**PROJECT**

**Automating Workloads with ARM Templates**

Project 1

DESCRIPTION

**Business Scenario**

The Rand Enterprises Corporation wants to test 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 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, they are seeking to improve the performance on that aspect. To facilitate the same, they have decided to introduce Azure CDN of Standard\_Akamai SKU.

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.

**Overview**

The main tasks for this exercise are as follows:

1. Define the network   
2. Extend that with Compute & Storage  
3. Create the Storage account for Images & implement CDN

FOLLOWING STEPS TO ACHIEVE ABOVE GOAL

Run Window PowerShell

Now connect to Azure account by login process by command in opened powershell window

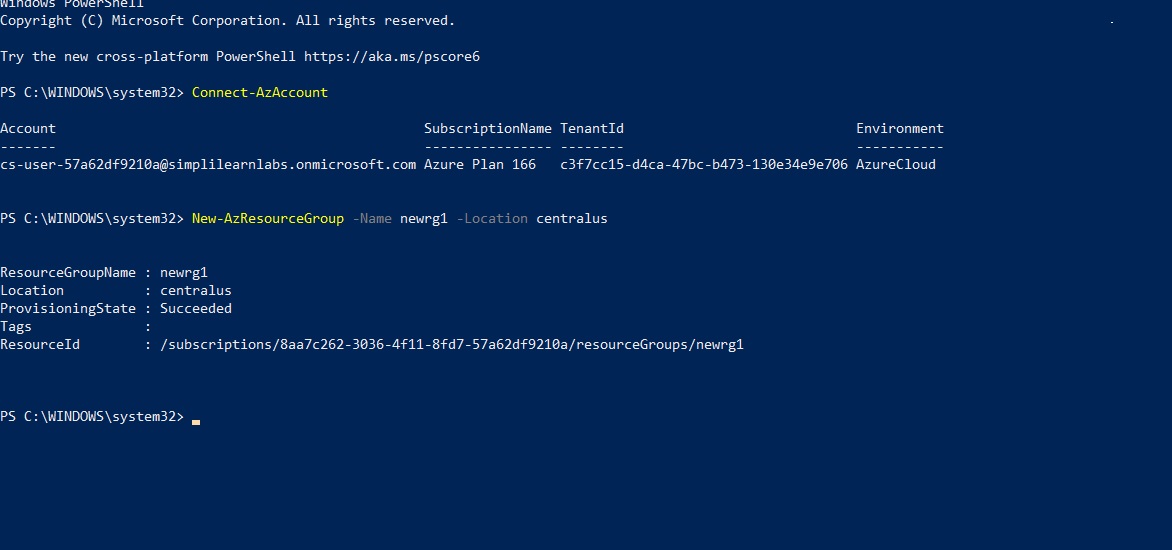
* Connect-AzAccount

Login window opens. Fill username and password to get log in.

STEP 1: CREATING RESOURCE GROUP

Run command to create resource group

* New-AzResourceGroup -Name newrg1 -Location centralus



Step 2: CREATING VIRTUAL MACHINE USING ARM TEMPLATE IS FOLLOWS

**TEMPLATE.JSON**

{

  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",

  "contentVersion": "1.0.0.0",

  "parameters": {

    "adminUsername": {

      "type": "string",

      "metadata": {

        "description": "Username for the Virtual Machine."

      }

    },

    "adminPassword": {

      "type": "securestring",

      "minLength": 12,

      "metadata": {

        "description": "Password for the Virtual Machine."

      }

    },

    "dnsLabelPrefix": {

      "type": "string",

      "defaultValue": "[toLower(concat(parameters('vmName'),'-', uniqueString(resourceGroup().id, parameters('vmName'))))]",

      "metadata": {

        "description": "Unique DNS Name for the Public IP used to access the Virtual Machine."

      }

    },

    "publicIpName": {

      "type": "string",

      "defaultValue": "myPublicIP",

      "metadata": {

        "description": "Name for the Public IP used to access the Virtual Machine."

      }

    },

    "publicIPAllocationMethod": {

      "type": "string",

      "defaultValue": "Dynamic",

      "allowedValues": [

        "Dynamic",

        "Static"

      ],

      "metadata": {

        "description": "Allocation method for the Public IP used to access the Virtual Machine."

      }

    },

    "publicIpSku": {

      "type": "string",

      "defaultValue": "Basic",

      "allowedValues": [

        "Basic",

        "Standard"

      ],

      "metadata": {

        "description": "SKU for the Public IP used to access the Virtual Machine."

