Automating Workloads with ARM Templates

Project for CalTech Az303 Course

David Hill, Jr.

Description

The Rand Enterprises Corporation wants to test the ARM template to bring infrastructure as code into practice. They have decided to work on project RandEnt to verify the functionality.

The operations team at Rand decides to define entire networking architecture using the ARM template, once that's in place they intended to create the storage account along with virtual machine housing their application.

As Rand Enterprises works extensively on delivering Image-based content for their global audience in a secure way by avoiding the Azure Storage account access to the public internet. The communication from the application in the VM to the Azure Storage account must take place via the internal network of Azure.

The expectation of the operation team is to Rather than deploying resources in Azure independently, they should leverage Azure ARM templates to deploy and provision all resources in templatize format.

Following requirements should be met:

- ☒ Define the Network
- ☒ Extend the network with Compute and Storage
- ☒ Create the Storage account container for Images & configure service endpoints

Define the Network

The first part of this project was defining the network which I did by using the azure code snippets and modifying the values. I made the names of every resource in this project a configurable parameter so that the template can be reused with different values. In this network I created 2 subnets named "Subnet-1" and "Subnet-2" with an address space of "10.0.0.0/16". See the network object I used below:

```
{
    "name": "[parameters('networkName')]",
    "type": "Microsoft.Network/virtualNetworks",
    "apiVersion": "2019-11-01",
    "location": "[resourceGroup().location]",
    "tags": {
```

```
        "displayName": "[parameters('networkName')]"
    },
    "properties": {
        "addressSpace": {
            "addressPrefixes": [
                "10.0.0.0/16"
            ]
        },
        "subnets": [
            {
                "name": "Subnet-1",
                "properties": {
                    "addressPrefix": "10.0.0.0/24"
                }
            },
            {
                "name": "Subnet-2",
                "properties": {
                    "addressPrefix": "10.0.1.0/24"
                }
            }
        ]
    }
}
```

For this example I set networkName = randent-vnet.

Extend the network with Compute and Storage

To extend the network with Compute and Storage I first used the code snippets to generate a storage account object. I then added a blob container resource in the object to be created with te storage account. The code to create the storage account and blob container is as follows:

```
{
    "name": "[parameters('storageAcctName')]",
    "type": "Microsoft.Storage/storageAccounts",
    "apiVersion": "2019-06-01",
    "tags": {
        "displayName": "[parameters('storageAcctName')]"
    },
    "location": "[resourceGroup().location]",
    "kind": "StorageV2",
    "sku": {
        "name": "Standard_LRS",
        "tier": "Standard"
    },

    "resources": [
```

```
        {
            "type": "blobServices/containers",
            "apiVersion": "2019-06-01",
            "name": "[concat('default/',
                    parameters('blobContainerName'))]",
            "dependsOn": [
                "[parameters('storageAcctName')]"
            ],
            "properties": {
                "publicAccess": "Container"
            }
        }
    ]
}
```

