

# Microsoft Azure Storage and the Azure CLI

---

## Overview

---

Microsoft Azure Storage is a set of services that allows you to store large volumes of data in a cost-effective manner and in a way that makes the data readily and reliably available to services and applications that consume it. Data committed to Azure Storage can be stored in blobs, tables, queues, or files. [Azure blobs](#) are ideal for storing images, videos, and other other unstructured data, and are frequently used to provide input to and capture output from other Azure services such as [Azure Machine Learning](#) and [Azure Stream Analytics](#). [Azure tables](#) provide NoSQL storage for semi-structured data. [Azure queues](#) support queued message transfers between applications (or parts of applications) and can be used to make applications more scalable and robust by eliminating hard dependencies between them. Finally, [Azure Files](#), which are currently in preview, use the Server Message Block (SMB) 2.1 protocol to share files in the cloud.

Data stored in Microsoft Azure Storage can be accessed over HTTP or HTTPS using straightforward REST APIs, or it can be accessed using rich client libraries available for many popular languages and platforms, including .NET, Java, Android, Node.js, PHP, Ruby, and Python.

In this lab, you'll learn how to work with Azure storage accounts, storage containers, and storage blobs. You'll also get familiar with some of the tools used to manage them, including the [Azure Portal](#) and the [Azure Cross-Platform Command-Line Interface](#), or *Azure CLI*, which works on a variety of operating systems and is arguably the most important tool (other than the Azure Portal) at your disposal for working with Microsoft Azure. The knowledge you gain will be used in later labs featuring Azure services that rely on blob storage for input and output.

## Objectives

In this hands-on lab, you will learn how to:

- Create storage accounts using the Azure Portal
- Create storage containers using the Azure CLI
- Create blobs using the Azure CLI
- Automate common storage tasks by scripting CLI commands
- Delete storage accounts using the Azure Resource Manager

## Prerequisites

The following is required to complete this hands-on lab:

- A Microsoft Azure subscription - [sign up for a free trial](#)
- 

## Exercises

---

This hands-on lab includes the following exercises:

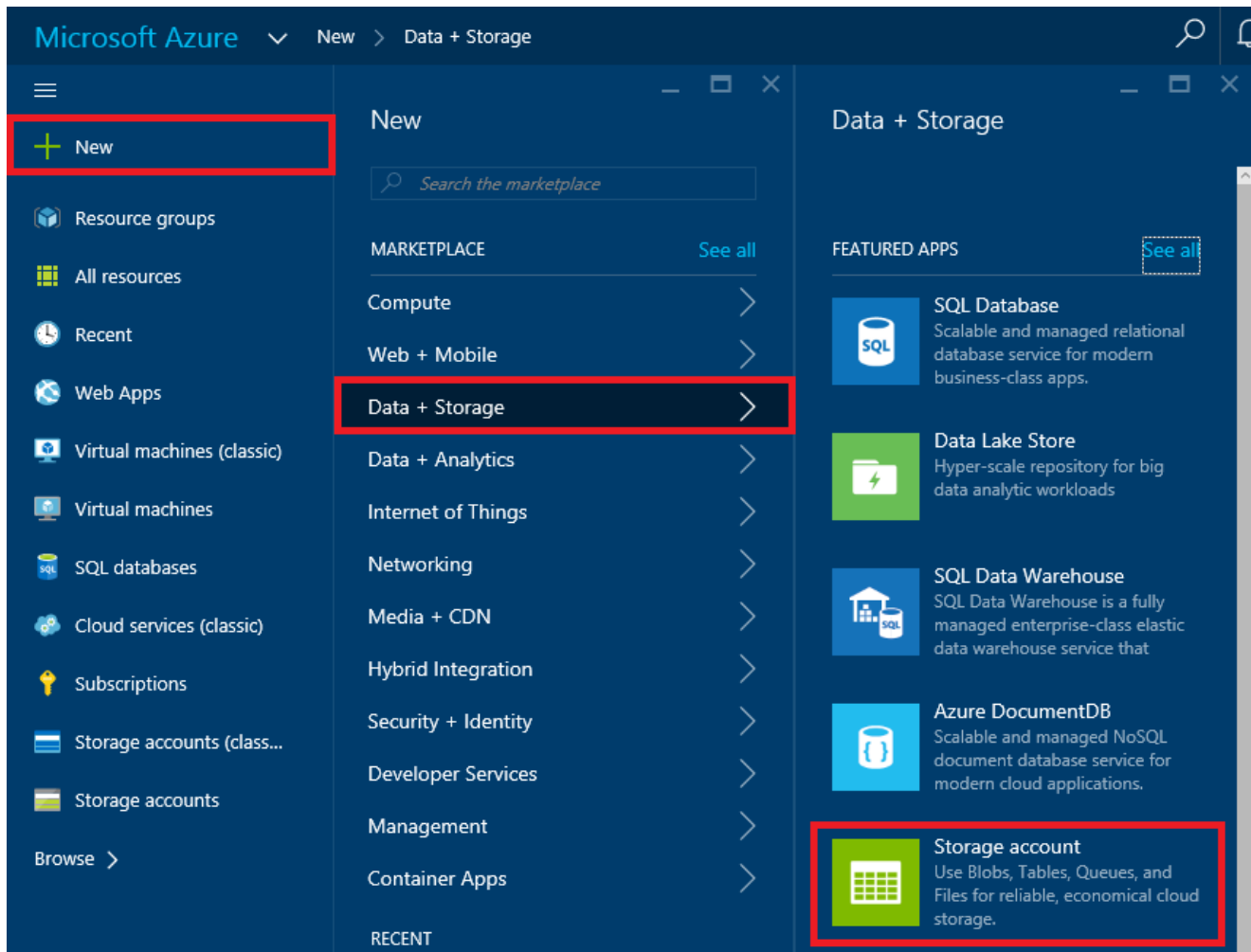
- [Exercise 1: Use the Azure Portal to create a storage account](#)
- [Exercise 2: Install and configure the Azure CLI](#)
- [Exercise 3: Use the Azure CLI to create a container and upload blobs](#)
- [Exercise 4: Automate storage tasks by scripting CLI commands](#)
- [Exercise 5: Delete the resource group](#)

Estimated time to complete this lab: **60** minutes.

