Azure Storage Actions documentation

Azure Storage Actions is a serverless framework that you can use to perform common data operations on millions of objects across multiple storage accounts. The current release of Azure Storage Actions enables you to create storage tasks that can perform operations on blobs in Azure Storage accounts based on a set of conditions that you define.

About Storage Actions?

OVERVIEW

What is Azure Storage Actions?

Get started

QUICKSTART

Quickstart: Create, assign, and run a storage task

HOW-TO GUIDE

Create a storage task

CONCEPT

Storage task scenarios

Best practices for storage tasks

Known issues and limitations with storage tasks

Costs and billing

Storage task composition

HOW-TO GUIDE

Define storage task conditions and operations

CONCEPT

Conditions

Operations

Storage task assignments

HOW-TO GUIDE

Create and manage a storage task assignment

CONCEPT

Storage task assignment

Storage task assignment roles

Monitor task runs

CONCEPT

Storage task runs

Monitor Azure Storage Actions

What is Azure Storage Actions

08/01/2025

Azure Storage Actions is a fully managed platform designed to automate data management tasks for Azure Blob Storage and Azure Data Lake Storage. You can use it to perform common data operations on millions of objects across multiple storage accounts without provisioning extra compute capacity and without requiring you to write code.

You can use Azure Storage Actions to automate tasks such as moving data to more cost-effective tiers, manage the retention of versions, snapshots or sensitive data sets, rehydrating data from archive storage so that it is available for immediate use, or manage blob index tags and metadata for better organization and data retrieval.

> **Important**
>
> Azure Storage Actions is generally available in these regions.

Terms and definitions

The resource that you provision to perform data operations is called a *storage task*. A *storage task* can perform operations on blobs in Azure Storage accounts based on a set of conditions that you define.

A storage task contains a set of *conditions*, *operations*. To execute a storage task, you must create and *assignment*. The following table describes each term.

Expand table

Component	Description
Conditions	A *condition* a collection of one or more *clauses*. Each clause contains a property, a value, and an operator. When the storage task runs, it uses the operator to compare a property with a value to determine whether a clause is met by the target object. For example, a clause might evaluate whether a `creation-time` property of a blob is greater than five days ago.
Operations	An operation is the action a storage task performs on each object that meets the defined set of conditions. Deleting a blob is an example of an operation.
Assignments	An assignment identifies a storage account and a subset of objects to target in that account. It also specifies when the task runs and where execution reports are stored.

Composition

Start by creating a storage task. To provision a storage task, you must define at least one condition and one operation. The easiest way to compose conditions is by using a visual designer in the Azure portal. You can use a built-in preview capability in that designer to see the impact of your conditions against test data. See Define storage task conditions and operations.

> **Note**
>
> You can also create storage task definitions by using REST, SDKs, PowerShell, Azure CLI, Bicep, Terraform, or ARM templates.

See these articles to learn how to define a storage task:

- Create a storage task
- Define storage task conditions and operations
- Storage task conditions
- Storage task operations

Execution

To use a storage task, you must create a storage task assignment. An assignment identifies a storage account and a subset of objects to target in that account. It also specifies when the task runs and where execution reports are stored. See Storage task assignment.

Tasks run asynchronously according to the schedule that you specify in the assignment. An execution report is created when the run completes. That report itemizes the results of the task run on each object that was targeted by the task. See Analyze storage task runs.

The overview page of the task presents metrics and visualizations that summarize how many objects met the task condition, and the result of the operations attempted by the storage task on each object. The charts enable you to quickly drill into a specific execution instance. See Monitor Azure Storage Actions.

See these articles to learn how to assign a storage task:

- Create and manage a storage task assignment
- Azure roles for storage task assignments

> **Note**

Storage task assignment can't target general-purpose v1 and legacy Blob Storage accounts because those accounts don't support the latest features. If you have a general-purpose v1 or legacy Blob Storage account, we recommend you to upgrade to **general-purpose v2 accounts** to use all the latest features.

Events

Azure Storage Actions events allow applications to react to events, such as the completion of a storage task run. It does so without the need for complicated code or expensive and inefficient polling services.

Azure Storage Actions events are pushed using Azure Event Grid to subscribers such as Azure Functions, Azure Logic Apps, or even to your own http listener. Event Grid provides reliable event delivery to your applications through rich retry policies and dead-lettering. Event Grid uses event subscriptions to route event messages to subscribers. First, subscribe an endpoint to an event. Then, when an event is triggered, the Event Grid service sends data about that event to the endpoint.

See the Azure Storage Actions events schema article to view the full list of the events that Azure Storage Actions supports.

Pricing and billing

Pricing is based on the execution of storage task assignments. Each time your storage task assignment executes, you're billed a task execution instance charge. You also incur a charge based on the count of objects scanned and evaluated against the conditions of the storage task. That charge is based on a single price per million objects scanned. The final meter applies to the count of operations performed on objects in the storage account. This charge is also based on a single price per million objects. Meters are applied to each executing instance. If a storage task assignment is scheduled to execute repeatedly, then you're billed for each separate instance.

At the end of your billing cycle, the charges for each meter are summed. Your bill or invoice shows a section for all Azure Storage Actions costs. There's a separate line item for each meter. These charges appear in the subscription of the storage account where the task assignment is configured. To learn more about Azure Storage Actions billing meters along with example calculations for common scenarios, see Plan to manage costs for Azure Storage Actions

Supported Regions

Azure Storage Actions is generally available in the following public cloud regions:

- Australia Central
- Australia East
- Australia Southeast
- Brazil South
- Brazil Southeast
- Canada Central
- Central India
- Central US
- East US 2
- France Central
- Germany North
- Germany West Central
- Israel Central
- Italy North
- Japan East
- Japan West
- Jio India Central
- Korea Central
- Korea South
- North Central US
- North Europe
- Norway East
- Norway West
- South Africa North

- South Africa West
- South Central US
- South India
- Spain Central
- Sweden Central
- Sweden South
- Switzerland North
- Switzerland West
- UAE Central
- UAE North
- UK South
- UK West
- West Central US
- West Europe
- West US
- West US 2
- West US 3

Next steps

- Quickstart: Create, assign, and run a storage task by using the Azure portal
- Known issues with storage tasks

Quickstart: Create, assign, and run a storage task

Article • 05/07/2025

In this quickstart, you learn how to use the Azure portal to create a storage task and assign it to an Azure Storage account. Then, you'll review the results of the run. The storage task applies a time-based immutability policy any Microsoft Word documents that exist in the storage account.

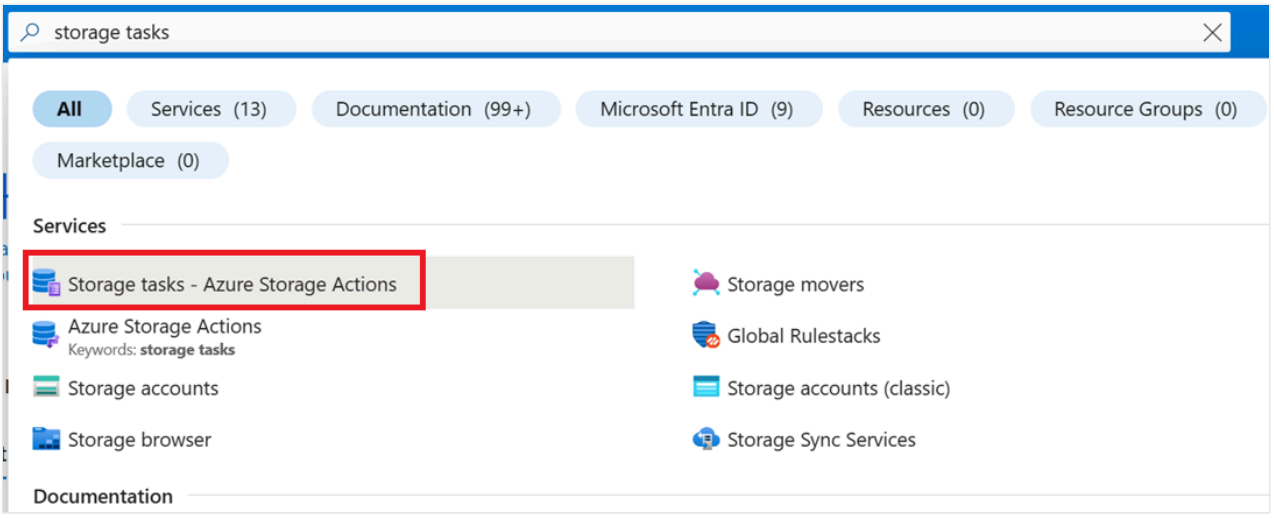
Prerequisites

- An Azure subscription. See create an account for free.
- An Azure storage account. See create a storage account. As you create the account, make sure to enable version-level immutability support and that you don't enable the hierarchical namespace feature.

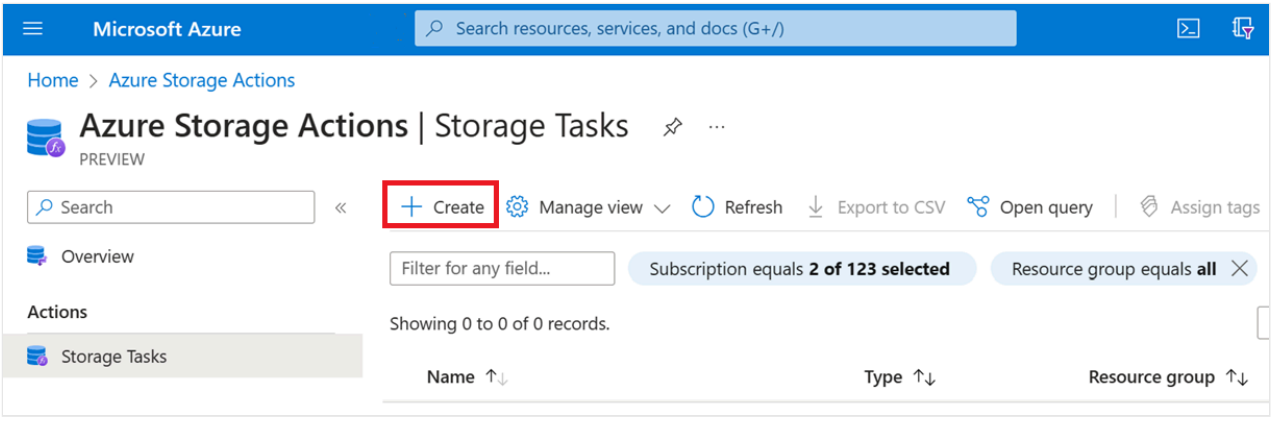
  During the public, you can target only storage accounts that are in the same region as the storage tasks.
- The Storage Blob Data Owner role is assigned to your user identity in the context of the storage account or resource group.
- A custom role assigned to your user identity in the context of the resource group which contains the RBAC actions necessary to assign a task to a storage account. See Permissions required to assign a task.
- A blob container with one or more Microsoft Word documents stored in that container.

Create a task

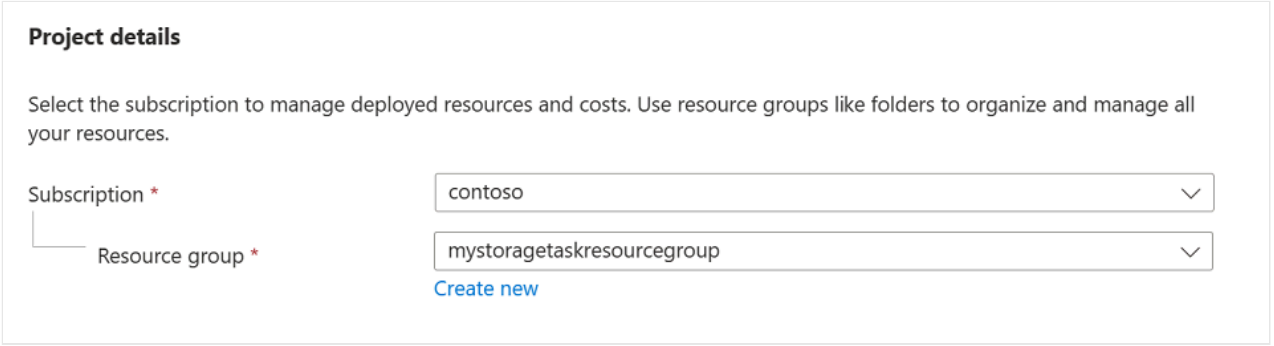
1. In the Azure portal, search for *Storage tasks*.
2. Under **Services**, select **Storage tasks - Azure Storage Actions**.



3. On the **Azure Storage Actions | Storage Tasks** page, select **Create**.



4. In the **Basics** page, under **Project details**, make sure that the correct subscription is selected. Then, select the same resource group that contains your new storage account.



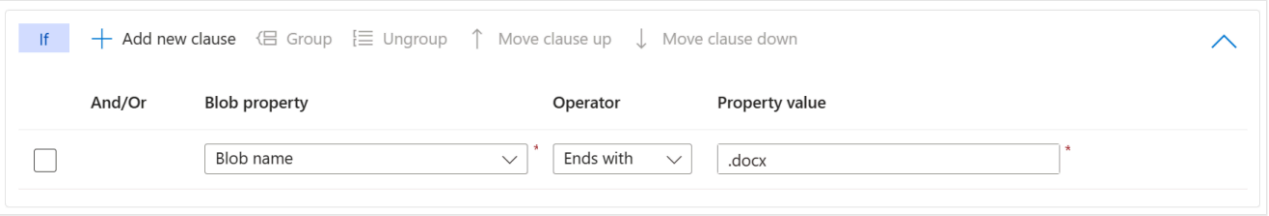
5. Under **Instance details**, enter *mystoragetask* for the **Storage task name**, and select any region that is supported by this service.

6. Select **Next** to open the **Conditions** page.

Add clauses to a condition

You can specify the conditions of a storage task by making selections in **If** section of the **Visual Builder** tab. Every storage task has at least one condition with one clause in that condition.

1. In the **Select a property** drop-down list of the **If** section, select **Blob name**.
2. For the **Operator** of that condition, select **Ends with**, and in the **Enter a string** box, enter *.docx*.

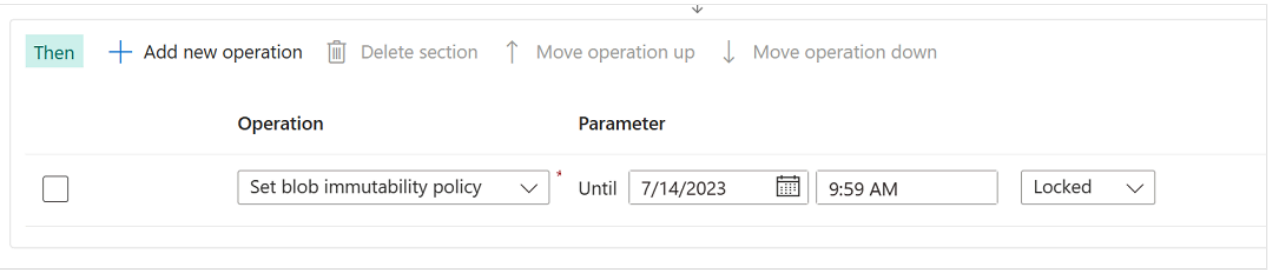


This condition allows operations only on Word documents.

Add operations

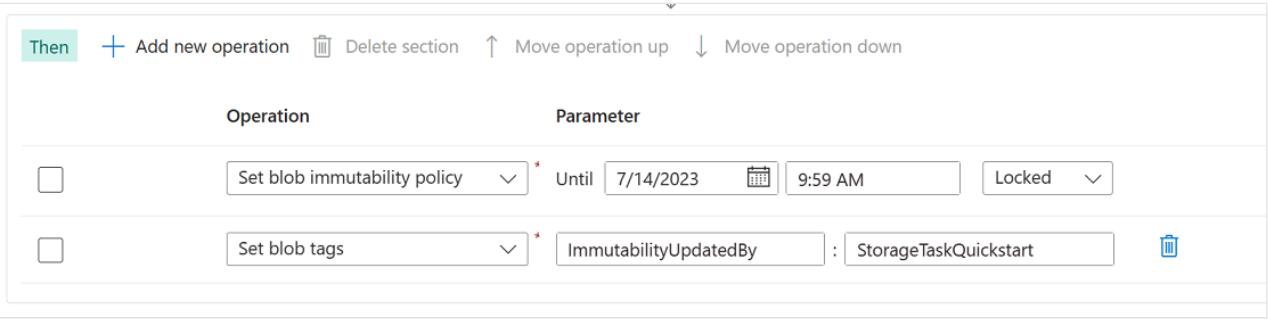
You can specify the operations that a storage task performs by making selections in **Then** section of the **Visual Builder** tab. Every storage task has at least one operation to perform when a blob or container meets the specified condition.

1. In the **Select an operation** drop-down list of the **Then** section, select **Set blob immutability policy**.



This operation applies a time-based immutability policy to Microsoft Word documents.

2. Select **Add new operation**, and then in the **Select a operation** drop-down list, select **Set blob tags**.
3. In the **Enter a tag name** box, Enter *ImmutabilityUpdatedBy*, and in the **Enter a tag value** box, enter *StorageTaskQuickstart*.



This operation adds a blob index tag to each Word document in that container.

4. Select **Next** to open the **Assignments** page.

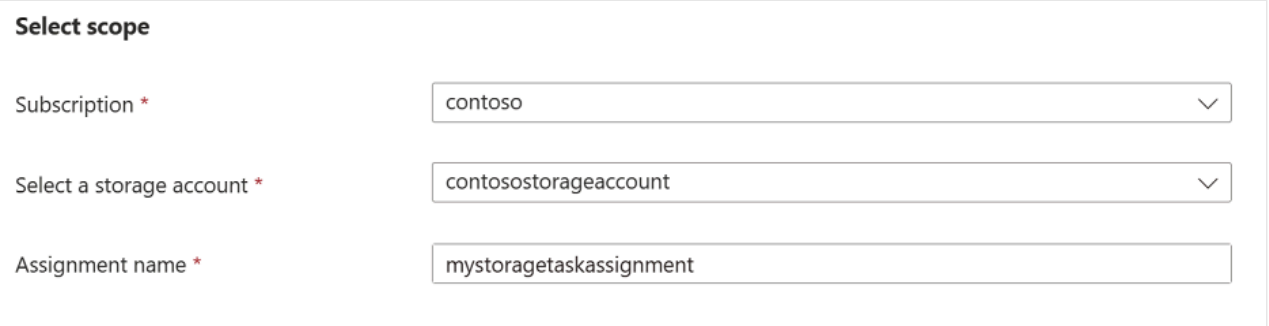
Add an assignment

A storage task *assignment* specifies a storage account. After you enable the storage task, the conditions and operations of your task will be applied to that storage account. The assignment also contains configuration properties which help you target specific blobs, or specify when and how often the task runs. You can add an assignment for each account you want to target.

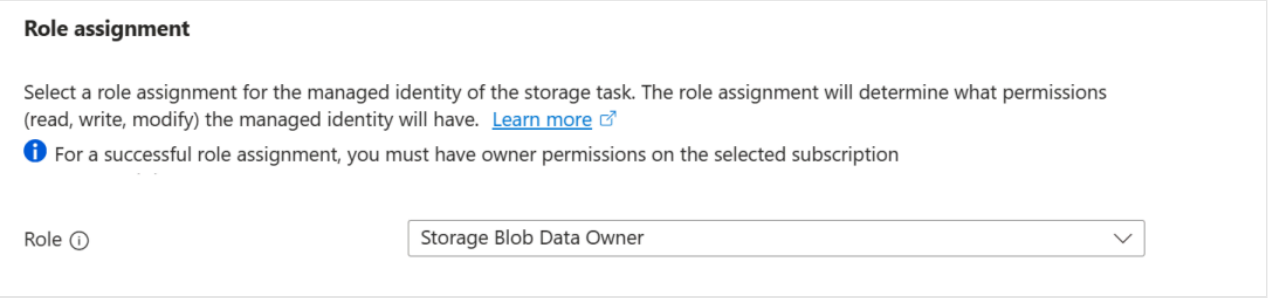
1. Select **Add assignment**.

   The **Add assignment** pane appears.

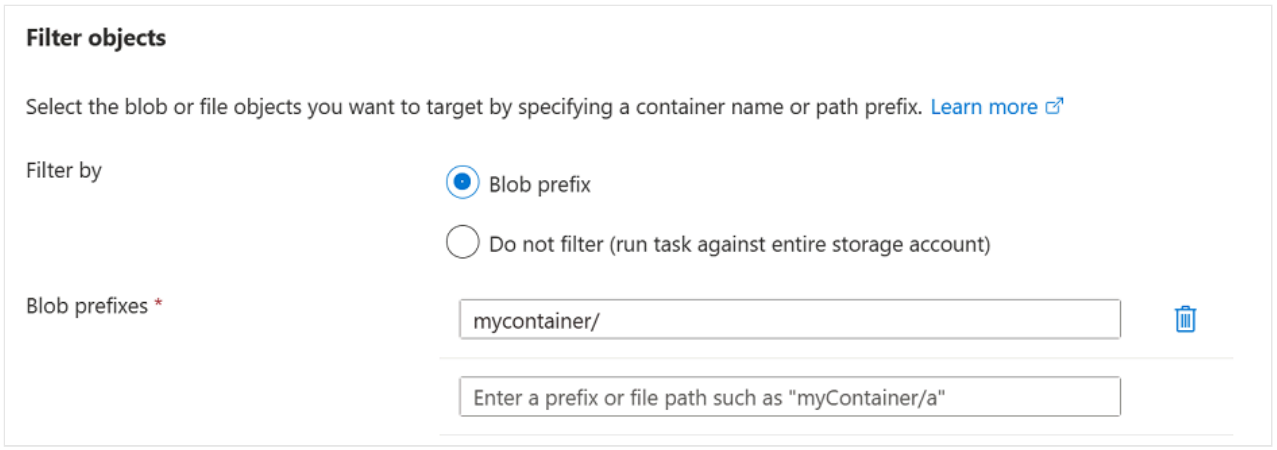
2. In the **Select scope** section, select your subscription and storage account and name the assignment *mystoragetaskassignment*.



3. In the **Role assignment** section, in the **Role** drop-down list, select the **Storage Blob Data Owner** to assign that role to the system-assigned managed identity of the storage task.

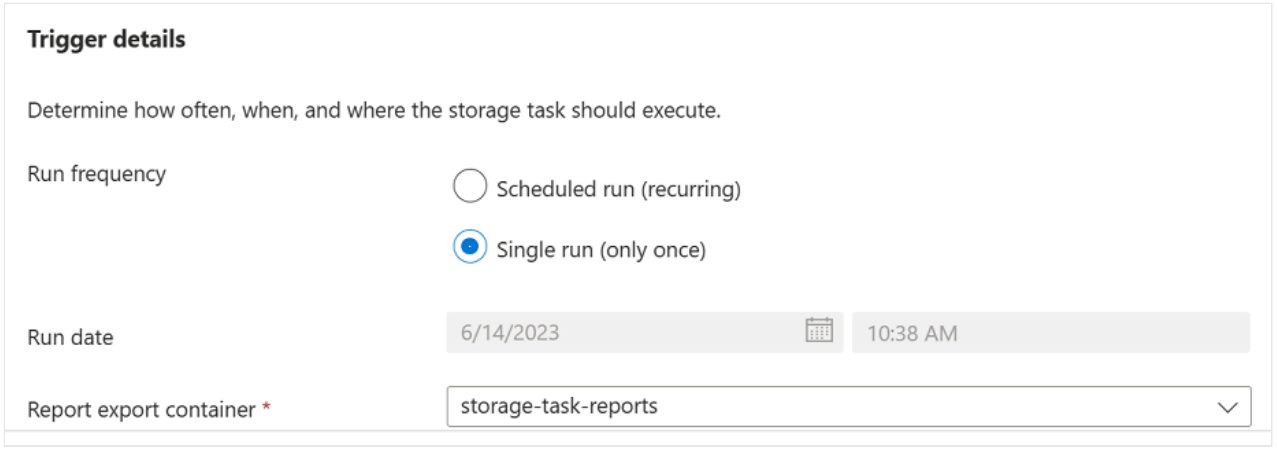


4. In the **Filter objects** section, make sure that the **Blob prefix** option is selected. Then, in the **Blob prefixes** box, enter the prefix of the container that you're using to complete this quickstart followed by the `/` character. For example, if your test container is named `mycontainer`, then enter `mycontainer/`.



   Filters help you narrow the scope of execution. If you want the task to evaluate all of the containers and blobs in an account, then you can select the **Do not filter** option instead.

5. In the **Trigger details** section, select **Single run (only once)** and then select the container where you'd like to store the execution reports.



6. Select **Add**.

7. In the **Tags** tab, select **Next**.

8. In the **Review + Create** tab, select **Review + create**.

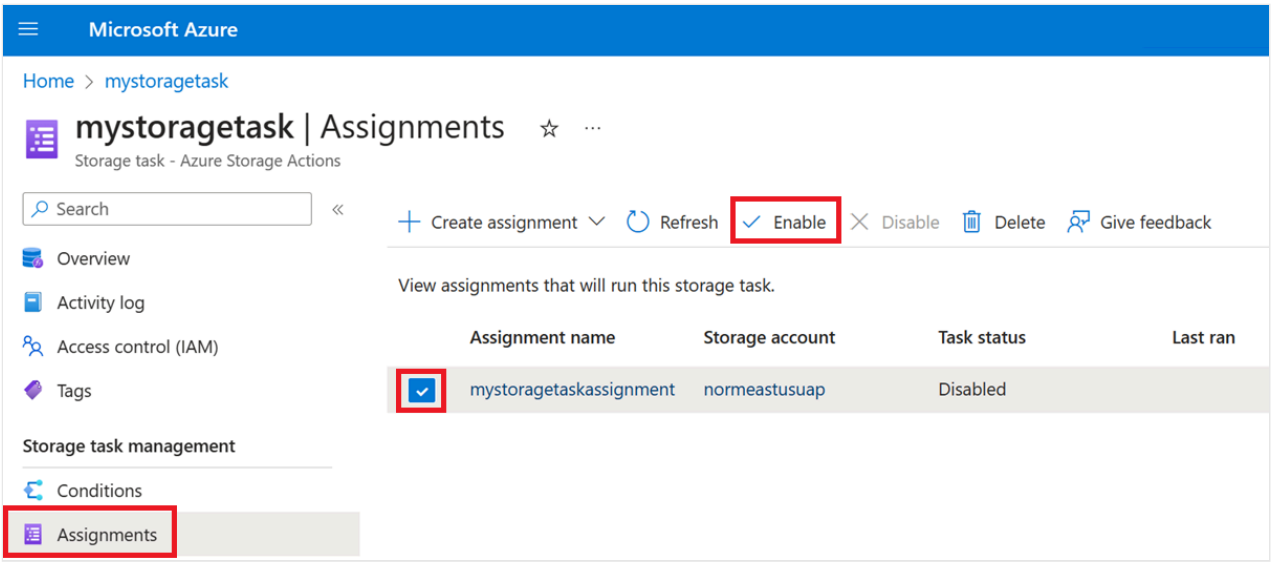
   When the task is deployed, the **Your deployment is complete** page appears.

9. Select **Go to resource** to open the **Overview** page of the storage task.

Enable the task assignment

Storage task assignments are disabled by default. Enable assignments from the **Assignments** page.

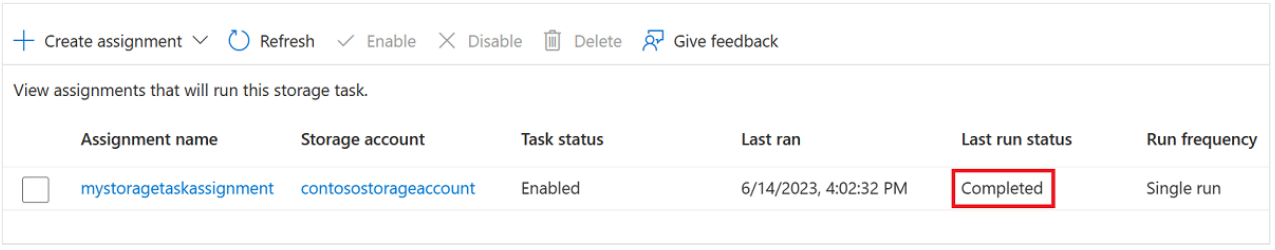
1. Select **Assignments**, select the **mystoragetaskassignment** assignment, and then select **Enable**.



   The task assignment is queued to run.

2. Periodically select **Refresh** to view an updated status.

   Until the task runs and then completes, the string **In progress** appears beneath the **Last run status** column. When the task completes, the string **Completed** appears in that column.



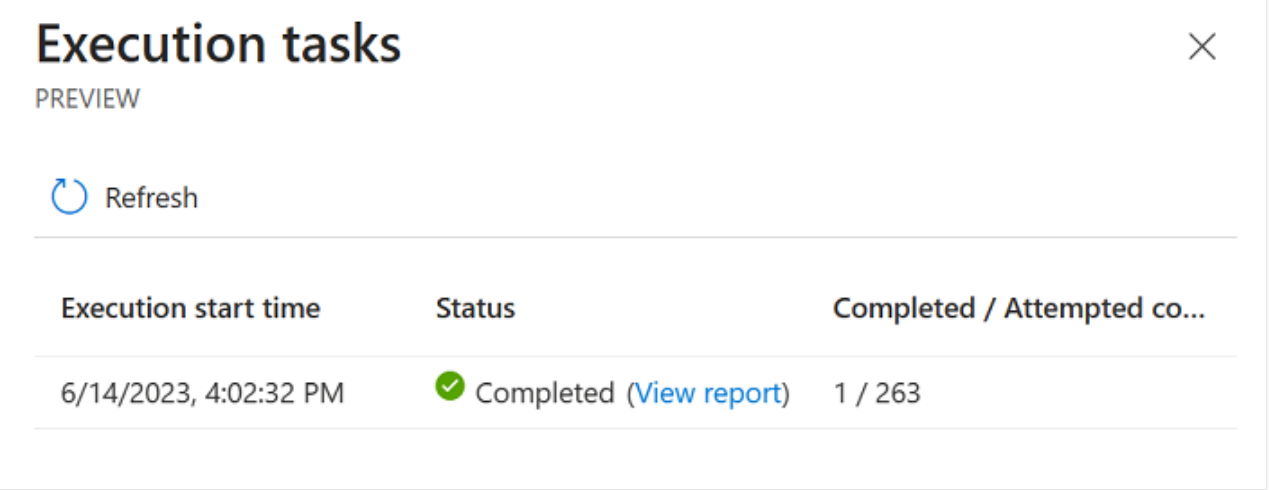
View results of the task run

After the task completes running, you can view the results of the run.

1. With the **Assignments** page still open, select **View task runs**.

   The **Execution tasks** pane appears, and in that pane, a line item which describes the report appears.

2. Select the **View report** link to download a report.



The report appears as a comma-separated list of the container, the blob, and the operation performed along with a status. You can also view these comma-separated reports in the container that you specified when you configured the assignment.

