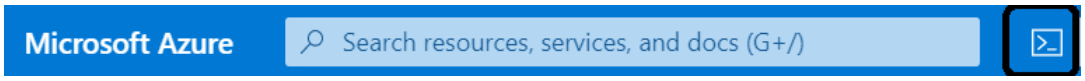
Experiment - Provisioning an Azure storage account using Cloud Shell

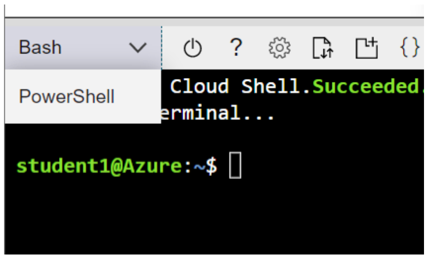
Cloud Shell uses the PowerShell is a scripting language used to programmatically manage various tasks. In this experiment, we'll learn to provision an Azure storage account using PowerShell.

Getting ready

Before you start, we need to log in to the Azure subscription from the Cloud Shell console. Open a new Cloud Shell.



By default, this will launch a Bash Cloud Shell. Switch to PowerShell by selecting the drop down option for PowerShell and select Confirm on the dialog box.

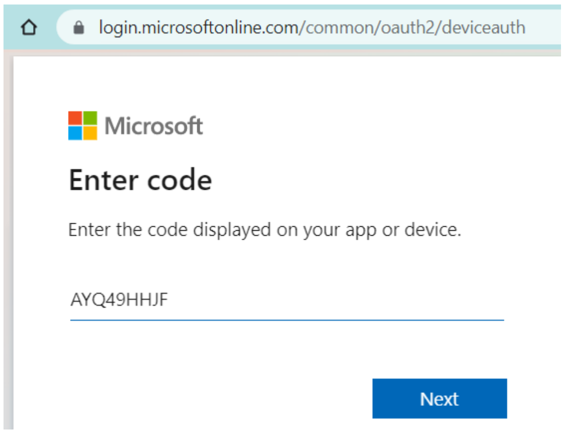


```
Cloud Shell.Succeeded.
erminal...

student1@Azure:~$
```

Execute the following command in a new Cloud Shell window:

```
Connect-AzAccount
```

Then, follow the instructions to log in to the Azure account. First open a web browser as noted and enter the code that was presented in PowerShell



login.microsoftonline.com/common/oauth2/deviceauth

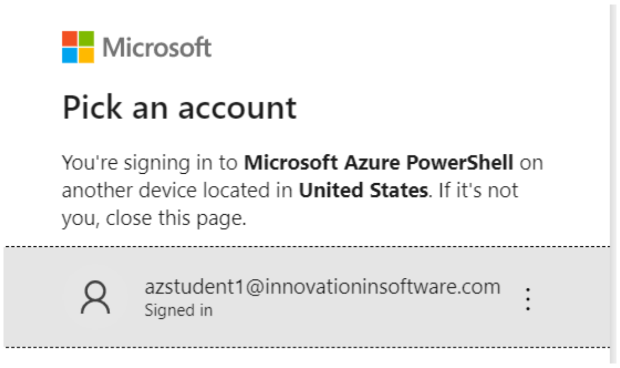
Microsoft

Enter code

Enter the code displayed on your app or device.

AYQ49HHJF

Select **Next**.


Microsoft

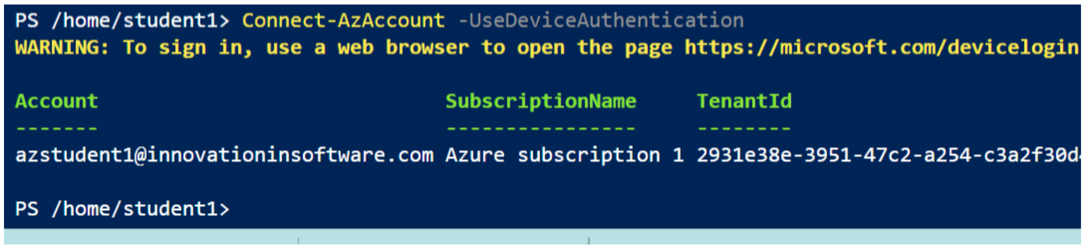
Pick an account

You're signing in to **Microsoft Azure PowerShell** on another device located in **United States**. If it's not you, close this page.