### Exercise 1: Use the Azure Portal to create a storage account

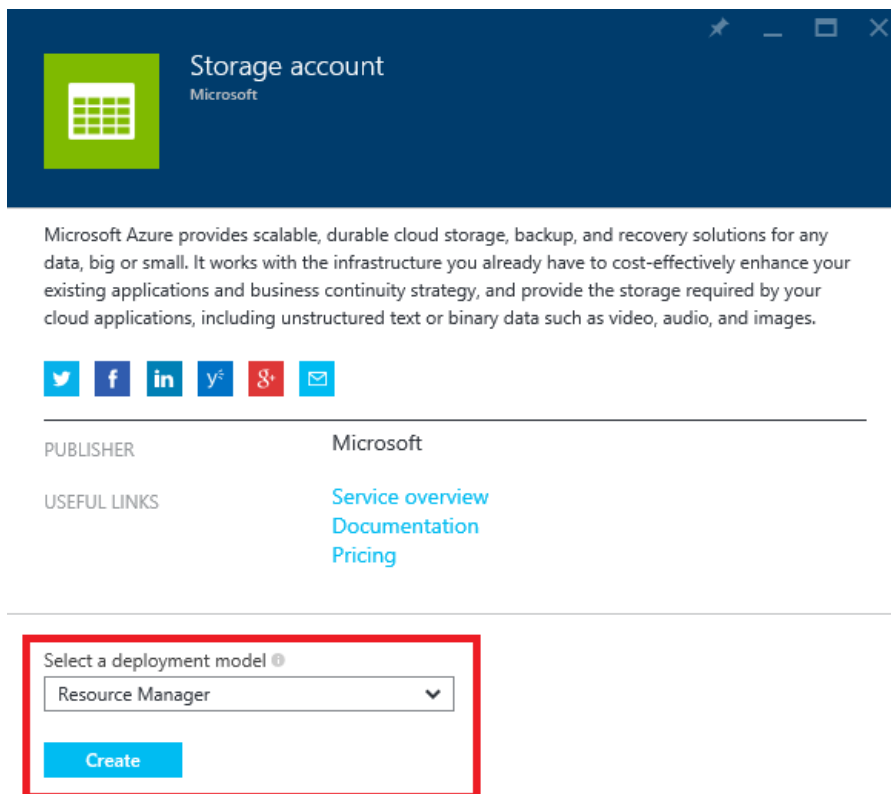
The [Azure Portal](#) allows you to perform basic storage operations such as creating storage accounts and managing the access keys associated with those accounts. In this exercise, you'll use the portal to create a storage account.

1. Go to the [Azure Portal](#) and sign in using the Microsoft credentials associated with your subscription.
2. The first step in using Azure Storage is to create one or more storage accounts. To create a storage account, click **+ NEW** in the ribbon on the left. Then click **Data + Storage**, followed by **Storage account**.



*Creating a storage account*

3. Select **Resource Manager** in the drop-down list under **Select a deployment model**, and then click the **Create** button.



Storage account

Microsoft

Microsoft Azure provides scalable, durable cloud storage, backup, and recovery solutions for any data, big or small. It works with the infrastructure you already have to cost-effectively enhance your existing applications and business continuity strategy, and provide the storage required by your cloud applications, including unstructured text or binary data such as video, audio, and images.

[Twitter](#) [Facebook](#) [LinkedIn](#) [YouTube](#) [Google+](#) [Email](#)

PUBLISHER Microsoft

USEFUL LINKS [Service overview](#)  
[Documentation](#)  
[Pricing](#)

Select a deployment model ⓘ  
Resource Manager ▼  
Create

#### Selecting a deployment model

The default deployment model, **Classic**, creates a "classic" storage account that doesn't fall under the purview of the **Azure Resource Manager**. Specifying **Resource Manager** as the deployment model provides you with more flexibility later on by ensuring that the account is explicitly added to a resource group, and it makes the storage account a first-class citizen in the Azure environment. For more information, see [Understanding Resource Manager deployment and classic deployment](#).

Resource groups are a relatively recent addition to Azure and are a powerful construct for grouping resources such as storage accounts, databases, and virtual machines together so that they can be managed as a group. Imagine that you created a complex application consisting of multiple storage accounts, a cluster of VMs, a SQL database, and perhaps a Stream Analytics solution and a pair of event hubs. Now you want to create a new instance of the application using a different account. By assembling all these resources into a resource group, you can take advantage of [Azure deployment templates](#) to script the creation of the entire application. In addition, you can use role-based security to restrict access to resources in a resource group, and you can delete the application — and all the resources that comprise it — by deleting the resource group. You will take advantage of resource groups and deployment templates in subsequent labs.

4. Enter a name for the new storage account in **Name** field. The name is important, because it forms one part of the URL through which storage items created under this account will be accessed. Storage account names can be 3 to 24 characters in length and can only contain numbers and lowercase letters. In addition, the name you enter must be unique within Azure; if someone else has chosen the same name, you'll be notified that the name isn't available.

Once you have a name that Azure will accept (as indicated by the green check mark in the text field), type "A4R-Labs" (without quotation marks) into the **Resource group** field, and click **Location** and choose the region nearest you. Then click the **Create** button to create the new storage account.

Create storage account

\*

Name

a4rlabs

✓

.core.windows.net

\*

Type

Standard-RAGRS

>

Diagnostics ⓘ

DisabledEnabled

Subscription

\*

Resource Group

A4R-Labs

×

✓

Select existing

Location

East US

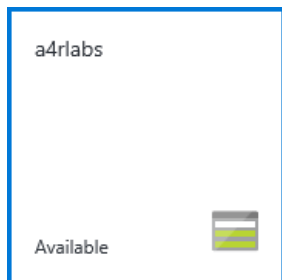
▼

☒ Pin to dashboard

Create

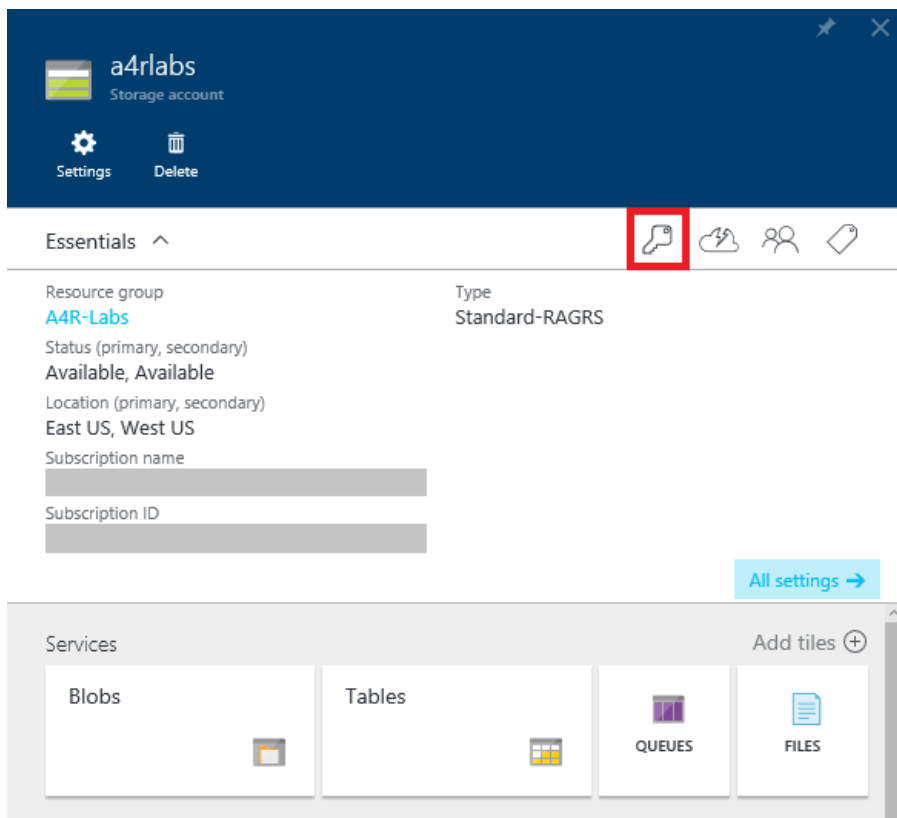
*Specifying parameters for a new storage account*

- After a few moments (it generally takes just a few seconds, but can sometimes take several minutes), a tile representing the new storage account will appear on your dashboard. Click the tile to open a blade for the storage account.