Next steps

Create a storage task

Quickstart: Create, assign, and run a storage task by using PowerShell

Article • 05/07/2025

In this quickstart, you learn how to use Azure PowerShell to create a storage task and assign it to an Azure Storage account. Then, you'll review the results of the run. The storage task applies a time-based immutability policy on any Microsoft Word documents that exist in the storage account.

Prerequisites

- An Azure subscription. See create an account for free.
- An Azure storage account. See create a storage account. As you create the account, make sure to enable version-level immutability support and that you don't enable the hierarchical namespace feature.

  During the public, you can target only storage accounts that are in the same region as the storage tasks.
- The Storage Blob Data Owner role is assigned to your user identity in the context of the storage account or resource group.
- A custom role assigned to your user identity in the context of the resource group which contains the RBAC actions necessary to assign a task to a storage account. See Permissions required to assign a task.
- .NET Framework is 4.7.2 or greater installed. For more information, see Download .NET Framework.
- PowerShell version `5.1` or higher.

Install the PowerShell module

1. Make sure you have the latest version of PowerShellGet installed.

   PowerShell

   ```
   Install-Module PowerShellGet -Repository PSGallery -Force
   ```

2. Close and then reopen the PowerShell console.

3. Install version 7.1.1-preview or later of the **Az.Storage** PowerShell module. You might need to uninstall other versions of the PowerShell module. For more information about installing Azure PowerShell, see Install Azure PowerShell with PowerShellGet.

```PowerShell
Install-Module Az.Storage -Repository PsGallery -RequiredVersion 7.1.1-preview -AllowClobber -AllowPrerelease -Force
```

4. Install **Az.StorageAction** module.

```PowerShell
Install-Module -Name Az.StorageAction -Repository PSGallery -Force
```

For more information about how to install PowerShell modules, see Install the Azure PowerShell module

Sign in to your Azure account

1. Open a Windows PowerShell command window, and then sign in to your Azure account with the `Connect-AzAccount` command and follow the on-screen directions.

```PowerShell
Connect-AzAccount
```

2. If your identity is associated with more than one subscription, and you aren't prompted to select the subscription, then set your active subscription to subscription of the storage account that you want operate upon. In this example, replace the `<subscription-id>` placeholder value with the ID of your subscription.

```PowerShell
Select-AzSubscription -SubscriptionId <subscription-id>
```

Create a storage task

1. Define a *condition* by using JSON. A condition is a collection of one or more clauses. Each clause contains a property, a value, and an operator. In the following JSON, the property

is `Name`, the value is `.docx`, and the operator is `endsWith`. This clause allows operations only on Microsoft Word documents.

```
PowerShell

$conditions = "[[endsWith(Name, '.docx')]]"
```

For a complete list of properties and operators, see Storage task conditions.

> **Tip**
>
> You can add multiple conditions to the same string and separate them with a comma.

2. Define each operation by using the `New-AzStorageActionTaskOperationObject` command.

   The following operation creates an operation that sets an immutability policy.

```
PowerShell

$policyoperation = New-AzStorageActionTaskOperationObject `
-Name SetBlobImmutabilityPolicy `
-Parameter @{"untilDate" = (Get-Date).AddDays(1); "mode" = "locked"} `
-OnFailure break `
-OnSuccess continue
```

   The following operation sets a blob index tag in the metadata of a Word document.

```
PowerShell

$tagoperation = New-AzStorageActionTaskOperationObject -Name SetBlobTags `
-Parameter @{"tagsetImmutabilityUpdatedBy"="StorageTaskQuickstart"} `
-OnFailure break `
-OnSuccess continue
```

3. Create a storage task by using the `New-AzStorageActionTask` command, and pass in the conditions and operations that you defined earlier. This example creates a storage task named `mystoragetask` in resource group `mystoragetaskresourcegroup` in the West US region.

```
PowerShell

$task = New-AzStorageActionTask `
-Name mystoragetask `
```

```
-ResourceGroupName mystoragetaskresourcegroup `
-Location westus `
-Enabled `
-Description 'my powershell storage task' `
-IfCondition $conditions `
-IfOperation $policyoperation,$tagoperation `
-EnableSystemAssignedIdentity:$true
```

Create an assignment

A storage task *assignment* specifies a storage account. After you enable the storage task, the conditions and operations of your task will be applied to that storage account. The assignment also contains configuration properties which help you target specific blobs, or specify when and how often the task runs. You can add an assignment for each account you want to target.

1. Create a storage task assignment by using the `New-AzStorageTaskAssignment` command. The following assignment targets the `mycontainer` container of an account named `mystorageaccount`. This assignment specifies that the task will run only one time, and will save execution reports to a folder named `storage-tasks-report`. The task is scheduled to run `10` minutes from the present time.

   PowerShell

   ```
   $startTime = (Get-Date).AddMinutes(10)

   New-AzStorageTaskAssignment `
   -ResourceGroupName mystoragetaskresourcegroup `
   -AccountName mystorageaccount `
   -name mystoragetaskAssignment `
   -TaskId $task.Id `
   -ReportPrefix "storage-tasks-report" `
   -TriggerType RunOnce `
   -StartOn $startTime.ToUniversalTime() `
   -Description "task assignment" `
   -Enabled:$true `
   -TargetPrefix "mycontainer/" `
   -TargetExcludePrefix ""
   ```

2. Give the storage task permission to perform operations on the target storage account. Assign the role of `Storage Blob Data Owner` to the system-assigned managed identity of the storage task by using the `New-AzRoleAssignment` command.

   PowerShell

   ```
   New-AzRoleAssignment `
   -ResourceGroupName mystoragetaskresourcegroup `
   ```

```
-ResourceName mystorageaccount `
-ResourceType "Microsoft.Storage/storageAccounts" `
-ObjectId $task.IdentityPrincipalId  `
-RoleDefinitionName "Storage Blob Data Owner"
```

View the results of a task run

After the task completes running, get a run report summary for each assignment by using the `Get-AzStorageActionTasksReport` command.

PowerShell

```
Get-AzStorageActionTasksReport `
-ResourceGroupName mystoragetaskresourcegroup `
-StorageTaskName mystoragetask | Format-List
```

The `SummaryReportPath` field of each report summary contains a path to a detailed report. That report contains comma-separated list of the container, the blob, and the operation performed along with a status.

Clean up resources

Remove all of the assets you've created. The easiest way to remove the assets is to delete the resource group. Removing the resource group also deletes all resources included within the group. In the following example, removing the resource group removes the storage account and the resource group itself.

PowerShell

```
Remove-AzResourceGroup -Name $ResourceGroup
```

Next steps

Create a storage task

Microsoft Azure PowerShell storage actions cmdlets reference

- Storage PowerShell cmdlets

Quickstart: Create, assign, and run a storage task by using Azure CLI

Article • 05/07/2025

In this quickstart, you learn how to use Azure CLI to create a storage task and assign it to an Azure Storage account. Then, you'll review the results of the run. The storage task applies a time-based immutability policy any Microsoft Word documents that exist in the storage account.

Prerequisites

- An Azure subscription. See create an account for free.
- An Azure storage account. See create a storage account. As you create the account, make sure to enable version-level immutability support and that you don't enable the hierarchical namespace feature.

  During the public, you can target only storage accounts that are in the same region as the storage tasks.
- The Storage Blob Data Owner role is assigned to your user identity in the context of the storage account or resource group.
- A custom role assigned to your user identity in the context of the resource group which contains the RBAC actions necessary to assign a task to a storage account. See Permissions required to assign a task.

Prepare your environment for the Azure CLI

- Use the Bash environment in Azure Cloud Shell. For more information, see Get started with Azure Cloud Shell.

  Launch Cloud Shell
- If you prefer to run CLI reference commands locally, install the Azure CLI. If you're running on Windows or macOS, consider running Azure CLI in a Docker container. For more information, see How to run the Azure CLI in a Docker container.
  - If you're using a local installation, sign in to the Azure CLI by using the az login command. To finish the authentication process, follow the steps displayed in your terminal. For other sign-in options, see Authenticate to Azure using Azure CLI.

- When you're prompted, install the Azure CLI extension on first use. For more information about extensions, see Use and manage extensions with the Azure CLI.
- Run az version to find the version and dependent libraries that are installed. To upgrade to the latest version, run az upgrade.

- This article requires version 2.57.0 or later of the Azure CLI. If using Azure Cloud Shell, the latest version is already installed.

Sign in to your Azure account

1. Sign in to your Azure account with the `az login` command.

   Azure CLI

   ```
   az login
   ```

   If the CLI can open your default browser, it will do so and load an Azure sign-in page.

   Otherwise, open a browser page at https://aka.ms/devicelogin and enter the authorization code displayed in your terminal. Then, sign in with your account credentials in the browser.

2. If your identity is associated with more than one subscription, and you aren't prompted to select the subscription, then set your active subscription to subscription of the storage account that you want operate upon. In this example, replace the `<subscription-id>` placeholder value with the ID of your subscription.

   Azure CLI

   ```
   az account set --subscription <subscription-id>
   ```

Create a storage task

1. Define a *condition* by using JSON. A condition a collection of one or more clauses. Each clause contains a property, a value, and an operator. In the following JSON, the property is `Name`, the value is `.docx`, and the operator is endsWith. This clause allows operations only on Microsoft Word documents. To learn more about the structure of conditions and a complete list of properties and operators, see Storage task conditions.

   Azure CLI

```
conditionclause="[[endsWith(Name,'/.docx'/)]]"
```

> **Note**
>
> Azure CLI uses shorthand syntax. Shorthand syntax is a simplified representation of a JSON string. To learn more, see How to use shorthand syntax with Azure CLI.

2. Define each operation. The following example defines an operation that sets an immutability policy, and an operation that sets a blob index tag in the metadata of a Word document.

Azure CLI

```
policyoperation="{name:'SetBlobImmutabilityPolicy',parameters:
{untilDate:'2024-10-
20T22:30:40',mode:'locked'},onSuccess:'continue',onFailure:'break'}"
tagoperation="{name:'SetBlobTags',parameters:
{'tagsetImmutabilityUpdatedBy':'StorageTaskQuickstart'},onSuccess:'continue',
onFailure:'break'}"
action="{if:
{condition:'"${conditionclause}"',operations:/["${policyoperation}","${tagope
ration}"]}}"
```

3. Create a storage task by using the `az storage-actions task create` command, and pass in the conditions and operations that you defined earlier. This example creates a storage task named `mystoragetask` in resource group `mystoragetaskresourcegroup` in the West US region.

Azure CLI

```
az storage-actions task create \
   -g mystoragetaskresourcegroup \
   -n mystoragetask \
   --identity "{type:SystemAssigned}" \
   --action "{if:{condition:'"${conditionclause}"',operations:
["${policyoperation}","${tagoperation}"]}}" \
   --description "My storage task" --enabled true
```

Add an assignment

A storage task *assignment* specifies a storage account. After you enable the storage task, the conditions and operations of your task will be applied to that storage account. The assignment

also contains configuration properties which help you target specific blobs, or specify when and how often the task runs. You can add an assignment for each account you want to target.

1. Create a storage task assignment by using the `az storage account task-assignment create` command. The following assignment targets the `mycontainer` container of an account named `mystorageaccount`. This assignment specifies that the task will run only one time, and will save execution reports to a folder named `storage-tasks-report`. The task is scheduled to run `10` minutes from the present time.

   Azure CLI

   ```
   id=$(az storage-actions task show -g mystoragetaskresourcegroup -n
   mystoragetask --query "id")
   current_datetime=$(date +"%Y-%m-%dT%H:%M:%S")
   executioncontextvariable="{target:{prefix:[mycontainer/],excludePrefix:
   []},trigger:{type:'RunOnce',parameters:{startOn:'"${current_datetime}"'}}}"

   az storage account task-assignment create \
      -g mystoragetaskresourcegroup \
      -n mystoragetaskassignment \
      --account-name mystorageaccount \
      --description 'My Storage task assignment' \
      --enabled false \
      --task-id $id \
      --execution-context $executioncontextvariable \
      --report "{prefix:storage-tasks-report}"
   ```

2. Give the storage task permission to perform operations on the target storage account. Assign the role of `Storage Blob Data Owner` to the system-assigned managed identity of the storage task.

   Azure CLI

   ```
   $roleDefinitionId="b7e6dc6d-f1e8-4753-8033-0f276bb0955b" \
   $principalID=az storage-actions task show -g mystoragetaskresourcegroup -n
   mystoragetask --query "identity.principalId"
   $storageAccountID=az storage account show --name mystorageaccount --resource-
   group mystoragetaskresourcegroup --query "id"

   az role assignment create \
      --assignee-object-id $principalID \
      --scope $storageAccountID \
      --role $roleDefinitionId \
      --description "My role assignment"
   ```

View the results of a task run

After the task completes running, get a run report summary for each assignment by using the blah command.

Azure CLI

```
az storage account task-assignment list-report \
   --account-name mystorageaccount \
   --resource-group mystoragetaskresourcegroup \
   --name mystoragetaskassignment
```

The `SummaryReportPath` field of each report summary contains a path to a detailed report. That report contains comma-separated list of the container, the blob, and the operation performed along with a status.

Clean up resources

Remove all of the assets you've created. The easiest way to remove the assets is to delete the resource group. Removing the resource group also deletes all resources included within the group. In the following example, removing the resource group removes the storage account and the resource group itself.

Azure CLI

```
az group delete \
   --name <resource-group> \
   --no-wait
```

Next steps

Create a storage task

Create a storage task by using Azure Resource Manager template (ARM template)

Article • 05/07/2025

This quickstart describes how to create a storage task by using an Azure Resource Manager template (ARM template).

An Azure Resource Manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. You describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "_generator": {
      "name": "bicep",
      "version": "0.32.4.45862",
      "templateHash": "11663963517791910133"
    }
  },
  "parameters": {
    "storageTaskName": {
      "type": "string",
      "minLength": 3,
```

```
      "maxLength": 18,
      "metadata": {
        "description": "The name of storage task."
      }
    },
    "description": {
      "type": "string",
      "metadata": {
        "description": "A description of the storage task."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "The region in which to create the storage task."
      }
    },
    "lockedUntilDate": {
      "type": "string",
      "defaultValue": "[dateTimeAdd(utcNow(), 'P1D')]",
      "metadata": {
        "description": "Locks the file for one day."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.StorageActions/storageTasks",
      "apiVersion": "2023-01-01",
      "name": "[parameters('storageTaskName')]",
      "location": "[parameters('location')]",
      "identity": {
        "type": "SystemAssigned"
      },
      "properties": {
        "action": {
          "if": {
            "condition": "[[[endsWith(Name, '.docx')]]",
            "operations": [
              {
                "name": "SetBlobImmutabilityPolicy",
                "onSuccess": "continue",
                "onFailure": "break",
                "parameters": {
                  "untilDate": "[parameters('lockedUntilDate')]",
                  "mode": "locked"
                }
              },
              {
                "name": "SetBlobTags",
                "onSuccess": "continue",
                "onFailure": "break",
                "parameters": {
                  "tagsetImmutabilityUpdatedBy": "StorageTaskQuickstart"
```

```
                }
              }
            ]
          }
        },
        "description": "[parameters('description')]",
        "enabled": true
      }
    }
  ]
}
```

Deploy the template

1. Select the following link to sign in to Azure and open a template. The template creates a key vault and a secret.

   Deploy to Azure

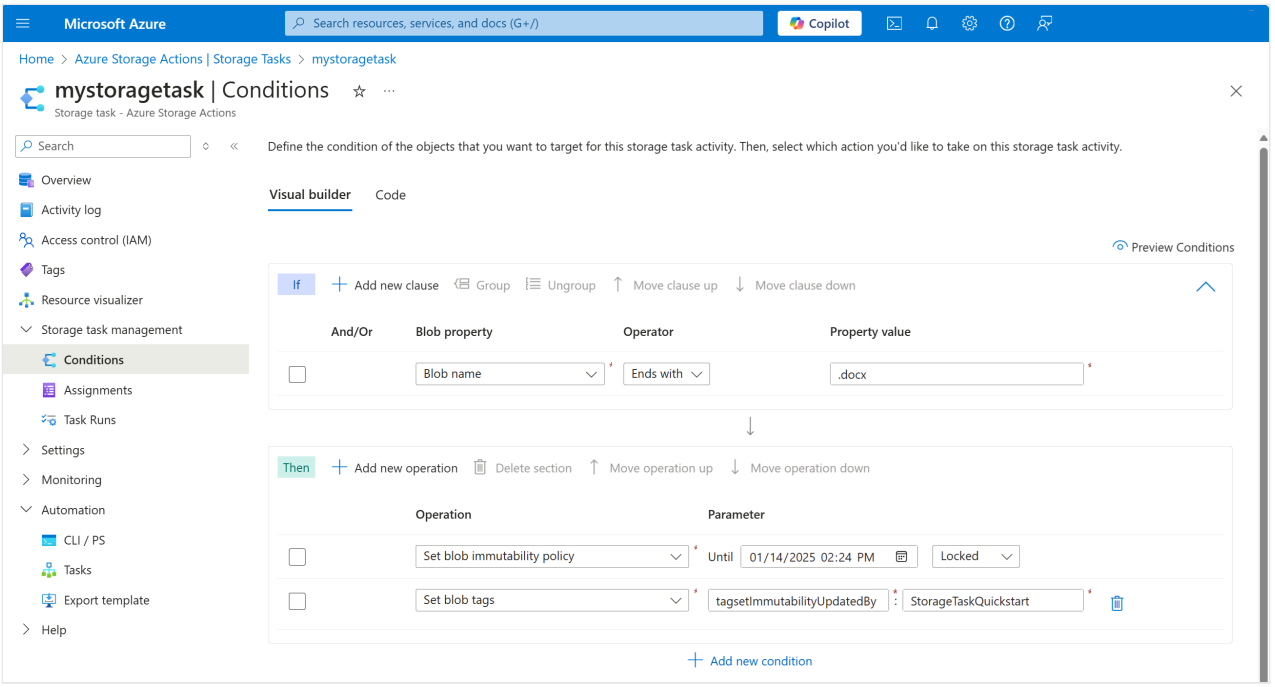
2. Specify the subscription, resource group, and the storage task name. Then, select **Review + create** to deploy the template.

You can also use the Azure PowerShell, Azure CLI, and REST API. To learn other deployment methods, see Deploy templates.

Review deployed resources

1. In the Azure portal, search for *Storage Tasks*. Then, under **Services**, select **Storage tasks - Azure Storage Actions**.

2. In the list of storage tasks, search for the name of the storage task that you deployed.



Clean up resources

When no longer needed, delete the resource group. The resource group and all the resources in the resource group are deleted. Use the following command to delete the resource group and all its contained resources.

Azure CLI

Azure CLI

```azurecli
az group delete --name <resource-group-name>
```

Replace `<resource-group-name>` with the name of your resource group.

Next steps

Assign a storage task to a storage account.

Create and manage a storage task assignment

Quickstart: Create a storage task with Bicep

Article • 05/07/2025

This quickstart describes how to create a storage task by using Bicep.

Bicep is a domain-specific language (DSL) that uses declarative syntax to deploy Azure resources. It provides concise syntax, reliable type safety, and support for code reuse. Bicep offers the best authoring experience for your infrastructure-as-code solutions in Azure.

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Review the Bicep file

The Bicep file used in this quickstart is from Azure Quickstart Templates.

Bicep

```
@sys.description('The name of storage task.')
@minLength(3)
@maxLength(18)
param storageTaskName string

@sys.description('A description of the storage task.')
param description string

@sys.description('The region in which to create the storage task.')
param location string = resourceGroup().location

@sys.description('Locks the file for one day.')
param lockedUntilDate string = dateTimeAdd(utcNow(), 'P1D')

resource storageTask 'Microsoft.StorageActions/storageTasks@2023-01-01' = {
  name: storageTaskName
  location: location
  identity: {
    type: 'SystemAssigned'
  }
  properties: {
    action: {
      if: {
        condition: '[[endsWith(Name, \'.docx\')]]'
        operations: [
         {
            name: 'SetBlobImmutabilityPolicy'
            onSuccess: 'continue'
            onFailure: 'break'
```

```
            parameters: {
              untilDate: lockedUntilDate
              mode: 'locked'
            }
          }
          {
            name: 'SetBlobTags'
            onSuccess: 'continue'
            onFailure: 'break'
            parameters: {
                tagsetImmutabilityUpdatedBy: 'StorageTaskQuickstart'
            }
          }
        ]
      }

    }
    description: description
    enabled: true
  }
}
```

The Microsoft.StorageActions/storageTasks Azure resource is defined in the Bicep file.

Deploy the Bicep file

1. Save the Bicep file as **main.bicep** to your local computer.
2. Deploy the Bicep file using either Azure CLI or Azure PowerShell.

Azure CLI

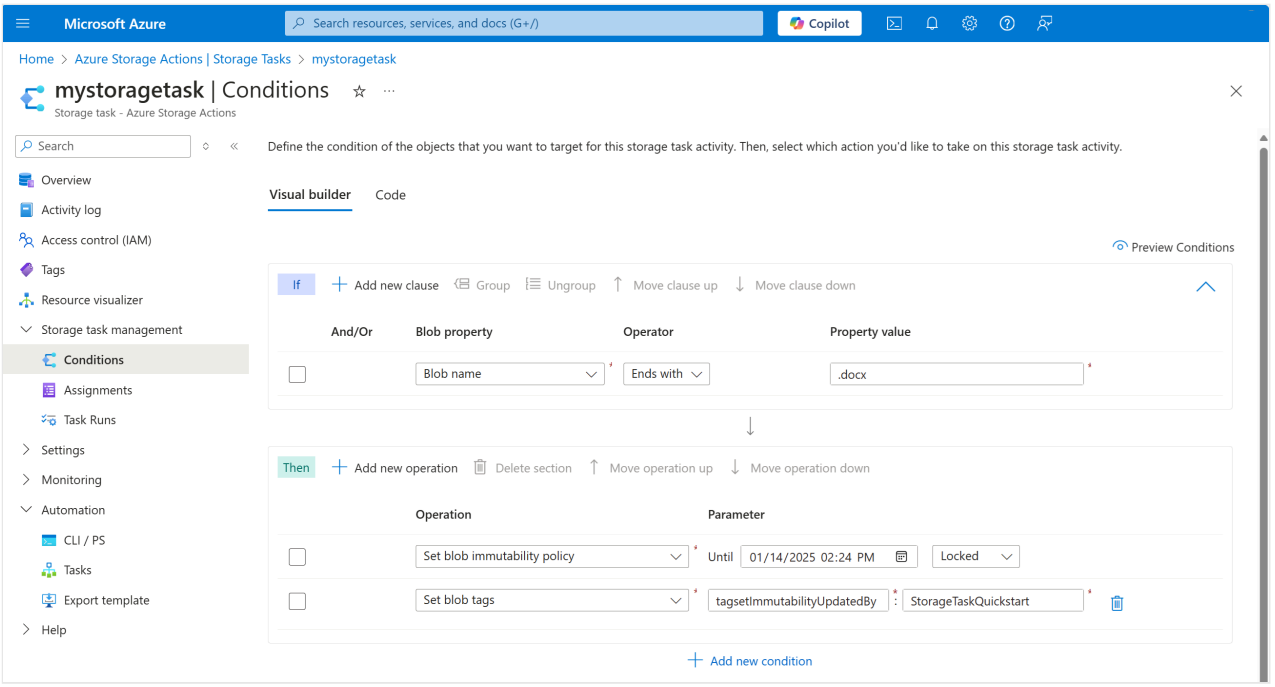
Azure CLI

```
az group create --name exampleRG --location <region>

az deployment group create --resource-group exampleRG --template-file
main.bicep --parameters storageTaskName="<storage-task-name>" description="
<description>"
```

Review deployed resources

1. In the Azure portal, search for *Storage Tasks*. Then, under **Services**, select **Storage tasks - Azure Storage Actions**.
2. In the list of storage tasks, search for the name of the storage task that you deployed.



Clean up resources

When no longer needed, delete the resource group. The resource group and all the resources in the resource group are deleted. Use the following command to delete the resource group and all its contained resources.

Azure CLI

Azure CLI

```azurecli
az group delete --name <resource-group-name>
```

Replace `<resource-group-name>` with the name of your resource group.

Next steps

Assign a storage task to a storage account.

Create and manage a storage task assignment

Quickstart: Create a storage task using Terraform

Article • 05/07/2025

A storage task can perform operations on blobs in an Azure Storage account. As you create a task, you can define the conditions that must be met by each object (container or blob), and the operations to perform on the object. You can also identify one or more Azure Storage account targets. See What are Azure Storage Actions?.

In this how-to article, you learn how to create a storage task using Terraform.

Terraform enables the definition, preview, and deployment of cloud infrastructure. Using Terraform, you create configuration files using HCL syntax. The HCL syntax allows you to specify the cloud provider - such as Azure - and the elements that make up your cloud infrastructure. After you create your configuration files, you create an *execution plan* that allows you to preview your infrastructure changes before they're deployed. Once you verify the changes, you apply the execution plan to deploy the infrastructure.

In this article, you learn how to:

- ✓ Generate a random name for the resource group.
- ✓ Create a new Azure resource group with the generated name.
- ✓ Generate a random string to be used as the storage task name.
- ✓ Calculate a future date by offsetting the current date by a certain number of days.
- ✓ Create a new Azure API resource of type "Microsoft.StorageActions/storageTasks".
- ✓ Specify the required providers for Terraform, including their sources and versions.
- ✓ Configure the Azure provider with specific features.
- ✓ Define several variables, including the location of the resource group, the prefix for the resource group name, the number of offset days, and the description of the storage task.

Prerequisites

- Create an Azure account with an active subscription. You can create an account for free.
- Install and configure Terraform

Implement the Terraform code

The sample code for this article is located in the Azure Terraform GitHub repo. You can view the log file containing the test results from current and previous versions of Terraform. See more articles and sample code showing how to use Terraform to manage Azure resources

1. Create a directory in which to test and run the sample Terraform code, and make it the current directory.

2. Create a file named `providers.tf` and insert the following code.

Terraform

```
terraform {
  required_providers {
    azapi = {
      source  = "Azure/azapi"
      version = "~>2.0"
    }
    azurerm = {
      source  = "hashicorp/azurerm"
      version = "~>4.0"
    }
    random = {
      source  = "hashicorp/random"
      version = "~>3.0"
    }
  }
}

provider "azurerm" {
  features {
    resource_group {
      prevent_deletion_if_contains_resources = false
    }
  }
}
```

3. Create a file named `main.tf` and insert the following code.

Terraform

```
data "azurerm_client_config" "current" {}

# Generate random resource group name
resource "random_pet" "rg_name" {
  prefix = var.resource_group_name_prefix
}

resource "azurerm_resource_group" "rg" {
  location = var.resource_group_location
  name     = random_pet.rg_name.id
}

# Generate random value for the storage task name
resource "random_string" "storage_task_name" {
  length  = 8
  lower   = true
```

```
  numeric = false
  special = false
  upper   = false
}

resource "time_offset" "locked_until_date" {
  offset_days = var.offset_days
}

resource "azapi_resource" "my_terraform_task" {
  type      = "Microsoft.StorageActions/storageTasks@2023-01-01"
  name      = random_string.storage_task_name.result
  parent_id = azurerm_resource_group.rg.id
  location  = azurerm_resource_group.rg.location
  identity {
    type = "SystemAssigned"
  }
  body = {
    properties = {
      action = {
        if = {
          condition = "[[endsWith(Name, '.docx')]]"
          operations = [
            {
              name      = "SetBlobImmutabilityPolicy"
              onFailure = "break"
              onSuccess = "continue"
              parameters = {
                untilDate : time_offset.locked_until_date.rfc3339
                mode : "locked"
              }
            }
          ]
        }
      }
      description = var.storage_task_description
      enabled     = true
    }
  }
}
```

4. Create a file named `variables.tf` and insert the following code.

Terraform

```
variable "resource_group_location" {
  type        = string
  description = "Location of the resource group."
  default     = "westus"
}
variable "resource_group_name_prefix" {
  type        = string
  description = "Prefix of the resource group name that's combined with a
```

```
random ID so name is unique in your Azure subscription."
  default     = "rg"
}
variable "offset_days" {
  type        = number
  description = "The number of days to lock the file."
  default     = 1
}
variable "storage_task_description" {
  type        = string
  description = "Description of the storage task"
  default     = "My terraform storage task"
}
```

5. Create a file named `outputs.tf` and insert the following code.

Terraform

```
output "resource_group_name" {
  value = azurerm_resource_group.rg.name
}

output "storage_task_name" {
  value = azapi_resource.my_terraform_task.name
}
```

> **Important**
>
> If you're using the 4.x azurerm provider, you must explicitly specify the Azure subscription ID to authenticate to Azure before running the Terraform commands.
>
> One way to specify the Azure subscription ID without putting it in the `providers` block is to specify the subscription ID in an environment variable named `ARM_SUBSCRIPTION_ID`.
>
> For more information, see the Azure provider reference documentation.

Initialize Terraform

Run terraform init to initialize the Terraform deployment. This command downloads the Azure provider required to manage your Azure resources.

Console

```
terraform init -upgrade
```

Key points:

- The `-upgrade` parameter upgrades the necessary provider plugins to the newest version that complies with the configuration's version constraints.

Create a Terraform execution plan

Run terraform plan to create an execution plan.

Console

```
terraform plan -out main.tfplan
```

Key points:

- The `terraform plan` command creates an execution plan, but doesn't execute it. Instead, it determines what actions are necessary to create the configuration specified in your configuration files. This pattern allows you to verify whether the execution plan matches your expectations before making any changes to actual resources.
- The optional `-out` parameter allows you to specify an output file for the plan. Using the `-out` parameter ensures that the plan you reviewed is exactly what is applied.

Apply a Terraform execution plan

Run terraform apply to apply the execution plan to your cloud infrastructure.

Console

```
terraform apply main.tfplan
```

Key points:

- The example `terraform apply` command assumes you previously ran `terraform plan -out main.tfplan`.
- If you specified a different filename for the `-out` parameter, use that same filename in the call to `terraform apply`.
- If you didn't use the `-out` parameter, call `terraform apply` without any parameters.

Verify the results

Azure PowerShell

1. Get the Azure resource group name.

```
Console

$resource_group_name=$(terraform output -raw resource_group_name)
```

2. Get the storage task name.

```
Console

$storage_task_name=$(terraform output -raw storage_task_name)
```

3. Run Get-AzStorageActionTask to get the storage task properties.

```
Azure PowerShell

Get-AzStorageActionTask -Name $storage_task_name -ResourceGroupName
$resource_group_name
```

Clean up resources

When you no longer need the resources created via Terraform, do the following steps:

1. Run terraform plan and specify the `destroy` flag.

```
Console

terraform plan -destroy -out main.destroy.tfplan
```

Key points:

- The `terraform plan` command creates an execution plan, but doesn't execute it. Instead, it determines what actions are necessary to create the configuration specified in your configuration files. This pattern allows you to verify whether the execution plan matches your expectations before making any changes to actual resources.
- The optional `-out` parameter allows you to specify an output file for the plan. Using the `-out` parameter ensures that the plan you reviewed is exactly what is applied.

2. Run terraform apply to apply the execution plan.