      }

    },

    "OSVersion": {

      "type": "string",

      "defaultValue": "2016-Datacenter",

      "allowedValues": [

        "2008-R2-SP1",

        "2012-Datacenter",

        "2012-R2-Datacenter",

        "2016-Nano-Server",

        "2016-Datacenter-with-Containers",

        "2016-Datacenter",

        "2019-Datacenter",

        "2019-Datacenter-Core",

        "2019-Datacenter-Core-smalldisk",

        "2019-Datacenter-Core-with-Containers",

        "2019-Datacenter-Core-with-Containers-smalldisk",

        "2019-Datacenter-smalldisk",

        "2019-Datacenter-with-Containers",

        "2019-Datacenter-with-Containers-smalldisk"

      ],

      "metadata": {

        "description": "The Windows version for the VM. This will pick a fully patched image of this given Windows version."

      }

    },

    "vmSize": {

      "type": "string",

      "defaultValue": "Standard\_B2S",

      "metadata": {

        "description": "Size of the virtual machine."

      }

    },

    "location": {

      "type": "string",

      "defaultValue": "[resourceGroup().location]",

      "metadata": {

        "description": "Location for all resources."

      }

    },

    "vmName": {

      "type": "string",

      "defaultValue": "simple-vm",

      "metadata": {

        "description": "Name of the virtual machine."

      }

    }

  },

  "variables": {

    "storageAccountName": "[concat('bootdiags', uniquestring(resourceGroup().id))]",

    "nicName": "myVMNic",

    "addressPrefix": "10.0.0.0/16",

    "subnetName": "Subnet",

    "subnetPrefix": "10.0.0.0/24",

    "virtualNetworkName": "MyVNET",

    "subnetRef": "[resourceId('Microsoft.Network/virtualNetworks/subnets', variables('virtualNetworkName'), variables('subnetName'))]",

    "networkSecurityGroupName": "default-NSG"

  },

  "resources": [

    {

      "type": "Microsoft.Storage/storageAccounts",

      "apiVersion": "2019-06-01",

      "name": "[variables('storageAccountName')]",

      "location": "[parameters('location')]",

      "sku": {

        "name": "Standard\_LRS"

      },

      "kind": "Storage",

      "properties": {}

    },

    {

      "type": "Microsoft.Network/publicIPAddresses",

      "apiVersion": "2020-06-01",

      "name": "[parameters('publicIPName')]",

      "location": "[parameters('location')]",

      "sku": {

        "name": "[parameters('publicIpSku')]"

      },

      "properties": {

        "publicIPAllocationMethod": "[parameters('publicIPAllocationMethod')]",

        "dnsSettings": {

          "domainNameLabel": "[parameters('dnsLabelPrefix')]"

        }

      }

    },

    {

      "type": "Microsoft.Network/networkSecurityGroups",

      "apiVersion": "2020-06-01",

      "name": "[variables('networkSecurityGroupName')]",

      "location": "[parameters('location')]",

      "properties": {

        "securityRules": [

          {

            "name": "default-allow-3389",

            "properties": {

              "priority": 1000,

              "access": "Allow",

              "direction": "Inbound",

              "destinationPortRange": "3389",

              "protocol": "Tcp",

              "sourcePortRange": "\*",

              "sourceAddressPrefix": "\*",

              "destinationAddressPrefix": "\*"

            }

          }

        ]

      }

    },

    {

      "type": "Microsoft.Network/virtualNetworks",

      "apiVersion": "2020-06-01",

      "name": "[variables('virtualNetworkName')]",

      "location": "[parameters('location')]",

      "dependsOn": [

        "[resourceId('Microsoft.Network/networkSecurityGroups', variables('networkSecurityGroupName'))]"

      ],

      "properties": {

        "addressSpace": {

          "addressPrefixes": [

            "[variables('addressPrefix')]"

          ]

        },

        "subnets": [

          {

            "name": "[variables('subnetName')]",

            "properties": {

              "addressPrefix": "[variables('subnetPrefix')]",

              "networkSecurityGroup": {

                "id": "[resourceId('Microsoft.Network/networkSecurityGroups', variables('networkSecurityGroupName'))]"

              }

            }

          }

        ]

      }

    },

    {

      "type": "Microsoft.Network/networkInterfaces",

      "apiVersion": "2020-06-01",

      "name": "[variables('nicName')]",

      "location": "[parameters('location')]",

      "dependsOn": [

        "[resourceId('Microsoft.Network/publicIPAddresses', parameters('publicIPName'))]",

        "[resourceId('Microsoft.Network/virtualNetworks', variables('virtualNetworkName'))]"

      ],

      "properties": {

        "ipConfigurations": [

          {

            "name": "ipconfig1",

            "properties": {

              "privateIPAllocationMethod": "Dynamic",

              "publicIPAddress": {

                "id": "[resourceId('Microsoft.Network/publicIPAddresses', parameters('publicIPName'))]"

              },

              "subnet": {

                "id": "[variables('subnetRef')]"

              }

            }

          }

        ]

      }

    },

    {

      "type": "Microsoft.Compute/virtualMachines",

      "apiVersion": "2020-06-01",

      "name": "[parameters('vmName')]",

      "location": "[parameters('location')]",

      "dependsOn": [

        "[resourceId('Microsoft.Storage/storageAccounts', variables('storageAccountName'))]",

