無料電子ブック

学習

azure

Free unaffiliated eBook created from
Stack Overflow contributors.

#azure

1: のまり

Azureは、Microsoftがクラウドコンピューティングサービスをしているブランドです。 Microsoft Azureプラットフォームでされるなサービスのはのとおりです。

- サービスとしてのインフラストラクチャIaaSLinuxおよびWindows Azureマシン
- サービスとしてのプラットフォームPaaSApp Serviceは、アプリWebおよびモバイルのためのなプラットフォームをし、
- クラウドストレージSQLとnoSQLストレージサービス
- ソフトウェアとしてのサービスSaaSスケジューラ、バックアップ、、、セキュリティと

Azureのなをでるためのインフォグラフィックがあります https : //azure.microsoft.com/en-us/resources/infographics/azure/ ここで Azureのすべてのをカテゴリにブラウズしてフィルタリングすることができます。

Examples

Azure NシリーズGPUUBUNTU 16.04 LTSでCUDA、cudnn、Tensorflowをインストール

5ごした、はこのなをつけました

- システムにCUDAGPUがあることをするには、のコマンドをします。

```
lspci | grep -i NVIDIA
```

ののようながされますNVIDIA Tesla K80 / M60カードを。

```
af8a:00:00.0 3D controller: NVIDIA Corporation GK210GL [Tesla K80] (rev a1)
```

- ヌーボードライバをにする

```
sudo -i
rmmod nouveau
```

`sudo reboot`し、ドライバがしくインストールされていることをします。

```
lsmod | grep -i nvidia
```

- に、Nvidiaから**CUDA**パッケージをダウンロードしてください。

```
wget https://developer.nvidia.com/compute/cuda/8.0/prod/local_installers/cuda-repo-ubuntu1604-8-0-local_8.0.44-1_amd64-deb
```

-... CUDAツールキットをにしてインストールすることをらせる

```
sudo dpkg -i cuda-repo-ubuntu1604-8-0-local_8.0.44-1_amd64-deb
sudo apt-get update
sudo apt-get install -y cuda
```

- GPUのステータスをするには、のコマンドをします。

```
nvidia-smi
```

に、 **cuDNN**をダウンロードします...

```
wget http://developer.download.nvidia.com/compute/redist/cudnn/v5.1/cudnn-8.0-linux-x64-
v5.1.tgz
```

-... unzip、lib64とincludeフォルダをコピーします

```
tar -zxf cudnn-8.0-linux-x64-v5.1.tgz
sudo cp cuda/lib64/* /usr/local/cuda/lib64/
sudo cp cuda/include/* /usr/local/cuda/include/
sudo rm -R cuda
```

- いくつかのクリーンアップをい、ダウンロードしたアーカイブをする

```
rm cuda-repo-ubuntu1604-8-0-local_8.0.44-1_amd64-deb
rm cudnn-8.0-linux-x64-v5.1.tgz
```

CPU / GPUで**Tensorflow**をインストールするには、ここをクリックしてください

https://www.tensorflow.org/install/install_linux#installing_with_anaconda

1. https://www.lutzroeder.com/blog/2016-12-27-tensorflow-azure 2.
https://www.tensorflow.org/install/install_linux#installing_with_anaconda

オンラインでのまりをむ https://riptutorial.com/ja/azure/topic/1060/のまり

2: Azure DocumentDB

Examples

アカウントにする.NET

あなたのDocumentDBデータベースにするには、するがあります`DocumentClient`あなたのエンドポイント**URI**とサービスキー あなたは、のポータルからすることができますで。

まず、のをするがあります。

```
using System;
using Microsoft.Azure.Documents.Client;
```

に、をしてクライアントをできます。

```
var endpointUri = "<your endpoint URI>";
var primaryKey = "<your key>";
var client = new DocumentClient(new Uri(endpointUri), primaryKey);
```

データベースをする.NET

DocumentDB データベースは、 `DocumentClient`クラスの`CreateDatabaseAsync`メソッドをしてできます。データベースは、コレクションにされたJSONドキュメントストレージのコンテナです。

```
using System.Net;
using System.Threading.Tasks;
using Microsoft.Azure.Documents;
using Microsoft.Azure.Documents.Client;
```

データベースをするには

```
async Task CreateDatabase(DocumentClient client)
{
    var databaseName = "<your database name>";
    await client.CreateDatabaseAsync(new Database { Id = databaseName });
}
```

データベースがすでにするかどうかをし、にしてデータベースをすることもできます。

```
async Task CreateDatabaseIfNotExists(DocumentClient client)
{
    var databaseName = "<your database name>";
    try
    {
        await client.ReadDatabaseAsync(UriFactory.CreateDatabaseUri(databaseName));
    }
    catch (DocumentClientException e)
```

```
    {
        // If the database does not exist, create a new database
        if (e.StatusCode == HttpStatusCode.NotFound)
        {
            await client.CreateDatabaseAsync(new Database { Id = databaseName });

        }
        else
        {
            // Rethrow
            throw;
        }
    }
}
```

コレクションをする.NET

コレクションは、 `DocumentClient`クラスの`CreateDocumentCollectionAsync`メソッドをしてできます。コレクションは、JSONドキュメントとするJavaScriptアプリケーションロジックのコンテナです。

```
async Task CreateCollection(DocumentClient client)
{
    var databaseName = "<your database name>";
    var collectionName = "<your collection name>";

    DocumentCollection collectionInfo = new DocumentCollection();
    collectionInfo.Id = collectionName;

    // Configure collections for maximum query flexibility including string range queries.
    collectionInfo.IndexingPolicy = new IndexingPolicy(new RangeIndex(DataType.String) {
Precision = -1 });

    // Here we create a collection with 400 RU/s.
    await client.CreateDocumentCollectionAsync(UriFactory.CreateDatabaseUri(databaseName),
        collectionInfo, new RequestOptions { OfferThroughput = 400 });
}
```

または、コレクションがするかどうかをし、にじてできます。

```
async Task CreateDocumentCollectionIfNotExists(DocumentClient client)
{
    var databaseName = "<your database name>";
    var collectionName = "<your collection name>";
    try
    {
        await
client.ReadDocumentCollectionAsync(UriFactory.CreateDocumentCollectionUri(databaseName,
collectionName));
    }
    catch (DocumentClientException e)
    {
        // If the document collection does not exist, create a new collection
        if (e.StatusCode == HttpStatusCode.NotFound)
        {
            DocumentCollection collectionInfo = new DocumentCollection();
```

```
            collectionInfo.Id = collectionName;

            // Configure collections for maximum query flexibility including string range
queries.
            collectionInfo.IndexingPolicy = new IndexingPolicy(new RangeIndex(DataType.String)
{ Precision = -1 });

            // Here we create a collection with 400 RU/s.
            await
client.CreateDocumentCollectionAsync(UriFactory.CreateDatabaseUri(databaseName),
                collectionInfo, new RequestOptions { OfferThroughput = 400 });
        }
        else
        {
            // Rethrow
            throw;
        }
    }
}
```

JSONドキュメントの.NET

ドキュメントは、 `DocumentClient`クラスの`CreateDocumentAsync`メソッドをしてできます。ドキュメントはユーザーののJSONコンテンツです。

```
async Task CreateFamilyDocumentIfNotExists(DocumentClient client, string databaseName, string
collectionName, Family family)
{
    try
    {
        await client.ReadDocumentAsync(UriFactory.CreateDocumentUri(databaseName,
collectionName, family.Id));
    }
    catch (DocumentClientException e)
    {
        if (e.StatusCode == HttpStatusCode.NotFound)
        {
            await
client.CreateDocumentAsync(UriFactory.CreateDocumentCollectionUri(databaseName,
collectionName), family);
        }
        else
        {
            // Rethrow
            throw;
        }
    }
}
```

されたをすのクラスをつ

```
public class Family
{
    [JsonProperty(PropertyName = "id")]
    public string Id { get; set; }
    public string LastName { get; set; }
    public Parent[] Parents { get; set; }
```

```
    public Child[] Children { get; set; }
    public Address Address { get; set; }
    public bool IsRegistered { get; set; }
    public override string ToString()
    {
        return JsonConvert.SerializeObject(this);
    }
}

public class Parent
{
    public string FamilyName { get; set; }
    public string FirstName { get; set; }
}

public class Child
{
    public string FamilyName { get; set; }
    public string FirstName { get; set; }
    public string Gender { get; set; }
    public int Grade { get; set; }
    public Pet[] Pets { get; set; }
}

public class Pet
{
    public string GivenName { get; set; }
}

public class Address
{
    public string State { get; set; }
    public string County { get; set; }
    public string City { get; set; }
}
```