```
Console

terraform apply main.destroy.tfplan
```

Troubleshoot Terraform on Azure

Troubleshoot common problems when using Terraform on Azure.

Next steps

Create and manage a storage task assignment

Quickstart: Create and assign a storage task by using .NET

Article • 05/07/2025

In this quickstart, you learn how to use the Azure Storage Actions client library for .NET to create a storage task and assign it to an Azure Storage account. Then, you'll review the results of the run. The storage task applies a time-based immutability policy on any Microsoft Word documents that exist in the storage account.

API reference documentation | Library source code | Package (NuGet) | Samples

Prerequisites

- Azure subscription - create one for free
- An Azure storage account. See create a storage account. As you create the account, make sure to enable version-level immutability support and that you don't enable the hierarchical namespace feature.
- The Storage Blob Data Owner role is assigned to your user identity in the context of the storage account or resource group.
- A custom role assigned to your user identity in the context of the resource group which contains the RBAC actions necessary to assign a task to a storage account. See Permissions required to assign a task.
- .NET Framework is 4.7.2 or greater installed. For more information, see Download .NET Framework.

Setting up

This section walks you through preparing a project.

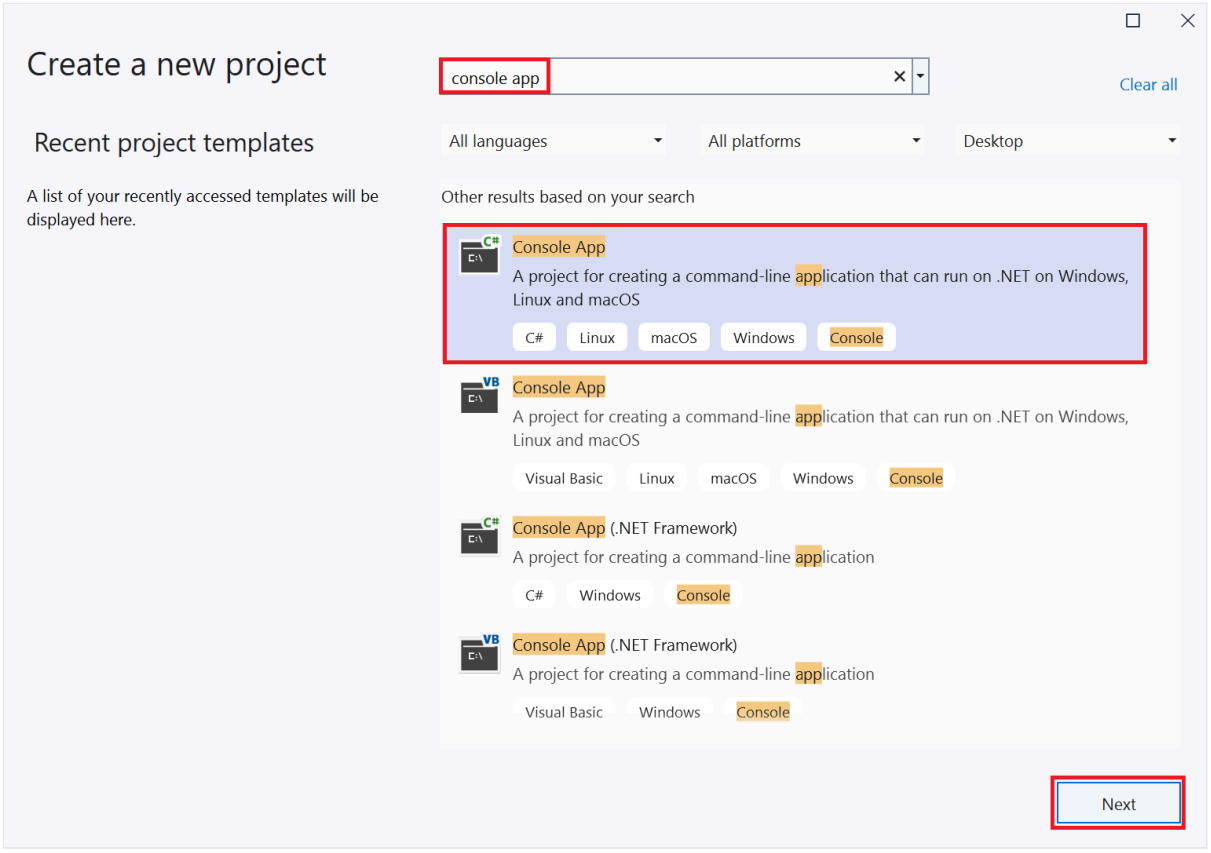
Create the project

Create a .NET console app using either the .NET CLI or Visual Studio 2022.

Visual Studio 2022

1. At the top of Visual Studio, navigate to **File** > **New** > **Project...**

2. In the dialog window, enter *console app* into the project template search box and select the first result. Choose **Next** at the bottom of the dialog.



3. For the **Project Name**, enter *StorageTaskQuickstart*. Leave the default values for the rest of the fields and select **Next**.

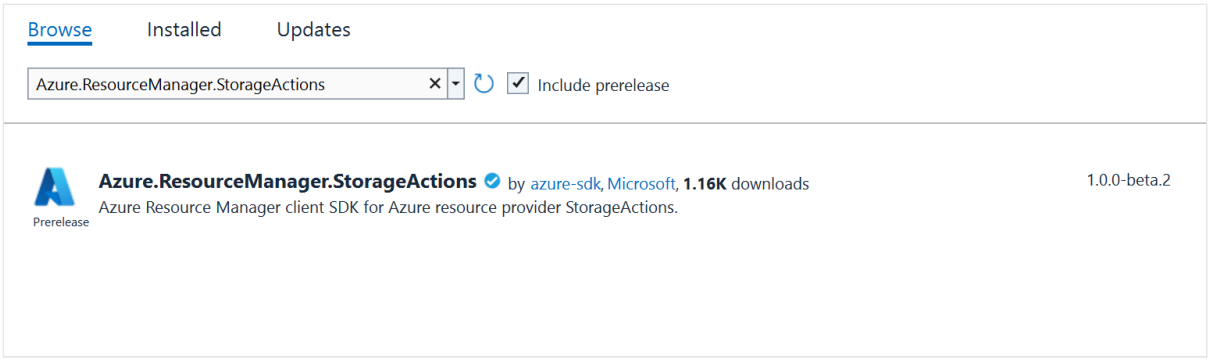
4. For the **Framework**, ensure the latest installed version of .NET is selected. Then choose **Create**. The new project opens inside the Visual Studio environment.

Install the packages

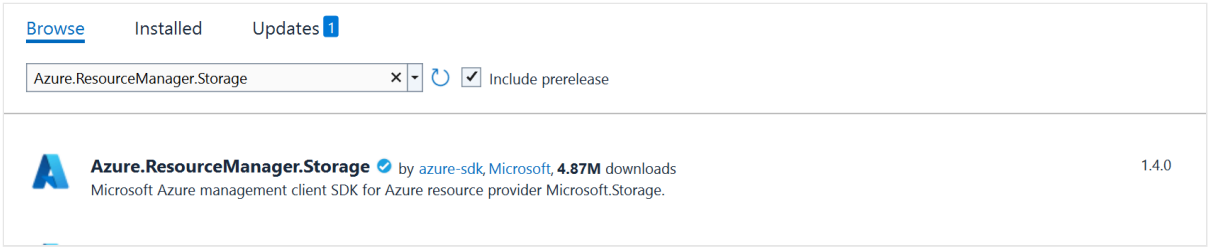
Visual Studio 2022

1. In **Solution Explorer**, right-click the **Dependencies** node of your project. Select **Manage NuGet Packages**.

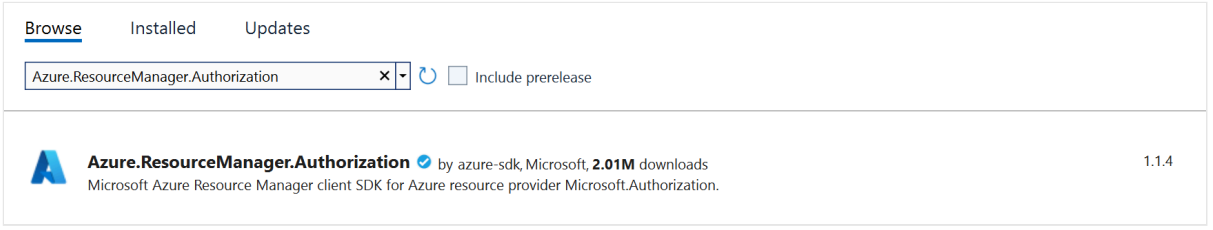
2. In the resulting window, select the **Include prerelease** checkbox, and then search for *Azure.ResourceManager.StorageActions*. Select the appropriate result, and select **Install**.



3. Next, search for *Azure.ResourceManager.Storage*. Select the appropriate result, and select **Install**.



4. Search for *Azure.ResourceManager.Authorization**. Select the appropriate result, and select **Install**.



Set up the app code

Replace the starting code in the `Program.cs` file so that it matches the following example, which includes the necessary `using` statements for this QuickStart.

C#

```
using Azure;
using Azure.Core;
using Azure.ResourceManager;
using Azure.ResourceManager.Models;
using Azure.ResourceManager.Resources;
using Azure.ResourceManager.StorageActions;
using Azure.ResourceManager.StorageActions.Models;
using Azure.ResourceManager.Storage.Models;
using Azure.ResourceManager.Storage;
```

```
using Azure.ResourceManager.Authorization;
using Azure.ResourceManager.Authorization.Models;
```

Sign in and connect your app code to Azure using DefaultAzureCredential

1. For local development, make sure you're authenticated with the same Microsoft Entra account you assigned the role to. You can authenticate via popular development tools, such as the Azure CLI or Azure PowerShell. The development tools with which you can authenticate vary across languages.

   Azure CLI

   Sign-in to Azure through the Azure CLI using the following command:

   Azure CLI

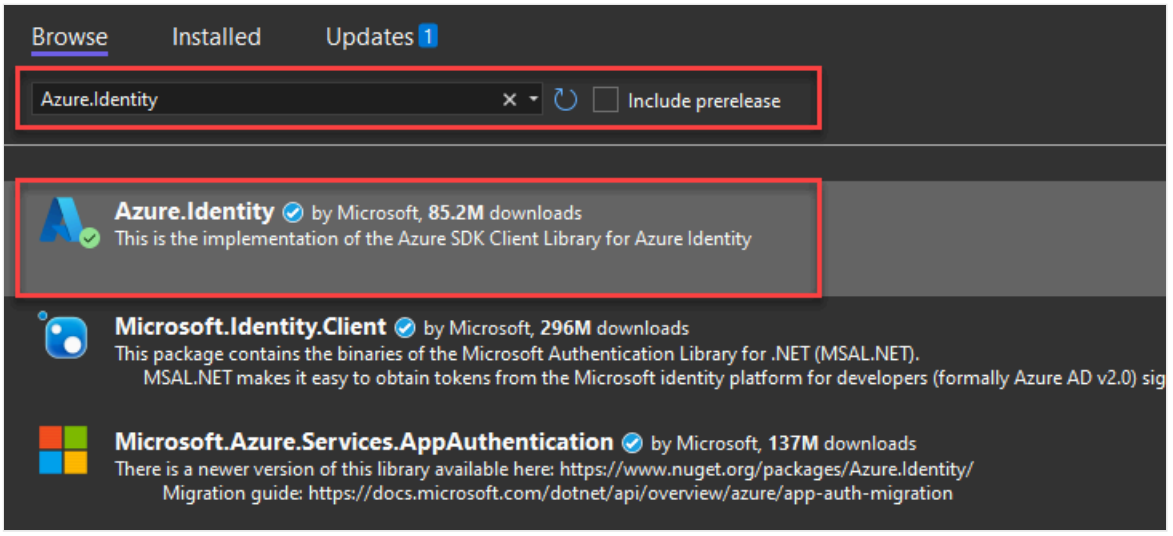
   ```
   az login
   ```

2. To use `DefaultAzureCredential`, add the **Azure.Identity** package to your application.

   Visual Studio

   a. In **Solution Explorer**, right-click the **Dependencies** node of your project. Select **Manage NuGet Packages**.

   b. In the resulting window, search for *Azure.Identity*. Select the appropriate result, and select **Install**.



3. Add the following using statement to the *Program.cs** file:

```
C#

using Azure.Identity;
```

4. Add the following code to the *Program.cs* file. When the code is run on your local workstation during development, it will use the developer credentials of the prioritized tool you're logged into to authenticate to Azure, such as the Azure CLI or Visual Studio.

```
C#

TokenCredential cred = new DefaultAzureCredential();
ArmClient client = new ArmClient(cred);
```

Create a storage task

1. Pick a resource group in your subscription where you'd like to create a storage task. Then, get the storage task collection of that resource group.

   - Replace the `<subscription-id>` placeholder value in this example with the ID of your subscription.
   - Replace the `<resource-group>` placeholder value in this example with the name of your resource group.

```
C#

string subscriptionId = "<subscription-id>";
string resourceGroupName = "<resource-group>";
```

```
 ResourceIdentifier resourceGroupResourceId =
     ResourceGroupResource.CreateResourceIdentifier(subscriptionId,
resourceGroupName);

 ResourceGroupResource resourceGroupResource =
     client.GetResourceGroupResource(resourceGroupResourceId);

 StorageTaskCollection storageTaskcollection =
resourceGroupResource.GetStorageTasks();
```

2. Define a *condition clause* by using JSON. A condition has a collection of one or more clauses. A clause contains a property, a value, and an operator. In the following JSON, the property is `Name`, the value is `.docx`, and the operator is `endsWith`. This clause allows operations only on Microsoft Word documents.

```
C#

string clause = "[[endsWith(Name, '.docx')]]"
```

   For a complete list of properties and operators, see Storage task conditions.

   > **Tip**
   >
   > You can add multiple conditions to the same string and separate them with a comma.

3. Define the complete condition by including the clause that you previously defined along with two operations. The first operation in this definition sets an immutability policy. The second operation sets a blob index tag in the metadata of the Word document.

```
C#

 StorageTaskIfCondition condition = new(clause, new
StorageTaskOperationInfo[]
 {
     new
StorageTaskOperationInfo(StorageTaskOperationName.SetBlobImmutabilityPolicy)
     {
         Parameters =
         {
             ["untilDate"] = "2025-10-20T22:30:40",
             ["mode"] = "locked"
         },
         OnSuccess = OnSuccessAction.Continue,
         OnFailure = OnFailureAction.Break,
     },
     new StorageTaskOperationInfo(StorageTaskOperationName.SetBlobTags)
```

```
    {
        Parameters =
        {
            ["tagsetImmutabilityUpdatedBy"] = "StorageTaskQuickstart"
        },
        OnSuccess = OnSuccessAction.Continue,
        OnFailure = OnFailureAction.Break,
    }

});
```

4. Create the storage task and then add it to the storage task collection.

C#

```
StorageTaskProperties storageTaskProperties =
    new(true, "My storage task", new StorageTaskAction(condition));

StorageTaskData storageTaskData = new(
    new AzureLocation("westus"),
    new ManagedServiceIdentity("SystemAssigned"),
    storageTaskProperties);

var storageTaskResult = (storageTaskcollection.CreateOrUpdate
    (WaitUntil.Completed, "mystoragetask", storageTaskData)).Value;

Console.WriteLine($"Succeeded on id: {storageTaskResult.Data.Id}");
```

Create an assignment

To use a storage task, you must create a *storage task assignment*. The assignment is saved as part of the storage account resource instance, and defines among other settings, a subset of objects to target, when and how often a task runs against those objects, and where the execution reports are stored.

1. Specify prefix filters, run frequency, start times by creating an execution context. The following example targets the `mycontainer` container and is scheduled to run `10` minutes from the present time.

C#

```
ExecutionTriggerParameters executionTriggerParameters = new()
{
    StartOn = DateTime.Now.AddMinutes(10).ToUniversalTime()
};

ExecutionTrigger executionTrigger =
    new(ExecutionTriggerType.RunOnce, executionTriggerParameters);
```

```
StorageTaskAssignmentExecutionContext storageTaskAssignmentExecutionContext =
    new(executionTrigger)
{
    Target = new ExecutionTarget
    {
        Prefix = { "mycontainer/" },
        ExcludePrefix = { },
    }
};
```

2. Create the storage task assignment and then add it to the storage task assignment collection. The following assignment includes the execution context that you created in the previous step. It also specifies the target storage account and is configured to save execution reports to a folder named `storage-tasks-report`.

C#

```
string accountName = "mystorageaccount";

ResourceIdentifier storageAccountResourceId =
    StorageAccountResource.CreateResourceIdentifier
    (subscriptionId, resourceGroupName, accountName);

StorageAccountResource storageAccount =
    client.GetStorageAccountResource(storageAccountResourceId);

StorageTaskAssignmentCollection storageTaskAssignmentcollection =
    storageAccount.GetStorageTaskAssignments();

StorageTaskAssignmentProperties storageTaskAssignmentProperties = new(
    new ResourceIdentifier(storageTaskResult.Data.Id.ToString()), true,
    "My Storage task assignment",
    storageTaskAssignmentExecutionContext,
    new("storage-tasks-report"));

// Create and execute the storage task assignment
storageTaskAssignmentcollection.CreateOrUpdate(
WaitUntil.Started,
"myStorageTaskAssignment",
new StorageTaskAssignmentData(storageTaskAssignmentProperties));
```

3. Give the storage task permission to perform operations on the target storage account. Assign the role of `Storage Blob Data Owner` to the system-assigned managed identity of the storage task.

C#

```
var roleDefId = $"/subscriptions/" + subscriptionId +
```

```
"/providers/Microsoft.Authorization/roleDefinitions/<b7e6dc6d-f1e8-4753-8033-
0f276bb0955b>";

var operationContent = new RoleAssignmentCreateOrUpdateContent(
    new ResourceIdentifier(roleDefId),
    storageTaskResult.Data.Identity.PrincipalId ?? throw new
InvalidOperationException("PrincipalId is null"))
{
    PrincipalType = RoleManagementPrincipalType.ServicePrincipal
};

ResourceIdentifier roleAssignmentResourceId =
RoleAssignmentResource.CreateResourceIdentifier(storageAccount.Id,
Guid.NewGuid().ToString());

RoleAssignmentResource roleAssignment =
client.GetRoleAssignmentResource(roleAssignmentResourceId);

ArmOperation <RoleAssignmentResource> lro = await
roleAssignment.UpdateAsync(WaitUntil.Completed, operationContent);
RoleAssignmentResource result = lro.Value;
RoleAssignmentData resourceData = result.Data;
Console.WriteLine($"Succeeded on id: {resourceData.Id}");
```

Run the code

If you're using Visual Studio, press F5 to build and run the code and interact with the console app. If you're using the .NET CLI, navigate to your application directory, then build and run the application.

Console

```
dotnet build
```

Console

```
dotnet run
```

View the results of a task run

After the task completes running, get a run report summary for each assignment.

C#

```
    ResourceIdentifier storageTaskResourceId =
StorageTaskResource.CreateResourceIdentifier
```

```
        (subscriptionId, resourceGroupName, "mystoragetask");
    StorageTaskResource storageTask =
client.GetStorageTaskResource(storageTaskResourceId);
    // invoke the operation and iterate over the result
    await foreach
(Azure.ResourceManager.StorageActions.Models.StorageTaskReportInstance item in
storageTask.GetStorageTasksReportsAsync())
    {
        Console.WriteLine($"Succeeded: {item.Properties.SummaryReportPath}");
    }
```

The `SummaryReportPath` field of each report summary contains a path to a detailed report. That report contains comma-separated list of the container, the blob, and the operation performed along with a status.

Clean up resources

Remove all of the assets you've created. The easiest way to remove the assets is to delete the resource group. Removing the resource group also deletes all resources included within the group. In the following example, removing the resource group removes the storage account and the resource group itself.

PowerShell

```
Remove-AzResourceGroup -Name $ResourceGroup
```

Next step

What is Azure Storage Actions

Best practices for using storage tasks

09/15/2025

This article provides you with best practice guidelines for using storage tasks.

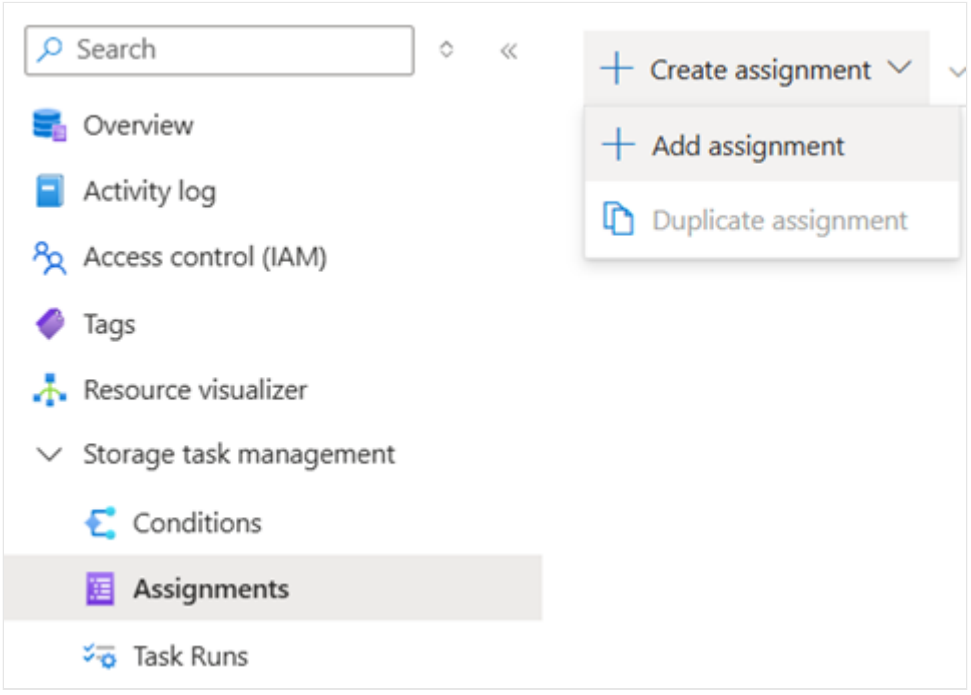
Composition

Design conditions that can apply to multiple storage accounts. Consider enabling blob soft delete before using delete operations in your conditions, and carefully review the impact of condition groupings. This section describes each of these recommendations.

Define a single storage task condition that can be applied to multiple storage accounts simultaneously

By making conditions reusable, you'll eliminate the need to create individual storage task conditions for each storage account and simplifies the management of a large number of storage accounts.

To utilize this feature, create one storage task condition to scan blobs across your storage accounts and assign it to each account. This can be accomplished by adding multiple storage task assignments for a single storage task in the Azure portal using the **Assignments** menu option of a storage task.



After all these task assignments are enabled, the storage task execution will operate on the blobs in their respective accounts concurrently, thus enabling horizontal scaling.

Consider enabling blob soft delete in a storage account before using the delete operation

For enhanced data protection, Microsoft recommends enabling blob soft delete on your storage account. Blob soft delete provides an extra layer of retention and peace of mind by allowing you to recover blobs that are accidentally deleted. By enabling blob soft delete, you can undelete blobs within the retention period, minimizing the risk of data loss due to inadvertent deletions. If blobs are accidentally deleted on a soft-deleted storage account, an undelete operation in a storage task can be used to recover the blob.

Understand the impact of assigning a storage task to an account that has a lifecycle management policy enabled

A delete operation might fail if a lifecycle management policy attempts to delete the same blob. Therefore, if a storage task contains a delete operation and the task is assigned to an account that has an active lifecycle management policy that defines a delete action, you might experience high failure rates if they target the same set of blobs.

Understand the impact of storage task condition groupings

Verify that the way you group conditions leads to the desired result. Incorrect grouping might result in unexpected operations. Test the grouped conditions thoroughly using the preview capability. Ensure that the conditions match the blobs as anticipated.

Validation

Validate conditions as you compose them, and once again before you enable a storage task assignment. This section describes each of these recommendations.

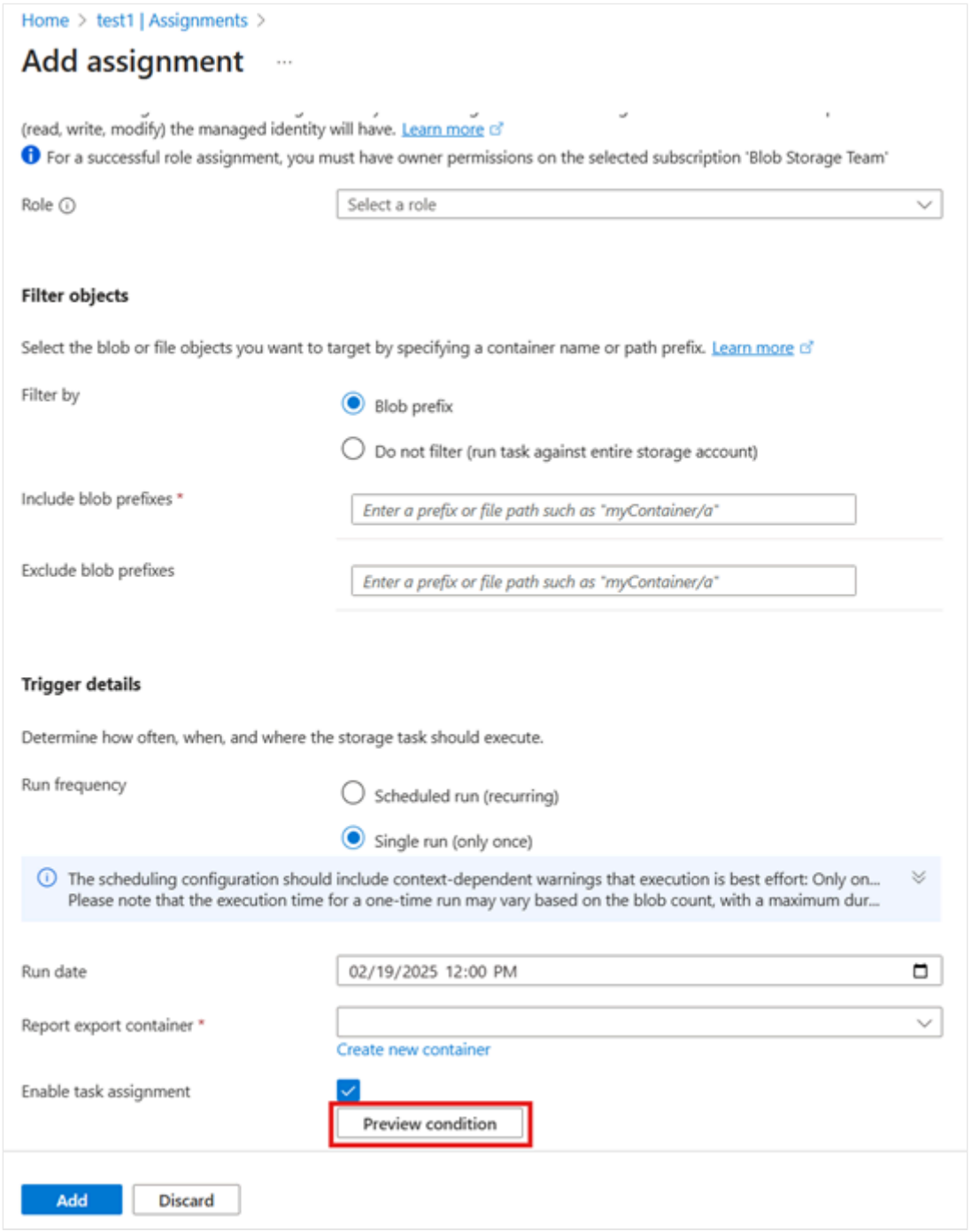
Validate conditions as you compose them

You should validate each condition that you compose by using the condition preview feature. This feature provides an interactive experience that shows which of the selected blobs meet the condition predicate that you configured. The condition preview experience doesn't make any changes to the data. You can validate the condition against various sets of blobs by specifying different storage accounts or blob path prefixes. Incorrectly composed conditions can adversely affect your data, making this step crucial to ensure the condition is composed correctly.

See Preview the effect of conditions

Validate conditions before you commit a storage task assignment

The condition preview feature also appears in the **Add Assignment** pane. Before committing an assignment, use this feature to verify that the storage task operates on the correct set of blobs in the storage account. If the storage task is used in other storage account assignments, no unintended blobs are impacted.



Scale and performance

Enable storage tasks one at a time. Apply techniques to optimize conditions and task assignment scheduling.

Enable a single storage task assignment at a time

Storage Actions currently supports the execution of one storage task assignment at a time on a storage account. If two storage tasks are assigned to an account and enabled simultaneously, the first task is executed while the second task is queued until the first task completes. This applies to both single-run and recurrent scheduled task assignments.

For scheduled task assignments, if the previous task iteration is still in progress, new iterations are skipped. The next scheduled task will only run at its designated trigger time after the previous task completes. When scheduling recurrent tasks, consider the scale implications where task assignments applied to large storage accounts might take longer to complete. Therefore, it's advisable to schedule them such that each task run can finish before the next iteration to prevent skipping subsequent iterations.

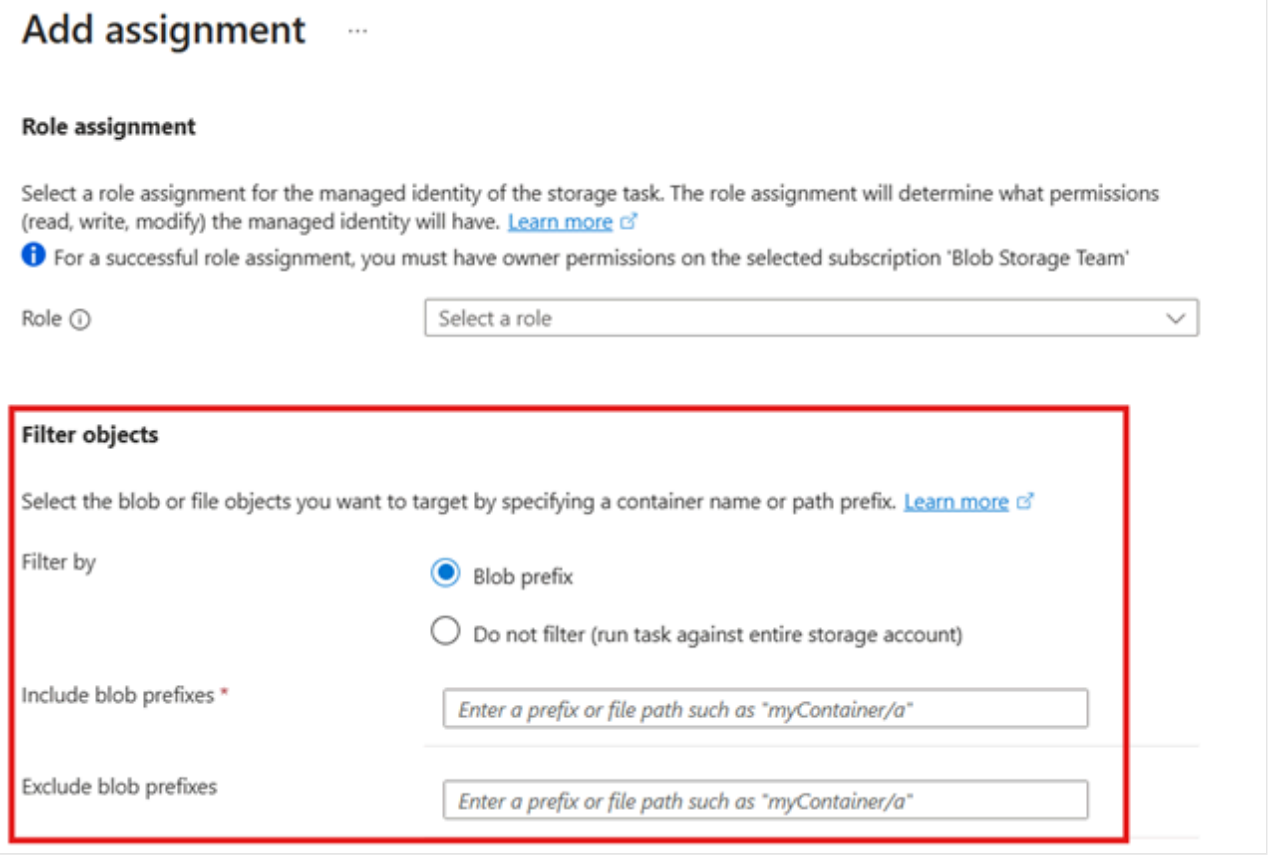
For single-run task assignments, if a parallel task is already in progress, the new task execution is deferred for 60 minutes plus extra random minutes before attempting again. In general, to avoid confusion regarding which task assignment is being executed, Microsoft recommends enabling only one task assignment at a time.

Workarounds on scale limits

Storage Actions have defined scale limits. See Scale limits

To optimize the management of scale limits, consider implementing the following workarounds:

1. **Task segmentation by prefix**: Instead of assigning a single task to process all blobs in a storage account, create multiple tasks, each responsible for a specific filtered subset of blobs based on their prefixes. This segmentation approach distributes the workload more evenly and helps stay within scale limits. You can add filters during task assignment as shown:



2. **Staggered scheduling**: Schedule tasks to run at different times, especially for large-scale operations. By staggering the execution times, you avoid concurrent tasks that could breach concurrency limits and cause task execution contention.

3. **Incremental processing**: Break down large tasks into smaller, incremental steps. This method ensures that each task segment can complete within the given limits, reducing the risk of incomplete operations.

4. **Monitoring and adjust**: Regularly monitor task performance and progress. Adjust the task conditions / prefixes and schedules as necessary to ensure efficient processing within scale limits.

By employing these strategies, you can effectively manage and work around the imposed scale limits, ensuring smooth and efficient task executions.

Reliability

Storage Actions perform more reliably in accounts with geo-redundant storage (GRS), or geo-zone-redundant storage (GZRS) configurations.

Use geo redundancy for business continuity

Storage accounts with GRS and GZRS replicate data to a secondary region in the event of storage account failovers. The business continuity of storage actions significantly depends on

the redundancy configuration of the target storage account. Storage accounts that are configured with geo-redundancy benefit from an automated failover process. This automatic management ensures that future task assignment run iterations, whether single or recurrent, executes in the secondary region without issues. However, storage tasks that were in progress at the time of failover might encounter failures. New storage tasks and storage task assignments continue to function as expected.

Consistent monitoring of the storage account is crucial. With a failover, you should thoroughly review task reporting and monitoring to verify the successful completion of all blob operations and to identify any discrepancies that need attention.

Monitoring

Periodically monitor storage task executions and avoid deleting the report container where task execution reports are stored.

Monitor tasks periodically

You should periodically monitor the storage task execution to ensure that tasks are running as expected. Review task reports, metrics, monitoring dashboards. Check for any errors, and verify that the tasks are completing within the expected timeframes.

Ensure the result report container isn't deleted

Storage Actions generate detailed reports in CSV format which are written into the result reporting container configured during task assignment. These reports provide insights into the task execution operations where each row line in the CSV file includes information about the operations performed, the status of each operation, and any errors encountered. It's important to make sure that the result reporting container that is configured during task assignment isn't deleted from the storage account during the task execution. If the result reporting container is deleted during the task run, the task execution can fail.

Storage Actions generate detailed reports in CSV format, which are written into the result reporting container that is configured during task assignment. These reports provide insights into task execution operations, with each row in the CSV file including information about the operations performed, the status of each operation, and any errors encountered. It's important to ensure that the result reporting container isn't deleted from the storage account during task execution. If the result reporting container is deleted during the task run, the task execution can fail.

Storage Actions lifecycle

Managing tasks using a central library subscription

To efficiently manage your tasks and task assignments, Consider using a central subscription to contain a library of storage tasks. This approach allows you to assign these tasks to numerous storage accounts across different subscriptions and regions simultaneously, without having to configure them individually for each region or subscription. By centralizing your task management, you can streamline the process, reduce administrative overhead, and ensure consistency in task execution across your entire Azure environment.

See also

- Azure Storage Actions overview
- Azure RBAC best practices
- AKS best practices

Storage task scenarios

Article • 05/07/2025

Large data lakes can have thousands of data sets with different object types that need various processing methods. Depending on their attributes, individual objects in a blob container might need specific retention or expiry periods, different tiering transitions, or tagging with different labels. With Azure Storage Actions, you can define tasks to scan billions of blobs, examining each one based on properties like file extension, naming pattern, index tags, blob metadata, or system properties such as creation time, content type, and blob tier. This approach simplifies many recurring or one-off use cases. This article describes scenarios where Storage Actions have been applied or could be applied.

Managing retention and expiry with object tags

A financial services agency uses Azure Blob Storage to ingest customer service call recordings. These recordings have blob tags that indicate if a trading order was placed or account information was updated. The retention requirements for these recordings vary based on the call type. With Azure Storage Actions, they can now define a task that automatically manages the retention and expiry durations of the ingested recordings using a combination of blob tags and creation time.

Managing data protection in datasets

A leading travel services company uses blob versioning and snapshots, but their datasets have different protection needs. Sensitive data requires strict version history, while others don't. Keeping extensive version and snapshot history for all datasets is too expensive. With Azure Storage Actions, they can now use metadata and tags to manage the retention and lifecycle of versions and snapshots more flexibly.

Cost optimization based on naming patterns and file types

Many Azure Storage customers need to manage the tiering, expiry, and retention of blobs based on path-prefix, naming conventions, or file type. These attributes can be combined with blob properties like size, creation time, last modified or accessed times, access tier, version counts, and more to process the objects as needed.

One-off processing of blobs at scale

Azure Storage Actions can be used for one-time processing of billions of objects, in addition to ongoing data management operations. For example, you can define tasks to rehydrate a large dataset from the archive tier, reset tags on part of a dataset when restarting an analytic pipeline, initialize blob tags for a new or updated process, or clean up redundant and outdated datasets.

See also

- Storage task operations
- Define conditions and operations

Azure roles for storage tasks

Article • 05/07/2025

This article describes the least privileged built-in Azure roles or RBAC actions required to read, update, or delete a storage task and to view task assignments.

Permission to manage a storage task

You must assign a role to any security principal in your organization that needs access to the storage task. To learn how to assign an Azure role, see Assign Azure roles using the Azure portal.

While the **Contributor** role provides all of the permissions necessary to manage a storage task, the least privileged built-in role is the **Storage Actions Contributor** role.

If you prefer to use a custom role, make sure that your role contains all of the necessary RBAC actions. Use the following table as a guide.

Expand table

Permission level	RBAC actions for custom roles
List and read storage tasks	Microsoft.StorageActions/storageTasks/read
Create and update storage tasks	Microsoft.StorageActions/storageTasks/write
Delete storage tasks	Microsoft.StorageActions/storageTasks/delete
List storage task assignments	Microsoft.StorageActions/storageTasks/storageTaskAssignments/read
List storage task run reports	Microsoft.StorageActions/storageTasks/reports/read
Preview storage task conditions	Microsoft.StorageActions/locations/previewActions/action
Move a storage task to another resource group	Microsoft.Resources/subscriptions/resourceGroups/moveResources/action Microsoft.Resources/subscriptions/resourceGroups/write

See also

- Azure roles required to assign tasks

Storage task conditions

Article • 05/07/2025

A storage task contains a set of conditions and operations. This article describes the JSON format of a condition. Understanding that format is important if you plan to create a storage task by using a tool other than the Azure portal (For example: Azure PowerShell, or Azure CLI). This article also lists the properties and operators that you can use to compose the clauses of a condition.

This article focuses on **conditions**. To learn more about **operations**, see Storage task operations.

Condition format

A condition a collection of one or more *clauses*. Each clause contains a *property*, a *value*, and an *operator*. When the storage task runs, it uses the operator to compare a property with a value to determine whether a clause is met by the target object. In a clause, the **operator** always appears first followed by the **property**, and then the **value**. The following image shows how each element is positioned in the expression.

