Agenda: Managing ARM Templates.

- Understanding Azure Resource Manager (ARM)
- Exporting and Importing ARM templates.
- ARM Resource Providers
- Deploy ARM Templates
    - Using PowerShell
    - Azure CLI
    - Azure Portal
    - REST API
    - Incremental and Complete Deployments

About Azure Resource Manager

The infrastructure that makes up your application is often composed of various different components. For instance, you might simply be running a web site, but behind the scenes you have an Azure web site deployed, a Storage account for tables, blobs, and queues, a couple of VMs running a database cluster, etc…

The old way of doing things was to use the **Service Management API**. Each piece of infrastructure was **treated separately** in the Azure Portal. Deployment consisted of manually creating them, or lots of effort creating scripts using PowerShell. You would have to handle trying to initialize infrastructure in serial or parallel yourself. If you wanted to deploy just part of your infrastructure you would have to understand **what was already there** and take actions accordingly, either manually or in your scripts.

Azure Resource Manager Overview:

Azure Resource Manager enables you to work with the resources in your solution as a group. You can deploy, update, or delete all the resources for your solution in a single, coordinated operation.

- **Azure Resource Manager** allows you to define **a declarative template** for your group of resources.
- A resource manager template is **a JavaScript Object Notation (JSON)** file that defines one or more resources to deploy to a resource group.
- You use that template for deployment and it can work for different environments such as **testing, staging, and production**.
- Azure handles things like **parallelizing** as much of the deployment as possible without any extra effort on your part.
- It also deploys in an idempotent way i.e. **incrementally adds** anything **missing** while leaving anything already deployed in place, so you can **rerun** the template deployment multiple times safely.

About Resource Group:

In the Azure Portal things now reside in Resource Groups. Multiple different services and bits of infrastructure can all be **grouped** together under a single Resource Group. For example, if you want to clean up a deployment that consists of 10 different resources, you can simply delete the group and Azure handles removing everything within that group.

There are some important factors to consider when defining your resource group:

1. All the resources in your group should share the same lifecycle. You deploy, update, and delete them together.
2. Each resource can only exist in one resource group.
3. You can add or remove a resource to a resource group at any time.
4. You can move a resource from one resource group to another group or another subscription in same Account.
5. A resource group can contain resources that reside in different regions.
6. A resource group can be used to scope access control for administrative actions.
7. A resource can interact with resources in other resource groups. This interaction is common when the two resources are related but do not share the same lifecycle (for example, web apps connecting to a database).

Note: When creating a resource group, you need to provide a location for that resource group for **storing metadata about the resources**. Therefore, when you specify a location for the resource group, you are specifying where that metadata is stored. For compliance reasons, you may need to ensure that your data is stored in a particular region.

Azure Resource Manager Templates

Resource Manager enables you to export a Resource Manager template from existing resources in your subscription.

It is important to note that there are **two** different ways to export a template:

1. You can **export the actual template that you used for a deployment**. The exported template includes all the parameters and variables exactly as they appeared in the original template. This approach is helpful when you have deployed resources through the portal. Now, you want to see how to construct the template to create those resources.
2. You can **export a template that represents the <u>current state</u> of the resource group**. It creates a template that is a snapshot of the resource group. The exported template has many hard-coded values and probably not as many parameters as you would typically define. This approach is useful when you have modified the resource group through the portal or scripts. Now, you need to capture the resource group as a template.

Option 1: Steps to export the actual template from history:

1. Create a **SQL Database Server** in a **New Resource Group**
2. Resource groups → Select the resource group → Essentials → Deployments (Notice that the blade shows the result of the last deployment)→ Select this link.

