
    "fullReferenceOutput": {
      "type": "object",
      "value": "[reference(parameters('storageAccountName'), '2021-04-01', 'Full')]"
    }
  }
}
```

The preceding example returns the two objects. The properties object is in the following format:

JSON

```
{
   "creationTime": "2017-10-09T18:55:40.5863736Z",
   "primaryEndpoints": {
     "blob": "https://examplestorage.blob.core.windows.net/",
     "file": "https://examplestorage.file.core.windows.net/",
     "queue": "https://examplestorage.queue.core.windows.net/",
     "table": "https://examplestorage.table.core.windows.net/"
   },
   "primaryLocation": "southcentralus",
   "provisioningState": "Succeeded",
   "statusOfPrimary": "available",
   "supportsHttpsTrafficOnly": false
}
```

The full object is in the following format:

JSON

```
{
  "apiVersion":"2021-04-01",
  "location":"southcentralus",
  "sku": {
    "name":"Standard_LRS",
    "tier":"Standard"
  },
  "tags":{},
  "kind":"Storage",
  "properties": {
    "creationTime":"2017-10-09T18:55:40.5863736Z",
    "primaryEndpoints": {
      "blob":"https://examplestorage.blob.core.windows.net/",
      "file":"https://examplestorage.file.core.windows.net/",
      "queue":"https://examplestorage.queue.core.windows.net/",
      "table":"https://examplestorage.table.core.windows.net/"
    },
    "primaryLocation":"southcentralus",
    "provisioningState":"Succeeded",
    "statusOfPrimary":"available",
    "supportsHttpsTrafficOnly":false
  },
  "subscriptionId":"<subscription-id>",
  "resourceGroupName":"functionexamplegroup",
  "resourceId":"Microsoft.Storage/storageAccounts/examplestorage",
  "referenceApiVersion":"2021-04-01",
  "condition":true,
  "isConditionTrue":true,
  "isTemplateResource":false,
  "isAction":false,
  "provisioningOperation":"Read"
}
```

The following example template references a storage account that isn't deployed in this template. The storage account already exists within the same subscription.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
```

```
    "storageResourceGroup": {
      "type": "string"
    },
    "storageAccountName": {
      "type": "string"
    }
  },
  "resources": [],
  "outputs": {
    "ExistingStorage": {
      "type": "object",
      "value": "[reference(resourceId(parameters('storageResourceGroup'), 'Microsoft.Storage/storageAccounts',
parameters('storageAccountName')), '2021-04-01')]"
    }
  }
}
```

resourceGroup

See the resourceGroup scope function.

In Bicep, use the resourcegroup scope function.

resourceId

`resourceId([subscriptionId], [resourceGroupName], resourceType, resourceName1, [resourceName2], ...)`

Returns the unique identifier of a resource. You use this function when the resource name is ambiguous or not provisioned within the same template. The format of the returned identifier varies based on whether the deployment happens at the scope of a resource group, subscription, management group, or tenant.

In Bicep, use the resourceId function.

Parameters

Parameter	Required	Type	Description
subscriptionId	No	string (In GUID format)	Default value is the current subscription. Specify this value when you need to retrieve a resource in another subscription. Only provide this value when deploying at the scope of a resource group or subscription.
resourceGroupName	No	string	Default value is current resource group. Specify this value when you need to retrieve a resource in another resource group. Only provide this value when deploying at the scope of a resource group.
resourceType	Yes	string	Type of resource including resource provider namespace.
resourceName1	Yes	string	Name of resource.
resourceName2	No	string	Next resource name segment, if needed.

Continue adding resource names as parameters when the resource type includes more segments.

Return value

The resource ID is returned in different formats at different scopes:

- Resource group scope:

  JSON

  ```
  /subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/{resourceProviderNamespace}/{r
  esourceType}/{resourceName}
  ```

- Subscription scope:

  ```JSON
  /subscriptions/{subscriptionId}/providers/{resourceProviderNamespace}/{resourceType}/{resourceName}
  ```

- Management group or tenant scope:

  ```JSON
  /providers/{resourceProviderNamespace}/{resourceType}/{resourceName}
  ```

To avoid confusion, we recommend that you don't use `resourceId` when working with resources deployed to the subscription, management group, or tenant. Instead, use the ID function that is designed for the scope.

- For subscription-level resources, use the subscriptionResourceId function.
- For management group-level resources, use the managementGroupResourceId function. Use the extensionResourceId function to reference a resource that is implemented as an extension of a management group. For example, custom policy definitions that are deployed to a management group are extensions of the management group. Use the tenantResourceId function to reference resources that are deployed to the tenant but available in your management group. For example, built-in policy definitions are implemented as tenant level resources.
- For tenant-level resources, use the tenantResourceId function. Use `tenantResourceId` for built-in policy definitions because they're implemented at the tenant level.

Remarks

The number of parameters you provide varies based on whether the resource is a parent or child resource, and whether the resource is in the same subscription or resource group.

To get the resource ID for a parent resource in the same subscription and resource group, provide the type and name of the resource.

```JSON
"[resourceId('Microsoft.ServiceBus/namespaces', 'namespace1')]"
```

To get the resource ID for a child resource, pay attention to the number of segments in the resource type. Provide a resource name for each segment of the resource type. The name of the segment corresponds to the resource that exists for that part of the hierarchy.

```JSON
"[resourceId('Microsoft.ServiceBus/namespaces/queues/authorizationRules', 'namespace1', 'queue1', 'auth1')]"
```

To get the resource ID for a resource in the same subscription but different resource group, provide the resource group name.

```JSON
"[resourceId('otherResourceGroup', 'Microsoft.Storage/storageAccounts', 'examplestorage')]"
```

To get the resource ID for a resource in a different subscription and resource group, provide the subscription ID and resource group name.

```JSON
"[resourceId('xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx', 'otherResourceGroup',
'Microsoft.Storage/storageAccounts','examplestorage')]"
```

Often, you need to use this function when using a storage account or virtual network in an alternate resource group. The following example shows how a resource from an external resource group can easily be used:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "location": {
      "type": "string"
    },
    "virtualNetworkName": {
      "type": "string"
    },
    "virtualNetworkResourceGroup": {
      "type": "string"
    },
    "subnet1Name": {
      "type": "string"
    },
    "nicName": {
      "type": "string"
    }
  },
  "variables": {
    "subnet1Ref": "[resourceId(parameters('virtualNetworkResourceGroup'),
'Microsoft.Network/virtualNetworks/subnets', parameters('virtualNetworkName'), parameters('subnet1Name'))]"
  },
  "resources": [
    {
      "type": "Microsoft.Network/networkInterfaces",
      "apiVersion": "2021-02-01",
      "name": "[parameters('nicName')]",
      "location": "[parameters('location')]",
      "properties": {
        "ipConfigurations": [
          {
            "name": "ipconfig1",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "subnet": {
                "id": "[variables('subnet1Ref')]"
              }
            }
          }
        ]
      }
    }
  ]
}
```

Resource ID example

The following example returns the resource ID for a storage account in the resource group:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [],
  "outputs": {
    "sameRGOutput": {
      "type": "string",
      "value": "[resourceId('Microsoft.Storage/storageAccounts','examplestorage')]"
    },
    "differentRGOutput": {
      "type": "string",
      "value": "[resourceId('otherResourceGroup', 'Microsoft.Storage/storageAccounts','examplestorage')]"
    },
    "differentSubOutput": {
```

```
      "type": "string",
      "value": "[resourceId('11111111-1111-1111-1111-111111111111', 'otherResourceGroup',
'Microsoft.Storage/storageAccounts','examplestorage')]"
    },
    "nestedResourceOutput": {
      "type": "string",
      "value": "[resourceId('Microsoft.SQL/servers/databases', 'serverName', 'databaseName')]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
sameRGOutput	String	/subscriptions/{current-sub-id}/resourceGroups/examplegroup/providers/Microsoft.Storage/storageAccounts/examplestorage
differentRGOutput	String	/subscriptions/{current-sub-id}/resourceGroups/otherResourceGroup/providers/Microsoft.Storage/storageAccounts/examplestorage
differentSubOutput	String	/subscriptions/11111111-1111-1111-1111-111111111111/resourceGroups/otherResourceGroup/providers/Microsoft.Storage/storageAccounts/examplestorage
nestedResourceOutput	String	/subscriptions/{current-sub-id}/resourceGroups/examplegroup/providers/Microsoft.SQL/servers/serverName/databases/databaseName

subscription

See the subscription scope function.

In Bicep, use the subscription scope function.

subscriptionResourceId

`subscriptionResourceId([subscriptionId], resourceType, resourceName1, [resourceName2], ...)`

Returns the unique identifier for a resource deployed at the subscription level.

In Bicep, use the subscriptionResourceId function.

Parameters

Parameter	Required	Type	Description
subscriptionId	No	string (in GUID format)	Default value is the current subscription. Specify this value when you need to retrieve a resource in another subscription.
resourceType	Yes	string	Type of resource including resource provider namespace.
resourceName1	Yes	string	Name of resource.
resourceName2	No	string	Next resource name segment, if needed.

Continue adding resource names as parameters when the resource type includes more segments.

Return value

The identifier is returned in the following format:

JSON

```
/subscriptions/{subscriptionId}/providers/{resourceProviderNamespace}/{resourceType}/{resourceName}
```

Remarks

You use this function to get the resource ID for resources that are deployed to the subscription rather than a resource group. The returned ID differs from the value returned by the resourceId function by not including a resource group value.

subscriptionResourceID example

The following template assigns a built-in role. You can deploy it to either a resource group or subscription. It uses the `subscriptionResourceId` function to get the resource ID for built-in roles.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "principalId": {
      "type": "string",
      "metadata": {
        "description": "The principal to assign the role to"
      }
    },
    "builtInRoleType": {
      "type": "string",
      "allowedValues": [
        "Owner",
        "Contributor",
        "Reader"
      ],
      "metadata": {
        "description": "Built-in role to assign"
      }
    },
    "roleNameGuid": {
      "type": "string",
      "defaultValue": "[newGuid()]",
      "metadata": {
        "description": "A new GUID used to identify the role assignment"
      }
    }
  },
  "variables": {
    "Owner": "[subscriptionResourceId('Microsoft.Authorization/roleDefinitions', '8e3af657-a8ff-443c-a75c-
2fe8c4bcb635')]",
    "Contributor": "[subscriptionResourceId('Microsoft.Authorization/roleDefinitions', 'b24988ac-6180-42a0-ab88-
20f7382dd24c')]",
    "Reader": "[subscriptionResourceId('Microsoft.Authorization/roleDefinitions', 'acdd72a7-3385-48ef-bd42-
f606fba81ae7')]"
  },
  "resources": [
    {
      "type": "Microsoft.Authorization/roleAssignments",
      "apiVersion": "2020-10-01-preview",
      "name": "[parameters('roleNameGuid')]",
      "properties": {
        "roleDefinitionId": "[variables(parameters('builtInRoleType'))]",
        "principalId": "[parameters('principalId')]"
      }
    }
  ]
}
```

managementGroupResourceId

`managementGroupResourceId([managementGroupResourceId],resourceType, resourceName1, [resourceName2], ...)`

Returns the unique identifier for a resource deployed at the management group level.

In Bicep, use the managementGroupResourceId function.

Parameters

Parameter	Required	Type	Description
managementGroupResourceId	No	string (in GUID format)	Default value is the current management group. Specify this value when you need to retrieve a resource in another management group.
resourceType	Yes	string	Type of resource including resource provider namespace.
resourceName1	Yes	string	Name of resource.
resourceName2	No	string	Next resource name segment, if needed.

Continue adding resource names as parameters when the resource type includes more segments.

Return value

The identifier is returned in the following format:

JSON

```
/providers/Microsoft.Management/managementGroups/{managementGroupName}/providers/{resourceType}/{resourceName}
```

Remarks

You use this function to get the resource ID for resources that are deployed to the management group rather than a resource group. The returned ID differs from the value returned by the resourceId function by not including a subscription ID and a resource group value.

managementGroupResourceID example

The following template creates and assigns a policy definition. It uses the `managementGroupResourceId` function to get the resource ID for policy definition.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-08-01/managementGroupDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "targetMG": {
      "type": "string",
      "metadata": {
        "description": "Target Management Group"
      }
    },
    "allowedLocations": {
      "type": "array",
      "defaultValue": [
        "australiaeast",
        "australiasoutheast",
        "australiacentral"
      ],
      "metadata": {
        "description": "An array of the allowed locations, all other locations will be denied by the created
policy."
      }
    }
  },
  "functions": [],
  "variables": {
    "mgScope": "[tenantResourceId('Microsoft.Management/managementGroups', parameters('targetMG'))]",
    "policyDefinitionName": "LocationRestriction"
  },
  "resources": [
```

```
    {
      "type": "Microsoft.Authorization/policyDefinitions",
      "apiVersion": "2020-03-01",
      "name": "[variables('policyDefinitionName')]",
      "properties": {
        "policyType": "Custom",
        "mode": "All",
        "parameters": {},
        "policyRule": {
          "if": {
            "not": {
              "field": "location",
              "in": "[parameters('allowedLocations')]"
            }
          },
          "then": {
            "effect": "deny"
          }
        }
      }
    },
    {
      "type": "Microsoft.Authorization/policyAssignments",
      "apiVersion": "2020-03-01",
      "name": "location-lock",
      "properties": {
        "scope": "[variables('mgScope')]",
        "policyDefinitionId": "[managementGroupResourceId('Microsoft.Authorization/policyDefinitions',
variables('policyDefinitionName'))]"
      },
      "dependsOn": [
        "[format('Microsoft.Authorization/policyDefinitions/{0}', variables('policyDefinitionName'))]"
      ]
    }
  ]
}
```

tenantResourceId

`tenantResourceId(resourceType, resourceName1, [resourceName2], ...)`

Returns the unique identifier for a resource deployed at the tenant level.

In Bicep, use the tenantResourceId function.

Parameters

Parameter	Required	Type	Description
resourceType	Yes	string	Type of resource including resource provider namespace.
resourceName1	Yes	string	Name of resource.
resourceName2	No	string	Next resource name segment, if needed.

Continue adding resource names as parameters when the resource type includes more segments.

Return value

The identifier is returned in the following format:

JSON

```
/providers/{resourceProviderNamespace}/{resourceType}/{resourceName}
```

Remarks

You use this function to get the resource ID for a resource that is deployed to the tenant. The returned ID differs from the values returned by other resource ID functions by not including resource group or subscription values.

tenantResourceId example

Built-in policy definitions are tenant level resources. To deploy a policy assignment that references a built-in policy definition, use the `tenantResourceId` function.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "policyDefinitionID": {
      "type": "string",
      "defaultValue": "0a914e76-4921-4c19-b460-a2d36003525a",
      "metadata": {
        "description": "Specifies the ID of the policy definition or policy set definition being assigned."
      }
    },
    "policyAssignmentName": {
      "type": "string",
      "defaultValue": "[guid(parameters('policyDefinitionID'), resourceGroup().name)]",
      "metadata": {
        "description": "Specifies the name of the policy assignment, can be used defined or an idempotent name
as the defaultValue provides."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Authorization/policyAssignments",
      "name": "[parameters('policyAssignmentName')]",
      "apiVersion": "2020-09-01",
      "properties": {
        "scope": "[subscriptionResourceId('Microsoft.Resources/resourceGroups', resourceGroup().name)]",
        "policyDefinitionId": "[tenantResourceId('Microsoft.Authorization/policyDefinitions',
parameters('policyDefinitionID'))]"
      }
    }
  ]
}
```

Next steps

- For a description of the sections in an ARM template, see Understand the structure and syntax of ARM templates.
- To merge multiple templates, see Using linked and nested templates when deploying Azure resources.
- To iterate a specified number of times when creating a type of resource, see Resource iteration in ARM templates.
- To see how to deploy the template you've created, see Deploy resources with ARM templates and Azure PowerShell.

Scope functions for ARM templates

Article • 04/09/2023

Resource Manager provides the following functions for getting deployment scope values in your Azure Resource Manager template (ARM template):

- managementGroup
- resourceGroup
- subscription
- tenant

To get values from parameters, variables, or the current deployment, see Deployment value functions.

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **scope** functions.

managementGroup

`managementGroup()`

Returns an object with properties from the management group in the current deployment.

In Bicep, use the managementGroup scope function.

Remarks

`managementGroup()` can only be used on a management group deployments. It returns the current management group for the deployment operation. Use to get properties for the current management group.