```
[[[<operator>(<property>,<value>)]]]
```

The following clause allows operations only on Microsoft Word documents. This clause targets all documents that end with the file extension `.docx`. Therefore, the operator is `endsWith`, the property is `Name`, and the value is `.docx`.

JSON

```
{
   "condition": "[[[endsWith(Name, '.docx')]]]"
}
```

You can also apply a *not* operator before the operator in a condition clause. The *not* operator is a special operator that you can position before any operator to give the opposite result of a clause. The following clause allows operations on any blob that **is not** a Microsoft Word document.

JSON

```
{
   "condition": "[[not(endsWith(Name, '*.docx'))]]"
```

```
}
```

For a complete list of operator and property names, see the Supported operators and Supported properties section of this article.

Multiple clauses in a condition

A condition can contain multiple clauses separated by a comma along with either the string `and` or `or`. The string `and` targets objects that meet the criteria in all clauses in the condition while `or` targets objects that meet the criterion in any of the clauses in the condition. The following image shows the position of the `and` and `or` string along with two clauses.

The following JSON shows a condition that contains two clauses. Because the `and` string is used in this expression, both clauses must evaluate to `true` before an operation is performed on the object.

JSON

```
{
"condition": "[[and(endsWith(Name, '.docx'), equals(Tags.Value[readyForLegalHold], 'Yes'))]]"
}
```

Groups of conditions

Grouped clauses operate as a single unit separate from the rest of the clauses. Grouping clauses is similar to putting parentheses around a mathematical equation or logic expression. The `and` or `or` string for the first clause in the group applies to the whole group.

The following image shows two clauses grouped together.

The following condition allows operations only on Microsoft Word documents where the `readyForLegalHold` tag of the document is set to a value of `Yes`. Operations are also performed on objects that are greater than 100 bytes even if the other two conditions aren't true.

JSON

```
{
"condition": "[[or(and(endsWith(Name, '*.docx'),
equals(Tags.Value[readyForLegalHold], 'Yes')), greater(Content-Length, '100'))]]"
}
```

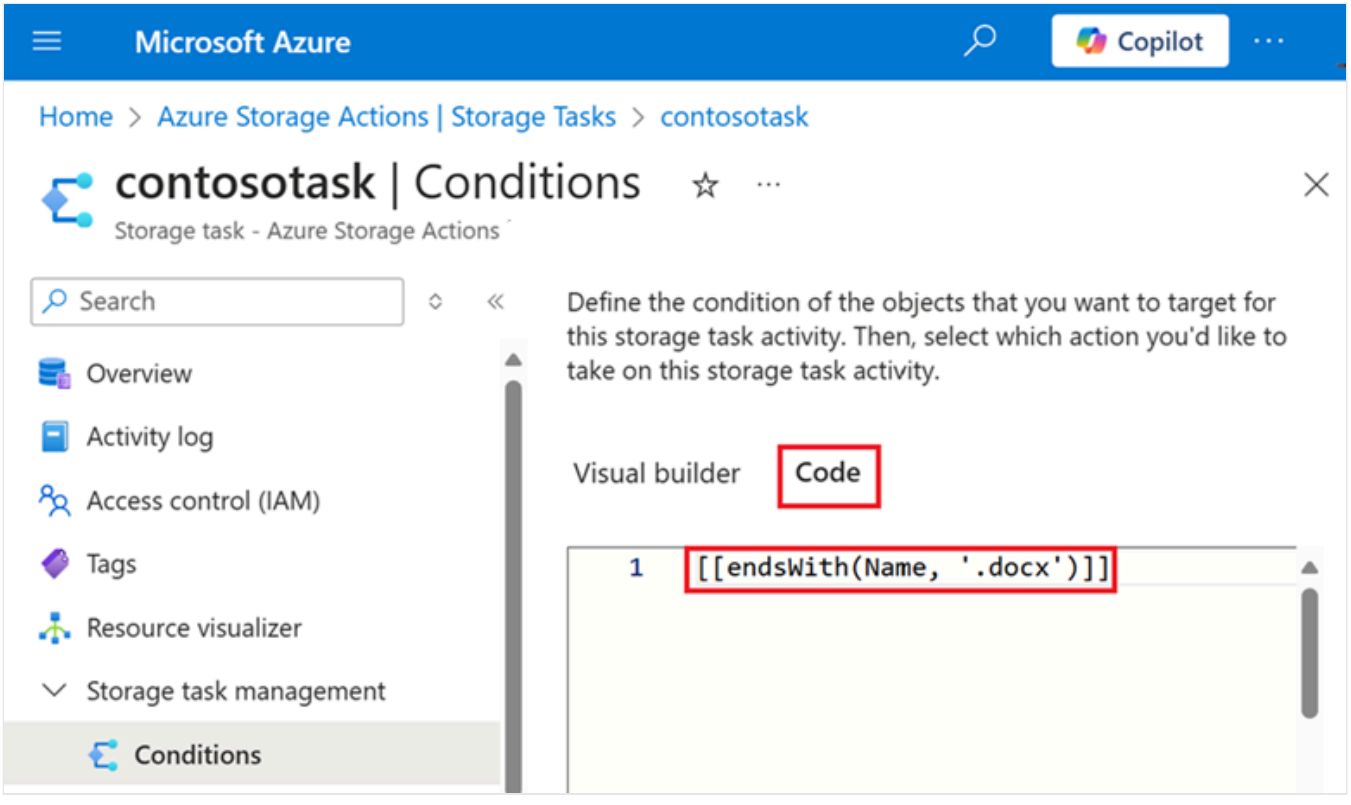
You can apply a *not* operator to a group to test for the opposite result of a group of clauses. The following condition allows operations only on blobs that **are not** Microsoft Word documents where the `readyForLegalHold` tag of the blob is set to a value of `Yes`. Operations are also performed on objects that are greater than 100 bytes even if the other two conditions aren't true.

JSON

```
{
"condition": "[[or(not(and(endsWith(Name, '*.docx'),
equals(Tags.Value[readyForLegalHold], 'Yes'))), greater(Content-Length, '100'))]]"
}
```

Code view in the Azure portal

The visual editor available in the Azure portal, can generate the JSON of a condition for you. You can define your conditions by using the editor, and then obtain the JSON expression by opening **Code** tab. This approach can be useful when creating complicated sets of conditions as JSON expressions can become large, unwieldy, and difficult to create by hand. The following image shows the **Code** tab in the visual editor.



To learn more about the visual editor, see Define storage task conditions and operations.

Condition preview

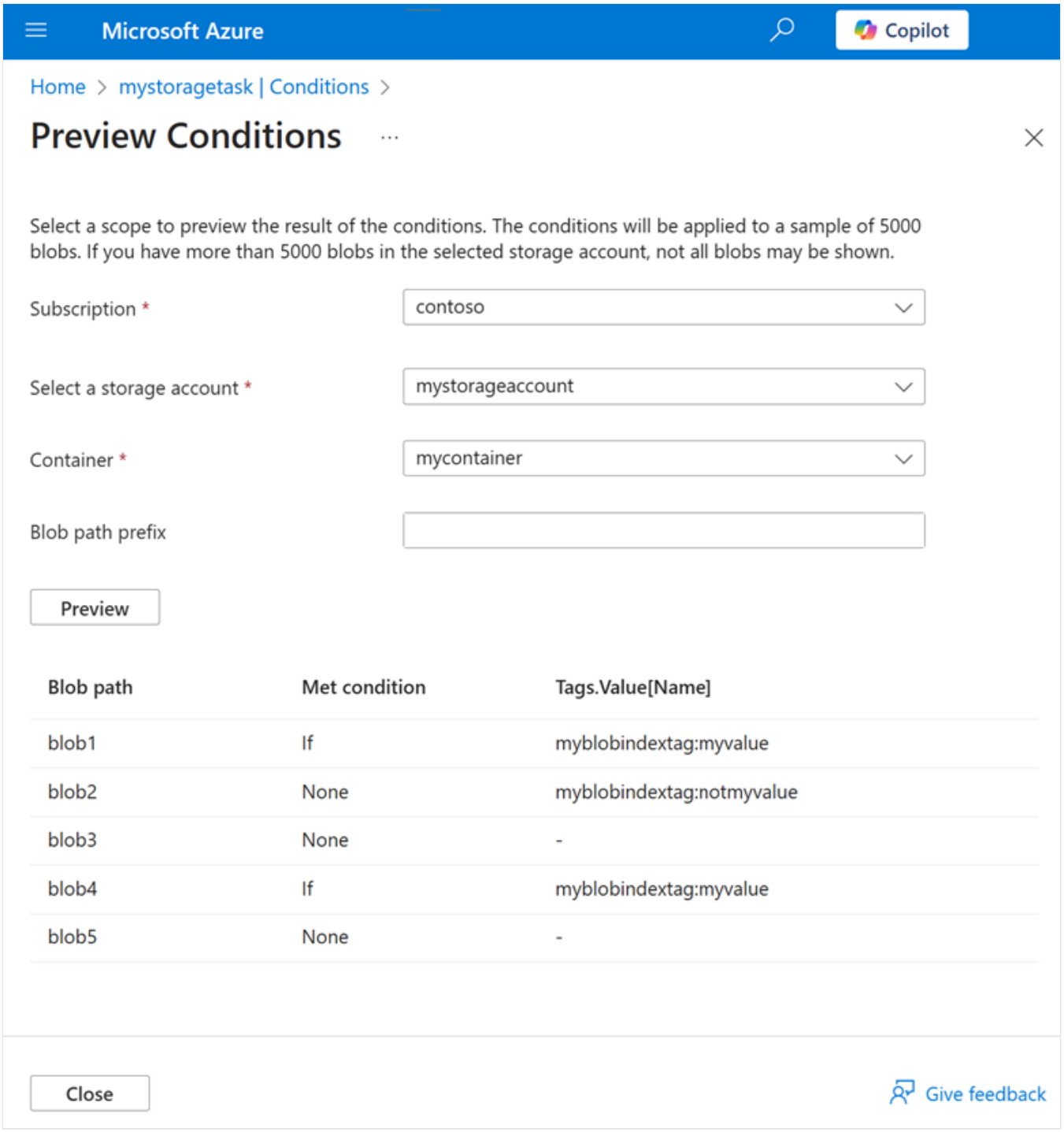
You can view a list of blobs that would be impacted by the conditions that you've defined. That way, you can find issues and optimize conditions before applying them to production data. A preview doesn't make changes to the objects in a target storage account so it's safe to apply to test against production data.

While condition preview is available in PowerShell, Azure CLI, and SDK environments, the easiest way to preview the effect of conditions is by using the **Preview Conditions** window in the Azure portal. You can open this window was you define conditions and as you assign storage tasks.

To preview the effect of conditions, you must specify a target subscription, storage account, and container. Because a can only show up to 5,000 blobs, you can also specify a prefix to narrow the list.

> **Note**
>
> You can't use wildcard characters in the blob prefix.

The following image shows an example of a preview result in the Azure portal.



The preview result appears in a table that shows objects which meet the condition along with objects that didn't meet the condition. You can sort by field that appears in the table.

If conditions refer to properties that don't exist in the target storage account, an error appears. For example, blob index tags aren't available for accounts that have a hierarchical namespace. If a clause in a condition refers to blob index tags, a validation error appears.

Supported properties

The following table shows the properties that you can use to compose each clause of a condition. A clause can contain string, boolean, numeric, and date and time properties.

Expand table

String	Date and time[3]	Numeric	Boolean
AccessTier[1]	AccessTierChangeTime	Content-Length	Deleted
Metadata.Value	Creation-Time	TagCount	IsCurrentVersion
Name	DeletedTime		
BlobType[2]	LastAccessTime		
Container.Metadata.Value[Name]	Last-Modified		
Container.Name			
Container.Metadata.Value[Name]			
Container.Name			
Tags.Value[Name]			
VersionId			

[1] Allowed values are `Hot`, `Cool`, or `Archive`.

[2] Allowed values are `BlockBlob`, `PageBlob`, or `AppendBlob`

[3] Can be set to a specific time or to a metadata value dynamically obtained from objects. See Reference a value from object metadata.

Supported operators

The following table shows the operators that you can use in a clause to evaluate the value of each type of property.

Expand table

String	Date and time	Numeric	Boolean
contains	equals	equals	equals
empty	greater	greater	
equals	greaterOrEquals	greaterOrEquals	
endsWith	less	less	
length	lessOrEquals	lessOrEquals	

String	Date and time	Numeric	Boolean
startsWith	addToTime		
Matches			

The **not** operator is a special operator that you can position before any of the operators that appear in this table to give the opposite result of conditional clause, also called the negative result.

See also

- Storage task operations
- Define conditions and operations

Storage task operations

Article • 05/07/2025

A storage task contains a set of conditions and operations. An operation is an action that a storage task performs on each object that meets the requirements of each condition. This article describes the JSON format of a storage task operation. Understanding that format is important if you plan to create a storage task by using a tool other than the Azure portal (For example: Azure PowerShell, or Azure CLI). This article also lists the operations, operation parameters, and the allowable values of each parameter.

This article focuses on **operations**. To learn more about **conditions**, see Storage task conditions.

Operation format

An operation has a name along with zero, one, or multiple parameters. The following image shows how these elements appear for an operation in the JSON template of a storage task.

```
[{
    "name": <name>,
    "parameters": {
        <parameter-name>: <value>,
        <parameter-name>: <value>,
    },
    "onSuccess": <action>,
    "onFailure": <action>
}]
```

The following table describes each element.

Expand table

Element	Description
`name`	The name of the operation.[1]
`parameters`	A collection of one or more parameters. Each parameter has parameter name and a parameter value.[1]
`onSuccess`	The action to take when the operation is successful for an object.
`onFailure`	The action to take when the operation fails for an object.

[1] For a complete list of operation names, operation parameters, and parameter values, see the Supported operations section of this article.

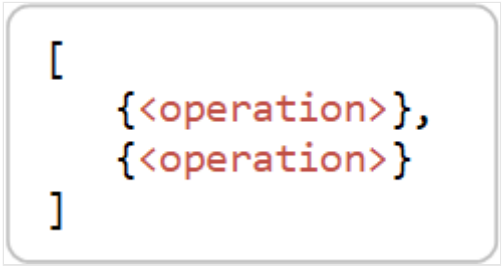
The following operation applies a time-based immutability policy to the object.

JSON

```
{
    "operations": [
        {
            "name": "SetBlobImmutabilityPolicy",
            "parameters": {
                "untilDate": "2024-11-15T21:54:22",
                "mode": "locked"
            },
            "onSuccess": "continue",
            "onFailure": "break"
        }
    ]
}
```

Multiple operations

Separate multiple operations by using a comma. The following image shows the position of two operations in list of operations.



The following JSON shows two operations separate by a comma.

JSON

```
"operations": [
    {
        "name": "SetBlobImmutabilityPolicy",
        "parameters": {
            "untilDate": "2024-11-15T21:54:22",
            "mode": "locked"
        },
        "onSuccess": "continue",
        "onFailure": "break"
    },
    {
        "name": "SetBlobTags",
```