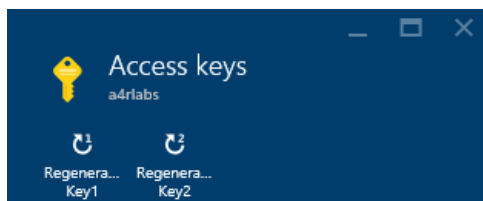
*Tile representing the new storage account*

- In the blade for the storage account, click the key icon to open the "Access keys" blade.



#### Viewing access keys

- This blade shows the access keys associated with the account. You'll use the primary access key a lot when using the Azure CLI, because any CLI command that accesses storage will require an account name and an account key for authentication. Go ahead and save the primary key in a location where you can easily access it later in this lab. You can click the button to the right of the first **KEY1** to copy it to the clipboard, and then paste it into your favorite text editor or wherever else is convenient.



#### STORAGE ACCOUNT NAME

a4rlabs

#### Access keys

##### KEY1

60j+mdptKE0fHgS8q8APTbu0sZxoCnA+f

##### KEY2

WPJv/eUUI6zhX8HLvJHg9n6wNvY1lvYf6i

#### Connection strings

##### KEY1

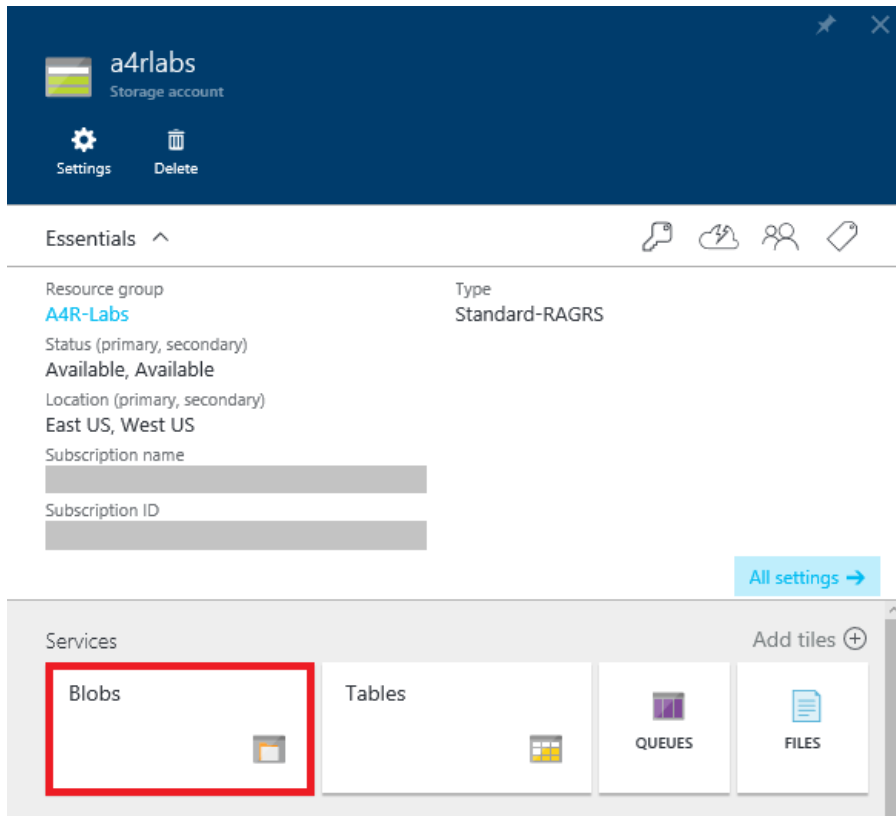
DefaultEndpointsProtocol=https;Account

##### KEY2

DefaultEndpointsProtocol=https;Account

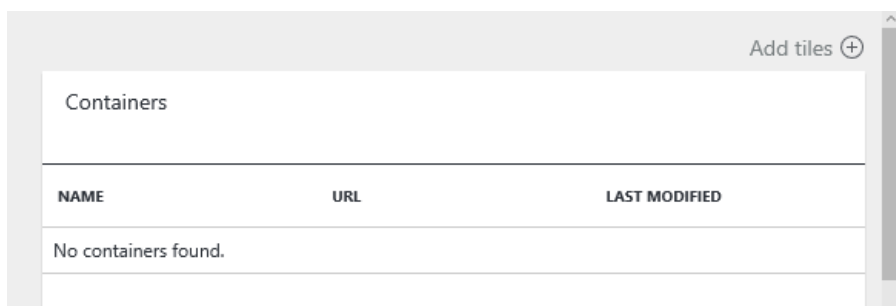
### The storage account's access keys

- Return to the blade for the storage account and click **Blobs** to view a list of containers associated with this account.



### Viewing storage containers

- The storage account currently has no containers. Before you create a blob, you must create a container to store it in. You can create containers in the Azure Preview Portal, but you can't create blobs. In this lab, you will create containers and blobs with the Azure Cross-Platform Command-Line Interface.



### The empty storage account

## Exercise 2: Install and configure the Azure CLI

The [Azure Cross-Platform Command-Line Interface \(CLI\)](#) is an open-source tool that provides a set of cross-platform commands for working with Microsoft Azure. The CLI provides most of the same functionality as the Azure Portal. It also offers features the portal does not, such as the ability to upload blobs to Azure Storage. In this exercise, you will install and configure the Azure CLI.

- The Azure CLI is a Node.js application, and it is installed with the Node.js package manager (npm). To determine whether Node.js and npm are installed on your computer, open a command-prompt or terminal window and execute the following command:

```
npm -v
```

If npm responds with a version number, then **skip to Step 4** to install the CLI. Otherwise, proceed to Step 2.

2. If you are running Windows or OS X, visit the [Node.js Web site](#) and follow the instructions there for downloading and installing Node.js. Then **skip to Step 4** to install the CLI.
3. If you are running Linux, navigate to [Install the Azure CLI](#) and follow the instructions there to install Node.js and npm on Linux.
4. Now that Node.js and npm are installed, run the following command to install the Azure CLI:

```
npm install -g azure-cli
```

As noted in [Install the Azure CLI](#), if you're running Linux, you might have to prefix the **npm** command with the **sudo** command in order for npm to work.

5. Once the Azure CLI is installed, you can use the **azure** command from your operating system's command line to perform Azure-related tasks. To see a list of commands available, and to verify that the Azure CLI is properly installed, execute the following command:

azure

You should see output similar to the following:

```

info:
info:      _ _ _ _ _
info:      / \  |  /  |  |  \  \  |
info:    _ _ /  \  \ /  |  |  /  \  _ _
info:  (  _ /  \  \ /  \  \ /  \  \  )
info:    (  _ _ )      _ _ _ )  _ _
info:      (  _ _ _ _ _ )  (  _ _ _ )
info:
info: Microsoft Azure: Microsoft's Cloud Platform
info:
info: Tool version 0.9.12
help:
help: Display help for a given command
help:   help [options] [command]
help:
help: Log in to an Azure subscription using Active Directory. Currently, the user can login only via Microsoft organization
help:   login [options] [username]
help:
help: Log out from Azure subscription using Active Directory. Currently, the user can log out only via Microsoft organization
help:   logout [options] [username]
help:
help: Open the portal in a browser
help:   portal [options]
help:
help: Commands:
help:   account      Commands to manage your account information and publish settings
help:   config       Commands to manage your local settings
help:   hdinsight    Commands to manage HDInsight clusters and jobs
help:   mobile       Commands to manage your Mobile Services
help:   network      Commands to manage your networks
help:   sb           Commands to manage your Service Bus configuration
help:   service      Commands to manage your Cloud Services
help:   site         Commands to manage your Web Sites
help:   sql          Commands to manage your SQL Server accounts
help:   storage      Commands to manage your Storage objects
help:   vm           Commands to manage your Virtual Machines
help:
help: Options:
help:   -h, --help    output usage information
help:   -v, --version  output the application version