Return value

An object with the properties for the current management group.

Management group example

The following example returns properties for the current management group.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-08-01/managementGroupDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "variables": {
    "mgInfo": "[managementGroup()]"
  },
  "resources": [],
  "outputs": {
    "mgResult": {
      "type": "object",
      "value": "[variables('mgInfo')]"
    }
  }
}
```

It returns:

JSON

```
"mgResult": {
  "type": "Object",
  "value": {
    "id": "/providers/Microsoft.Management/managementGroups/examplemg1",
    "name": "examplemg1",
    "properties": {
      "details": {
        "parent": {
          "displayName": "Tenant Root Group",
          "id": "/providers/Microsoft.Management/managementGroups/00000000-0000-0000-0000-000000000000",
          "name": "00000000-0000-0000-0000-000000000000"
        },
        "updatedBy": "00000000-0000-0000-0000-000000000000",
        "updatedTime": "2020-07-23T21:05:52.661306Z",
        "version": "1"
      },
      "displayName": "Example MG 1",
      "tenantId": "00000000-0000-0000-0000-000000000000"
    },
    "type": "/providers/Microsoft.Management/managementGroups"
  }
}
```

The next example creates a new management group and uses this function to set the parent management group.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-08-01/managementGroupDeploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "mgName": {
      "type": "string",
      "defaultValue": "[format('mg-{0}', uniqueString(newGuid()))]"
    }
  },
  "resources": [
    {
      "type": "Microsoft.Management/managementGroups",
      "apiVersion": "2020-05-01",
      "scope": "/",
      "name": "[parameters('mgName')]",
      "properties": {
        "details": {
          "parent": {
            "id": "[managementGroup().id]"
          }
        }
      }
    }
  ],
  "outputs": {
    "newManagementGroup": {
      "type": "string",
      "value": "[parameters('mgName')]"
    }
  }
}
```

resourceGroup

`resourceGroup()`

Returns an object that represents the current resource group.

In Bicep, use the resourceGroup scope function.

Return value

The returned object is in the following format:

JSON

```
{
  "id":
"/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}",
  "name": "{resourceGroupName}",
  "type":"Microsoft.Resources/resourceGroups",
  "location": "{resourceGroupLocation}",
  "managedBy": "{identifier-of-managing-resource}",
  "tags": {
  },
  "properties": {
    "provisioningState": "{status}"
  }
}
```

The **managedBy** property is returned only for resource groups that contain resources that are managed by another service. For Managed Applications, Databricks, and AKS, the value of the property is the resource ID of the managing resource.

Remarks

The `resourceGroup()` function can't be used in a template that is deployed at the subscription level. It can only be used in templates that are deployed to a resource group. You can use the `resourceGroup()` function in a linked or nested template (with inner scope) that targets a resource group, even when the parent template is deployed to the subscription. In that scenario, the linked or nested template is deployed at the resource group level. For more information about targeting a resource group in a subscription level deployment, see Deploy Azure resources to more than one subscription or resource group.

A common use of the resourceGroup function is to create resources in the same location as the resource group. The following example uses the resource group location for a default parameter value.

JSON

```
"parameters": {
  "location": {
    "type": "string",
    "defaultValue": "[resourceGroup().location]"
  }
}
```

You can also use the `resourceGroup` function to apply tags from the resource group to a resource. For more information, see Apply tags from resource group.

When using nested templates to deploy to multiple resource groups, you can specify the scope for evaluating the `resourceGroup` function. For more information, see Deploy Azure resources to more than one subscription or resource group.

Resource group example

The following example returns the properties of the resource group.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [],
  "outputs": {
    "resourceGroupOutput": {
      "type": "object",
      "value": "[resourceGroup()]"
    }
  }
}
```

The preceding example returns an object in the following format:

JSON

```
{
  "id": "/subscriptions/{subscription-id}/resourceGroups/examplegroup",
  "name": "examplegroup",
  "type":"Microsoft.Resources/resourceGroups",
  "location": "southcentralus",
  "properties": {
    "provisioningState": "Succeeded"
  }
}
```

subscription

`subscription()`

Returns details about the subscription for the current deployment.

In Bicep, use the subscription scope function.

Return value

The function returns the following format:

JSON

```
{
  "id": "/subscriptions/{subscription-id}",
  "subscriptionId": "{subscription-id}",
  "tenantId": "{tenant-id}",
  "displayName": "{name-of-subscription}"
}
```

Remarks

When using nested templates to deploy to multiple subscriptions, you can specify the scope for evaluating the subscription function. For more information, see Deploy Azure resources to more than one subscription or resource group.

Subscription example

The following example shows the subscription function called in the outputs section.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [],
  "outputs": {
    "subscriptionOutput": {
      "type": "object",
      "value": "[subscription()]"
    }
  }
}
```

tenant

`tenant()`

Returns the tenant of the user.

In Bicep, use the tenant scope function.

Remarks

`tenant()` can be used with any deployment scope. It always returns the current tenant. Use this function to get properties for the current tenant.

When setting the scope for a linked template or extension resource, use the syntax: `"scope": "/"`.

Return value

An object with properties about the current tenant.

Tenant example

The following example returns the properties for a tenant.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "variables": {
    "tenantInfo": "[tenant()]"
  },
  "resources": [],
  "outputs": {
    "tenantResult": {
      "type": "object",
      "value": "[variables('tenantInfo')]"
    }
  }
}
```

It returns:

JSON

```
"tenantResult": {
  "type": "Object",
  "value": {
    "countryCode": "US",
    "displayName": "Contoso",
    "id": "/tenants/00000000-0000-0000-0000-000000000000",
    "tenantId": "00000000-0000-0000-0000-000000000000"
  }
}
```

Next steps

- For a description of the sections in an ARM template, see Understand the structure and syntax of ARM templates.
- To merge multiple templates, see Using linked and nested templates when deploying Azure resources.
- To iterate a specified number of times when creating a type of resource, see Resource iteration in ARM templates.
- To see how to deploy the template you've created, see Deploy resources with ARM templates and Azure PowerShell.

String functions for ARM templates

Article • 04/09/2023

Resource Manager provides the following functions for working with strings in your Azure Resource Manager template (ARM template):

- base64
- base64ToJson
- base64ToString
- concat
- contains
- dataUri
- dataUriToString
- empty
- endsWith
- first
- format
- guid
- indexOf
- join
- json
- last
- lastIndexOf
- length
- newGuid
- padLeft
- replace
- skip
- split
- startsWith
- string
- substring
- take
- toLower
- toUpper
- trim
- uniqueString
- uri
- uriComponent
- uriComponentToString

> **Tip**
>
> We recommend **Bicep** because it offers the same capabilities as ARM templates and the syntax is easier to use. To learn more, see **string** functions.

base64

`base64(inputString)`

Returns the base64 representation of the input string.

In Bicep, use the base64 function.

Parameters

Parameter	Required	Type	Description
inputString	Yes	string	The value to return as a base64 representation.

Return value

A string containing the base64 representation.

Examples

The following example shows how to use the `base64` function.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "stringData": {
      "type": "string",
      "defaultValue": "one, two, three"
    },
    "jsonFormattedData": {
      "type": "string",
      "defaultValue": "{'one': 'a', 'two': 'b'}"
    }
  },
  "variables": {
```

```
      "base64String": "[base64(parameters('stringData'))]",
      "base64Object": "[base64(parameters('jsonFormattedData'))]"
    },
    "resources": [
    ],
    "outputs": {
      "base64Output": {
        "type": "string",
        "value": "[variables('base64String')]"
      },
      "toStringOutput": {
        "type": "string",
        "value": "[base64ToString(variables('base64String'))]"
      },
      "toJsonOutput": {
        "type": "object",
        "value": "[base64ToJson(variables('base64Object'))]"
      }
    }
  }
```

The output from the preceding example with the default values is:

Name	Type	Value
base64Output	String	b25lLCB0d28sIHRocmVl
toStringOutput	String	one, two, three
toJsonOutput	Object	{"one": "a", "two": "b"}

base64ToJson

`base64ToJson(base64Value)`

Converts a base64 representation to a JSON object.

In Bicep, use the base64ToJson function.

Parameters

Parameter	Required	Type	Description
base64Value	Yes	string	The base64 representation to convert to a JSON object.

Return value

A JSON object.

Examples

The following example uses the `base64ToJson` function to convert a base64 value:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "stringData": {
      "type": "string",
      "defaultValue": "one, two, three"
    },
    "jsonFormattedData": {
      "type": "string",
      "defaultValue": "{'one': 'a', 'two': 'b'}"
    }
  },
  "variables": {
    "base64String": "[base64(parameters('stringData'))]",
    "base64Object": "[base64(parameters('jsonFormattedData'))]"
  },
  "resources": [
  ],
  "outputs": {
    "base64Output": {
      "type": "string",
      "value": "[variables('base64String')]"
    },
    "toStringOutput": {
      "type": "string",
      "value": "[base64ToString(variables('base64String'))]"
    },
    "toJsonOutput": {
      "type": "object",
      "value": "[base64ToJson(variables('base64Object'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
base64Output	String	b25lLCB0d28sIHRocmVl

Name	Type	Value
toStringOutput	String	one, two, three
toJsonOutput	Object	{"one": "a", "two": "b"}

base64ToString

`base64ToString(base64Value)`

Converts a base64 representation to a string.

In Bicep, use the base64ToString function.

Parameters

Parameter	Required	Type	Description
base64Value	Yes	string	The base64 representation to convert to a string.

Return value

A string of the converted base64 value.

Examples

The following example uses the `base64ToString` function to convert a base64 value:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "stringData": {
      "type": "string",
      "defaultValue": "one, two, three"
    },
    "jsonFormattedData": {
      "type": "string",
      "defaultValue": "{'one': 'a', 'two': 'b'}"
    }
  },
  "variables": {
    "base64String": "[base64(parameters('stringData'))]",
```

```
      "base64Object": "[base64(parameters('jsonFormattedData'))]"
    },
    "resources": [
    ],
    "outputs": {
      "base64Output": {
        "type": "string",
        "value": "[variables('base64String')]"
      },
      "toStringOutput": {
        "type": "string",
        "value": "[base64ToString(variables('base64String'))]"
      },
      "toJsonOutput": {
        "type": "object",
        "value": "[base64ToJson(variables('base64Object'))]"
      }
    }
  }
```

The output from the preceding example with the default values is:

Name	Type	Value
base64Output	String	b25lLCB0d28sIHRocmVl
toStringOutput	String	one, two, three
toJsonOutput	Object	{"one": "a", "two": "b"}

concat

`concat(arg1, arg2, arg3, ...)`

Combines multiple string values and returns the concatenated string, or combines multiple arrays and returns the concatenated array.

In Bicep, use string interpolation instead of the `concat` function.

Parameters

Parameter	Required	Type	Description
arg1	Yes	string or array	The first string or array for concatenation.
more arguments	No	string or array	More strings or arrays in sequential order for concatenation.

This function can take any number of arguments, and can accept either strings or arrays for the parameters. However, you can't provide both arrays and strings for parameters. Strings are only concatenated with other strings.

Return value

A string or array of concatenated values.

Examples

The following example shows how to combine two string values and return a concatenated string.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "prefix": {
      "type": "string",
      "defaultValue": "prefix"
    }
  },
  "resources": [],
  "outputs": {
    "concatOutput": {
      "type": "string",
      "value": "[concat(parameters('prefix'), '-', uniqueString(resourceGroup().id))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
concatOutput	String	prefix-5yj4yjf5mbg72

The following example shows how to combine two arrays.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "firstArray": {
      "type": "array",
      "defaultValue": [
        "1-1",
        "1-2",
        "1-3"
      ]
    },
    "secondArray": {
      "type": "array",
      "defaultValue": [
        "2-1",
        "2-2",
        "2-3"
      ]
    }
  },
  "resources": [
  ],
  "outputs": {
    "return": {
      "type": "array",
      "value": "[concat(parameters('firstArray'), parameters('secondArray'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
return	Array	["1-1", "1-2", "1-3", "2-1", "2-2", "2-3"]

contains

`contains(container, itemToFind)`

Checks whether an array contains a value, an object contains a key, or a string contains a substring. The string comparison is case-sensitive. However, when testing if an object contains a key, the comparison is case-insensitive.

In Bicep, use the contains function.

Parameters

Parameter	Required	Type	Description
container	Yes	array, object, or string	The value that contains the value to find.
itemToFind	Yes	string or int	The value to find.

Return value

True if the item is found; otherwise, **False**.

Examples

The following example shows how to use contains with different types:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "stringToTest": {
      "type": "string",
      "defaultValue": "OneTwoThree"
    },
    "objectToTest": {
      "type": "object",
      "defaultValue": {
        "one": "a",
        "two": "b",
        "three": "c"
      }
    },
    "arrayToTest": {
      "type": "array",
      "defaultValue": [ "one", "two", "three" ]
    }
  },
  "resources": [
  ],
  "outputs": {
    "stringTrue": {
      "type": "bool",
      "value": "[contains(parameters('stringToTest'), 'e')]"
    },
    "stringFalse": {
      "type": "bool",
      "value": "[contains(parameters('stringToTest'), 'z')]"
```

```
      },
      "objectTrue": {
        "type": "bool",
        "value": "[contains(parameters('objectToTest'), 'one')]"
      },
      "objectFalse": {
        "type": "bool",
        "value": "[contains(parameters('objectToTest'), 'a')]"
      },
      "arrayTrue": {
        "type": "bool",
        "value": "[contains(parameters('arrayToTest'), 'three')]"
      },
      "arrayFalse": {
        "type": "bool",
        "value": "[contains(parameters('arrayToTest'), 'four')]"
      }
    }
  }
```

The output from the preceding example with the default values is:

Name	Type	Value
stringTrue	Bool	True
stringFalse	Bool	False
objectTrue	Bool	True
objectFalse	Bool	False
arrayTrue	Bool	True
arrayFalse	Bool	False

dataUri

`dataUri(stringToConvert)`

Converts a value to a data URI.

In Bicep, use the dataUri function.

Parameters

Parameter	Required	Type	Description

Parameter	Required	Type	Description
stringToConvert	Yes	string	The value to convert to a data URI.

Return value

A string formatted as a data URI.

Examples

The following example converts a value to a data URI, and converts a data URI to a string.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "stringToTest": {
      "type": "string",
      "defaultValue": "Hello"
    },
    "dataFormattedString": {
      "type": "string",
      "defaultValue": "data:; base64,SGVsbG8sIFdvcmxkIQ=="
    }
  },
  "resources": [],
  "outputs": {
    "dataUriOutput": {
      "value": "[dataUri(parameters('stringToTest'))]",
      "type": "string"
    },
    "toStringOutput": {
      "type": "string",
      "value": "[dataUriToString(parameters('dataFormattedString'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
dataUriOutput	String	data:text/plain;charset=utf8; base64,SGVsbG8=

Name	Type	Value
toStringOutput	String	Hello, World!

dataUriToString

`dataUriToString(dataUriToConvert)`

Converts a data URI formatted value to a string.

In Bicep, use the dataUriToString function.

Parameters

Parameter	Required	Type	Description
dataUriToConvert	Yes	string	The data URI value to convert.

Return value

A string containing the converted value.

Examples

The following example template converts a value to a data URI, and converts a data URI to a string.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "stringToTest": {
      "type": "string",
      "defaultValue": "Hello"
    },
    "dataFormattedString": {
      "type": "string",
      "defaultValue": "data:; base64,SGVsbG8sIFdvcmxkIQ=="
    }
  },
  "resources": [],
  "outputs": {
```

```
    "dataUriOutput": {
      "value": "[dataUri(parameters('stringToTest'))]",
      "type": "string"
    },
    "toStringOutput": {
      "type": "string",
      "value": "[dataUriToString(parameters('dataFormattedString'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
dataUriOutput	String	data:text/plain;charset=utf8; base64,SGVsbG8=
toStringOutput	String	Hello, World!

empty

`empty(itemToTest)`

Determines if an array, object, or string is empty.

In Bicep, use the empty function.

Parameters

Parameter	Required	Type	Description
itemToTest	Yes	array, object, or string	The value to check if it's empty.

Return value

Returns **True** if the value is empty; otherwise, **False**.

