**Deploy Azure policy for Standardization of Resource naming and Tagging using ARM Template**

06/11/2024

**­­Documents Control**

Prepared For:

**Internal Only**

Prepared By:

**Nitin Kansara**

@cobweb.com

**Author**

|  |  |
| --- | --- |
| Author Name | Nitin Kansara |
| Email | Nitin.Kansara@cobweb.com |
| Telephone | +97144553135 |
| Web | [www.cobweb.com/ae](http://www.cobweb.com/ae) |
| Address | Cobweb Solutions Limited  Grosvenor Business Tower  Suite 1911  Barsha Heights  Dubai  P.O. Box 500449  United Arab Emirates |

**Document Validity**

|  |
| --- |
| Printed copy uncontrolled |

**Document History**

| **Version** | **Date** | **Revised By** | **Comments** |
| --- | --- | --- | --- |
| 1.0 | 06/11/2024 | Nitin Kansara | First Draft |
|  |  |  |  |

**Document Distribution**

| **Name** | **Title** | **Date of Issue** | **Version** |
| --- | --- | --- | --- |
|  |  | 06/11/2024 | 1.0 |
|  |  |  |  |

**Copyright & Confidentiality Notice**

*This document is Company Restricted and is not to be copied, distributed, stored, duplicated, or transmitted without the express written permission of Cobweb Solutions Limited Senior Management. All Trademarks are hereby acknowledged. All references to third party products, partners and existing Cobweb Solutions Limited clients should be considered as confidential and may not be distributed without Cobweb Solutions.*

Contents

[Introduction 4](#_Toc187314107)

[Define Roles and Permissions 4](#_Toc187314108)

[Global Admin 4](#_Toc187314109)

[RBAC template 6](#_Toc187314110)

[Considerations for RBAC in Landing Zone 7](#_Toc187314111)

[Steps to Deploy 7](#_Toc187314112)

# Introduction

The governance policies embedded within the template are essential for maintaining compliance and ensuring secure operations. These policies are implemented at various scopes, ensuring that all levels of the Azure environment adhere to the required guidelines. The template also integrates role-based access control (RBAC) to manage permissions effectively and assign roles to users, groups, or service principals. This setup enhances security by ensuring that access is appropriately granted and managed across the organization.

# Prerequisites

Before we begin, ensure you have the following prerequisites in place:

* **Azure Subscription**: Ensure you have an active Azure subscription.
* **Azure CLI or Azure PowerShell**: Install Azure CLI or Azure PowerShell locally or use the Azure Cloud Shell available in the Azure portal.
* **Access**: Ensure you have the necessary permissions to deploy resources in the chosen Azure subscription and resource group.

To streamline the deployment of resources and ensure consistency across the environment, we can create an ARM (Azure Resource Manager) template to enforce naming standards. This policy will help maintain an organized and manageable Azure environment by ensuring that all resources follow predefined naming conventions.

Creating the ARM Template involves defining the policy rule, which specifies the naming standards to be enforced, and then deploying this policy to the desired scope. Below is an example of an ARM Template that enforces naming standards for resources:

JSON

{

"$schema": "[URL]#",

"contentVersion": "1.0.0.0",

"resources": [

{

"type": "Microsoft.Authorization/policyDefinitions",

"apiVersion": "2019-09-01",

"name": "namingPolicy",

"properties": {

"displayName": "Enforce Naming Standards",

"description": "This policy enforces the naming standards for resources.",

"policyType": "Custom",

"mode": "All",

"metadata": {

"category": "Naming Standards"

},

"parameters": {

"prefix": {

"type": "String",

"metadata": {

"displayName": "Prefix",

"description": "Prefix to enforce on resource names."

}

},

"suffix": {

"type": "String",

"metadata": {

"displayName": "Suffix",

"description": "Suffix to enforce on resource names."

}

}

},

"policyRule": {

"if": {

"not": {

"field": "name",

"like": "[concat(parameters('prefix'), '\*', parameters('suffix'))]"

}

},

"then": {

"effect": "deny"

}

}

}

},

{

"type": "Microsoft.Authorization/policyAssignments",

"apiVersion": "2019-09-01",

"name": "namingPolicyAssignment",

"properties": {

"displayName": "Enforce Naming Standards Assignment",

"policyDefinitionId": "[resourceId('Microsoft.Authorization/policyDefinitions', 'namingPolicy')]",

"scope": "[subscription().id]",

"parameters": {

"prefix": {

"value": "proj-"

},

"suffix": {

"value": "-env"

}

}

}

}

]