        "[resourceId('Microsoft.Network/networkInterfaces', variables('nicName'))]"

      ],

      "properties": {

        "hardwareProfile": {

          "vmSize": "[parameters('vmSize')]"

        },

        "osProfile": {

          "computerName": "[parameters('vmName')]",

          "adminUsername": "[parameters('adminUsername')]",

          "adminPassword": "[parameters('adminPassword')]"

        },

        "storageProfile": {

          "imageReference": {

            "publisher": "MicrosoftWindowsServer",

            "offer": "WindowsServer",

            "sku": "[parameters('OSVersion')]",

            "version": "latest"

          },

          "osDisk": {

            "createOption": "FromImage",

            "managedDisk": {

              "storageAccountType": "Standard\_LRS"

            }

          },

        },

        "networkProfile": {

          "networkInterfaces": [

            {

              "id": "[resourceId('Microsoft.Network/networkInterfaces', variables('nicName'))]"

            }

          ]

        },

        "diagnosticsProfile": {

          "bootDiagnostics": {

            "enabled": true,

            "storageUri": "[reference(resourceId('Microsoft.Storage/storageAccounts', variables('storageAccountName'))).primaryEndpoints.blob]"

          }

        }

      }

    }

  ],

  "outputs": {

    "hostname": {

      "type": "string",

      "value": "[reference(parameters('publicIPName')).dnsSettings.fqdn]"

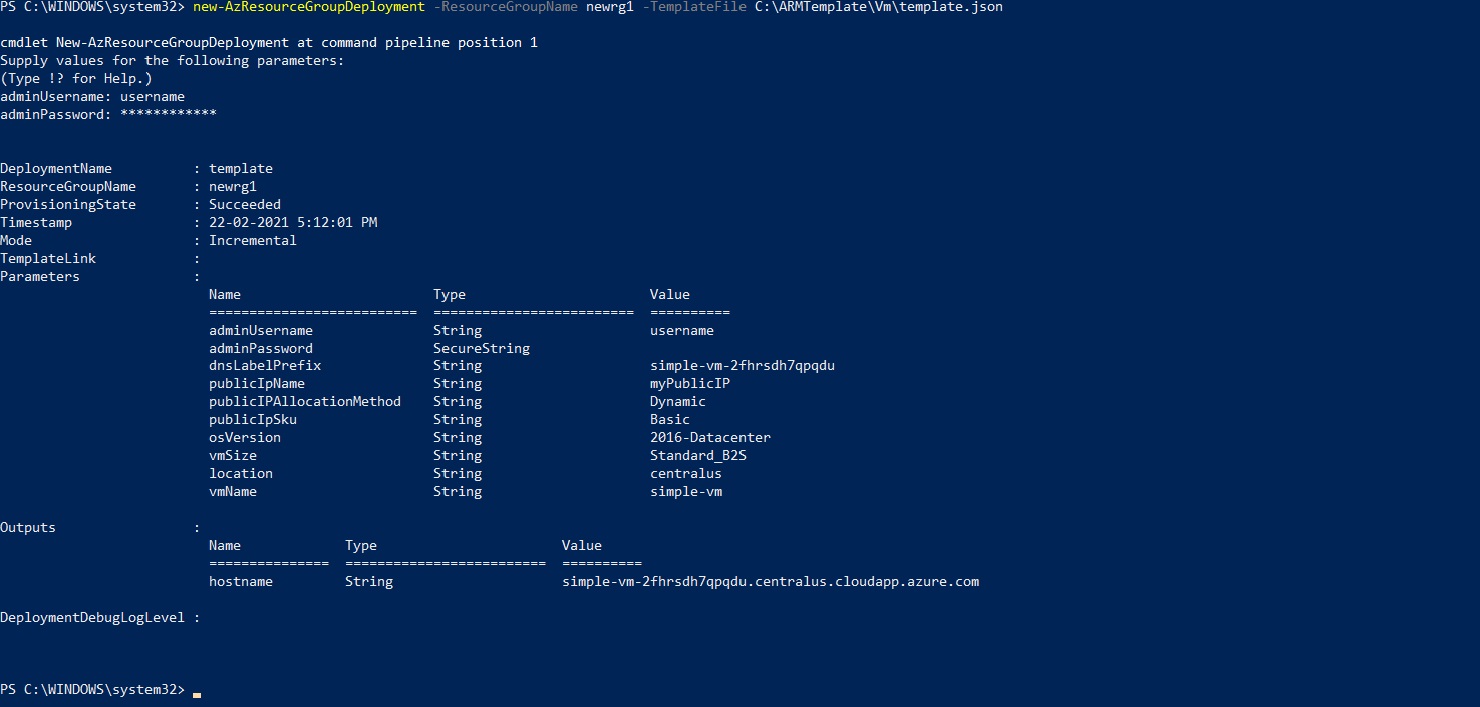
    }

  }

}

IN POWER SHELL EXECUTION FOLLOWING COMMAND

new-AzResourceGroupDeployment -ResourceGroupName newrg1 -TemplateFile C:\ARMTemplate\Vm\template.json



Following parameters needed to fill are

Adminusername : username

Adminpassword : goto@1234567

It create vm at backend on azure portal.