ドキュメントのクエリ.NET

DocumentDBは、コレクションにされているJSONドキュメントにするなクエリをサポートしています。

LINQクエリをする

```
IQueryable<Family> familyQuery = this.client.CreateDocumentQuery<Family>(
    UriFactory.CreateDocumentCollectionUri(databaseName, collectionName), queryOptions)
    .Where(f => f.LastName == "Andersen");
```

SQLクエリで

```
IQueryable<Family> familyQueryInSql = this.client.CreateDocumentQuery<Family>(
    UriFactory.CreateDocumentCollectionUri(databaseName, collectionName),
    "SELECT * FROM Family WHERE Family.lastName = 'Andersen'",
```

```
    queryOptions);
```

LINQ クエリのページ

FeedOptionsをするためにされRequestContinuationのクエリでしたを

```
public async Task<IEnumerable<Family>> QueryWithPagination(int Size_of_Page)
{
    var queryOptions = new FeedOptions() { MaxItemCount = Size_of_Page };
    string continuationToken = string.Empty;
    do
    {
        if (!string.IsNullOrEmpty(continuationToken))
        {
            queryOptions.RequestContinuation = continuationToken;
        }

        IDocumentQuery<Family> familyQuery = this.client.CreateDocumentQuery<Family>(
            UriFactory.CreateDocumentCollectionUri(databaseName, collectionName),
queryOptions)
            .Where(f => f.LastName == "Andersen").AsDocumentQuery();

        var queryResult = await familyQuery.ExecuteNextAsync<Family>();
        continuationToken = queryResult.ResponseContinuation;
        yield return queryResult;

    } while (!string.IsNullOrEmpty(continuationToken));
}
```

びしてContinuation Tokenをクライアントにすことができます。そのため、クライアントがのペ―ジをとするときに、ペ―ジきのがされます。ヘルパ―クラスとをう

```
public class PagedResults<T>
{
    public PagedResults()
    {
        Results = new List<T>();
    }
    public string ContinuationToken { get; set; }
    public List<T> Results { get; set; }
}

public async Task<PagedResults<Family>> QueryWithPagination(int Size_of_Page, string
continuationToken = "")
{
    var queryOptions = new FeedOptions() { MaxItemCount = Size_of_Page };
    if (!string.IsNullOrEmpty(continuationToken))
    {
        queryOptions.RequestContinuation = continuationToken;
    }

    return await familyQuery = this.client.CreateDocumentQuery<Family>(
        UriFactory.CreateDocumentCollectionUri(databaseName, collectionName), queryOptions)
        .Where(f => f.LastName == "Andersen").ToPagedResults();
}
```

```
public static class DocumentDBExtensions
{
    public static async Task<PagedResults<T>> ToPagedResults<T>(this IQueryable<T> source)
        {
            var documentQuery = source.AsDocumentQuery();
            var results = new PagedResults<T>();
            try
            {
                var queryResult = await documentQuery.ExecuteNextAsync<T>();
                if (!queryResult.Any())
                {
                    return results;
                }
                results.ContinuationToken = queryResult.ResponseContinuation;
                results.Results.AddRange(queryResult);
            }
            catch
            {
                //documentQuery.ExecuteNextAsync throws an exception on empty queries
                return results;
            }

            return results;
        }
}
```

ドキュメントをする.NET

DocumentDBは、 `DocumentClient`クラスの`ReplaceDocumentAsync`メソッドをしてJSONドキュメントをきえることをサポートしています。

```
await client.ReplaceDocumentAsync(UriFactory.CreateDocumentUri(databaseName, collectionName,
familyName), updatedFamily);
```

ドキュメントをする.NET

DocumentDBは、 `DocumentClient`クラスの`DeleteDocumentAsync`メソッドをしてJSONドキュメントをすることをサポートしています。

```
await client.DeleteDocumentAsync(UriFactory.CreateDocumentUri(databaseName, collectionName,
documentName));
```

データベースの.NET

データベースをすると、データベースとすべてのリソースコレクション、ドキュメントなどがされます。

```
await this.client.DeleteDatabaseAsync(UriFactory.CreateDatabaseUri(databaseName));
```

オンラインでAzure DocumentDBをむ https://riptutorial.com/ja/azure/topic/5176/azure-

documentdb

3: Azure Mediaサービスアカウント

メディアサービスアカウントはAzureのクラウドベースのメディアサービスにアクセスできるAzureベースのアカウントです。したメディアファイルのメタデータをし、わりにのメディアコンテンツをします。メディアサービスアカウントをするには、するストレージアカウントがです。メディアサービスアカウントをするには、のストレージアカウントをするか、しいアカウントをすることができます。メディアサービスアカウントとストレージアカウントは々にわれるため、メディアサービスアカウントをしてもコンテンツはストレージアカウントでになります。ストレージアカウントのは、メディアサービスアカウントのとじでなければなりません。

Examples

メディアサービスアカウントでのアセットの

```
public static string CreateBLOBContainer(string containerName)
        {
            try
            {
                string result = string.Empty;
                CloudMediaContext mediaContext;
                mediaContext = new CloudMediaContext(mediaServicesAccountName,
mediaServicesAccountKey);
                IAsset asset = mediaContext.Assets.Create(containerName,
AssetCreationOptions.None);
                return asset.Uri.ToString();
            }
            catch (Exception ex)
            {
                return ex.Message;
            }
        }
```

からアイテムをする

```
private static void GetAllTheAssetsAndFiles(MediaServicesCredentials _medServCredentials)
       {
           try
           {
               string result = string.Empty;
               CloudMediaContext mediaContext;
               mediaContext = new CloudMediaContext(_medServCredentials);
               StringBuilder myBuilder = new StringBuilder();
               foreach (var item in mediaContext.Assets)
               {
                   myBuilder.AppendLine(Environment.NewLine);
                   myBuilder.AppendLine("--My Assets--");
                   myBuilder.AppendLine("Name: " + item.Name);
                   myBuilder.AppendLine("++++++++++++++++++++");

                   foreach (var subItem in item.AssetFiles)
                   {
```

```
                    myBuilder.AppendLine("File Name: "+subItem.Name);
                    myBuilder.AppendLine("Size: " + subItem.ContentFileSize);
                    myBuilder.AppendLine("++++++++++++++++++++++");
                }
            }
            Console.WriteLine(myBuilder);
        }
        catch (Exception)
        {
            throw;
        }
    }
```

オンラインでAzure Mediaサービスアカウントをむ
https://riptutorial.com/ja/azure/topic/4997/azure-mediaサービスアカウント

4: Azure Powershell

Examples

クラシックモードとARMモード

PowerShellをしてAzureをするは、2つのなるがあります。これらのをするくのMicrosoftドキュメントがされます。

「クラシックモードサービス」

これは、AzureをしAzureをするいです。よりくのサービスがしいARMモードにしているにもかかわらず、クラシックモードをしてのみできるAzureにはまだいくつかのサービスがあります。

クラシックモードでしているマシンにインストールされているモジュールをするには、のをいます。

```
Get-Module -ListAvailable Azure.*
```

「リソースマネージャARM」

これはAzureをするしいですされているREST APIにづいています。 Powershellクラシックモードからでできるサービスのほとんどは、いくつかのをいてしいモードをしてできるようになりました。これは、ARMモードではサポートされていないのサービスがないり、Azureリソースをするためのましいです。

ARMモードでするコマンドがまれているマシンにインストールされているモジュールをするには、のをします。

```
Get-Module -ListAvailable AzureRM*
```

Azureにログイン

クラシックサービスモード

```
Add-AzureAccount
```

Azure Active Directoryをしてされ、PowerShellは12にがれるアクセストークンをします。したがって、12にをりすがあります。

または、のコマンドレットをするもあります。

```
Get-AzurePublishSettingsFile
```

これにより、パブリッシュファイルをダウンロードできるブラウザウィンドウがきます。このファイルには、PowerShellのをするがまれています。に、をしてパブリッシュファイルをインポートできます。

```
Import-AzurePublishSettingsFile
```