```

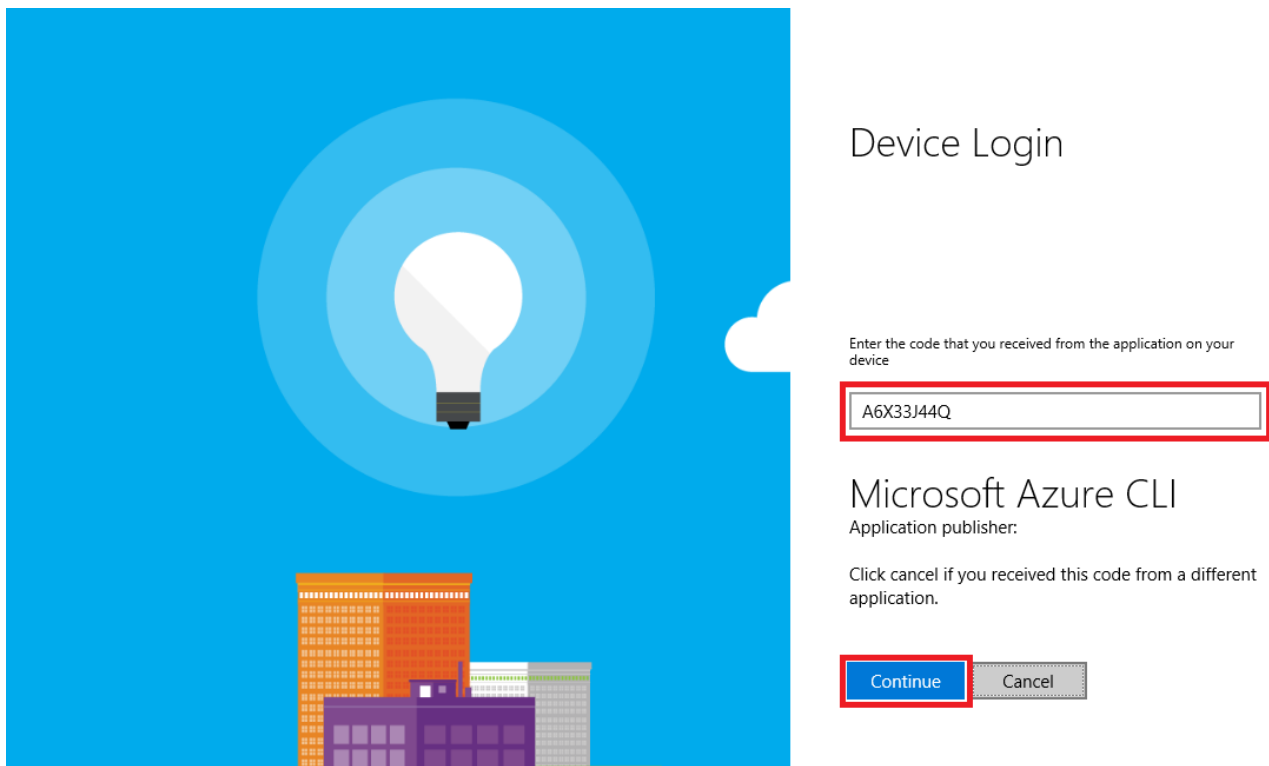
6. Before you can do much with the Azure CLI, you must connect it to an Azure subscription. One way to do that is to log in using your Microsoft account. To begin the login process, execute the following command, substituting your Microsoft account user name for *username*:

```
azure login username
```

The CLI will respond by displaying a message containing an alphanumeric code and a login URL:

```
info: Executing command login
info: To sign in, use a web browser to open the page https://aka.ms/devicelogin. Enter the code A6X33J44Q to authenticate.
```

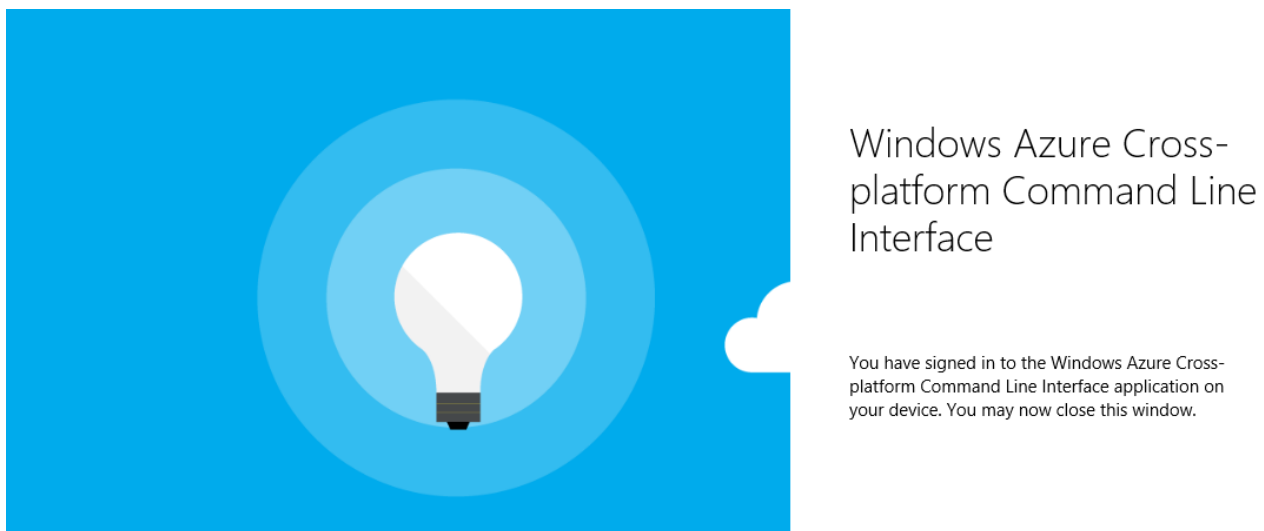
7. Go to <https://aka.ms/devicelogin> and type in the code obtained in the previous step. Then click **Continue**.



#### Entering the access code

If you have multiple Microsoft accounts, you will be asked which one you wish to use to log in. Select the account that you used to set up your Azure subscription for this lab, and if prompted, enter the password for the account.

8. If the login is successful, you will be told that you have signed in and invited to close the browser window. Close your browser and return to the CLI.