azstudent1@innovationinsoftware.com
Signed in

Choose your student account that was assigned for this session, in this example **azstudent1**

The next dialog will ask you if you're trying to sign into Microsoft Azure PowerShell. Select **Continue**.



```
PS /home/student1> Connect-AzAccount -UseDeviceAuthentication
WARNING: To sign in, use a web browser to open the page https://microsoft.com/devicelogin

Account                             SubscriptionName     TenantId
-------                             ----------------     --------
azstudent1@innovationinsoftware.com Azure subscription 1 2931e38e-3951-47c2-a254-c3a2f30d

PS /home/student1>
```

Note that the success messaging if all went as planned.

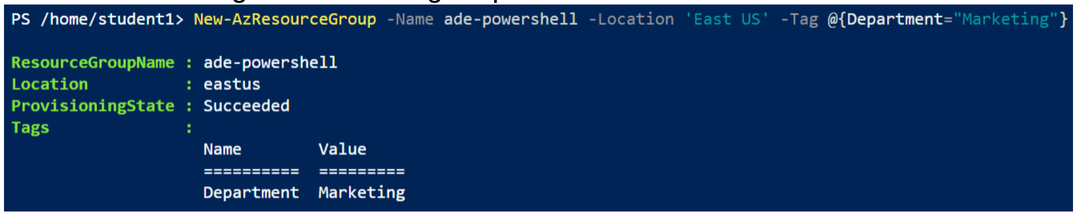
How to do it…

The steps for this experiment are as follows:

1. Execute the following command in a PowerShell window to create a new resource group. If you want to create the Azure storage account in an existing resource group, this step isn't required:

```
New-AzResourceGroup -Name ade-powershell -Location 'East US' -Tag
@{Department="Marketing"}
```

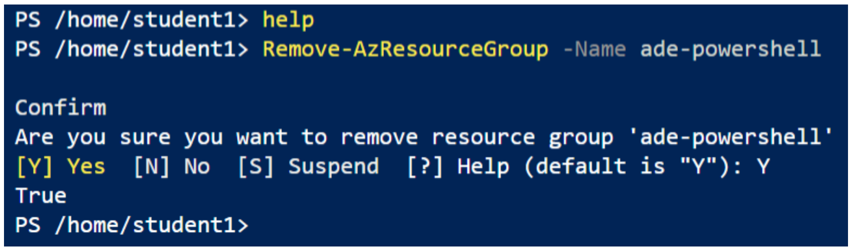
You should get the following output:



```
PS /home/student1> New-AzResourceGroup -Name ade-powershell -Location 'East US' -Tag @{Department="Marketing"}

ResourceGroupName : ade-powershell
Location          : eastus
ProvisioningState : Succeeded
Tags              :
                    Name        Value
                    ==========  =========
                    Department  Marketing
```

2. Execute the following command to remove the new resource group we just created.

```
Remove-AzResourceGroup -Name ade-powershell
```

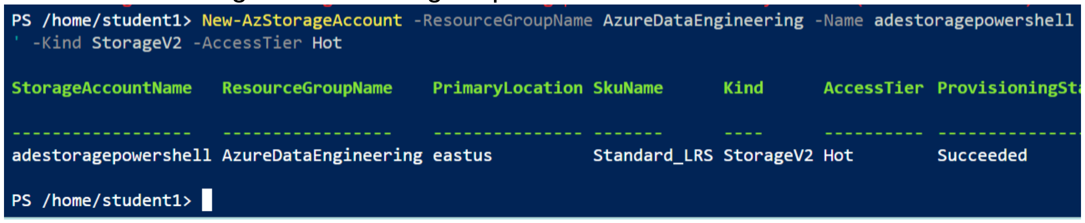


```
PS /home/student1> help
PS /home/student1> Remove-AzResourceGroup -Name ade-powershell

Confirm
Are you sure you want to remove resource group 'ade-powershell'
[Y] Yes  [N] No  [S] Suspend  [?] Help (default is "Y"): Y
True
PS /home/student1>
```