パブリッシュファイルには、サブスクリプションのにをつがまれています。はにつか、してください。

リソースマネージャー

Resource Managerでは、Azure Active Directoryと12のアクセストークンのみをできます。、できる2つのコマンドがあります。

```
Login-AzureRmAccount
Add-AzureRmAccount
```

の

Azureアカウントのにのサブスクリプションがある。したいものをすることがですデフォルトでこれをしてください。ったサブスクリプションでリソースにこっているをけることができます。

クラシックモード

```
Set-AzureSubscription
Select-AzureSubscription
```

リソースマネージャー

```
Select-AzureRmSubscription
```

のコマンドは、りえたいサブスクリプションをするためのサブスクリプションIDなどをするようにします。アクセスのあるサブスクリプションのこのをするには、のコマンドをします。

```
Get-AzureSubscription
```

のAzure PowerShellバージョンをする

インストールしたAzure PowerShellのバージョンをするには、のコマンドをします。

```
Get-Module -ListAvailable -Name Azure -Refresh
```

このコマンドは、のPowerShellセッションでAzure PowerShellモジュールをロードしていないでも、インストールされているバージョンをします。

Azureアセットをする

Azureコマンドレットをすると、CコードまたはAzureポータルをするPowerShellをして、Azureアセットでじアクションのいくつかをできます。

たとえば、のをすると、Azure BLOBのをローカルディレクトリにダウンロードできます。

```
New-Item -Path .\myblob -ItemType Directory
$context = New-AzureStorageContext -StorageAccountName MyAccountName -StorageAccountKey {key
from the Azure portal}
$blob = Get-AzureStorageBlob -Container MyContainerName -Context $context
$blob | Get-AzureStorageBlobContent -Destination .\myblob\
```

トラフィックマネージャの

Azure PowerShellをすると、 Azure Portalでできないのをのようにすることができます

- にすべてのTraffic Managerのエンドポイントをする
- ドメインのわりにAzure `ResourceId`してのサービスにするため、Azureエンドポイントに Locationをでするはありません

まず、 ログイン してRMサブスクリプションをするがありま す 。

TrafficManagerプロファイルをする

PowerShellによるトラフィックマネージャのは、の3つのステップでされます。

1. TMプロファイルをする
   `$profile = Get-AzureRmTrafficManagerProfile -ResourceGroupName my-resource-group -Name my-traffic-manager`
   または、こののようにしてください 。
2. TMプロファイルのと
   `$profile`フィールドと`$profile.Endpoints`をチェックして、エンドポイントのをしてください 。
3. `Set-AzureRmTrafficManagerProfile -TrafficManagerProfile $profile`をし`Set-AzureRmTrafficManagerProfile -TrafficManagerProfile $profile`。

エンドポイントをする

のすべてのエンドポイントは`$profile.Endpoints`リストにされているので、インデックスですることができます

```
$profile.Endpoints[0].Weight = 100
```

またはで

```
$profile.Endpoints | ?{ $_.Name -eq 'my-endpoint' } | %{ $_.Weight = 100 }
```

すべてのエンドポイントをクリアするには

```
$profile.Endpoints.Clear()
```

のエンドポイントをするには

```
Remove-AzureRmTrafficManagerEndpointConfig -TrafficManagerProfile $profile -EndpointName 'my-endpoint'
```

しいエンドポイントのをするには

```
Add-AzureRmTrafficManagerEndpointConfig -TrafficManagerProfile $profile -EndpointName "my-endpoint" -Type AzureEndpoints -TargetResourceId "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/my-resource-group/providers/Microsoft.ClassicCompute/domainNames/my-azure-service" -EndpointStatus Enabled -Weight 100
```

ごのように、のケースでは、ドメインではなく、ResourceIdをしてのサービスにりんでいます。

にめておく

TMとそのエンドポイントのは、 `Set-AzureRmTrafficManagerProfile -TrafficManagerProfile $profile` びすまでされません。これにより、のでTMをにすることができます。

トラフィックマネージャはDNSのであり、クライアントにえられるIPアドレスにはあるのがありますTTLともばれますが、 `$profile.Ttl` フィールドにで`$profile.Ttl`ます。したがって、TMをした、TTLがれになるまで、キャッシュされたいエンドポイントをしけるクライアントもあります。

オンラインでAzure Powershellをむ https://riptutorial.com/ja/azure/topic/3961/azure-powershell

5: Azure Resource Managerテンプレート

- ARMテンプレートのはにされています。https : //azure.microsoft.com/en-us/documentation/articles/resource-group-authoring-templates/

Examples

リソースをする

Extension Azureのリソースは、のリソースをするリソースです。

このテンプレートは、Azure Key VaultとDiagnosticSettingsをします。

すべきこと

- リソースは、リソースの`resources`のにされます
- `dependsOn`リソースをする`dependsOn`をつがありますARMがリソースとにをしないようにするため

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "keyVaultName": {
      "type": "string",
      "metadata": {
        "description": "Name of the Vault"
      }
    },
    "tenantId": {
      "type": "string",
      "metadata": {
        "description": "Tenant ID of the directory associated with this key vault"
      }
    },
    "location": {
      "type": "string",
      "metadata": {
        "description": "Key Vault location"
      }
    },
    "storageAccountResourceGroup": {
      "type": "string",
      "metadata": {
        "description": "Resource Group of the storage account where key vault activities will
be logged"
      }
    },
    "storageAccountName": {
      "type": "string",
      "metadata": {
        "description": "Name of the storage account where key vault activities will be logged.
```

```
Must be in same region as the key vault."
      }
    }
    },
  "resources": [
    {
      "type": "Microsoft.KeyVault/vaults",
      "name": "[parameters('keyVaultName')]",
      "apiVersion": "2015-06-01",
      "location": "[parameters('location')]",
      "properties": {
        "enabledForDeployment": "false",
        "enabledForDiskEncryption": "false",
        "enabledForTemplateDeployment": "false",
        "tenantId": "[variables('tenantId')]",
        "sku": {
          "name": "Standard",
          "family": "A"
        }
      },
      "resources": [
          {
      "type": "Microsoft.KeyVault/vaults/providers/diagnosticSettings",
      "name": "[concat(parameters('keyVaultName'), '/Microsoft.Insights/service')]",
      "apiVersion": "2015-07-01",
      "dependsOn": [
        "[concat('Microsoft.keyvault/vaults/', parameters('keyVaultName'))]"
      ],
      "properties": {
        "storageAccountId": "[resourceId(parameters('storageAccountResourceGroup'),
'Microsoft.Storage/storageAccounts', parameters('storageAccountName'))]",
        "logs": [{
            "category": "AuditEvent",
            "enabled": true,
            "retentionPolicy": {
                "enabled": true,
                "days": 90
            }
        }]
    }
    }]
    }
  ],
  "outputs": {
      "keyVaultUrl": {
          "type": "string",
          "value": "[reference(resourceId('Microsoft.KeyVault/vaults',
parameters('keyVaultName'))).vaultUri]"
      }
  }
}
```

オンラインでAzure Resource Managerテンプレートをむ
https://riptutorial.com/ja/azure/topic/3923/azure-resource-managerテンプレート

6: Azure-オートメーション

パラメーター

パラメータ	
resourceGroupName	ストレージアカウントがかれているAzureリソースグループ
	アカウントのにされたAzure Run Asサービスpricipal
StorageAccountName	Azure Storageアカウントの
ContainerName	blobコンテナ
DaysOld	BLOBをするまでの

Azure Active Directoryへのアクセスがあることをして、アカウントにAzureがRunAsアカウントをします。これはあなたにくのをします。

Examples

よりいBlobストレージのBlobをする

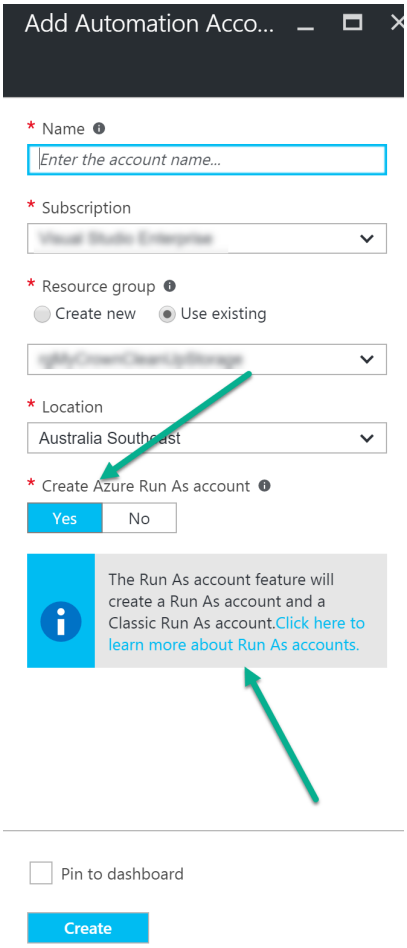
は、AzureストレージコンテナでよりいブロブをするAzure Powershellオートメーションランブックのです。

これは、コストとスペースをするためにいSQLバックアップをするのにです。

それはであるくのパラメータをとする。

デバッグにつコメントきのコードがいくつかっています。

Azureは、アカウントをするときににAzureができるサービスプリンシパルをします。 Azure Active Directoryにアクセスするがあります。をる

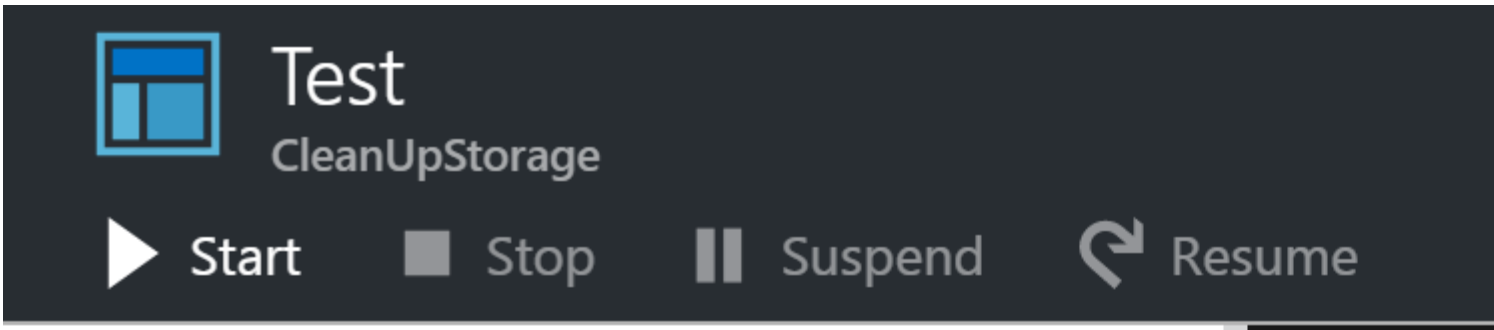