Next was to add compute, to do that I used the azure code snippets one again to create a virtual machine object. Along with this, I also created network security group, public IP, and network interface objects. See the code I used below:

```
{
    "name": "[parameters('publicIPName')]",
    "type": "Microsoft.Network/publicIPAddresses",
    "apiVersion": "2019-11-01",
    "location": "[resourceGroup().location]",
    "tags": {
        "displayName": "[parameters('publicIPName')]"
    },
    "properties": {
        "publicIPAllocationMethod": "Dynamic",
        "dnsSettings": {
            "domainNameLabel": "[toLower('randent')]"
        }
    }
},
{
    "name": "[parameters('nsgName')]",
    "type": "Microsoft.Network/networkSecurityGroups",
    "apiVersion": "2018-08-01",
    "location": "[resourceGroup().location]",
    "properties": {
        "securityRules": [
            {
                "name": "nsgRule1",
                "properties": {
                    "description": "description",
                    "protocol": "Tcp",
                    "sourcePortRange": "*",
                    "destinationPortRange": "3389",
                    "sourceAddressPrefix": "*",
```

```
                    "destinationAddressPrefix": "*",
                    "access": "Allow",
                    "priority": 100,
                    "direction": "Inbound"
                }
            }
        ]
    }
},
{
    "name": "[parameters('networkInterfaceName')]",
    "type": "Microsoft.Network/networkInterfaces",
    "apiVersion": "2019-11-01",
    "location": "[resourceGroup().location]",
    "tags": {
        "displayName": "[parameters('networkInterfaceName')]"
    },
    "dependsOn": [
        "[resourceId('Microsoft.Network/virtualNetworks',
                parameters('networkName'))]"
    ],
    "properties": {
        "ipConfigurations": [
            {
                "name": "ipConfig1",
                "properties": {
                    "privateIPAllocationMethod": "Dynamic",
                    "subnet": {
                        "id": "[resourceId(
                            'Microsoft.Network/virtualNetworks/subnets',
                            parameters('networkName'), 'Subnet-1')]"
                    }
                }
            }
        ]
    }
},
{
    "name": "[parameters('vmName')]",
    "type": "Microsoft.Compute/virtualMachines",
    "apiVersion": "2019-07-01",
    "location": "[resourceGroup().location]",
    "dependsOn": [
        "[resourceId('Microsoft.Storage/storageAccounts',
                toLower(parameters('storageAcctName')))]",
        "[resourceId('Microsoft.Network/networkInterfaces',
                parameters('networkInterfaceName'))]"
    ],
```

```
        "tags": {
            "displayName": "randent"
        },
        "properties": {
            "hardwareProfile": {
                "vmSize": "Standard_B1ms"
            },
            "osProfile": {
                "computerName": "randent",
                "adminUsername": "adminUser",
                "adminPassword": "*********"
            },
            "storageProfile": {
                "imageReference": {
                    "publisher": "MicrosoftWindowsServer",
                    "offer": "WindowsServer",
                    "sku": "2016-Datacenter",
                    "version": "latest"
                },
                "osDisk": {
                    "name": "randentOSDisk",
                    "caching": "ReadWrite",
                    "createOption": "FromImage"
                }
            },
            "networkProfile": {
                "networkInterfaces": [
                    {
                        "id": "[resourceId(
                            'Microsoft.Network/networkInterfaces',
                            parameters('networkInterfaceName'))]"
                    }
                ]
            },
            "diagnosticsProfile": {
                "bootDiagnostics": {
                    "enabled": true,
                    "storageUri": "[reference(resourceId(
                        'Microsoft.Storage/storageAccounts/',
                        toLower(parameters('storageAcctName')))
                        ).primaryEndpoints.blob]"
                }
            }
        }
    }
```

To deploy these resources I first had to login to my azure account with the command:

```
az login
```

I then created a resource group with the command:

```
az group create --name RandEnt --location eastus
```

This command creates a resource group named RandEnt in the East US location

Finally, to deploy the azure resources I ran the following command:

```
az deployment group create
    --resource-group RandEnt
    --template-file AZ303_Project_Template.json
    --parameters AZ303_Project_Template.parameters.json
```

After a few minutes the resources were created successfully, in the figure below you can see the resources in the resource group.

