Microsoft

Azure Resource Manager (ARM) Overview

Microsoft Services

Agenda

- Azure Resource Manager (ARM)
- Templates
- Resource Groups
- Resource Providers
- Access Control
- Resource Locks
- Template Security
- Activity Logs

What is Azure Resource Manager (ARM)

- Azure Resource Manager (ARM) is a deployment model that is the successor to Azure Service Management (ASM) and allows you to deploy and manage Azure resources in a more improved way
- Went GA in 2015 and has a modular approach towards resources
- Some of the ASM challenges that are resolved with ARM are
  - Parallel processing
  - Deploy resources using templates
  - Resources are not tied to other resources
  - Role Based Access Control (RBAC)

Azure Resource Manager & JSON

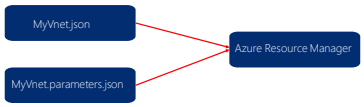
- ARM uses JSON (JavaScript Object Notation) to exchange data between a client and the ARM service.
- JSON is a data format that is used to exchange data between a web browser and a server, but it is less verbose, complex and can deal with highly structured data.

Benefits of Azure Resource Manager

Deploy, manage, and monitor all of the resources for your solution as a group.	Redeploy your solution throughout the development lifecycle.	Manage your infrastructure through templates rather than scripts.	Administrator defined dependencies.	Apply access control to all services in your resource group.

What are ARM Templates?

- An ARM template is a file that contains configuration which is passed to Azure Resource Manager for processing.
- The configuration in an ARM template is written in JSON format and saved with a .json extension.
- ARM templates commonly consist of 6 elements; **$schema**, **contentVersion**, **parameters**, **variables**, **resources** and **outputs**.
- The **parameters** element can **optionally** exist in a separate text file called a parameters file which is saved with a .parameters.json extension.



ARM Template Elements

$schema is the location of the JSON schema file that describes the version of the template language.

contentVersion is an arbitrary number that is used to describe the version of the template.

Parameters are values that are provided when deployment is executed in order to customize resource deployment e.g. "MyStorageAccount".

Variables are values that are provided once but are referenced one or more times within an ARM template in order to simplify template language expressions e.g. "storageAccountName": "StorageAccount1".

Resources are used to define the resource types that are deployed or updated in a resource group e.g. a storage account i.e. Microsoft.Storage/storageAccounts.

Outputs are values that are returned after deployment.

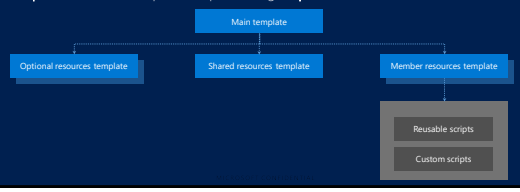
An ARM Template

- An ARM template in it's simplest structure:

```
{
   "$schema": "http://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
   "contentVersion": "",
   "parameters": {  },
   "variables": {  },
   "resources": [  ],
   "outputs": {  }
}
```

ARM Template Linking

- ARM templates can be linked to other templates to break down the deployment into smaller modules that can be useful for testing and reuse.
- A linked template configuration consists of a **main**, **shared resources** and **member resources** template at a minimum.
- An **optional resources** template and pre-existing **scripts** can also be included.



Template Linking Operations & Guidelines

Parameters are passed from a main template to a linked template for processing.	The linked template can also pass an output variable back to the source template, enabling a two-way data exchange between templates.	Use of a local file or a file that is only available on your local network for the linked template is not allowed.	A URI value that includes either http or https to point to the linked template is allowed.	Linked templates can be placed in an Azure storage account, and have it's URI used.

Demo: ARM Templates

Microsoft

Template Authoring

Microsoft Services

Authoring ARM Templates

- ARM templates can be authored using different tools, some of the most common tools in use today are:
  - Visual Studio with Azure SDK
  - Visual Studio Code with Azure extension
  - Azure Portal
  - GitHub
- Template files must be <4MB in size.
- Parameter files must be <64 KB in size.

Visual Studio

Visual Studio Code

- Lightweight code editor
- GA in 2015
- One of the most popular cloud resource deployment tools in use today
- Cross-platform integration
- Support for debugging, embedded Git control and GitHub

Visual Studio

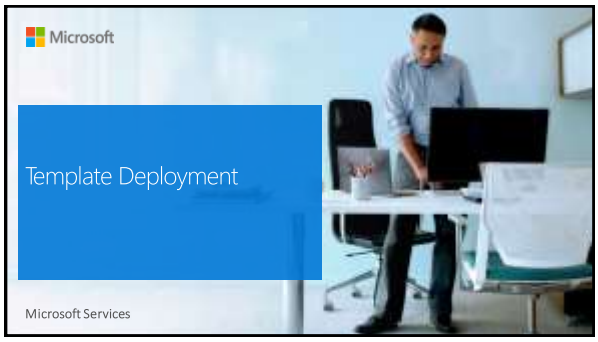
- Full integrated development environment
- Provides 14 predefined ARM templates from it's gallery.
- Automatically populates JSON tags when resources are added, but does not remove variables and parameters when resources are removed.