```
<#
.DESCRIPTION
    Removes all blobs older than a number of days back using the Run As Account (Service
Principal)

.NOTES
    AUTHOR: Russ
    LASTEDIT: Oct 03, 2016   #>

param(
    [parameter(Mandatory=$true)]
    [String]$resourceGroupName,

    [parameter(Mandatory=$true)]
    [String]$connectionName,

    # StorageAccount name for content deletion.
    [Parameter(Mandatory = $true)]
    [String]$StorageAccountName,

    # StorageContainer name for content deletion.
    [Parameter(Mandatory = $true)]
    [String]$ContainerName,

    [Parameter(Mandatory = $true)]
    [Int32]$DaysOld

)
$VerbosePreference = "Continue";
try
{
# Get the connection "AzureRunAsConnection "
```

```
$servicePrincipalConnection=Get-AutomationConnection -Name $connectionName

"Logging in to Azure..."
Add-AzureRmAccount `
    -ServicePrincipal `
    -TenantId $servicePrincipalConnection.TenantId `
    -ApplicationId $servicePrincipalConnection.ApplicationId `
    -CertificateThumbprint $servicePrincipalConnection.CertificateThumbprint
catch {
if (!$servicePrincipalConnection)
{
    $ErrorMessage = "Connection $connectionName not found."
    throw $ErrorMessage
} else{
    Write-Error -Message $_.Exception
    throw $_.Exception
}
$keys = Get-AzureRMStorageAccountKey -ResourceGroupName $resourceGroupName -AccountName
$StorageAccountName
# get the storage account key
Write-Host "The storage key is: "$StorageAccountKey;
# get the context
$StorageAccountContext = New-AzureStorageContext -storageAccountName $StorageAccountName -
StorageAccountKey $keys.Key1 #.Value;
$StorageAccountContext;
$existingContainer = Get-AzureStorageContainer -Context $StorageAccountContext -Name
$ContainerName;
#$existingContainer;
if (!$existingContainer)
{
 "Could not find storage container";
}
else
{
$containerName = $existingContainer.Name;
Write-Verbose ("Found {0} storage container" -f $containerName);
$blobs = Get-AzureStorageBlob -Container $containerName -Context $StorageAccountContext;
$blobsremoved = 0;

if ($blobs -ne $null)
{
    foreach ($blob in $blobs)
    {
        $lastModified = $blob.LastModified
        if ($lastModified -ne $null)
        {
            #Write-Verbose ("Now is: {0} and LastModified is:{1}" -f [DateTime]::Now,
[DateTime]$lastModified);
            #Write-Verbose ("lastModified: {0}" -f $lastModified);
            #Write-Verbose ("Now: {0}" -f [DateTime]::Now);
            $blobDays = ([DateTime]::Now - $lastModified.DateTime)  #[DateTime]

            Write-Verbose ("Blob {0} has been in storage for {1} days" -f $blob.Name,
$blobDays);

            Write-Verbose ("blobDays.Days: {0}" -f $blobDays.Hours);
            Write-Verbose ("DaysOld: {0}" -f $DaysOld);

            if ($blobDays.Days -ge $DaysOld)
            {
                Write-Verbose ("Removing Blob: {0}" -f $blob.Name);
```

```
                Remove-AzureStorageBlob -Blob $blob.Name -Container $containerName -Context
$StorageAccountContext;
                $blobsremoved += 1;
            }
            else {
                Write-Verbose ("Not removing blob as it is not old enough.");
            }
        }
    }
}