#### Successful login

9. The next step is to make sure that if there are multiple subscriptions associated with your account, the one you used in Exercise 1 is set as the default. To view the subscriptions associated with the account you imported, execute the following command:



```
azure account list
```

If only one subscription is listed, move on to the next step. If two or more subscriptions are listed, execute the following command, replacing *subscription* with the name or ID of the subscription you used in Exercise 1:

```
azure account set subscription
```

10. Next, execute the following command to switch the CLI to Azure Resource Manager mode. This is necessary because the storage account you created in Exercise 1 is a Resource-Manager account:

```
azure config mode arm
```

11. Now use the following command to list all of the Resource-Manager storage accounts associated with your subscription and confirm that the account you created in Exercise 1 is present:

```
azure storage account list
```

12. In order to use a storage account from the CLI, you must know the storage account's name, and you must have the storage account's access key. You can get that key from the Azure Portal, (as you did in Exercise 1), or you can get it from the CLI. To display a list of keys associated with the storage account, run the following command, replacing *accountname* with the name of the storage account and *resourcegroup* with the resource group you created in Exercise 1:

```
azure storage account keys list accountname -g resourcegroup
```

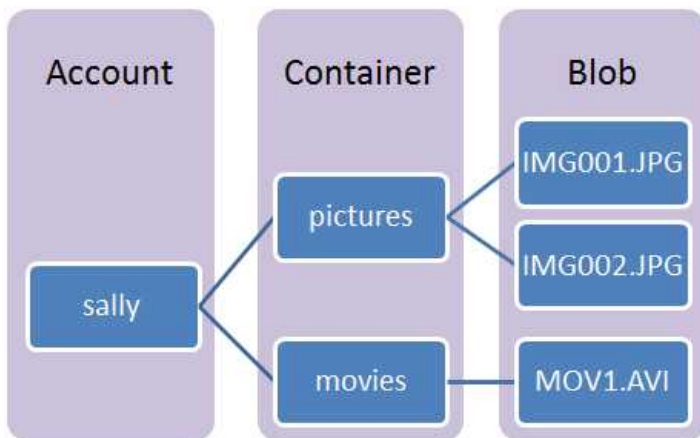
The CLI will respond something like this:

```
info:    Executing command storage account keys list
+ Getting storage account keys
data:    Primary: Ttm20TIsZZ4Mo440EKp8nsX+pB6S3x3AFILxoQ7XgG5y9aZDE+Zaphs11QuMqd/DaS72j8d6S3M5ZtqGKioqOA==
data:    Secondary: V31u0fm23lgdDJKLd6541bmVFTpgJ2xC3k1whQ120+h/TC9wr2+Ck1EqzUsZ00HG563d1m1uc2CSiF/blStNA==
info:    storage account keys list command OK
```

The keys listed here are the same ones you saw in the "Access keys" blade in Exercise 1. If you haven't already made a copy of the primary access key, do it now so you can easily retrieve it later.

### Exercise 3: Use the Azure CLI to create a container and upload blobs

Before you can create a blob, you must create a container to store it in. A container is similar to a folder in a file system. A storage account can have an unlimited number of containers, and a container can store an unlimited number of blobs. Container names must be from 3 to 63 characters in length and may contain numbers, dashes, and lowercase letters. Dashes cannot be consecutive, and a container name cannot start with a dash. The following diagram illustrates the blob storage schema:



Blob storage schema

In this exercise, you will create a container named "images" in the storage account you created in Exercise 1. Then you will upload blobs to it and

learn how to access those blobs in the portal.

1. At a command prompt or terminal window, execute the following command, replacing *accountname* with the name of your storage account and *accountkey* (surrounded by double quote marks) with the account's primary access key:

```
azure storage container create -a accountname -k "accountkey" -p blob images
```

The CLI should respond with output similar to the following:

```
info:    Executing command storage container create
+ Creating storage container images
+ Getting Storage container information
data:    {
data:      name: 'images',
data:      metadata: {},
data:      etag: '"0x8D29F33E5925FA6"',
data:      lastModified: 'Fri, 07 Aug 2015 14:24:25 GMT',
data:      leaseStatus: 'unlocked',
data:      leaseState: 'available',
data:      requestId: '09cd3b60-0001-00cc-571c-d14c48000000',
data:      publicAccessLevel: 'Blob'
data:    }
info:    storage container create command OK
```

Notice the line in the output that reads "publicAccessLevel: 'Blob'". By default, the containers you create are private, which means that the container and its contents can only be accessed by the owner of the storage account (or anyone who has the container's access key). However, the "-p blob" switch you included in the command that created the container allows anonymous read access to blobs in that container. This is generally the way you configure a container that holds images and other public assets for Web sites.

2. The next step is to create a blob by uploading a file to the "images" container. The file you will upload is named *azure-banner.jpg* and is provided for you in the "resources" subdirectory of this lab. At the command prompt, navigate to this lab's "resources" subdirectory.
3. Execute the following command, replacing *accountname* with the storage account's name and *accountkey* (surrounded by double quotes) with the storage account's key to create a blob named "banner.jpg" in the "images" container:

```
azure storage blob upload -a accountname -k "accountkey" azure-banner.jpg images banner.jpg
```

In this command, "azure-banner.jpg" is the name of and path to the file you wish to upload (no path name is required since it's in the current directory), "images" is the container you're uploading to, and "banner.jpg" is the name assigned to the blob.

If the blob is successfully uploaded, you will receive the following confirmation from the CLI:

```
info:    Executing command storage blob upload
+ Checking blob banner.jpg in container images
+ Uploading azure-banner.jpg to blob azure-banner.jpg in container images
Percentage: 100.0% (28.80KB/28.80KB) Average Speed: 28.80KB/S Elapsed Time: 00:00:00
+ Getting Storage blob information
data:    Property      Value
data:    -----
data:    container      images
data:    blob           banner.jpg
data:    blobType       BlockBlob
data:    contentLength   29490
data:    contentType    image/jpeg
data:    contentMD5      +AKWVCtqIG0gsV0oc2fQqw==
info:    storage blob upload command OK
```

4. Because the container's access level is "Blob," you should be able to fetch the blob with a simple HTTP request. To prove it, open your browser and navigate to the following URL, once more replacing *accountname* with the name of your storage account:

```
http://accountname.blob.core.windows.net/images/banner.jpg
```

Here's what you'll see in your browser:



Image blob downloaded from Azure storage

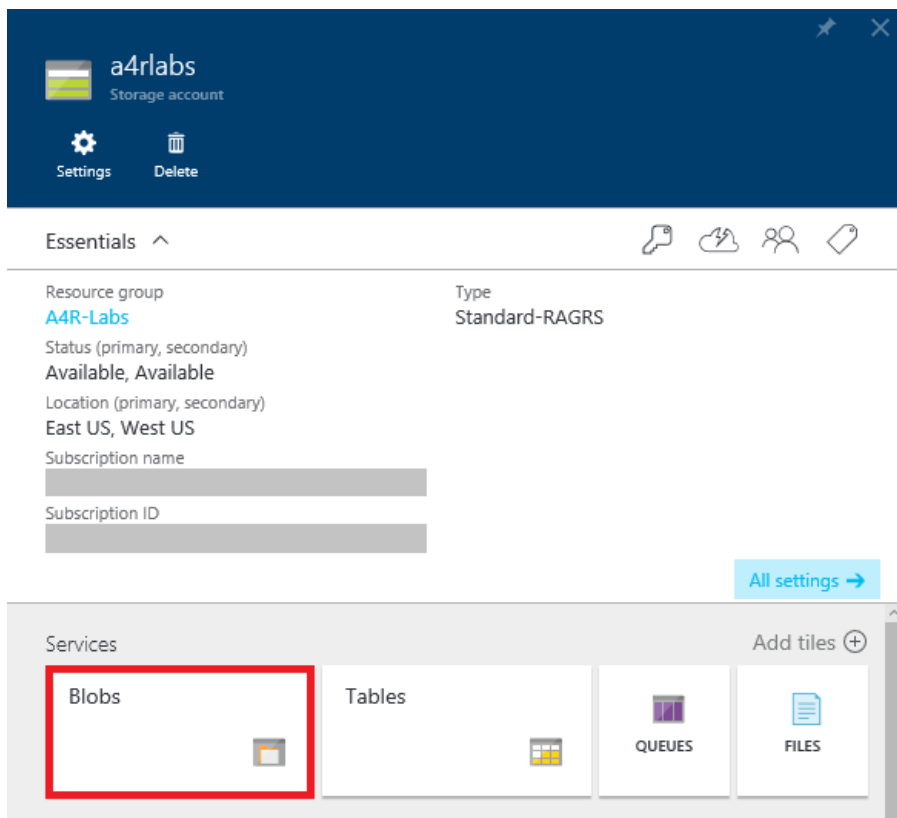
5. You can also see the blob that you uploaded in the Azure Portal. To see it, return to the [Azure Portal](#) in your browser. Click **Storage accounts** in the ribbon on the left. Then click the storage account you created in Exercise 1. (If **Storage accounts** doesn't appear in the ribbon, click **Resource groups** instead and navigate to the storage account through the "A4R-Labs" resource group.)