3. Execute the following command to create a new Azure storage account in the **AzureDataEngineering** resource group that was created in Experiment01:

    New-AzStorageAccount -ResourceGroupName AzureDataEngineering -Name adestoragepowershell -SkuName Standard_LRS -Location 'East US' -Kind StorageV2 -AccessTier Hot

    You should get the following output:



```
PS /home/student1> New-AzStorageAccount -ResourceGroupName AzureDataEngineering -Name adestoragepowershell
' -Kind StorageV2 -AccessTier Hot

StorageAccountName   ResourceGroupName    PrimaryLocation SkuName      Kind      AccessTier ProvisioningSt

------------------   -----------------    --------------- -------      ----      ---------- --------------
adestoragepowershell AzureDataEngineering eastus          Standard_LRS StorageV2 Hot        Succeeded

PS /home/student1> █
```

How it works…

There's a single command to create an Azure storage account using PowerShell – **New-AzStorageAccount**. The **SkuName** parameter specifies the performance tier and the **Kind** parameter specifies the account kind. In the later experiments, we'll look at how to assign public/private endpoints to an Azure storage account using PowerShell.

Creating containers and uploading files to Azure Blob storage using PowerShell

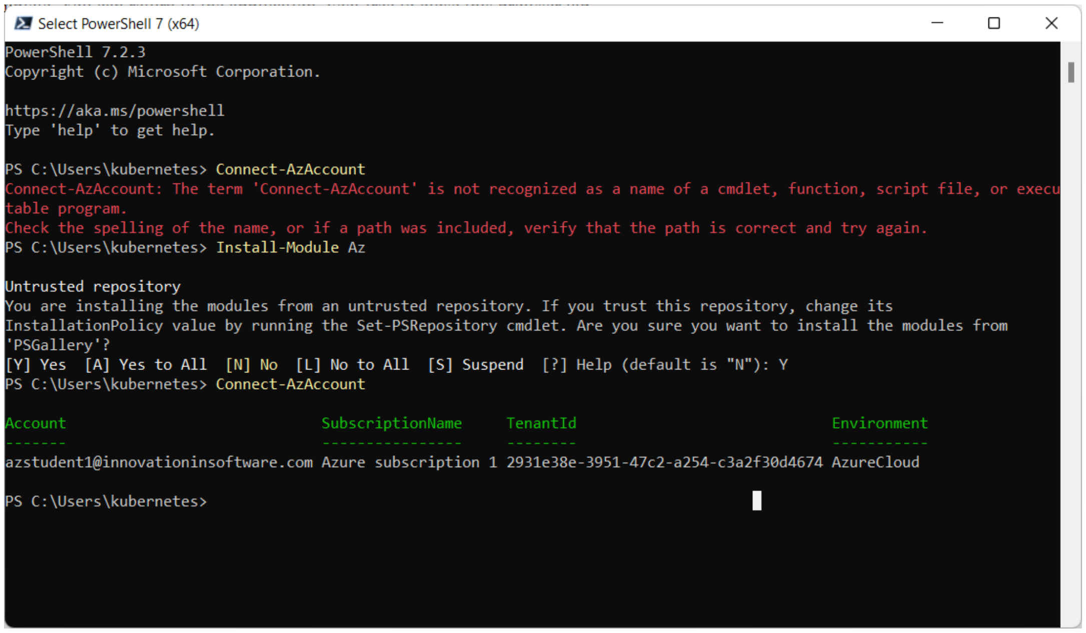
In this experiment, we'll create a new container and will upload files to Azure Blob storage using PowerShell.