Examples

The following example checks whether an array, object, and string are empty.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
```

```
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "testArray": {
      "type": "array",
      "defaultValue": []
    },
    "testObject": {
      "type": "object",
      "defaultValue": {}
    },
    "testString": {
      "type": "string",
      "defaultValue": ""
    }
  },
  "resources": [
  ],
  "outputs": {
    "arrayEmpty": {
      "type": "bool",
      "value": "[empty(parameters('testArray'))]"
    },
    "objectEmpty": {
      "type": "bool",
      "value": "[empty(parameters('testObject'))]"
    },
    "stringEmpty": {
      "type": "bool",
      "value": "[empty(parameters('testString'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
arrayEmpty	Bool	True
objectEmpty	Bool	True
stringEmpty	Bool	True

endsWith

`endsWith(stringToSearch, stringToFind)`

Determines whether a string ends with a value. The comparison is case-insensitive.

In Bicep, use the endsWith function.

Parameters

Parameter	Required	Type	Description
stringToSearch	Yes	string	The value that contains the item to find.
stringToFind	Yes	string	The value to find.

Return value

True if the last character or characters of the string match the value; otherwise, **False**.

Examples

The following example shows how to use the `startsWith` and `endsWith` functions:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [],
  "outputs": {
    "startsTrue": {
      "type": "bool",
      "value": "[startsWith('abcdef', 'ab')]"
    },
    "startsCapTrue": {
      "type": "bool",
      "value": "[startsWith('abcdef', 'A')]"
    },
    "startsFalse": {
      "type": "bool",
      "value": "[startsWith('abcdef', 'e')]"
    },
    "endsTrue": {
      "type": "bool",
      "value": "[endsWith('abcdef', 'ef')]"
    },
    "endsCapTrue": {
      "type": "bool",
      "value": "[endsWith('abcdef', 'F')]"
    },
    "endsFalse": {
      "type": "bool",
      "value": "[endsWith('abcdef', 'e')]"
```

```
      }
    }
  }
```

The output from the preceding example with the default values is:

Name	Type	Value
startsTrue	Bool	True
startsCapTrue	Bool	True
startsFalse	Bool	False
endsTrue	Bool	True
endsCapTrue	Bool	True
endsFalse	Bool	False

first

`first(arg1)`

Returns the first character of the string, or first element of the array.

In Bicep, use the first function.

Parameters

Parameter	Required	Type	Description
arg1	Yes	array or string	The value to retrieve the first element or character.

Return value

A string of the first character, or the type (string, int, array, or object) of the first element in an array.

Examples

The following example shows how to use the first function with an array and string.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "arrayToTest": {
      "type": "array",
      "defaultValue": [ "one", "two", "three" ]
    }
  },
  "resources": [
  ],
  "outputs": {
    "arrayOutput": {
      "type": "string",
      "value": "[first(parameters('arrayToTest'))]"
    },
    "stringOutput": {
      "type": "string",
      "value": "[first('One Two Three')]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
arrayOutput	String	one
stringOutput	String	O

format

`format(formatString, arg1, arg2, ...)`

Creates a formatted string from input values.

In Bicep, use the format function.

Parameters

Parameter	Required	Type	Description
formatString	Yes	string	The composite format string.
arg1	Yes	string, integer, or boolean	The value to include in the formatted string.

Parameter	Required	Type	Description
more arguments	No	string, integer, or boolean	More values to include in the formatted string.

Remarks

Use this function to format a string in your template. It uses the same formatting options as the System.String.Format method in .NET.

Examples

The following example shows how to use the format function.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "greeting": {
      "type": "string",
      "defaultValue": "Hello"
    },
    "name": {
      "type": "string",
      "defaultValue": "User"
    },
    "numberToFormat": {
      "type": "int",
      "defaultValue": 8175133
    }
  },
  "resources": [
  ],
  "outputs": {
    "formatTest": {
      "type": "string",
      "value": "[format('{0}, {1}. Formatted number: {2:N0}', parameters('greeting'), parameters('name'), parameters('numberToFormat'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value

Name	Type	Value
formatTest	String	Hello, User. Formatted number: 8,175,133

guid

`guid(baseString, ...)`

Creates a value in the format of a globally unique identifier based on the values provided as parameters.

In Bicep, use the guid function.

Parameters

Parameter	Required	Type	Description
baseString	Yes	string	The value used in the hash function to create the GUID.
more parameters as needed	No	string	You can add as many strings as needed to create the value that specifies the level of uniqueness.

Remarks

This function is helpful when you need to create a value in the format of a globally unique identifier. You provide parameter values that limit the scope of uniqueness for the result. You can specify whether the name is unique down to subscription, resource group, or deployment.

The returned value isn't a random string, but rather the result of a hash function on the parameters. The returned value is 36 characters long. It isn't globally unique. To create a new GUID that isn't based on that hash value of the parameters, use the newGuid function.

The following examples show how to use guid to create a unique value for commonly used levels.

Unique scoped to subscription

JSON

```
"[guid(subscription().subscriptionId)]"
```

Unique scoped to resource group

JSON

```
"[guid(resourceGroup().id)]"
```

Unique scoped to deployment for a resource group

JSON

```
"[guid(resourceGroup().id, deployment().name)]"
```

Return value

A string containing 36 characters in the format of a globally unique identifier.

Examples

The following example returns results from `guid`:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {},
  "variables": {},
  "resources": [],
  "outputs": {
    "guidPerSubscription": {
      "type": "string",
      "value": "[guid(subscription().subscriptionId)]"
    },
    "guidPerResourceGroup": {
      "type": "string",
      "value": "[guid(resourceGroup().id)]"
    },
    "guidPerDeployment": {
      "type": "string",
      "value": "[guid(resourceGroup().id, deployment().name)]"
    }
  }
}
```

indexOf

`indexOf(stringToSearch, stringToFind)`

Returns the first position of a value within a string. The comparison is case-insensitive.

In Bicep, use the indexOf function.

Parameters

Parameter	Required	Type	Description
stringToSearch	Yes	string	The value that contains the item to find.
stringToFind	Yes	string	The value to find.

Return value

An integer that represents the position of the item to find. The value is zero-based. If the item isn't found, -1 is returned.

Examples

The following example shows how to use the `indexOf` and `lastIndexOf` functions:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [],
  "outputs": {
    "firstT": {
      "type": "int",
      "value": "[indexOf('test', 't')]"
    },
    "lastT": {
      "type": "int",
      "value": "[lastIndexOf('test', 't')]"
    },
    "firstString": {
      "type": "int",
      "value": "[indexOf('abcdef', 'CD')]"
    },
    "lastString": {
      "type": "int",
```

```
        "value": "[lastIndexOf('abcdef', 'AB')]"
      },
      "notFound": {
        "type": "int",
        "value": "[indexOf('abcdef', 'z')]"
      }
    }
  }
```

The output from the preceding example with the default values is:

Name	Type	Value
firstT	Int	0
lastT	Int	3
firstString	Int	2
lastString	Int	0
notFound	Int	-1

join

`join(inputArray, delimiter)`

Joins a string array into a single string, separated using a delimiter.

In Bicep, use the join function.

Parameters

Parameter	Required	Type	Description
inputArray	Yes	array of string	An array of string to join.
delimiter	Yes	The delimiter to use for splitting the string.	

Return value

A string.

Examples

The following example joins the input string array into strings delimited by using different delimiters.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "variables": {
    "arrayString": [
      "one",
      "two",
      "three"
    ]
  },
  "resources": [],
  "outputs": {
    "firstOutput": {
      "type": "string",
      "value": "[join(variables('arrayString'), ',')]"
    },
    "secondOutput": {
      "type": "string",
      "value": "[join(variables('arrayString'), ';')]"
    }
  }
}
```

The output from the preceding example is:

Name	Type	Value
firstOutput	String	"one,two,three"
secondOutput	String	"one;two;three"

json

`json(arg1)`

Converts a valid JSON string into a JSON data type. For more information, see json function.

In Bicep, use the json function.

last

`last(arg1)`

Returns last character of the string, or the last element of the array.

In Bicep, use the last function.

Parameters

Parameter	Required	Type	Description
arg1	Yes	array or string	The value to retrieve the last element or character.

Return value

A string of the last character, or the type (string, int, array, or object) of the last element in an array.

Examples

The following example shows how to use the `last` function with an array and string.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "arrayToTest": {
      "type": "array",
      "defaultValue": [ "one", "two", "three" ]
    }
  },
  "resources": [
  ],
  "outputs": {
    "arrayOutput": {
      "type": "string",
      "value": "[last(parameters('arrayToTest'))]"
    },
    "stringOutput": {
      "type": "string",
      "value": "[last('One Two Three')]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
arrayOutput	String	three
stringOutput	String	e

lastIndexOf

`lastIndexOf(stringToSearch, stringToFind)`

Returns the last position of a value within a string. The comparison is case-insensitive.

In Bicep, use the lastIndexOf function.

Parameters

Parameter	Required	Type	Description
stringToSearch	Yes	string	The value that contains the item to find.
stringToFind	Yes	string	The value to find.

Return value

An integer that represents the last position of the item to find. The value is zero-based. If the item isn't found, -1 is returned.

Examples

The following example shows how to use the `indexOf` and `lastIndexOf` functions:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [],
  "outputs": {
    "firstT": {
      "type": "int",
      "value": "[indexOf('test', 't')]"
    },
```

```
    "lastT": {
      "type": "int",
      "value": "[lastIndexOf('test', 't')]"
    },
    "firstString": {
      "type": "int",
      "value": "[indexOf('abcdef', 'CD')]"
    },
    "lastString": {
      "type": "int",
      "value": "[lastIndexOf('abcdef', 'AB')]"
    },
    "notFound": {
      "type": "int",
      "value": "[indexOf('abcdef', 'z')]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
firstT	Int	0
lastT	Int	3
firstString	Int	2
lastString	Int	0
notFound	Int	-1

length

`length(string)`

Returns the number of characters in a string, elements in an array, or root-level properties in an object.

In Bicep, use the length function.

Parameters

Parameter	Required	Type	Description

Parameter	Required	Type	Description
arg1	Yes	array, string, or object	The array to use for getting the number of elements, the string to use for getting the number of characters, or the object to use for getting the number of root-level properties.

Return value

An int.

Examples

The following example shows how to use the `length` function with an array and string:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "arrayToTest": {
      "type": "array",
      "defaultValue": [
        "one",
        "two",
        "three"
      ]
    },
    "stringToTest": {
      "type": "string",
      "defaultValue": "One Two Three"
    },
    "objectToTest": {
      "type": "object",
      "defaultValue": {
        "propA": "one",
        "propB": "two",
        "propC": "three",
        "propD": {
          "propD-1": "sub",
          "propD-2": "sub"
        }
      }
    }
  },
  "resources": [],
  "outputs": {
    "arrayLength": {
```

```
        "type": "int",
        "value": "[length(parameters('arrayToTest'))]"
      },
      "stringLength": {
        "type": "int",
        "value": "[length(parameters('stringToTest'))]"
      },
      "objectLength": {
        "type": "int",
        "value": "[length(parameters('objectToTest'))]"
      }
    }
  }
```

The output from the preceding example with the default values is:

Name	Type	Value
arrayLength	Int	3
stringLength	Int	13
objectLength	Int	4

newGuid

`newGuid()`

Returns a value in the format of a globally unique identifier. **This function can only be used in the default value for a parameter.**

In Bicep, use the newGuid function.

Remarks

You can only use this function within an expression for the default value of a parameter. Using this function anywhere else in a template returns an error. The function isn't allowed in other parts of the template because it returns a different value each time it's called. Deploying the same template with the same parameters wouldn't reliably produce the same results.

The newGuid function differs from the guid function because it doesn't take any parameters. When you call guid with the same parameter, it returns the same identifier each time. Use guid when you need to reliably generate the same GUID for a specific environment. Use newGuid when you need a different identifier each time, such as deploying resources to a test environment.

The newGuid function uses the Guid structure in the .NET Framework to generate the globally unique identifier.

If you use the option to redeploy an earlier successful deployment, and the earlier deployment includes a parameter that uses newGuid, the parameter isn't reevaluated. Instead, the parameter value from the earlier deployment is automatically reused in the rollback deployment.

In a test environment, you may need to repeatedly deploy resources that only live for a short time. Rather than constructing unique names, you can use newGuid with uniqueString to create unique names.

Be careful redeploying a template that relies on the newGuid function for a default value. When you redeploy and don't provide a value for the parameter, the function is reevaluated. If you want to update an existing resource rather than create a new one, pass in the parameter value from the earlier deployment.

Return value

A string containing 36 characters in the format of a globally unique identifier.

Examples

The following example shows a parameter with a new identifier.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "guidValue": {
      "type": "string",
      "defaultValue": "[newGuid()]"
    }
  },
  "resources": [
  ],
  "outputs": {
    "guidOutput": {
      "type": "string",
      "value": "[parameters('guidValue')]"
    }
  }
}
```

The output from the preceding example varies for each deployment but will be similar to:

Name	Type	Value
guidOutput	string	b76a51fc-bd72-4a77-b9a2-3c29e7d2e551

The following example uses the `newGuid` function to create a unique name for a storage account. This template might work for test environment where the storage account exists for a short time and isn't redeployed.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "guidValue": {
      "type": "string",
      "defaultValue": "[newGuid()]"
    }
  },
  "variables": {
    "storageName": "[concat('storage',
uniqueString(parameters('guidValue')))]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-04-01",
      "name": "[variables('storageName')]",
      "location": "West US",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "StorageV2",
      "properties": {}
    }
  ],
  "outputs": {
    "nameOutput": {
      "type": "string",
      "value": "[variables('storageName')]"
    }
  }
}
```

The output from the preceding example varies for each deployment but will be similar to:

Name	Type	Value
nameOutput	string	storagenziwvyru7uxie

padLeft

`padLeft(valueToPad, totalLength, paddingCharacter)`

Returns a right-aligned string by adding characters to the left until reaching the total specified length.

In Bicep, use the padLeft function.

Parameters

Parameter	Required	Type	Description
valueToPad	Yes	string or int	The value to right-align.
totalLength	Yes	int	The total number of characters in the returned string.
paddingCharacter	No	single character	The character to use for left-padding until the total length is reached. The default value is a space.

If the original string is longer than the number of characters to pad, no characters are added.

Return value

A string with at least the number of specified characters.

Examples

The following example shows how to pad the user-provided parameter value by adding the zero character until it reaches the total number of characters.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
```

```
      "testString": {
        "type": "string",
        "defaultValue": "123"
      }
    },
    "resources": [],
    "outputs": {
      "stringOutput": {
        "type": "string",
        "value": "[padLeft(parameters('testString'),10,'0')]"
      }
    }
  }
```

The output from the preceding example with the default values is:

Name	Type	Value
stringOutput	String	0000000123

replace

`replace(originalString, oldString, newString)`

Returns a new string with all instances of one string replaced by another string.

In Bicep, use the replace function.

Parameters

Parameter	Required	Type	Description
originalString	Yes	string	The value that has all instances of one string replaced by another string.
oldString	Yes	string	The string to be removed from the original string.
newString	Yes	string	The string to add in place of the removed string.

Return value

A string with the replaced characters.

Examples

The following example shows how to remove all dashes from the user-provided string, and how to replace part of the string with another string.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "testString": {
      "type": "string",
      "defaultValue": "123-123-1234"
    }
  },
  "resources": [],
  "outputs": {
    "firstOutput": {
      "type": "string",
      "value": "[replace(parameters('testString'),'-', '')]"
    },
    "secondOutput": {
      "type": "string",
      "value": "[replace(parameters('testString'),'1234', 'xxxx')]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
firstOutput	String	1231231234
secondOutput	String	123-123-xxxx

skip

`skip(originalValue, numberToSkip)`

Returns a string with all the characters after the specified number of characters, or an array with all the elements after the specified number of elements.

In Bicep, use the skip function.

Parameters

Parameter	Required	Type	Description
originalValue	Yes	array or string	The array or string to use for skipping.
numberToSkip	Yes	int	The number of elements or characters to skip. If this value is 0 or less, all the elements or characters in the value are returned. If it's larger than the length of the array or string, an empty array or string is returned.

Return value

An array or string.