}

# Setting Up Your Environment

Below is a high-level outline for creating an Azure Policies for standardization of Resource Naming and Tagging. This template provides flexibility for:

* Single VNet
* Hub-and-Spoke architecture
* Single VNet with a third-party NVA
* Single VNet with Azure Firewall
* Hub-and-Spoke architecture with NVA or Azure Firewall

|  |
| --- |
| JSON |
| {  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",  "contentVersion": "1.0.0.0",  "parameters": {  "architectureType": {  "type": "string",  "allowedValues": [  "SingleVNet",  "HubAndSpoke",  "SingleVNetWithNVA",  "SingleVNetWithFirewall",  "HubAndSpokeWithNVAOrFirewall"  ],  "defaultValue": "SingleVNet",  "metadata": {  "description": "Choose the network architecture to deploy."  }  },  "location": {  "type": "string",  "defaultValue": "eastus",  "metadata": {  "description": "Location for the resources."  }  },  "vnetAddressPrefix": {  "type": "string",  "defaultValue": "10.0.0.0/16",  "metadata": {  "description": "Address prefix for the Virtual Network."  }  },  "subnetPrefix": {  "type": "array",  "defaultValue": [  "10.0.1.0/24",  "10.0.2.0/24"  ],  "metadata": {  "description": "Subnet prefixes for the Virtual Network."  }  }  },  "variables": {  "firewallName": "AzureFirewall",  "nvaName": "ThirdPartyNVA",  "hubVnetName": "HubVNet",  "spokeVnetName": "SpokeVNet"  },  "resources": [  {  "condition": "[equals(parameters('architectureType'), 'SingleVNet')]",  "type": "Microsoft.Network/virtualNetworks",  "apiVersion": "2023-02-01",  "name": "SingleVNet",  "location": "[parameters('location')]",  "properties": {  "addressSpace": {  "addressPrefixes": [  "[parameters('vnetAddressPrefix')]"  ]  },  "subnets": [  {  "name": "Subnet1",  "properties": {  "addressPrefix": "[parameters('subnetPrefix')[0]"  }  },  {  "name": "Subnet2",  "properties": {  "addressPrefix": "[parameters('subnetPrefix')[1]"  }  }  ]  }  },  {  "condition": "[equals(parameters('architectureType'), 'HubAndSpoke')]",  "type": "Microsoft.Network/virtualNetworks",  "apiVersion": "2023-02-01",  "name": "[variables('hubVnetName')]",  "location": "[parameters('location')]",  "properties": {  "addressSpace": {  "addressPrefixes": [  "10.1.0.0/16"  ]  },  "subnets": [  {  "name": "HubSubnet",  "properties": {  "addressPrefix": "10.1.1.0/24"  }  }  ]  }  },  {  "condition": "[or(equals(parameters('architectureType'), 'HubAndSpoke'), equals(parameters('architectureType'), 'HubAndSpokeWithNVAOrFirewall'))]",  "type": "Microsoft.Network/virtualNetworks",  "apiVersion": "2023-02-01",  "name": "[variables('spokeVnetName')]",  "location": "[parameters('location')]",  "properties": {  "addressSpace": {  "addressPrefixes": [  "10.2.0.0/16"  ]  },  "subnets": [  {  "name": "SpokeSubnet",  "properties": {  "addressPrefix": "10.2.1.0/24"  }  }  ]  }  },  {  "condition": "[equals(parameters('architectureType'), 'SingleVNetWithNVA')]",  "type": "Microsoft.Network/virtualNetworks",  "apiVersion": "2023-02-01",  "name": "SingleVNetWithNVA",  "location": "[parameters('location')]",  "properties": {  "addressSpace": {  "addressPrefixes": [  "[parameters('vnetAddressPrefix')]"  ]  },  "subnets": [  {  "name": "GatewaySubnet",  "properties": {  "addressPrefix": "10.0.3.0/24"  }  }  ]  }  },  {  "condition": "[equals(parameters('architectureType'), 'SingleVNetWithFirewall')]",  "type": "Microsoft.Network/virtualNetworks",  "apiVersion": "2023-02-01",  "name": "SingleVNetWithFirewall",  "location": "[parameters('location')]",  "properties": {  "addressSpace": {  "addressPrefixes": [  "[parameters('vnetAddressPrefix')]"  ]  },  "subnets": [  {  "name": "AzureFirewallSubnet",  "properties": {  "addressPrefix": "10.0.4.0/24"  }  }  ]  }  }  ]  } |

## Explanation of the ARM Template:

1. **architectureType**

* **Type:** string
* **Allowed Values:**
  + SingleVNet: Deploys a single virtual network.
  + HubAndSpoke: Deploys a hub-and-spoke architecture.
  + SingleVNetWithNVA: Deploys a single VNet with a third-party Network Virtual Appliance (NVA).
  + SingleVNetWithFirewall: Deploys a single VNet with Azure Firewall.
  + HubAndSpokeWithNVAOrFirewall: Deploys a hub-and-spoke architecture with either an NVA or Azure Firewall in the hub.
* **Default Value:** SingleVNet
* **Description:** This parameter allows the user to choose which network architecture to deploy. The template uses conditional logic to deploy the appropriate resources based on this selection.