```
        "parameters": {
            "ImmutabilityUpdatedBy": "contosoStorageTask"
        },
        "onSuccess": "continue",
        "onFailure": "break"
    }
]
```

Supported operations

The following table shows the supported operations, parameters, and parameter values:

Expand table

Operation	Parameters	Values
SetBlobTier	tier	Hot | Cool | Archive
SetBlobExpiry	expiryTime, expiryOption	(expiryTime): Number of milliseconds (expiryOption): Absolute | NeverExpire | RelativeToCreation | RelativeToNow
DeleteBlob	None	None
UndeleteBlob	None	None
SetBlobTags	Tag name[1]	Tag value
SetBlobImmutabilityPolicy	untilDate, mode	(untilDate): DateTime of when policy ends (mode): locked | unlocked
SetBlobLegalHold	legalHold	true | false

[1] The name of this parameter is the name of the tag.

See also

- Storage task conditions
- Define conditions and operations

Storage task assignment

Article • 05/07/2025

To use a storage task, you must create a *storage task assignment*. The assignment is saved as part of the storage account resource instance, and defines among other settings, a subset of objects to target, when and how often a task runs against those objects, and where the execution reports are stored.

To create an assignment, your identity must be assigned the appropriate Azure built-in role or a custom role with the appropriate RBAC actions. See Azure roles required to assign tasks. To learn how to create a storage task assignment, see Create and manage a storage task assignment.

Assignment settings

The following table describes the configuration settings of a storage task assignment.

> **Note**
>
> The names that appear in the following table appear in the **Add Assignment** page of the Azure portal. If plan to create an assignment by using REST, an SDK, PowerShell, or Azure CLI, see the appropriate reference content set to obtain the names of specific properties used to configure each setting.

Expand table

Setting	Required or optional	Description
Subscription	Required	The subscription of the storage account that you want to add to this assignment.
Storage account name	Required	The storage account that you want to add to this assignment. You must be an owner of the storage account. This field appears only if you create the assignment in the context of a storage task.
Storage task name	Required	The storage task to which you would like to assign your storage account. This field appears only if you create the assignment in the context of a storage account.
Storage task assignment name	Required	The name of the assignment. Assignment names must be between 2 and 62 characters in length and might contain only letters and numbers.

Setting	Required or optional	Description
Filter by	Required	Option to either filter objects by using a prefix or to run the task against the entire storage account.
Include blob prefixes	Optional	The string prefix that is used to narrow the scope of blobs that are evaluated by the task. This field is required only if you choose to filter by using a blob prefix.
Exclude blob prefixes	Optional	A string prefix that is used to exclude blobs that are evaluated by the task.
Run frequency	Required	Option to either run the task one time or multiple times.
Start from	Required	The date and time to begin running the task. Applicable only when scheduling a task to run multiple times.
End by	Required	The date and time stop running the task. Applicable only when scheduling a task to run multiple times.
Repeat very (in days)	Required	The interval in days between each run. Applicable only when scheduling a task to run multiple times.
Report export container	Required	The container where task execution reports are stored.

Storage task authorization

As part of the assignment process, you'll assign a role to the managed identity associated with the storage task. By default, a system-assigned managed identity is created when the storage task is provisioned. However, the user that creates the storage task can optionally associate a user-assigned managed identity with the storage task. The type of managed identity that is associated with the storage task can't be changed after the storage task is provisioned.

When assigning a role, you must choose an Azure built-in or custom role that has the permission necessary to perform the operations defined in a storage task upon the target storage account. See Permission for a task to perform operations.

After you save the assignment, the managed identity is validated to ensure that has the correct permissions to perform the tasks defined in the storage task. If you use the Azure portal to create the assignment, this validation step also occurs after you assign the role to the managed identity.

Network access to storage accounts

You must grant access to trusted Azure services in the network settings of each target storage account. To learn more, see Grant access to trusted Azure services.

See also

- Storage task operations
- Define conditions and operations

Azure roles required to assign tasks

Article • 05/07/2025

This article describes the least privileged built-in Azure roles or RBAC actions required to manage a storage task assignment.

Permission to manage storage task assignments

To create an assignment, your identity must be assigned either the Storage Blob Data Owner role or a custom role that contains the following RBAC actions:

- Microsoft.Authorization/roleAssignments/write
- Microsoft.Authorization/roleAssignments/delete
- Microsoft.Storage/storageAccounts/reports/read
- Microsoft.Storage/storageAccounts/read
- Microsoft.Storage/storageAccounts/blobServices/read
- Microsoft.Storage/storageAccounts/storageTaskAssignments/read
- Microsoft.Storage/storageAccounts/storageTaskAssignments/write
- Microsoft.Storage/storageAccounts/storageTaskAssignments/delete
- Microsoft.Storage/storageAccounts/storageTaskAssignments/reports/read

To learn how to create a custom role, see Azure custom roles.

Permission for a task to perform operations

As you create an assignment, you must choose an Azure built-in or custom role that has the permission necessary to perform the specified operations on the target storage account or storage account container. That role is assigned to the managed identity of the storage task. You can choose only roles that are assigned to your user identity.

The Storage Blob Data Owner role provides all of the permissions necessary for a storage task to perform all data operations. If you prefer to use a custom role, you must make sure that your role contains the RBAC actions necessary to perform the operations. The following table shows the RBAC actions required by each operation.

Expand table

Permission	RBAC actions for a custom role
SetBlobTier	Microsoft.Storage/storageAccounts/blobServices/containers/blobs/read Microsoft.Storage/storageAccounts/blobServices/containers/blobs/write

Permission	RBAC actions for a custom role
	Microsoft.Storage/storageAccounts/blobServices/containers/blobs/tags/read Microsoft.Storage/storageAccounts/blobServices/containers/blobs/tags/write
SetBlobExpiry	Microsoft.Storage/storageAccounts/blobServices/containers/blobs/read Microsoft.Storage/storageAccounts/blobServices/containers/blobs/write Microsoft.Storage/storageAccounts/blobServices/containers/blobs/tags/read Microsoft.Storage/storageAccounts/blobServices/containers/blobs/tags/write
SetBlobTags	Microsoft.Storage/storageAccounts/blobServices/containers/blobs/read Microsoft.Storage/storageAccounts/blobServices/containers/blobs/write Microsoft.Storage/storageAccounts/blobServices/containers/blobs/tags/read Microsoft.Storage/storageAccounts/blobServices/containers/blobs/tags/write
SetBlobImmutabilityPolicy	Microsoft.Storage/storageAccounts/blobServices/containers/blobs/read Microsoft.Storage/storageAccounts/blobServices/containers/blobs/write Microsoft.Storage/storageAccounts/blobServices/containers/blobs/tags/read Microsoft.Storage/storageAccounts/blobServices/containers/blobs/tags/write Microsoft.Storage/storageAccounts/blobServices/containers/write
SetBlobLegalHold	Microsoft.Storage/storageAccounts/blobServices/containers/blobs/read Microsoft.Storage/storageAccounts/blobServices/containers/blobs/write Microsoft.Storage/storageAccounts/blobServices/containers/blobs/tags/read Microsoft.Storage/storageAccounts/blobServices/containers/blobs/tags/write Microsoft.Storage/storageAccounts/blobServices/containers/write
DeleteBlob	Microsoft.Storage/storageAccounts/blobServices/containers/blobs/read Microsoft.Storage/storageAccounts/blobServices/containers/blobs/write Microsoft.Storage/storageAccounts/blobServices/containers/blobs/tags/read Microsoft.Storage/storageAccounts/blobServices/containers/blobs/tags/write Microsoft.Storage/storageAccounts/blobServices/containers/blobs/delete
UndeleteBlob	Microsoft.Storage/storageAccounts/blobServices/containers/blobs/read Microsoft.Storage/storageAccounts/blobServices/containers/blobs/write Microsoft.Storage/storageAccounts/blobServices/containers/blobs/tags/read Microsoft.Storage/storageAccounts/blobServices/containers/blobs/tags/write Microsoft.Storage/storageAccounts/blobServices/containers/write

See also

- Create and manage an assignment

Reliability in Azure Storage Actions

07/16/2025

This article describes reliability support in Azure Storage Actions, and covers both intra-regional resiliency with availability zones and cross-region disaster recovery and business continuity. For a more detailed overview of reliability principles in Azure, see Azure reliability.

Azure Storage Actions is a serverless framework that you can use to perform common data operations on millions of objects across multiple storage accounts. The service itself is regional, and it doesn't have SKUs or support for availability zones. However, the control plane of the service automatically supports zone-redundancy. The data plane also may support redundancy depending on whether or not the storage account is running on a zone-redundant configuration.

Availability zone support

Availability zones are physically separate groups of datacenters within each Azure region. When one zone fails, services can fail over to one of the remaining zones.

While the Azure Storage Actions service is regional and doesn't offer SKUs or availability zones, zone redundancy is available from the control plane and conditionally from the data plane:

- **Control plane of the service is zone-redundant.** When a zone is down in one region, the control plane continues to be available. During a zone-down scenario, you can continue to manage task definition and assignment.
- **Data plane (task assignment execution) inherits the zonal properties from the parent storage account.** If the storage account is deployed to a failed zone, then the account becomes unavailable and from customer's perspective, the data plan isn't available. If the storage account is zone redundant, then the account continues to be available, and the service continue to perform operation on the account.

Zone down experience

In a zone-done scenario, the Storage Action service continues to be available. The progress of tasks depends on the availability zone support of storage accounts against which they are running. If the account is not affected by the downed zone, the tasks continue to make progress. Otherwise, the tasks fail.

Zone outage preparation and recovery

The Storage Action service isn't zonal, but the storage account is. If the storage account is affected by a zone outage, storage tasks that are assigned to the account fail. After the zone and storage account become available, scheduled tasks continue to run according to schedule. If the task is configured to run once, you may need to schedule the task to run again.

Cross-region disaster recovery and business continuity

Disaster recovery (DR) refers to practices that organizations use to recover from high-impact events, such as natural disasters or failed deployments that result in downtime and data loss. Regardless of the cause, the best remedy for a disaster is a well-defined and tested DR plan and an application design that actively supports DR. Before you start creating your disaster recovery plan, see Recommendations for designing a disaster recovery strategy.

For DR, Microsoft uses the shared responsibility model. In this model, Microsoft ensures that the baseline infrastructure and platform services are available. However, many Azure services don't automatically replicate data or fall back from a failed region to cross-replicate to another enabled region. For those services, you're responsible for setting up a disaster recovery plan that works for your workload. Most services that run on Azure platform as a service (PaaS) offerings provide features and guidance to support DR. You can use service-specific features to support fast recovery to help develop your DR plan.

Storage accounts with GRS and GZRS replicate data to a secondary region in the event of storage account failovers. The business continuity of storage actions significantly depends on the redundancy configuration of the target storage account. Storage accounts that are configured with geo-redundancy benefit from an automated failover process. This automatic management ensures that future task assignment run iterations, whether single or recurrent, executes in the secondary region without issues. However, storage tasks that were in progress at the time of failover might encounter failures. New storage tasks and storage task assignments continue to function as expected.

Consistent monitoring of the storage account is crucial. With a failover, you should thoroughly review task reporting and monitoring to verify the successful completion of all blob operations and to identify any discrepancies that need attention.

Outage detection, notification, and management

Storage tasks don’t send any notifications when there is an outage in the service itself. It is important that you check the status of the storage task and retry tasks after the service/region recovers.

Next steps

- Tutorial: Create a highly available multi-region app in Azure App Service
- Reliability in Azure

Plan to manage costs for Azure Storage Actions

Article • 05/07/2025

This article describes how you plan for and manage costs for Azure Storage Actions.

Costs for Azure Storage Actions are only a portion of the monthly costs in your Azure bill. Although this article explains how to plan for and manage costs for Azure Storage Actions, you're billed for all Azure services and resources used in your Azure subscription, including the third-party services.

Understand the full billing model for Azure Storage Actions

You're charged the billing meters for Storage Actions as well as the cost of operations performed on storage accounts.

Azure Storage Actions meters

Meters apply only when a storage task assignment is executed. Validating a storage task by using the preview capability is free.

Expand table

Meter	Unit	Description
Task execution instance charge	Per run / per instance	This meter is applied to each storage task assignment execution. If you've schedule an assignment to run repeatedly, then this charge is incurred for each run.
Objects targeted	Per million objects scanned and evaluated / per condition	Objects targeted are determined by the count of objects scanned and evaluated against the specified condition. This is based on the configuration of the task assignment, specifically the count of objects in the storage account under the optional prefixes selected, minus the objects under the excluded prefixes.
Operations performed	Per million operations performed.	Operations performed are counted based on the number of API calls made on objects, including actions such as deleting, setting immutability, tagging, tiering, setting a legal hold, and other operations supported by Storage Actions.

For official prices, see Azure Storage Actions pricing.

At the end of your billing cycle, the charges for each meter are summed. Your bill or invoice shows a section for all Azure Storage Actions costs. There's a separate line item for each meter. These charges appear in the subscription of the storage account where the task assignment is configured.

Azure Storage account meters

Most actions incur charges for operations on the storage account. For example, if a storage task adds an index tag to a blob, then you'll incur the cost of a Set Blob Tags operation on the target storage account. For information about how each Blob Storage operation maps to a price, see Map each REST operation to a price. To learn more about Blob Storage costs, see Plan and manage costs for Azure Blob Storage.

Examples

The examples in this section assume the following sample prices:

Expand table

Storage Action meter	price
Task execution instance charge	$0.25
Objects targeted	$.10
Operations performed	$1.00
Cost of the Set Blob Immutability Policy operation (per 10,000)	$0.0044
Cost of the Set Blob Tier operation (per 10,000)	$0.10

> **Important**
>
> These prices are meant only as examples, and shouldn't be used to calculate your costs. For official prices, see the appropriate pricing pages.

Example: Applying an immutability policy to blobs

In this example, the Set Blob Immutability Policy operation is applied to blobs that meet the conditions defined in the storage task.

A prefix filter narrows the scope of blobs to a single container which contains **1,000,000** blobs, but only **10%** of those blobs meet the conditions of the storage task.

Using sample prices, the following table estimates the cost of this storage task assignment run.

Expand table

Price factor	Value
Cost of executing the storage task assignment	**$0.25**
Cost of targeting an individual blob (price / .10)	$0.0000001
Cost to target 1,000,000 blobs	**$0.10**
Cost of performing a single operation (price / 1,000,000)	$0.000001
Cost to perform operations on 100,000 blobs	**$0.10**
Price of a single Set Blob Immutability Policy operation (price / 10,000)	$0.00000044
Cost to write (100,000 * price of a single write operation)	**$0.044**
Total cost	**$0.49**

> **Important**
>
> These prices are meant only as examples, and shouldn't be used to calculate your costs. For official prices, see the appropriate pricing pages.

Example: Transition blobs to a cooler tier

In this example, the Set Blob Tier operation is applied to blobs that meet the conditions defined in the storage task.

The prefix filter narrows the scope of blobs to a single container which holds **100,000,000** blobs, but only **1%** of those blobs meet the conditions of the storage task.

The following table estimates the cost incurred by a storage task assignment instance that moves blobs from the hot tier to the cool tier.

Expand table

Price factor	Value
Cost of executing the storage task assignment	**$0.25**
Cost of targeting an individual blob (price / .10)	$0.0000001

Price factor	Value
Cost to target 100,000,000 blobs	**$10.00**
Cost of performing a single operation (price / 1,000,000)	$0.000001
Cost to perform operations on 1,000,000 blobs	**$1.00**
Price of a single Set Blob Tier operation (price / 10,000)	$0.00001
Cost to write (1,000,000 * price of a single write operation)	**$10.00**
Total cost	**$21.25**

> **Important**
>
> These prices are meant only as examples, and shouldn't be used to calculate your costs. For official prices, see the appropriate pricing pages.

Using Azure Prepayment with Azure Storage Actions

You can pay for Azure Storage Actions charges with your Azure Prepayment credit. However, you can't use Azure Prepayment credit to pay for charges for third party products and services including those from the Azure Marketplace.

Create budgets

You can create budgets to manage costs and create alerts that automatically notify stakeholders of spending anomalies and overspending risks. Alerts are based on spending compared to budget and cost thresholds. Budgets and alerts are created for Azure subscriptions and resource groups, so they're useful as part of an overall cost monitoring strategy.

Budgets can be created with filters for specific resources or services in Azure if you want more granularity present in your monitoring. Filters help ensure that you don't accidentally create new resources that cost you additional money. For more information about the filter options available when you create a budget, see Group and filter options.

Export cost data

You can also export your cost data to a storage account. This is helpful when you need or others to do additional data analysis for costs. For example, a finance teams can analyze the

data using Excel or Power BI. You can export your costs on a daily, weekly, or monthly schedule and set a custom date range. Exporting cost data is the recommended way to retrieve cost datasets.

Next steps

- Learn more about Azure Blob Storage costs meters, see Plan and manage costs for Azure Blob Storage
- Learn how to optimize your cloud investment with Microsoft Cost Management.
- Learn more about managing costs with cost analysis.
- Learn about how to prevent unexpected costs.

Create a storage task

Article • 05/07/2025

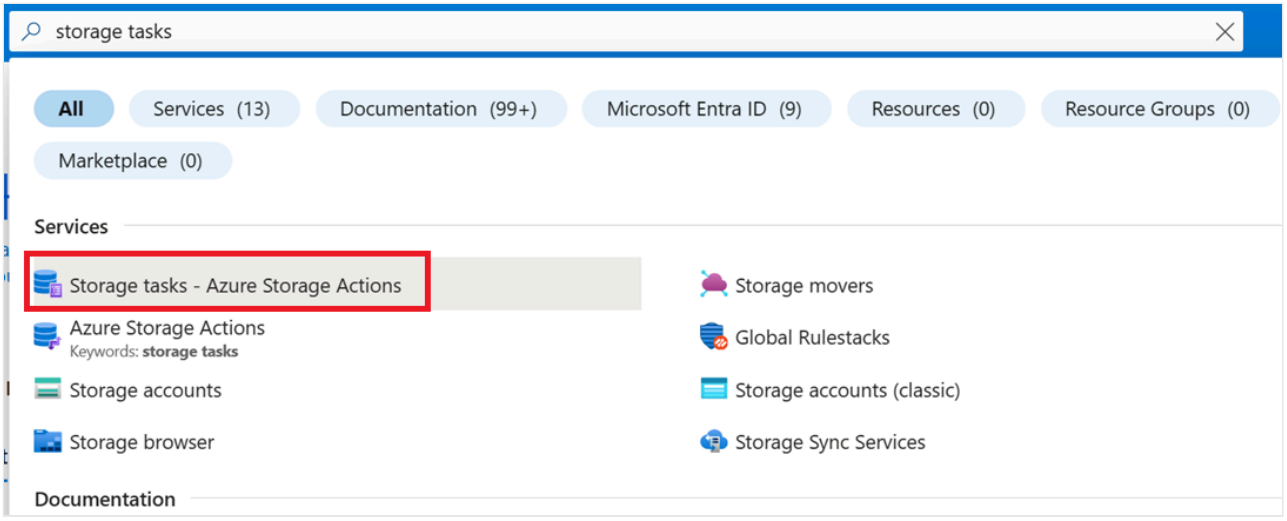
A storage task can perform operations on blobs in an Azure Storage account. As you create a task, you can define the conditions that must be met by each object (container or blob), and the operations to perform on the object. You can also identify one or more Azure Storage account targets. See What are Azure Storage Actions?.

In this how-to article, you'll learn how to create a storage task.

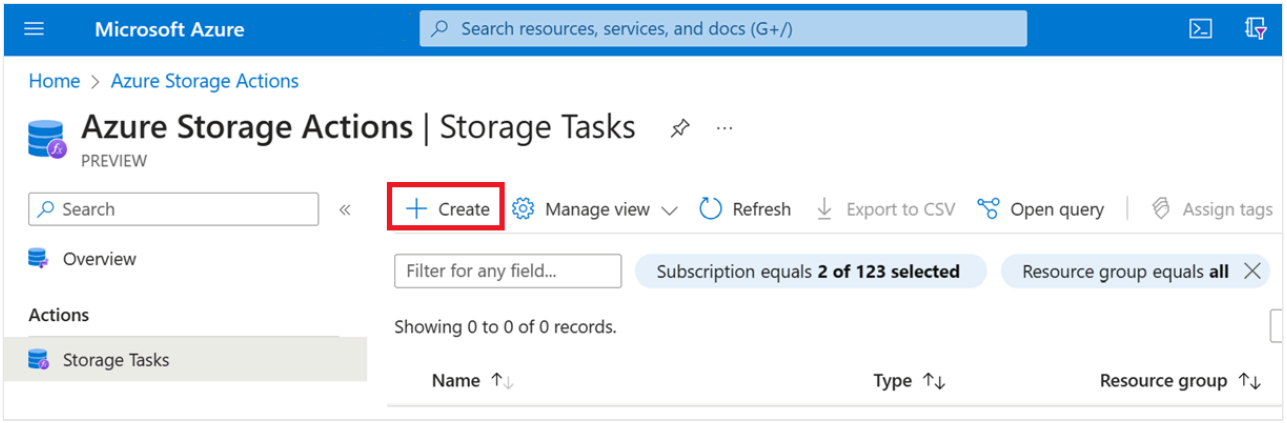
Create a storage task

Portal

In the Azure portal, search for *Storage Tasks*. Then, under **Services**, select **Storage tasks - Azure Storage Actions**.



On the **Azure Storage Actions | Storage Tasks** page, select **Create**.



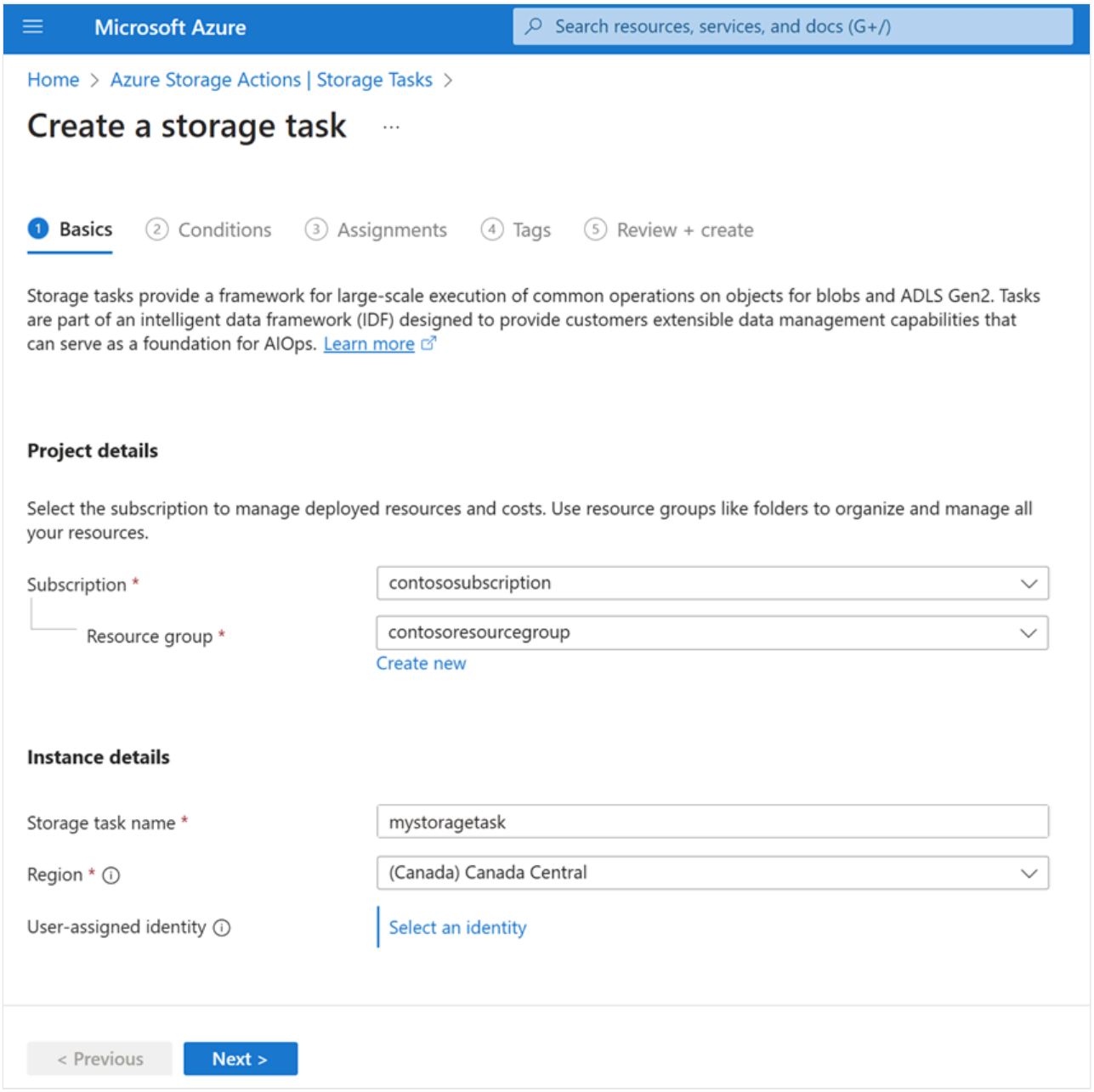
Basics tab

On the **Basics** tab, provide the essential information for your storage task. The following table describes the fields on the **Basics** tab.

Expand table

Section	Field	Required or optional	Description
Project details	Subscription	Required	Select the subscription for the new storage task.
Project details	Resource group	Required	Create a new resource group for this storage task, or select an existing one. For more information, see Resource groups.
Instance details	Storage task name	Required	Choose a unique name for your storage task. Storage task names must be between 3 and 18 characters in length and might contain only lowercase letters and numbers.
Instance details	Region	Required	Select the appropriate region for your storage task. For more information, see Regions and Availability Zones in Azure.
Instance details	User-assigned identity	optional	optionally associate a user-assigned managed identity with this storage task. A user-assigned managed identity is a managed identity is represented as a standalone Azure resource that is managed separately from the resources that use it. You can't associate one later. Therefore, if you want to use a user-assigned managed identity, you must select one as you create the storage task. By default, a system-assigned managed identity is created when the storage task is provisioned. To learn more, see Storage task assignment To select a user-assigned managed identity, choose **Select an identity**. On the **Select user assigned managed identity** page, filter for and then select the managed identity. Then, select **Add**. You can add select a user-assigned managed identity only as you create a storage task.

The following image shows an example of the **Basics** tab.



Conditions tab

On the **Conditions** tab, define the conditions that must be met by each object (container or blob), and the operations to perform on the object.

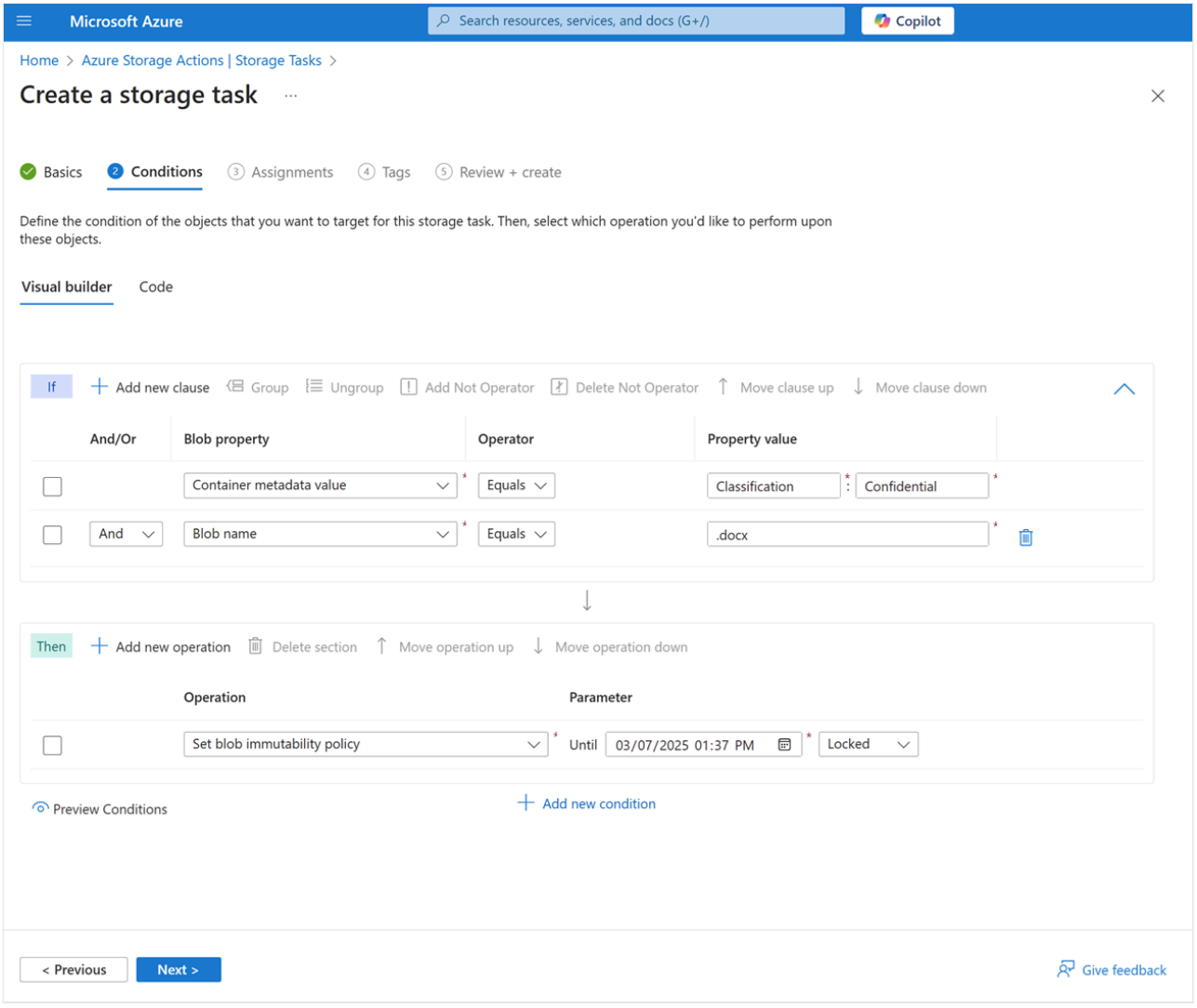
You must define at least one condition and one operation. To add a clause to a condition, select **Add new clause**. To add operations, select **Add new operation**.

The following table describes the fields on the **Conditions** tab.

Expand table

Section	Field	Required or optional	Description
If	And/Or	Required	An operator that combines two or more predicates to form a logical-AND or logical-OR expression.
If	Blob property	Required	The blob or container property that you like to use in the clause. See Supported blob properties
If	Operator	Required	The operator that defines how each property in the clause must relate to the corresponding value. See Supported operators
If	Property value	Required	The value that relates to the corresponding property.
Then	Operations	Required	The action to perform when objects meet the conditions defined in this task. See Supported operations
Then	Parameter	Required	A value used by the operation.

The following image shows an example of the **Conditions** tab.



You can select **Preview Conditions** to view a list of blobs that would be impacted by the conditions that you've defined. To learn more, see Preview the effect of conditions.

Assignments tab

An *assignment* identifies a storage account and a subset of objects in that account that the task will target. An assignment also defines when the task runs and where execution reports are stored.

To add an assignment, select **Add assignment**. This step is optional. You don't have to add an assignment to create the task.

The following table describes the fields that appear in the **Add assignment** pane.

select the role that you want to assign to the system-assigned managed identity of the storage task. To ensure a successful task assignment, use roles that have the Blob Data Owner permissions. To learn more, see Azure roles for storage tasks

Expand table

Section	Field	Required or optional	Description
Select scope	Subscription	Required	The subscription of the storage account that you want to add to this assignment.
Select scope	Select a storage account	Required	The storage account that you want to add to this assignment.
Select scope	Assignment name	Required	The name of the assignment. Assignment names must be between 2 and 62 characters in length and might contain only letters and numbers.
Role assignment	Assignment name	Required	The role that you want to assign to the managed identity of the storage task. To learn more about which role to choose, see Permission for a task to perform operations.
Filter objects	Filter by	Required	Option to either filter objects by using a prefix or to run the task against the entire storage account.
Filter objects	Blob prefixes	Optional	The string prefix that is used to narrow the scope of blobs that are evaluated by the task. This field is required only if you choose to filter by using a blob prefix.