Examples

The following example skips the specified number of elements in the array, and the specified number of characters in a string.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "testArray": {
      "type": "array",
      "defaultValue": [
        "one",
        "two",
        "three"
      ]
    },
    "elementsToSkip": {
      "type": "int",
      "defaultValue": 2
    },
    "testString": {
      "type": "string",
      "defaultValue": "one two three"
    },
    "charactersToSkip": {
      "type": "int",
      "defaultValue": 4
    }
  },
  "resources": [],
```

```
  "outputs": {
    "arrayOutput": {
      "type": "array",
      "value": "
[skip(parameters('testArray'),parameters('elementsToSkip'))]"
    },
    "stringOutput": {
      "type": "string",
      "value": "
[skip(parameters('testString'),parameters('charactersToSkip'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
arrayOutput	Array	["three"]
stringOutput	String	two three

split

`split(inputString, delimiter)`

Returns an array of strings that contains the substrings of the input string that are delimited by the specified delimiters.

In Bicep, use the split function.

Parameters

Parameter	Required	Type	Description
inputString	Yes	string	The string to split.
delimiter	Yes	string or array of strings	The delimiter to use for splitting the string.

Return value

An array of strings.

Examples

The following example splits the input string with a comma, and with either a comma or a semicolon.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "firstString": {
      "type": "string",
      "defaultValue": "one,two,three"
    },
    "secondString": {
      "type": "string",
      "defaultValue": "one;two,three"
    }
  },
  "variables": {
    "delimiters": [ ",", ";" ]
  },
  "resources": [],
  "outputs": {
    "firstOutput": {
      "type": "array",
      "value": "[split(parameters('firstString'),',')]"
    },
    "secondOutput": {
      "type": "array",
      "value": "[split(parameters('secondString'),variables('delimiters'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
firstOutput	Array	["one", "two", "three"]
secondOutput	Array	["one", "two", "three"]

startsWith

`startsWith(stringToSearch, stringToFind)`

Determines whether a string starts with a value. The comparison is case-insensitive.

In Bicep, use the startsWith function.

Parameters

Parameter	Required	Type	Description
stringToSearch	Yes	string	The value that contains the item to find.
stringToFind	Yes	string	The value to find.

Return value

True if the first character or characters of the string match the value; otherwise, **False**.

Examples

The following example shows how to use the `startsWith` and `endsWith` functions:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [],
  "outputs": {
    "startsTrue": {
      "type": "bool",
      "value": "[startsWith('abcdef', 'ab')]"
    },
    "startsCapTrue": {
      "type": "bool",
      "value": "[startsWith('abcdef', 'A')]"
    },
    "startsFalse": {
      "type": "bool",
      "value": "[startsWith('abcdef', 'e')]"
    },
    "endsTrue": {
      "type": "bool",
      "value": "[endsWith('abcdef', 'ef')]"
    },
    "endsCapTrue": {
      "type": "bool",
      "value": "[endsWith('abcdef', 'F')]"
    },
    "endsFalse": {
      "type": "bool",
      "value": "[endsWith('abcdef', 'e')]"
```

```
      }
    }
  }
```

The output from the preceding example with the default values is:

Name	Type	Value
startsTrue	Bool	True
startsCapTrue	Bool	True
startsFalse	Bool	False
endsTrue	Bool	True
endsCapTrue	Bool	True
endsFalse	Bool	False

string

`string(valueToConvert)`

Converts the specified value to a string.

In Bicep, use the string function.

Parameters

Parameter	Required	Type	Description
valueToConvert	Yes	Any	The value to convert to string. Any type of value can be converted, including objects and arrays.

Return value

A string of the converted value.

Examples

The following example shows how to convert different types of values to strings.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "testObject": {
      "type": "object",
      "defaultValue": {
        "valueA": 10,
        "valueB": "Example Text"
      }
    },
    "testArray": {
      "type": "array",
      "defaultValue": [
        "a",
        "b",
        "c"
      ]
    },
    "testInt": {
      "type": "int",
      "defaultValue": 5
    }
  },
  "resources": [],
  "outputs": {
    "objectOutput": {
      "type": "string",
      "value": "[string(parameters('testObject'))]"
    },
    "arrayOutput": {
      "type": "string",
      "value": "[string(parameters('testArray'))]"
    },
    "intOutput": {
      "type": "string",
      "value": "[string(parameters('testInt'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
objectOutput	String	{"valueA":10,"valueB":"Example Text"}
arrayOutput	String	["a","b","c"]
intOutput	String	5

substring

`substring(stringToParse, startIndex, length)`

Returns a substring that starts at the specified character position and contains the specified number of characters.

In Bicep, use the substring function.

Parameters

Parameter	Required	Type	Description
stringToParse	Yes	string	The original string from which the substring is extracted.
startIndex	No	int	The zero-based starting character position for the substring.
length	No	int	The number of characters for the substring. Must refer to a location within the string. Must be zero or greater. If omitted, the remainder of the string from the start position will be returned.

Return value

The substring. Or, an empty string if the length is zero.

Remarks

The function fails when the substring extends beyond the end of the string, or when length is less than zero. The following example fails with the error "The index and length parameters must refer to a location within the string. The index parameter: '0', the length parameter: '11', the length of the string parameter: '10'.".

JSON

```
"parameters": {
  "inputString": {
    "type": "string",
    "value": "1234567890"
  }
}, "variables": {
  "prefix": "[substring(parameters('inputString'), 0, 11)]"
}
```

Examples

The following example extracts a substring from a parameter.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "testString": {
      "type": "string",
      "defaultValue": "one two three"
    }
  },
  "resources": [],
  "outputs": {
    "substringOutput": {
      "type": "string",
      "value": "[substring(parameters('testString'), 4, 3)]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
substringOutput	String	two

take

`take(originalValue, numberToTake)`

Returns an array or string. An array has the specified number of elements from the start of the array. A string has the specified number of characters from the start of the string.

In Bicep, use the take function.

Parameters

Parameter	Required	Type	Description

Parameter	Required	Type	Description
originalValue	Yes	array or string	The array or string to take the elements from.
numberToTake	Yes	int	The number of elements or characters to take. If this value is 0 or less, an empty array or string is returned. If it's larger than the length of the given array or string, all the elements in the array or string are returned.

Return value

An array or string.

Examples

The following example takes the specified number of elements from the array, and characters from a string.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "testArray": {
      "type": "array",
      "defaultValue": [
        "one",
        "two",
        "three"
      ]
    },
    "elementsToTake": {
      "type": "int",
      "defaultValue": 2
    },
    "testString": {
      "type": "string",
      "defaultValue": "one two three"
    },
    "charactersToTake": {
      "type": "int",
      "defaultValue": 2
    }
  },
  "resources": [],
```

```
  "outputs": {
    "arrayOutput": {
      "type": "array",
      "value": "
[take(parameters('testArray'),parameters('elementsToTake'))]"
    },
    "stringOutput": {
      "type": "string",
      "value": "
[take(parameters('testString'),parameters('charactersToTake'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
arrayOutput	Array	["one", "two"]
stringOutput	String	on

toLower

`toLower(stringToChange)`

Converts the specified string to lower case.

In Bicep, use the toLower function.

Parameters

Parameter	Required	Type	Description
stringToChange	Yes	string	The value to convert to lower case.

Return value

The string converted to lower case.

Examples

The following example converts a parameter value to lower case and to upper case.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "testString": {
      "type": "string",
      "defaultValue": "One Two Three"
    }
  },
  "resources": [],
  "outputs": {
    "toLowerOutput": {
      "type": "string",
      "value": "[toLower(parameters('testString'))]"
    },
    "toUpperOutput": {
      "type": "string",
      "value": "[toUpper(parameters('testString'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
toLowerOutput	String	one two three
toUpperOutput	String	ONE TWO THREE

toUpper

`toUpper(stringToChange)`

Converts the specified string to upper case.

In Bicep, use the toUpper function.

Parameters

Parameter	Required	Type	Description
stringToChange	Yes	string	The value to convert to upper case.

Return value

The string converted to upper case.

Examples

The following example converts a parameter value to lower case and to upper case.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "testString": {
      "type": "string",
      "defaultValue": "One Two Three"
    }
  },
  "resources": [],
  "outputs": {
    "toLowerOutput": {
      "type": "string",
      "value": "[toLower(parameters('testString'))]"
    },
    "toUpperOutput": {
      "type": "string",
      "value": "[toUpper(parameters('testString'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
toLowerOutput	String	one two three
toUpperOutput	String	ONE TWO THREE

trim

`trim(stringToTrim)`

Removes all leading and trailing white-space characters from the specified string.

In Bicep, use the trim function.

Parameters

Parameter	Required	Type	Description
stringToTrim	Yes	string	The value to trim.

Return value

The string without leading and trailing white-space characters.

Examples

The following example trims the white-space characters from the parameter.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "testString": {
      "type": "string",
      "defaultValue": "    one two three    "
    }
  },
  "resources": [],
  "outputs": {
    "return": {
      "type": "string",
      "value": "[trim(parameters('testString'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
return	String	one two three

uniqueString

`uniqueString(baseString, ...)`

Creates a deterministic hash string based on the values provided as parameters.

In Bicep, use the uniqueString function.

Parameters

Parameter	Required	Type	Description
baseString	Yes	string	The value used in the hash function to create a unique string.
more parameters as needed	No	string	You can add as many strings as needed to create the value that specifies the level of uniqueness.

Remarks

This function is helpful when you need to create a unique name for a resource. You provide parameter values that limit the scope of uniqueness for the result. You can specify whether the name is unique down to subscription, resource group, or deployment.

The returned value isn't a random string, but rather the result of a hash function. The returned value is 13 characters long. It isn't globally unique. You may want to combine the value with a prefix from your naming convention to create a name that is meaningful. The following example shows the format of the returned value. The actual value varies by the provided parameters.

`tcvhiyu5h2o5o`

The following examples show how to use `uniqueString` to create a unique value for commonly used levels.

Unique scoped to subscription

JSON

```
"[uniqueString(subscription().subscriptionId)]"
```

Unique scoped to resource group

JSON

```
"[uniqueString(resourceGroup().id)]"
```

Unique scoped to deployment for a resource group

JSON

```
"[uniqueString(resourceGroup().id, deployment().name)]"
```

The following example shows how to create a unique name for a storage account based on your resource group. Inside the resource group, the name isn't unique if constructed the same way.

JSON

```
"resources": [{
  "name": "[concat('storage', uniqueString(resourceGroup().id))]",
  "type": "Microsoft.Storage/storageAccounts",
  ...
```

If you need to create a new unique name each time you deploy a template, and don't intend to update the resource, you can use the utcNow function with `uniqueString`. You could use this approach in a test environment. For an example, see utcNow.

Return value

A string containing 13 characters.

Examples

The following example returns results from `uniquestring`:

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [],
  "outputs": {
    "uniqueRG": {
      "type": "string",
      "value": "[uniqueString(resourceGroup().id)]"
    },
    "uniqueDeploy": {
      "type": "string",
      "value": "[uniqueString(resourceGroup().id, deployment().name)]"
    }
  }
}
```

uri

`uri(baseUri, relativeUri)`

Creates an absolute URI by combining the baseUri and the relativeUri string.

In Bicep, use the uri function.

Parameters

Parameter	Required	Type	Description
baseUri	Yes	string	The base uri string. Take care to observe the behavior about the handling of the trailing slash (`/`), as described following this table.
relativeUri	Yes	string	The relative uri string to add to the base uri string.

- If **baseUri** ends in a trailing slash, the result is **baseUri** followed by **relativeUri**.
- If **baseUri** doesn't end in a trailing slash one of two things happens.
  - If **baseUri** has no slashes at all (aside from the `//` near the front) the result is **baseUri** followed by **relativeUri**.
  - If **baseUri** has some slashes, but doesn't end with a slash, everything from the last slash onward is removed from **baseUri** and the result is **baseUri** followed by **relativeUri**.

Here are some examples:

```
uri('http://contoso.org/firstpath', 'myscript.sh') ->
http://contoso.org/myscript.sh
uri('http://contoso.org/firstpath/', 'myscript.sh') ->
http://contoso.org/firstpath/myscript.sh
uri('http://contoso.org/firstpath/azuredeploy.json', 'myscript.sh') ->
http://contoso.org/firstpath/myscript.sh
uri('http://contoso.org/firstpath/azuredeploy.json/', 'myscript.sh') ->
http://contoso.org/firstpath/azuredeploy.json/myscript.sh
```

For complete details, the **baseUri** and **relativeUri** parameters are resolved as specified in RFC 3986, section 5.

Return value

A string representing the absolute URI for the base and relative values.

Examples

The following example shows how to construct a link to a nested template based on the value of the parent template.

JSON

```
"templateLink": "[uri(deployment().properties.templateLink.uri,
'nested/azuredeploy.json')]"
```

The following example template shows how to use `uri`, `uriComponent`, and `uriComponentToString`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "variables": {
    "uriFormat": "[uri('http://contoso.com/resources/',
'nested/azuredeploy.json')]",
    "uriEncoded": "[uriComponent(variables('uriFormat'))]"
  },
  "resources": [
  ],
  "outputs": {
    "uriOutput": {
      "type": "string",
      "value": "[variables('uriFormat')]"
    },
    "componentOutput": {
      "type": "string",
      "value": "[variables('uriEncoded')]"
    },
    "toStringOutput": {
      "type": "string",
      "value": "[uriComponentToString(variables('uriEncoded'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
uriOutput	String	`http://contoso.com/resources/nested/azuredeploy.json`
componentOutput	String	`http%3A%2F%2Fcontoso.com%2Fresources%2Fnested%2Fazuredeploy.json`
toStringOutput	String	`http://contoso.com/resources/nested/azuredeploy.json`

uriComponent

`uricomponent(stringToEncode)`

Encodes a URI.

In Bicep, use the uriComponent function.

Parameters

Parameter	Required	Type	Description
stringToEncode	Yes	string	The value to encode.

Return value

A string of the URI encoded value.

Examples

The following example template shows how to use `uri`, `uriComponent`, and `uriComponentToString`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "variables": {
    "uriFormat": "[uri('http://contoso.com/resources/', 'nested/azuredeploy.json')]",
    "uriEncoded": "[uriComponent(variables('uriFormat'))]"
  },
  "resources": [
  ],
  "outputs": {
```

```
    "uriOutput": {
      "type": "string",
      "value": "[variables('uriFormat')]"
    },
    "componentOutput": {
      "type": "string",
      "value": "[variables('uriEncoded')]"
    },
    "toStringOutput": {
      "type": "string",
      "value": "[uriComponentToString(variables('uriEncoded'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
uriOutput	String	`http://contoso.com/resources/nested/azuredeploy.json`
componentOutput	String	`http%3A%2F%2Fcontoso.com%2Fresources%2Fnested%2Fazuredeploy.json`
toStringOutput	String	`http://contoso.com/resources/nested/azuredeploy.json`

uriComponentToString

`uriComponentToString(uriEncodedString)`

Returns a string of a URI encoded value.

In Bicep, use the uriComponentToString function.

Parameters

Parameter	Required	Type	Description
uriEncodedString	Yes	string	The URI encoded value to convert to a string.

Return value

A decoded string of URI encoded value.

