Create Resources - Docs

Table of Contents

Introduction

This documentation provides step-by-step instructions for deploying container-based applications to Azure Kubernetes Service (AKS) using the AzureResourceSetup PowerShell script. The script simplifies the process of setting up an AKS cluster and associated Azure Container Registry (ACR).

Prerequisites

- You have an active Azure account.
- You have a Windows machine with PowerShell 5.1 or higher installed.
- Scripts must be allowed to run on the local machine.
    - You can verify this by opening PowerShell and run the following command:

```
# If the output is "Restricted" you are not allowed to run scripts
Get-ExecutionPolicy
```

    - If scripts are not allowed to be run on the system, open PowerShell with administrator privileges and run the command:

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned
```

- Az module for PowerShell is installed. If not, run the following command in PowerShell with admin privileges to install it:

```
Install-Module -Name Az -Scope CurrentUser)
```

- Your Azure account has the necessary roles and permissions (refer to section 6 for details).

Overview

The script referenced in this documentation is designed to create all the necessary resources in Azure to deploy container-based applications.

The AzureResourceSetup script assists users to set up and configure an Azure Kubernetes Service (AKS) cluster with an associated Azure Container Registry (ACR) using PowerShell. The script guides the user through a WPF application to gather the required input parameters and then proceeds to create or use existing resources based on the user's preferences. The script also configures the necessary permissions for the AKS cluster to access the ACR and attaches the ACR to the cluster.

Input Parameters

The following parameters are required from the user:

- Resource Group: A new or existing Azure Resource Group to be used for the ACR and AKS cluster. Resource Groups are logical containers for resources that are deployed within an Azure subscription. They provide a way to monitor, control access, provision, and manage billing for collections of resources. This parameter is required to organize and manage the created resources (ACR and AKS cluster) within a specific group.
- Location: Azure region for the new resources (if not using existing resources).
- Container Registry Name: Name of the new or existing ACR. Azure Container Registry (ACR) is a managed, private Docker registry service for building, storing, and managing container images and artifacts. This parameter is needed to identify the specific ACR that will be associated with the AKS cluster. This enables the AKS cluster to pull container images from the ACR to run workloads.
- AKS Cluster Name: Name of the new AKS cluster. Azure Kubernetes Service (AKS) is a managed Kubernetes service for running containerized applications. This parameter is needed to give a unique name to the new AKS cluster being created, making it easier to identify and manage the cluster within the Azure portal.
- Node Count: Number of nodes to be created in the AKS cluster. The node count parameter determines the number of worker nodes in the AKS cluster, which affects the capacity and performance of the cluster. More nodes can provide increased capacity for running workloads, but they also increase the cost of maintaining the cluster.

Additionally, the user can choose to use existing resources:

- Use Existing Resource Group: Check this option to use an existing Resource Group.
- Existing Resource Group: Select the desired Resource Group from the dropdown list.
- Use Existing Container Registry: Check this option to use an existing Container Registry.
- Existing Container Registry: Select the desired Container Registry from the dropdown list.

Execution Flow

1. The script logs in to the Azure account with a specific tenant.
2. Loads the required assembly for Windows Forms.
3. Displays a form to collect input parameters from the user.
4. Retrieves a list of existing Resource Groups and ACRs and fills the dropdown lists.
5. Creates a new Resource Group if the user chooses not to use an existing one.
6. Creates a new ACR if the user chooses not to use an existing one.
7. Logs in to the created or chosen ACR.
8. Creates a new AKS cluster with the specified node count.
9. Grants the necessary permissions for the AKS cluster to access the ACR (AcrPull role assignment).
10. Attaches the ACR to the AKS cluster.
11. Connects kubectl to the AKS cluster using the imported credentials.
12. Displays a message box to inform the user that the script has finished running.

Required Azure Roles

To execute the operations in the script, you need specific rights in your Azure account. Below is a list of the required permissions and their corresponding Azure built-in roles:

1. Azure Resource Manager (ARM) operations:
    - Create, update, and delete Resource Groups
    - List Resource Groups

Required Role: *Contributor* or *Owner*

2. Azure Container Registry (ACR) operations:
    - Create, update, and delete ACR instances
    - List ACR instances and their properties
    - Retrieve ACR credentials

Required Role: *Contributor* or *Owner*

3. Azure Kubernetes Service (AKS) operations:
    - Create, update, and delete AKS clusters
    - List AKS clusters and their properties
    - Import AKS credentials for kubectl

Required Role: *Contributor* or *Owner*

4. Role assignments:
    - Assign roles to the AKS managed identity

Required Role: *User Access Administrator*

To execute all the operations in the script, your account should have at least the *Contributor* or *Owner* role, along with the *User Access Administrator* role, either at the subscription or resource group level.

Running the Script

To run the create_resource_form script, follow these steps:

1. Clone the Git repository containing the script from the provided URL to the desired location on your computer. You can do this by copying the URL, opening a terminal like PowerShell, and running the following command:

```
git clone <url>
```

2. Navigate to the folder where the AzureResourceSetup.ps1 is located, right click the script, and choose “Run with PowerShell”. A graphical interface will appear.
3. Fill in the required fields and select the appropriate options based on your preferences.
4. Click the 'OK' button to start the execution of the script.
5. The script will run, creating or using existing resources and configuring the necessary permissions.
6. Once the script finishes, a message box will be displayed, indicating that the script has completed its execution.

Troubleshooting and Tips

- If you encounter any errors while running the script, verify that your Azure account has the required roles mentioned in section 6. If you are unsure about your permissions, consult your Azure administrator.
- Make sure you have installed the Az module for PowerShell, as mentioned in the prerequisites.
- If you have multiple Azure accounts or subscriptions, make sure you are logged into the correct account and have the desired subscription set as default. You can use "Get-AzContext" to check the current context and "Set-AzContext" to set the desired subscription.
- The script may take some time to execute, especially when creating new resources or updating permissions. Please be patient and wait for the message box to appear, indicating the completion of the script.

Importing kubernetes config

When you connect to an Azure Kubernetes Service (AKS) cluster using the `Import-AzAksCredential` cmdlet in PowerShell, it retrieves the Kubernetes configuration (kubeconfig) for the specified AKS cluster. The kubeconfig file contains the necessary information, such as cluster API server address, cluster certificate, and user credentials, to connect to and interact with the Kubernetes API server.

When the script prompts you to import the Kubernetes config, it means that the script is asking for permission to update your local kubeconfig file with the configuration details of the AKS cluster you've created. By doing this, you can use `kubectl` to manage your AKS cluster directly from your local machine.

If you choose to import the configuration, the script will update your local kubeconfig file with the cluster information. After the import, you can use `kubectl` commands to manage your AKS cluster from your local machine. If you choose not to import the configuration, you'll need to manually configure `kubectl` to connect to your AKS cluster later.