

 免費電子書

學習

azure

Free unaffiliated eBook created from
Stack Overflow contributors.

#azure

.....	1
1: azure	2
.....	2
Examples.....	2
Azure NGPUUBUNTU 16.04 LTSCUDAcudnnTensorflow.....	2
2: Azure DocumentDB	4
Examples.....	4
.NET.....	4
.NET.....	4
.NET.....	5
JSON.NET.....	6
.NET.....	7
LINQ	7
SQL	7
LINQ	7
.NET.....	8
.NET.....	9
.NET.....	9
3: Azure Powershell	10
Examples.....	10
ARM.....	10
Azure.....	10
.....	10
Azure PowerShell.....	11
Azure.....	11
.....	11
.....	11
TrafficManager.....	11
.....	12
.....	12
4: Azure Service Fabric	13
.....	

Examples.....	13
.....	13
5: Azure	15
.....	15
Examples.....	15
.....	15
.....	15
6: Azure	16
Examples.....	16
Azure Blobblob.....	16
ASP.NETAzure Excel/Azure SQL Server.....	16
Microsoft Azureblob.....	20
7: Azure	21
Examples.....	21
Azure.....	21
8: Azure	22
.....	22
.....	22
Examples.....	22
BlobBlob.....	22
.....	26
9: Azure	29
Examples.....	29
ASM APIAzure VM.....	29
10: Azure	30
.....	30
Examples.....	30
.....	30
.....	32

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [azure](#)

It is an unofficial and free azure ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official azure.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

1: azure

AzureMicrosoft。 Microsoft Azure

- IaaS Linux Windows Azure
- PaaS App Service Web
- SQL NoSQL
- SaaS

Azure <https://azure.microsoft.com/en-us/resources/infographics/azure/>。 Azure。

Examples

Azure NGPU UBUNTU 16.04 LTS CUDA cudnn Tensorflow

5

- CUDA GPU

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

NVIDIA Tesla K80 / M60

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

- nouveau

```
sudo -i  
rmmod nouveau
```

- sudo reboot sudo reboot

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

- Nvidia CUDA.....

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

-...apt-get CUDA Toolkit

```
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

```
nvidia-smi
```

cuDNN ...

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

-...lib64

```
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
```

-

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

CPU / GPU Tensorflow

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

azure <https://riptutorial.com/zh-TW/azure/topic/1060/azure>

2: Azure DocumentDB

Examples

.NET

DocumentDBURI DocumentClient ◦

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

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

.NET

DocumentClientCreateDatabaseAsync DocumentDB ◦ JSON ◦

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

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

```
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;
        }
    }
}
```

```
}
```

.NET

DocumentClientCreateDocumentCollectionAsync ◦ JSONJavaScript ◦

```
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 });
}
```

```
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.NET

CreateDocumentAsyncDocumentClient ◦ JSON ◦

```
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;
        }
    }
}
```

```
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; }
}

```

.NET

DocumentDBJSON。

LINQ

```

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

```

SQL

```

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

```

LINQ

FeedOptionsRequestContinuation

```

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));
}

```

Continuation Token

```

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;
    }
}

```

.NET

DocumentDB DocumentClient ReplaceDocumentAsync JSON

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

.NET

DocumentDB DocumentClient DeleteDocumentAsync JSON

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

.NET

。

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

Azure DocumentDB <https://riptutorial.com/zh-TW/azure/topic/5176/azure-documentdb>

3: Azure Powershell

Examples

ARM

PowerShellAzureMicrosoft

“”

AzureAzure。 AzureARM。

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

“ARM”

AzureREST API。 PowershellAzure。 ARMAzure。

ARM

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

Azure

```
Add-AzureAccount
```

Azure Active DirectoryPowerShell12。 12。

cmdlet

```
Get-AzurePublishSettingsFile
```

。 PowerShell。

```
Import-AzurePublishSettingsFile
```

。 。

Azure Active Directory12。

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

Azure;。;

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

```
Select-AzureRmSubscription
```

ID.

```
Get-AzureSubscription
```

Azure PowerShell

Azure PowerShell

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

PowerShellAzure PowerShell.

Azure

Azure CmdletCAzurePowerShellAzure.

Azure blob

```
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\
```

Azure PowerShell[Azure Portal](#)

- Traffic Manager
- Azure `ResourceId`Azure

[RM](#) .