Examples

The following example shows how to use `uri`, `uriComponent`, and `uriComponentToString`.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "variables": {
    "uriFormat": "[uri('http://contoso.com/resources/', 'nested/azuredeploy.json')]",
    "uriEncoded": "[uriComponent(variables('uriFormat'))]"
  },
  "resources": [
  ],
  "outputs": {
    "uriOutput": {
      "type": "string",
      "value": "[variables('uriFormat')]"
    },
    "componentOutput": {
      "type": "string",
      "value": "[variables('uriEncoded')]"
    },
    "toStringOutput": {
      "type": "string",
      "value": "[uriComponentToString(variables('uriEncoded'))]"
    }
  }
}
```

The output from the preceding example with the default values is:

Name	Type	Value
uriOutput	String	`http://contoso.com/resources/nested/azuredeploy.json`
componentOutput	String	`http%3A%2F%2Fcontoso.com%2Fresources%2Fnested%2Fazuredeploy.json`
toStringOutput	String	`http://contoso.com/resources/nested/azuredeploy.json`

Next steps

- For a description of the sections in an ARM template, see Understand the structure and syntax of ARM templates.
- To merge multiple templates, see Using linked and nested templates when deploying Azure resources.

- To iterate a specified number of times when creating a type of resource, see Resource iteration in ARM templates.
- To see how to deploy the template you've created, see Deploy resources with ARM templates and Azure PowerShell.

Azure Resource Manager

Article • 01/31/2023

Azure Resource Manager enables you to deploy and manage the infrastructure for your Azure solutions. You organize related resources in resource groups, and deploy your resources with JSON templates. For an introduction to deploying and managing resources with Resource Manager, see Azure Resource Manager overview.

REST operation groups

Operation group	Description
Changes	Provides operations to get details about the changes that happen to deployed resources.
Deployment Operations	Provides operations to get details about the operations that happen during a deployment.
Deployment Scripts	Provides operations for working with scripts in a template.
Deployments	Provides operations to work with deployments, including deploying a template, exporting a template, and getting information about a deployment.
Features	Provides operations for users to access preview features from resource providers.
Management Locks	Provides operations to prevent users in your organization from deleting or modifying certain resources.
Providers	Provides operations to work with resource providers that support resources.
Resource Groups	Provides operations to create, delete, and access resource groups.
Resource Links	Provides operations to link resources that have logical relationships.
Resources	Provides operations to work with deployed resources.
Snapshots	Provides operations to get details about the historical snapshots on deployed resources.
Subscriptions	Provides operations to view the subscriptions for your Azure Active Directory tenant.

Operation group	Description
Tags	Provides operations to apply tag names and values to resources for organizing your resources.
Template Specs	Provides operations for working with templates that you can share in your organization.
Tenants	Provides operations to view the Azure Active Directory tenants for your account.

See also

- What are Azure Resource Manager templates?
- Template Syntax

Az.Resources

Reference

This topic displays help topics for the Azure Resource Manager Cmdlets.

Active Directory

Add-AzADAppPermission	Adds an API permission.
Add-AzADGroupMember	Adds member to group.
Get-AzADAppCredential	Lists key credentials and password credentials for an application.
Get-AzADAppFederatedCredential	Get federatedIdentityCredentials by Id from applications.
Get-AzADApplication	Lists entities from applications or get entity from applications by key
Get-AzADAppPermission	Lists API permissions the application has requested.
Get-AzADGroup	Lists entities from groups or get entity from groups by key
Get-AzADGroupMember	Lists members from group.
Get-AzADOrganization	Retrieve a list of organization objects.
Get-AzADServicePrincipal	Lists entities from service principals or get entity from service principals by key
Get-AzADSpCredential	Lists key credentials and password credentials for an service principal.
Get-AzADUser	Lists entities from users or get entity from users by key
New-AzADAppCredential	Creates key credentials or password credentials for an application.
New-AzADAppFederatedCredential	Create federatedIdentityCredential for applications.
New-AzADApplication	Adds new entity to applications
New-AzADGroup	Adds new entity to groups
New-AzADServicePrincipal	Adds new entity to servicePrincipals
New-AzADSpCredential	Creates key credentials or password credentials for an service principal.

New-AzADUser	Adds new entity to users
Remove-AzADAppCredential	Removes key credentials or password credentials for an application.
Remove-AzADAppFederatedCredential	Delete navigation property federatedIdentityCredentials for applications
Remove-AzADApplication	Deletes entity from applications
Remove-AzADAppPermission	Removes an API permission.
Remove-AzADGroup	Deletes entity from groups.
Remove-AzADGroupMember	Deletes member from group Users, contacts, and groups that are members of this group. HTTP Methods: GET (supported for all groups), POST (supported for security groups and mail-enabled security groups), DELETE (supported only for security groups) Read-only. Nullable. Supports $expand.
Remove-AzADServicePrincipal	Deletes entity from service principal.
Remove-AzADSpCredential	Removes key credentials or password credentials for an service principal.
Remove-AzADUser	Deletes entity from users.
Update-AzADAppFederatedCredential	Update the navigation property federatedIdentityCredentials in applications
Update-AzADApplication	Updates entity in applications
Update-AzADGroup	Update entity in groups
Update-AzADServicePrincipal	Updates entity in service principal
Update-AzADUser	Updates entity in users

Managed Applications

Get-AzManagedApplication	Gets managed applications
Get-AzManagedApplicationDefinition	Gets managed application definitions
New-AzManagedApplication	Creates an Azure managed application.
New-AzManagedApplicationDefinition	Creates a managed application definition.
Remove-AzManagedApplication	Removes a managed application

Remove-AzManagedApplicationDefinition	Removes a managed application definition
Set-AzManagedApplication	Updates managed application
Set-AzManagedApplicationDefinition	Updates managed application definition

Policy

Get-AzPolicyAlias	Get-AzPolicyAlias retrieves and outputs Azure provider resource types that have aliases defined and match the given parameter values. If no parameters are provided, all provider resource types that contain an alias will be output. The -ListAvailable switch modifies this behavior by listing all matching resource types including those without aliases.
Get-AzPolicyAssignment	Gets policy assignments.
Get-AzPolicyDefinition	Gets policy definitions.
Get-AzPolicyExemption	Gets policy exemptions.
Get-AzPolicySetDefinition	Gets policy set definitions.
Get-AzRoleManagementPolicy	Get the specified role management policy for a resource scope
Get-AzRoleManagementPolicyAssignment	Get the specified role management policy assignment for a resource scope
New-AzPolicyAssignment	Creates a policy assignment.
New-AzPolicyDefinition	Creates a policy definition.
New-AzPolicyExemption	Creates a policy exemption.
New-AzPolicySetDefinition	Creates a policy set definition.
New-AzRoleManagementPolicyAssignment	Create a role management policy assignment
Remove-AzPolicyAssignment	Removes a policy assignment.
Remove-AzPolicyDefinition	Removes a policy definition.
Remove-AzPolicyExemption	Removes a policy exemption.
Remove-AzPolicySetDefinition	Removes a policy set definition.

Remove-AzRoleManagementPolicy	Delete a role management policy
Remove-AzRoleManagementPolicyAssignment	Delete a role management policy assignment
Set-AzPolicyAssignment	Modifies a policy assignment.
Set-AzPolicyDefinition	Modifies a policy definition.
Set-AzPolicyExemption	Modifies a policy exemption.
Set-AzPolicySetDefinition	Modifies a policy set definition
Update-AzRoleManagementPolicy	Update a role management policy

Resources

Export-AzResourceGroup	Captures a resource group as a template and saves it to a file.
Export-AzTemplateSpec	Exports a Template Spec to the local filesystem
Get-AzDenyAssignment	Lists Azure RBAC deny assignments at the specified scope. By default it lists all deny assignments in the selected Azure subscription. Use respective parameters to list deny assignments to a specific user, or to list deny assignments on a specific resource group or resource. The cmdlet may call below Microsoft Graph API according to input parameters: • GET /directoryObjects/{id} • POST /directoryObjects/getByIds
Get-AzDeployment	Get deployment
Get-AzDeploymentOperation	Get deployment operation
Get-AzDeploymentScript	Gets or lists deployment scripts.
Get-AzDeploymentScriptLog	Gets the log of a deployment script execution.
Get-AzDeploymentWhatIfResult	Gets a template What-If result for a deployment at subscription scope.
Get-AzLocation	Gets all locations and the supported resource providers for each location.

Get-AzManagementGroup	Gets Management Group(s)
Get-AzManagementGroupDeployment	Get deployment at a management group
Get-AzManagementGroupDeploymentOperation	Get deployment operation for management group deployment
Get-AzManagementGroupDeploymentWhatIfResult	Gets a template What-If result for a deployment at management group scope.
Get-AzManagementGroupEntity	Lists all Entities under the current Tenant
Get-AzManagementGroupHierarchySetting	Gets the Hierarchy Settings under the current tenant
Get-AzManagementGroupNameAvailability	Checks if the Management Group name is available in the Tenant and a valid name.
Get-AzManagementGroupSubscription	Gets the details of Subscription(s) under a Management Group.
Get-AzPrivateLinkAssociation	Gets all the Azure Resource Management Private Link Association(s).
Get-AzProviderFeature	Gets information about Azure provider features.
Get-AzProviderOperation	Gets the operations for an Azure resource provider that are securable using Azure RBAC.
Get-AzProviderPreviewFeature	Gets a feature registration in your account.
Get-AzResource	Gets resources.
Get-AzResourceGroup	Gets resource groups.
Get-AzResourceGroupDeployment	Gets the deployments in a resource group.
Get-AzResourceGroupDeploymentOperation	Gets the resource group deployment operation
Get-AzResourceGroupDeploymentWhatIfResult	Gets a template What-If result for a deployment at resource group scope.
Get-AzResourceLock	Gets a resource lock.
Get-AzResourceManagementPrivateLink	Gets Azure Resource Management Private Link(s)
Get-AzResourceProvider	Gets a resource provider.
Get-AzRoleAssignment	Lists Azure RBAC role assignments at the specified scope. By default it lists all role assignments in the selected Azure subscription. Use respective parameters to list assignments to

	a specific user, or to list assignments on a specific resource group or resource. The cmdlet may call below Microsoft Graph API according to input parameters: • GET /users/{id} • GET /servicePrincipals/{id} • GET /groups/{id} • GET /directoryObjects/{id} • POST /directoryObjects/getByIds Please notice that this cmdlet will mark `ObjectType` as `Unknown` in output if the object of role assignment is not found or current account has insufficient privileges to get object type.
Get-AzRoleAssignmentSchedule	Get the specified role assignment schedule for a resource scope
Get-AzRoleAssignmentScheduleInstance	Gets the specified role assignment schedule instance.
Get-AzRoleAssignmentScheduleRequest	Get the specified role assignment schedule request.
Get-AzRoleDefinition	Lists all Azure RBAC roles that are available for assignment.
Get-AzRoleEligibilitySchedule	Get the specified role eligibility schedule for a resource scope
Get-AzRoleEligibilityScheduleInstance	Gets the specified role eligibility schedule instance.
Get-AzRoleEligibilityScheduleRequest	Get the specified role eligibility schedule request.
Get-AzRoleEligibleChildResource	Get the child resources of a resource on which user has eligible access
Get-AzTag	Gets predefined Azure tags | Gets the entire set of tags on a resource or subscription.
Get-AzTemplateSpec	Gets or lists Template Specs
Get-AzTenantBackfillStatus	Get the current Tenant Backfill Subscription Status
Get-AzTenantDeployment	Get deployment at tenant scope
Get-AzTenantDeploymentOperation	Get deployment operation for deployment at tenant scope

Get-AzTenantDeploymentWhatIfResult	Gets a template What-If result for a deployment at tenant scope.
Invoke-AzResourceAction	Invokes an action on a resource.
Move-AzResource	Moves a resource to a different resource group or subscription.
New-AzDeployment	Create a deployment at the current subscription scope.
New-AzManagementGroup	Creates a Management Group
New-AzManagementGroupDeployment	Create a deployment at a management group
New-AzManagementGroupHierarchySetting	Creates Hierarchy Settings under the current tenant
New-AzManagementGroupSubscription	Adds a Subscription to a Management Group.
New-AzPrivateLinkAssociation	Creates the Azure Resource Management Private Link Association.
New-AzResource	Creates a resource.
New-AzResourceGroup	Creates an Azure resource group.
New-AzResourceGroupDeployment	Adds an Azure deployment to a resource group.
New-AzResourceLock	Creates a resource lock.
New-AzResourceManagementPrivateLink	Create Azure Resource Management Private Link
New-AzRoleAssignment	Assigns the specified RBAC role to the specified principal, at the specified scope. The cmdlet may call below Microsoft Graph API according to input parameters: • GET /users/{id} • GET /servicePrincipals/{id} • GET /groups/{id} • GET /directoryObjects/{id} Please notice that this cmdlet will mark `ObjectType` as `Unknown` in output if the object of role assignment is not found or current account has insufficient privileges to get object type.
New-AzRoleAssignmentScheduleRequest	Creates a role assignment schedule request.

New-AzRoleDefinition	Creates a custom role in Azure RBAC. Provide either a JSON role definition file or a PSRoleDefinition object as input. First, use the Get-AzRoleDefinition command to generate a baseline role definition object. Then, modify its properties as required. Finally, use this command to create a custom role using role definition.
New-AzRoleEligibilityScheduleRequest	Creates a role eligibility schedule request.
New-AzTag	Creates a predefined Azure tag or adds values to an existing tag | Creates or updates the entire set of tags on a resource or subscription.
New-AzTemplateSpec	Creates a new Template Spec.
New-AzTenantDeployment	Create a deployment at tenant scope
Publish-AzBicepModule	Publishes a Bicep file to a registry.
Register-AzProviderFeature	Registers an Azure provider feature in your current subscription context.
Register-AzProviderPreviewFeature	Creates a feature registration in your account.
Register-AzResourceProvider	Registers a resource provider.
Remove-AzDeployment	Removes a deployment and any associated operations
Remove-AzDeploymentScript	Removes a deployment script and its associated resources.
Remove-AzManagementGroup	Removes a Management Group
Remove-AzManagementGroupDeployment	Removes a deployment at a management group and any associated operations
Remove-AzManagementGroupHierarchySetting	Deletes all Hierarchy Settings under the current tenant
Remove-AzManagementGroupSubscription	Removes a Subscription from a Management Group.
Remove-AzPrivateLinkAssociation	Delete a specific azure private link association.
Remove-AzResource	Removes a resource.
Remove-AzResourceGroup	Removes a resource group.
Remove-AzResourceGroupDeployment	Removes a resource group deployment and any associated operations.

Remove-AzResourceLock	Removes a resource lock.
Remove-AzResourceManagementPrivateLink	Deletes the Resource Manangement Private Link.
Remove-AzRoleAssignment	Removes a role assignment to the specified principal who is assigned to a particular role at a particular scope. The cmdlet may call below Microsoft Graph API according to input parameters: • GET /users/{id} • GET /servicePrincipals/{id} • GET /groups/{id} • GET /directoryObjects/{id} • POST /directoryObjects/getByIds Please notice that this cmdlet will mark `ObjectType` as `Unknown` in output if the object of role assignment is not found or current account has insufficient privileges to get object type.
Remove-AzRoleDefinition	Deletes a custom role in Azure RBAC. The role to be deleted is specified using the Id property of the role. Delete will fail if there are existing role assignments made to the custom role.
Remove-AzTag	Deletes predefined Azure tags or values | Deletes the entire set of tags on a resource or subscription.
Remove-AzTemplateSpec	Removes a Template Spec
Remove-AzTenantDeployment	Removes a deployment at tenant scope and any associated operations
Save-AzDeploymentScriptLog	Saves the log of a deployment script execution to disk.
Save-AzDeploymentTemplate	Saves a deployment template to a file.
Save-AzManagementGroupDeploymentTemplate	Saves a deployment template to a file.
Save-AzResourceGroupDeploymentTemplate	Saves a resource group deployment template to a file.
Save-AzTenantDeploymentTemplate	Saves a deployment template to a file.
Set-AzResource	Modifies a resource.

Set-AzResourceGroup	Modifies a resource group.
Set-AzResourceLock	Modifies a resource lock.
Set-AzRoleAssignment	Update an existing Role Assignment. The cmdlet may call below Microsoft Graph API according to input parameters: • GET /users/{id} • GET /servicePrincipals/{id} • GET /groups/{id} • GET /directoryObjects/{id} • POST /directoryObjects/getByIds Please notice that this cmdlet will mark `ObjectType` as `Unknown` in output if the object of role assignment is not found or current account has insufficient privileges to get object type.
Set-AzRoleDefinition	Modifies a custom role in Azure RBAC. Provide the modified role definition either as a JSON file or as a PSRoleDefinition. First, use the Get-AzRoleDefinition command to retrieve the custom role that you wish to modify. Then, modify the properties that you wish to change. Finally, save the role definition using this command.
Set-AzTemplateSpec	Modifies a Template Spec.
Start-AzTenantBackfill	Starts backfilling subscriptions for the current Tenant
Stop-AzDeployment	Cancel a running deployment
Stop-AzManagementGroupDeployment	Cancel a running deployment at a management group
Stop-AzResourceGroupDeployment	Cancels a resource group deployment.
Stop-AzRoleAssignmentScheduleRequest	Cancels a pending role assignment schedule request.
Stop-AzRoleEligibilityScheduleRequest	Cancels a pending role eligibility schedule request.
Stop-AzTenantDeployment	Cancel a running deployment at tenant scope
Test-AzDeployment	Validates a deployment.
Test-AzManagementGroupDeployment	Validates a deployment at a management group.

Test-AzResourceGroupDeployment	Validates a resource group deployment.
Test-AzTenantDeployment	Validates a deployment at tenant scope.
Unregister-AzProviderFeature	Unregisters an Azure provider feature in your account.
Unregister-AzProviderPreviewFeature	Removes a feature registration from your account.
Unregister-AzResourceProvider	Unregisters a resource provider.
Update-AzManagementGroup	Updates a Management Group
Update-AzManagementGroupHierarchySetting	Updates Hierarchy Settings under the current tenant
Update-AzTag	Selectively updates the set of tags on a resource or subscription.

az resource

Reference

Manage Azure resources.

Commands

az resource create	Create a resource.
az resource delete	Delete a resource.
az resource invoke-action	Invoke an action on the resource.
az resource link	Manage links between resources.
az resource link create	Create a new link between resources.
az resource link delete	Delete a link between resources.
az resource link list	List resource links.
az resource link show	Gets a resource link with the specified ID.
az resource link update	Update link between resources.
az resource list	List resources.
az resource lock	Manage Azure resource level locks.
az resource lock create	Create a resource-level lock.
az resource lock delete	Delete a resource-level lock.
az resource lock list	List lock information in the resource-level.
az resource lock show	Show the details of a resource-level lock.
az resource lock update	Update a resource-level lock.
az resource move	Moves resources from one resource group to another(can be under different subscription).
az resource patch	Update a resource by PATCH request.
az resource show	Get the details of a resource.
az resource tag	Tag a resource.
az resource update	Update a resource by PUT request.

az resource wait	Place the CLI in a waiting state until a condition of a resources is met.

az resource create

Edit

Create a resource.

Azure CLI