A screenshot of the Microsoft Azure Portal. The left-hand navigation pane is visible, with "Storage accounts" highlighted in a red box. The main content area shows the "Storage accounts" page. At the top, there are buttons for "Add", "Columns", and "Refresh". Below these is a search bar and a dropdown menu for "All subscriptions". A table lists the storage accounts. The first row, "a4rlabs", is highlighted in a red box. The table has columns for "NAME", "RESOURCE GROUP", "LOCATION", and "SUBSCRIPTION".

NAME	RESOURCE GROUP	LOCATION	SUBSCRIPTION
a4rlabs	A4R-Labs	East US	...

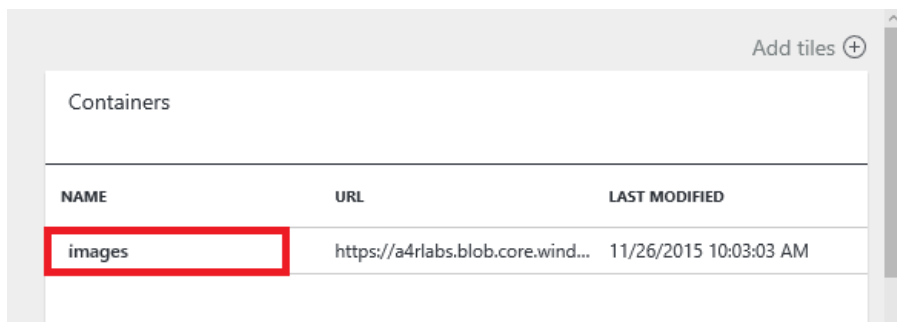
Viewing storage accounts

6. Click **Blobs** to view the containers in this storage account.



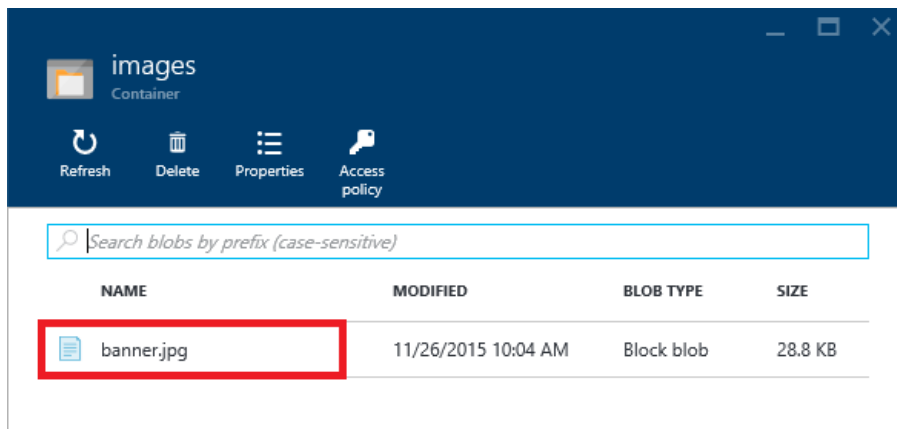
*Viewing storage containers*

- Click the "images" container to view its contents.



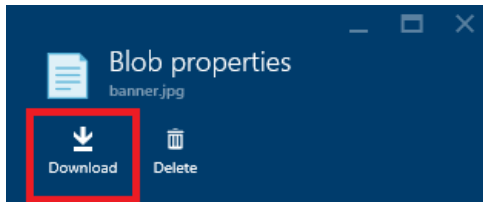
*Viewing the blobs in the "images" container*

- Verify that banner.jpg appears in the list of blobs. Then click it to open the "Blob properties" blade.



*Image blob uploaded to Azure storage*

9. Click the **Download** button at the top of the blade to download and open banner.jpg. Confirm that you see the same image you saw before.



NAME

banner.jpg

URL

<https://a4rlabs.blob.core.windows.net/im.>

LAST MODIFIED

11/26/2015 10:04 AM

TYPE

Block blob

#### Downloading a blob

10. In Step 4, you saw that the storage account name and the container name form parts of the URL through which a blob is accessed. But what if you wanted to create a hierarchy of containers? What if, for example, you wanted "images/banner.jpg" in the URL to be "images/azure/banner.jpg"? You can't create nested containers, but you *can* include forward slashes in blob names to simulate container hierarchies. To demonstrate, run the following command in the CLI, once more providing the storage account's name and key:

```
azure storage blob upload -a accountname -k "accountkey" azure-banner.jpg images azure/banner.jpg
```

11. Now enter the following URL in your browser and confirm that the image blob appears:

```
http://accountname.blob.core.windows.net/images/azure/banner.jpg
```

12. Finish up by executing the following command to delete the blob you just created:

```
azure storage blob delete -a accountname -k "accountkey" images azure/banner.jpg
```

## Exercise 4: Automate storage tasks by scripting CLI commands

One benefit of using the Azure CLI is that you can combine it with a scripting engine to automate time-consuming tasks. For example, what if you wanted to upload a directory full of images from your PC to blob storage? Rather than upload them one at a time with discrete CLI commands, you could write a script that enumerates the files in that directory and invokes an **azure storage blob upload** command on each one.

In this exercise, you'll write and test a pair of scripts that automate common Azure Storage tasks. Linux and OS X users will use Bash scripts, while Windows users will use PowerShell scripts.

There are free Bash shells available for Windows, too, including [Cygwin](#) and [Git for Windows](#). If you use Windows and care to install a Bash shell (or already have one installed), feel free to skip the PowerShell steps below and follow the instructions for Linux and OS X users.

