Conditionally deploy a resource in an Azure Resource Manager template

10/30/2018 • 3 minutes to read • Contributors all

In this article

Example template

Try the template

Next steps

There are some scenarios in which you need to design your template to deploy a resource based on a condition, such as whether or not a parameter value is present. For example, your template may deploy a virtual network and include parameters to specify other virtual networks for peering. If you've not specified any parameter values for peering, you don't want Resource Manager to deploy the peering resource.

To accomplish this, use the condition element in the resource to test the length of your parameter array. If the length is zero, return `false` to prevent deployment, but for all values greater than zero return `true` to allow deployment.

Example template

Let's look at an example template that demonstrates this. Our template uses the condition element to control deployment of the `Microsoft.Network/virtualNetworks/virtualNetworkPeerings` resource. This resource creates a peering between two Azure Virtual Networks in the same region.

Let's take a look at each section of the template.

The `parameters` element defines a single parameter named `virtualNetworkPeerings`:

JSON Copy

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "virtualNetworkPeerings": {
      "type": "array",
      "defaultValue": []
    }
  },
```

Our `virtualNetworkPeerings` parameter is an `array` and has the following schema:

JSON Copy

```
"virtualNetworkPeerings": [
    {
      "name": "firstVNet/peering1",
      "properties": {
          "remoteVirtualNetwork": {
              "id": "[resourceId('Microsoft.Network/virtualNetworks','secondVNet')]"
          },
          "allowForwardedTraffic": true,
          "allowGatewayTransit": true,
          "useRemoteGateways": false
      }
```

```
    }
]
```

The properties in our parameter specify the settings related to peering virtual networks. We'll provide the values for these properties when we specify the `Microsoft.Network/virtualNetworks/virtualNetworkPeerings` resource in the `resources` section:

JSON Copy

```
"resources": [
    {
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2017-05-10",
      "name": "[concat('vnp-', copyIndex())]",
      "condition": "[greater(length(parameters('virtualNetworkPeerings')), 0)]",
      "dependsOn": [
        "firstVNet", "secondVNet"
      ],
      "copy": {
          "name": "iterator",
          "count": "[length(variables('peerings'))]",
          "mode": "serial"
      },
      "properties": {
        "mode": "Incremental",
        "template": {
          "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTem-
plate.json#",
          "contentVersion": "1.0.0.0",
          "parameters": {
          },
          "variables": {
          },
          "resources": [
            {
              "type": "Microsoft.Network/virtualNetworks/virtualNetworkPeerings",
              "apiVersion": "2016-06-01",
              "location": "[resourceGroup().location]",
              "name": "[variables('peerings')[copyIndex()].name]",
              "properties": "[variables('peerings')[copyIndex()].properties]"
            }
          ],
          "outputs": {
          }
        }
      }
    }
]
```

There are a couple of things going on in this part of our template. First, the actual resource being deployed is an inline template of type `Microsoft.Resources/deployments` that includes its own template that actually deploys the `Microsoft.Network/virtualNetworks/virtualNetworkPeerings`.

Our `name` for the inline template is made unique by concatenating the current iteration of the `copyIndex()` with the prefix `vnp-`.

The `condition` element specifies that our resource should be processed when the `greater()` function evaluates to `true`. Here, we're testing if the `virtualNetworkPeerings` parameter array is `greater()` than zero. If it is, it evaluates to `true` and the `condition` is satisfied. Otherwise, it's `false`.

Next, we specify our `copy` loop. It's a `serial` loop that means the loop is done in sequence, with each resource waiting until the last resource has been deployed. The `count` property specifies the number of times the loop iterates.

Here, normally we'd set it to the length of the `virtualNetworkPeerings` array because it contains the parameter objects specifying the resource we want to deploy. However, if we do that, validation will fail if the array is empty because Resource Manager notices that we are attempting to access properties that do not exist. We can work around this, however. Let's take a look at the variables we'll need:

JSON Copy

```
  "variables": {
    "workaround": {
       "true": "[parameters('virtualNetworkPeerings')]",
       "false": [{
           "name": "workaround",
           "properties": {}
       }]
     },
     "peerings": "[variables('workaround')[string(greater(length(parameters('virtualNetwork-
Peerings')), 0))]]"
  },
```

Our `workaround` variable includes two properties, one named `true` and one named `false`. The `true` property evaluates to the value of the `virtualNetworkPeerings` parameter array. The `false` property evaluates to an empty object including the named properties that Resource Manager expects to see—note that `false` is actually an array, just as our `virtualNetworkPeerings` parameter is, which will satisfy validation.

Our `peerings` variable uses our `workaround` variable by once again testing if the length of the `virtualNetworkPeerings` parameter array is greater than zero. If it is, the `string` evaluates to `true` and the `workaround` variable evaluates to the `virtualNetworkPeerings` parameter array. Otherwise, it evaluates to `false` and the `workaround` variable evaluates to our empty object in the first element of the array.

Now that we've worked around the validation issue, we can simply specify the deployment of the `Microsoft.Network/virtualNetworks/virtualNetworkPeerings` resource in the nested template, passing the `name` and `properties` from our `virtualNetworkPeerings` parameter array. You can see this in the `template` element nested in the `properties` element of our resource.

Try the template

An example template is available on GitHub. To deploy the template, run the following Azure CLI commands:

bash Copy

```
az group create --location <location> --name <resource-group-name>
az group deployment create -g <resource-group-name> \
    --template-uri https://raw.githubusercontent.com/mspnp/template-examples/master/example2-
conditional/deploy.json
```

Next steps

- Use objects instead of scalar values as template parameters. See Use an object as a parameter in an Azure Resource Manager template