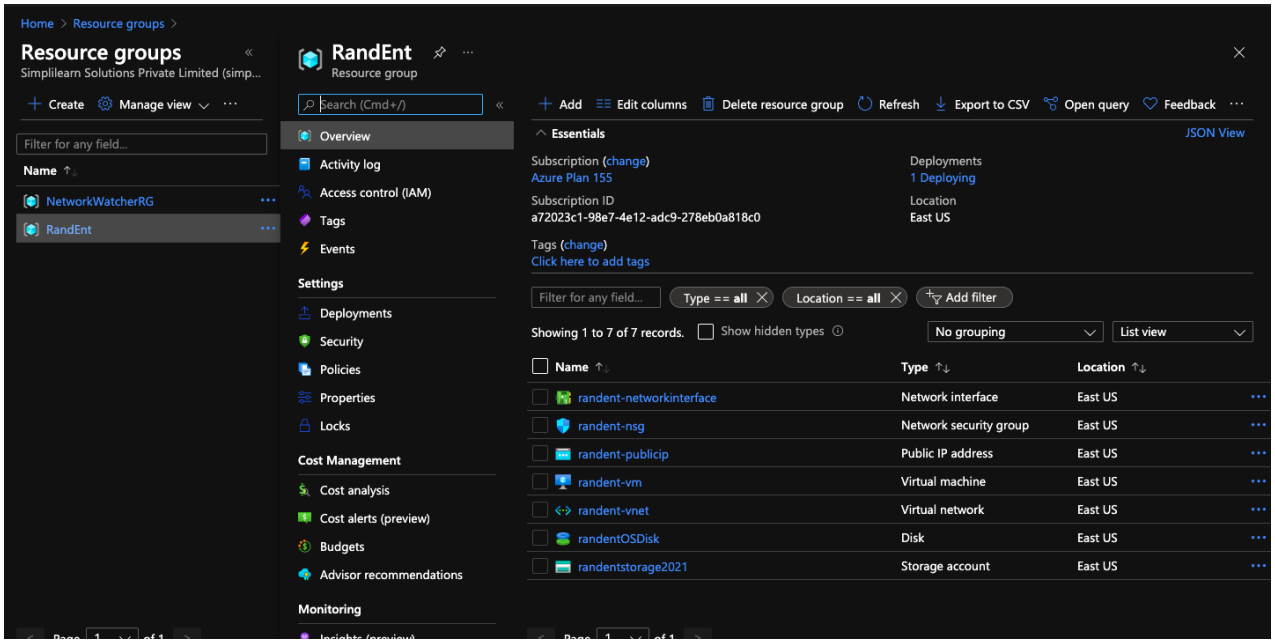

Figure 1: Azure Resources After Deployment from ARM Template

Create the Storage account container for Images & configure service endpoints

Now that our storage account container is created the only thing to do now is to configure the service endpoints. We can do that easily in the Azure portal. First I went to my storage account and uploaded an image for testing. Then, I went to the Networking section in my storage account to configure the service endpoints. To do this I set “Allow access from” “All networks” to “Selected networks” so that our images are not viewable from the public internet. I then added the randent-vnet network we created in our ARM deploymentg and gave access on Subnet-1. The confirmation of this is in the figure below:

Next we have to attach our network security group to the randent-vm and make sure it is open to the RDP port 3389 so we can access it remotely. I also had to make sure that the randent-network interface was connected to the randent network security

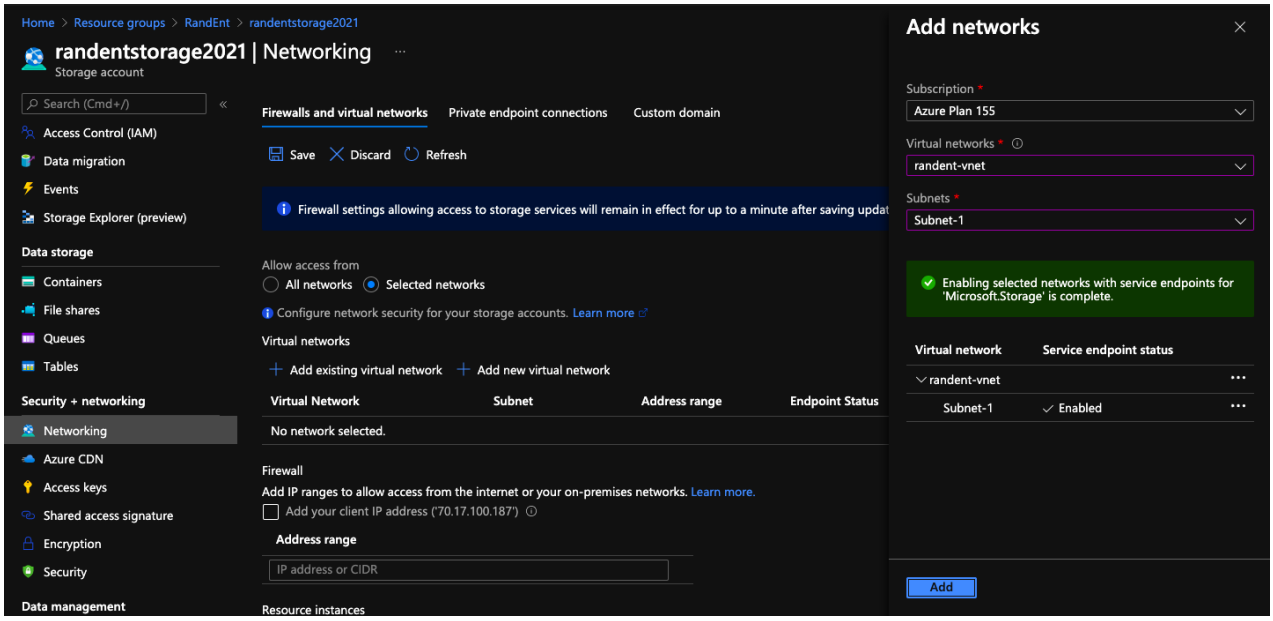

Figure 2: Confirmation of configured Service Endpoints

group. I then assigned the Public IP I created to the randnet-vm. After that I was able to connect to the VM through RDP.