Write-Verbose ("{0} blobs removed from container {1}." -f $blobsremoved, $containerName);
}
```

テストペインをして、なパラメータをしてすることができます。



Parameters

* RESOURCEGROUPNAME

Enter a value

Mandatory, String

* CONNECTIONNAME

Enter a value

Mandatory, String

* STORAGEACCOUNTNAME

Enter a value

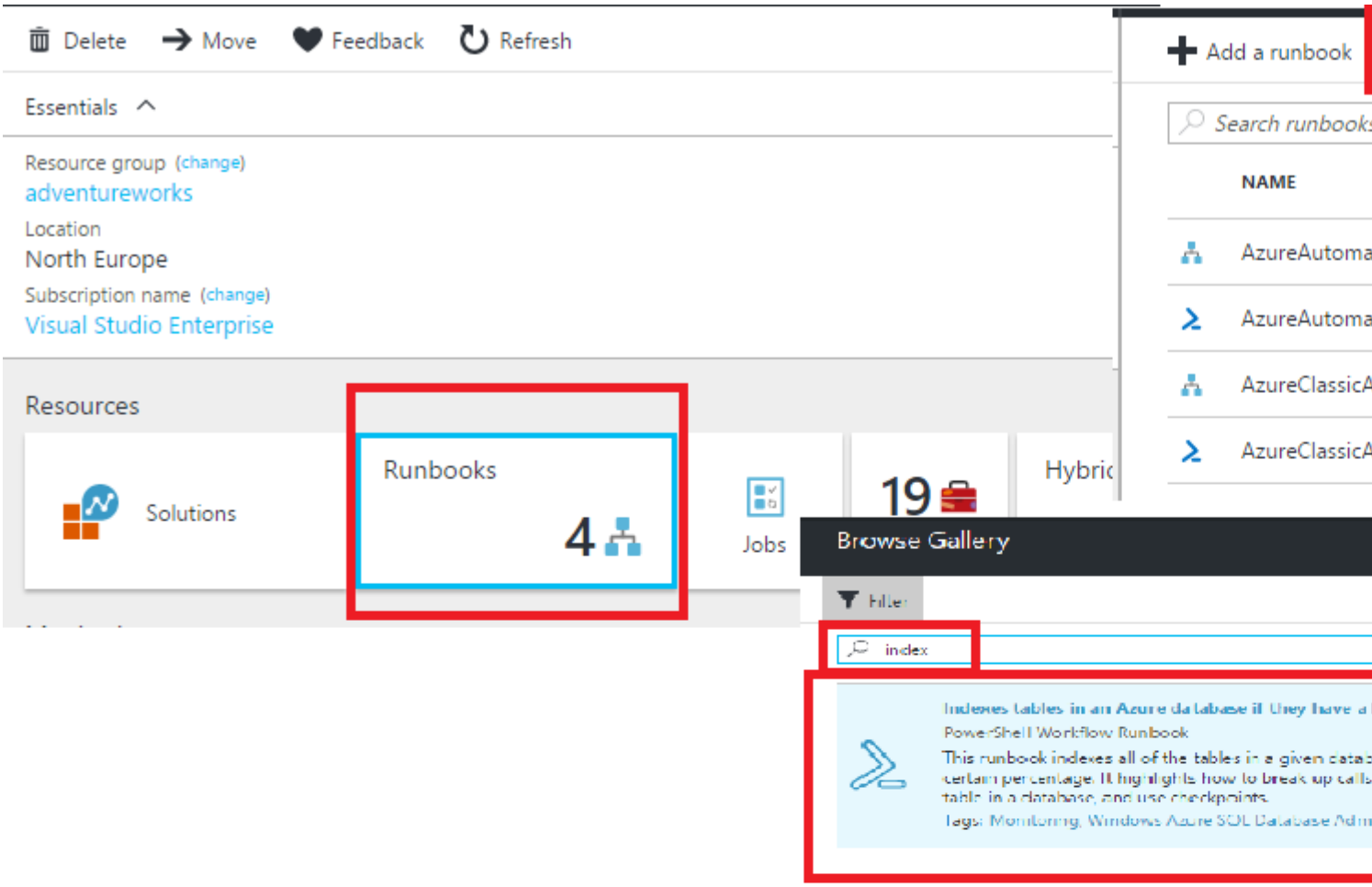
Mandatory, String

* CONTAINERNAME

Enter a value

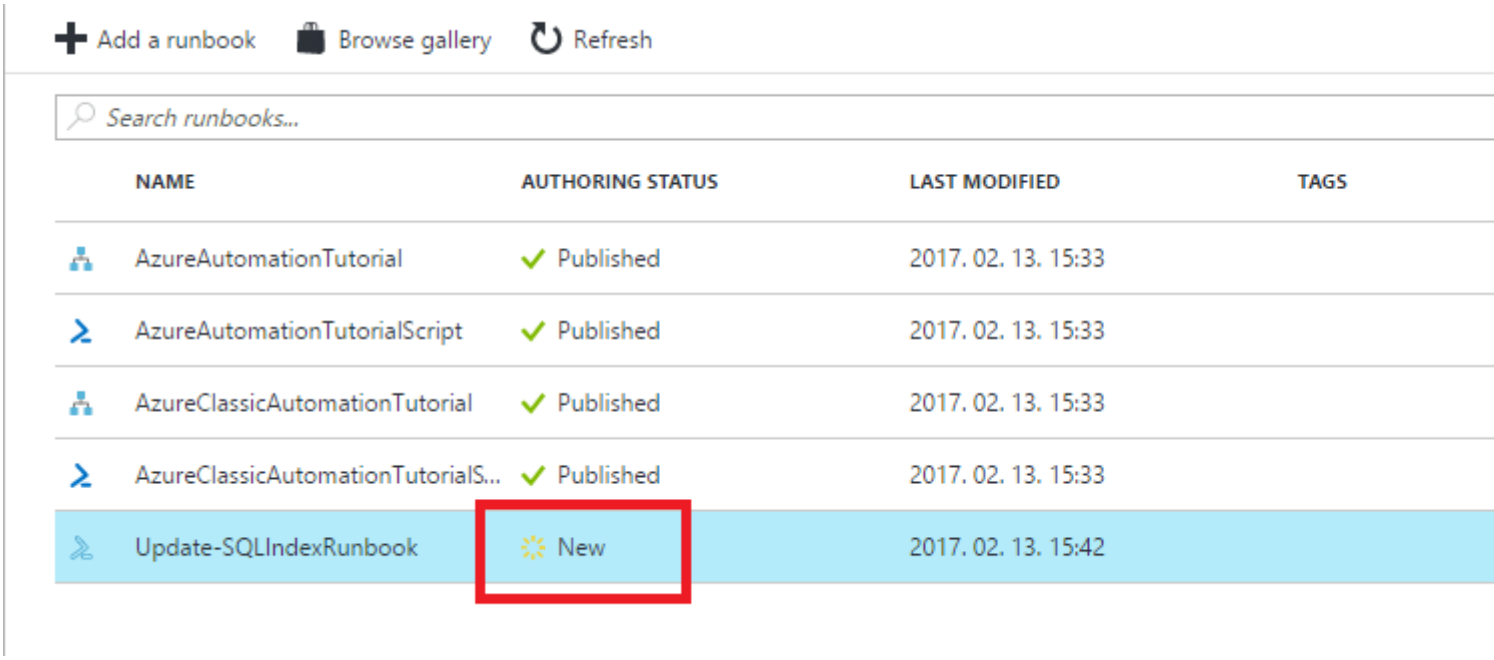
Mandatory, String

があります。にそれらにをえるやをにうはにしてください。 Azure Automationは、されたのをするためにすることができ、スケジュールをできるほぼのったランブックをします。あなたがしなければならないことはのとおりですまず、ランブックをインポートします。

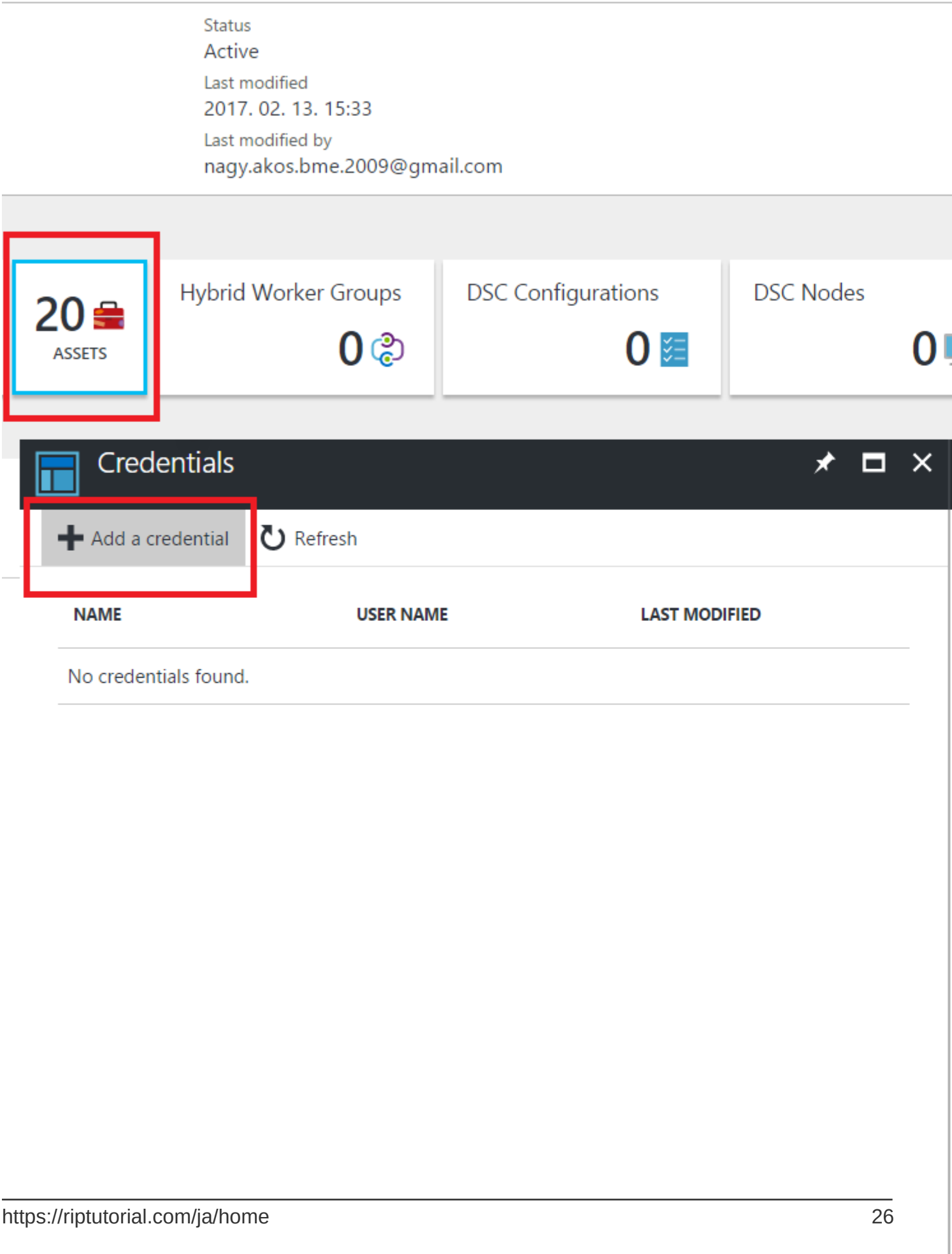


ランブックをインポートした、モードにり、ランブックでをすとアクティブになります。モード

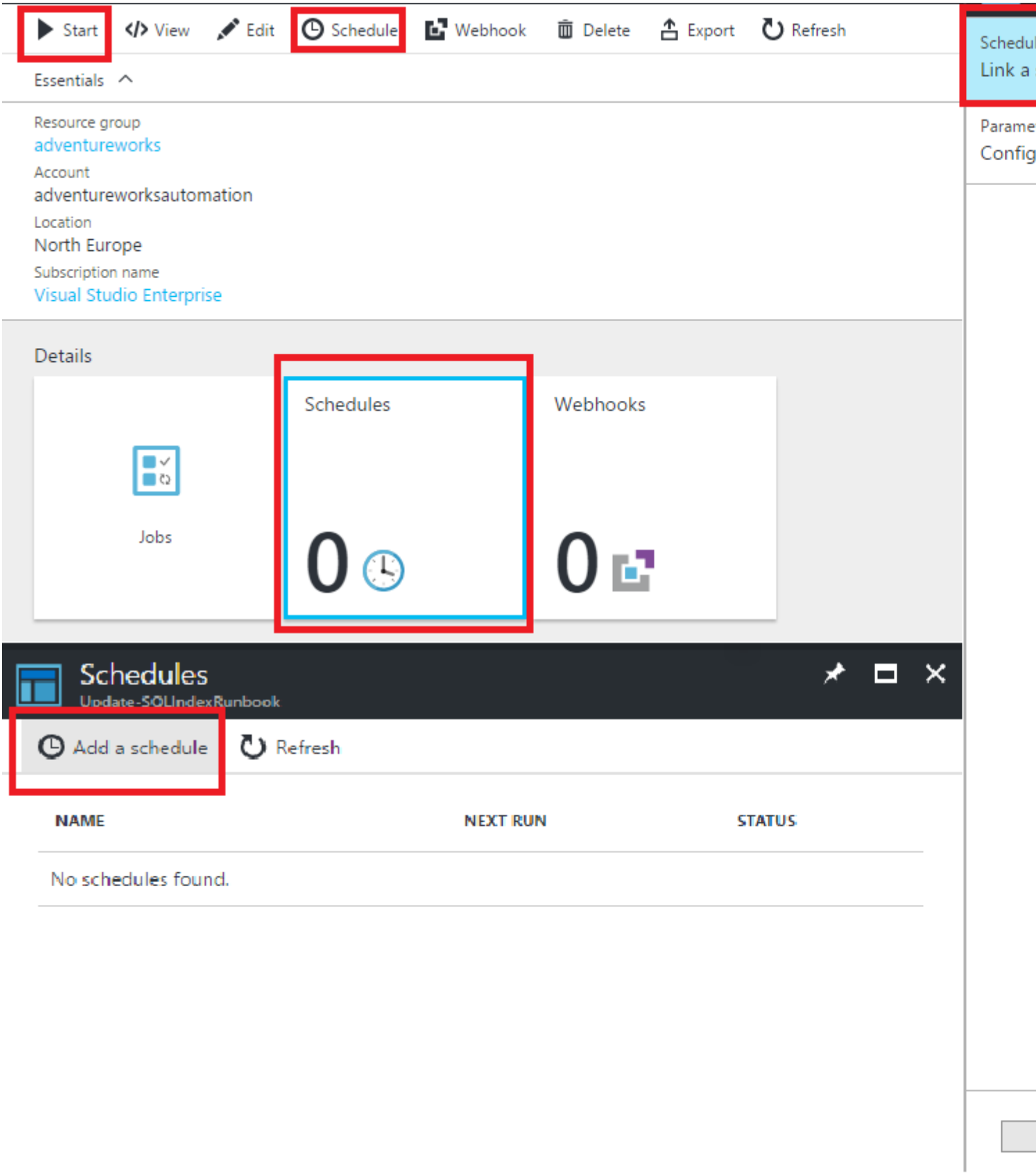
では、ランブックのソースコードをチェックアウトすることもできます。



に、データベースにするためにできるランブでをするがありますにとユーザーとパスワードをキーとのペアにして、そのキーをスクリプトからすることができますこののをする。これは、データベースにしてできるユーザーとパスワードのペアでなければならず、ユーザーはデータベースのにアクセスし、ALTER INDEXステートメントをするをっているがあります。


Status
Active
Last modified
2017. 02. 13. 15:33
Last modified by
nagy.akos.bme.2009@gmail.com
20
ASSETS
Hybrid Worker Groups
0
DSC Configurations
0
DSC Nodes
0
Credentials
Add a credential
Refresh
NAME
USER NAME
LAST MODIFIED
No credentials found.

に、されたパラメータでランブックをスケジュールします。 runbookをテストしてすぐにすることもできますただし、このもパラメータをするがあります。



なことに、はこのデフォルトランブックに2つのがあることをしました。

まず、のスキーマにあるテーブルのみをします。それだけではではありませんので、にんでください