TrafficManager

PowerShell

1. TM

```
$profile = Get-AzureRmTrafficManagerProfile -ResourceGroupName my-resource-group -Name my-traffic-manager
```

°

2. TM

```
$profile$profile.Endpoints°
```

3. Set-AzureRmTrafficManagerProfile -TrafficManagerProfile \$profile°

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

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

$profile.Endpoints.Clear()

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

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
```

ResourceIdAzure

```
Set-AzureRmTrafficManagerProfile -TrafficManagerProfile $profile TM TM
```

```
DNSIPTTL $profile.Ttl TMTTL
```

Azure Powershell <https://riptutorial.com/zh-TW/azure/topic/3961/azure-powershell>

4: Azure Service Fabric

Azure Service Fabric Azure PaaS. Compute Web.

Service Fabric Microsoft "Docker".

- ""
 - ""
 - ""
 - ""
 - ""

Examples

Service Fabric actor.NET/

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

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

/outref.

- actor; actor actor "".

Service Fabric SDK. actor actor await.

IDActor. IDDBActor actor userID.

actor IDActor ID actor ID

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

```
ActorProxy actor = actor. IMyActor myActor = ActorProxy.Create(actorId, new Uri("fabric/ MyApp / MyActorService");
```

actor. IDActor actor actor Task


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

Azure Service Fabric <https://riptutorial.com/zh-TW/azure/topic/3802/azure-service-fabric>

5: Azure

AzureAzure

Examples

```
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;
    }
}
```

```
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;
    }
}
```

Azure <https://riptutorial.com/zh-TW/azure/topic/4997/azure>

6: Azure

Examples

Azure Blobblob

AzureblobAPI。 Microsoft Azure Blobblob。

```
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();
    }
}
```

Azure Blobblob

ASP.NET Azure Excel/Azure SQL Server

Azure Excelblob Azure SQL Server Azure Excel blob。

- Microsoft Visual Studio 2015
- [Microsoft OfficeXML SDK 2.5](#)
- Azure
- Azure SQL Server

DocumentFormat.OpenXml。

1. DB Azure Excel blob
excel Azure。

```
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 ExcelDB

excel blob.

[SqlBulkCopy](#)db.

```

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;

```

```
}
```

<https://code.msdn.microsoft.com/How-to-ImportExport-Azure-0c858df9> 。

Microsoft Azureblob

APIMicrosoft Azureblob。 Microsoft AzurePowerShellblob。

```
$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."
    }
}
```

Microsoft AzureARMblobPowerShell

Azure <https://riptutorial.com/zh-TW/azure/topic/5405/azure>

7: Azure

Examples

Azure

Azure“REST”API HTTP API

Azure SDK。 C。

Azure。 Azure CLI PowerShell Azure ARM...

app.config。

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

URL `http://<storage account>.queue.core.windows.net/<queue>`

URL;。 。

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

。

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

queue。

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

。 “”

- “” Web Web VM ...
- 。 PaaS。 。

API

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

blob。

。

Azure <https://riptutorial.com/zh-TW/azure/topic/6008/azure>

8: Azure

resourceGroupName	Azure
connectionName	Azure Run Asservice pricipal
StorageAccountName	Azure
	blob
DaysOld	blob

Azure Active Directory AzureRunAs. ◦

Examples

BlobBlob

Azure PowershellRunbookRunbookAzureBlob. ◦

SQL. ◦

◦

◦

Azure. Azure Active Directory. ◦

Add Automation Acco... — □ ×

* Name ?

* Subscription

▼

* Resource group ?


Create new Use existing

▼

* Location

▼

* Create Azure Run As account ?

 The Run As account feature will create a Run As account and a Classic Run As account. [Click here to learn more about Run As accounts.](#)

Pin to dashboard

```
<#
.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);
}
```

