|  |
| --- |
| Virtual Machines Implementation |

TASK

*Please read all the points below before beginning implementation*

1. Use the ARM template for an initial deployment created in the previous task, which will be used for the linked templates execution.
2. Additionally, to the previously created SA and VNET templates, create the following linked templates:

* a template that will create a Key Vault and a Key Vault Secret resources with your Windows VM credentials. Key Vault Secret will be used for creating the VM credentials during the VM deployment.
* a template that will create Windows Server 2016 Datacenter VM containing network interface, Public IP, NSG rule (with the port 8080 opened) and DSC extension resources.  
    
  For Linux/CLI:
* a template will create Linux VM of your choice containing a network interface, PublicIP, NSG rule (with the port 8080 opened) a Custom Script extension resources.

1. Create a DSC configuration that configures an IIS role, port 8080 binding for the Default Website, opens port 8080 in the Windows Firewall and customizes index.html to display custom text.  
     
   For Linux/CLI:  
     
   Create a Bash/Python script to bootstrap your VM provisioning with Apache web server installation and its configuration to run on port 8080 and index.html customization to display custom text. using the commands appropriate for your Linux distro.
2. Use SAS tokens for accessing ALL templates in this task
3. Key Vault is referenced using Output section in the deployment
4. Key Vault secret is generated in PowerShell/Bash deployment script and passed as a securestring into the ARM template
5. Run deployment of the initial ARM template and deploy required resources.

Result:IIS default web site should be accessible from the Internet via created PublicIP and port 8080

TASK HINTS

1. Take into account, that Key Vault should have an access policy set up granting an access to its secrets to your trainer.
2. Key Vault creation should be performed within the same deployment with all other resources.
3. Use output sections in your VNET and Key Vault ARM templates and reference them during the respective resource’s creation.
4. Use front-end VNET created in the previous step for the VM deployment.
5. All resources VM is depending on should be deployed as well.
6. Use **Standard D2s v3** VM size for this task.
7. Do not use credentials in a plain text. The only allowance is to paste Key Vault secrets, which will be used for the deployment, in ARM template (for the purposes of the Lab ONLY!). Use Key Vault reference for all other use cases, for instance:

"adminPassword": {

"reference": {

"keyVault": {

"id": "/subscriptions/{your\_subscription\_ID}/resourceGroups/{you\_resource\_group\_name}/providers/Microsoft.KeyVault/vaults/{your\_KeyVault\_name}"

},

"secretName": "[variables('adminName')]"

}

}

1. Use the latest available versions of images and extensions.
2. Put your DSC configuration in a Blob. Generate SAS tokens for placing and accessing your configuration. Pass SAS tokens as parameters in your deployment scripts. DO NOT hardcode or pre-generate them!
3. Add your Key Vault, VNET, VM/DSC deployment templates to the initial template as resources.

REQUIREMENTS

1. One JSON file for describing all Azure resources is forbidden. **Please use linked templates.**

2. Linked templates **must be called from the initial deployment template** (main.json).

3. Main and parameters templates must be named as **main.json** and **parameters.json** accordingly.

4. Maximum **number of parameters** in Main.json is 7.

5. Main.json and parameters.json must be **executed from local folder**. Using **-TemplateParameterUri** and **-TemplateUri** options in PS script is forbidden.

6. All artifacts (JSONs and PS files) must be stored in **Azure Storage Account**. Using any GitHubs or other public repos is forbidden.  
7. Create a PowerShell/Bash (using Azure CLI) deployment script, which will be used for running all your deployments. The script should have the following functionality:

                a. Create resource group.

b. Create storage account and container within for artifacts (For example: JSONs, PS file(s), ZIP files).

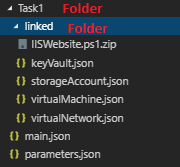
c. Upload the linked templates and other task-related artifacts to the created Storage Account.

d. Execute main.json file for deploy Azure resources.

8. Each ARM json file must have at least **1 output**.

9. All homework **artifacts must be executable** (e.g. if Mentor starts your script execution and it fails - all homework artifacts will be sent back for fixing)

10. All **resources must be deleted** after homework completion.

11. Use the next **folder structure** for storing artifacts. **Subfolder** with resources JSONs must be named **“linked”:**  


Useful links

* + [Microsoft.KeyVault vaults template reference](https://docs.microsoft.com/en-us/azure/templates/microsoft.keyvault/2018-02-14/vaults)

# [Outputs section in Azure Resource Manager templates](https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-manager-templates-outputs)

# [Using linked and nested templates when deploying Azure resources](https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-linked-templates#link-or-nest-a-template)

* + [Deploy private Resource Manager template with SAS token and Azure PowerShell](https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-manager-powershell-sas-token)