Azure Portal

- Provides direct access to hundreds of predefined ARM templates in the GitHub gallery that others have created.
- Template parameters can be specified using text boxes.

GitHub

- Access to hundreds of predefined ARM templates that others have created.
- No built in syntax error checking or template validation during authoring.



Incremental Deployments

By default, Resource Manager handles deployments as incremental updates to a resource group.	Leaves unchanged resources that exist in the resource group but are not specified in the template.	Adds resources that are specified in the template but do not exist in the resource group.	Does not re-provision resources that exist in the resource group in the same condition defined in the template.	Re-provisions existing resources that have updated settings in the template.

Complete Deployments

Deletes resources that exist in the resource group but are not specified in the template.	Adds resources that are specified in the template but do not exist in the resource group.	Does not re-provision resources that exist in the resource group in the same condition defined in the template.	Re-provisions existing resources that have updated settings in the template.	Type of deployment specified using the Mode property.

Deploying ARM Templates

- ARM templates can be deployed using different tools, some of the most common tools in use today are:
  - Visual Studio
  - Visual Studio Code
  - Azure Portal
  - GitHub
  - PowerShell
- These tools include the deployment of a resource group prior to the resource being deployed.
- Templates are validated on deployment.

Visual Studio Code

- Template is deployed using PowerShell from within Visual Studio Code.
- Deploy resources using templates locally or redeploy using existing templates.
- Templates are stored locally or in other repositories e.g. Github

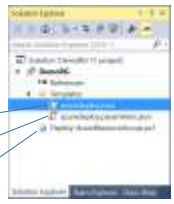
Visual Studio

- Template is deployed using PowerShell from within Visual Studio.
- Deploy resources using templates from the gallery or redeploy using existing templates.
- Templates are stored locally or in other repositories e.g. Github

JSON Template

Optional JSON Parameter file

PowerShell script for deployment



Azure Portal

- Deployed from Marketplace or custom.
- Templates are stored in Azure.
- Templates can be deployed, downloaded and reused.

GitHub

- GitHub deploy to Azure button calls new Azure portal and opens a custom deployment blade.
- Templates are stored in GitHub.

PowerShell With Templates

- New-AzResourceGroup cmdlet creates a new resource group. This is the test, this is the fastest way to do this serioes because htie does not use azure support eot
- New-AzResourceGroupDeployment cmdlet adds an Azure deployment to an existing resource group.
- Test-AzResourceGroupDeployment cmdlet verifies a resource group template prior to deployment.

PowerShell Without Templates

- Deploy resources using PowerShell without templates.
- Resource groups must exist prior to resource deployment.
- New-AzVirtualNetwork -ResourceGroupName TestRG -Name TestVNet `
  -AddressPrefix 192.168.0.0/16 -Location centralus

Demo: Visual Studio Deployment

Microsoft

Lab: Template Authoring & Deployment

Microsoft Services

What are Azure Resource Groups?

A resource is a manageable item that is available through Azure e.g. a virtual machine, storage account, virtual network and so on.

A resource group is a container that holds related resources for an application or service, or resources that you group together.

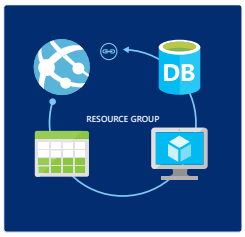
Each resource group can contain a maximum of 800 resources and can not be nested.

Each resource can only exist in one resource group.

Add or remove a resource to a resource group at any time.

A resource group can contain resources that reside in different regions.

Why Azure Resource Groups?



- Management – deploy, update, delete and get status on all resources in a resource group from a single point.
- Flexibility - some resources can be moved between resource groups.
- Portability - resource groups can be exported to templates for easier redeployment.
- Billing – a bill can be retrieved on a per resource group basis.
- Access Control – Permissions can be applied on a per resource group basis.

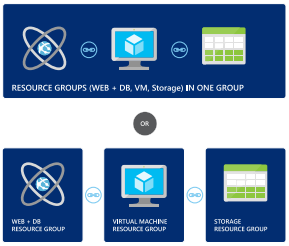
Resource Group Structure

Design:

Should resources be in the same group or a different one?

Consideration:

Do they have common lifecycle and management?



What are Azure Resource Group Tags?

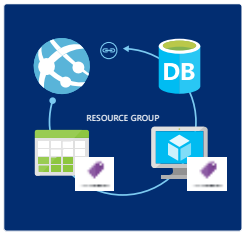
A resource group tag consists of a key/value pair that identifies resources with properties that you define e.g. Sales Project: 1

Each resource group or resource can have a maximum of 50 tags assigned to it.