1. **location**

* **Type:** string
* **Default Value:** eastus
* **Description:** Specifies the Azure region where the resources will be deployed. Users can change it to their desired location.

1. **vnetAddressPrefix**

* **Type:** string
* **Default Value:** 10.0.0.0/16
* **Description:** Defines the address space for the primary virtual network. This is used to create the VNet and its subnets.

1. **subnetPrefix**

* **Type:** array
* **Default Value:**
  + 10.0.1.0/24: For the first subnet.
  + 10.0.2.0/24: For the second subnet.
* **Description:** Specifies the address prefixes for the subnets in the VNet. This array enables you to define multiple subnets within the same address space. Each subnet will use a different portion of the vnetAddressPrefix.

**How the Parameters Work Together**

* The architectureType parameter determines which architecture will be deployed, and based on this, the template uses conditional logic (condition) to include only the relevant resources.
* The location parameter ensures that all resources are deployed in the specified Azure region.
* The vnetAddressPrefix and subnetPrefix define the networking setup (IP ranges) for the VNet and its subnets, providing flexibility for users to customize the network as needed.

## Steps to Deploy

1. **Prerequisites**

* **Azure Subscription**: Ensure you have an active Azure subscription.
* **Azure CLI or Azure PowerShell**: Install Azure CLI or Azure PowerShell locally, or use the Azure Cloud Shell available in the Azure portal.
* **Access**: Ensure you have the necessary permissions to deploy resources in the chosen Azure subscription and resource group.

1. **Save the Template**
2. Copy the ARM template into a JSON file, e.g., vnet-template.json.
3. **Modify Parameters (Optional)**
4. Open the JSON file and adjust default parameter values if needed (e.g., location, address prefixes, etc.).
5. Alternatively, you can create a separate parameters.json file to provide parameter overrides during deployment. For example:

|  |
| --- |
| JSON |
| *{*  *"$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",*  *"contentVersion": "1.0.0.0",*  *"parameters": {*  *"architectureType": {*  *"value": "SingleVNet"*  *},*  *"location": {*  *"value": "westus2"*  *},*  *"vnetAddressPrefix": {*  *"value": "10.0.0.0/16"*  *},*  *"subnetPrefix": {*  *"value": ["10.0.1.0/24", "10.0.2.0/24"]*  *}*  *}*  *}* |

1. **Deploy the Template**

You can deploy the ARM template using either the Azure Portal, Azure CLI, or PowerShell.

**Option 1: Deploy via Azure Portal**

1. Log in to the [Azure Portal](https://portal.azure.com).
2. Navigate to **"Deploy a custom template"**:
   * Search for "Template deployment" in the portal search bar.
   * Click on **"Create"** or **"Build your own template in the editor"**.
3. Upload the vnet-template.json file or paste its content into the editor.
4. Specify the parameters (e.g., architecture type, location) in the portal.
5. Click **Review + Create**, then **Create**.

**Option 2: Deploy via Azure CLI**

**Using Azure CLI**:

|  |
| --- |
| Bash |
| *az deployment group create --resource-group <ResourceGroupName> --template-file vnet-template.json*  or  *az deployment group create --resource-group <ResourceGroupName> --template-file vnet-template.json --parameters @parameters.json* |

**Using PowerShell**:

|  |
| --- |
| Template |
| *New-AzResourceGroup -Name <ResourceGroupName> -Location <Location>*  *New-AzResourceGroupDeployment -ResourceGroupName <ResourceGroupName> -TemplateFile "vnet-template.json"*  ***OR***  *New-AzResourceGroupDeployment -ResourceGroupName <ResourceGroupName> -TemplateFile "vnet-template.json" -TemplateParameterFile "parameters.json"* |

1. **Verify Deployment**

* Navigate to the Azure Portal and open the resource group where the template was deployed.
* Confirm that the VNet(s) and associated resources (subnets, NVAs, Firewall, etc.) were deployed as per the selected architecture.