

Azure Queue Storage is a service for storing large numbers of messages. These messages can access from anywhere via authenticated calls using HTTP or HTTPS. A queue message can be up to 64 KB in size. Queues are commonly used to create a backlog of work to process asynchronously.

The Queue service contains below components

Storage account: All access to Azure Storage is done through a storage account.

Queue: A queue contains a set of messages.

Message: A message, in any format, of up to 64 KB. The maximum time-to-live can be any positive number, or -1 indicating that the message doesn't expire. If this parameter is omitted, the default time-to-live is seven days.

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

[Home](#) > [Storage accounts](#) >

## Create storage account

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	Free Trial
Resource group *	(New) test-queue-store-rg

[Create new](#)

### Instance details

The default deployment model is Resource Manager, which supports the latest Azure features. You may choose to deploy using the classic deployment model instead. [Choose classic deployment model](#)

Storage account name * ⓘ	testqueuestore ✓
Location *	(US) East US
Performance ⓘ	<input checked="" type="radio"/> Standard <input type="radio"/> Premium
Account kind ⓘ	StorageV2 (general purpose v2)
Replication ⓘ	Read-access geo-redundant storage (RA-GRS)
Access tier (default) ⓘ	<input type="radio"/> Cool <input checked="" type="radio"/> Hot

[Review + create](#)

[< Previous](#)

[Next : Networking >](#)

## Create storage account

Basics **Networking** Data protection Advanced Tags Review + create

### Network connectivity

You can connect to your storage account either publicly, via public IP addresses or service endpoints, or privately, using a private endpoint.

Connectivity method \*

- Public endpoint (all networks)
- Public endpoint (selected networks)
- Private endpoint

**i** All networks will be able to access this storage account.  
[Learn more about connectivity methods](#)

### Network routing

Determine how to route your traffic as it travels from the source to its Azure endpoint. Microsoft network routing is recommended for most customers.

Routing preference \* ⓘ

- Microsoft network routing (default)
- Internet routing

[Review + create](#)

[< Previous](#)

[Next : Data protection >](#)

## Create storage account

Basics Networking **Data protection** Advanced Tags Review + create

Blob soft delete ⓘ

- Disabled
- Enabled

File share soft delete ⓘ

- Disabled
- Enabled

Versioning ⓘ

- Disabled
- Enabled

**i** The current combination of subscription, storage account kind, performance, replication and location does not support versioning.

# Create storage account

Basics   Networking   Data protection   Advanced   Tags   Review + create

## Security

Secure transfer required ⓘ    Disabled    Enabled

Blob public access ⓘ    Disabled    Enabled

Minimum TLS version ⓘ    ▾

## Azure Files

Large file shares ⓘ    Disabled    Enabled

**i** The current combination of storage account kind, performance, replication and location does not support large file shares.

## Data Lake Storage Gen2

Hierarchical namespace ⓘ    Disabled    Enabled

NFS v3 ⓘ    Disabled    Enabled

**i** Sign up is currently required to utilize the NFS v3 feature on a per-subscription basis. [Sign up for NFS v3](#)

**Review + create**   [< Previous](#)   [Next : Tags >](#)

**testqueuestore**  
Storage account

Search (Ctrl+/)   Open in Explorer → Move Refresh Delete Feedback

**Queue service**

- Queues

**Monitoring**

- Insights
- Alerts
- Metrics
- Workbooks
- Advisor recommendations
- Monitoring (classic)
- Alerts (classic)

**Containers**  
Scalable, cost-effective storage for unstructured data  
[Learn more](#)

**File shares**  
Serverless SMB file shares  
[Learn more](#)

**Tables**  
Tabular data storage  
[Learn more](#)

**Queues**  
Effectively scale apps according to traffic  
[Learn more](#)

**Tools and SDKs**

- [Storage Explorer \(preview\)](#)
- [PowerShell](#)
- [Azure CLI](#)
- [.NET](#)
- [Java](#)
- [Python](#)
- [Node.js](#)

**Monitoring**

Show data for last:  1 hour    6 hours    12 hours    1 day    7 days

Show data for:

**testqueuestore** | Queues  
Storage account

Search (Ctrl+/)   [+ Queue](#)   Refresh   Delete

**Queue service**

- Queues

**Monitoring**

- Insights

**Authentication method:** Access key ([Switch to Azure AD User Account](#))

Search queues by prefix

Queue	Url
You don't have any queues yet.	