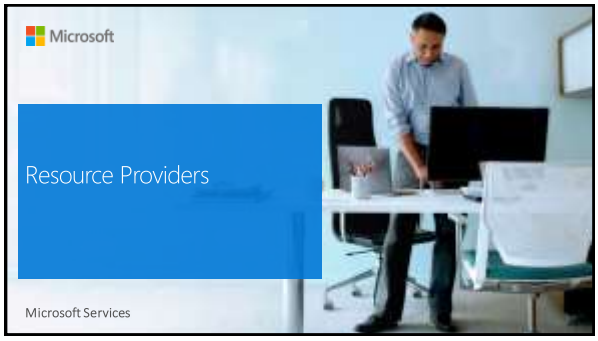
The tag name is limited to 512 characters, and the tag value is limited to 256 characters.

Applied at resource level and are not inherited by child objects i.e. if assigned to a resource group, the resources in that resource group are not tagged.

Why Azure Resource Group Tags?



- Grouping – resources within or outside a resource group can be further grouped by tagging.
- Billing – for supported services, tags can be used to group billing data e.g. a bill for all resources that are tagged environment:test.



What are Resource Providers?

- A resource provider is a service that supplies the resources that you deploy and manage through Azure Resource Manager.
- Each resource provider offers a set of operations for working with a particular resource type.
- Common resource providers are:
  - Microsoft.Compute which supplies the virtual machine resource.
  - Microsoft.Storage which supplies the storage account resource.
  - Microsoft.Network which supplies the virtual network resource.
- Resource providers have different regional availability and apiVersions.

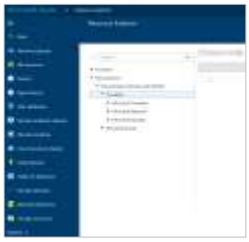
List available Resource Providers

- To list available resource providers, run:

```
Get-AzResourceProvider | Format-Table
```

View Resource Providers used by a Subscription

- In the new Azure portal, browse to Resource Explorer then expand Subscriptions/'Your Subscription Name/Providers.



Resource Provider Parameters

- Resource providers require input parameters in order to execute a task.

Input parameters for the resource being created, read, updated or deleted at a minimum include:

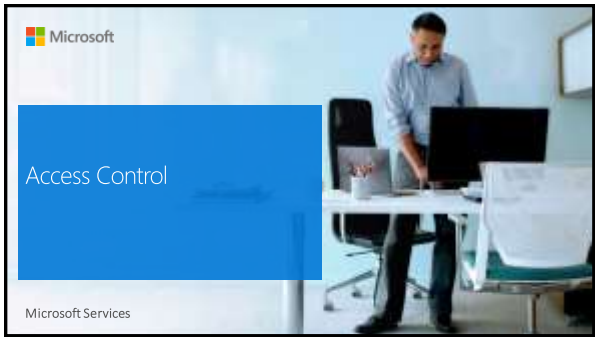
Id	Name
• The Azure wide unique id of the resource. (This includes it's resource group name.)	• The name of the resource e.g. "storageaccount1"
Type	**Location**
• Describes the type of resource e.g. "virtualNetworks"	• Describes the location of the resource e.g. "westeurope"

Example Resource Provider Parameters

- Example parameters to read, update or delete a virtual network resource VNET1.

```
"id": "/subscriptions/SubGUID/resourceGroups/RG1/providers/Microsoft.Network/virtualNetworks/VNet1",
"name": "VNet1",
"type": "Microsoft.Network/virtualNetworks",
"location": "westeurope"
```

Demo: Resource Groups, Resource Tags & View Resource Providers



Azure Policies

- Azure policies allow you to enforce polices during resource deployment e.g. specific VM size, Location and Naming Convention.
- Policies compliment RBAC, RBAC is user focused whereas MP's are resource focused.
- Created and managed using PowerShell or REST API.
- Applied at management group, subscription, resource group or resource level and is inherited by all child resources.
- Policy events are audited and can be viewed in the portal or using PowerShell.
- Policies are cumulative.

Policy Definition structure

- Policy definition is created using JSON.
- Consists of one or more **conditions/logical operators** which define the actions and an **effect** which tells what happens when the conditions are fulfilled.
- A policy contains the following at a minimum:
  - **Condition/Logical operators:** A set of conditions which can be manipulated through a set of logical operators.
  - **Effect:** This describes what the effect will be when the condition is satisfied – deny, audit or append.