3. History of deployments for the group → Select the deployment.
4. The blade displays a summary of the deployment. It includes the status of the deployment and its operations and the values that you provided for parameters. To see the template that you used for the deployment, select **View template**.
5. Resource Manager retrieves the following six files for you to deploy the template:
   a. **Template** - The template that defines the infrastructure for your solution. When you created the SQL Database Server account through the portal, Resource Manager used a template to deploy it and saved that template for future reference.
   b. **Parameters** - A parameter file that you can use to pass in values during deployment. It contains the values that you provided during the first deployment, but you can change any of these values when you redeploy the template.
   c. **CLI** - An Azure command-line-interface (CLI) script file that you can use to deploy the template.
   d. **PowerShell** - An Azure PowerShell script file that you can use to deploy the template.
   e. **.NET** - A .NET class that you can use to deploy the template.
   f. **Ruby** - A Ruby class that you can use to deploy the template
6. Create the **SQL Database**
7. Create the **App Service Plan** and **App Service**
8. Revisit the history of deployments as in step 2.
9. Create Template.json and Parameters.json.
10. Delete the resources created from the resource group.
11. Execute following commands in PowerShell

```
Login-AzureRMAccount
Test-AzureRmResourceGroupDeployment  -ResourceGroupName DemoRG -TemplateFile "D:\template.json" -TemplateParameterFile 'D:\parameters.json'
New-AzureRmResourceGroupDeployment -ResourceGroupName DemoRG -TemplateFile "D:\template.json" -TemplateParameterFile 'D:\parameters.json'
```

Important Changes:

a) Delete duplicate parameter "serverName"
b) Change name of parameter "name" to "webAppName"
c) Provide value for parameter "administratorLoginPassword"
d) Add the following to SQL Database Json:

```
    "dependsOn": [
"[concat('Microsoft.Sql/servers/', parameters('serverName'))]"
     ],
```

Sample: Template.json

```
{
 "$schema": "http://schema.management.azure.com/schemas/2014-04-01-preview/deploymentTemplate.json#",
 "contentVersion": "1.0.0.0",
 "parameters": {
  "administratorLogin": {
   "type": "string"
  },
  "administratorLoginPassword": {
   "type": "securestring"
  },
  "location": {
   "type": "string"
  },
  "serverName": {
   "type": "string"
  },
  "collation": {
   "type": "string"
  },
  "databaseName": {
   "type": "string"
  },
  "edition": {
   "type": "string"
  },
  "requestedServiceObjectiveId": {
   "type": "string"
  },
  "maxSizeBytes": {
   "type": "string"
  },
  "serverLocation": {
   "type": "string"
  },
  "sampleName": {
      "defaultValue": "",
      "type": "string"
    },
    "webAppName": {
      "type": "string"
    },
    "hostingPlanName": {
      "defaultValue": "",
      "type": "string"
    },
    "hostingEnvironment": {
      "type": "string"
    },
    "sku": {
      "type": "string"
    },
    "skuCode": {
      "type": "string"
    },
    "workerSize": {
      "type": "string"
    },
    "serverFarmResourceGroup": {
      "type": "string"
    },
    "subscriptionId": {
      "type": "string"
    }
  },
  "variables": {},
  "resources": [
    {
      "type": "Microsoft.Sql/servers",
      "name": "[parameters('serverName')]",
      "apiVersion": "2015-05-01-preview",
      "location": "[parameters('location')]",
   "properties": {
    "administratorLogin": "[parameters('administratorLogin')]",
    "administratorLoginPassword": "[parameters('administratorLoginPassword')]",
    "version": "12.0"
   },
   "resources": [
    {
     "type": "firewallrules",
     "name": "AllowAllWindowsAzureIps",
     "apiVersion": "2014-04-01-preview",
     "location": "[parameters('location')]",
     "properties": {
      "endIpAddress": "0.0.0.0",
      "startIpAddress": "0.0.0.0"
     },
     "dependsOn": [
      "[concat('Microsoft.Sql/servers/', parameters('serverName'))]"
     ]
    }
   ]
  },
  {
   "type": "Microsoft.Sql/servers/databases",
   "name": "[concat(parameters('serverName'), '/', parameters('databaseName'))]",
   "apiVersion": "2014-04-01-preview",
   "location": "[parameters('serverLocation')]",
   "dependsOn": [
    "[concat('Microsoft.Sql/servers/', parameters('serverName'))]"
   ],
   "properties": {
    "collation": "[parameters('collation')]",
    "edition": "[parameters('edition')]",
    "maxSizeBytes": "[parameters('maxSizeBytes')]",
    "requestedServiceObjectiveId": "[parameters('requestedServiceObjectiveId')]",
    "sampleName": "[parameters('sampleName')]"
   }
  },
  {
   "type": "Microsoft.Web/sites",
   "name": "[parameters('webAppName')]",
   "apiVersion": "2016-03-01",
   "location": "[parameters('location')]",
   "tags": {
    "[concat('hidden-related:', '/subscriptions/', parameters('subscriptionId'),'/resourcegroups/',
parameters('serverFarmResourceGroup'), '/providers/Microsoft.Web/serverfarms/',
parameters('hostingPlanName'))]": "empty"
   },
   "properties": {
    "name": "[parameters('webAppName')]",
    "serverFarmId": "[concat('/subscriptions/', parameters('subscriptionId'),'/resourcegroups/',
parameters('serverFarmResourceGroup'), '/providers/Microsoft.Web/serverfarms/',
parameters('hostingPlanName'))]",
    "hostingEnvironment": "[parameters('hostingEnvironment')]"
   },
   "dependsOn": [
    "[concat('Microsoft.Web/serverfarms/', parameters('hostingPlanName'))]"
   ]
  },
  {
   "type": "Microsoft.Web/serverfarms",
   "sku": {
    "Tier": "[parameters('sku')]",
    "Name": "[parameters('skuCode')]"
   },
   "name": "[parameters('hostingPlanName')]",
   "apiVersion": "2016-09-01",
   "location": "[parameters('location')]",
   "properties": {
    "name": "[parameters('hostingPlanName')]",
    "workerSizeId": "[parameters('workerSize')]",
    "reserved": false,
    "numberOfWorkers": "1",
        "hostingEnvironment": "[parameters('hostingEnvironment')]"
      }
    }
  ]
}
```