Search (Ctrl+/)

« + Queue Refresh | Delete

Queue service

Queues

Monitoring

Insights

Alerts

### Add queue

Queue name \*

test-queue 

OK Cancel

### test-queue

Queue

Search (Ctrl+/)

Refresh + Add message Dequeue message Clear queue

Overview

Access Control (IAM)

Settings

Access policy

Metadata

Authentication method: Access key [\(Switch to Azure AD User Account\)](#)

Search to filter items...

Id Message text

No results

### test-queue

Queue

Search (Ctrl+/)

Refresh + Add message Dequeue message Clear queue

Overview

Access Control (IAM)

Settings

Access policy

Metadata


### Add message to queue

Message text \*

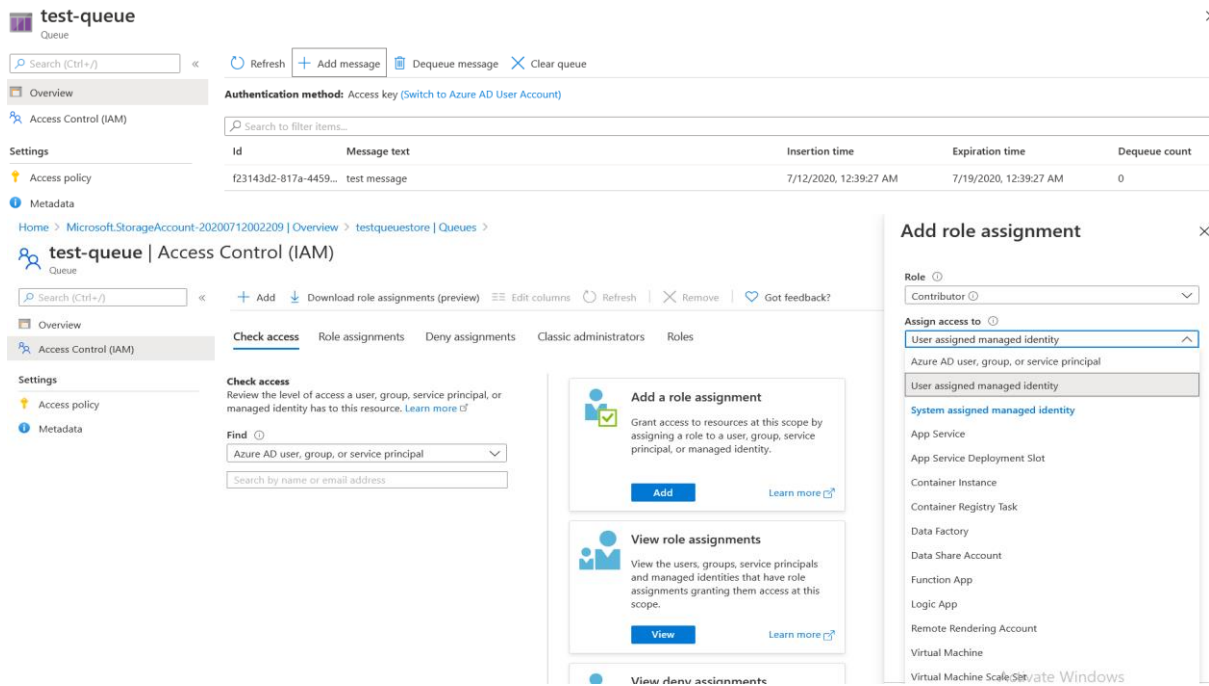
test message 

Expires in: \*

7 Days 

Encode the message body in Base64 

OK Cancel



## Queue storage operations with Azure PowerShell

sign to portal and Retrieve list of locations

Connect-AzAccount

Get-AzLocation | Select-Object Location

\$location = "eastus"