Getting ready

Before you start, perform the following steps:

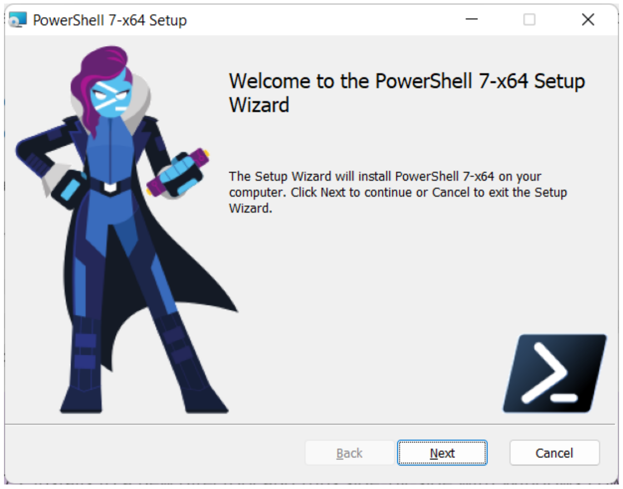
1. Make sure you have an existing Azure storage account. If not, create one by following the *Provisioning an Azure storage account using PowerShell* experiment.
2. Log in to your Azure subscription in PowerShell. To log in, run the **Connect-AzAccount** command in a new PowerShell window and follow the instructions.

We connected in the earlier portion of this experiment with Azure Portal Cloud Shell, but you could as easily do the CLI experiment from your Windows Workstation. If Connect-AzAccount fails then install the Azure module to proceed.



```
PowerShell 7.2.3
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

PS C:\Users\kubernetes> Connect-AzAccount
Connect-AzAccount: The term 'Connect-AzAccount' is not recognized as a name of a cmdlet, function, script file, or execu
table program.
Check the spelling of the name, or if a path was included, verify that the path is correct and try again.
PS C:\Users\kubernetes> Install-Module Az

Untrusted repository
You are installing the modules from an untrusted repository. If you trust this repository, change its
InstallationPolicy value by running the Set-PSRepository cmdlet. Are you sure you want to install the modules from
'PSGallery'?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N"): Y
PS C:\Users\kubernetes> Connect-AzAccount

Account                            SubscriptionName     TenantId                             Environment
-------                            ----------------     --------                             -----------
azstudent1@innovationinsoftware.com Azure subscription 1 2931e38e-3951-47c2-a254-c3a2f30d4674 AzureCloud

PS C:\Users\kubernetes>
```

For the most current version of PowerShell at the time of this delivery
https://github.com/PowerShell/PowerShell/releases



Welcome to the PowerShell 7-x64 Setup Wizard

The Setup Wizard will install PowerShell 7-x64 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

How to do it…

The steps for this experiment are as follows:

1. Execute the following commands to create the container in an Azure storage account:

```
$storageaccountname="adestoragepowershell"
$containername="logfiles"
$resourcegroup="AzureDataEngineering"
#Get the Azure Storage account context
      $storagecontext = (Get-AzStorageAccount -ResourceGroupName
$resourcegroup -Name $storageaccountname).Context;
```