◦



Test

CleanUpStorage



Start



Stop



Suspend



Resume

Parameters

* RESOURCEGROUPNAME ⓘ

Mandatory, String

* CONNECTIONNAME ⓘ

Mandatory, String

* STORAGEACCOUNTNAME ⓘ

Mandatory, String

* CONTAINERNAME ⓘ

Mandatory, String

Comple

Loggin

Enviro

{[Azur

Storag

BlobE

Table

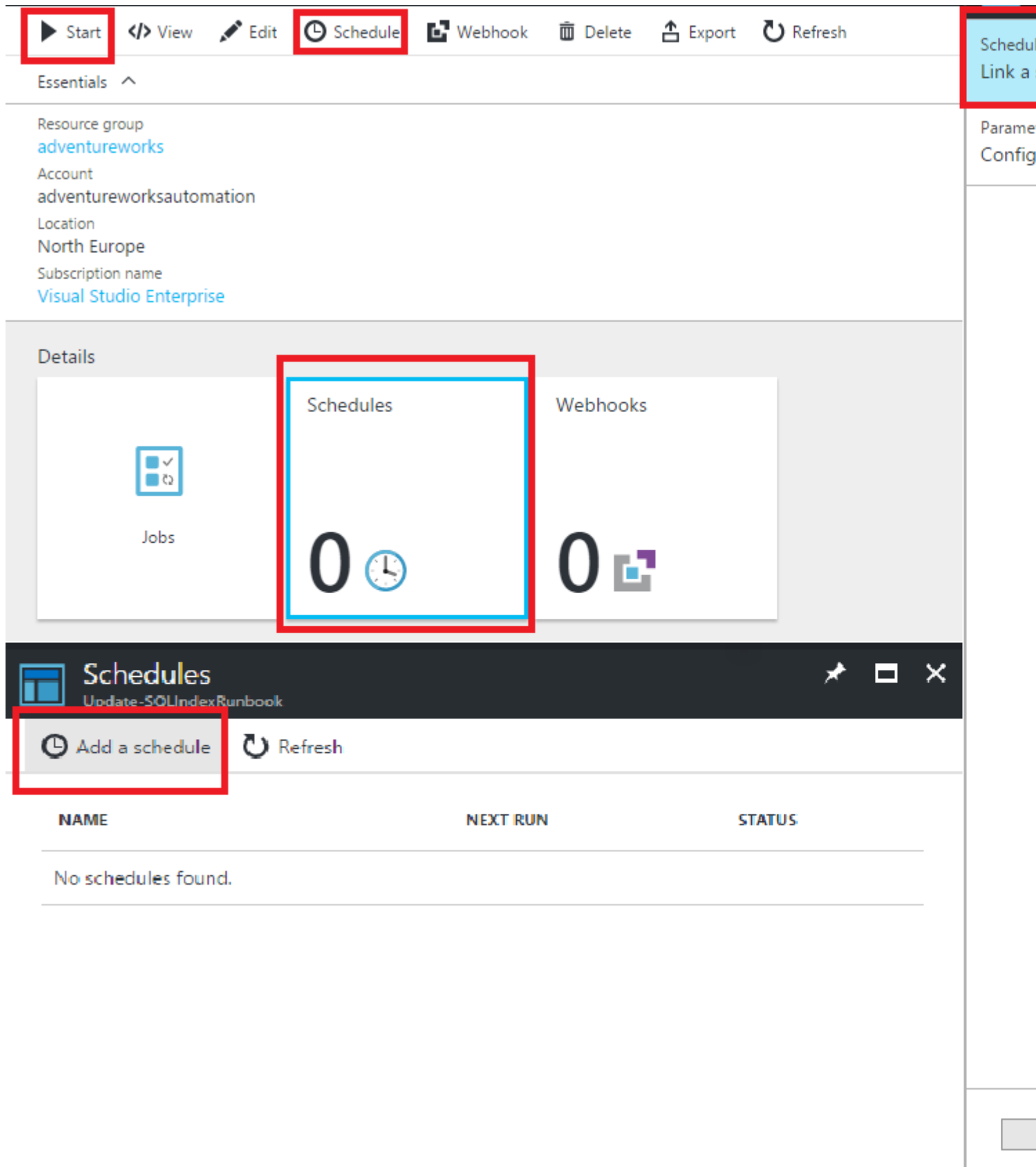
Queue

Contex

Name

Storag

Runbook: Runbook



Start View Edit Schedule Webhook Delete Export Refresh

Essentials ^

Resource group
adventureworks
Account
adventureworksautomation
Location
North Europe
Subscription name
Visual Studio Enterprise

Details

Jobs Schedules Webhooks

0 0

Schedules
Update-SQLIndexRunbook

Add a schedule Refresh

NAME	NEXT RUN	STATUS
No schedules found.		

Runbook.

o

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

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

“”=。 connectionstring。 connectionstring builder

```
$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)
```

o

```
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 NULL “DETAILED”。

Github <https://github.com/conwid/IndexRebuildScript>

Azure <https://riptutorial.com/zh-TW/azure/topic/7258/azure>

9: Azure

Examples

ASM APIAzure VM

```
# 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
```

ASM APIPowerShellAzureVM

Azure <https://riptutorial.com/zh-TW/azure/topic/6350/azure>

10: Azure

- ARM <https://azure.microsoft.com/en-us/documentation/articles/resource-group-authoring-templates/>

Examples

Azure◦

Azure Key VaultDiagnosticSettings◦

- resources
- dependsOnARM

```
{
  "$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]"
  }
}
}

```

Azure <https://riptutorial.com/zh-TW/azure/topic/3923/azure>

S. No		Contributors
1	azure	awh112 , Bernard Vander Beken , Community , KARANJ , lorenzo montanari , Sibeesh Venu , user2314737
2	Azure DocumentDB	gbellmann , Matias Quaranta
3	Azure Powershell	Anton Purin , CmdrTchort , frank tan , juunas , RedGreenCode
4	Azure Service Fabric	Lorenzo Dematté , Stephen Leppik
5	Azure	Sibeesh Venu
6	Azure	Dale Chen , Gaurav Mantri
7	Azure	Akos Nagy , RuSs
8	Azure	Dale Chen
9	Azure	BenV