```
SELECT  t.name AS TableName, t.OBJECT_ID FROM sys.tables t
```

それをのようにします

```
SELECT  '['+SCHEMA_NAME(t.schema_id)+'].['+t.name+']' AS TableName, t.OBJECT_ID FROM
sys.tables t
```

また、スクリプトでは、"、"などのどこにでもをうことができません。これらをするには、なをエスケープしてするか、ツールをすることができます。 2つのがあるはずです、ビルダーをするようにします。

```
$connStringBuilder = New-Object System.Data.SqlClient.SqlConnectionStringBuilder
$connStringBuilder["Server"] = "tcp:$using:SqlServer,$using:SqlServerPort"
$connStringBuilder["Database" ] = "$using:Database"
$connStringBuilder["User ID"] = "$using:SqlUsername"
$connStringBuilder["Password"] = "$using:SqlPass"
$connStringBuilder["Trusted_Connection"] = $False
$connStringBuilder["Encrypt"] = $True
$connStringBuilder["Connection Timeout"] = "30"
$connString = $connStringBuilder.ConnectionString
$Conn = New-Object System.Data.SqlClient.SqlConnection($connString)
```

にフラグメンテーションをするクエリをしすることもできます。これがせです。

```
SELECT a.object_id, avg_fragmentation_in_percent
    FROM sys.dm_db_index_physical_stats (
           DB_ID(N'$Database')
         , OBJECT_ID(0)
         , NULL
         , NULL
         , NULL) AS a
    JOIN sys.indexes AS b
    ON a.object_id = b.object_id AND a.index_id = b.index_id;
```

sys.dm_db_index_physical_statsののパラメータをNULLから 'DETAILED'にすると、インデックスがどのようにされているかをよりにできます。

はこのバージョンをGithubにアップロードしました https : //github.com/conwid/IndexRebuildScript

オンラインでAzure-オートメーションをむ https://riptutorial.com/ja/azure/topic/7258/azure-オートメーション

7: Azureサービスファブリック

Azureサービスファブリックは、Azureによってされるサービスの1つです。 Computeサービス Webロールとワーカーロールとはなり、コードは1つまたはのマシンでされるのではなく、コンテナでされ、のサービスとされます。

ここでは、またMicrosoftのやService Fabricにするドキュメントでは、 コンテナは「Docker」スタイルのコンテナではなく、よりなでされることにしてください。

クラスタをするのコンテナをつことができますはっています。コンテナでされるサービスは、「」ので、

- ゲストのファイル
- "できる"サービス
    - ステートフル
    - ステートレス
- 「できる」

Examples

できる

サービスファブリックのアクタは、の.NETインターフェイス/クラスのペアによってされます。