```
az resource create --properties
                   [--api-version]
                   [--id]
                   [--is-full-object]
                   [--latest-include-preview]
                   [--location]
                   [--name]
                   [--namespace]
                   [--parent]
                   [--resource-group]
                   [--resource-type]
```

Examples

Create an API app by providing a full JSON configuration.

Azure CLI

```
az resource create -g myRG -n myApiApp --resource-type Microsoft.web/sites \
    --is-full-object --properties "{ \"kind\": \"api\", \"location\": \
        \"West US\", \"properties\": { \"serverFarmId\": \
            \"/subscriptions/{SubID}/resourcegroups/{ResourceGroup} \
                /providers/Microsoft.Web/serverfarms/{ServicePlan}\" } }"
```

Create a resource by loading JSON configuration from a file.

Azure CLI

```
az resource create -g myRG -n myApiApp --resource-type Microsoft.web/sites -
-is-full-object --properties @jsonConfigFile
```

Create a web app with the minimum required configuration information.

Azure CLI

```
az resource create -g myRG -n myWeb --resource-type Microsoft.web/sites \
    --properties "{
\"serverFarmId\":\"/subscriptions/{SubID}/resourcegroups/ \
        {ResourceGroup}/providers/Microsoft.Web/serverfarms/{ServicePlan}\"
}"
```

Create a resource by using the latest api-version whether this version is a preview version.

Azure CLI

```
az resource create -g myRG -n myApiApp --resource-type Microsoft.web/sites -
-is-full-object --properties @jsonConfigFile --latest-include-preview
```

Create a site extension to a web app

Azure CLI

```
az resource create -g myRG --api-version "2018-02-01" \
    --name "
{sitename+slot}/siteextensions/Contrast.NetCore.Azure.SiteExtension"  \
        --resource-type Microsoft.Web/sites/siteextensions --is-full-object
\
            --properties "{ \"id\":
\"Contrast.NetCore.Azure.SiteExtension\", \
                \"location\": \"West US\", \"version\": \"1.9.0\" }"
```

Required Parameters

`--properties -p`

A JSON-formatted string containing resource properties.

Optional Parameters

`--api-version`

The api version of the resource (omit for the latest stable version).

`--id`

Resource ID.

`--is-full-object`

Indicate that the properties object includes other options such as location, tags, sku, and/or plan.
default value: False

`--latest-include-preview -v`

Indicate that the latest api-version will be used regardless of whether it is preview version (like 2020-01-01-preview) or not. For example, if the supported api-version of resource provider is 2020-01-01-preview and 2019-01-01: when passing in this parameter it will take the latest version 2020-01-01-preview, otherwise it will take the latest stable version 2019-01-01 without passing in this parameter.
default value: False

`--location -l`

Location. Values from: `az account list-locations`. You can configure the default location using `az configure --defaults location=<location>`.

`--name -n`

The resource name. (Ex: myC).

`--namespace`

Provider namespace (Ex: 'Microsoft.Provider').

`--parent`

The parent path (Ex: 'resA/myA/resB/myB').

`--resource-group -g`

Name of resource group. You can configure the default group using `az configure --defaults group=<name>`.

`--resource-type`

The resource type (Ex: 'resC'). Can also accept namespace/type format (Ex: 'Microsoft.Provider/resC').

Global Parameters

`--debug`

Increase logging verbosity to show all debug logs.

`--help -h`

Show this help message and exit.

`--only-show-errors`

Only show errors, suppressing warnings.

`--output -o`

Output format.

`--query`

JMESPath query string. See http://jmespath.org/ for more information and examples.

`--subscription`

Name or ID of subscription. You can configure the default subscription using `az account set -s NAME_OR_ID`.

`--verbose`

Increase logging verbosity. Use --debug for full debug logs.

az resource delete

Edit

Delete a resource.

Azure CLI

```
az resource delete [--api-version]
                   [--ids]
                   [--latest-include-preview]
                   [--name]
                   [--namespace]
```

```
                    [--no-wait]
                    [--parent]
                    [--resource-group]
                    [--resource-type]
```

Examples

Delete a virtual machine named 'MyVm'.

Azure CLI

```
az resource delete -g MyResourceGroup -n MyVm --resource-type
"Microsoft.Compute/virtualMachines"
```

Delete a web app using a resource identifier.

Azure CLI

```
az resource delete --ids /subscriptions/0b1f6471-1bf0-4dda-aec3-
111111111111/resourceGroups/MyResourceGroup/providers/Microsoft.Web/sites/My
Webapp
```

Delete a subnet using a resource identifier.

Azure CLI

```
az resource delete --ids /subscriptions/0b1f6471-1bf0-4dda-aec3-
111111111111/resourceGroups/MyResourceGroup/providers/Microsoft.Network/virt
ualNetworks/MyVnet/subnets/MySubnet
```

Delete a virtual machine named 'MyVm' by using the latest api-version whether this version is a preview version.

Azure CLI

```
az resource delete -g MyResourceGroup -n MyVm --resource-type
"Microsoft.Compute/virtualMachines" --latest-include-preview
```

Optional Parameters

`--api-version`

The api version of the resource (omit for the latest stable version).

`--ids`

One or more resource IDs (space-delimited). If provided, no other "Resource Id" arguments should be specified.

`--latest-include-preview -v`

Indicate that the latest api-version will be used regardless of whether it is preview version (like 2020-01-01-preview) or not. For example, if the supported api-version of resource provider is 2020-01-01-preview and 2019-01-01: when passing in this parameter it will take the latest version 2020-01-01-preview, otherwise it will take the latest stable version 2019-01-01 without passing in this parameter.
default value: False

`--name -n`

The resource name. (Ex: myC).

`--namespace`

Provider namespace (Ex: 'Microsoft.Provider').

`--no-wait`

Do not wait for the long-running operation to finish.
default value: False

`--parent`

The parent path (Ex: 'resA/myA/resB/myB').

`--resource-group -g`

Name of resource group. You can configure the default group using `az configure --defaults group=<name>`.

`--resource-type`

The resource type (Ex: 'resC'). Can also accept namespace/type format (Ex: 'Microsoft.Provider/resC').

Global Parameters

`--debug`

Increase logging verbosity to show all debug logs.

`--help -h`

Show this help message and exit.

`--only-show-errors`

Only show errors, suppressing warnings.

`--output -o`

Output format.

`--query`

JMESPath query string. See http://jmespath.org/ for more information and examples.

`--subscription`

Name or ID of subscription. You can configure the default subscription using `az account set -s NAME_OR_ID`.

`--verbose`

Increase logging verbosity. Use --debug for full debug logs.

az resource invoke-action

Edit

Invoke an action on the resource.

A list of possible actions corresponding to a resource can be found at https://docs.microsoft.com/rest/api/. All POST requests are actions that can be invoked and are specified at the end of the URI path. For instance, to stop a VM, the request URI is https://management.azure.com/subscriptions/{SubscriptionId}/resourceGroups/{ResourceGroup}/providers/Microsoft.Compute/virtualMachines/{VM}/powerOff?api-version=

{APIVersion} and the corresponding action is `powerOff`. This can be found at https://docs.microsoft.com/rest/api/compute/virtualmachines/virtualmachines-stop.

Azure CLI

```
az resource invoke-action --action
                          [--api-version]
                          [--ids]
                          [--latest-include-preview]
                          [--name]
                          [--namespace]
                          [--parent]
                          [--request-body]
                          [--resource-group]
                          [--resource-type]
```

Examples

Power-off a vm, specified by Id.

Azure CLI

```
az resource invoke-action --action powerOff \
  --ids
/subscriptions/{SubID}/resourceGroups/{ResourceGroup}/providers/Microsoft.Co
mpute/virtualMachines/{VMName}
```

Capture information for a stopped vm.

Azure CLI

```
az resource invoke-action --action capture \
  --ids /subscriptions/{SubID}/resourceGroups/{ResourceGroup}/providers/ \
    Microsoft.Compute/virtualMachines/{VMName} \
  --request-body "{ \"vhdPrefix\": \"myPrefix\",
\"destinationContainerName\": \
    \"myContainer\", \"overwriteVhds\": true }"
```

Invoke an action on the resource. (autogenerated)

Azure CLI

```
az resource invoke-action --action capture --name MyResource --resource-
group MyResourceGroup --resource-type Microsoft.web/sites
```

Required Parameters

`--action`

The action that will be invoked on the specified resource.

Optional Parameters

`--api-version`

The api version of the resource (omit for the latest stable version).

`--ids`

One or more resource IDs (space-delimited). If provided, no other "Resource Id" arguments should be specified.

`--latest-include-preview -v`

Indicate that the latest api-version will be used regardless of whether it is preview version (like 2020-01-01-preview) or not. For example, if the supported api-version of resource provider is 2020-01-01-preview and 2019-01-01: when passing in this parameter it will take the latest version 2020-01-01-preview, otherwise it will take the latest stable version 2019-01-01 without passing in this parameter.
default value: False

`--name -n`

The resource name. (Ex: myC).

`--namespace`

Provider namespace (Ex: 'Microsoft.Provider').

`--parent`

The parent path (Ex: 'resA/myA/resB/myB').

`--request-body`

JSON encoded parameter arguments for the action that will be passed along in the post request body. Use @{file} to load from a file.

`--resource-group -g`

Name of resource group. You can configure the default group using `az configure --defaults group=<name>`.

`--resource-type`

The resource type (Ex: 'resC'). Can also accept namespace/type format (Ex: 'Microsoft.Provider/resC').

Global Parameters

`--debug`

Increase logging verbosity to show all debug logs.

`--help -h`

Show this help message and exit.

`--only-show-errors`

Only show errors, suppressing warnings.

`--output -o`

Output format.

`--query`

JMESPath query string. See http://jmespath.org/ for more information and examples.

`--subscription`

Name or ID of subscription. You can configure the default subscription using `az account set -s NAME_OR_ID`.

`--verbose`

Increase logging verbosity. Use --debug for full debug logs.

az resource list

Edit

List resources.

Azure CLI

```
az resource list [--location]
                 [--name]
                 [--namespace]
                 [--resource-group]
                 [--resource-type]
                 [--tag]
```

Examples

List all resources in the West US region.

Azure CLI

```
az resource list --location westus
```

List all resources with the name 'resourceName'.

Azure CLI

```
az resource list --name 'resourceName'
```

List all resources with the tag 'test'.

Azure CLI

```
az resource list --tag test
```

List all resources with a tag that starts with 'test'.

Azure CLI

```
az resource list --tag 'test*'
```

List all resources with the tag 'test' that have the value 'example'.

Azure CLI

```
az resource list --tag test=example
```

Optional Parameters

`--location -l`

Location. Values from: `az account list-locations`. You can configure the default location using `az configure --defaults location=<location>`.

`--name -n`

The resource name. (Ex: myC).

`--namespace`

Provider namespace (Ex: 'Microsoft.Provider').

`--resource-group -g`

Name of resource group. You can configure the default group using `az configure --defaults group=<name>`.

`--resource-type`

The resource type (Ex: 'resC'). Can also accept namespace/type format (Ex: 'Microsoft.Provider/resC').

`--tag`

A single tag in 'key[=value]' format. Use "" to clear existing tags.

Global Parameters

`--debug`

Increase logging verbosity to show all debug logs.

`--help -h`

Show this help message and exit.

`--only-show-errors`

Only show errors, suppressing warnings.

`--output -o`

Output format.

`--query`

JMESPath query string. See http://jmespath.org/ for more information and examples.

`--subscription`

Name or ID of subscription. You can configure the default subscription using `az account set -s NAME_OR_ID`.

`--verbose`

Increase logging verbosity. Use --debug for full debug logs.

az resource move

Edit

Moves resources from one resource group to another(can be under different subscription).

Azure CLI

```
az resource move --destination-group
                 --ids
                 [--destination-subscription-id]
```

Required Parameters

`--destination-group`

The destination resource group name.

`--ids`

The space-separated resource ids to be moved.

Optional Parameters

`--destination-subscription-id`

The destination subscription identifier.

Global Parameters

`--debug`

Increase logging verbosity to show all debug logs.

`--help -h`

Show this help message and exit.

`--only-show-errors`

Only show errors, suppressing warnings.

`--output -o`

Output format.

`--query`

JMESPath query string. See http://jmespath.org/ for more information and examples.

`--subscription`

Name or ID of subscription. You can configure the default subscription using `az account set -s NAME_OR_ID`.

`--verbose`

Increase logging verbosity. Use --debug for full debug logs.

az resource patch

Edit

Update a resource by PATCH request.

It supports updating resources with JSON-formatted string. If the patch operation fails, please try run 'az resource update' instead.

Azure CLI

```
az resource patch --properties
                  [--api-version]
                  [--ids]
                  [--is-full-object]
                  [--latest-include-preview]
                  [--name]
                  [--namespace]
                  [--parent]
                  [--resource-group]
                  [--resource-type]
```

Examples

Update a webapp by using the latest api-version whether this version is a preview version.

Azure CLI

```
az resource patch --ids
/subscriptions/{SubID}/resourceGroups/{ResourceGroup}/providers/Microsoft.We
b/sites/{WebApp} \
  --latest-include-preview --is-full-object --properties "{ \"tags\": {
\"key\": \"value\" } }"
```

Update a resource by using JSON configuration from a file.

Azure CLI

```
az resource patch --name MyResource --resource-group MyResourceGroup --
resource-type Microsoft.web/sites \
  --is-full-object --properties @jsonConfigFile
```

Update an API app by providing a JSON configuration.

Azure CLI

```
az resource patch --name MyApiApp --resource-group MyResourceGroup --
resource-type Microsoft.web/sites \
    --is-full-object --properties "{ \"kind\": \"api\", \"properties\": {
\"serverFarmId\": \
            \"/subscriptions/{SubID}/resourcegroups/{ResourceGroup} \
                /providers/Microsoft.Web/serverfarms/{ServicePlan}\" } }"
```

Required Parameters

`--properties -p`

A JSON-formatted string containing resource properties.

Optional Parameters

`--api-version`

The api version of the resource (omit for the latest stable version).

`--ids`

One or more resource IDs (space-delimited). If provided, no other "Resource Id" arguments should be specified.

`--is-full-object`

Indicate that the properties object includes other options such as location, tags, sku, and/or plan.
default value: False

`--latest-include-preview -v`

Indicate that the latest api-version will be used regardless of whether it is preview version (like 2020-01-01-preview) or not. For example, if the supported api-version of resource provider is 2020-01-01-preview and 2019-01-01: when passing in this parameter it will take the latest version 2020-01-01-preview, otherwise it will take the latest stable version 2019-01-01 without passing in this parameter.
default value: False

`--name -n`

The resource name. (Ex: myC).

`--namespace`

Provider namespace (Ex: 'Microsoft.Provider').

`--parent`

The parent path (Ex: 'resA/myA/resB/myB').

`--resource-group -g`

Name of resource group. You can configure the default group using `az configure --defaults group=<name>`.

`--resource-type`

The resource type (Ex: 'resC'). Can also accept namespace/type format (Ex: 'Microsoft.Provider/resC').

Global Parameters

`--debug`

Increase logging verbosity to show all debug logs.

`--help -h`

Show this help message and exit.

`--only-show-errors`

Only show errors, suppressing warnings.

`--output -o`

Output format.

`--query`

JMESPath query string. See http://jmespath.org/ for more information and examples.

`--subscription`

Name or ID of subscription. You can configure the default subscription using `az account set -s NAME_OR_ID`.

`--verbose`

Increase logging verbosity. Use --debug for full debug logs.

az resource show

Edit

Get the details of a resource.

Azure CLI