Sample: **Parameters.json**

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "administratorLogin": {
      "value": "dssadmin"
    },
    "administratorLoginPassword": {
      "value": "Password@123"
    },
    "location": {
      "value": "eastus"
    },
    "serverName": {
      "value": "dssdemodbserver"
    },
    "collation": {
      "value": "SQL_Latin1_General_CP1_CI_AS"
    },
    "databaseName": {
      "value": "dssdemodb"
    },
    "edition": {
      "value": "Standard"
    },
    "requestedServiceObjectiveId": {
      "value": "789681b8-ca10-4eb0-bdf2-e0b050601b40"
    },
    "maxSizeBytes": {
```

```
    "value": "268435456000"
  },
  "serverLocation": {
    "value": "eastus"
  },
  "sampleName": {
    "value": ""
  },
  "webAppName": {
    "value": "DssDemoWebApp2"
  },
  "hostingPlanName": {
    "value": "DssDemoAppPlan"
  },
  "hostingEnvironment": {
    "value": ""
  },
  "sku": {
    "value": "Basic"
  },
  "skuCode": {
    "value": "B1"
  },
  "workerSize": {
    "value": "0"
  },
  "serverFarmResourceGroup": {
    "value": "DemoRG"
  },
  "subscriptionId": {
    "value": "24784a25-4b3b-4fbe-bd67-045821454fda"
  }
 }
}
```

Option 2: Steps to export a template that represents the current state of the resource group:

1. Resource Group Properties → select **Export Templates**.
2. To **edit** the template, you can now either download the template files locally or you can you can save the template to your library and work on it through the portal.
3. Let's continue with later option of saving to library. When adding a template to the library, give the template a name and description. Then, select **Save**

Customize the Template:

4. To View the template, Go to Left Navigation → More Services → Search for "Template"
5. Select the template with the name you saved.
6. Select **Edit** to customize the template → Select ARM Template Option
7. Add the following parameter to parameters section:

```
"servicelocation": {
        "defaultValue": "South Central US",
        "type": "string",
        "allowedValues": [
            "South Central US",
            "Central US",
            "East US",
            "West US"
        ]
}
```

8. Add the following variable to Variables section

```
"serverfarms_dssdemoappplan_name" :
"[concat(parameters('sites_DssDemoWebApp_name'),'plan')]",
"databases_master_name" :  "[concat(parameters('servers_dssdemoserver_name'),'/master')]",
"firewallRules_AllowAllWindowsAzureIps_name":
"[concat(parameters('servers_dssdemoserver_name'),'/AllowAllWindowsAzureIps')]",
```

9. Replace all occurrences of

```
"location": "South Central US"
with
"location": "[parameters('servicelocation')]"
```

10. Replace all occurrences of

```
parameters('serverfarms_dssdemoappplan_name')
With
variables('serverfarms_dssdemoappplan_name')
```

11. Replace all occurrences of