1. If you're using Windows, **skip to Step 5**. The next few steps are for users running Linux, OS X, and other operating systems that support Bash scripts.
2. Go to a terminal window and navigate to the directory containing this lab. In that directory, you'll find a subdirectory named "resources" containing a number of JPG images.
3. In the lab directory — the one containing the "resources" subdirectory — use your favorite editor to create a text file named copyimages.sh containing the following statements. Replace *accountname* with the name of your storage account, and *accountkey* with the storage account's primary access key:

```
cd resources
for f in *.jpg
do
    azure storage blob upload -a accountname -k "accountkey" "${f##*/}" images "${f##*/}"
done
cd ..
```

4. Execute the following command to run copyimages.sh:

```
bash copyimages.sh
```

**Now skip to Step 9.** Steps 5 through 8 are for PowerShell users only.

5. To script Azure commands with PowerShell, you first need to [install and configure Azure PowerShell](#). If Azure PowerShell is not installed on your system, take the time to install it now.
6. Once you have installed Azure PowerShell, you will need to configure it to allow script execution. (For security reasons, PowerShell by default does not allow PowerShell scripts to execute.) Start Azure PowerShell **as an administrator** and execute the following command:

```
Set-ExecutionPolicy RemoteSigned
```


7. Now close the Azure PowerShell window and open another one, this time **not** running as an administrator. At the Azure PowerShell command prompt, navigate to the directory for this lab — the one containing the "resources" subdirectory, which holds a collection of JPG images — and create a text file named copyimages.ps1 containing the statements below. Replace *accountname* and *accountkey* with your storage account's name and primary access key:

```
cd resources
$context = New-AzureStorageContext -StorageAccountName accountname -StorageAccountKey "accountkey"
foreach ($file in Get-ChildItem *.jpg) {
    Set-AzureStorageBlobContent -Blob $file.Name -Container "images" -File $file.Name -Context $context -Force
}
cd ..
```

8. Execute the following command to upload all the images in the "resources" subdirectory as blobs to Azure Storage:

```
.\copyimages.ps1
```

9. Return to the [Azure Portal](#) and open your storage account's "images" container. Verify that all the .jpg files in the "resources" subdirectory were uploaded to the container. (Note that if the blade showing the contents of the "images" container was left open in the portal, you will need to click the **Refresh** button to see the changes.)



NAME	MODIFIED	BLOB TYPE	SIZE
banner.jpg	9/25/2015 8:05 PM	Block blob	28.8 KB
azure-banner.jpg	9/26/2015 8:49 AM	Block blob	28.8 KB
Desktop_surface3_hero2.jpg	9/26/2015 8:49 AM	Block blob	319.32 KB
Desktop_surface3_Hero2fader2.j...	9/26/2015 8:49 AM	Block blob	102.5 KB
Desktop_surface3_Hero2fader1.j...	9/26/2015 8:49 AM	Block blob	670.34 KB
Desktop_home-surface-hub.jpg	9/26/2015 8:49 AM	Block blob	355.1 KB
Desktop_Surface-3-hero.jpg	9/26/2015 8:49 AM	Block blob	152.79 KB
Desktop_Nyx-good-looking.jpg	9/26/2015 8:49 AM	Block blob	486.26 KB
Desktop_HP-carousel-docking-s...	9/26/2015 8:49 AM	Block blob	444.9 KB
Desktop_Creativity-untethered.j...	9/26/2015 8:49 AM	Block blob	260.95 KB
Desktop_Cortana_Pillar3.jpg	9/26/2015 8:49 AM	Block blob	438.17 KB

*Blobs uploaded to the images container*

- Now return to the command prompt and make sure you're in the directory for this lab.
- Next, you're going to create a script that renames a blob. Technically, you can't rename a blob, but you *can* delete a blob and upload a new blob with the new name. If you prefer Bash scripts, create a new text file named `renameblob.sh` containing the following statements. As usual, replace *accountname* and *accountkey* with your storage account's name and key.

```
# syntax: renameblob.sh [container] [old-blob-name] [new-blob-name] [path]
azure storage blob delete -a accountname -k "accountkey" $1 "$2"
azure storage blob upload -a accountname -k "accountkey" "$4" $1 "$3"
```

- If you are a Windows user and prefer PowerShell scripts instead, create a new text file named `renameblob.ps1` containing the following statements. As usual, replace *accountname* and *accountkey* with your storage account's name and primary access key:

```
param([string]$container, [string]$oldname, [string]$newname, [string]$path)
$context = New-AzureStorageContext -StorageAccountName accountname -StorageAccountKey accountkey
Remove-AzureStorageBlob -Container $container -Blob $oldname -Context $context
Set-AzureStorageBlobContent -Blob $newname -Container $container -File $path -Context $context -Force
```

- Now use the script you just created to rename one of the blobs in the "images" container. Use the following command to execute the Bash script:

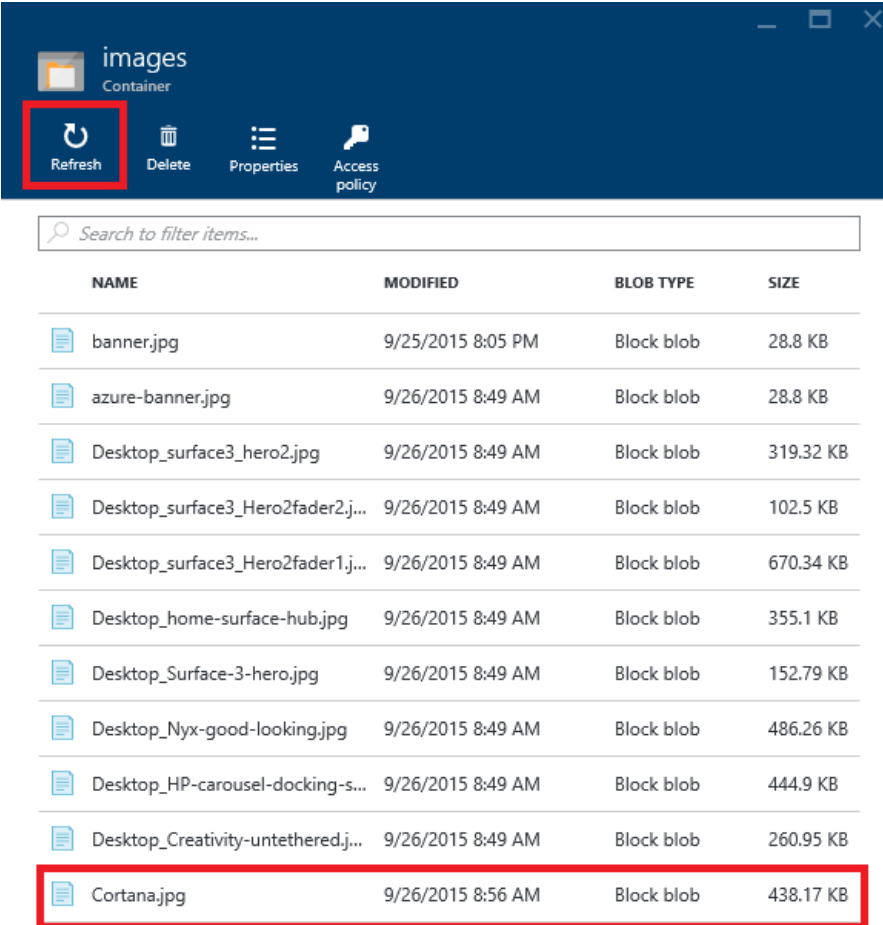
```
bash renameblob.sh images Desktop_Cortana_Pillar3.jpg Cortana.jpg resources/Desktop_Cortana_Pillar3.jpg
```

The equivalent Azure PowerShell command is:

```
.\renameblob.ps1 images Desktop_Cortana_Pillar3.jpg Cortana.jpg resources/Desktop_Cortana_Pillar3.jpg
```

- Return to the [Azure Portal](#), open the "images" container, and confirm that the blob named `Desktop_Cortana_Pillar3.jpg` was renamed to `Cortana.jpg`. (If the blade showing the contents of the "images" container was left open, you will need to click **Refresh** button to see the

change.)