Create resource group

\$resourceGroup = "howtoqueuesrg"

New-AzResourceGroup -ResourceGroupName \$resourceGroup -Location \$location

Create storage account

\$storageAccountName = "howtoqueuestorage"

\$storageAccount = New-AzStorageAccount -ResourceGroupName \$resourceGroup `

-Name \$storageAccountName `

-Location \$location `

-SkuName Standard\_LRS

\$ctx = \$storageAccount.Context

Create a queue

\$queueName = "howtoqueue"

\$queue = New-AzStorageQueue -Name \$queueName -Context \$ctx

Retrieve a queue

# Retrieve a specific queue

\$queue = Get-AzStorageQueue -Name \$queueName -Context \$ctx

# Show the properties of the queue

\$queue

# Retrieve all queues and show their names

Get-AzStorageQueue -Context \$ctx | Select-Object Name

Add a message to a queue

# Create a new message using a constructor of the CloudQueueMessage class

```
$queueMessage = [Microsoft.Azure.Storage.Queue.CloudQueueMessage]::new("This is message 1")
```

# Add a new message to the queue

```
$queue.CloudQueue.AddMessageAsync($QueueMessage)
```

# Add two more messages to the queue

```
$queueMessage = [Microsoft.Azure.Storage.Queue.CloudQueueMessage]::new("This is message 2")
```

```
$queue.CloudQueue.AddMessageAsync($QueueMessage)
```

```
$queueMessage = [Microsoft.Azure.Storage.Queue.CloudQueueMessage]::new("This is message 3")
```

```
$queue.CloudQueue.AddMessageAsync($QueueMessage)
```

Read a message from the queue, then delete it

# Set the amount of time you want to entry to be invisible after read from the queue

# If it is not deleted by the end of this time, it will show up in the queue again

```
$invisibleTimeout = [System.TimeSpan]::FromSeconds(10)
```

# Read the message from the queue, then show the contents of the message. Read the other two messages, too.

```
$queueMessage = $queue.CloudQueue.GetMessageAsync($invisibleTimeout,$null,$null)
```

```
$queueMessage.Result
```

```
$queueMessage = $queue.CloudQueue.GetMessageAsync($invisibleTimeout,$null,$null)
```

```
$queueMessage.Result
```

```
$queueMessage = $queue.CloudQueue.GetMessageAsync($invisibleTimeout,$null,$null)
```

```
$queueMessage.Result
```

# After 10 seconds, these messages reappear on the queue.

# Read them again, but delete each one after reading it.

# Delete the message.

```
$queueMessage = $queue.CloudQueue.GetMessageAsync($invisibleTimeout,$null,$null)
```

```
$queueMessage.Result
```

```
$queue.CloudQueue.DeleteMessageAsync($queueMessage.Result.Id,$queueMessage.Result.popReceipt)
```

```
$queueMessage = $queue.CloudQueue.GetMessageAsync($invisibleTimeout,$null,$null)
```

```
$queueMessage.Result
```

```
$queue.CloudQueue.DeleteMessageAsync($queueMessage.Result.Id,$queueMessage.Result.popReceipt)
```

```
$queueMessage = $queue.CloudQueue.GetMessageAsync($invisibleTimeout,$null,$null)
```

```
$queueMessage.Result
```

```
$queue.CloudQueue.DeleteMessageAsync($queueMessage.Result.Id,$queueMessage.Result.popReceipt)
```

Delete a queue

# Delete the queue

```
Remove-AzStorageQueue -Name $queueName -Context $ctx
```

Clean up resources

```
Remove-AzResourceGroup -Name $resourceGroup
```