Section	Field	Required or optional	Description
Trigger details	Run frequency	Required	Option to either run the task one time or multiple times.
Trigger details	Start from	Required	The date and time to begin running the task.
Trigger details	End by	Required	The date and time to stop running the task.
Trigger details	Repeat very (in days)	Required	The interval in days between each run.
Trigger details	Report export container	Required	The container where task execution reports are stored.

The following image shows an example of the **Add assignment** pane.

Add assignment

Assign storage task to a storage account. To add or edit assignment, you must have contributor role access to the storage account. Learn more

Select scope

Subscription * contososubscription

Select a storage account * contoso

Assignment name * myassignment

Role assignment

Select a role assignment for the managed identity of the storage task. The role assignment will determine what permissions (read, write, modify) the managed identity will have. Learn more

For a successful role assignment, you must have owner permissions on the selected subscription 'ADLStorePMTeam'

Role Select a role

Filter objects

Select the blob or file objects you want to target by specifying a container name or path prefix. Learn more

Filter by

- (•) Blob prefix
- ( ) Do not filter (run task against entire storage account)

Include blob prefixes * mycontainer/

Enter a prefix or file path such as "myContainer/a"

Exclude blob prefixes *Enter a prefix or file path such as "myContainer/a"*

Trigger details

Determine how often, when, and where the storage task is queued to run.

If another assignment run is in progress, a new run can't start. Instead, it is added to the queue and begins only after the current run completes or reaches the 14-day limit. Queued scheduled runs or one-time runs are skipped if there is a parallel assignment run in-progress. Assignments processing a large number of blobs may take up to 14 days to finish. Learn more

Run frequency

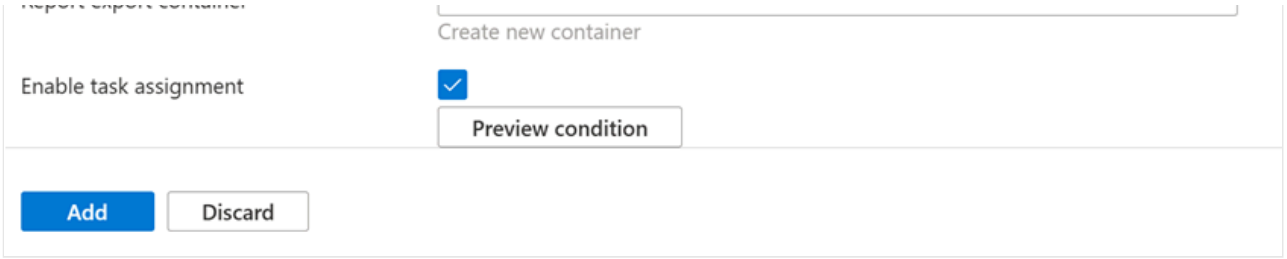
- (•) Scheduled run (recurring)
- ( ) Single run (only once)

Start from 05/02/2025 10:53 AM

End by 05/02/2026 10:53 AM

Repeat every (in days) 7

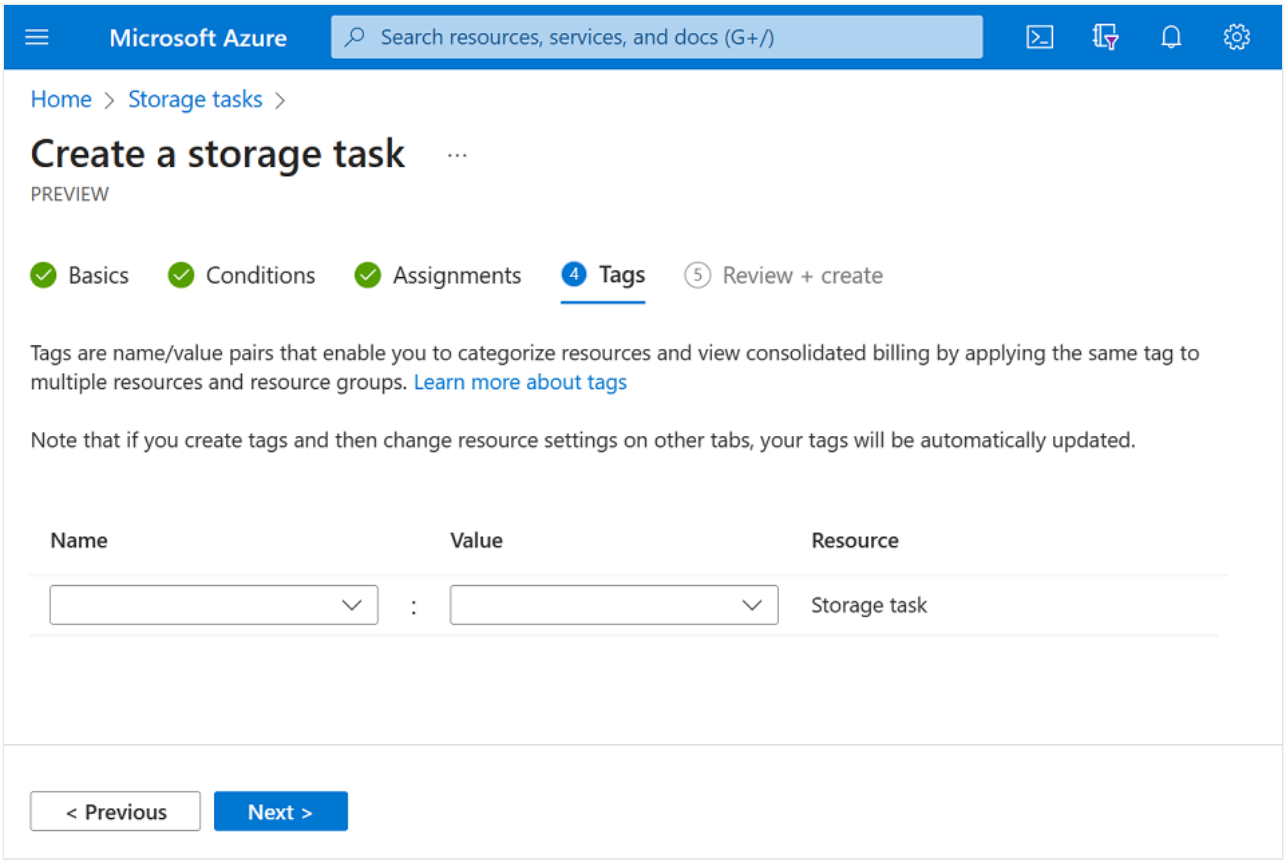
Report export container *



Tags tab

On the **Tags** tab, you can specify Resource Manager tags to help organize your Azure resources. For more information, see Tag resources, resource groups, and subscriptions for logical organization.

The following image shows a standard configuration of the index tag properties for a new storage account.

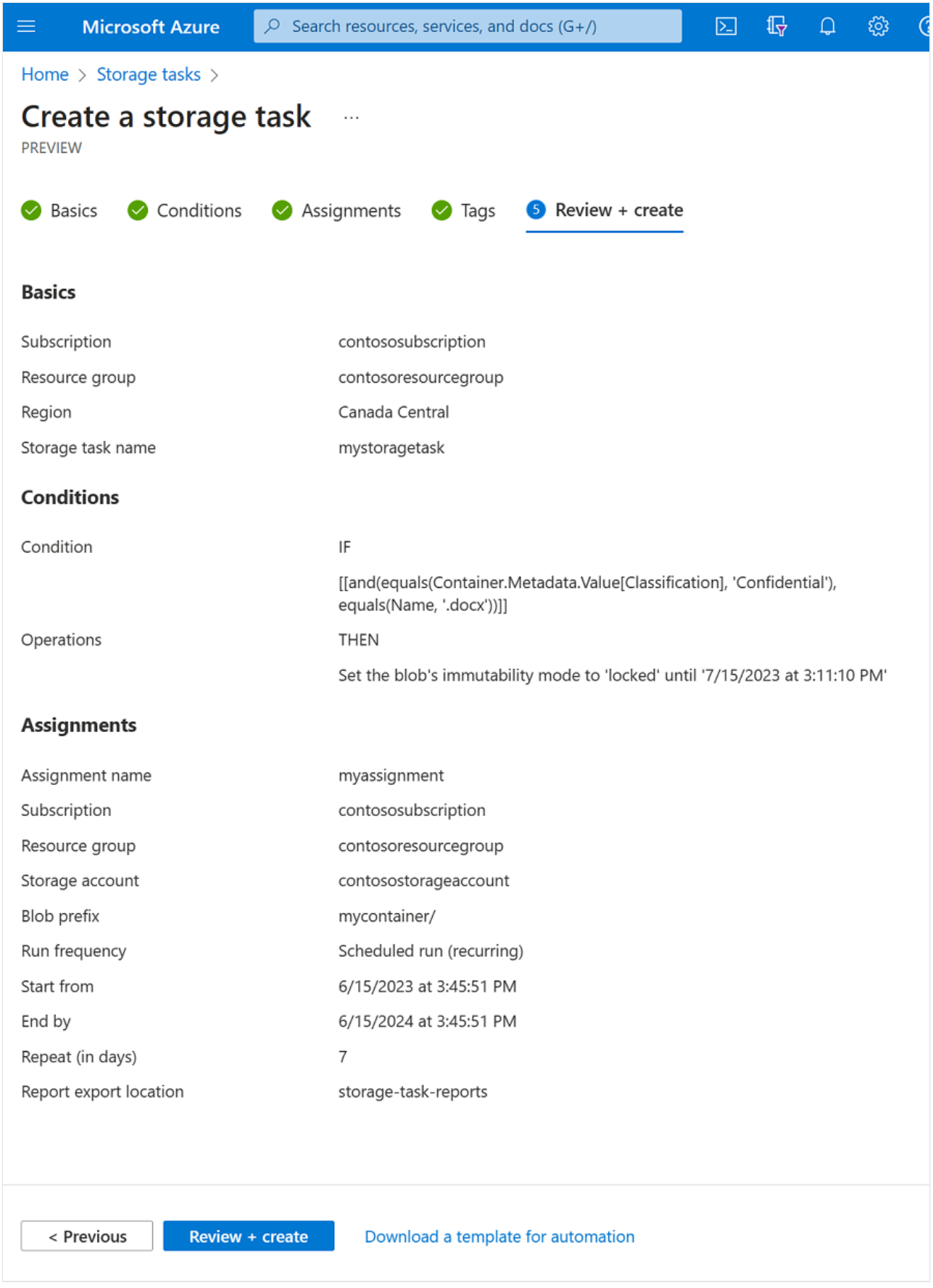


Review + create tab

When you navigate to the **Review + create** tab, Azure runs validation on the storage task settings that you have chosen. If validation passes, you can proceed to create the storage task.

If validation fails, then the portal indicates which settings need to be modified.

The following image shows the **Review** tab data prior to the creation of a new storage task.



See also

- Azure Storage Actions overview
- Create, assign, and run a storage task
- Define conditions and operations

Define storage task conditions and operations

Article • 05/07/2025

You can use a visual editor to define the conditions and operations of a storage task.

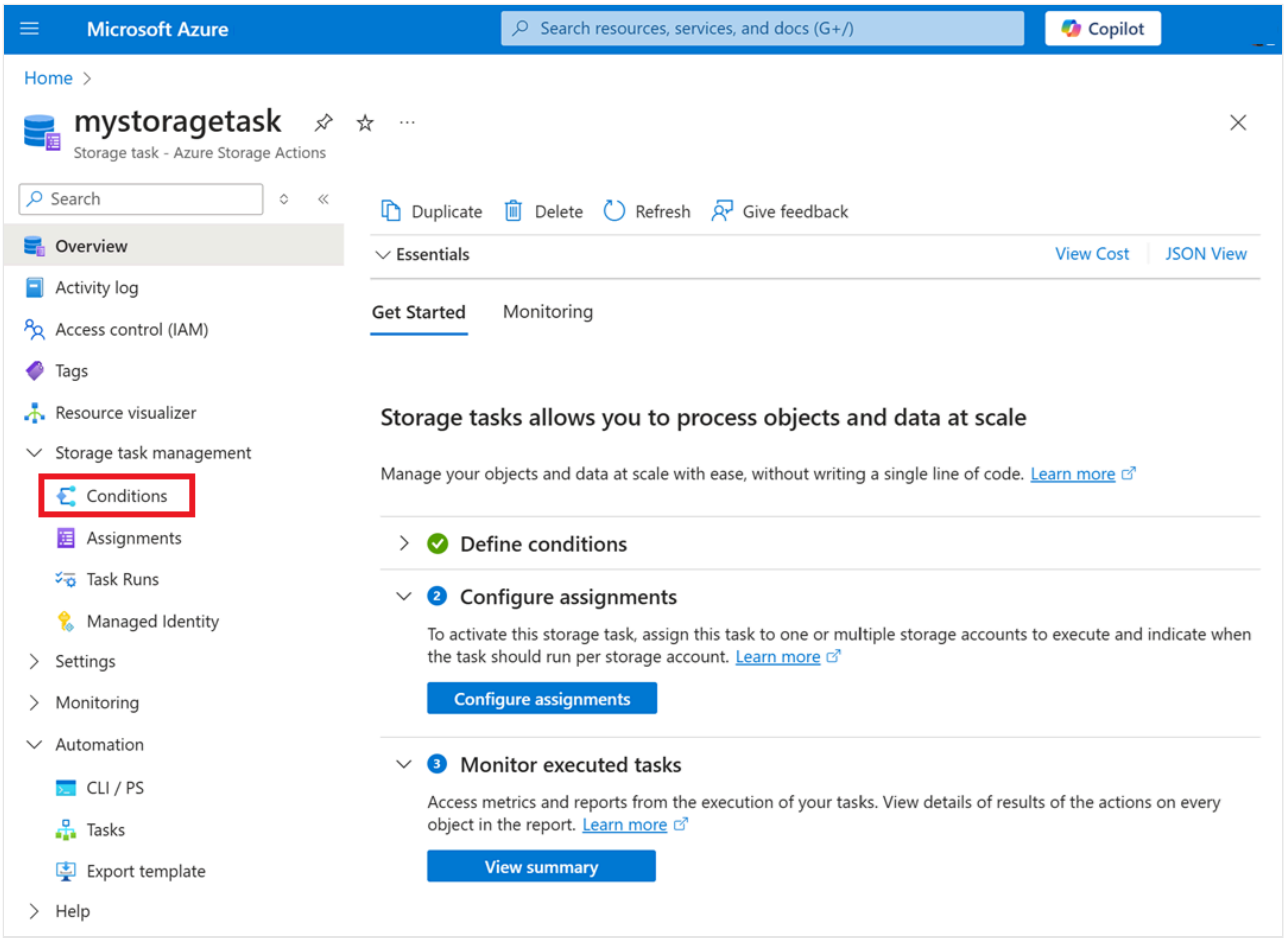
An *operation* is an action taken on each object that meets the conditions defined in the task. A *condition* contains one or more conditional *clauses*. Each clause defines the relationship between a property and a value. To execute an operation defined in the storage task, the terms of that relationship must be met by each object.

Define conditions

Define a condition by adding clauses. A clause defines the relationship between a property and a value. To execute an operation defined in the storage task, the terms of that relationship must be met by each object.

Portal

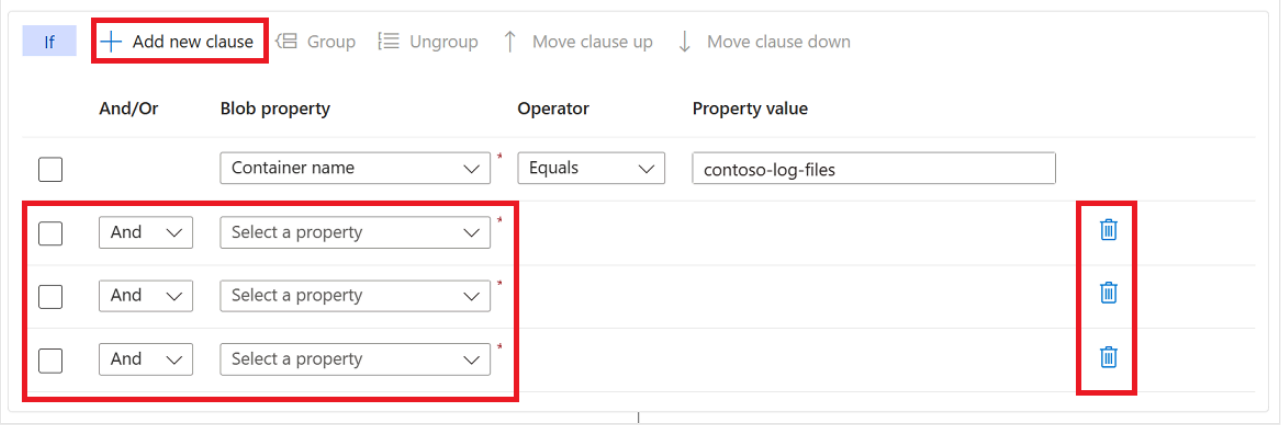
Navigate to the storage task in the Azure portal and then under **Storage task management**, select **Conditions**.



The **Visual builder** tab of the **Conditions** pane appears.

Add and remove clauses

To add a clause, select **Add new clause**, and to remove a clause, select the delete icon( ) that appears next to it.



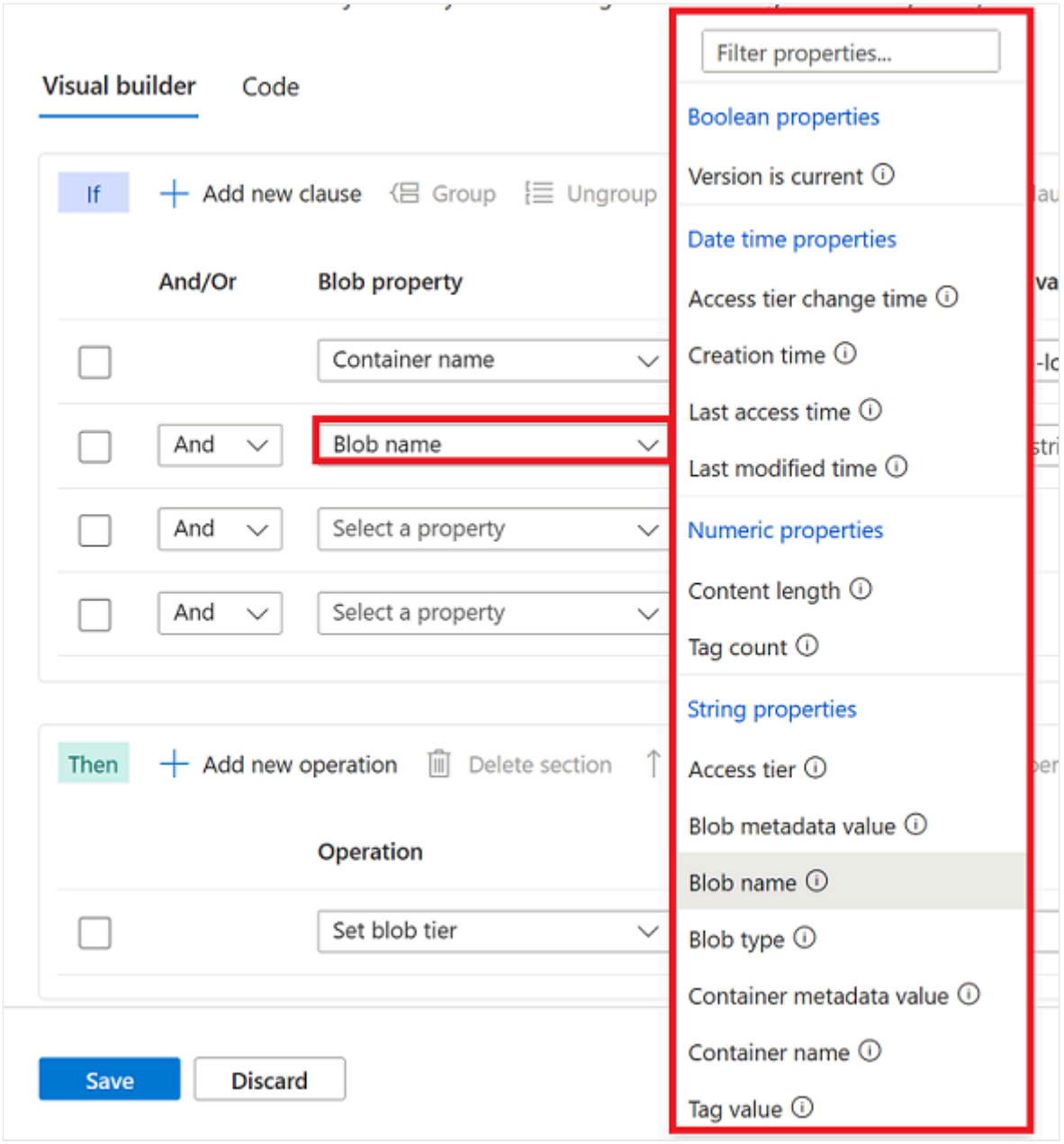
Specify the terms of a clause

To define a clause, choose a property, specify a value for that property, and then choose an operator that relates them together.

Choose a property

In the **Blob property** drop-down list, choose a property. See Supported blob properties.

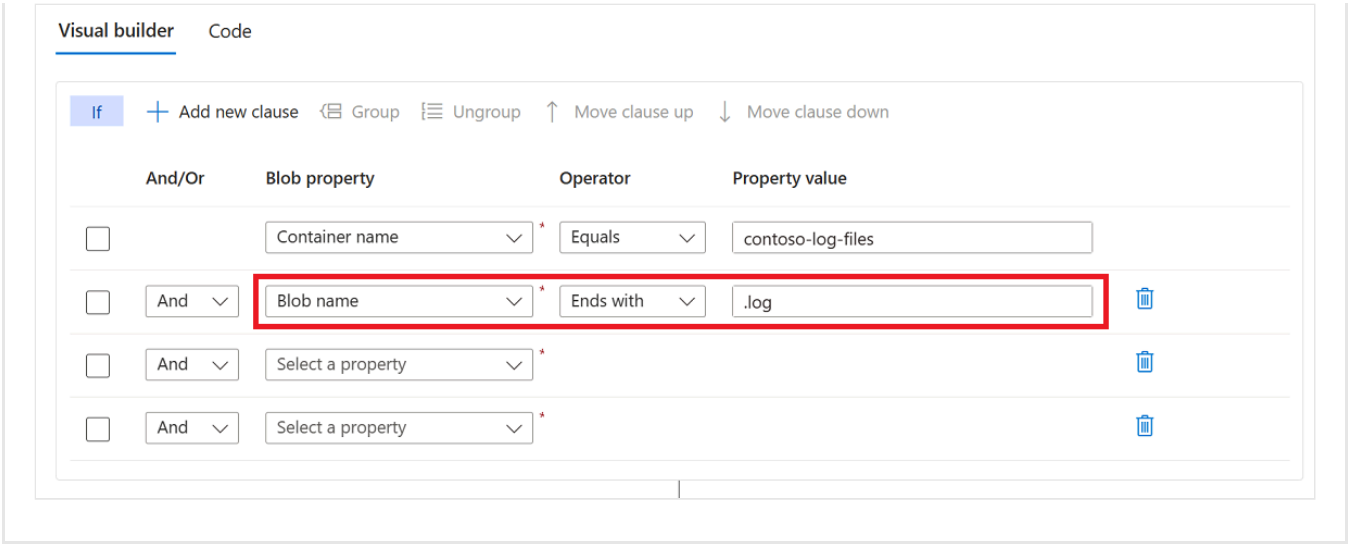
The following example selects the **Blob name** property.



Choose a value and operator

In the **Property value** box, enter a value and in the **Operator** drop-down list, choose an operator. See Supported Operators.

The following example specifies a value of `.log` along with the **Ends with** operator. This condition allows the operation defined in this storage task to execute only on blobs that have a `.log` file extension.



Use a wildcard in string values

You can use the `*` and `?` wildcard characters in the value of a string property. The `*` character represents zero or more characters while a `?` character represents exactly one character.

For example, if you want your clause to evaluate to true only for blobs that are Word documents, you would use the string `*.docx`. However, if you want only documents are named with a single character such as `1.docx` or `2.docx`, then you would use the string `?.docx`.

You can use the `*` or `?` anywhere in a string. You can escape these characters by adding a `\` just before the character.

Reference a value from object metadata

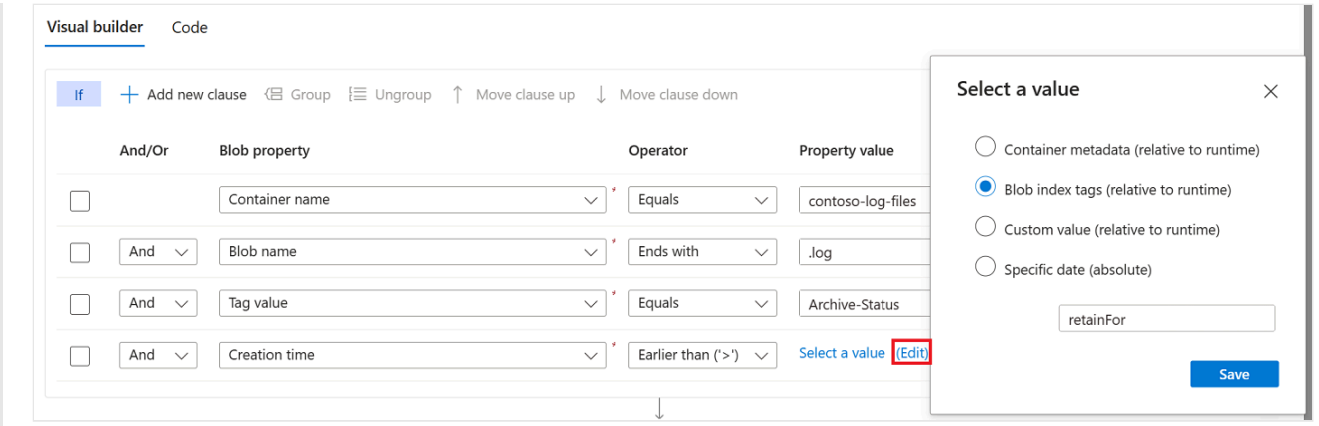
Clauses that include a date and time property can reference a value from the metadata of a container or an index tag of a blob. These values are obtained dynamically at runtime when the task executes.

In your storage account, you can add a key to the metadata of a container or to the index tags of a blob. The value of that key must be a ISO 8601 -formatted time interval. For example, you might add a key named `retainfor` along with a string value of `PT5M` which represents an interval of five minutes.

Portal

To reference a key, select the **Edit** link that appears in the **Property value** column. Then, in the **Select a value** dialog box, select **Container metadata** or **Blob Index tags**.

The following example adds the **Creation time** property, the **Earlier than** operator, and references a key named `retainFor` in the index tags of each blob that is evaluated.



This condition tests whether a blob was created earlier than a certain time duration relative to now (the current date and time). For example, if the value retrieved from the `retainFor` tag is five minutes, then this condition checks if the blob was created more than 5 minutes ago.

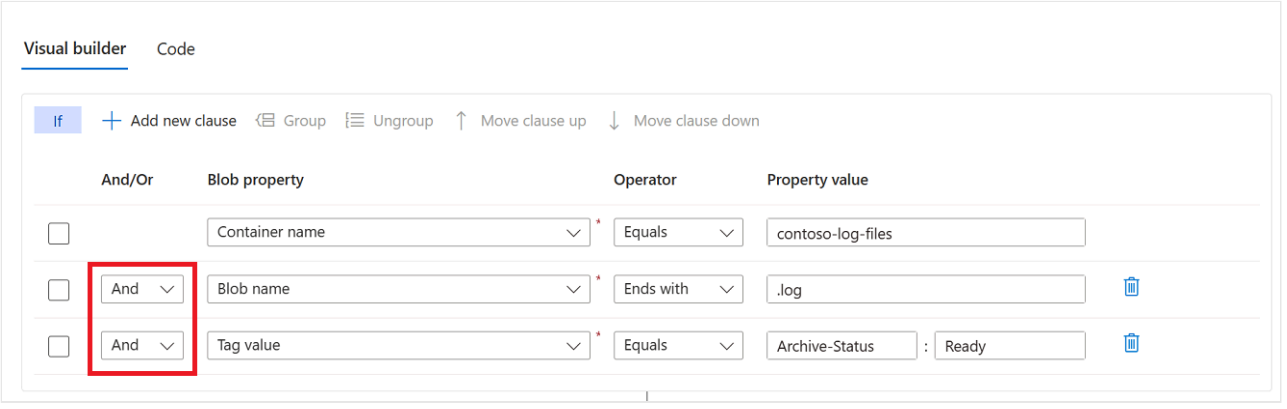
If the key isn't present for an evaluated object, then the condition evaluates to false. If the key value is a string that doesn't conform to the ISO 8601 standard, then an error is reported in the execution report.

Apply And / Or to a clause

You add **And** or **Or** to a clause. Specify **And** if you want to target objects that meet the criteria in both the current clause and the previous clause of the condition. Specify **Or** to target objects that meet the criterion in either the current clause or the previous clause.

Portal

The following example shows clauses that use **And**. In this example, the storage task targets objects that have a `.log` extension and which have a tag named `Archive-Status` set to `Ready`.



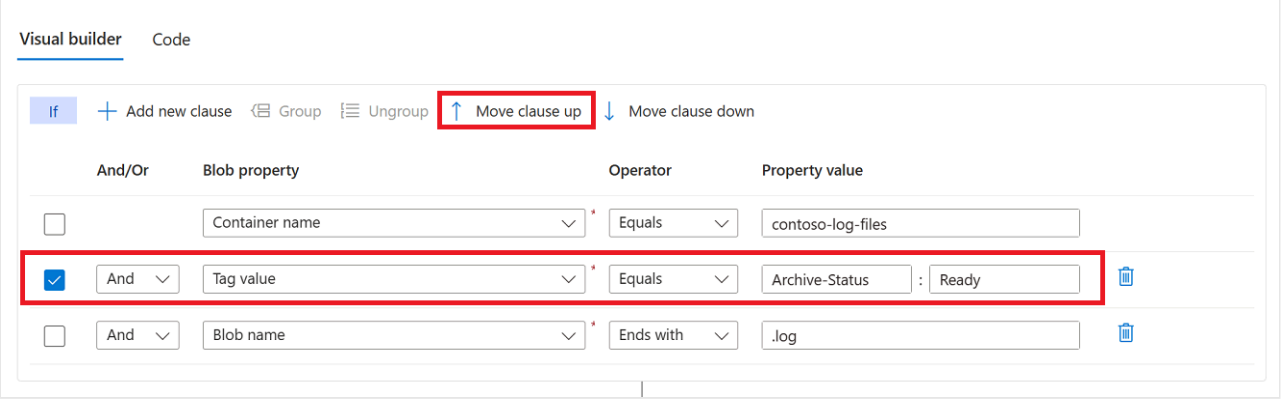
Change the order of clauses

You can arrange clauses in an order that you believe will improve the performance of a task run. For example, instead of first testing all blobs in an account against a name filter, you might elevate a clause that targets a specific container. That small adjustment can prevent the task from performing unnecessary evaluations.

The following example reverses the order of the clauses shown in the previous example. This condition evaluates the index tag clause first and then the blob name clause.

Portal

First, select the clause. Then, select **Move clause up** or **Move clause down** to change its position in the list.



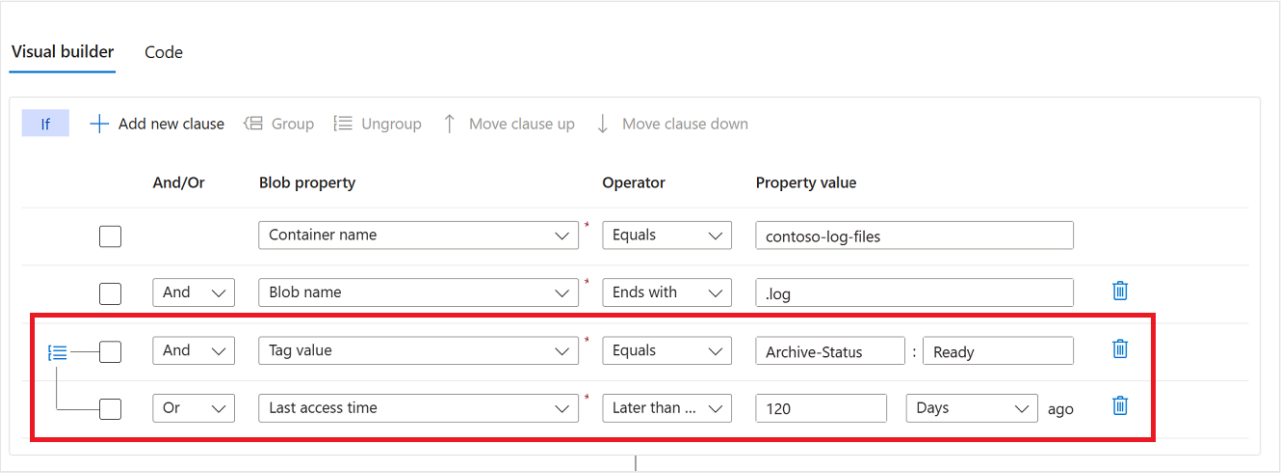
Group and ungroup clauses

Grouped clauses operate as a single unit separate from the rest of the clauses. Grouping clauses is similar to putting parentheses around a mathematical equation or logic expression. The **And** or **Or** operator for the first clause in the group applies to the whole group.

The following example shows two conditions grouped together. In this example, the operation executes if a blob has the `.log` extension and either a tag named `Archive-Status` is set to the value of `Ready` or the file hasn't been accessed in 120 days.

Portal

Select the checkbox that appears next to each clause you want to group together. Then, select **Group**.



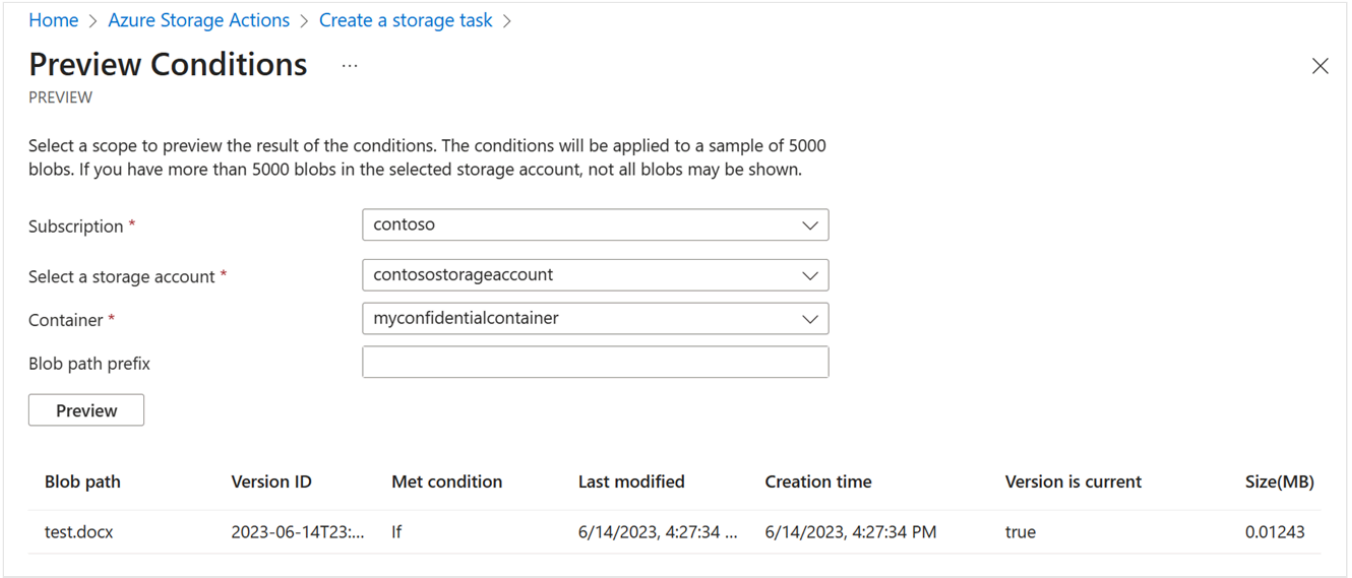
To ungroup clauses, select the ungroup icon ( ) or select each clause in the group, and select **Ungroup**.

Preview the effect of conditions

You can view a list of blobs that would be impacted by the conditions that you've defined.

In the conditions editor, select **Preview conditions**.

In the **Preview Conditions**, you can specify a target subscription, storage account, and container. Because a can only show up to 5,000 blobs, you can also specify a prefix to narrow the list.



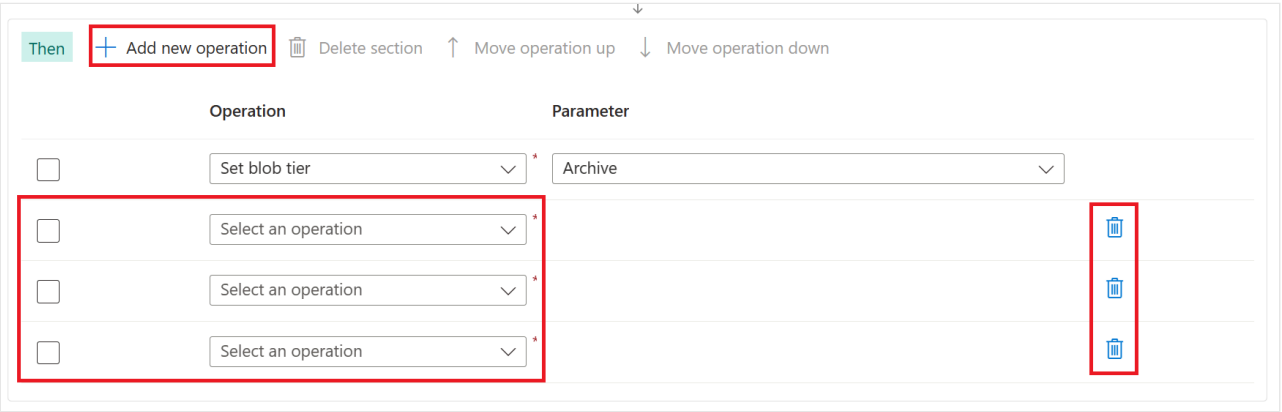
Define operations

An operation is an action taken on each object that meets the conditions defined in the task.

Portal

Add and remove operations

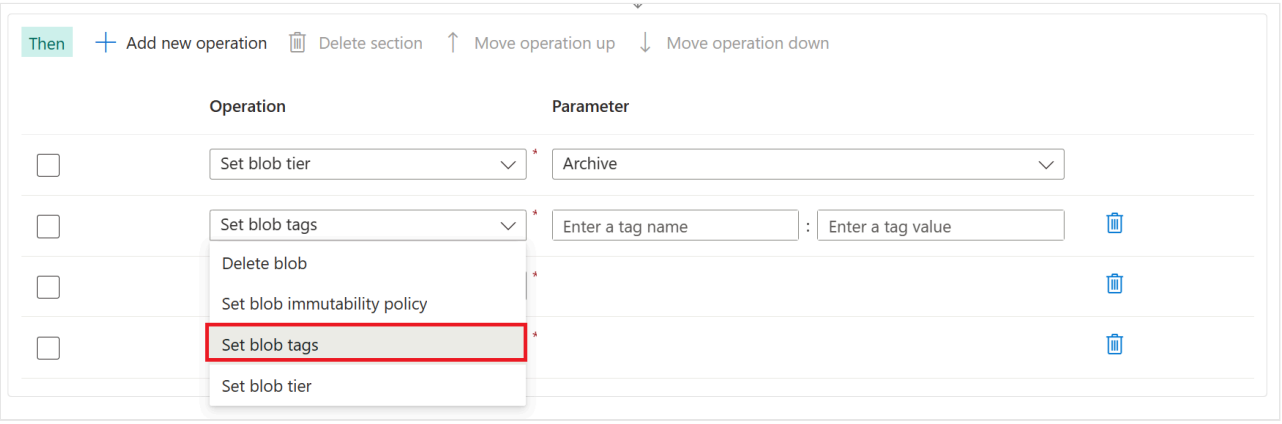
To add an operation, select **Add new operation**, and to remove an operation, select the delete icon( ) that appears next to it.



Choose an operation

In the **Operation** drop-down list, choose an operation. See Supported operations.

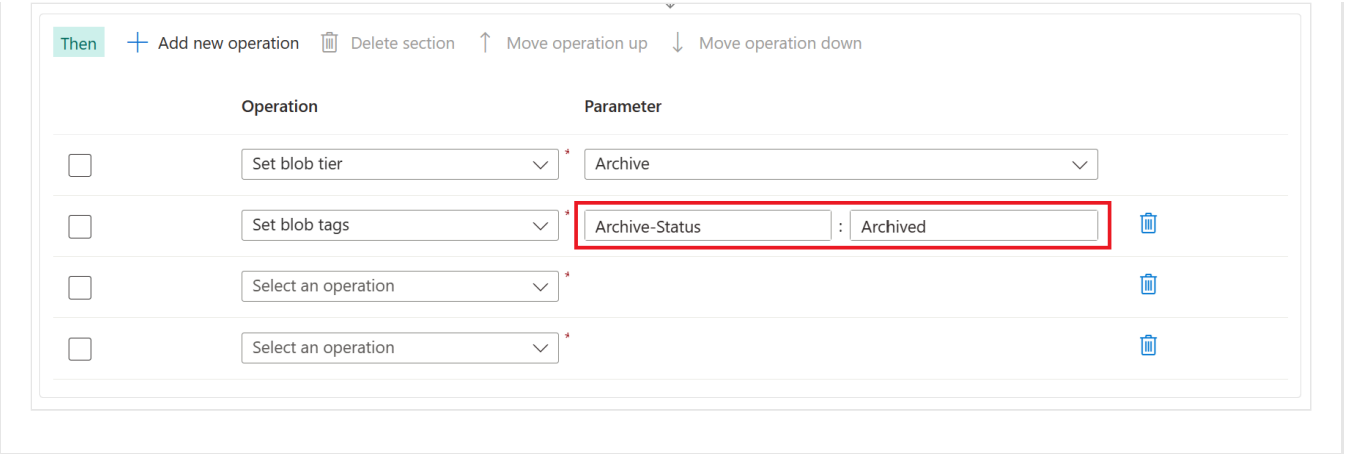
The following example selects the **Set blob tags** property.



Choose a parameter

Enter or select the parameters that are appropriate for the operation.

The following example sets the `Archive-Status` tag to the value `Archived`.



To learn more about the structure of operations and to find a complete list of operations, see Storage task operations.

Change the order of operations

You can arrange operations in any order.

In this example, the existing order makes sense. Blobs are first archived and the tags are set. It wouldn't make sense to set the tag before changing the tier just in case the attempt to change the tier of a blob didn't succeed. If the set blob tag operation appeared first in the list, you might consider moving that operation beneath the set blob tier operation.

Portal

To move an operation, select the checkbox that appears beside it. Then, select **Move operation up** or **Move operation down** to change its position in the list.

See also

- Azure Storage Actions Overview

Create and manage a storage task assignment

Article • 05/07/2025

An *assignment* identifies a storage account and a subset of objects in that account that the task will target. An assignment also defines when the task runs and where execution reports are stored.

This article helps you create an assignment, and then enable that assignment to run. To learn more about storage task assignments, see Storage task assignments.

> **Important**
>
> Before you enable storage task assignment, make sure that you grant access to trusted Azure services in the network settings of each target storage account. To learn more, see Grant access to trusted Azure services.

Create and manage an assignment

Create an assignment for each storage account you want to target. A storage task can contain up to 50 assignments.

> **Note**
>
> In the current release, you can target only storage accounts that are in the same region as the storage tasks.

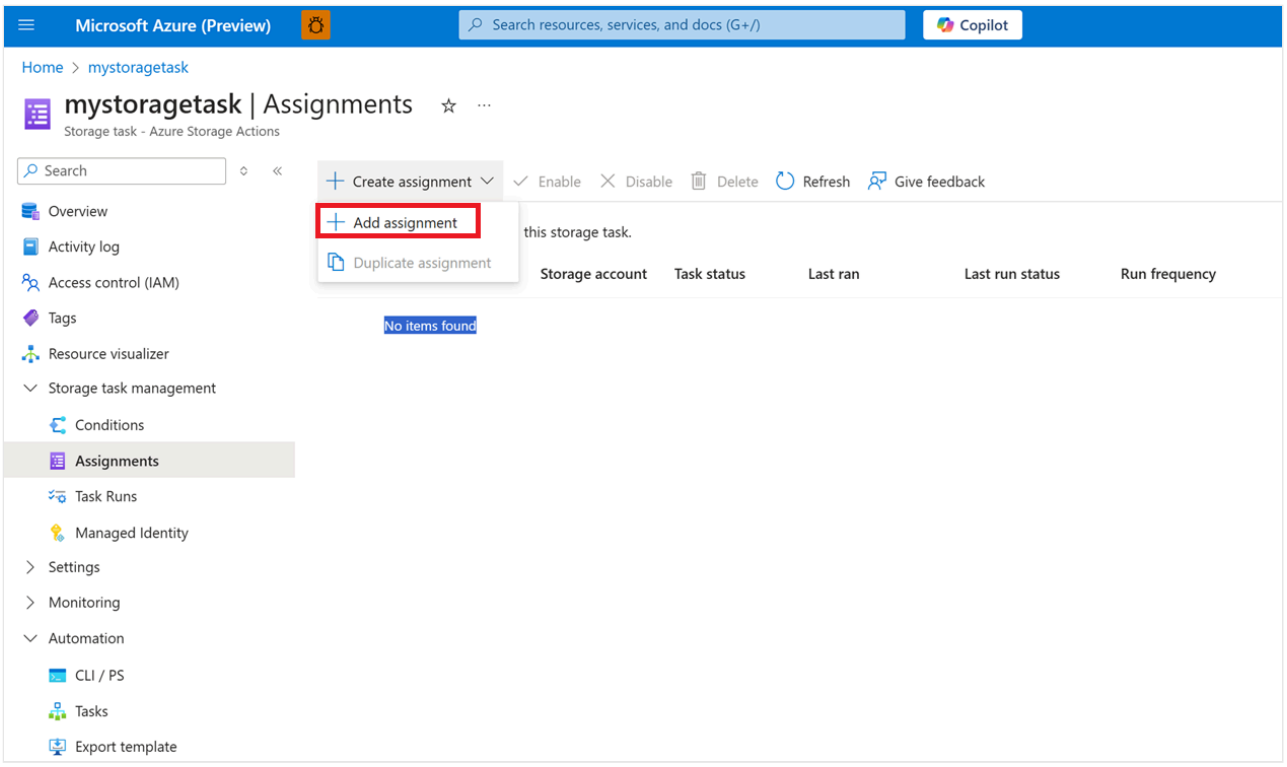
Portal

Create an assignment from the storage task menu

You can create an assignment in the context of a storage task. This option can be convenient if you're the task author and you want to target multiple storage accounts. For each assignment, you'll identify the storage account that you want to target.

Navigate to the storage task in the Azure portal and then under **Storage task management**, select **Assignments**.

In the **Assignments** page, select **+ Add assignment** and the **Add assignment** pane will appear.

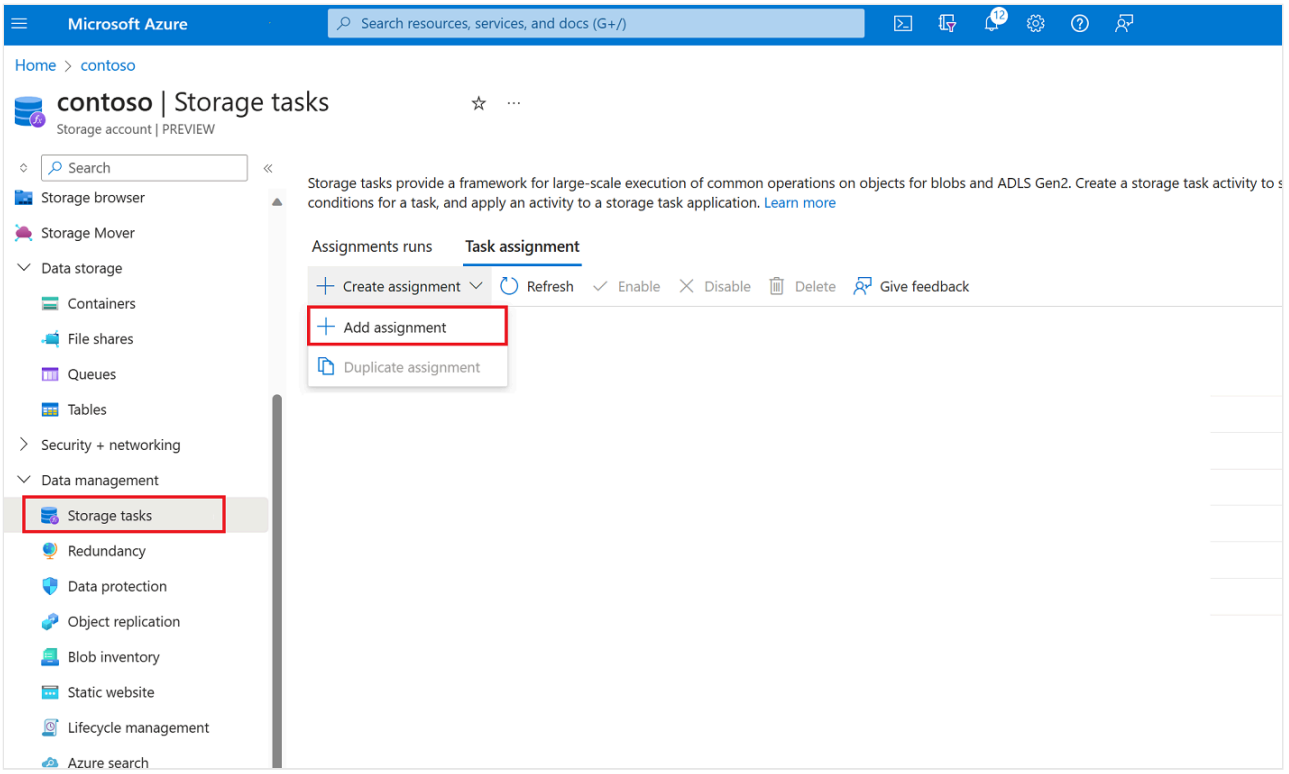


Create an assignment from the storage account menu

You can also create an assignment in the context of a storage account. This option can be convenient if you want to use an existing task to process objects in your storage account. For each assignment, you'll identify the storage task that you want to assign to your account.

Navigate to the storage account in the Azure portal and then under **Data management**, select **Storage tasks**.

In the **Storage tasks** page, select the **Task assignment** tab, select **+ Create assignment**, and then select **+ Add assignment**.



The **Add assignment** pane appears.

Select a scope

In the **Select scope** section, select a subscription and name the assignment. Then, select the storage account that you want to target.

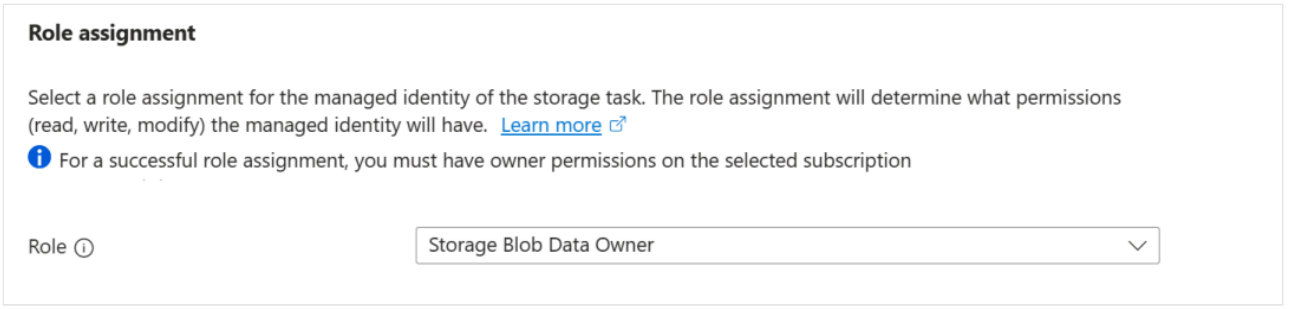
If you opened the **Add assignment** pane in the context of the storage account, you'll select a storage task instead of the storage account.

For a description of each property, see Assignment settings.

Add a role assignment

In the **Role assignment** section, in the **Role** drop-down list, select the role that you want to assign to the managed identity of the storage task. To ensure a successful task assignment, use roles that have the Blob Data Owner permissions. To learn more, see Azure roles required to assign tasks.

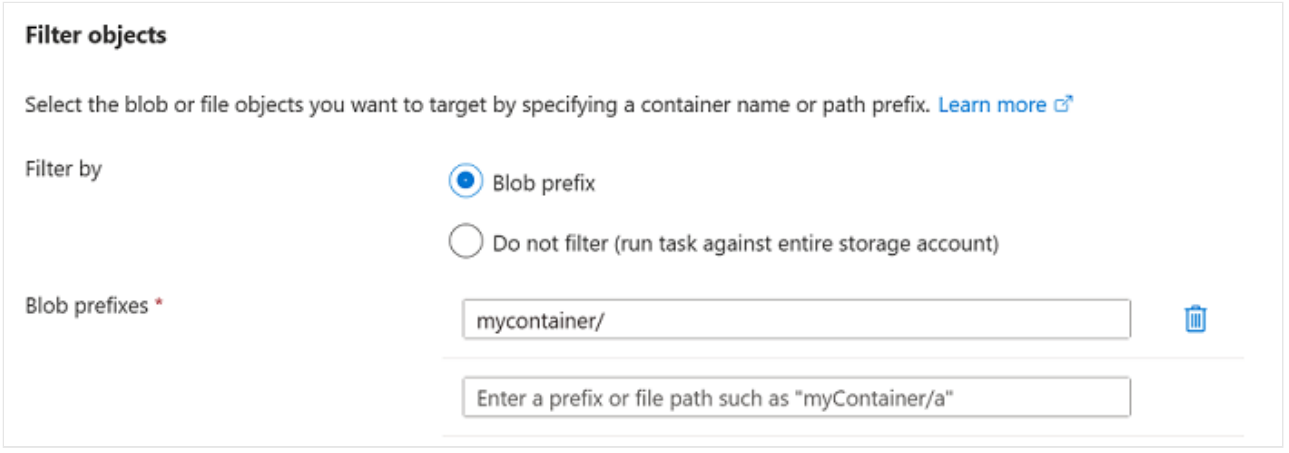
> **Note**
>
> You choose the managed identity type (system-assigned or user-assigned) as part of creating the storage task.



For a description of each property, see Assignment settings.

Add a filter

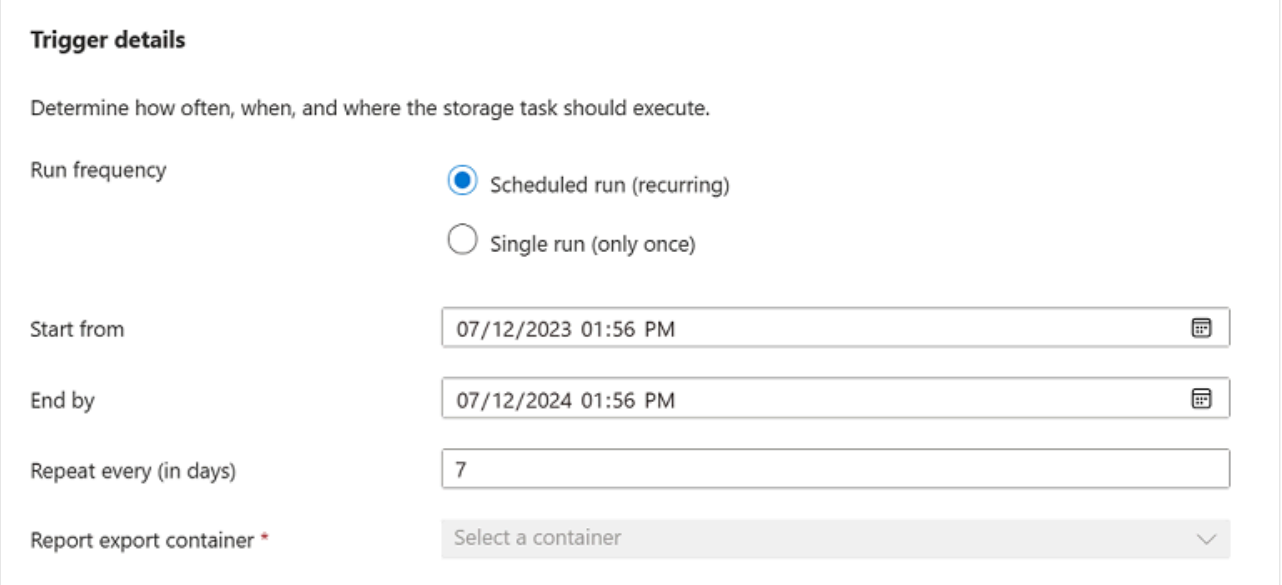
In the **Filter objects** section, choose whether you want to target a subset of blobs based on a filter. Filters help you narrow the scope of execution. If you want the task to evaluate all of the containers and blobs in an account, then you can select the **Do not filter** option. The following example uses a filter to target only blobs that exist in a container that is named `mycontainer`.



For a description of each property, see Assignment settings.

Define the trigger

In the **Trigger details** section, select how often you'd like this task to run. You can choose to run this task only once, or run the task recurring. If you decide to run this task on a recurring basis, choose a start and end time and specify the number of days in between each run. You can also specify where you'd like to store the execution reports.

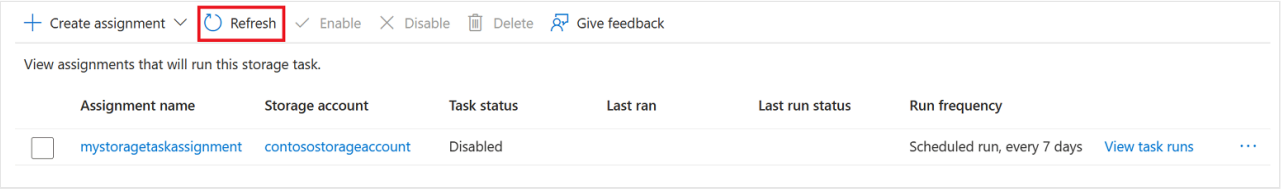


For a description of each property, see Assignment settings.

Save the assignment

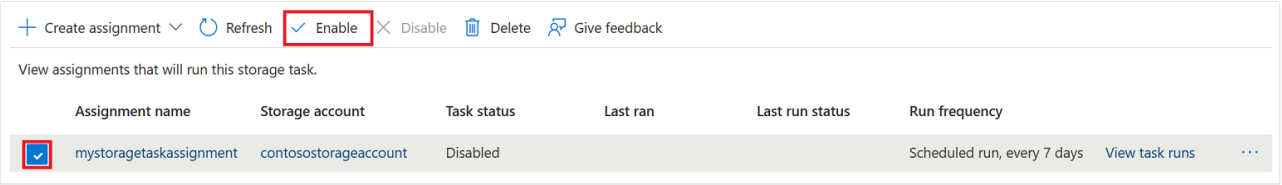
Select the **Add** button to create the assignment.

The **Add assignment pane** closes. When deployment is complete, the assignment appears in the **Assignments** page. If you don't see the assignment in that page, then select the **Refresh** button.



Enable an assignment

The assignment is disabled by default. To enable the assignment so that it will be scheduled to run, select the checkbox that appears beside the assignment, and then select **Enable**.



After the task runs, an execution report is generated and then stored in the container that you specified when you created the assignment. For more information about that report as well as how to view metrics that capture the number of objects targeted, the number of

operations attempted, and the number of operations that succeeded, see Analyze storage task runs.

Edit an assignment

An assignment becomes a sub resource of the targeted storage account. Therefore, after you create the assignment, you can edit only it's run frequency. The other fields of an assignment become read only. The **Single run (only once)** option becomes read only as well.

- To edit the run frequency of an assignment in the context of a storage task, navigate to the storage task in the Azure portal and then under **Storage task management**, select **Assignments**.
- To edit the run frequency of an assignment in the context of a storage account, navigate to the storage account in the Azure portal and then under **Data management**, select **Storage tasks**.

See also

- Storage task assignments
- Azure Storage Actions overview
- Analyze storage task runs

Monitor Azure Storage Actions

Article • 05/05/2025

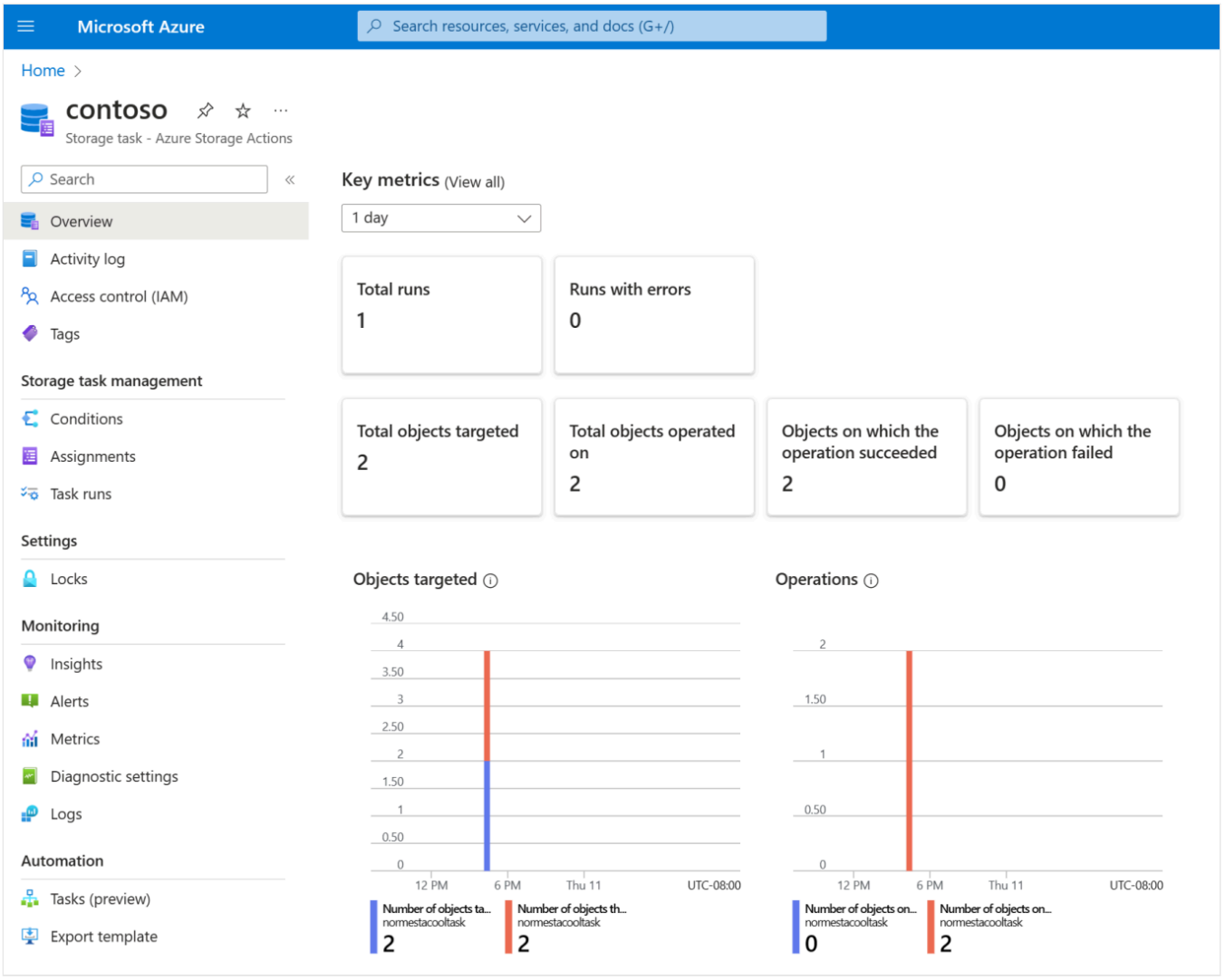
When you have critical applications and business processes relying on Azure resources, you want to monitor those resources for their availability, performance, and operation. This article describes the monitoring data generated by Azure Storage Actions. Azure Storage Actions uses Azure Monitor. If you're unfamiliar with the features of Azure Monitor common to all Azure services that use it, read Monitoring Azure resources with Azure Monitor.

Monitoring overview page in Azure portal

The following metrics appear in the **Overview** page of each storage task.

- Total number of runs
- Total number of runs that resulted in an error
- Total number of objects targeted by task runs
- Total number of objects operated on
- Total number of objects where an operation succeeded
- Total number of objects where an operation failed

The following image shows how these metrics appear in the **Overview** page.



These metrics include runs from multiple storage task assignments, but only assignments that target storage accounts to which you have read permission. These metrics appear as tiles that you can select to view a list of the task runs which comprise the metric. Each listed task provides a link to a detailed execution report. For more information about how to drill into metrics and reports for each task run, see Analyze storage task runs.

Monitoring data

Azure Storage Actions collects the same kinds of monitoring data as other Azure resources that are described in Monitoring data from Azure resources. For information about the metrics and logs that are created by Azure Storage tasks, see Monitoring Azure Storage Actions data reference.

Collection and routing

Platform metrics and the activity log are collected and stored automatically, but can be routed to other locations by using a diagnostic setting. To learn how to create a diagnostic setting, see Create diagnostic setting to collect platform logs and metrics in Azure.

Analyzing metrics

You can analyze metrics for Azure Storage Actions with metrics from other Azure services using metrics explorer by opening **Metrics** from the **Azure Monitor** menu. See Getting started with Azure Metrics Explorer for details on using this tool.

For a list of the platform metrics collected for Azure Storage Actions, see Monitoring Azure Storage Actions metrics. You can also see a list all resource metrics supported in Azure Monitor.

Analyzing logs

Azure Storage Actions don't support resource logs. Azure Storage Actions support the activity log log only. The activity log is a type of platform log that provides insight into subscription-level events. You can view it independently or route it to Azure Monitor Logs, where you can do much more complex queries using Log Analytics.

Alerts

Azure Monitor alerts proactively notify you when important conditions are found in your monitoring data. They allow you to identify and address issues in your system before your customers notice them. You can set alerts on metrics and the activity log.

Next steps

- See Monitoring Azure Storage Actions data reference for a reference of the metrics, logs, and other important values created by Azure Storage Actions.
- See Monitoring Azure resources with Azure Monitor for details on monitoring Azure resources.

Analyze storage task runs

Article • 05/07/2025

You can view a list of task runs. For each completed run, you can open a report that lists each object that met the conditions of the task and the outcome of the operation against that object.

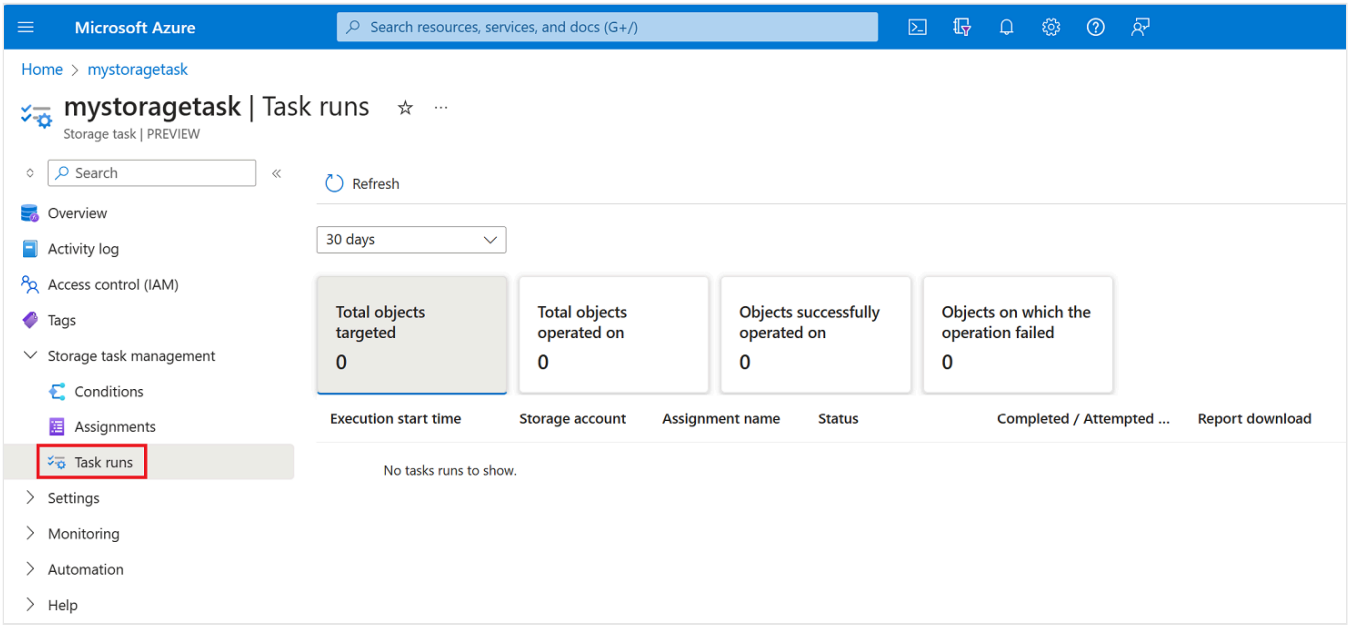
View a list of task runs

After a task run completes, it appears in a list that you can open and review.

Storage task view

You can see all of the runs attempted by a storage task by opening a list of runs from the storage task menu. You'll see only those runs against accounts to which you have read permission.

Navigate to the storage task in the Azure portal and then under **Storage task management**, select **Task runs**.



Metrics appear as tiles that you can select to view a list of the task runs which comprise the metric. For example, to view list of task runs where at least one operation failed, select the **Objects on which the operation failed** tile. Then, a filtered list of task runs appear. You can filter this list by time frame. The following table describes each field that appears in this list.

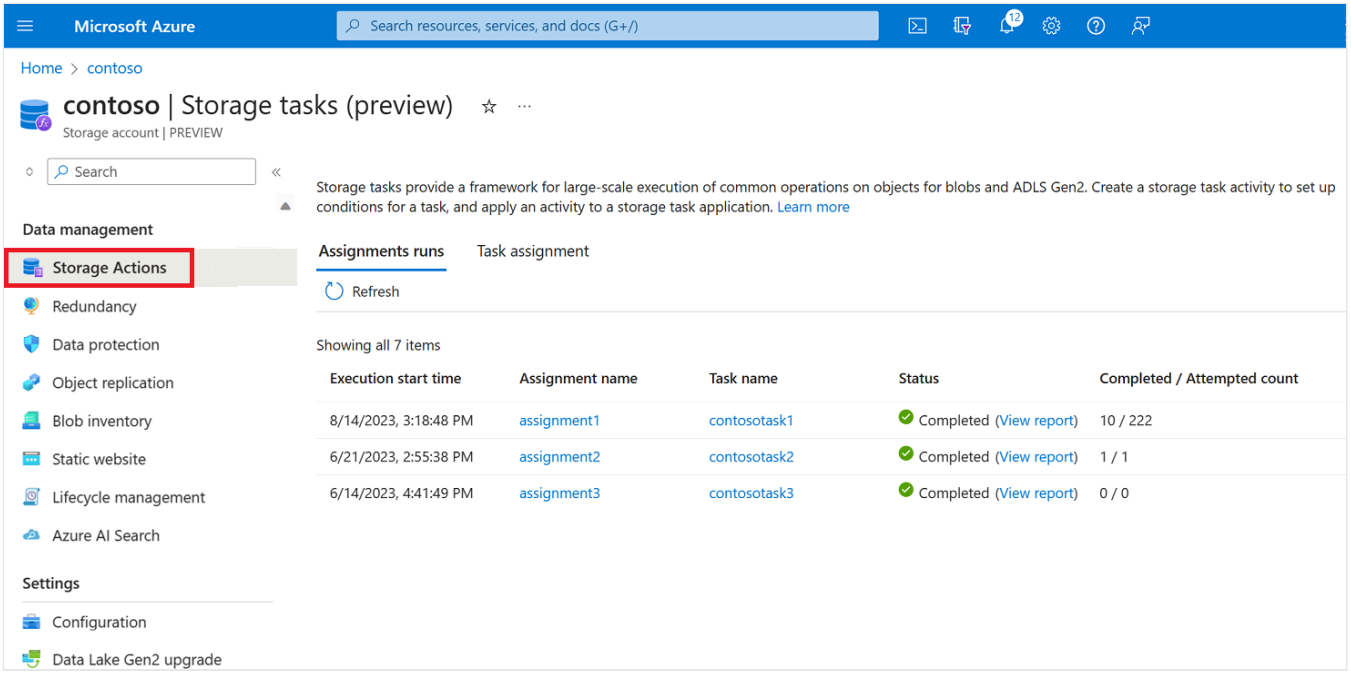
Expand table

Column	Description
Execution start time	The date and time when the task run started.
Storage account name	The name of the storage account that was targeted by the task assignment.
Assignment name	The name of the assignment. This name is specified by the creator of the assignment at configuration time.
Status	Specifies whether the task completed or is pending execution.
Completed / attempted count	The number of objects which met the conditions of the storage task and the total number of objects targeted by the storage task.
Report download	A link to an execution report. This report appears only when the status is `Completed`.

Storage account view

You can see all runs against a specific storage account by opening a list of runs from the storage account menu.

Navigate to the storage account in the Azure portal and then under **Data management**, select **Storage tasks**.



A list of task runs appears. You can filter this list by time frame. The following table describes each field that appears in this list.

Expand table

Column	Description
Execution start time	The date and time when the task run started.
Assignment name	The name of the assignment. This name is specified by the creator of the assignment at configuration time.
Task name	The name of the storage task. This name is specified by the creator of the assignment at configuration time.
Status	Specifies whether the task completed or is pending execution. When the status is `Completed`, a link to an execution report appears.
Completed / attempted count	The number of objects which met the conditions of the storage task and the total number of objects targeted by the storage task.

View execution reports

Each task run generates an execution report. That report is stored in a container that is located in the targeted storage account. The name of that container is specified when the assignment is created. A link to the report appears next to each run in the task run list. Use that link to open a report which contains status information about each object that was targeted by the run.

To open the report, select the **View report** link for any listed task run. Then, the report begins downloading. The report is packaged as a compressed file so you have to extract the contents of that file to a directory on your client.

The report is formatted as a CSV file. Each row of the report contains the details about the execution of the operation on each object that is targeted by the task. The following table describes the columns of information that appear in the execution report:

Expand table

Name	Description
Container	Definition
Blob	The fully qualified name of the blob.
Snapshot	The ID of the snapshot for this task. This field is empty if the object is a base blob.
VersionId	The ID of the version of this task. This field is empty if the object is a base blob.
Operation	The operation attempted.

Name	Description
Result	This field contains a value of `Success` or `Failed`. This field is set to `N/A` if the application of the defined conditions on the object evaluates to false.
Error description	This field is empty if the operation succeeds or the application of the defined conditions on the object evaluates to false.
Error code	This field is empty if the operation succeeds or the application of the defined conditions on the object evaluates to false.
Matched condition block	The condition block associated with the row.

The following example shows an execution report:

	A	B	C	D	E	F	G	H	I	J	K
1	Container	Blob	Snapshot	VersionId	Operation	Result	Error description	Error code	Matched condition block		
2	testcontainer	testblobA1			SetBlobTier	Success			IF		
3	testcontainer	testblobA1			SetBlobTags	Success			IF		
4	testcontainer	testblobA3				N/A			No conditions were met		
5	testcontainer	testblobB0				N/A			No conditions were met		
6	testcontainer	testblobB2			SetBlobTags	Success			IF		
7	testcontainer	testblobB2			SetBlobTier	Success			IF		
8	testcontainer	testblobB4				N/A			No conditions were met		
9	testcontainer	testblobB5			SetBlobTags	Failure	Encountered internal error	InternalError (HTTP - 500)	IF		
10	testcontainer	testblobB5			SetBlobTier	Failure	Encountered internal error	InternalError (HTTP - 500)	IF		
11	testcontainer	testblobB6			SetBlobTags	Failure	Operation not supported on blob type	RuntimeError (HTTP - 403)	IF		
12	testcontainer	testblobB6			SetBlobTier	Failure	Operation not supported on blob type	RuntimeError (HTTP - 403)	IF		
13											

See also

- Monitor Azure Storage Actions
- Storage Tasks Overview

Resource Manager

Article • 09/24/2024

REST Operation Groups

Expand table

Operation Group
Operations
Storage Task Assignment
Storage Tasks
Storage Tasks Report

Microsoft Azure Storage Actions management client library for .NET

04/23/2025

Azure Storage Actions is a serverless framework that you can use to perform common data operations on millions of objects across multiple storage accounts.

This library follows the new Azure SDK guidelines, and provides many core capabilities:

```
- Support MSAL.NET, Azure.Identity is out of box for supporting MSAL.NET.
- Support [OpenTelemetry](https://opentelemetry.io/) for distributed tracing.
- HTTP pipeline with custom policies.
- Better error-handling.
- Support uniform telemetry across all languages.
```

Getting started

Install the package

Install the Microsoft Azure Storage Actions management library for .NET with NuGet:

.NET CLI

```
dotnet add package Azure.ResourceManager.StorageActions --prerelease
```

Prerequisites

- You must have an Microsoft Azure subscription.

Authenticate the Client

To create an authenticated client and start interacting with Microsoft Azure resources, see the quickstart guide here.

Key concepts

Key concepts of the Microsoft Azure SDK for .NET can be found here

Documentation

Documentation is available to help you learn how to use this package:

- Quickstart.
- API References.
- Authentication.

Examples

Code samples for using the management library for .NET can be found in the following locations

- .NET Management Library Code Samples

Troubleshooting

- File an issue via GitHub Issues.
- Check previous questions or ask new ones on Stack Overflow using Azure and .NET tags.

Next steps

For more information about Microsoft Azure SDK, see this website.

Contributing

For details on contributing to this repository, see the contributing guide.

This project welcomes contributions and suggestions. Most contributions require you to agree to a Contributor License Agreement (CLA) declaring that you have the right to, and actually do, grant us the rights to use your contribution. For details, visit https://cla.microsoft.com.

When you submit a pull request, a CLA-bot will automatically determine whether you need to provide a CLA and decorate the PR appropriately (for example, label, comment). Follow the instructions provided by the bot. You'll only need to do this action once across all repositories using our CLA.

This project has adopted the Microsoft Open Source Code of Conduct. For more information, see the Code of Conduct FAQ or contact opencode@microsoft.com with any other questions or comments.

com.azure.resourcemanager.storageactions

Package: com.azure.resourcemanager.storageactions

Maven Artifact: com.azure.resourcemanager:azure-resourcemanager-storageactions:1.0.0-beta.3

Package containing the classes for StorageActionsMgmtClient. The Azure Storage Actions Management API.

Classes

Expand table

StorageActionsManager	Entry point to StorageActionsManager.
StorageActionsManager.Configurable	The Configurable allowing configurations to be set.

storageactions Package

Packages

Expand table

aio
models
operations

Classes

Expand table

StorageActionsMgmtClient	The Azure Storage Actions Management API.

StorageActionsManagementClient class

Package: @azure/arm-storageactions

Extends ServiceClient

Constructors

Expand table

StorageActionsManagementClient(TokenCredential, string, StorageActionsManagementClientOptionalParams)	Initializes a new instance of the StorageActionsManagementClient class.

Properties

Expand table

$host
apiVersion
operations
storageTaskAssignmentOperations
storageTasks
storageTasksReport
subscriptionId

Inherited Properties

Expand table

pipeline	The pipeline used by this client to make requests

Inherited Methods

Expand table

sendOperationRequest<T>(OperationArguments, OperationSpec)	Send an HTTP request that is populated using the provided OperationSpec.
sendRequest(PipelineRequest)	Send the provided httpRequest.

Constructor Details

StorageActionsManagementClient(TokenCredential, string, StorageActionsManagementClientOptionalParams)

Initializes a new instance of the StorageActionsManagementClient class.

TypeScript