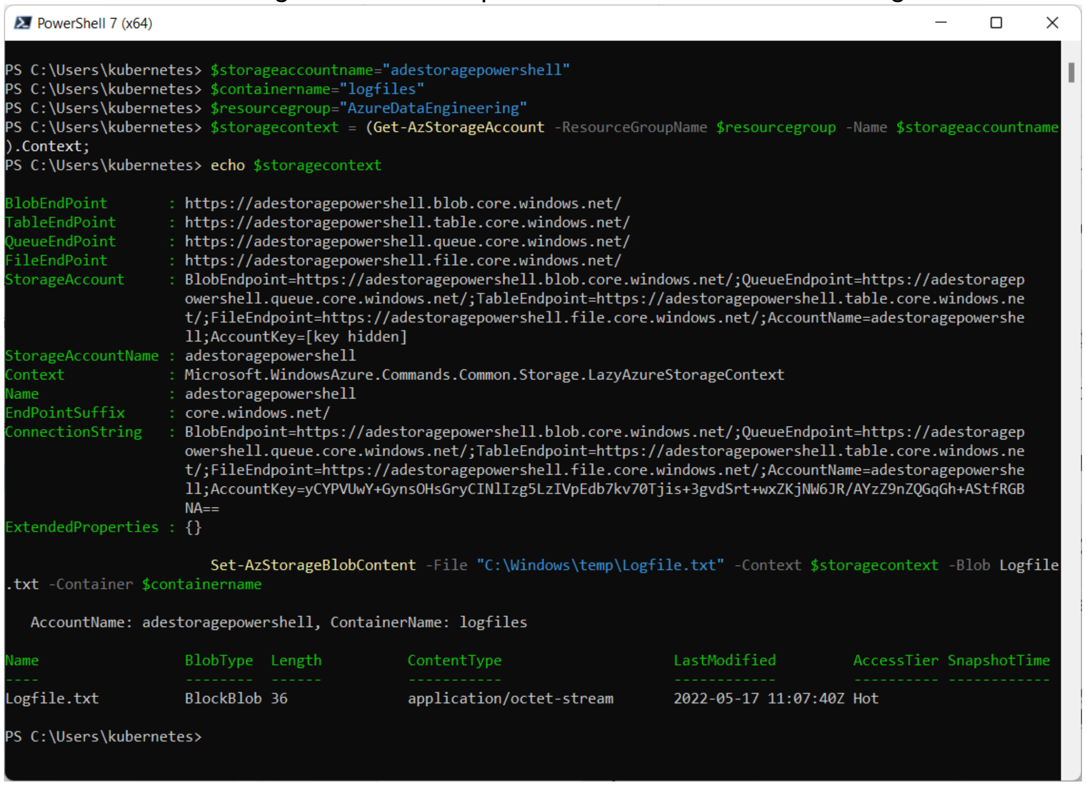
```
#Create a new container
New-AzStorageContainer -Name $containername -Context
$storagecontext
```

Container creation is usually very quick. You should get the following output:

2. Create a sample text file like C:\Windows\temp\Logfile.txt or similar
3. Execute the following commands to upload a text file to an existing container:

```
#upload single file to container
Set-AzStorageBlobContent -File "C:\Windows\temp\Logfile.txt" -Context
$storagecontext -Blob Logfile.txt -Container $containername
```

You should get a similar output to that shown in the following screenshot:



PowerShell 7 (x64)