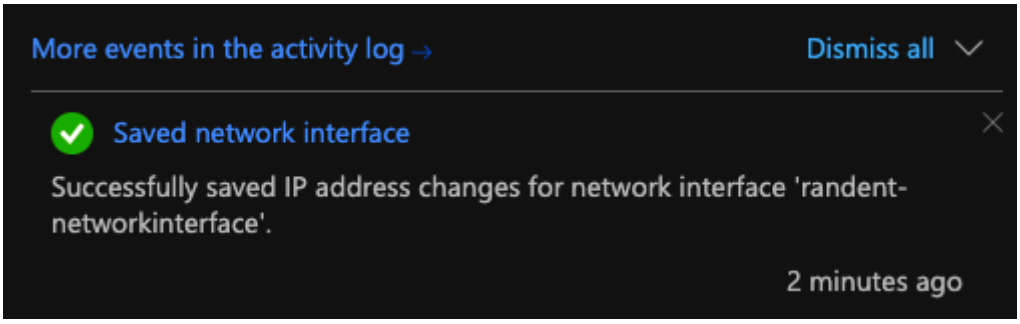

Figure 3: Assigning the Public IP to the VM

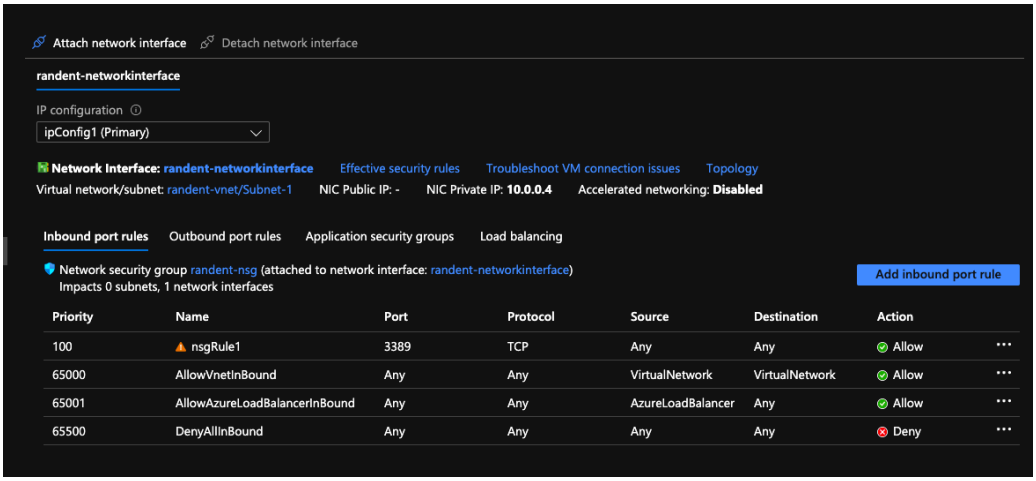

Figure 4: Configuring the Network Interface

Finally we get our result which is the ability to access the image from the VM.