```
new StorageActionsManagementClient(credentials: TokenCredential,
subscriptionId: string, options?: StorageActionsManagementClientOptionalParams)
```

Parameters

`credentials` TokenCredential

Subscription credentials which uniquely identify client subscription.

`subscriptionId` string

The ID of the target subscription. The value must be an UUID.

`options` StorageActionsManagementClientOptionalParams

The parameter options

Property Details

$host

TypeScript

```
$host: string
```

Property Value

string

apiVersion

```
TypeScript

apiVersion: string
```

Property Value

string

operations

```
TypeScript

operations: Operations
```

Property Value

Operations

storageTaskAssignmentOperations

```
TypeScript

storageTaskAssignmentOperations: StorageTaskAssignmentOperations
```

Property Value

StorageTaskAssignmentOperations

storageTasks

```
TypeScript

storageTasks: StorageTasks
```

Property Value

StorageTasks

storageTasksReport

```
TypeScript

storageTasksReport: StorageTasksReport
```

Property Value

StorageTasksReport

subscriptionId

```
TypeScript

subscriptionId: string
```

Property Value

string

Inherited Property Details

pipeline

The pipeline used by this client to make requests

```
TypeScript

pipeline: Pipeline
```

Property Value

Pipeline

Inherited From coreClient.ServiceClient.pipeline

Inherited Method Details

sendOperationRequest<T>(OperationArguments, OperationSpec)

Send an HTTP request that is populated using the provided OperationSpec.

TypeScript

```
function sendOperationRequest<T>(operationArguments: OperationArguments,
operationSpec: OperationSpec): Promise<T>
```

Parameters

operationArguments OperationArguments

The arguments that the HTTP request's templated values will be populated from.

operationSpec OperationSpec

The OperationSpec to use to populate the httpRequest.

Returns

Promise<T>

Inherited From coreClient.ServiceClient.sendOperationRequest

sendRequest(PipelineRequest)

Send the provided httpRequest.

TypeScript

```
function sendRequest(request: PipelineRequest): Promise<PipelineResponse>
```

Parameters

request PipelineRequest

Returns

Promise<PipelineResponse>

Inherited From coreClient.ServiceClient.sendRequest

Az.StorageAction Module

Microsoft Azure PowerShell: StorageAction cmdlets

StorageAction

Expand table

Cmdlet	Description
Get-AzStorageActionTask	Get the storage task properties
Get-AzStorageActionTaskAssignment	Lists Resource IDs of the Storage Task Assignments associated with this Storage Task.
Get-AzStorageActionTasksReport	Fetch the storage tasks run report summary for each assignment.
Invoke-AzStorageActionTaskPreviewAction	Runs the input conditions against input object metadata properties and designates matched objects in response.
New-AzStorageActionTask	Asynchronously create a new storage task resource with the specified parameters. If a storage task is already created and a subsequent create request is issued with different properties, the storage task properties will be updated. If a storage task is already created and a subsequent create request is issued with the exact same set of properties, the request will succeed.
New-AzStorageActionTaskOperationObject	Create an in-memory object for StorageTaskOperation.
New-AzStorageActionTaskPreviewBlobPropertiesObject	Create an in-memory object for StorageTaskPreviewBlobProperties.
New-AzStorageActionTaskPreviewKeyValuePropertiesObject	Create an in-memory object for StorageTaskPreviewKeyValueProperties.
Remove-AzStorageActionTask	Delete the storage task resource.
Update-AzStorageActionTask	Asynchronously update a new storage task resource with the specified parameters. If a storage task is already created and a subsequent update request is issued with different properties, the storage task properties will be updated. If a storage task is already created and

Cmdlet	Description
	a subsequent update request is issued with the exact same set of properties, the request will succeed.

az storage-actions

> **Note**
>
> This reference is part of the **storage-actions** extension for the Azure CLI (version 2.70.0 or higher). The extension will automatically install the first time you run an **az storage-actions** command. Learn more about extensions.

Manage StorageActions.

Commands

Expand table

Name	Description	Type	Status
az storage-actions task	Manage StorageTask.	Extension	GA
az storage-actions task create	Create a new storage task resource with the specified parameters. If a storage task is already created and a subsequent create request is issued with different properties, the storage task properties will be updated. If a storage task is already created and a subsequent create request is issued with the exact same set of properties, the request will succeed.	Extension	GA
az storage-actions task delete	Delete the storage task resource.	Extension	GA
az storage-actions task list	List all the storage tasks available under the subscription.	Extension	GA
az storage-actions task list-assignment	List all the storage task assignments available under the given resource group.	Extension	GA
az storage-actions task list-report	List the storage tasks run report summary for each assignment.	Extension	GA
az storage-actions task preview-action	Runs the input conditions against input object metadata properties and designates matched objects in response.	Extension	GA

Name	Description	Type	Status
az storage-actions task show	Get the storage task properties.	Extension	GA
az storage-actions task update	Update a storage task resource with the specified parameters. If a storage task is already created and a subsequent update request is issued with different properties, the storage task properties will be updated. If a storage task is already created and a subsequent update request is issued with the exact same set of properties, the request will succeed.	Extension	GA
az storage-actions task wait	Place the CLI in a waiting state until a condition is met.	Extension	GA

Azure Storage Actions monitoring data reference

Article • 05/05/2025

See Monitoring Azure Storage Actions for details on collecting and analyzing monitoring data for Azure Storage Actions.

Metrics

This section lists all the automatically collected platform metrics for Azure Storage Actions.

Expand table

Metric Type	Resource Provider / Type Namespace and link to individual metrics
Storage tasks	Microsoft.StorageActions/storageTasks
Storage tasks	Microsoft.StorageActions/storageAccounts/storageTasks

Metric dimensions

For more information on what metric dimensions are, see Multi-dimensional metrics.

Azure Storage Actions support the following dimensions for metrics in Azure Monitor.

Expand table

Dimension Name	Description
AccountName	The name of a storage account
TaskAssignmentId	The ID of the task assignment

Activity log

The following table lists the operations that Azure Storage Actions might record in the activity log. This list of entries is a subset of the possible entries that you might find in the activity log.

Expand table

Namespace	Description
Microsoft.StorageActions/storageTasks/read	Reads an existing storage task.
Microsoft.StorageActions/storageTasks/delete	Deletes a storage task.
Microsoft.StorageActions/storageTasks/promote/action	Promotes specific version of storage task to current version.
Microsoft.StorageActions/storageTasks/write	Edits a storage task.
Microsoft.StorageActions/storageAccounts/storageTasks/delete	Deletes a storage task.
Microsoft.StorageActions/storageAccounts/storageTasks/read	Reads an existing storage task.
Microsoft.StorageActions/storageAccounts/storageTasks/executionsummary/action	Opens task runs.
Microsoft.StorageActions/storageAccounts/storageTasks/assignmentexecutionsummary/action	Opens task runs from the Assignments pane.
Microsoft.StorageActions/storageAccounts/storageTasks/write	Edits a storage task.

See all the possible resource provider operations in the activity log.

For more information on the schema of Activity Log entries, see Activity Log schema.

See Also

- See Monitoring Azure Storage Actions for a description of monitoring Azure Storage Actions.
- See Monitoring Azure resources with Azure Monitor for details on monitoring Azure resources.

Microsoft.StorageActions resource types

Article • 12/09/2024

This article lists the available versions for each resource type.

For a list of changes in each API version, see change log

Resource types and versions

Expand table

Types	Versions
Microsoft.StorageActions/storageTasks	2023-01-01

Feedback

Was this page helpful? Yes No

Microsoft.Storage storageAccounts/storageTaskAssignments

09/16/2025

Bicep resource definition

The storageAccounts/storageTaskAssignments resource type can be deployed with operations that target:

For a list of changed properties in each API version, see change log.

Resource format

To create a Microsoft.Storage/storageAccounts/storageTaskAssignments resource, add the following Bicep to your template.

Bicep

```
resource symbolicname
'Microsoft.Storage/storageAccounts/storageTaskAssignments@2025-01-01' = {
  parent: resourceSymbolicName
  name: 'string'
  properties: {
    description: 'string'
    enabled: bool
    executionContext: {
      target: {
        excludePrefix: [
          'string'
        ]
        prefix: [
          'string'
        ]
      }
      trigger: {
        parameters: {
          endBy: 'string'
          interval: int
          intervalUnit: 'string'
          startFrom: 'string'
          startOn: 'string'
        }
        type: 'string'
      }
    }
    report: {
```

```
      prefix: 'string'
    }
    runStatus: {}
    taskId: 'string'
  }
}
```

Property Values

Microsoft.Storage/storageAccounts/storageTaskAssignments

Expand table

Name	Description	Value
name	The resource name	string Constraints: Min length = 3 Max length = 24 Pattern = `^[a-z0-9]{3,24}$` (required)
parent	In Bicep, you can specify the parent resource for a child resource. You only need to add this property when the child resource is declared outside of the parent resource. For more information, see Child resource outside parent resource.	Symbolic name for resource of type: storageAccounts
properties	Properties of the storage task assignment.	StorageTaskAssignmentProperties (required)

ExecutionTarget

Expand table

Name	Description	Value
excludePrefix	List of object prefixes to be excluded from task execution. If there is a conflict between include and exclude prefixes, the exclude prefix will be the determining factor	string[]
prefix	Required list of object prefixes to be included for task execution	string[]

ExecutionTrigger

Expand table

Name	Description	Value
parameters	The trigger parameters of the storage task assignment execution	TriggerParameters (required)
type	The trigger type of the storage task assignment execution	'OnSchedule' 'RunOnce' (required)

StorageTaskAssignmentExecutionContext

Expand table

Name	Description	Value
target	Execution target of the storage task assignment	ExecutionTarget
trigger	Execution trigger of the storage task assignment	ExecutionTrigger (required)

StorageTaskAssignmentProperties

Expand table

Name	Description	Value
description	Text that describes the purpose of the storage task assignment	string (required)
enabled	Whether the storage task assignment is enabled or not	bool (required)
executionContext	The storage task assignment execution context	StorageTaskAssignmentExecutionContext (required)
report	The storage task assignment report	StorageTaskAssignmentReport (required)
runStatus	Run status of storage task assignment	StorageTaskReportProperties
taskId	Id of the corresponding storage task	string (required)

StorageTaskAssignmentReport

Expand table

Name	Description	Value
prefix	The container prefix for the location of storage task assignment report	string (required)

StorageTaskReportProperties

Expand table

Name	Description	Value

TriggerParameters

Expand table

Name	Description	Value
endBy	When to end task execution. This is a required field when ExecutionTrigger.properties.type is 'OnSchedule'; this property should not be present when ExecutionTrigger.properties.type is 'RunOnce'	string
interval	Run interval of task execution. This is a required field when ExecutionTrigger.properties.type is 'OnSchedule'; this property should not be present when ExecutionTrigger.properties.type is 'RunOnce'	int Constraints: Min value = 1
intervalUnit	Run interval unit of task execution. This is a required field when ExecutionTrigger.properties.type is 'OnSchedule'; this property should not be present when ExecutionTrigger.properties.type is 'RunOnce'	'Days'
startFrom	When to start task execution. This is a required field when ExecutionTrigger.properties.type is 'OnSchedule'; this property should not be present when ExecutionTrigger.properties.type is 'RunOnce'	string
startOn	When to start task execution. This is a required field when ExecutionTrigger.properties.type is 'RunOnce'; this property should not be present when ExecutionTrigger.properties.type is 'OnSchedule'	string

Known issues and limitations with storage tasks

06/30/2025

This article describes limitations and known issues of storage tasks. The issues that appear in this article reflect the current state of the service. This list will change over time as support continues to expand.

Scale limits

Expand table

Scale factor	Supported limit
Storage tasks per subscription	5,000
Storage task assignments per storage task	5,000
Storage task assignments per subscription	10,000
Storage task assignments per storage account	50
Storage task nested grouping of clauses per condition	10

Azure Storage Actions autoscales its processing tasks based on the volume of data in a storage account, subject to internal limits. The duration of execution depends on the number of blobs in the storage account, as well as their hierarchy in Azure Data Lake Storage Gen2. The first execution of a task over a path prefix might take longer than subsequent executions. Azure Storage Actions are also designed to be self-regulating and to allow application workloads on the storage account to take precedence. As a result, the scale and the duration of execution also depend on the available transaction capacity given the storage account's maximum request limit. The following are typical processing scales, which might be higher if you have more transaction capacity available, or might be lower for lesser spare transaction capacity on the storage account.

Task assignments applied on storage accounts across regions

Task assignments can only be applied on storage accounts that are in the same region as the storage tasks.

Billing doesn't show task assignment name

Billing meters show up on the bill with only the storage account name. Subscription bill doesn't show the task assignment name for which the meter was emitted. To correlate the meter with the task assignment, you must look at the resource metrics for Storage Actions filtered by the storage account for that day.

Propagation of task definition updates

Task assignments aren't updated when changes are made to a task definition. New task assignments must be created after deleting the older ones to pick up any changes.

Stopping task assignments

You can stop an in-progress run by removing the role assignment for the underlying managed identity.

Move for storage account resource is blocked when a task assignment exists

The workaround is to delete the storage task assignment and then move the storage account resource.

Restrictions on moving a storage task

You can't move a storage task to another region or to another subscription. You can't move a subscription that contains a storage task to another tenant.

Concurrency limit for execution

Storage tasks have a limit on the number of task assignments that can be executed concurrently on each storage account. To ensure optimal performance, make sure that task assignments on a single storage account are scheduled to run with a reasonable time interval between them based on the objects targeted, to ensure task runs complete in time. Task assignment executions exceeding the concurrency limit for a storage account are paused until other assignments have completed.

Scale dependence on transaction capacity available for the storage account

Storage task assignment execution is autoscaled depending on the transaction request capacity available on the storage account. Scale is higher when more transaction capacity is available and lower when less transaction capacity is available.

When the targeted storage account has lower available transaction capacity, storage task execution might be throttled resulting in longer than expected duration for completing the task assignment execution.

For more information about scale limits, see Scalability and performance targets for Blob storage.

> **Note**
>
> You can request higher capacity and ingress limits. To request an increase, contact Azure Support.

Storage task runs can write to the report export container without permission to the container

As you create a task assignment, you'll assign a role to the system-assigned managed identity of the storage task. When the storage task runs, it can operate only on containers where it's managed identity is assigned the required role. This isn't the case with the report export container that you choose during task assignment. While a storage task can't operate on existing blobs in that container, a task doesn't require the correct role to write reports to that container.

String operators on container metadata, blob metadata, and blob index tags don't work if the values are numbers

You can't use string operators on container metadata, blob metadata, and blob index tags along with numbers as value. For example, equals(Tags.Value[Year], '2022') where the value '2022' is a number, along with string operator equals, doesn't evaluate correctly.

Assignments fail when they reference a storage account name that starts with a digit

If you assign a storage task to a storage account that has a name, which starts with a digit, the storage task assignment fails.

Whitespace characters in Blob index tags and metadata aren't yet supported

Whitespace characters in the key and value of blob tags are acceptable inputs. However, storage task conditions are unable to process the whitespace characters. If a key or value contains a whitespace character, an error appears when the task runs.

Blob name property value contains or matches "." is unsupported

The string field input on blob name clause accepts ".doc" or ".pdf" as inputs but fails to deploy the task resource. The service resource provider validation catches it and throws the error. The value of the property 'Name' is '.doc' and it doesn't follow the pattern '^[a-zA-Z0-9]+$'"}]}.

Storage task assignments operate on an incomplete list of blobs when used with multiple directory filters in accounts that have a hierarchical namespace

If multiple filters are used in storage task assignments, not all directory prefixes are scanned for blobs to be operated on.

Using whitespace characters in the path prefix during task assignment isn't supported

Storage accounts that have a hierarchical namespace display location information as `container1 / subcontainer1` with a whitespace character between the string and the `/` character. An error appears if you copy and paste this information into the path prefix field during assignment.

Moving storage tasks and task assignments

Moving storage tasks and task assignments across different resource groups and subscriptions isn't supported. This limitation means that any storage tasks and their associated task assignments can't be transferred between resource groups or subscriptions.

Cleaning up task assignments before deleting storage accounts or storage tasks

- Before deleting a storage account, delete all task assignments associated with that storage account.
- Before deleting a storage task, delete all task assignments referencing that storage task.

Operating on storage accounts in a private network is unsupported in PREVIEW regions

When applying storage task assignments to storage accounts with IP or network access restrictions, task execution may fail. This occurs because the tasks require access via the public endpoint, which can be blocked by firewall or virtual network rules. To prevent this, ensure your storage account is configured to allow appropriate network access.

Storage task runs are stuck in the in progress state

If during the assignment process, you assign a role which doesn't have the required permission, the storage task run will take 14 days to fail. To unblock the task run, you can add the required role to the managed identity of the storage task. Otherwise, the task assignment will be stuck in the **in progress** state until the task run ends in 14 days.

Premium Block Blobs

Creating assignments on premium block blobs storage accounts doesn't work.

Soft deleted blobs are included in listing during scanning as objects targeted

The workaround is to exclude the specific prefixes which are soft deleted.

No option to choose priority when rehydrating blobs to an online tier

When rehydrating archived blobs, there's no option to choose a priority. The blobs are rehydrated with the standard priority.

See Also

- Azure Storage Actions overview

Storage task troubleshooting

Article • 05/07/2025

This article describes issues that you might encounter when you use storage tasks.

Permission-related errors

Permission-related errors occur when the necessary permissions aren't granted to create the task or task assignment resource, or for the task assignment's managed identity to perform the defined operations. It's essential to review and update the permissions to ensure that the tasks have the required permissions to execute successfully. See Azure roles for storage task and Azure roles required to assign tasks.

Scale limits exceeded errors

Scale limits exceeded errors arise when the task assignments and task execution operations exceed the predefined scale limits. It's important to monitor and adjust the scope of the task assignment accordingly to prevent such errors. See Scale limits.

Internal errors

These are unforeseen errors that occur within the system. Diagnosing internal errors often requires a deeper dive into the system logs and might necessitate contacting support for resolution.

See also

- Storage task operations
- Define conditions and operations