```
parameters('databases_master_name')
With
variables('databases_master_name')
```

12. Replace all occurrences of

```
parameters('firewallRules_AllowAllWindowsAzureIps_name')
With
variables('firewallRules_AllowAllWindowsAzureIps_name')
```

13. Delete the parameters
    a. 'serverfarms_dssdemoappplan_name',
    b. 'databases_master_name'
    c. 'firewallRules_AllowAllWindowsAzureIps_name'

Deploy the Template using Portal

To Save the template:

1. More Services → Template → +Add
2. Name="DemoTemplate", Description="This is a test template" → OK
3. Copy the content of Template.json and pate in the Text Editor → OK
4. Click Add.

To Execute the Template

5. More Services → Search for "Template"
6. Select the template with the name you saved earlier (DemoTemplate).
7. Template Blade → Deploy
8. Click on Edit parameters → Copy and paste content from Parameters.json → Save
9. Create a New RG and change the parameters as needed → Check I agree . . . → Purchase

Deploy the Template using PowerShell

1. Login to Azure

```
Login-AzureRmAccount
```

2. If you have multiple subscriptions, provide the subscription ID you wish to use for deployment

```
Set-AzureRmContext -SubscriptionID <YourSubscriptionId>
```

3. Create a New Resource Group

```
New-AzureRmResourceGroup -Name ExampleResourceGroup -Location "West US"
```

4. Validate your deployment settings to find problems before creating actual resources.

Test-AzureRmResourceGroupDeployment -ResourceGroupName ExampleResourceGroup -TemplateFile <PathToTemplate>

5. To deploy resources to your resource group

New-AzureRmResourceGroupDeployment -Name ExampleDeployment -ResourceGroupName ExampleResourceGroup -TemplateFile <PathToTemplate>

Use a local parameter file:

New-AzureRmResourceGroupDeployment -Name ExampleDeployment -ResourceGroupName ExampleResourceGroup -**TemplateFile** <PathToTemplate> -**TemplateParameterFile** <PathToParameterFile>

You have the following options for providing parameter values:

New-AzureRmResourceGroupDeployment -Name ExampleDeployment -ResourceGroupName ExampleResourceGroup -TemplateFile <PathToTemplate> **-myParameterName1 "parameterValue1" -myParameterName2 "parameterValue2"**

Use a Parameter object:

$parameters = @{"<ParameterName>"="<Parameter Value>"}

New-AzureRmResourceGroupDeployment -Name ExampleDeployment -ResourceGroupName ExampleResourceGroup -**TemplateFile** <PathToTemplate> -**TemplateParameterObject** $parameters

Deploy the Template using Azure CLI

1. Change to directory where template and parameters file is saved
2. Login to Azure:

   az Login
3. Execute the following to create the resources

```
az group create --name DemoRG --location "South Central US"

az group deployment create \
  --name ExampleDeployment \
  --mode Complete \
  --resource-group DemoRG \
  --template-file template.json \
 --parameter-file parameters.json
```

Note: ***template-file*** parameter can be replaced with ***template-uri*** if the json file is store externally on internet

```
  --template-uri "https://www.sites.com/template.json"
```

Incremental and Complete Deployments

When deploying your resources, you specify that the deployment is either an **incremental** update or a **complete** update.

- In complete mode, Resource Manager **deletes** resources that exist in the resource group but are not specified in the template.
- In incremental mode, Resource Manager **leaves unchanged** resources that exist in the resource group but are not specified in the template.

Existing Resource Group contains:

- Resource A
- Resource B
- Resource C

Template defines:

- Resource A
- Resource B
- Resource D

When deployed in **incremental** mode, the resource group contains:

- Resource A
- Resource B
- Resource C
- Resource D

When deployed in **complete** mode, Resource C is deleted. The resource group contains:

- Resource A
- Resource B
- Resource D

PowerShell Command:

```
New-AzureRmResourceGroupDeployment -Name "ExampleDeployment123" -Mode Complete -ResourceGroupName DemoRG -TemplateFile "D:\template.json" -TemplateParameterFile 'D:\parameters.json'
```

CLI Command:

```
az group deployment create --name ExampleDeployment --mode Complete --resource-group DemoRG --template-file template.json --parameter-file @parameters.json
```