```
az resource show [--api-version]
                 [--ids]
                 [--include-response-body {false, true}]
                 [--latest-include-preview]
                 [--name]
                 [--namespace]
                 [--parent]
                 [--resource-group]
                 [--resource-type]
```

Examples

Show a virtual machine resource named 'MyVm'.

Azure CLI

```
az resource show -g MyResourceGroup -n MyVm --resource-type
"Microsoft.Compute/virtualMachines"
```

Show a web app using a resource identifier.

Azure CLI

```
az resource show --ids /subscriptions/0b1f6471-1bf0-4dda-aec3-
111111111111/resourceGroups/MyResourceGroup/providers/Microsoft.Web/sites/My
Webapp
```

Show a subnet.

Azure CLI

```
az resource show -g MyResourceGroup -n MySubnet --namespace
Microsoft.Network --parent virtualnetworks/MyVnet --resource-type subnets
```

Show a subnet using a resource identifier.

Azure CLI

```
az resource show --ids /subscriptions/0b1f6471-1bf0-4dda-aec3-
111111111111/resourceGroups/MyResourceGroup/providers/Microsoft.Network/virt
ualNetworks/MyVnet/subnets/MySubnet
```

Show an application gateway path rule.

Azure CLI

```
az resource show -g MyResourceGroup --namespace Microsoft.Network --parent
applicationGateways/ag1/urlPathMaps/map1 --resource-type pathRules -n rule1
```

Show a virtual machine resource named 'MyVm' by using the latest api-version whether this version is a preview version.

Azure CLI

```
az resource show -g MyResourceGroup -n MyVm --resource-type
"Microsoft.Compute/virtualMachines" --latest-include-preview
```

Optional Parameters

`--api-version`

The api version of the resource (omit for the latest stable version).

`--ids`

One or more resource IDs (space-delimited). If provided, no other "Resource Id" arguments should be specified.

`--include-response-body`

Use if the default command output doesn't capture all of the property data.

accepted values: false, true

default value: False

`--latest-include-preview -v`

Indicate that the latest api-version will be used regardless of whether it is preview version (like 2020-01-01-preview) or not. For example, if the supported api-version of resource provider is 2020-01-01-preview and 2019-01-01: when passing in this parameter it will take the latest version 2020-01-01-preview, otherwise it will take the latest stable version 2019-01-01 without passing in this parameter.

default value: False

`--name -n`

The resource name. (Ex: myC).

`--namespace`

Provider namespace (Ex: 'Microsoft.Provider').

`--parent`

The parent path (Ex: 'resA/myA/resB/myB').

`--resource-group -g`

Name of resource group. You can configure the default group using `az configure --defaults group=<name>`.

`--resource-type`

The resource type (Ex: 'resC'). Can also accept namespace/type format (Ex: 'Microsoft.Provider/resC').

Global Parameters

`--debug`

Increase logging verbosity to show all debug logs.

`--help -h`

Show this help message and exit.

`--only-show-errors`

Only show errors, suppressing warnings.

`--output -o`

Output format.

`--query`

JMESPath query string. See http://jmespath.org/ for more information and examples.

`--subscription`

Name or ID of subscription. You can configure the default subscription using `az account set -s NAME_OR_ID`.

`--verbose`

Increase logging verbosity. Use --debug for full debug logs.

az resource tag

Edit

Tag a resource.

Azure CLI

```
az resource tag --tags
                [--api-version]
                [--ids]
                [--is-incremental]
                [--latest-include-preview]
                [--name]
                [--namespace]
                [--parent]
                [--resource-group]
                [--resource-type]
```

Examples

Tag the virtual machine 'MyVm' with the key 'vmlist' and value 'vm1'.

Azure CLI

```
az resource tag --tags vmlist=vm1 -g MyResourceGroup -n MyVm --resource-type
"Microsoft.Compute/virtualMachines"
```

Tag a web app with the key 'vmlist' and value 'vm1', using a resource identifier.

Azure CLI

```
az resource tag --tags vmlist=vm1 --ids
/subscriptions/{SubID}/resourceGroups/{ResourceGroup}/providers/Microsoft.We
b/sites/{WebApp}
```

Tag the virtual machine 'MyVm' with the key 'vmlist' and value 'vm1' incrementally. It doesn't empty the existing tags.

Azure CLI

```
az resource tag --tags vmlist=vm1 -g MyResourceGroup -n MyVm --resource-type
"Microsoft.Compute/virtualMachines" -i
```

Tag the virtual machine 'MyVm' with the key 'vmlist' and value 'vm1' by using the latest api-version whether this version is a preview version.

Azure CLI

```
az resource tag --tags vmlist=vm1 -g MyResourceGroup -n MyVm --resource-type
"Microsoft.Compute/virtualMachines" --latest-include-preview
```

Required Parameters

`--tags`

Space-separated tags: key[=value] [key[=value] ...]. Use "" to clear existing tags.

Optional Parameters

`--api-version`

The api version of the resource (omit for the latest stable version).

`--ids`

One or more resource IDs (space-delimited). If provided, no other "Resource Id" arguments should be specified.

`--is-incremental -i`

The option to add tags incrementally without deleting the original tags. If the key of new tag and original tag are duplicated, the original value will be overwritten.

`--latest-include-preview -v`

Indicate that the latest api-version will be used regardless of whether it is preview version (like 2020-01-01-preview) or not. For example, if the supported api-version of resource provider is 2020-01-01-preview and 2019-01-01: when passing in this parameter it will take the latest version 2020-01-01-preview, otherwise it will take the latest stable version 2019-01-01 without passing in this parameter.
default value: False

`--name -n`

The resource name. (Ex: myC).

`--namespace`

Provider namespace (Ex: 'Microsoft.Provider').

`--parent`

The parent path (Ex: 'resA/myA/resB/myB').

`--resource-group -g`

Name of resource group. You can configure the default group using `az configure --defaults group=<name>`.

`--resource-type`

The resource type (Ex: 'resC'). Can also accept namespace/type format (Ex: 'Microsoft.Provider/resC').

Global Parameters

`--debug`

Increase logging verbosity to show all debug logs.

`--help -h`

Show this help message and exit.

`--only-show-errors`

Only show errors, suppressing warnings.

`--output -o`

Output format.

`--query`

JMESPath query string. See http://jmespath.org/ for more information and examples.

`--subscription`

Name or ID of subscription. You can configure the default subscription using `az account set -s NAME_OR_ID`.

`--verbose`

Increase logging verbosity. Use --debug for full debug logs.

az resource update

Edit

Update a resource by PUT request.

It supports the generic update (using property path) to update resources. If the update operation fails, please try run 'az resource patch' instead.

Azure CLI

```
az resource update [--add]
                   [--api-version]
                   [--force-string]
                   [--ids]
                   [--include-response-body {false, true}]
                   [--latest-include-preview]
```

```
                    [--name]
                    [--namespace]
                    [--parent]
                    [--remove]
                    [--resource-group]
                    [--resource-type]
                    [--set]
```

Examples

Update a webapp by using the latest api-version whether this version is a preview version.

Azure CLI

```
az resource update --ids
/subscriptions/{SubID}/resourceGroups/{ResourceGroup}/providers/Microsoft.We
b/sites/{WebApp} --set tags.key=value --latest-include-preview
```

Update a resource. (autogenerated)

Azure CLI

```
az resource update --ids $id --set properties.connectionType=Proxy
```

Update a resource. (autogenerated)

Azure CLI

```
az resource update --name MyResource --resource-group MyResourceGroup --
resource-type subnets --set tags.key=value
```

Optional Parameters

`--add`

Add an object to a list of objects by specifying a path and key value pairs. Example: --add property.listProperty <key=value, string or JSON string>.

default value: []

`--api-version`

The api version of the resource (omit for the latest stable version).

`--force-string`

When using 'set' or 'add', preserve string literals instead of attempting to convert to JSON.

default value: False

`--ids`

One or more resource IDs (space-delimited). If provided, no other "Resource Id" arguments should be specified.

`--include-response-body`

Use if the default command output doesn't capture all of the property data.

accepted values: false, true

default value: False

`--latest-include-preview -v`

Indicate that the latest api-version will be used regardless of whether it is preview version (like 2020-01-01-preview) or not. For example, if the supported api-version of resource provider is 2020-01-01-preview and 2019-01-01: when passing in this parameter it will take the latest version 2020-01-01-preview, otherwise it will take the latest stable version 2019-01-01 without passing in this parameter.

default value: False

`--name -n`

The resource name. (Ex: myC).

`--namespace`

Provider namespace (Ex: 'Microsoft.Provider').

`--parent`

The parent path (Ex: 'resA/myA/resB/myB').

`--remove`

Remove a property or an element from a list. Example: --remove property.list OR --remove propertyToRemove.

default value: []

`--resource-group -g`

Name of resource group. You can configure the default group using `az configure --defaults group=<name>`.

`--resource-type`

The resource type (Ex: 'resC'). Can also accept namespace/type format (Ex: 'Microsoft.Provider/resC').

`--set`

Update an object by specifying a property path and value to set. Example: --set property1.property2=.

default value: []

Global Parameters

`--debug`

Increase logging verbosity to show all debug logs.

`--help -h`

Show this help message and exit.

`--only-show-errors`

Only show errors, suppressing warnings.

`--output -o`

Output format.

`--query`

JMESPath query string. See http://jmespath.org/ for more information and examples.

`--subscription`

Name or ID of subscription. You can configure the default subscription using `az account set -s NAME_OR_ID`.

`--verbose`

Increase logging verbosity. Use --debug for full debug logs.

az resource wait

Edit

Place the CLI in a waiting state until a condition of a resources is met.

Azure CLI

```
az resource wait [--api-version]
                 [--created]
                 [--custom]
                 [--deleted]
                 [--exists]
                 [--ids]
                 [--include-response-body {false, true}]
                 [--interval]
                 [--name]
                 [--namespace]
                 [--parent]
                 [--resource-group]
                 [--resource-type]
                 [--timeout]
                 [--updated]
```

Examples

Place the CLI in a waiting state until a condition of a resources is met. (autogenerated)

Azure CLI

```
az resource wait --exists --ids
/subscriptions/{SubID}/resourceGroups/{ResourceGroup}/providers/Microsoft.We
b/sites/{WebApp}
```

Place the CLI in a waiting state until a condition of a resources is met. (autogenerated)

Azure CLI

```
az resource wait --exists --ids
/subscriptions/{SubID}/resourceGroups/{ResourceGroup}/providers/Microsoft.We
b/sites/{WebApp} --include-response-body true
```

Place the CLI in a waiting state until a condition of a resources is met. (autogenerated)

Azure CLI