```
public interface IMyActor : IActor
{
    Task<string> HelloWorld();
}


internal class MyActor : Actor, IMyActor
{
    public Task<string> HelloWorld()
    {
        return Task.FromResult("Hello world!");
    }
}
```

インターフェイス/クラスのペアのすべてのメソッドはでなければならず、outまたはrefの paramersをつことはできません。

アクターモデルについてえると、なぜメッセージによってオブジェクトがにするのかをするのはです。メッセージは、メソッドをしてアクタークラスにされます。はアクターランタイムアクター「コンテナ」によってされ、びしにされます。

サービスファブリックSDKは、コンパイルにプロキシをします。このプロキシは、アクタークライアントがそのメソッドをびすつまり、アクターにメッセージをしてを`await` `await`されます。

クライアントは、IDをじてアクタをする。 IDはにることができますDB、のアクタからしたもの、またはそのアクタにリンクされたuserID、またはのオブジェクトのシリアルです。

しいアクター—をするがあり、IDだけがな、されたActorIdクラスには、ランダムにされたアクター—IDをするメソッドがあります

```
ActorId actorId = ActorId.NewId();
```

に、 `ActorProxy`クラスをして、 `ActorProxy`のプロキシオブジェクトをできます。アクター—をアクティブにしたり、まだどのメソッドもびすことはありません。 IMyActor myActor = ActorProxy.CreateactorId、しいUri "fabric/ MyApp / MyActorService";

に、プロキシをしてアクタのメソッドをびすことができます。されたIDをつアクタがしないは、アクティブになりクラスタのコンテナの1つにされます、にアクタにメッセ—ジがされ、メソッドびしがされ、アクタえ

```
await myActor.HelloWorld();
```

オンラインでAzureサービスファブリックをむ https://riptutorial.com/ja/azure/topic/3802/azureサービスファブリック

8: Azureストレージオプション

Examples

Azure Blobストレージでblobファイルのをする

AzureでblobファイルのをできるAPIはありません。このコードスニペットは、Microsoft Azure Blob Storageのblobファイルのをするをしています。

```
StorageCredentials cred = new StorageCredentials("[Your storage account name]", "[Your storage
account key]");

CloudBlobContainer container = new CloudBlobContainer(new Uri("http://[Your storage account
name].blob.core.windows.net/[Your container name] /"), cred);

string fileName = "OldFileName";
string newFileName = "NewFileName";

CloudBlockBlob blobCopy = container.GetBlockBlobReference(newFileName);

if (!await blobCopy.ExistsAsync())
{
    CloudBlockBlob blob = container.GetBlockBlobReference(fileName);

    if (await blob.ExistsAsync())
    {
        await blobCopy.StartCopyAsync(blob);
        await blob.DeleteIfExistsAsync();
    }
}
```

については、Azure Blobストレージでblobファイルのをするをしてください。

ASP.NETのAzure SQL ServerへのAzure Excelファイルのインポート/エクスポート

このサンプルは、AzureワークシートAzure ExcelファイルblobをAzure SQL ServerのDBにインポートすると、DBからAzure Excel blobにエクスポートするをしています。

- Microsoft Visual Studio 2015バージョン
- Microsoft OfficeのオープンXML SDK 2.5
- Azureストレージアカウント
- Azure SQL Server

DocumentFormat.OpenXmlをプロジェクトにします。

1. DBからAzure Excel BLOBにデータをエクスポートする
   してサーバーストレージにし、Azureにアップロードします。

```
public static string DBExportToExcel()
{
    string result = string.Empty;
    try
    {
        //Get datatable from db
        DataSet ds = new DataSet();
        SqlConnection connection = new SqlConnection(connectionStr);
        SqlCommand cmd = new SqlCommand($"SELECT {string.Join(",", columns)} FROM
{tableName}", connection);
        using (SqlDataAdapter adapter = new SqlDataAdapter(cmd))
        {
            adapter.Fill(ds);
        }
        //Check directory
        if (!Directory.Exists(directoryPath))
        {
            Directory.CreateDirectory(directoryPath);
        }
        // Delete the file if it exists
        string filePath = $"{directoryPath}//{excelName}";
        if (File.Exists(filePath))
        {
            File.Delete(filePath);
        }

        if (ds.Tables.Count > 0 && ds.Tables[0] != null || ds.Tables[0].Columns.Count > 0)
        {
            DataTable table = ds.Tables[0];

            using (var spreadsheetDocument = SpreadsheetDocument.Create(filePath,
SpreadsheetDocumentType.Workbook))
            {
                // Create SpreadsheetDocument
                WorkbookPart workbookPart = spreadsheetDocument.AddWorkbookPart();
                workbookPart.Workbook = new Workbook();
                var sheetPart = spreadsheetDocument.WorkbookPart.AddNewPart<WorksheetPart>();
                var sheetData = new SheetData();
                sheetPart.Worksheet = new Worksheet(sheetData);
                Sheets sheets =
spreadsheetDocument.WorkbookPart.Workbook.AppendChild<Sheets>(new Sheets());
                string relationshipId =
spreadsheetDocument.WorkbookPart.GetIdOfPart(sheetPart);
                Sheet sheet = new Sheet() { Id = relationshipId, SheetId = 1, Name =
table.TableName };
                sheets.Append(sheet);

                //Add header to sheetData
                Row headerRow = new Row();
                List<String> columns = new List<string>();
                foreach (DataColumn column in table.Columns)
                {
                    columns.Add(column.ColumnName);

                    Cell cell = new Cell();
                    cell.DataType = CellValues.String;
                    cell.CellValue = new CellValue(column.ColumnName);
                    headerRow.AppendChild(cell);
                }
                sheetData.AppendChild(headerRow);
```

```
                //Add cells to sheetData
                foreach (DataRow row in table.Rows)
                {
                    Row newRow = new Row();
                    columns.ForEach(col =>
                    {
                        Cell cell = new Cell();
                        //If value is DBNull, do not set value to cell
                        if (row[col] != System.DBNull.Value)
                        {
                            cell.DataType = CellValues.String;
                            cell.CellValue = new CellValue(row[col].ToString());
                        }
                        newRow.AppendChild(cell);
                    });
                    sheetData.AppendChild(newRow);
                }
                result = $"Export {table.Rows.Count} rows of data to excel successfully.";
            }
        }

        // Write the excel to Azure storage container
        using (FileStream fileStream = File.Open(filePath, FileMode.Open))
        {
            bool exists = container.CreateIfNotExists();
            var blob = container.GetBlockBlobReference(excelName);
            blob.DeleteIfExists();
            blob.UploadFromStream(fileStream);
        }
    }
    catch (Exception ex)
    {
        result =$"Export action failed. Error Message: {ex.Message}";
    }
    return result;
}
```

2. Azure ExcelファイルをDBにインポートする
   BLOBデータをみることはできないので、それをサーバーストレージにしてからするがあります。
   SqlBulkCopyをして、dbにデータをします。

```
public static string ExcelImportToDB()
{
    string result = string.Empty;
    try
    {
        //Check directory
        if (!Directory.Exists(directoryPath))
        {
            Directory.CreateDirectory(directoryPath);
        }
        // Delete the file if it exists
        string filePath = $"{directoryPath}//{excelName}";
        if (File.Exists(filePath))
        {
            File.Delete(filePath);
        }
```

```
        // Download blob to server disk.
        container.CreateIfNotExists();
        CloudBlockBlob blob = container.GetBlockBlobReference(excelName);
        blob.DownloadToFile(filePath, FileMode.Create);

        DataTable dt = new DataTable();
        using (SpreadsheetDocument spreadSheetDocument = SpreadsheetDocument.Open(filePath,
false))
        {
            //Get sheet data
            WorkbookPart workbookPart = spreadSheetDocument.WorkbookPart;
            IEnumerable<Sheet> sheets =
spreadSheetDocument.WorkbookPart.Workbook.GetFirstChild<Sheets>().Elements<Sheet>();
            string relationshipId = sheets.First().Id.Value;
            WorksheetPart worksheetPart =
(WorksheetPart)spreadSheetDocument.WorkbookPart.GetPartById(relationshipId);
            Worksheet workSheet = worksheetPart.Worksheet;
            SheetData sheetData = workSheet.GetFirstChild<SheetData>();
            IEnumerable<Row> rows = sheetData.Descendants<Row>();