NAME	MODIFIED	BLOB TYPE	SIZE
banner.jpg	9/25/2015 8:05 PM	Block blob	28.8 KB
azure-banner.jpg	9/26/2015 8:49 AM	Block blob	28.8 KB
Desktop_surface3_hero2.jpg	9/26/2015 8:49 AM	Block blob	319.32 KB
Desktop_surface3_Hero2fader2.j...	9/26/2015 8:49 AM	Block blob	102.5 KB
Desktop_surface3_Hero2fader1.j...	9/26/2015 8:49 AM	Block blob	670.34 KB
Desktop_home-surface-hub.jpg	9/26/2015 8:49 AM	Block blob	355.1 KB
Desktop_Surface-3-hero.jpg	9/26/2015 8:49 AM	Block blob	152.79 KB
Desktop_Nyx-good-looking.jpg	9/26/2015 8:49 AM	Block blob	486.26 KB
Desktop_HP-carousel-docking-s...	9/26/2015 8:49 AM	Block blob	444.9 KB
Desktop_Creativity-untethered.j...	9/26/2015 8:49 AM	Block blob	260.95 KB
Cortana.jpg	9/26/2015 8:56 AM	Block blob	438.17 KB

#### The renamed blob

There's much more you can do when scripting the Azure CLI than these simple examples demonstrate. For example, you can pipe the output from **azure** commands to other commands such as **grep** and **awk**, and you can use the **-v** (or **--verbose**) switch to output JSON data. For more information and some cool examples, see [How to script the Azure CLI for Mac, Linux, and Windows](#) on the Azure Web site.

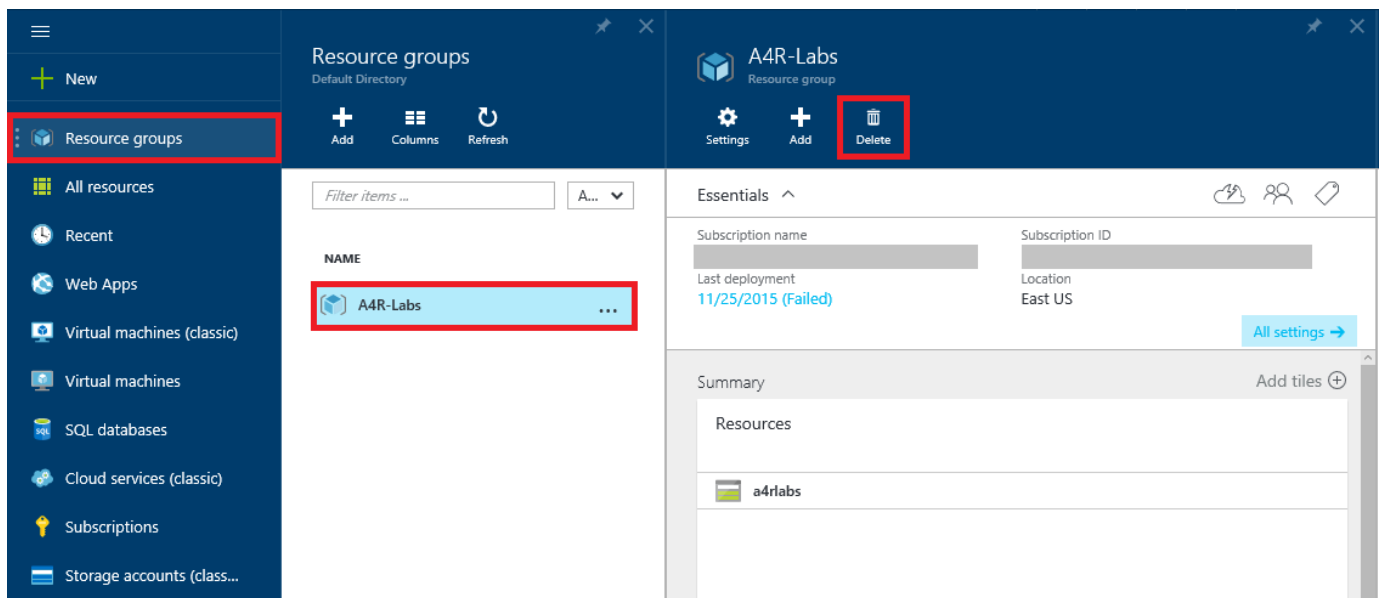
## Exercise 5: Delete the resource group

When you created a storage account in Exercise 1, you made it part of a resource group named "A4R-Labs." One of the benefits of using resource groups is that deleting a resource group deletes all the resources inside it, including storage accounts. Deleting a resource group is a convenient way to delete complex Azure deployments without having to delete individual resources one by one.

In this exercise, you'll use the Azure Portal to delete the storage account you created in Exercise 1, and along with it the containers and blobs you created in Exercises 3 and 4.

1. Open the [Azure Portal](#) in your browser and click **Resource groups** in the ribbon on the left. Then, in the "Resource groups" blade, click the resource group you wish to delete ("A4R-Labs"). Finally, click **Delete** in the blade for the resource group.





#### Deleting a resource group

- Because deleting a resource group is a permanent action that can't be undone, you must confirm that you want to delete it. Do so by typing the name of the resource group into the box labeled **TYPE THE RESOURCE GROUP NAME**. Then click **Delete** to delete the resource group and everything inside it.

Warning! Deleting the "A4R-Labs" resource group is irreversible. The action you're about to take can't be undone. Going further will delete this resource group and all the resources in it permanently.

TYPE THE RESOURCE GROUP NAME:

A4R-Labs
✓

Affected resources

	NAME	TYPE
	a4rlabs	Storage accounts

Delete

Cancel

#### Confirming resource-group deletion

- After a few minutes, you will be notified that the resource group was deleted. If the deleted resource group still appears in the "Resource groups" blade, click that blade's **Refresh** button to update the list of resource groups. The deleted resource group should go away.

## Summary

Here's a quick summary of the important concepts that you learned in this lab:

- Azure Storage is a set of services for storing data durably and reliably
- The Azure Portal enables you to perform basic storage operations, such as creating storage accounts and viewing blobs and containers

- Azure Storage blobs can contain any type of data, just like files in a file system
- The Azure Cross-Platform Command-Line Interface (CLI) is a cross-platform tool that supports many features the Azure Portal does not, such as the ability to upload blobs
- The Azure CLI can be combined with scripting languages to simplify storage tasks that require multiple commands
- Storage accounts and other resources that are placed inside a resource group can easily be deleted by deleting the resource group itself

Now that you're familiar with storage accounts, containers, and blobs, as well as some of the tools for managing them, you'll put your knowledge to work in subsequent labs. Knowing the basics of Azure Storage is an essential first step in working with Azure data services.

---

Copyright 2015 Microsoft Corporation. All rights reserved. Except where otherwise noted, these materials are licensed under the terms of the Apache License, Version 2.0. You may use it according to the license as is most appropriate for your project on a case-by-case basis. The terms of this license can be found in <http://www.apache.org/licenses/LICENSE-2.0>.