STEP 3: CREATING STORAGE ACCOUNT BY ARM TEMPLATE

FOLLOWING COMMAND EXECUTE AT POERSHELL

new-AzResourceGroupDeployment -ResourceGroupName newrg1 -TemplateFile C:\ARMTemplate\storage\template.json

**TEMPLATE.JSON**

{

  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",

  "contentVersion": "1.0.0.0",

  "parameters": {

    "location": {

      "type": "string"

    },

    "accountType": {

      "type": "string"

    },

    "kind": {

      "type": "string"

    },

    "accessTier": {

      "type": "string"

    },

    "supportsHttpsTrafficOnly": {

      "type": "bool"

    }

  },

  "variables": {

    "storageAccountName": "proj303arm"

  },

  "resources": [

    {

      "name": "[variables('storageAccountName')]",

      "type": "Microsoft.Storage/storageAccounts",

      "apiVersion": "2018-07-01",

      "location": "[parameters('location')]",

      "properties": {

        "accessTier": "[parameters('accessTier')]",

        "supportsHttpsTrafficOnly": "[parameters('supportsHttpsTrafficOnly')]"

      },

      "dependsOn": [

      ],

      "sku": {

        "name": "[parameters('accountType')]"

      },

      "kind": "[parameters('kind')]"

    }

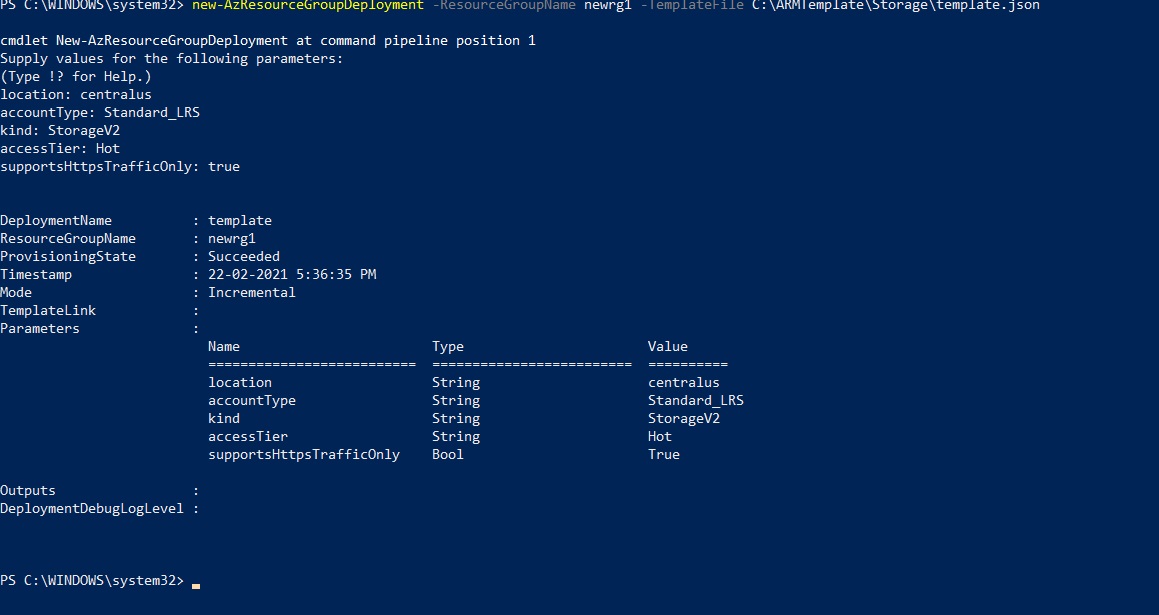
  ],

  "outputs": {

  }

}

**In powershell execution:**



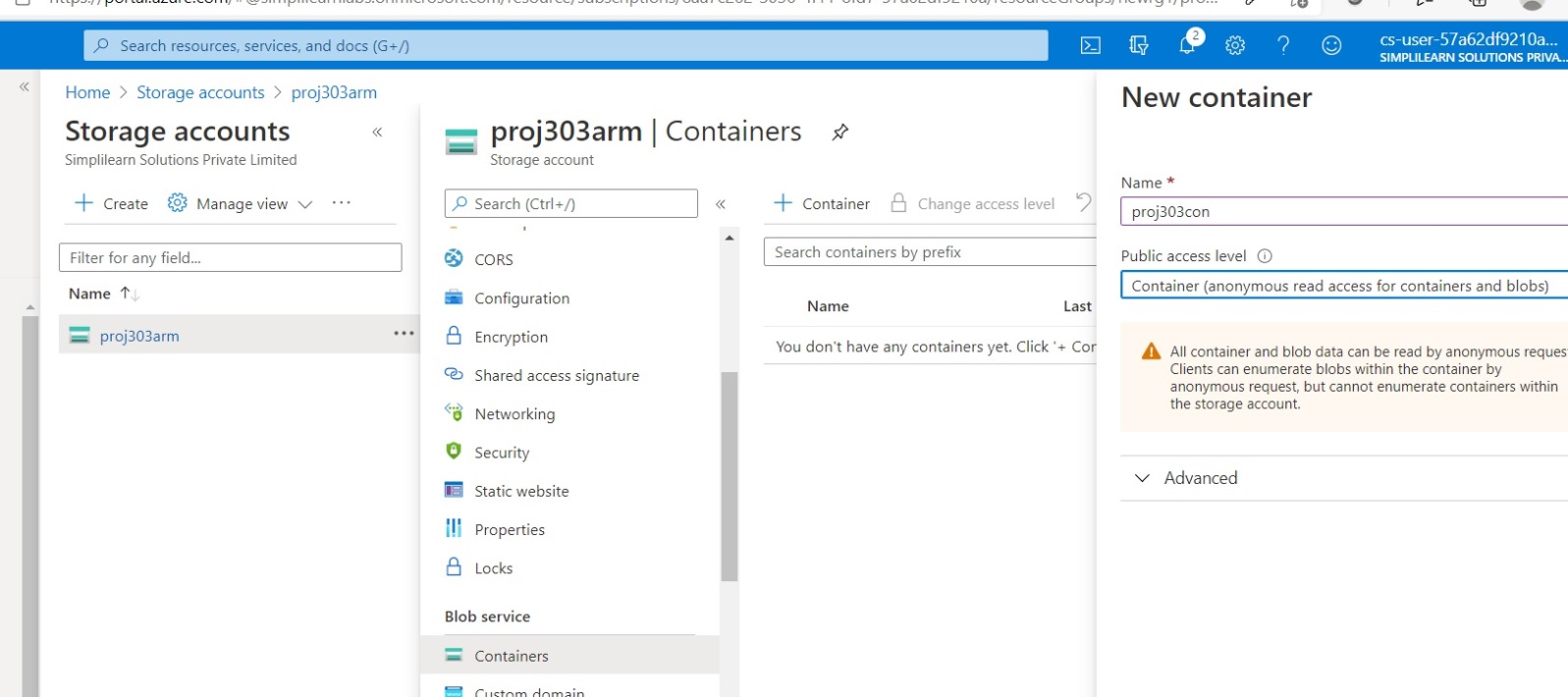
FOLLOWING PARAMETERS ARE NEEDED TO FILL

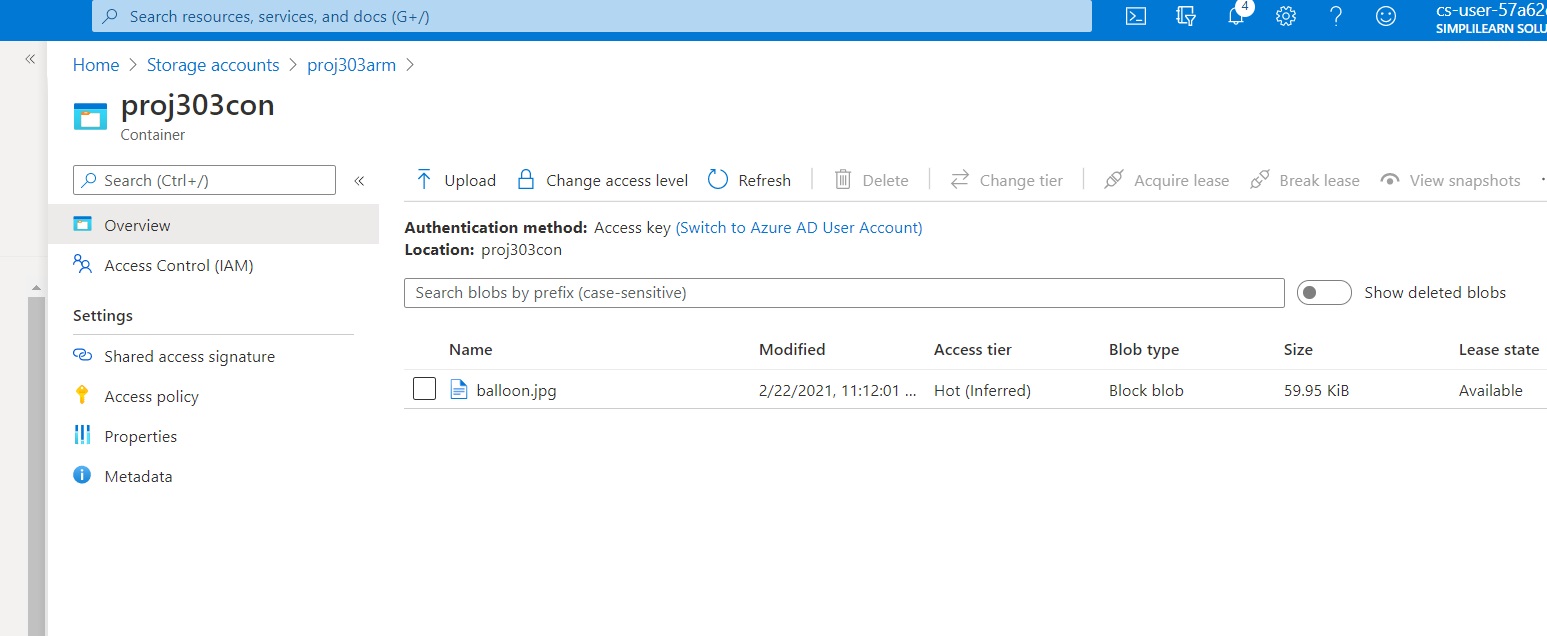
* LOCATION : CENTRALUS
* ACCOUNT\_TYPE : Standard\_LRS
* Kind : Storagev2
* accessTier : Hot
* supportshttpsTrafficonly : true