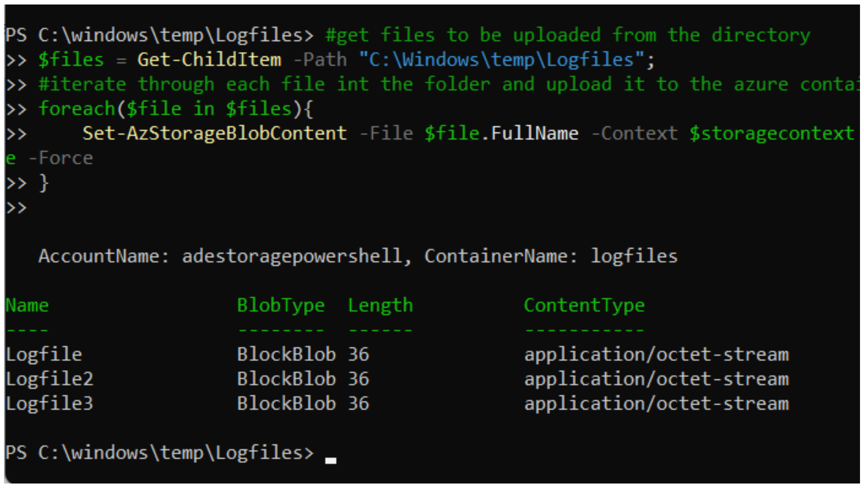
```
PS C:\Users\kubernetes> $storageaccountname="adestoragepowershell"
PS C:\Users\kubernetes> $containername="logfiles"
PS C:\Users\kubernetes> $resourcegroup="AzureDataEngineering"
PS C:\Users\kubernetes> $storagecontext = (Get-AzStorageAccount -ResourceGroupName $resourcegroup -Name $storageaccountname
).Context;
PS C:\Users\kubernetes> echo $storagecontext

BlobEndPoint        : https://adestoragepowershell.blob.core.windows.net/
TableEndPoint       : https://adestoragepowershell.table.core.windows.net/
QueueEndPoint       : https://adestoragepowershell.queue.core.windows.net/
FileEndPoint        : https://adestoragepowershell.file.core.windows.net/
StorageAccount      : BlobEndpoint=https://adestoragepowershell.blob.core.windows.net/;QueueEndpoint=https://adestoragep
                      owershell.queue.core.windows.net/;TableEndpoint=https://adestoragepowershell.table.core.windows.ne
                      t/;FileEndpoint=https://adestoragepowershell.file.core.windows.net/;AccountName=adestoragepowershe
                      ll;AccountKey=[key hidden]
StorageAccountName : adestoragepowershell
Context             : Microsoft.WindowsAzure.Commands.Common.Storage.LazyAzureStorageContext
Name                : adestoragepowershell
EndPointSuffix      : core.windows.net/
ConnectionString    : BlobEndpoint=https://adestoragepowershell.blob.core.windows.net/;QueueEndpoint=https://adestoragep
                      owershell.queue.core.windows.net/;TableEndpoint=https://adestoragepowershell.table.core.windows.ne
                      t/;FileEndpoint=https://adestoragepowershell.file.core.windows.net/;AccountName=adestoragepowershe
                      ll;AccountKey=yCYPVUwY+GynsOHsGryCINlIzg5LzIVpEdb7kv70Tjis+3gvdSrt+wxZKjNW6JR/AYzZ9nZQGqGh+AStfRGB
                      NA==
ExtendedProperties : {}

                        Set-AzStorageBlobContent -File "C:\Windows\temp\Logfile.txt" -Context $storagecontext -Blob Logfile
.txt -Container $containername

   AccountName: adestoragepowershell, ContainerName: logfiles

Name                 BlobType  Length          ContentType                    LastModified         AccessTier SnapshotTime
----                 --------  ------          -----------                    ------------         ---------- ------------
Logfile.txt          BlockBlob 36              application/octet-stream       2022-05-17 11:07:40Z Hot

PS C:\Users\kubernetes>
```

4. Execute the following commands to upload all the files in a directory to an Azure container.

*****__Critically Important Note__: the folder must exist, have one or more files and make sure that no files with your organizational data are in the source folder.

```
#get files to be uploaded from the directory
$files = Get-ChildItem -Path "C:\Windows\temp\Logfiles";

#iterate through each file int the folder and upload it
to the azure container
foreach($file in $files){
    Set-AzStorageBlobContent -File $file.FullName
        -Context $storagecontext -Blob $file.BaseName -Container
$containername -Force
}
```

You should get a similar output to that shown in the following screenshot:



```
PS C:\windows\temp\Logfiles> #get files to be uploaded from the directory
>> $files = Get-ChildItem -Path "C:\Windows\temp\Logfiles";
>> #iterate through each file int the folder and upload it to the azure contai
>> foreach($file in $files){
>>     Set-AzStorageBlobContent -File $file.FullName -Context $storagecontext
e -Force
>> }
>>

   AccountName: adestoragepowershell, ContainerName: logfiles

Name                 BlobType  Length          ContentType
----                 --------  ------          -----------
Logfile              BlockBlob 36              application/octet-stream
Logfile2             BlockBlob 36              application/octet-stream
Logfile3             BlockBlob 36              application/octet-stream

PS C:\windows\temp\Logfiles> _
```

How it works…

The storage container is created using the **New-AzStorageContainer** command. It takes two parameters – **container name** and **storage context**. The storage context can be set using the **Get-AzStorageAccount** command context property.

To upload files to the container, we used the Set-AzStorageBlobContent command. The command requires storage context, a file path to be uploaded, and the container name. To upload multiple files, we can iterate through the folder and upload each file using the **Set-AzStorageBlobContent** command.