Create an Azure Policy

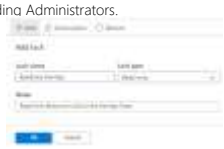
```
 $locationpolicy = New-AzPolicyDefinition -Name regionPolicyDefinition -
Description "Policy to allow resource creation in Central US only" -
Policy '{

  "if" : {
    "not" : {
      "field" : "location",
      "in" : ["centralus"]
    }
  },
  "then" : {
    "effect" : "deny"
  }
}'
```


Assign an Azure Policy

New-AzPolicyAssignment -Name locationPolicyAssignment -PolicyDefinition $locationpolicy -Scope /subscriptions/[YourSubscriptionID]/resourceGroups/ARMPolicies

Resource Locks

- Azure Resource Locks allow you to prevent the accidental deletion or modification of resource groups or resources in your subscription.
- There are 2 resource lock levels: Delete or ReadOnly.
  - Delete means authorized users can still read and modify a resource, but they can't delete it.
  - ReadOnly means authorized users can read from a resource, but they can't delete it or perform any actions on it.
- Applies to Everyone including Administrators.



Resource Lock Management

- Create resource locks using the portal, ARM template, PowerShell or REST API.
- Applied at the subscription, resource group or resource level.
- When a lock is applied at a parent scope, all child resources inherit the same lock, even resources you add later inherit the lock from the parent.
- Owner and User Access Administrator can create and delete resource locks.

Sensitive Information & Templates

- Sensitive information such as VM secrets, certificates and network routing information should not be specified in an ARM template.
- Use Azure Key Vault with Resource Manager to orchestrate and securely store VM secrets and certificates.
- Using Key Vault means that the ARM template references a URI that contains the secrets.
- The loading of secrets into a VM at deployment occurs via direct channel between the Azure Fabric and the Key Vault within the confines of the Microsoft datacenter.
- Maintain separate templates for vault creation and VM deployment.

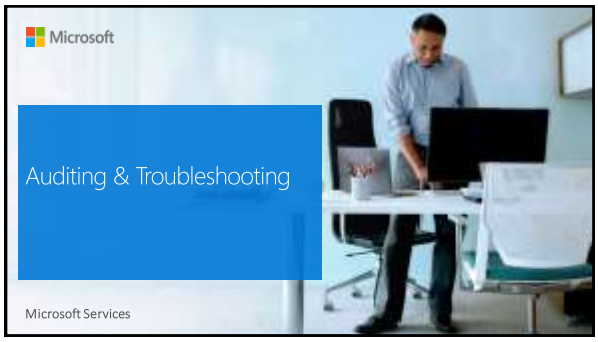
Service Principals for Cross Subscription Access

- Use service principals with role based access control to restrict permission for cross subscription access e.g. a cloud service provider accessing a customer subscription.
- Scope access to a specific resource group or resource e.g. a storage account.
- Grant the most restrictive permission required e.g. read only access.
- Enable auditing on resources that are accessed.
- Use organizational accounts for more control.

Network Information & Templates

- Many scenarios will have requirements that specify how traffic to one or more VM instances in your virtual network is controlled.
- Use a Network Security Group (NSG) to define this part of the ARM template.
- NSG's control all inbound and outbound traffic to a NIC or subnet as opposed to an endpoint based ACL which only works on the public port that is exposed.
- A NIC or subnet can be associated with only 1 NSG and each NSG can contain up to 200 rules.

Demo: Azure policies & Resource Locks

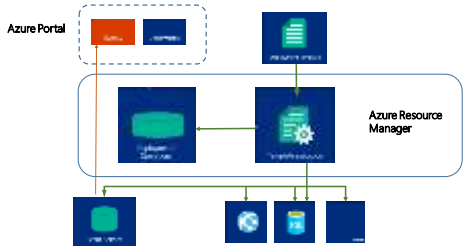


Event Logging Process

- As resources are being deployed, the resource providers will interact with the Azure Event service.
- Event service gathers event information about what is happening behind the scenes.
- As the template execution is taking place, information is being requested from the resource providers, which in turn, are reporting all actions they are taking to the Event service.
- The Event service then notifies the event viewer of operations which are logged and made viewable in the Audit Logs blade.
- Includes all activities that are performed.

Microsoft Confidential

Resource Deployment Flow



Activity Logs

- Use activity logs to find an error when troubleshooting or to monitor how a user in your organization modified a resource.
- Using activity logs, you can determine:
  - What operations were taken on the resources in your subscription.
  - Who initiated the operation (although operations initiated by a backend service do not return a user as the caller).
  - When the operation occurred.
  - The status of the operation.
  - The values of other properties that might help you research the operation.
- The activity log contains all write operations (PUT, POST & DELETE) performed on your resources and does not include read operations (GET).

View Activity Logs

- View activity logs using Azure portal, PowerShell, Azure CLI or REST API.
- Activity logs are retained for 90 days but can only be queried for 15 days or less.
- Use Get-AzLog -ResourceGroup <ResourceGroupName> to view activity logs using PowerShell.