At azure portal ,storage account created with name proj303arm

At storage account create a container name : proj303con

At this container upload a jpg file named balloon.jpg





STEP 4 : CREATE CDN PROFILE

FOLLOWING COMMAND EXECUTE AT POWERSHELL

new-AzResourceGroupDeployment -ResourceGroupName newrg1 -TemplateFile C:\ARMTemplate\CDNTemplate\TEMPLATE.json

**TEMPLATE.JSON**

{

    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",

    "contentVersion": "1.0.0.0",

    "parameters": {

        "profiles\_testcdn\_name": {

            "defaultValue": "proj303cdn",

            "type": "String"

        }

    },

    "variables": {},

    "resources": [

        {

            "type": "Microsoft.Cdn/profiles",

            "apiVersion": "2020-04-15",

            "name": "[parameters('profiles\_testcdn\_name')]",

            "location": "Global",

            "sku": {

                "name": "Standard\_Akamai"

            },

            "kind": "cdn",

            "properties": {}

        },

        {

            "type": "Microsoft.Cdn/profiles/endpoints",

            "apiVersion": "2020-04-15",

            "name": "[concat(parameters('profiles\_testcdn\_name'), '/imagesendpoint')]",

            "location": "Global",

            "dependsOn": [

                "[resourceId('Microsoft.Cdn/profiles', parameters('profiles\_testcdn\_name'))]"

            ],

            "properties": {

                "originHostHeader": "proj303arm.blob.core.windows.net",

                "contentTypesToCompress": [

                    "text/plain",

                    "text/html",

                    "text/css",

                    "text/javascript",

                    "application/x-javascript",

                    "application/javascript",

                    "application/json",

                    "application/xml"

                ],

                "isCompressionEnabled": true,

                "isHttpAllowed": true,

                "isHttpsAllowed": true,

                "queryStringCachingBehavior": "IgnoreQueryString",

                "origins": [

                    {

                        "name": "proj303arm-blob-core-windows-net",

                        "properties": {

                            "hostName": "proj303arm.blob.core.windows.net",

                            "httpPort": 80,

                            "httpsPort": 443,

                            "enabled": true

                        }

                    }

                ],

                "originGroups": [],

                "geoFilters": [],

                "urlSigningKeys": []

            }

        },

        {

            "type": "Microsoft.Cdn/profiles/endpoints/origins",

            "apiVersion": "2020-04-15",

            "name": "[concat(parameters('profiles\_testcdn\_name'), '/imagesendpoint/', parameters('profiles\_testcdn\_name'), 'proj303-blob-core-windows-net')]",

            "dependsOn": [

                "[resourceId('Microsoft.Cdn/profiles/endpoints', parameters('profiles\_testcdn\_name'), 'imagesendpoint')]",

                "[resourceId('Microsoft.Cdn/profiles', parameters('profiles\_testcdn\_name'))]"