```
az resource wait --exists --name MyResource --resource-group MyResourceGroup
--resource-type subnets
```

Optional Parameters

`--api-version`

The api version of the resource (omit for the latest stable version).

`--created`

Wait until created with 'provisioningState' at 'Succeeded'.

default value: False

`--custom`

Wait until the condition satisfies a custom JMESPath query. E.g. provisioningState!='InProgress', instanceView.statuses[?code=='PowerState/running'].

`--deleted`

Wait until deleted.

default value: False

`--exists`

Wait until the resource exists.

default value: False

`--ids`

One or more resource IDs (space-delimited). If provided, no other "Resource Id" arguments should be specified.

`--include-response-body`

Use if the default command output doesn't capture all of the property data.

accepted values: false, true

default value: False

`--interval`

Polling interval in seconds.

default value: 30

`--name -n`

The resource name. (Ex: myC).

`--namespace`

Provider namespace (Ex: 'Microsoft.Provider').

`--parent`

The parent path (Ex: 'resA/myA/resB/myB').

`--resource-group -g`

Name of resource group. You can configure the default group using `az configure --defaults group=<name>`.

`--resource-type`

The resource type (Ex: 'resC'). Can also accept namespace/type format (Ex: 'Microsoft.Provider/resC').

`--timeout`

Maximum wait in seconds.

default value: 3600

`--updated`

Wait until updated with provisioningState at 'Succeeded'.

default value: False

Global Parameters

`--debug`

Increase logging verbosity to show all debug logs.

`--help -h`

Show this help message and exit.

`--only-show-errors`

Only show errors, suppressing warnings.

`--output -o`

Output format.

`--query`

JMESPath query string. See http://jmespath.org/ for more information and examples.

`--subscription`

Name or ID of subscription. You can configure the default subscription using `az account set -s NAME_OR_ID`.

`--verbose`

Increase logging verbosity. Use --debug for full debug logs.

Microsoft.Azure.Management.Resource Manager Namespace

Reference

Classes

DataPolicyManifests OperationsExtensions	Extension methods for DataPolicyManifestsOperations.
DeploymentOperations Extensions	Extension methods for DeploymentOperations.
DeploymentScriptsClient	The APIs listed in this specification can be used to manage Deployment Scripts resource through the Azure Resource Manager.
DeploymentScriptsOperations Extensions	Extension methods for DeploymentScriptsOperations.
DeploymentsOperations Extensions	Extension methods for DeploymentsOperations.
FeatureClient	Azure Feature Exposure Control (AFEC) provides a mechanism for the resource providers to control feature exposure to users. Resource providers typically use this mechanism to provide public/private preview for new features prior to making them generally available. Users need to explicitly register for AFEC features to get access to such functionality.
FeatureClientExtensions	Extension methods for FeatureClient.
FeaturesOperationsExtensions	Extension methods for FeaturesOperations.
ManagementLinkClient	Azure resources can be linked together to form logical relationships. You can establish links between resources belonging to different resource groups. However, all the linked resources must belong to the same subscription. Each resource can be linked to 50 other resources. If any of the linked resources are deleted or moved, the link owner must clean up the remaining link.
ManagementLockClient	Azure resources can be locked to prevent other users in your organization from deleting or modifying resources.
ManagementLocksOperations Extensions	Extension methods for ManagementLocksOperations.

OperationsExtensions	Extension methods for Operations.
PolicyAssignmentsOperations Extensions	Extension methods for PolicyAssignmentsOperations.
PolicyClient	To manage and control access to your resources, you can define customized policies and assign them at a scope.
PolicyDefinitionsOperations Extensions	Extension methods for PolicyDefinitionsOperations.
PolicyExemptionsOperations Extensions	Extension methods for PolicyExemptionsOperations.
PolicySetDefinitions OperationsExtensions	Extension methods for PolicySetDefinitionsOperations.
PrivateLinkAssociation OperationsExtensions	Extension methods for PrivateLinkAssociationOperations.
ProviderResourceTypes OperationsExtensions	Extension methods for ProviderResourceTypesOperations.
ProvidersOperations Extensions	Extension methods for ProvidersOperations.
ResourceGroupsOperations Extensions	Extension methods for ResourceGroupsOperations.
ResourceLinksOperations Extensions	Extension methods for ResourceLinksOperations.
ResourceManagementClient	Provides operations for working with resources and resource groups.
ResourceManagementPrivate LinkOperationsExtensions	Extension methods for ResourceManagementPrivateLinkOperations.
ResourcePrivateLinkClient	Provides operations for managing private link resources
ResourcesOperations Extensions	Extension methods for ResourcesOperations.
SafeJsonConvertWrapper	
SubscriptionClient	All resource groups and resources exist within subscriptions. These operation enable you get information about your subscriptions and tenants. A tenant is a dedicated instance of Azure Active Directory (Azure AD) for your organization.
SubscriptionClientExtensions	Extension methods for SubscriptionClient.
SubscriptionFeature	Extension methods for

RegistrationsOperations Extensions	SubscriptionFeatureRegistrationsOperations.
SubscriptionsOperations Extensions	Extension methods for SubscriptionsOperations.
TagsOperationsExtensions	Extension methods for TagsOperations.
TemplateSpecsClient	The APIs listed in this specification can be used to manage Template Spec resources through the Azure Resource Manager.
TemplateSpecsOperations Extensions	Extension methods for TemplateSpecsOperations.
TemplateSpecVersions OperationsExtensions	Extension methods for TemplateSpecVersionsOperations.
TenantsOperationsExtensions	Extension methods for TenantsOperations.

Interfaces

IDataPolicyManifests Operations	DataPolicyManifestsOperations operations.
IDeploymentOperations	DeploymentOperations operations.
IDeploymentScriptsClient	The APIs listed in this specification can be used to manage Deployment Scripts resource through the Azure Resource Manager.
IDeploymentScripts Operations	DeploymentScriptsOperations operations.
IDeploymentsOperations	DeploymentsOperations operations.
IFeatureClient	
IFeaturesOperations	FeaturesOperations operations.
IManagementLinkClient	Azure resources can be linked together to form logical relationships. You can establish links between resources belonging to different resource groups. However, all the linked resources must belong to the same subscription. Each resource can be linked to 50 other resources. If any of the linked resources are deleted or moved, the link owner must clean up the remaining link.
IManagementLockClient	Azure resources can be locked to prevent other users in your organization from deleting or modifying resources.

IManagementLocks Operations	ManagementLocksOperations operations.
IOperations	Operations operations.
IPolicyAssignmentsOperations	PolicyAssignmentsOperations operations.
IPolicyClient	
IPolicyDefinitionsOperations	PolicyDefinitionsOperations operations.
IPolicyExemptionsOperations	PolicyExemptionsOperations operations.
IPolicySetDefinitions Operations	PolicySetDefinitionsOperations operations.
IPrivateLinkAssociation Operations	PrivateLinkAssociationOperations operations.
IProviderResourceTypes Operations	ProviderResourceTypesOperations operations.
IProvidersOperations	ProvidersOperations operations.
IResourceGroupsOperations	ResourceGroupsOperations operations.
IResourceLinksOperations	ResourceLinksOperations operations.
IResourceManagementClient	Provides operations for working with resources and resource groups.
IResourceManagementPrivate LinkOperations	ResourceManagementPrivateLinkOperations operations.
IResourcePrivateLinkClient	Provides operations for managing private link resources
IResourcesOperations	ResourcesOperations operations.
ISubscriptionClient	All resource groups and resources exist within subscriptions. These operation enable you get information about your subscriptions and tenants. A tenant is a dedicated instance of Azure Active Directory (Azure AD) for your organization.
ISubscriptionFeature RegistrationsOperations	SubscriptionFeatureRegistrationsOperations operations.
ISubscriptionsOperations	SubscriptionsOperations operations.
ITagsOperations	TagsOperations operations.
ITemplateSpecsClient	The APIs listed in this specification can be used to manage Template Spec resources through the Azure Resource Manager.

ITemplateSpecsOperations	TemplateSpecsOperations operations.
ITemplateSpecVersions Operations	TemplateSpecVersionsOperations operations.
ITenantsOperations	TenantsOperations operations.

com.microsoft.azure.management.resources

Reference

Package: com.microsoft.azure.management.resources

Maven Artifact: com.microsoft.azure:azure-mgmt-resources:1.41.4

This package contains the classes for ResourceManagementClient. Provides operations for working with resources and resource groups.

Classes

Alias	The alias type.
AliasPath	The type of the paths for alias.
AliasPathAttributes	Defines values for AliasPathAttributes.
AliasPathMetadata	The AliasPathMetadata model.
AliasPathTokenType	Defines values for AliasPathTokenType.
AliasPathType	The type of the paths for alias.
AliasPattern	The type of the pattern for an alias path.
ApiProfile	The ApiProfile model.
BasicDependency	Deployment dependency information.
DebugSetting	The debug setting.
Dependency	Deployment dependency information.
DeploymentExtendedFilter	Deployment filter.
DeploymentOperationProperties	Deployment operation properties.
DeploymentProperties	Deployment properties.
DeploymentPropertiesExtended	Deployment properties with additional details.
DeploymentWhatIf	Deployment What-if operation parameters.
DeploymentWhatIfProperties	Deployment What-if properties.

DeploymentWhatIfSettings	Deployment What-If operation settings.
DeploymentsWhatIfAt ManagementGroupScope Headers	Defines headers for WhatIfAtManagementGroupScope operation.
DeploymentsWhatIfAt SubscriptionScopeHeaders	Defines headers for WhatIfAtSubscriptionScope operation.
DeploymentsWhatIfAtTenant ScopeHeaders	Defines headers for WhatIfAtTenantScope operation.
DeploymentsWhatIfHeaders	Defines headers for WhatIf operation.
ErrorAdditionalInfo	The resource management error additional info.
ErrorDefinition	Error description and code explaining why resource name is invalid.
ErrorResponse	Error Response.
ErrorResponseException	Exception thrown for an invalid response with ErrorResponse information.
ExportTemplateRequest	Export resource group template request parameters.
ExpressionEvaluationOptions	Specifies whether template expressions are evaluated within the scope of the parent template or nested template.
ExpressionEvaluationOptions ScopeType	Defines values for ExpressionEvaluationOptionsScopeType.
FeatureProperties	Information about feature.
ForceDeletionResourceType	Resource type for force deletion.
GenericResourceFilter	Resource filter.
HttpMessage	HTTP message.
Identity	Identity for the resource.
IdentityUserAssignedIdentities Value	The IdentityUserAssignedIdentitiesValue model.
LocationMetadata	Location metadata information.
ManagedByTenant	Information about a tenant managing the subscription.
OnErrorDeployment	Deployment on error behavior.
OnErrorDeploymentExtended	Deployment on error behavior with additional details.
OperationDisplay	The object that represents the operation.

PairedRegion	Information regarding paired region.
ParametersLink	Entity representing the reference to the deployment parameters.
Plan	Plan for the resource.
PolicyDefinitionReference	The policy definition reference.
PolicyMode	Defines values for PolicyMode.
PolicySku	The policy sku.
PolicyType	Defines values for PolicyType.
ProviderResourceType	Resource type managed by the resource provider.
ProvisioningState	Defines values for ProvisioningState.
RegionCategory	Defines values for RegionCategory.
RegionType	Defines values for RegionType.
ResourceErrorResponse	Error response indicates ARM is not able to process the incoming request.
ResourceGroupFilter	Resource group filter.
ResourceGroupPatchable	Resource group information.
ResourceGroupProperties	The resource group properties.
ResourceManagementError WithDetails	The detailed error message of resource management.
ResourceName	Name and Type of the Resource.
ResourceNameStatus	Defines values for ResourceNameStatus.
ResourceProviderOperation DisplayProperties	Resource provider operation's display properties.
ResourceReference	The resource Id model.
ResourcesMoveInfo	Parameters of move resources.
ScopedDeployment	Deployment operation parameters.
ScopedDeploymentWhatIf	Deployment What-if operation parameters.
Sku	SKU for the resource.
StatusMessage	Operation status message object.
SubscriptionPolicies	Subscription policies.

TagCount	Tag count.
Tags	A dictionary of name and value pairs.
TagsPatchOperation	Defines values for TagsPatchOperation.
TagsPatchResource	Wrapper resource for tags patch API request only.
TargetResource	Target resource.
TemplateLink	Entity representing the reference to the template.
WhatIfChange	Information about a single resource change predicted by What-If operation.
WhatIfPropertyChange	The predicted change to the resource property.

Interfaces

Deployment	An immutable client-side representation of an Azure deployment.
Deployment.Definition	Container interface for all the deployment definitions.
Deployment.DefinitionStages	Grouping of all the deployment definition stages.
Deployment.DefinitionStages.Blank	The first stage of deployment definition.
Deployment.DefinitionStages.WithCreate	A deployment definition with sufficient inputs to create a new deployment in the cloud, but exposing additional optional inputs to specify.
Deployment.DefinitionStages.WithGroup	A deployment definition allowing resource group to be specified.
Deployment.DefinitionStages.WithMode	A deployment definition allowing the deployment mode to be specified.
Deployment.DefinitionStages.WithParameters	A deployment definition allowing the parameters to be specified.
Deployment.DefinitionStages.WithTemplate	A deployment definition allowing the template to be specified.
Deployment.Execution	Container interface for all the deployment execution.
Deployment.ExecutionStages	Grouping of all the deployment execution stages.
Deployment.ExecutionStages.	The first stage of deployment execution.

Blank	
Deployment.ExecutionStages.WithExecute	
Deployment.ExecutionStages.WithWhatIf	A deployment execution allowing What-if parameters to be specified.
Deployment.ExecutionStages.WithWhatIfDeploymentMode	A deployment execution allowing deployment mode to be specified.
Deployment.ExecutionStages.WithWhatIfLocation	A deployment execution allowing data storage location to be specified.
Deployment.ExecutionStages.WithWhatIfOnErrorDeploymentType	A deployment execution allowing on error deployment type to be specified.
Deployment.ExecutionStages.WithWhatIfParameter	A deployment execution allowing parameter to be specified.
Deployment.ExecutionStages.WithWhatIfResultFormat	A deployment execution allowing result format to be specified.
Deployment.ExecutionStages.WithWhatIfTemplate	A deployment execution allowing template to be specified.
Deployment.Update	The template for a deployment update operation, containing all the settings that can be modified.
Deployment.UpdateStages	Grouping of all the deployment updates stages.
Deployment.UpdateStages.WithMode	A deployment update allowing to change the deployment mode.
Deployment.UpdateStages.WithParameters	A deployment update allowing to change the parameters.
Deployment.UpdateStages.WithTemplate	A deployment update allowing to change the template.
DeploymentExportResult	An immutable client-side representation of an Azure deployment template export result.
DeploymentOperation	An immutable client-side representation of a deployment operation.
DeploymentOperations	Entry point to deployment operation management API.
Deployments	Entry point to template deployment in Azure.
Feature	An immutable client-side representation of an Azure feature.
Features	Entry point to features management API.

GenericResource	An immutable client-side representation of an Azure generic resource.
GenericResource.Definition	The entirety of the generic resource definition.
GenericResource.Definition Stages	Grouping of generic resource definition stages.
GenericResource.Definition Stages.Blank	A generic resource definition allowing region to be specified.
GenericResource.Definition Stages.WithApiVersion	A generic resource definition allowing api version to be specified.
GenericResource.Definition Stages.WithCreate	A deployment definition with sufficient inputs to create a new resource in the cloud, but exposing additional optional inputs to specify.
GenericResource.Definition Stages.WithGroup	A generic resource definition allowing resource group to be specified.
GenericResource.Definition Stages.WithParentResource	A generic resource definition allowing parent resource to be specified.
GenericResource.Definition Stages.WithPlan	A generic resource definition allowing plan to be specified.
GenericResource.Definition Stages.WithProvider Namespace	A generic resource definition allowing provider namespace to be specified.
GenericResource.Definition Stages.WithResourceType	A generic resource definition allowing resource type to be specified.
GenericResource.Update	The template for a generic resource update operation, containing all the settings that can be modified.
GenericResource.Update Stages	Grouping of generic resource update stages.
GenericResource.Update Stages.WithApiVersion	The template for a generic resource update operation for specifying the resource provider API version.
GenericResource.Update Stages.WithParentResource	A generic resource update allowing to change the parent resource.
GenericResource.Update Stages.WithPlan	A generic resource update allowing to change the resource plan.
GenericResource.Update Stages.WithProperties	A generic resource update allowing to change the resource properties.

GenericResources	Entry point to generic resources management API.
Location	An immutable client-side representation of an Azure location.
PolicyAssignment	An immutable client-side representation of an Azure policy assignment.
PolicyAssignment.Definition	Container interface for all the definitions that need to be implemented.
PolicyAssignment.Definition Stages	Grouping of all the policy assignment definition stages.
PolicyAssignment.Definition Stages.Blank	The first stage of the policy assignment.
PolicyAssignment.Definition Stages.WithCreate	A policy assignment with sufficient inputs to create a new policy assignment in the cloud, but exposing additional optional inputs to specify.
PolicyAssignment.Definition Stages.WithDisplayName	A policy assignment allowing the display name to be set.
PolicyAssignment.Definition Stages.WithPolicyDefinition	A policy assignment allowing the policy definition to be set.
PolicyAssignment.Definition Stages.WithScope	A policy assignment specifying the scope of the policy.
PolicyAssignments	Entry point to policy assignment management API.
PolicyDefinition	An immutable client-side representation of an Azure policy.
PolicyDefinition.Definition	Container interface for all the definitions that need to be implemented.
PolicyDefinition.Definition Stages	Grouping of all the policy definition stages.
PolicyDefinition.Definition Stages.Blank	The first stage of the policy definition.
PolicyDefinition.Definition Stages.WithCreate	A policy definition with sufficient inputs to create a new policy in the cloud, but exposing additional optional inputs to specify.
PolicyDefinition.Definition Stages.WithDescription	A policy definition allowing policy description to be set.
PolicyDefinition.Definition Stages.WithDisplayName	A policy definition allowing the display name to be set.
PolicyDefinition.Definition	A policy definition allowing the policy rule to be set.

Stages.WithPolicyRule	
PolicyDefinition.Definition Stages.WithPolicyType	A policy definition specifying the policy type to be blob.
PolicyDefinition.Update	The template for a policy update operation, containing all the settings that can be modified.
PolicyDefinition.UpdateStages	Grouping of all the policy update stages.
PolicyDefinition.UpdateStages. WithDescription	A policy definition allowing the policy description to be set.
PolicyDefinition.UpdateStages. WithDisplayName	A policy definition allowing the display name to be set.
PolicyDefinition.UpdateStages. WithPolicyRule	A policy definition allowing the policy rule to be set.
PolicyDefinition.UpdateStages. WithPolicyType	A policy definition allowing the policy type to be set.
PolicyDefinitions	Entry point to tenant management API.
Provider	An immutable client-side representation of an Azure resource provider.
Providers	Entry point to providers management API.
ResourceConnector	Defines a connector that connects other resources to a resource group.
ResourceConnector. Builder<T>	Implementations of this interface defines how to create a connector.
ResourceGroup	An immutable client-side representation of an Azure resource group.
ResourceGroup.Definition	Container interface for all the definitions that need to be implemented.
ResourceGroup.Definition Stages	Grouping of all the resource group definition stages.
ResourceGroup.Definition Stages.Blank	A resource group definition allowing location to be set.
ResourceGroup.Definition Stages.WithCreate	A resource group definition with sufficient inputs to create a new resource group in the cloud, but exposing additional optional inputs to specify.
ResourceGroup.Update	The template for a resource group update operation, containing all the settings that can be modified.

ResourceGroup.UpdateStages	Grouping of all the resource group update stages.
ResourceGroupExportResult	An immutable client-side representation of an Azure deployment template export result.
ResourceGroups	Entry point to resource group management API.
Subscription	An immutable client-side representation of an Azure subscription.
Subscriptions	Entry point to subscription management API.
TagOperations	Entry point to tag management API.
TagResource	An immutable client-side representation of an Azure resource with tags.
Tenant	An immutable client-side representation of an Azure tenant.
Tenants	Entry point to tenant management API.
WhatIfOperationResult	An immutable client-side representation of an Azure deployment What-if operation result.

Enums

AliasPatternType	Defines values for AliasPatternType.
AliasType	Defines values for AliasType.
ChangeType	Defines values for ChangeType.
DeploymentMode	Defines values for DeploymentMode.
OnErrorDeploymentType	Defines values for OnErrorDeploymentType.
PropertyChangeType	Defines values for PropertyChangeType.
ProvisioningOperation	Defines values for ProvisioningOperation.
ResourceGroupExport TemplateOptions	The export template options.
ResourceIdentityType	Defines values for ResourceIdentityType.
SpendingLimit	Defines values for SpendingLimit.
SubscriptionState	Defines values for SubscriptionState.
TenantCategory	Defines values for TenantCategory.

WhatIfResultFormat	Defines values for WhatIfResultFormat.

Azure Resources libraries for python

Article • 03/14/2023

Overview

Deploy, monitor, and manage resources in groups with Azure Resource Manager.

Management API

Use the management API to create resource groups and deploy resources from templates.

Bash

```
pip install azure-mgmt-resource
```

Example

Create a new resource group in the Azure Eastern US region.

Python

```
from azure.mgmt.resource import ResourceManagementClient

LOCATION = 'eastus'
GROUP_NAME ='sample_resource_group'

resource_client = ResourceManagementClient(credentials, subscription_id)
resource_client.resource_groups.create_or_update(GROUP_NAME, {'location':
LOCATION})
```

Explore the Management APIs

Samples

Manage Azure resources and resource groups

View the complete list of Azure Resource Manager samples.