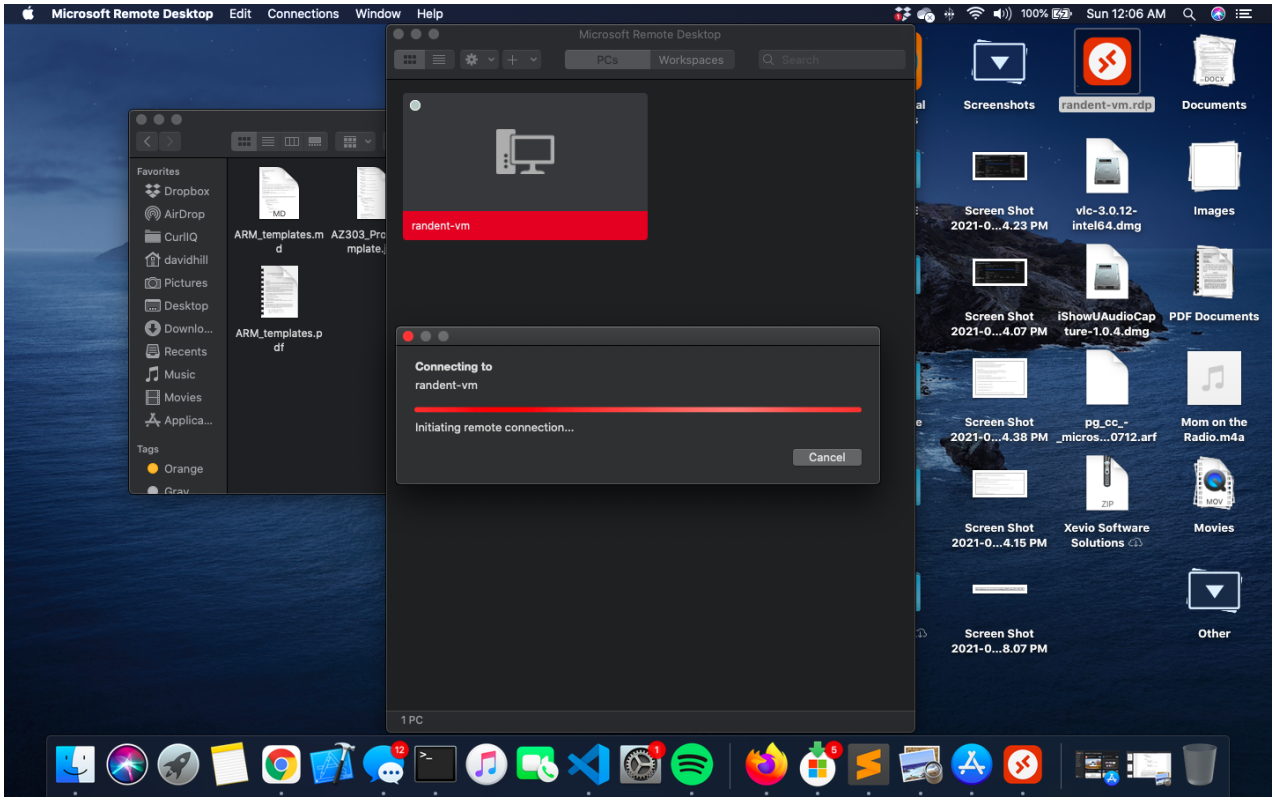

Figure 5: Connecting to the VM

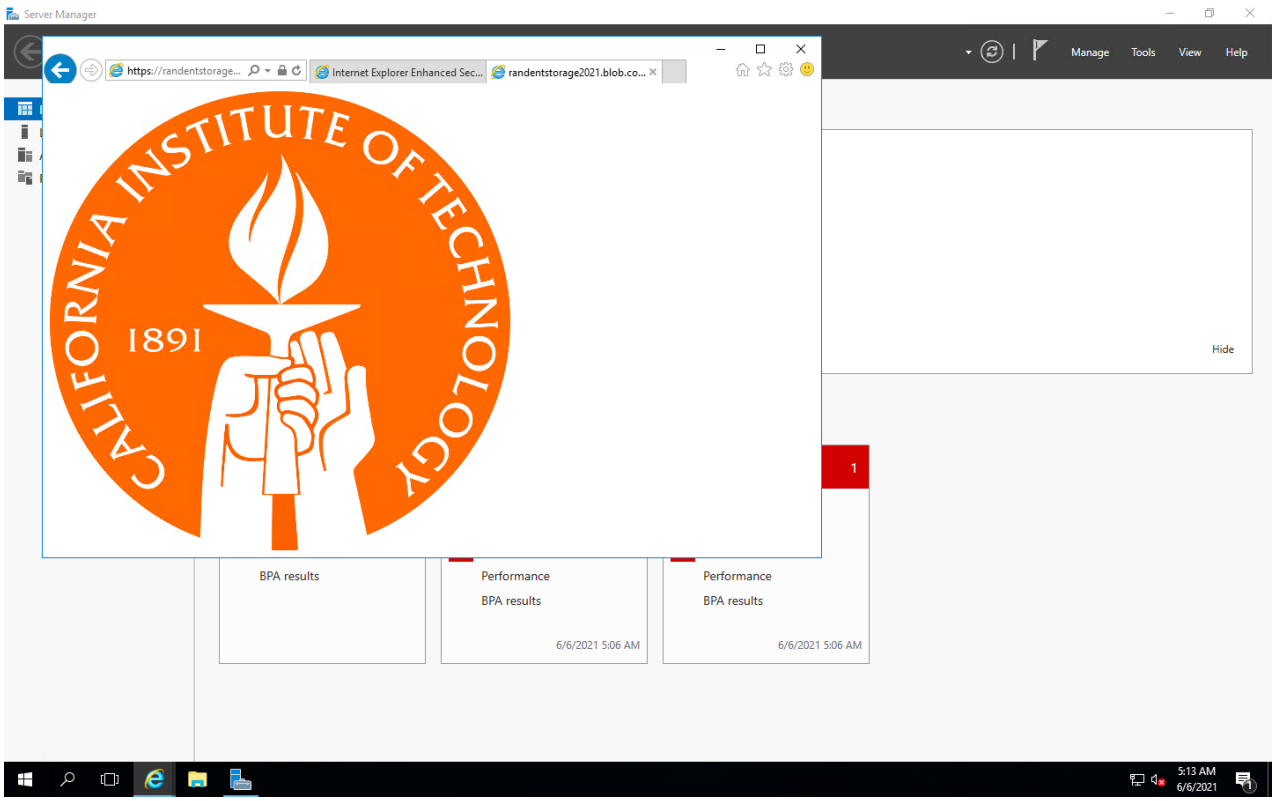

Figure 6: Accessing the Blob Image from the VM on the RandEnt network