            ],

            "properties": {

                "hostName": "proj303arm.blob.core.windows.net",

                "httpPort": 80,

                "httpsPort": 443,

                "enabled": true

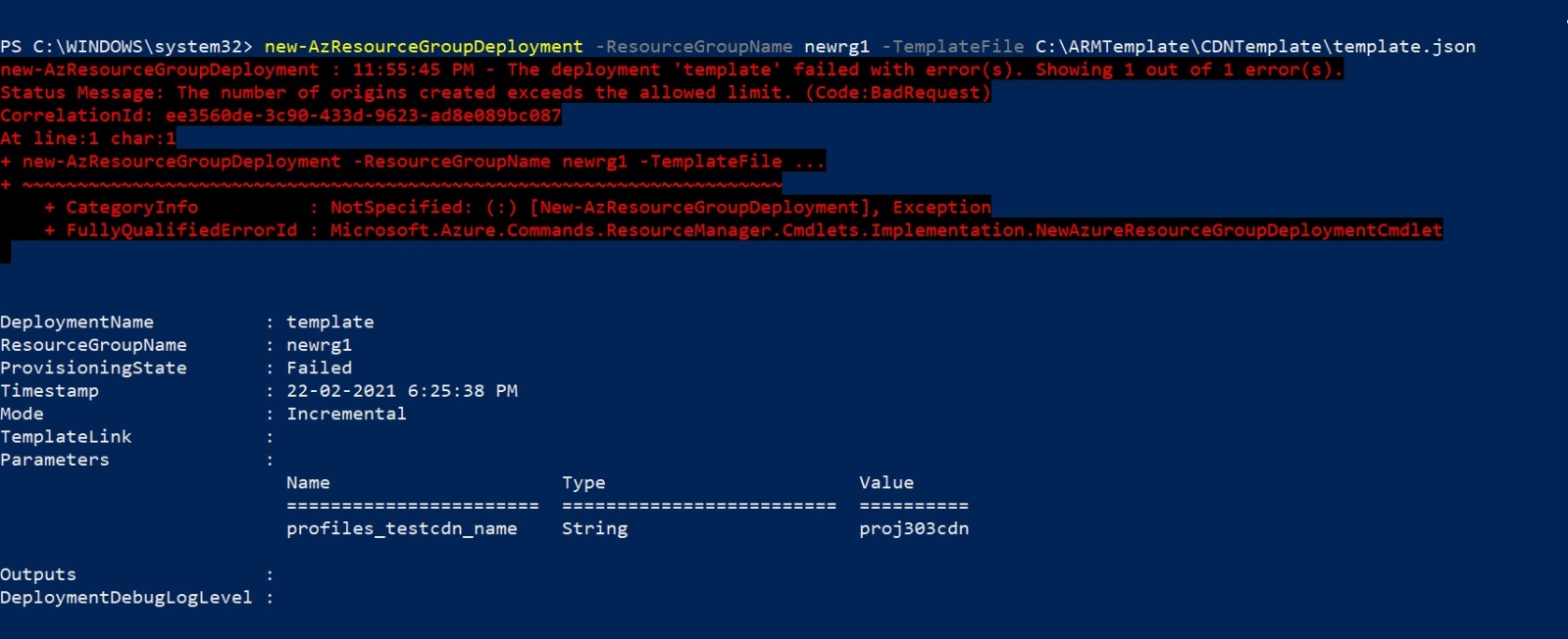
            }

        }

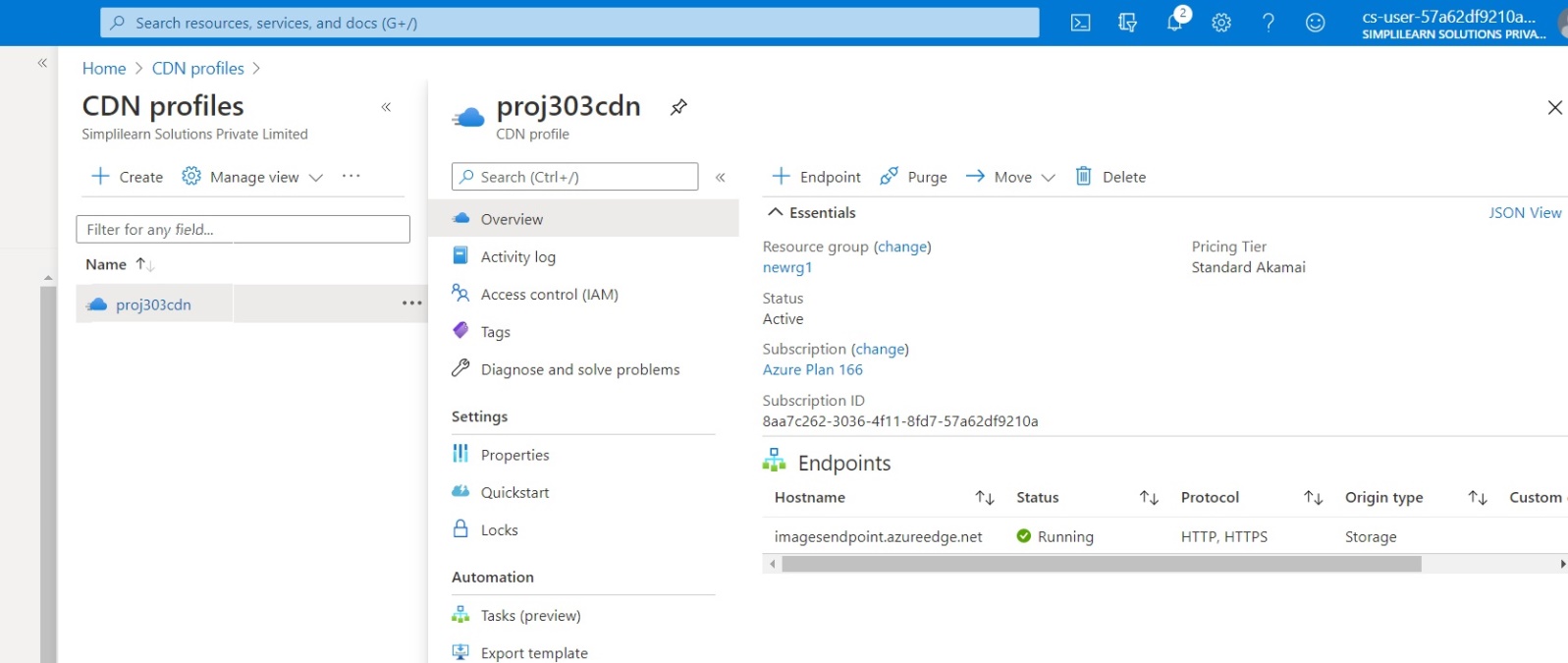
    ]