            // Set columns
            foreach (Cell cell in rows.ElementAt(0))
            {
                dt.Columns.Add(cell.CellValue.InnerXml);
            }

            //Write data to datatable
            foreach (Row row in rows.Skip(1))
            {
                DataRow newRow = dt.NewRow();
                for (int i = 0; i < row.Descendants<Cell>().Count(); i++)
                {
                    if (row.Descendants<Cell>().ElementAt(i).CellValue != null)
                    {
                        newRow[i] = row.Descendants<Cell>().ElementAt(i).CellValue.InnerXml;
                    }
                    else
                    {
                        newRow[i] = DBNull.Value;
                    }
                }
                dt.Rows.Add(newRow);
            }
        }

        //Bulk copy datatable to DB
        SqlBulkCopy bulkCopy = new SqlBulkCopy(connectionStr);
        try
        {
            columns.ForEach(col => { bulkCopy.ColumnMappings.Add(col, col); });
            bulkCopy.DestinationTableName = tableName;
            bulkCopy.WriteToServer(dt);
        }
        catch (Exception ex)
        {
            throw ex;
        }
        finally
        {
            bulkCopy.Close();
        }
```

```
        result = $"Import {dt.Rows.Count} rows of data to DB successfully.";
    }
    catch (Exception ex)
    {
        result = $"Import action failed. Error Message: {ex.Message}";
    }
    return result;
}
```

については、 https//code.msdn.microsoft.com/How-to-ImportExport-Azure-0c858df9をしてください 。

Microsoft Azureでブロブストレージのロックをする

Microsoft AzureのblobストレージのロックをできるAPIはありません。このコードスニペットは、Microsoft AzurePowerShellのblobストレージのロックされたリースをするをしています。

```
 $key = (Get-AzureRmStorageAccountKey -ResourceGroupName
$selectedStorageAccount.ResourceGroupName -name $selectedStorageAccount.StorageAccountName -
ErrorAction Stop)[0].value
        $storageContext = New-AzureStorageContext -StorageAccountName
$selectedStorageAccount.StorageAccountName -StorageAccountKey $key -ErrorAction Stop
        $storageContainer = Get-AzureStorageContainer -Context $storageContext -Name
$ContainerName -ErrorAction Stop
        $blob = Get-AzureStorageBlob -Context $storageContext -Container  $ContainerName -Blob
$BlobName -ErrorAction Stop
        $leaseStatus = $blob.ICloudBlob.Properties.LeaseStatus;
        If($leaseStatus -eq "Locked")
        {
             $blob.ICloudBlob.BreakLease()
             Write-Host "Successfully broken lease on '$BlobName' blob."
        }
        Else
        {
            #$blob.ICloudBlob.AcquireLease($null, $null, $null, $null, $null)
            Write-Host "The '$BlobName' blob's lease status is unlocked."
        }
```

については、Microsoft AzurePowerShellでARMによってBlobストレージのロックされたリースをするをしてください。

オンラインでAzureストレージオプションをむ https://riptutorial.com/ja/azure/topic/5405/azureストレージオプション

9: Azureストレージオプション

Examples

Azureストレージキューにする

Azureのストレージオプションは、 "REST" APIまたは、よりいHTTP APIをします。

Azure SDKは、ののクライアントをします。たとえば、Cクライアントライブラリをしてストレージオブジェクトキューの1つをするをてみましょう。

Azure Storageへのすべてのアクセスは、ストレージアカウントをじてわれます。ストレージアカウントは、ポータル、Azure CLI、PowerShell、Azure Resource ManagerARMなどのいくつかのでできます。

このでは、すでに`app.config`ファイルにしているとします。

```
// Retrieve storage account from connection string.
CloudStorageAccount storageAccount = CloudStorageAccount.Parse(
    CloudConfigurationManager.GetSetting("StorageConnectionString"));
```

キューには、のURLでアクセスできます`http://<storage account>.queue.core.windows.net/<queue>`

クライアントライブラリはこのURLをします。キューでなければなりませんをするだけです。のステップは、キューをするためにされるキュークライアントへのをすることですキューはされたストレージアカウントにまれています。

```
CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();
```

クライアントをしてキューへのをします。

```
CloudQueue queue = queueClient.GetQueueReference("<queue>");
```

これで、この`queue`プロキシをして、のを`queue`ることができます。

、のは、キューがまだしないにすることです

```
queue.CreateIfNotExists();
```

のをします。なぜ "しなければ"いくつかのがあります。

- このコードをする「か」ののインスタンスをしているがあります「か」は、WebロールまたはworkerロールのようなCompute Serviceですが、Web App、Fabric Service、 VM ...
- あなたのアプリケーションはいつでもリブートすることができます。これは、にPaaSサービスの、インスタンスがであるクラウドであることをえておいてください。あなたは、ロー

カルにされたアプリとじにあなたのアプリケーションをするはありません。

さらに、じAPIびしのバージョンをするがあります。

```
await queue.CreateIfNotExistsAsync();
```

このではキューをしましたが、このはのストレージオブジェクトブロブ、テーブル、ファイルににできます。

ストレージオブジェクトをしたら、そのストレージオブジェクトをするがいました。

オンラインでAzureストレージオプションをむ https://riptutorial.com/ja/azure/topic/6008/azureストレージオプション

10: Azureマシン

Examples

クラシックASM APIをしてAzure VMをする

```
# 1. Login Azure by admin account
Add-AzureAccount
#
# 2. Select subscription name
$subscriptionName = Get-AzureSubscription | Select -ExpandProperty SubscriptionName
#
# 3. Create storage account
$storageAccountName = $VMName
# here we use VMName to play the storage account name and create it, you can choose your name
or use existed one to replace the storage account creation operation
New-AzureStorageAccount -StorageAccountName $storageAccountName -Location $Location | Out-Null
#
# 4. Select subscription name and storage account name for current context
Select-AzureSubscription -SubscriptionName $subscriptionName -Current | Out-Null
Set-AzureSubscription -SubscriptionName $subscriptionName -CurrentStorageAccountName
$storageAccountName | Out-Null
#
# 5. Select a VM image name
$label = $VMLabelPattern
# take care, please ensure the VM image location resides to the same location of your storage
account and service below
$imageName = Get-AzureVMImage | where { $_.Label -like $label } | sort PublishedDate -
Descending | select -ExpandProperty ImageName -First 1
#
# 6. Create cloud service
$svcName = $VMName
# here we use VMName to play the service name and create it, you can choose your name or use
existed one to replace the service creation operation
New-AzureService -ServiceName $svcName -Location $Location | Out-Null
#
# 7. Build command set
$vmConfig = New-AzureVMConfig -Name $VMName -InstanceSize $VMSize -ImageName $imageName
#
# 8. Set local admin of this vm
$cred=Get-Credential -Message "Type the name and password of the local administrator account."
$vmConfig | Add-AzureProvisioningConfig -Windows -AdminUsername $cred.Username -Password
$cred.GetNetworkCredential().Password
#
# 9. Execute the final cmdlet to create the VM
New-AzureVM -ServiceName $svcName -VMs $vmConfig | Out-Null
```

については、 のASM APIをしてPowershellでAzureマシンVMをするをしてください。

オンラインでAzureマシンをむ https://riptutorial.com/ja/azure/topic/6350/azureマシン

クレジット

S. No		Contributors
1	のまり	awh112, Bernard Vander Beken, Community, KARANJ, lorenzo montanari, Sibeesh Venu, user2314737
2	Azure DocumentDB	gbellmann, Matias Quaranta
3	Azure Mediaサービスアカウント	Sibeesh Venu
4	Azure Powershell	Anton Purin, CmdrTchort, frank tan, juunas, RedGreenCode
5	Azure Resource Managerテンプレート	BenV
6	Azure-オートメーション	Akos Nagy, RuSs
7	Azureサービスファブリック	Lorenzo Dematté, Stephen Leppik
8	Azureストレージオプション	Dale Chen, Gaurav Mantri
9	Azureマシン	Dale Chen