}

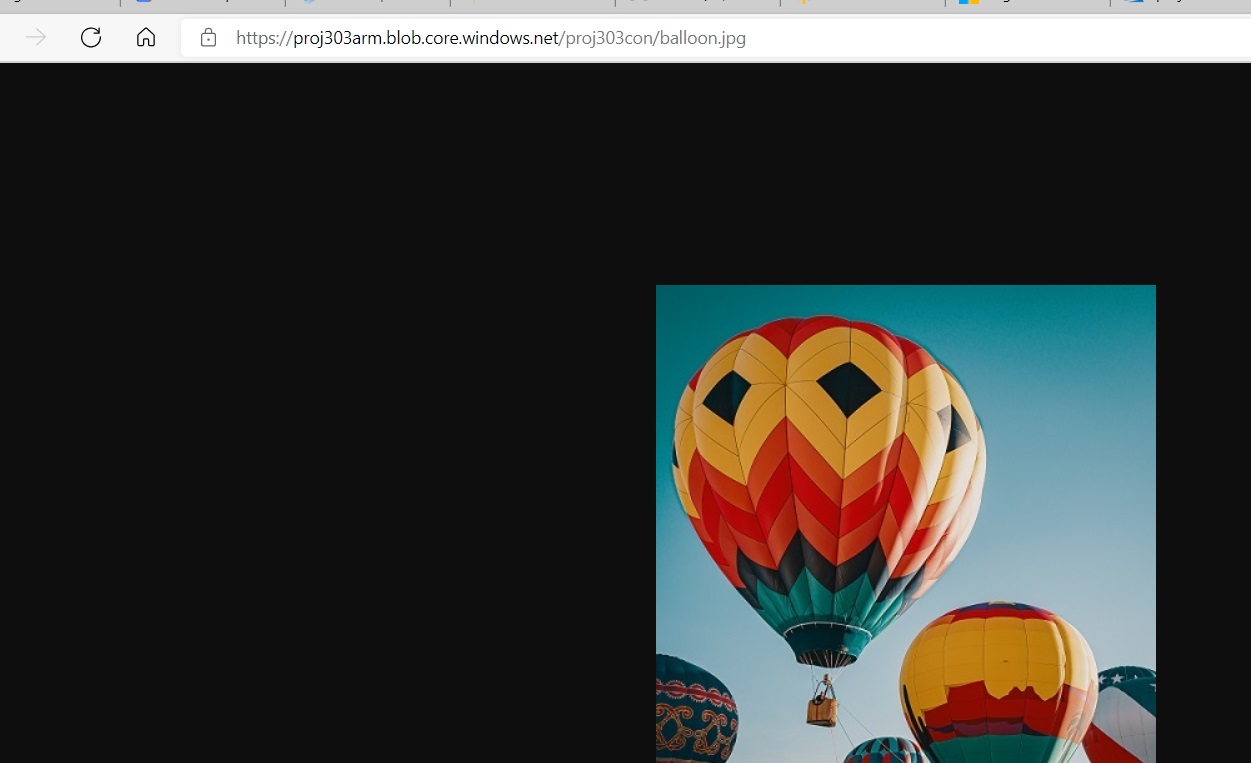
**IN EXECUTE AT POWERSHELL**

****

**IT CREATE CDN ACCOUNT AT AZURE PORTAL**

****

**FROM CDNPROFILE ACCOUNT COPY THE ENDPOINT AND RUN AT BROWSER**

****